Н.В. Максимов, И.И. Попов, Т.Л. Партыка

Архитектура ЭВМ и вычислительные системы

Н. В. Максимов, Т. Л. Партыка,
И. И. Попов

АРХИТЕКТУРА ЭВМ И ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ

5-е издание, переработанное и дополненное

Рекомендовано Министерством образования и науки Российской Федерации в качестве учебника для студентов учреждений среднего профессионального образования, обучающихся по группе специальностей «Информатика и вычислительная техника»

УДК 004.2(075.32)
ББК 32.973-02я723
М17

Рецензенты:
Бачинин Ю. Г. — кандидат технических наук,
доцент кафедры «Проектирование АИС» РЭА им. Г.В. Плеханова;
Емельянов А. А. — доктор экономических наук, профессор,
декан факультета «Информатика» в НОУ «Московский международный институт эконометрики, информатики, финансов и права»
(ММИЭИФП)

Максимов Н. В., Партыка Т. Л., Попов И. И.
М17 Архитектура ЭВМ и вычислительных систем : учебник / Н. В. Максимов, Т. Л. Партыка, И. И. Попов. — 5-е изд., перераб. и доп. — М. : ФОРУМ : ИНФРА-М, 2013. — 512 с. : ил. — (Профессиональное образование).

ISBN 978-5-91134-742-0 (ФОРУМ)
ISBN 978-5-16-006732-2 (ИНФРА-М)

Рассмотрены вопросы организации и функционирования вычислительных устройств, машин и систем. Описаны логические, информационные, алгоритмико-вычислительные основы построения систем. Значительное внимание уделено архитектурам вычислительных машин и систем, их классификациям, составным компонентам — информационно-вычислительным средам и коммутационно-коммуникационным средам. В качестве примера подробно представлены технические, структурные, архитектурные компоненты персональных машин и средства их комплексирования.

Для студентов учреждений среднего профессионального образования, обучающихся по группе специальностей «Информатика и вычислительная техника».

УДК 004.2(075.32)
ББК 32.973-02я723

ISBN 978-5-91134-742-0 (ФОРУМ)
ISBN 978-5-16-006732-2 (ИНФРА-М)

Введение

Человечество прошло длинный путь, прежде чем достигло современного уровня развития и применения компьютеров и информационных технологий.

Вплоть до XVII в. деятельность общества в целом и каждого человека в отдельности была направлена на овладение веществом, т. е. есть познание свойств вещества и изготовление сначала примитивных, а потом все более сложных орудий труда вплоть до механизмов и машин, позволяющих изготовлять материальные ценности.

Затем в процессе становления индустриального общества на первый план вышла проблема овладения энергией — сначала тепловой, затем электрической и атомной, что позволило освоить массовое производство потребительских ценностей и, как следствие, повысить уровень жизни людей и изменить характер их труда.

В то же время человечеству свойственна потребность выражать, запоминать и распространять информацию об окружающем мире — эти устремления привели к появлению письменности, книгопечатания, живописи, фотографии, радиосвязи, телевидения.

В далеком прошлом люди считали на пальцах или делали насечки на костях. Древнейшим «счетным инструментом», который был представлен самой природой в распоряжение человека, была его собственная рука. На заре человеческой цивилизации были изобретены различные системы счисления, позволяющие осуществлять торговые сделки, рассчитывать астрономические циклы, проводить другие вычисления. Спустя несколько тысячелетий появились первые ручные вычислительные средства.

В наши дни сложнейшие вычислительные задачи, как и множество других операций, казалось бы, не связанных с числами,

решаются именно с помощью компьютеров. Это еще один тип машин, построенный для того, чтобы увеличить эффективность и качество выполняемых работ и повысить производительность труда. Однако ни одна другая машина в истории не принесла столь быстрых и глубоких изменений (посадка аппаратов на поверхность Луны и исследование планет Солнечной системы, управление медицинской аппаратурой, решение сложных экономических и управленческих задач и многое другое).

Электронные вычислительные машины (ЭВМ) за последние полвека преобразили цивилизацию, вовлекая человечество в процесс информатизации, который охватывает все сферы, все отрасли общественной жизни, прочно входят в жизнь каждого человека, воздействуют на его образ мышления и поведение.

ЭВМ являются важнейшим из факторов информатизации, которые включают в себя:

- технические средства (ЭВМ и системы);
- программные средства и системы (программный фактор);
- информационный фактор — собственно информация, т. е. сигналы, сообщения, массивы данных, файлы и базы данных (БД);
- интеллектуальные усилия и человеческий труд (человеческий, гуманитарный фактор).

Перечисленные факторы соответствуют также историческим этапам развития информатизации. Можно выделить следующие фазы, на каждой из которых доминирует какой-либо из упомянутых факторов:

- *технический период*, за который сложились основные представления о структуре универсальных электронных вычислительных машин (ЭВМ), определилась архитектура и типы устройств — с 1946 по 1964 гг. (приблизительно);
- *программный период* — выработалась современная классификация программных средств, их структур и взаимосвязей, сложились языки программирования, были разработаны первые компиляторы и принципы процедурной обработки — с 1954 по 1970 гг.;
- *информационный период* — в центре внимания исследователей и разработчиков оказываются структуры данных, языки описания (ЯОД) и манипулирования (ЯМД) данными, непроцедурные подходы к построению систем обработки информации — с 1970 г. по настоящее время;

- *гуманитарный период*, связанный с резким возрастанием круга пользователей автоматизированных информационных технологий (АИТ) и повышением роли интерфейсных и навигационных возможностей соответствующих систем (с начала 90-х гг. прошлого века). Традиционные АИТ были подчинены производителям информации (информационных систем) и доводили одинаковое содержание (создавали однотипные возможности) до всех адресатов. Новые АИТ направлены на индивидуального пользователя, предоставляя возможность получения информации, нужной именно ему.

Конечно, данная периодизация условна, и говорить об окончании технического периода или исчерпании пределов развития не приходится. Именно ЭВМ развиваются наиболее высокими темпами, увлекая за собой остальные факторы информатизации.

Настоящий учебник посвящается данной проблематике.

Первая глава посвящена истории развития вычислительных устройств, машин и приборов, основным аспектам основ ЭВМ — информационным (представление, кодирование, обработка информации в ЭВМ), логическим (основные функции, операции, элементы и узлы) и алгоритмическим (структура и основные элементы программ и алгоритмов).

В **главе 2** рассматриваются различные принципы классификации вычислительных машин, узлы ЭВМ, представления об архитектуре ЭВМ, вычислительных систем (ВС) и суперкомпьютеров.

В **третьей главе** рассматриваются общие принципы структуры и архитектуры процессоров, технологии повышения их производительности, конкретные микроархитектуры процессоров Intel, AMD, IBM, система команд x86 в представлениях макроассемблера.

Глава 4 посвящена вопросам организации оперативной памяти, внутренним интерфейсам и (кратко) внешним интерфейсам, а также разновидностям архитектуры набора микросхем системной платы (чипсет).

К сожалению, из-за ограничений на объем книги был опущен целый ряд вопросов, связанных с современным состоянием периферийных устройств ЭВМ, которые однако подробно рассмотрены, например, в работах авторов [10, 11, 15].

Учебник базируется на материалах, накопленных авторами в процессе практической, исследовательской, а также преподавательской деятельности (МИФИ, МИСИ, РГГУ, РЭА

им. Г.В. Плеханова, МФПА). Авторы выражают благодарность рецензентам, а также коллегам, принявшим участие в обсуждении материала, — Н.В. Максимову (МИФИ), К.И. Курбакову (РЭА им. Г.В. Плеханова), а также студентам РГГУ, МФПА, РЭА им. Г.В. Плеханова за предоставленные иллюстративные материалы.

Глава 1
ВЫЧИСЛИТЕЛЬНЫЕ УСТРОЙСТВА И МАШИНЫ. ОСНОВНЫЕ ПРИНЦИПЫ

Закладка фундамента компьютерной революции происходила медленно и далеко не гладко. Отправной точкой этого процесса можно считать изобретение счетов (более 1500 лет назад). Они оказались очень эффективным инструментом и вскоре распространились по всему миру (кое-где применяясь и по сей день). С XVII в. европейские мыслители были увлечены идеей создания счетных устройств.

Первая счетная машина появилась лишь в 1642 г. Ее изобрел французский математик Паскаль (см. табл. 1.1). Построенная на основе зубчатых колес, она могла суммировать десятичные числа. Все четыре арифметических действия выполняла машина, созданная в 1673 г. немецким математиком Лейбницем, которая стала прототипом арифмометров, использовавшихся с 1820 г. вплоть до 1960 гг. XX в. Впервые идея программно-управляемой счетной машины, имеющей арифметическое устройство, устройства управления, ввода и печати (хотя и использующей десятичную систему счисления), была выдвинута в 1822 г. английским математиком Бэббиджем. Проект опережал технические возможности своего времени и не был реализован.

Каждому, знакомому с современными компьютерами, механические счетные машины и приборы покажутся, пожалуй, забавными и неуклюжими устройствами. Однако, ознакомившись с историей развития счетных машин, можно поразиться изобретательности, хитроумию и настойчивости их создателей. Уместно вспомнить слова Б. Паскаля о том, что для создания «арифметической машины» ему потребовалось все ранее приобретенные знания по геометрии, физике и механике.

Выполнявшиеся в 40-х годах XX в. работы по созданию вычислительных машин с программным управлением, были тесно связаны с появлением новой фундаментальной науки — кибер-

нетики, или науки об управлении и коммуникации. Судьба этой науки в нашей стране (бывшем СССР) была трудной. Долгое время она считалась «буржуазной лженаукой». Только в начале 50-х гг. прошлого века появились первые советские вычислительные машины. Несмотря на это роль отечественных ученых в области кибернетики и вычислительной техники неоценима.

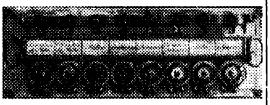
1.1. Вычислительные устройства и приборы, история вопроса

Рассмотрим вначале табл. 1.1, в которой приведены краткие сведения об истории развития вычислительных средств и методов в лицах, событиях и объектах («время — события — люди») [10—12].

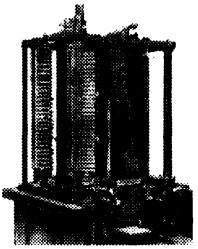
Таблица 1.1. **Основные события в истории развития вычислительных методов, приборов, автоматов и машин**

Авторы и изделия	Хронология и описание (комментарий)
 Джон Непер (1550—1617)	Шотландец Джон Непер в 1614 г. опубликовал «Описание удивительных таблиц логарифмов». Он установил, что сумма логарифмов чисел *a* и *b* равна логарифму произведения этих чисел. Поэтому умножение удается свести к операции сложения. Им же был разработан инструмент перемножения чисел — «костяшки Непера», состоявший из набора сегментированных стерженьков, при перемещении которых при сложении чисел в прилегающих друг к другу по горизонтали сегментах получали результат их умножения. «Костяшки Непера» вскоре были вытеснены логарифмической линейкой (Р. Биссакар, конец 1620 г.) и другими вычислительными устройствами (в основном механического типа)
 Вильгельм Шиккард (1592—1636)	Считалось, что первую механическую счетную машину изобрел великий французский математик и физик Б. Паскаль в 1642 г. Однако в 1957 г. Ф. Гаммер (ФРГ, директор Кеплеровского научного центра) обнаружил доказательства создания Вильгельмом Шиккардом механической вычислительной машины приблизительно за два десятилетия до изобретения Паскаля. Он назвал ее «часы для счета». Машина предназначалась для выполнения четырех арифметических действий и состояла из частей: суммирующего устройства; множительного устройства; механизма для промежуточных результатов. Суммирующее устройство состояло из зубчатых передач и представляло простейшую форму арифмометра. Предложенная схема механического счета считается классической

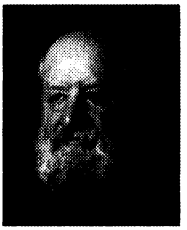
Продолжение табл. 1.1

Авторы и изделия	Хронология и описание (комментарий)
	Однако эту простую и эффективную схему пришлось изобретать заново, так как сведения о машине Шиккарда не стали всеобщим достоянием
 Блэз Паскаль (1623 —1662)  Машина Б. Паскаля	В 1642 г., когда Паскалю было 19 лет, была изготовлена первая действующая модель суммирующей машины. Через несколько лет Блэз Паскаль создал механическую суммирующую машину («паскалина»), которая позволяла складывать числа в десятичной системе счисления. В этой машине цифры шестизначного числа задавались путем соответствующих поворотов дисков (колес) с цифровыми делениями, а результат операции можно было прочитать в шести окошках — по одному на каждую цифру. Диск единиц был связан с диском десятков, диск десятков — с диском сотен и т. д. Другие операции выполнялись с помощью довольно неудобной процедуры повторных сложений, и в этом заключался основной недостаток «паскалины». Приблизительно за десятилетие он построил более 50 различных вариантов машины. Изобретенный Паскалем принцип связанных колес явился основой, на которой строилось большинство вычислительных устройств на протяжении следующих трех столетий
 Готфрид Вильгельм Лейбниц (1646 —1716)	В 1672 г., находясь в Париже, Лейбниц познакомился с голландским математиком и астрономом Христианом Гюйгенсом. Видя, как много вычислений приходится делать астроному, Лейбниц решил изобрести механическое устройство для расчетов. В 1673 г. он завершил создание механического калькулятора. Развив идеи Паскаля, Лейбниц использовал операцию сдвига для поразрядного умножения чисел. Сложение производилось на нем по существу так же, как и на «паскалине», однако Лейбниц включил в конструкцию движущуюся часть (прообраз подвижной каретки будущих настольных калькуляторов) и ручку, с помощью которой можно было крутить ступенчатое колесо или — в последующих вариантах машины — цилиндры, расположенные внутри аппарата
 Жозеф-Мари Жаккар (1775—1834)	Развитие вычислительных устройств связано с появлением перфорационных карт и их применением. Появление же перфорационных карт связано с ткацким производством. В 1804 г. инженер Жозеф-Мари Жаккар построил полностью автоматизированный станок (станок Жаккара), способный воспроизводить сложнейшие узоры. Работа станка программировалась с помощью колоды перфокарт, каждая из которых управляла одним ходом челнока. Переход к новому рисунку происходил заменой колоды перфокарт

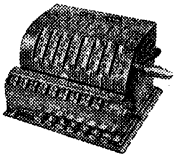
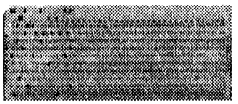
Продолжение табл. 1.1

Авторы и изделия	Хронология и описание (комментарий)
 Чарльз Бэббидж (1791—1871)  Аналитическая машина Ч. Бэббиджа	Он обнаружил погрешности в таблицах логарифмов Непера, которыми широко пользовались при вычислениях астрономы, математики, штурманы дальнего плавания. В 1821 г. приступил к разработке своей вычислительной машины, которая помогла бы выполнить более точные вычисления. В 1822 г. была построена *разностная машина* (пробная модель), способная рассчитывать и печатать большие математические таблицы. Это было очень сложное, большое устройство и предназначалось для автоматического вычисления логарифмов. Работа модели основывалась на принципе, известном в математике как «метод конечных разностей»: при вычислении многочленов используется только операция сложения и не выполняется умножение и деление, которые значительно труднее поддаются автоматизации. В последующем он пришел к идее создания более мощной — *аналитической машины*. Она не просто должна была решать математические задачи определенного типа, а выполнять разнообразные вычислительные операции в соответствии с инструкциями, задаваемыми оператором. По замыслу это был первый универсальный программируемый компьютер. Аналитическая машина в своем составе должна была иметь такие компоненты, как «мельница» (арифметическое устройство по современной терминологии) и «склад» (память). Инструкции (команды) вводились в аналитическую машину с помощью перфокарт (использовалась идея программного управления Жаккара с помощью перфокарт). Шведский издатель, изобретатель и переводчик Пер Георг Шойц, воспользовавшись советами Бэббиджа, построил видоизмененный вариант этой машины. В 1855 г. машина Шойца была удостоена золотой медали на Всемирной выставке в Париже. В дальнейшем один из принципов, лежащих в основе идеи аналитической машины, — использование перфокарт — нашел воплощение в статистическом табуляторе, построенном американцем Германом Холлеритом (для ускорения обработки результатов переписи населения в США в 1890 г.)
 Огаста Ада Байрон (графиня Лавлейс) (1815—1852)	Графиня Огаста Ада Лавлейс, дочь поэта Байрона, совместно с Ч. Бэббиджем работала над созданием программ для его счетных машин. Ее работы в этой области были опубликованы в 1843 г. Однако в то время считалось неприличным для женщины издавать свои сочинения под полным именем, и Лавлейс поставила на титуле только свои инициалы. В материалах Бэббиджа и комментариях Лавлейс намечены такие понятия, как «подпрограмма» и «библиотека подпрограмм», «модификация команд» и «индексный регистр», которые стали употребляться только в 50-х гг. XX в. Сам термин «библиотека» был введен Бэббиджем, а термины «рабочая ячейка» и «цикл» предложила А. Лавлейс. «Можно с полным

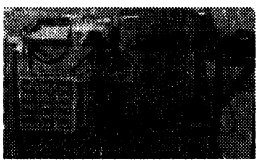
Продолжение табл. 1.1

Авторы и изделия	Хронология и описание (комментарий)
	основанием сказать, что аналитическая машина точно так же плетет алгебраические узоры, как ткацкий станок Жаккара воспроизводит цветы и листья», — писала графиня Лавлейс. Она фактически была первой программисткой (в ее честь был назван язык программирования Ада)
 Джордж Буль (1815—1864)	В 1847 г. Буль написал статью «Математический анализ логики», а в 1854 г. развил свои идеи в работе под названием «Исследование законов мышления». Эти труды внесли революционные изменения в логику как науку. Дж. Буль изобрел своеобразную *алгебру* — систему обозначений и правил, применяемую ко всевозможным объектам — от чисел и букв до предложений (в дальнейшем, это раздел математической логики — *булева алгебра*). Пользуясь этой системой, Буль мог закодировать высказывания (утверждения) с помощью своего языка, а затем манипулировать ими подобно тому, как в математике манипулируют обычными числами. Три основные операции системы — это И, ИЛИ и НЕ
 Пафнутий Львович Чебышев (1821—1894)	Им была разработана теория машин и механизмов, написан ряд работ, посвященных синтезу шарнирных механизмов. Среди многочисленных изобретенных им механизмов имеется несколько моделей арифмометров, первая из которых была сконструирована не позднее 1876 г. Арифмометр Чебышева для того времени был одной из самых оригинальных вычислительных машин. В своих конструкциях Чебышев предложил принцип непрерывной передачи десятков и автоматический переход каретки с разряда на разряд при умножении. Оба эти изобретения вошли в широкую практику в 30-е гг. XX в. в связи с применением электропривода и распространением полуавтоматических и автоматических клавишных вычислительных машин. С появлением этих и других изобретений стало возможно значительно увеличить скорость работы механических счетных устройств
 Алексей Николаевич Крылов (1863—1945)	Русский кораблестроитель, механик, математик, академик АН СССР. В 1904 г. он предложил конструкцию машины для интегрирования обыкновенных дифференциальных уравнений. В 1912 г. такая машина была построена. Это была первая *интегрирующая машина* непрерывного действия, позволяющая решать дифференциальные уравнения до четвертого порядка

Продолжение табл. 1.1

Авторы и изделия	Хронология и описание (комментарий)
 Вильгодт Теофил Однер (1845—1905)  Арифмометр Однера	Выходец из Швеции Вильгодт Теофил Однер в 1869 г. приехал в Петербург. Некоторое время он работал на заводе «Русский дизель» на Выборгской стороне, на котором в 1874 г. был изготовлен первый образец его арифмометра. Созданные на базе ступенчатых валиков Лейбница, первые серийные арифмометры имели большие размеры в первую очередь потому, что на каждый разряд нужно было выделять отдельный валик. Однер вместо ступенчатых валиков применил более совершенные и компактные зубчатые колеса с меняющимся числом зубцов — (колеса Однера). В 1890 г. Однер получает патент на выпуск арифмометров и в этом же году было продано 500 арифмометров (очень большое количество по тем временам). Арифмометры в России назывались «Арифмометр Однера», «Оригинал-Однер», «Арифмометр системы Однер» и др. В России до 1917 г. было выпущено примерно 23 тыс. арифмометров Однера. После революции производство арифмометров было налажено на Сущевском механическом заводе им. Ф. Э. Дзержинского в Москве. С 1931 г. они стали называться арифмометры «Феликс». Далее в нашей стране были созданы модели арифмометров с клавишным вводом и электроприводом
 Герман Холлерит (1860—1929)  Перфокарта Холлерита	После окончания Колумбийского университета поступает на работу в контору по переписи населения в Вашингтоне. В это время США приступили к исключительно трудоемкой (длившейся семь с половиной лет) ручной обработке данных, собранных в ходе переписи населения в 1880 г. К 1890 г. Холлерит завершил разработку *системы табуляции на базе применения перфокарт*. На каждой карте имелось 12 рядов, в каждом из которых можно было пробить по 20 отверстий, они соответствовали таким данным, как возраст, пол, место рождения, количество детей, семейное положение и прочим сведениям, включенным в вопросник переписи. Содержимое заполненных формуляров переносилось на карты путем соответствующего перфорирования. Перфокарты загружались в специальные устройства, соединенные с табуляционной машиной, где они нанизывались на ряды тонких игл, по одной игле на каждую из 240 перфорируемых позиций на карте. Когда игла попадала в отверстие, она замыкала контакт в соответствующей электрической цепи машины. Полный статистический анализ результатов занял два с половиной года (втрое быстрее по сравнению с предыдущей переписью). Впоследствии Холлерит организовал фирму «Computer Tabulating Recording» (CTR). Молодой коммивояжер этой компании Том Уотсон первым увидел потенциальную прибыльность продажи счетных машин на основе перфокарт американским бизнесменам. Позднее он возглавил компанию и в 1924 г. переименовал ее в IBM (International Business Machines)

Продолжение табл. 1.1

Авторы и изделия	Хронология и описание (комментарий)
 Ваннёвар Буш (1890—1974)	В 1930 г. построил механическое вычислительное устройство *дифференциальный анализатор*. Это была машина, на которой можно было решать сложные дифференциальные уравнения. Однако она обладала многими серьезными недостатками, прежде всего, гигантскими размерами. Механический анализатор Буша представлял собой сложную систему валиков, шестеренок и проволок, соединенных в серию больших блоков, которые занимали целую комнату. При постановке задачи машине оператор должен был вручную подбирать множество шестеренчатых передач. На это уходило обычно 2—3 дня. Позднее В. Буш предложил прототип современного гипертекста — проект MEMEX (MEMory EXtention — расширение памяти) как автоматизированное бюро, в котором человек хранил бы свои книги, записи, любую получаемую им информацию таким образом, чтобы в любой момент воспользоваться ею с максимальной быстротой и удобством. Фактически это должно было быть сложное устройство, снабженное клавиатурой и прозрачными экранами, на которые бы проецировались тексты и изображения, хранящиеся на микрофильмах. В MEMEX устанавливались бы логические и ассоциативные связи между любыми двумя блоками информации. В идеале речь идет о громадной библиотеке, универсальной информационной базе
 Атанасофф Джон Винсент (1903—1995)  Копия машины Atanasoff Berry Computer (ABC) в музее Университета шт. Айова	Профессор физики, автор первого проекта цифровой вычислительной машины на основе двоичной, а не десятичной системы счисления. Простота двоичной системы счисления в сочетании с простотой физического представления двух символов (0, 1) вместо десяти (0, 1, ..., 9) в электрических схемах компьютера перевешивала неудобства, связанные с необходимостью перевода из двоичной системы в десятичную и обратно. Кроме того, применение двоичной системы счисления способствовало уменьшению размеров вычислительной машины и снизила бы ее себестоимость. В 1939 г. Атанасофф совместно с Клиффордом Берри построил модель устройства и стал искать финансовую помощь для продолжения работы. Машина Атанасоффа была практически готова в декабре 1941 г., но находилась в разобранном виде. В связи с началом Второй мировой войны все работы по реализации этого проекта были приостановлены. Лишь в 1973 г. приоритет Атанасоффа как автора первого проекта такой архитектуры вычислительной машины был подтвержден решением федерального суда США

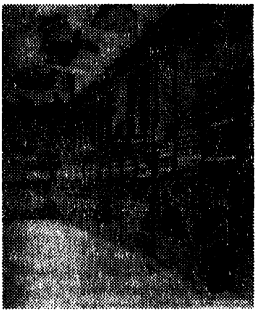
Продолжение табл. 1.1

Авторы и изделия	Хронология и описание (комментарий)
 Говард Эйкен  «Марк-1»	В 1937 г. Г. Эйкен выдвинул проект большой счетной машины и стал искать спонсоров, согласных ее профинансировать. Спонсором выступил Томас Уотсон, президент корпорации IBM; его вклад в проект составил около 500 тыс. долларов США. Проектирование новой машины «Марк-1», основанной на электромеханических реле, началось в 1939 г. в лабораториях Нью-йоркского филиала IBM и продолжалось до 1944 г. Готовый компьютер содержал около 750 тыс. деталей и весил 35 т. Машина оперировала двоичными числами до 23 разрядов и перемножала два числа максимальной разрядности примерно за 4 с. Поскольку создание «Марк-1» длилось достаточно долго, пальма первенства досталась не ему, а релейному двоичному компьютеру Z3 Конрада Цузе, построенному в 1941 г. Стоит отметить, что машина Z3 была значительно меньше машины Эйкена и к тому же дешевле в производстве
 Конрад Цузе (1910—1995)  Машина Z3	В 1934 г., будучи студентом технического вуза (в Берлине), не имея ни малейшего представления о работах Ч. Бэббиджа, К. Цузе начал разрабатывать универсальную вычислительную машину, во многом подобную аналитической машине Бэббиджа. В 1938 г. он завершил постройку машины, занимавшую площадь 4 кв. м., названную Z1 (от Zuse). Это была *полностью электромеханическая программируемая цифровая машина*. Она имела клавиатуру для ввода условий задач. Результаты вычислений высвечивались на панели с множеством маленьких лампочек. Ее восстановленная версия хранится в музее Verker und Technik в Берлине. Именно Z1 считают в Германии первым в мире компьютером. Позднее Цузе стал кодировать инструкции для машины, пробивая отверстия в использованной 35-миллиметровой фотопленке. Машина, работавшая с перфорированной лентой, получила название Z2. В 1941 г. Цузе построил программно-управляемую машину, основанную на двоичной системе счисления — Z3. Эта машина по многим своим характеристикам превосходила другие машины, построенные независимо и параллельно в иных странах. В 1942 г. Цузе совместно с австрийским инженером-электриком Хельмутом Шрайером предложил создать компьютер принципиально нового типа — на вакуумных электронных лампах. Эта машина должна была работать в тысячу раз быстрее, чем любая из машин, имевшихся в то время в Германии.

Продолжение табл. 1.1

Авторы и изделия	Хронология и описание (комментарий)
	Говоря о потенциальных сферах применения быстродействующего компьютера, Цузе и Шрайер отмечали возможность его использования для расшифровки закодированных сообщений (такие разработки уже велись в различных странах)
 Алан Тьюринг (1912—1954)	Английский математик дал математическое определение алгоритма через построение, названное «машиной Тьюринга». Увы, совершенно невозможно привести здесь ее фотографию, как это мы сделали для других устройств, поскольку это сугубо теоретическое понятие!... В современном научном обороте широко используется понятие «полнота по Тьюрингу», отражающее универсальную описательную способность алгоритмического языка. В период Второй мировой войны разведчики и дипломаты фашистской Германии использовали аппарат «Enigma» для шифрования сообщений. Без знания ключа и схемы коммутации (их меняли 3 раза в день) расшифровать сообщение было невозможно. С целью раскрытия секрета британская разведка собрала группу блестящих и несколько эксцентричных ученых, среди которых был Алан Тьюринг. В конце 1943 г. группа сумела построить мощную машину (вместо электромеханических реле в ней применялись около 2000 электронных вакуумных ламп), названную «Колосс». Перехваченные сообщения кодировались, наносились на перфоленту и вводились в машину посредством фотоэлектрического считывающего устройства со скоростью 5000 символов в секунду. Машина имела пять таких считывающих устройств. В процессе дешифрования машина сопоставляла зашифрованное сообщение с ранее вскрытыми кодами «Enigma». О роли Тьюринга в работе группы можно судить по следующему высказыванию члена этой группы, математика И. Дж. Гуда: «Я не хочу сказать, что мы выиграли войну благодаря Тьюрингу, но беру на себя смелость сказать, что без него мы могли бы ее и проиграть»
 Джон Мочли (1907—1980)	Длительное время первой ЭВМ считалась машина ЭНИАК (ENIAC, Electronic Numerial Integrator and Computer — электронный цифровой интегратор и вычислитель). Ее разработали американские ученые Дж. Мочли и Преспер Экерт с 1943 по 1945 г. Она предназначалась для расчета траекторий полетов снарядов и представляла собой сложнейшее для середины XX в. инженерное сооружение длиной более 30 м, объемом 85 куб. м, массой 30 т. В ЭНИАКе были использованы 18 тыс. электронных ламп, 1500 реле, машина потребляла около 150 кВт. Далее возникла идея создания машины с программным обеспечением, хранимым в памяти машины, что изменило бы принципы организации

Продолжение табл. 1.1

Авторы и изделия	Хронология и описание (комментарий)
 Преспер Экерт (1919—1995)  ЭНИАК	вычислений и подготовило почву для появления современных языков программирования (ЭДВАК — Электронный Автоматический Вычислитель с дискретными переменными, EDVAC — Electronic Discret Variable Automatic Computer). Эта машина была создана в 1950 г. В более емкой внутренней памяти содержались и данные, и программа. Программы записывались электронным способом в специальных устройствах — линиях задержки. Основным новшеством было то, что в ЭДВАКе данные кодировались не в десятичной системе, а в двоичной (сократилось количество используемых электронных ламп). Дж. Мочли и П. Экерт после создания своей собственной компании задались целью создать универсальный компьютер для широкого коммерческого применения — ЮНИВАК (UNIVAC, Universal Automatic Computer — универсальный автоматический компьютер). Примерно за год до того, как первый ЮНИВАК вступил в эксплуатацию в Бюро переписи населения в США, партнеры оказались в тяжелом финансовом положении и вынуждены были продать свою компанию фирме «Ремингтон Рэнд». Однако Юнивак не стал первым коммерческим компьютером. Им стала машина ЛЕО (LEO — Lyons' Electronic Office), которая применялась в Англии для расчета зарплаты работникам чайных магазинов (фирма «Лайонс»). В 1973 г. федеральный суд США признал их авторские права на изобретение электронного цифрового компьютера недействительными, а идеи — заимствованными у Дж. Атанасоффа
 Джон фон Нейман (1903—1957)	Работая в группе Дж. Мочли и П. Экерта, фон Нейман подготовил отчет — «Предварительный доклад о машине ЭДВАК», в котором обобщил планы работы над машиной. Это была первая работа по цифровым электронным компьютерам, с которой познакомились определенные круги научной общественности (по соображениям секретности работы в этой области не публиковались). С этого момента компьютер был признан объектом, представлявшим научный интерес. В своем докладе фон Нейман выделил и детально описал пять ключевых компонентов того, что ныне называют «архитектурой фон Неймана» современного компьютера. В нашей стране независимо от фон Неймана были сформулированы более детальные и полные принципы построения электронных цифровых вычислительных машин (Сергей Алексеевич Лебедев)

Продолжение табл. 1.1

Авторы и изделия	Хронология и описание (комментарий)
 Сергей Алексеевич Лебедев (1902—1974)	В 1946 г. С. А. Лебедев становится директором института электротехники и организует в его составе свою лабораторию моделирования и регулирования. В 1948 г. С. А. Лебедев ориентировал свою лабораторию на создание МЭСМ (Малая электронная счетная машина). МЭСМ была вначале задумана как модель (первая буква в аббревиатуре МЭСМ) Большой электронной счетной машины (БЭСМ). Однако в процессе ее создания стала очевидной целесообразность превращения ее в малую ЭВМ. Из-за засекреченности работ, проводимых в области вычислительной техники, соответствующих публикаций в открытой печати не было. Основы построения ЭВМ, разработанные С. А. Лебедевым независимо от Дж. фон Неймана, заключаются в следующем: 1) в состав ЭВМ должны входить устройства арифметики, памяти, ввода-вывода информации, управления; 2) программа вычислений кодируется и хранится в памяти подобно числам; 3) для кодирования чисел и команд следует использовать двоичную систему счисления: 4) вычисления должны осуществляться автоматически на основе хранимой в памяти программы и операций над командами; 5) помимо арифметических операций вводятся также логические — сравнения, условного и безусловного переходов, конъюнкция, дизъюнкция, отрицания; 6) память строится по иерархическому принципу; 7) для вычислений используются численные методы решения задач.
 МЭСМ	25 декабря 1951 г. МЭСМ была принята в эксплуатацию. Это была первая в СССР быстродействующая электронная цифровая машина.
 ЭВМ БЭСМ-6	В 1948 г. создается Институт точной механики и вычислительной техники (ИТМ и ВТ) АН СССР, которому правительство поручило разработку новых средств вычислительной техники, и С. А. Лебедев приглашается заведовать лабораторией № 1 (1951 г). Когда БЭСМ была готова (1953 г.), она ничуть не уступала новейшим американским образцам. С 1953 г. до конца своей жизни С. А. Лебедев был директором ИТМ и ВТ АН СССР, избран действительным членом АН СССР и возглавил работы по созданию нескольких поколений ЭВМ. В начале 60-х гг. создается первая ЭВМ из серии больших электронных счетных машин (БЭСМ) — БЭСМ-1. При создании БЭСМ-1 были применены оригинальные научные и конструкторские решения. Благодаря этому она была тогда самой производительной машиной в Европе (8—10 тысяч операций в секунду)

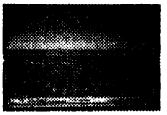
Продолжение табл. 1.1

Авторы и изделия	Хронология и описание (комментарий)
	и одной из лучших в мире. Под руководством С. А. Лебедева были созданы и внедрены в производство еще две ламповые ЭВМ — БЭСМ-2 и М-20. В 60-х гг. были созданы полупроводниковые варианты М-20: М-220 и М-222, а также БЭСМ-3М и БЭСМ-4. При проектировании БЭСМ-6 впервые был применен метод предварительного имитационного моделирования (сдача в эксплуатацию была осуществлена в 1967 г.). С. А. Лебедев одним из первых понял огромное значение совместной работы математиков и инженеров в создании вычислительных систем. По инициативе С. А. Лебедева все схемы БЭСМ-6 были записаны формулами булевой алгебры. Это открыло широкие возможности для автоматизации проектирования и подготовки монтажной и производственной документации
АВМ  Настольная АВМ МН-7  МН-8 — первая в СССР прецизионная АВМ большой мощности	АВМ — аналоговые вычислительные машины (40-е — 70-е гг. XX в.) или ВМ непрерывного действия, обрабатывают информацию, представленную в виде непрерывного ряда значений. Если говорить об отечественных АВМ, то в 1949—1950 гг. были созданы первые АВМ, называемые интеграторами постоянного тока — ИПТ-1—ИПТ-5. Они предназначались для решения линейных дифференциальных уравнений с постоянными и переменными коэффициентами. Разработанные в 1952—1953 гг. АВМ получают наименование «моделирующие установки постоянного тока» (МПТ). С 1954 г. АВМ получают название «моделирующие установки нелинейные» (МН). В течение 1954—1959 гг. разрабатываются следующие АВМ: МН-2, секционная АВМ для решения дифференциальных уравнений 6-го порядка; МН-7, настольная АВМ 6-го порядка. В 1963 г. появилась МН-16, предназначенная для моделирования ракет и ракетных систем. В 1965 г. выпущены вычислительное устройство для авиационных тренажеров «Счет-22» и АВМ «Доза» для расчета дозных полей при лучевой терапии. В 1967—1968 гг. разработан «Сеграф-1» для исследования сетевых графиков и «Трансграф-1» для моделирования транспортных задач. Эквивалентное быстродействие АВМ достигало десятков мегафлопс (миллионов операций с плавающей запятой в секунду), т. е. чтобы решать систему дифференциальных уравнений с подобной скоростью, ЦВМ должны были обладать именно таким быстродействием, до которого в те времена было еще очень далеко

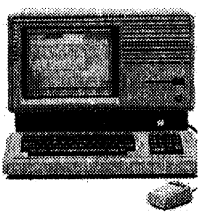
Продолжение табл. 1.1

Авторы и изделия	Хронология и описание (комментарий)
IBM  IBM/360  ЕС 1045	Невозможно пропустить ключевой этап в развитии вычислительных средств и методов, связанный с деятельностью фирмы IBM. Исторически первые ЭВМ классической структуры и состава — Computer Installation System/360 (фирменное наименование — «Вычислительная установка системы 360», в дальнейшем известная как просто IBM/360) были выпущены в 1964 г., и с последующими модификациями (IBM/370, IBM/375) поставлялись вплоть до середины 80-х гг., когда под влиянием микроЭВМ (ПК) не начали постепенно сходить со сцены. ЭВМ данной серии послужили основой для разработки в СССР и странах-членах СЭВ так называемой Единой системы ЭВМ (ЕС ЭВМ), которые в течение нескольких десятилетий являлись основой отечественной компьютеризации. Машины включали следующие компоненты: • центральный процессор (32-разрядный) с двухадресной системой команд; • главную (оперативную) память (от 128 Кбайт до 2 Мбайт); • накопители на магнитных дисках (НМД, МД) со сменными пакетами дисков (например, IBM-2314 — 7,25 Мбайт, IBM-2311 — 29 Мбайт, IBM 3330 — 100 Мбайт), аналогичные (иногда совместимые) устройства известны и для других из вышеупомянутых серий; • накопители на магнитных лентах (НМЛ, МЛ) катушечного типа, ширина ленты 0,5 дюйма, длина от 2400 футов (720 м) и менее (обычно 360 и 180 м), плотность записи от 256 байт на дюйм (обычная) и бóльшая в 2—8 раз (повышенная). Соответственно рабочая емкость накопителя определялась размером катушки и плотностью записи и достигала 160 Мбайт на бобину МЛ; • устройства печати — построчные печатающие устройства барабанного типа с фиксированным (обычно 64 или 128 знаков) набором символов, включающих заглавную латиницу и кириллицу (либо заглавную и строчную латиницу) и стандартное множество служебных символов; вывод информации осуществлялся на бумажную ленту шириной 42 или 21 см со скоростью до 20 строк/с; • терминальные устройства (видеотерминалы, а первоначально — электрические пишущие машинки), предназначенные для интерактивного взаимодействия с пользователем (IBM 3270, DEC VT-100 и пр.), подключаемые к системе для выполнения функций управления вычислительным процессом (консоль оператора — 1—2 шт. на ЭВМ) и интерактивной отладки программ и обработки данных (терминал пользователя — от 4 до 64 шт. на ЭВМ).

Продолжение табл. 1.1

Авторы и изделия	Хронология и описание (комментарий)
	Перечисленные стандартные наборы устройств ЭВМ 60—80-х гг. и их характеристики приведены здесь как историческая справка для читателя, который может их самостоятельно оценить, сравнив с современными и известными ему данными. Фирмой IBM была предложена в качестве оболочки ЭВМ IBM/360 первая функционально полноценная ОС — OS/360. Разработка и внедрение ОС позволили разграничить функции операторов, администраторов, программистов, пользователей, а также существенно (в десятки и сотни раз) повысить производительность ЭВМ и степень загрузки технических средств. Версии OS/360/370/375 — MFT (мультипрограммирование с фиксированным количеством задач), MVT (с переменным количеством задач), SVS (система с виртуальной памятью), SVM (система виртуальных машин) — последовательно сменяли друг друга и во многом определили современные представления о роли ОС
 Билл Гейтс и Пол Аллен  Альтаир	1974 г. Фирма Intel разработала первый универсальный 8-разрядный микропроцессор 8080 с 4500 транзисторами. Эдвард Робертс, молодой офицер ВВС США, инженер-электронщик, построил на базе процессора 8080 микрокомпьютер Альтаир, имевший огромный коммерческий успех, продававшийся по почте и широко использовавшийся для домашнего применения. В 1975 г. Молодой программист Пол Аллен и студент Гарвардского университета Билл Гейтс реализовали для Альтаира язык Бейсик. Впоследствии они основали фирму Майкрософт (Microsoft)

Окончание табл. 1.1

Авторы и изделия	Хронология и описание (комментарий)
 Стивен Джобс и Стефан Возняк	В 1976 г. студенты Стив Возняк и Стив Джобс, устроив мастерскую в гараже, реализовали компьютер Apple-1, положив начало корпорации Apple. В 1983 г. корпорация Apple Computers построила персональный компьютер Lisa — первый офисный компьютер, управляемый манипулятором «мышь»
 Apple-1	
 Lisa	

Теоретической основой организации и функционирования вычислительных машин и систем являются следующие дисциплины:

- элементы теории информации — измерение количества информации, пропускная способность каналов, «сигнал—шум», децибеллы, кодирование и сжатие—восстановление информации;
- элементы теории чисел и вычислительной математики — системы счисления, представление чисел в различных системах, операции над числами, точность представлений и результатов операций;

- элементы математической логики — логические выражения и переменные, операции над ними, эквивалентные преобразования выражений, схемные элементы, узлы и переключательные схемы ЭВМ, базирующиеся на подобных преобразованиях;
- элементы теории алгоритмов (*алгорифмов*, «понаучному») — циклические, ветвящиеся, итерационные процессы, их свойства (эффективная вычислимость), зависимость точности вычислений от их длительности и пр.;
- другие разделы прикладной математики — в частности, теория графов, топологические преобразования конфигураций сетей и пр.

Очевидно, что мы не можем в ограниченных рамках данного учебного пособия подробно осветить указанные вопросы, однако далее мы приводим краткое изложение основных положений некоторых из этих дисциплин.

1.2. Информация, кодирование и обработка в ЭВМ

Понятие «информация» является таким же фундаментальным, как понятия «материя», «энергия» и другие философские категории. Это атрибут, свойство сложных систем, связанное с их развитием и самоорганизацией [24, 25]. Известно большое количество различных определений информации, отличие информации от данных, знаний и пр. Мы здесь ограничимся только рассмотрением некоторых практически важных понятий и определений.

Определение и классификация информации

В настоящее время наука пытается найти общие свойства и закономерности, присущие многогранному понятию «информация», но пока это понятие во многом остается интуитивным и получает различные смысловые наполнения в различных отраслях человеческой деятельности.

В обиходе информацией называют *любые данные или факты*, которые представляют какой-либо интерес. Например, сообщение о каких-либо событиях, о чьей-либо деятельности и т. п.

«Информировать» в этом смысле означает «сообщить нечто, неизвестное раньше».

В технике под информацией понимают *сообщения*, передаваемые в форме знаков или сигналов.

В кибернетике под информацией понимают ту часть *знаний*, которая используется для ориентирования, активного действия, управления, т. е. в целях сохранения, совершенствования, развития системы.

Приведем несколько определений информации:

- отрицание энтропии (Л. Бриллюэн);
- мера сложности структур (Моль);
- отраженное разнообразие (Урсул);
- содержание процесса отражения (Тузов);
- вероятность выбора (Яглом);
- снятая неопределенность наших знаний о чем-то (К. Шеннон);
- обозначение содержания, полученного из внешнего мира в процессе нашего приспособления к нему и приспособления к нему наших чувств (Н. Винер).

Информация может классифицироваться, например, по следующим основаниям:

- признаки, отражающие структуру данных и форму представления информации (табл. 1.2);
- содержание предметной области применения.

Таблица 1.2. **Некоторые классы информации (по структуре и форме)**

Основание для классификации	Классы информации			
По уровням сложности	Сигнал	Сообщение, документ	Информационный массив	Информационный ресурс
По типу сигнала	Аналоговая (непрерывная)	Цифровая (дискретная)		
По уровням доступа и организации	Данные в регистровой памяти	Данные в оперативной памяти	Файлы данных на внешних устройствах	Базы данных
По способам кодирования и представления (данные, файлы и БД)	Цифровая (вычислительные данные, двоичные)	Символьная (алфавитно-цифровая, строчная)	Графическая	
По организации данных (файлы и БД)	Табличная	Текстовая	Графическая	

Аналоговая информация. Исторически первой технологической формой получения, передачи, хранения информации являлось аналоговое (непрерывное) представление звукового, оптического, электрического или другого сигнала (сообщения). Магнитная аудио- и видеозапись, фотографирование, запись на шеллачные или виниловые грампластинки, проводное и радиовещание — основные способы хранения и передачи информации в аналоговой форме (рис. 1.1). Заметим, что с начала 50-х гг. (а во многом и сейчас) под термином ***теория информации*** подразумевались теоретические методы, связанные с обеспечением как можно более точного приема, передачи, записи, воспроизведения, преобразования непрерывных сигналов (основные понятия — *линейность, нелинейность, шум, спектр сигнала, полоса пропускания и пр.*).

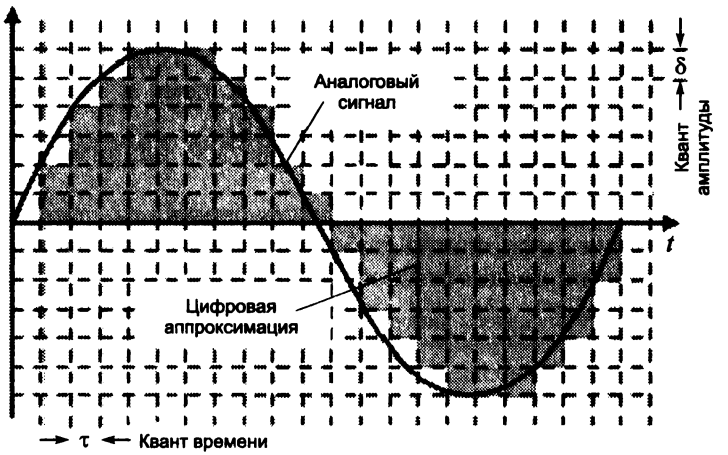


Рис. 1.1. Аналоговый сигнал и его дискретная (цифровая) аппроксимация (оцифровка)

Более чем 50-летнее развитие теории и практики ЭВМ приводит к вытеснению (в том числе и на бытовом уровне) аналоговых устройств и сигналов цифровыми.

Аналого-цифровое (дискретное) преобразование — АЦП (analog-to-digital conversion — ADC) заключается в формировании последовательностей *n*-разрядных двоичных слов, представляющих с заданной точностью аналоговые сигналы.

Наиболее популярным примером, несомненно, является аудиокомпакт-диск (digital audio CD). В этом случае звуковой

сигнал (см. рис. 1.1) преобразуется в дискретную аппроксимацию («многоуровневый ступенчатый сигнал»), при этом вначале происходит *квантование во времени,* которое заключается в измерении в дискретные промежутки времени необходимого параметра аналогового сигнала.

Затем осуществляется *квантование по амплитуде* сигнала. При квантовании аналогового сигнала происходит округление его мгновенных значений до некоторой заданной фиксированной величины, называемой *уровнем.* Наименьшее изменение аналогового сигнала, которое регистрируется устройством, осуществляющим преобразование, называется *разрешением.*

При *квантовании по амплитуде* каждая ступенька представляется последовательностью бинарных цифровых сигналов. Принятый в настоящее время стандарт аудио CD использует так называемый «16-разрядный звук с частотой сканирования 44 кГц». Для рис. 1.1 «в переводе на нормальный язык» это означает, что «длина ступеньки» (τ) равна 1/44 000 с, а «высота ступеньки» (δ) составляет 1/65 536 максимальной громкости сигнала (поскольку $2^{16} = 65\,536$). При этом частотный диапазон воспроизведения составляет 0—22 кГц, а динамический диапазон — 96 дБ (недостижимое для магнитной или механической звукозаписи качество записи).

Аналого-цифровые (дискретные) преобразователи чаще всего изготавливаются в виде интегральных схем. В необходимых случаях осуществляется обратное — дискретно-аналоговое или цифроаналоговое преобразование — ЦАП (digital-to-analog conversion — DAC).

Цифровая информация. Дискретные сигналы по сравнению с аналоговыми имеют ряд важных преимуществ: помехоустойчивость, легкость восстановления формы, простоту аппаратуры передачи.

В дальнейшем оцифрованный аналоговый сигнал может передаваться или записываться в различных форматах, например:

- PCM (Pulse Code Modulation — импульсно-кодовая модуляция) — способ цифрового кодирования сигнала с помощью записи абсолютных значений амплитуд $A(t)$ (рис. 1.2, *а*). Именно в таком виде данные записываются на аудио CD;
- ADPCM (Adaptive Delta PCM — адаптивная относительная импульсно-кодовая модуляция) — запись значений сигнала

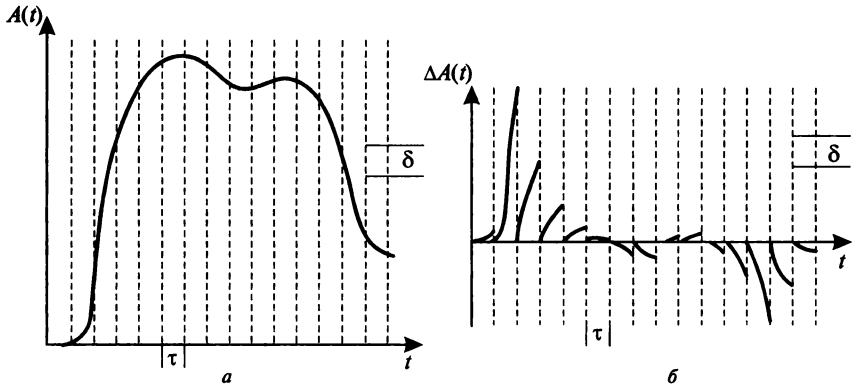


Рис. 1.2. Примеры кодирования аналоговых сигналов:
а — импульсно-кодовая модуляция; *б* — адаптивная относительная импульсно-кодовая модуляция

не в абсолютных, а в относительных приращениях ($\Delta A(t)$) амплитуд (рис. 1.2, *б* и т. д.).

Передача данных. Канал передачи — это комплекс технических средств и среды распространения, обеспечивающий передачу сигнала электросвязи в определенной полосе частот или с определенной скоростью передачи между сетевыми станциями и узлами.

При обмене данными по каналам используется три метода передачи данных:

- симплексная (однонаправленная) передача (телевидение, радио);
- полудуплексная (прием и передача информации осуществляются поочередно);
- дуплексная (двунаправленная), каждая станция одновременно передает и принимает данные.

Для передачи данных в информационных системах наиболее часто применяется последовательная передача. Широко используются следующие методы последовательной передачи — асинхронная и синхронная.

При асинхронной передаче каждый символ передается отдельной посылкой (рис. 1.3).

Стартовые биты предупреждают приемник о начале передачи. Затем передается символ. Для определения достоверности передачи используется бит четности (бит четности равен 1, если количество единиц в символе нечетно, и 0 — в противном слу-

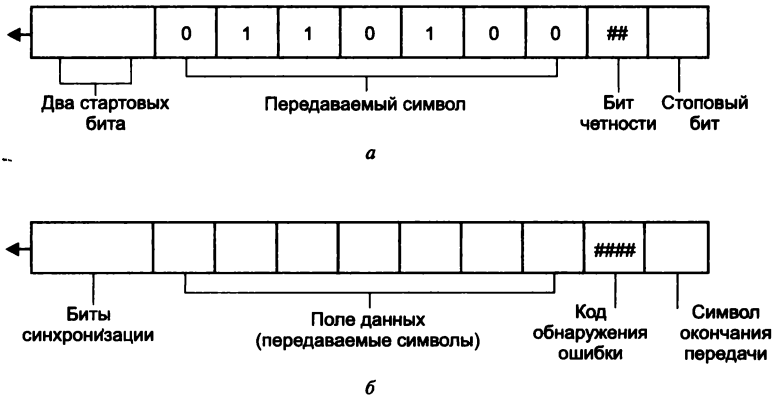


Рис. 1.3. Асинхронная (*а*) и синхронная (*б*) передача данных

чае. Последний бит («стоп-бит») сигнализирует об окончании передачи.

При использовании синхронного метода данные передаются блоками. Для согласования работы приемника и передатчика в начале блока передаются биты синхронизации. Затем передаются данные, код обнаружения ошибки и символ окончания передачи. При синхронной передаче данные могут передаваться и как символы, и как поток битов. В качестве кода обнаружения ошибки обычно используется код CRC.

Практически всегда дискретный сигнал имеет два либо три значения. Нередко его называют также *цифровым сигналом*. Дискретные сигналы по сравнению с аналоговыми имеют ряд важных преимуществ: помехоустойчивость, легкость восстановления формы, простоту аппаратуры передачи данных.

Обычно в цифровых системах используются двоичные сигналы (рис. 1.4, *а*), имеющие значения «+», «–». Вместе с тем при передаче данных в большинстве случаев применяются троичные сигналы (рис. 1.4, *б*) со значениями «+», «0», «–». Здесь «единица» представляется отсутствием потенциала в канале, тогда как «нуль» характеризуется положительным либо отрицательным импульсом. При этом полярность импульсов, представляющих «нули», должна чередоваться, т. е. за положительным «+» импульсом должен следовать отрицательный «–» и наоборот. В форме троичного сигнала осуществляется не только кодирование передаваемых данных, но также обеспечивается синхронизация работы канала и проверка целостности данных.

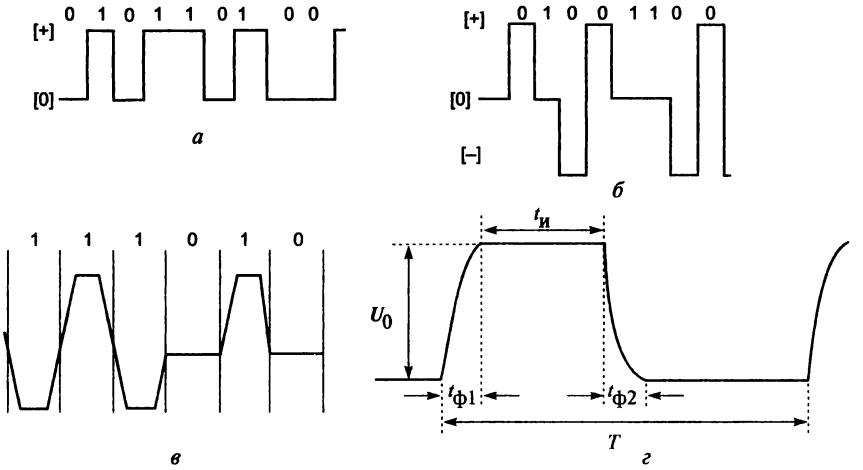


Рис. 1.4. Примеры дискретных сигналов:
а — двоичный; *б* — троичный; *в* — биполярный; *г* — элементы импульсного сигнала

Биполярные коды также часто используются в каналах передачи данных (рис. 1.4, *в*). Здесь единицы представляются чередующимися положительными и отрицательными импульсами. Отсутствие импульсов определяет состояние «нуль». Биполярное кодирование обеспечивает обнаружение одиночной ошибки. Так, если вместо нуля появится единица либо единица ошибочно сменится на нуль, то это легко обнаруживается. В обоих случаях нарушается чередование полярности импульсов.

Реальный импульсный сигнал (рис. 1.4, *г*) характеризуется амплитудой U_0, длительностью импульсов $t_и$, передним фронтом $t_{ф1}$, задним фронтом $t_{ф2}$ и периодом повторения T. Отношение длительности периода T к длительности импульсов $t_и$ называется скважностью.

Измерение количества информации

Термин «информация» имеет корень «form» (форма), что разумно трактовать как «информирование — придание формы, вывод из состояния неопределенности, бесформенности», поэтому логично подходить к определению понятия «количество информации», исходя из того, что информацию, содержащуюся в сообщении, можно трактовать в смысле ее новизны или, иначе,

уменьшения неопределенности знаний «приемника информации» об объекте.

В качестве единицы информации Клод Шеннон предложил принять один б и т (от *англ.* bit — binary digit — двоичная цифра).

В вычислительной технике битом называют наименьшую «порцию» памяти компьютера, необходимую для хранения одного из двух знаков 0 и 1, используемых для машинного представления данных и команд.

Поскольку бит — слишком мелкая единица измерения, на практике чаще применяется более крупная единица — б а й т, равная восьми битам. В частности, восемь бит требуется для того, чтобы закодировать любой из 256 символов ASCII ($256 = 2^8$).

«Имеют хождение» также более крупные производные единицы информации: килобайт (Кбайт, KB) = 10^3 байт, мегабайт (Мбайт, MB) = 10^3 Кбайт, гигабайт (Гбайт, GB) = 10^3 Мбайт, терабайт (Тбайт, TB) = 10^3 Гбайт, петабайт (Пбайт, PB) = 10^3 Тбайт.

Заметим, что часто возникает неоднозначность в интерпретации, например, 1 Мбайта, который может рассматриваться как 1000 Кбайт (десятичный мегабайт), так и 1024 Кбайт (бинарный мегабайт). В табл. 1.3 приведены последствия такой неоднозначности.

Таблица 1.3. **Отклонения, возникающие при одинаковом прочтении десятичных и двоичных префиксов единиц измерения информации**

Префикс	Сокращение	Подразумеваемое двоичное значение	Десятичное значение	Относительное отклонение
кило	к	2^{10}	10^3	1,024 = 1024/1000
мега	М	2^{20}	10^6	1,0486 = 1 048 576/1 000 000
гига	Г	2^{30}	10^9	1,0737
тера	Т	2^{40}	10^{12}	1,0995
пета	П	2^{50}	10^{15}	1,1259
экса	Э	2^{60}	10^{18}	1,1529
зетта	З	2^{70}	10^{21}	1,1806
йотта	Й	2^{80}	10^{24}	1,2089

Очевидно, что чем больше изображаемое число, тем большего значения может достигать расхождение, вызванное неоднозначным пониманием использованного префикса. В частности, разница между «двоичным» и «десятичным» килобайтом состав-

ляет 2,4 %, но между «двоичным» и «десятичным» гигабайтом — уже более 7 %. Для того чтобы разрешить эту путаницу, было предложено ввести двоичные префиксы.

В марте 1999 г. МЭК — Международная электротехническая комиссия (International Electrotecnical Comission — IEC) предложила новый стандарт для обозначения двоичных чисел. Префиксы МЭК схожи с привычными префиксами СИ (Международной системы измерения физических единиц) — они начинаются одинаково, но второй слог двоичных префиксов — *би* (от *англ.* binary — «двоичный»). Стандарт был утвержден, но введенные названия используются нерегулярно, очевидно, из-за их необычности: «килобит» звучит привычнее, нежели «кибибит». Российский ГОСТ 8.417—2002 («Единицы величин») также определяет свое написание двоичных префиксов для байтов (табл. 1.4).

Таблица 1.4. **Соответствие префиксов МЭК и ГОСТ 8.417—2002**

Префикс	Сокращения МЭК для битов/байтов	Сокращение ГОСТ 8.417—2002 (для байтов)	Значение
Киби, KiB	Kibibit, Кибибит, KibiByte, KiB, КиБ	Кбайт	$2^{10} = 1024$
Миби, MiB	Mibibit, Мибибит, MibiByte, MiB, МиБ	Мбайт	$2^{20} = 1\,048\,576$
Гиби, GiB	Gibibit, Гибибит, GibiByte, GiB, ГиБ	Гбайт	$2^{30} = 1\,073\,741\,824$
Тиби, TiB	Тибибит, Gibibit, TibiByte, TiB, ТиБ	Тбайт	$2^{40} = 1\,099\,511\,627\,776$
Пиби, PiB	Пибибит, ПиБ	Пбайт	2^{50}
Эксби	Эксбибит, ЭиБ	Эбайт	2^{60}
Зиби	Зибибит, ЗиБ	Эбайт	2^{70}
Йоби	Йобибит, ЙиБ	Йбайт	2^{80}

Пока что нельзя утверждать, что данные стандарты всеми и безусловно соблюдаются, поскольку разночтение продолжает оставаться.

Двоичные префиксы используются:

- обычно в файловых менеджерах и другом программном обеспечении для краткого определения размеров файлов — если программа сообщает, что размер файла равен 100 Кбайт (100 KB), то это соответствует приблизительно 102,4 тыс. байт;

- производителями различных видов полупроводниковых запоминающих устройств (ОЗУ, ПЗУ, флэш-память), например «карта Secure Digital на 1 Гбайт»;
- емкость компакт-дисков (например, «700 Мбайт») задается обычно в «двоичных мегабайтах»;
- согласно ГОСТ 8.417—2002 приставки К-, М- и Г- (прописными буквами) применительно к байтам имеют двоичное значение.

Приставки «кило», «мега», «гига» трактуются как десятичные:

- в телекоммуникационных приложениях, например «канал на 128 килобит в секунду»;
- как исторически сложившаяся терминология при указании объема жестких дисков (диск на «40 гигабайт» имеет размер 40 млрд байт). Сторонники противоположного подхода обвиняют фирмы-производители в завышении цифр объема носителей с помощью более мелкой единицы («коммерческий мегабайт»);
- емкость DVD (4,7 гигабайт) также задается в десятичных гигабайтах;
- при неформальном общении (например, про файл в 100 тысяч байт могут сказать «файл в 100 килобайт»).

Есть и иные примеры, в частности, размер трехдюймовой дискеты в 1,44 Мбайт (включая служебные данные — загрузочный сектор, и таблицы FAT) задается в двоично-десятичных мегабайтах (один мегабайт равняется 1 024 000 байтам).

Для описания скорости передачи данных можно использовать термин *бод*. Число бод равно количеству значащих изменений сигнала (потенциала, фазы, частоты), происходящих в секунду. Первоначально бод использовался в телеграфии. Для двоичных сигналов нередко принимают, что *бод равен биту в секунду*, например 1200 бод = 1200 бит/с. Однако единого мнения о правильности использования этого термина нет, особенно при высоких скоростях, где число бит в секунду не совпадает с числом бод.

Кодирование символьной информации

Код (*code*) — совокупность знаков, символов и правил представления информации. Рассмотрим методы дискретного представления информации, или кодирования (которые, надо ска-

зать, появились задолго до вычислительных машин). Первым широко известным примером является азбука Морзе (табл. 1.5), в которой буквы латиницы (или кириллицы) и цифры кодируются сочетаниями из «точек» и «тире». Воспользуемся данным кодом для иллюстрации основных понятий, связанных с кодированием (не вдаваясь в теорию кодирования).

Таблица 1.5. **Фрагменты кода Морзе**

Символ входного алфавита	Мнемоническое обозначение по МСС*	Кодовая (знаковая) комбинация
A	alfa	.-
B	bravo	-...
C	charlie	-.-.
D	delta	-..
E	echo	.
...	...	...
Y	yankee	-.—
Z	zulu	—..
1	one	.——
...	...	...
9	nine	——.

* Международный Свод Сигналов.

Кодируемые (обозначаемые) элементы входного алфавита обычно называют символами.

Символом (служит условным знаком какого-нибудь понятия, явления), как правило, является цифра, буква, знак пунктуации или иероглиф естественного языка, знак препинания, знак пробела, специальный знак, символ операции. Кроме этого, учитываются *управляющие («непечатные») символы.*

Кодирующие (обозначающие) элементы выходного алфавита называются знаками; количество различных знаков в выходном алфавите назовем *значностью* (*-арностью*, *-ичностью*, например «бинарный» или «двоичный» код); количество знаков в кодирующей последовательности для одного символа — *разрядностью кода.*

Пространственно-временно́е расположение знаков кода приводит к понятиям *параллельных* или *последовательных* кодов. При последовательном коде каждый временной такт предназначен для отображения одного разряда слова. Здесь все разряды слова фиксируются по очереди одним и

тем же элементом и проходят через одну и ту же линию передачи (например, радио- или оптические сигналы либо передача данных по двум проводам, двухжильному кабелю).

При параллельном коде все знаки символа представляются в одном временном такте, каждый знак проходит через отдельную линию (например, по четырем проводам, четырехжильному кабелю), образуя символ (т. е. символ передается в один прием, в один момент времени).

Для последовательного кода характерно временно́е разделение каналов при передаче информации, для параллельного — пространственное. В зависимости от применяемого кода различаются устройства параллельного и последовательного действия.

Применительно к азбуке Морзе (АМ):

- *символами* являются элементы языкового алфавита (буквы A—Z или А—Я) и цифровой алфавит (здесь — цифры 0—9);
- знаками являются «точка» и «тире» (или «+» и «–» либо «1» и «0», короче — два любых разных знака);
- поскольку знаков два, АМ является *двузначным* (*бинарным*, *двоичным*) кодом, если бы их было 3, то мы имели бы дело с *троичным*, *тернарным*, *трехзначным* кодом;
- поскольку число знаков в АМ колеблется от 1 (буквы Е, Т) до 5 (цифры), здесь имеет место код с *переменной разрядностью* (в АМ часто встречающиеся в тексте символы обозначены более короткими кодовыми комбинациями, нежели редкие символы);
- поскольку знаки передаются последовательно (электрические импульсы, звуковые или оптические сигналы разной длины, соответствующие «точкам» и «тире»), АМ есть *последовательный код*.

Первые опыты телеграфной и радиосвязи осуществлялись именно посредством АМ, причем приемное устройство записывало импульсы переменной длины в виде «точек» и «тире» на движущуюся телеграфную ленту, однако уже в начале XX в. был осуществлен переход на 5-разрядный (5-битовый) телеграфный код.

В табл. 1.6, 1.7 приводится перечень наиболее известных кодов, некоторые из них использовались первоначально для связи, кодирования данных, а затем для представления информации в ЭВМ.

Таблица 1.6. **Характеристики некоторых наиболее известных кодов**

Наименование кода	Расшифровка/перевод	Другие названия	Разрядность	Комментарий
Baudot	Код Бодо	IA-1 — international alphabet № 1	5	В прошлом — европейский стандарт для телеграфной связи
M2	MKKTT-2 CCITT-2	IA-2	5	Телеграфный код, предложенный Международным Комитетом по телефонии и телеграфии (MKKTT) и заменивший код Бодо
ASCII-7	American Standard Code for Information Interchange	ISO-7 IA-5, USASCII, ANSI X3.4	7	Код для передачи данных, поддерживает 128 символов, включающих прописные и строчные символы латиницы, цифры, специальные значки и управляющие символы. После добавления некоторых национальных символов (10 бинарных комбинаций) был принят Международной организацией по стандартизации (ISO) как стандарт ISO-7
ASCII-8	То же		8	Для внутреннего и внешнего представления данных в вычислительных системах. Включает стандартную часть (128 символов) и национальную (128 символов). В зависимости от национальной части кодовые таблицы различаются
EBCDIC	Expanded Binary Coded Decimal Information Code		8	Предложен фирмой IBM для машин серий IBM/360-375 (внутреннее представление данных в памяти), а затем распространившийся и на системы других производителей
Hollerith	Код Холлерита	Код перфокарт (ПК)	12	Предложен для ПК (1913 г.), затем использовавшийся для кодирования информации перед вводом в ЭВМ с ПК
UNICODE	UNIversal Code		16	Поскольку в 16-разрядном UNICODE можно закодировать 65 536 символов вместо 128 в ASCII, то отпадает необходимость в создании модификаций таблиц кодов. UNICODE охватывает 28 000 букв, знаков, слогов, иероглифов национальных языков мира

Таблица 1.7. **Фрагменты некоторых кодовых таблиц (указаны 16-ричные коды символов)**

Символ	IA-2	Бодо	ISO-7	EBCDIC	ASCII-8	Холлерит
A	03	10	41	C1	A1	900
B	19	06	42	C2	A2	880
C	0E	16	43	C3	A3	840
D	09	1E	44	C4	A4	820
a			61	81	E1	
b			62	82	E2	
c			63	83	E3	
d			64	84	E4	
. (точка)	1C	05	2E	4B	4E	842
, (запятая)	0C	09	2C	6B	4C	242
: (двоеточие)	1E		3B	5E	5B	40A
? (вопрос)	10	0D	3F	6F	5F	206

Избыточные коды

При записи и передаче данных часто используются *избыточные коды,* т. е. такие, которые за счет усложнения структуры позволяют повысить надежность передачи данных.

Коды с обнаружением ошибок. Распространенным методом обнаружения ошибок является контроль по четности. В этом случае при записи байта информации в запоминающее устройство генерируется дополнительный контрольный бит, в который записывается «0», если это число четное, и «1», если оно нечетное. Если при чтении ранее записанного байта вновь получить контрольный бит и сравнить его с уже имеющимся, то можно судить о достоверности получаемой информации.

Широко используется для обнаружения ошибок в блоках данных также код с циклическим контролем — *циклический избыточный код обнаружения ошибок* (Cyclic Redundance Check — CRC). Здесь вычисляется контрольная сумма содержимого блока дан-

ных перед его передачей, включается в одно из полей блока, а затем повторно вычисляется после передачи. Несовпадение результатов свидетельствует об ошибке в передаваемом содержимом.

Корректирующие коды. В ответственных приложениях, требующих повышенной надежности хранения информации, применяются более серьезные, чем контроль четности, методы обеспечения целостности данных. К ним относятся корректирующие коды (Error Correction Code — ECC), позволяющие не только обнаруживать ошибки, но и восстанавливать искаженную информацию за счет ее избыточности. Так, существуют модули памяти со схемами ECC, в которых для хранения контрольной информации используется не один, а два бита, в которых хранится остаток от деления числа на 4 (деление по модулю 4). Благодаря этим данным схема ECC умеет обнаруживать и исправлять одиночные искаженные биты, а также обнаруживать (но не исправлять) двойные ошибки. Модули памяти с ECC обычно стоят заметно дороже и применяются в основном в серверах. В общем случае ECC применяются во всех современных дисковых и ленточных накопителях. За счет информационной избыточности закодированных данных удается восстанавливать поврежденные блоки информации длиной в сотни байт. Наиболее широко применяются помехоустойчивые коды Рида — Соломона (Reed — Solomon), а также код Хемминга, позволяющий исправлять одиночные ошибки, появляющиеся в блоках данных.

Кодирование и обработка чисел

Кроме кодирования символов, в ЭВМ очевидную важность имеет кодирование и представление чисел.

Системы счисления. Человек привык считать предметы десятками, сотнями: десять единиц образуют десяток, десять десятков — сотню, десять сотен — тысячу и т. д. Это — десятичная система счисления, которая не является единственно возможной (известна, например, двенадцатеричная система счисления).

Система счисления — способ именования и изображения чисел с помощью символов, имеющих определенные количественные значения. В зависимости от способа изображения чисел системы счисления делятся на:

- непозиционные;
- позиционные.

Непозиционные системы счисления. В такой системе цифры не меняют своего количественного значения при изменении их расположения в числе.

Самый простой и очевидный пример — система счисления, где количество обозначается I (палочкой/единицей):

1 = I;

2 = I I;

5 = I I I I I;

10 = I I I I I I I I I I.

Пусть далее следующие символы (цифры в гипотетической системе счисления) соответствуют числам (в десятичной системе счисления):

♊ — 1;

♎ — 6;

♒ —12;

♐ — 24;

♍ — 60;

♌ — 365,

и пусть есть правило, по которому число можно записать любой комбинацией таких символов, чтобы сумма обозначаемых ими чисел была равна заданному числу.

Тогда 444 можно записать по крайней мере двумя способами:

♌♍♒♎♊ (365 + 60 + 12 + 6 + 1);

♎♊♌♍♒ (6 + 1 + 365 + 60 + 12),

т. е. ♌♍♒♎♊ = ♎♊♌♍♒.

Такая система счисления является непозиционной, так как цифры не меняют своего количественного значения при изменении их расположения в числе.

Позиционные системы счисления. В этом случае количественное значение каждой цифры зависит от ее места (позиции) в числе.

Десятичная система счисления является позиционной, так как значение каждой цифры зависит от ее места (позиции) в числе.

Например,

$23 = 2 \times 10 + 3$;

$32 = 3 \times 10 + 2$

и $23 \neq 32$

Римская система счисления является смешанной, так как значение каждой цифры частично зависит от ее места (позиции) в числе. Так, в числах

VII

VI

IV

V обозначает 5, а **I** обозначает 1. Но, с другой стороны, важно, как цифры расположены относительно друг друга:

VII = 5 + 1 + 1 = 7;

VI = 5 + 1 = 6;

IV = 5 – 1 = 4.

Наиболее естественный способ представления числа в компьютерной системе заключается в использовании строки битов, называемой двоичным числом — числом в двоичной системе счисления (символ также может быть представлен строкой битов, или символа).

Основание позиционной системы счисления — количество (P) различных цифр, используемых для изображения числа в позиционной системе счисления. Значения цифр лежат в пределах от 0 до $P-1$.

В общем случае запись любого числа N в системе счисления с основанием P будет представлять собой ряд (многочлен) вида:

$$N = a_{m-1} \times P^{m-1} + a_{m-2} \times P^{m-2} + \ldots + a_k \times P^k + \ldots$$

$$\ldots + a_1 \times P^1 + a_0 \times P^0 + \ldots + a_{-1} \times P^{-1} + a_{-2} \times P^{-2} + \ldots + a_{-s} \times P^s. \quad (1.1)$$

Нижние индексы определяют местоположение цифры в числе (разряд):

- положительные значения индексов — для целой части числа (m разрядов);
- отрицательные значения — для дробной (s разрядов).

Максимальное целое число, которое может быть представлено в m разрядах:

$$N_{\max} = P^m - 1.$$

Минимальное значащее, не равное 0 число, которое можно записать в s разрядах дробной части:

$$N_{\min} = P^{-s}.$$

Имея в целой части числа m разрядов, а в дробной — s, можно записать P^{m+s} разных чисел.

Двоичная система счисления (основание $P = 2$) использует для представления информации две цифры — 0 и 1.

Существуют простые правила перевода чисел из одной системы счисления в другую, основанные, в том числе, и на выражении (1.1).

Например, двоичное число 101110,101 равно десятичному числу 46,625:

$$101110{,}101_2 = 1 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 +$$

$$+ 1 \times 2^{-1} + 0 \times 2^{-2} + 1 \times 2^{-3} = 46{,}625_{10}.$$

Практически перевод из двоичной системы в десятичную можно легко выполнить, надписав над каждым разрядом соответствующий ему вес и сложив затем произведения значений соответствующих цифр на их веса.

Например, двоичное число 01000001_2 равно 65_{10}. Действительно, $64 \times 1 + 1 \times 1 = 65$.

Вес	128	64	32	16	8	4	2	1
Цифра	0	1	0	0	0	0	0	1

Таким образом, для перевода числа из позиционной системы счисления с любым основанием в десятичную систему счисления можно воспользоваться выражением (1.1).

Обратный перевод из десятичной системы счисления в систему счисления с другим основанием непосредственно по (1.1) затруднителен, поскольку все арифметические действия, предусмотренные этой формулой, следует выполнять в той системе счисления, в которую число переводится. Обратный перевод выполняется значительно проще, если предварительно преобразовать отдельно целую и дробную части выражения (1.1) к виду:

$$N_{\text{цел}} = (((...(a_{m-1} \times P + a_{m-2}) \times P + ... + a_2) \times P + a_1) \times P + a_0);$$

$$N_{\text{др}} = P^{-1} \times (a_{-1} + P^{-1} \times (a_{-2} + P^{-1} \times (a_{-3} + ... + P^{-1} \times (a_{-s+1} + P^{-1} \times a_{-s})...))).$$

Алгоритм перевода числа из десятичной системы счисления в систему счисления с основанием P, основанный на этих выражениях, позволяет оперировать числами в той системе счисления, из которой число переводится, и может быть сформулиро-

ван следующим образом (при переводе смешанного числа следует переводить его целую и дробную части отдельно):

- для перевода целой части числа ее, а затем целые части получающихся частных от деления следует последовательно делить на основание P до тех пор, пока очередная целая часть частного не окажется равной 0. Остатки от деления, записанные последовательно справа налево, образуют целую часть числа в системе счисления с основанием P;
- для перевода дробной части числа ее, а затем дробные части получающихся произведений следует последовательно умножать на основание P до тех пор, пока очередная дробная часть произведения не окажется равной 0 или не будет достигнута нужная точность дроби. Целые части произведений, записанные после запятой последовательно слева направо, образуют дробную часть числа в системе счисления с основанием P.

Пусть требуется перевести смешанное число (например, 46,625) из десятичной в двоичную систему счисления.

1. Переводим целую часть числа:

46 : 2 = 23 (остаток 0);

23 : 2 = 11 (остаток 1);

11 : 2 = 5 (остаток 1);

5 : 2 = 2 (остаток 1);

2 : 2 = 1 (остаток 0);

1 : 2 = 0 (остаток 1).

Записываем остатки последовательно справа налево — 101110, т. е.

$46_{10} = 101110_2$.

2. Переводим дробную часть числа:

0,625 × 2 = 1,250;

0,250 × 2 = 0,500;

0,500 × 2 = 1,000 (дробная часть равна 0 ⇒ стоп).

Записываем целые части получающихся произведений после запятой последовательно слева направо — 0,101, т. е.

$0{,}625_{10} = 0{,}101_2$.

Окончательно: $46{,}625_{10} = 101110{,}101_2$.

Кроме двоичной и десятичной в компьютерах могут использоваться также двоично-десятичная и шестнадцатеричная системы счисления (табл. 1.8).

Таблица 1.8. **Перевод цифр из двоичной системы счисления в восьмеричную, шестнадцатеричную и десятичную и наоборот**

Триада	Восьмеричная цифра	Тетрада	Шестнадцатеричная цифра	Десятичное число	Двоично-десятичная запись
000	0	0000	0	0	0000–0000
001	1	0001	1	1	0000–0001
010	2	0010	2	2	0000–0010
011	3	0011	3	3	0000–0011
100	4	0100	4	3	0000–0100
101	5	0101	5	5	0000–0101
110	6	0110	6	6	0000–0110
111	7	0111	7	7	0000–0111
		1000	8	8	0000–1000
		1001	9	9	0000–1001
		1010*	A	10	0001–0000
		1011*	B	11	0001–0001
		1100*	C	12	0001–0010
		1101*	D	13	0001–0011
		1110*	E	14	0001–0100
		1111*	F	15	0001–0101

* Запрещены в двоично-десятичном представлении.

Шестнадцатеричная система счисления часто используется при программировании. Перевод чисел из шестнадцатеричной системы счисления в двоичную весьма прост — он выполняется поразрядно.

Для изображения цифр, больших 9, в шестнадцатеричной системе счисления применяются буквы A = 10, B = 11, C = 12, D = 13, E = 14, F = 15.

Например, шестнадцатеричное число F17B в двоичной системе имеет вид: 1111000101111011, а в десятичной — 61819.

Двоично-десятичная система счисления получила большое распространение в современных компьютерах ввиду легкости перевода в десятичную систему и обратно. Она используется там, где основное внимание уделяется не простоте технического построения машины, а удобству работы пользователя. В двоич-

но-десятичной системе счисления основанием системы счисления является число 10, но каждая десятичная цифра (0, 1, ..., 9) кодируется четырьмя двоичными цифрами.

Двоично-десятичная система не экономична с точки зрения реализации технического построения машины (примерно на 20 % увеличивается требуемое оборудование), но более удобна при подготовке задач и при программировании.

Представление чисел в ЭВМ

В ЭВМ применяются две формы представления чисел:

- естественная форма, или форма с фиксированной запятой (точкой) — ФЗ (ФТ);
- нормальная форма, или форма с плавающей запятой (точкой) — ПЗ (ПТ).

Фиксированная запятая (точка). В форме представления с *фиксированной запятой (точкой)* числа изображаются в виде последовательности цифр с постоянным для всех чисел положением запятой, отделяющей целую часть от дробной.

Например, пусть числа представлены в десятичной системе счисления и имеют пять разрядов в целой части числа (до запятой) и пять — в дробной части (после запятой). Числа, записанные в такую разрядную сетку, имеют вид:

+00721.35500;

+00000.00328;

–10301.20260.

Эта форма наиболее проста, естественна, но имеет небольшой диапазон представления чисел.

Диапазон значащих чисел N в системе счисления с основанием P при наличии m разрядов в целой части и s разрядов в дробной части числа (без учета знака числа) будет таким:

$$P^{-s} \le N \le P^m - P^{-s}.$$

Например, при $P = 2$, $m = 10$ и $s = 6$ числа изменяются в диапазоне $0{,}015 < N < 1024$. Если в результате операции получится число, выходящее за допустимые пределы, произойдет переполнение разрядной сетки, и дальнейшие вычисления теряют смысл. В современных компьютерах естественная форма пред-

ставления используется как вспомогательная и только для целых чисел.

В памяти ЭВМ числа с фиксированной точкой хранятся в трех форматах:

- полуслово — это обычно 16 бит или 2 байта;
- слово — 32 бита или 4 байта;
- двойное слово — 64 бита или 8 байтов.

Отрицательные числа с ФТ записываются в разрядную сетку в дополнительных кодах, которые образуются прибавлением единицы к младшему разряду обратного кода. Обратный код получается заменой единиц на нули, а нулей на единицы в прямом двоичном коде.

Плавающая запятая (точка). В форме представления с плавающей запятой (точкой) число изображается в виде двух групп цифр:

- мантисса;
- порядок.

При этом абсолютная величина мантиссы должна быть меньше 1, а порядок должен быть целым числом. В общем виде число в форме с плавающей запятой может быть представлено так:

$$N = \pm M \times P^{\pm r},$$

где M — мантисса числа ($|M| < 1$);

r — порядок числа (целое число);

P — основание системы счисления.

Например, приведенные ранее числа в нормальной форме запишутся следующим образом:

$+0{,}721355 \times 10^{3}$;

$+0{,}328 \times 10^{-3}$;

$-0{,}103012026 \times 10^{5}$.

Нормальная форма представления обеспечивает большой диапазон отображения чисел и является основной в современных компьютерах. Так, диапазон значащих чисел в системе счисления с основанием P при наличии m разрядов у мантиссы и s разрядов у порядка (без учета знаковых разрядов порядка и мантиссы) будет:

$$P^{-m} \times P^{-(P^s-1)} \le N \le (1 - P^{-m}) \times P^{(P^s-1)}.$$

Например, при $P = 2$, $m = 22$ и $s = 10$ диапазон чисел простирается примерно от 10^{-300} до 10^{300}. Для сравнения отметим, что количество секунд, которые прошли с момента образования планет Солнечной системы, составляет около 10^{18}.

Следует заметить, что все числа с плавающей запятой хранятся в машине в так называемом *нормализованном* виде.

Нормализованным называют такое число, в старшем разряде мантиссы которого стоит единица (следовательно, для нормализованных двоичных чисел $0{,}5 < |M| < 1$).

Нормализованные, т. е. приведенные к правильной дроби, числа:

$10{,}35_{10} = 0{,}1035_{10} \times 10^2$;

$0{,}00007245_8 = 0{,}7245_8 \times 8^{-4}$;

$F5C{,}9B_{16} = 0{,}F5C9B_{16} \times 16^3$.

Алгебраическое представление двоичных чисел. Для алгебраического представления чисел, т. е. для их представления с учетом знака, в вычислительных машинах используются специальные коды:

- прямой;
- обратный;
- дополнительный.

При этом два последних кода позволяют заменить операцию вычитания на операцию сложения с отрицательным числом. Дополнительный код обеспечивает более быстрое выполнение операций, поэтому в ЭВМ применяется чаще именно он.

Знак числа обычно кодируется двоичной цифрой, при этом:

- код 0 означает знак + (плюс);
- код 1 означает знак – (минус).

Прямой код числа N обозначим $[N]_{пр}$.

Пусть $N = a_1, a_2, a_3, ..., a_m$, тогда:

- при $N > 0$ $[N]_{пр} = 0, a_1, a_2, a_3, ..., a_m$;
- при $N < 0$ $[N]_{пр} = 1, a_1, a_2, a_3, ..., a_m$;
- при $N = 0$ имеет место неоднозначность $[0]_{пр} = 0, 0... = 1, 0...$

Если при сложении в ЭВМ оба слагаемых имеют одинаковый знак, то операция сложения выполняется обычным путем. Если при сложении слагаемые имеют разные знаки, то сначала необходимо выявить бóльшее по абсолютной величине число, из него произвести вычитание меньшего по абсолютной величине числа, и разности присвоить знак большего числа.

Выполнение операций умножения и деления в прямом коде осуществляется обычным образом, но знак результата определяется по совпадению или несовпадению знаков участвовавших в операции чисел.

Операцию вычитания в этом коде нельзя заменить операцией сложения с отрицательным числом, поэтому возникают сложности, связанные с займом значений из старших разрядов уменьшаемого числа. В связи с этим прямой код в ЭВМ почти не применяется.

Обратный код числа N обозначим $[N]_{обр}$.

Пусть $N = a_1, a_2, a_3, ..., a_m$ и $\tilde{a}$ обозначает *инверсию a*, т. е. если $a = 1$, то $\tilde{a} = 0$, и наоборот. Тогда:

- при $N > 0$ $[N]_{обр} = 0, a_1, a_2, a_3, ..., a_m$;
- при $N < 0$ $[N]_{обр} = 1, \tilde{a}_1, \tilde{a}_2, \tilde{a}_3, ..., \tilde{a}_m$;
- при $N = 0$ имеет место неоднозначность $[0]_{обр} = 0{,}00...0 = 1{,}11...1$.

Для того чтобы получить обратный код отрицательного числа, необходимо все цифры этого числа *инвертировать*, т. е. в знаковом разряде поставить 1, во всех значащих разрядах нули заменить единицами, а единицы — нулями.

Например,

для $N = 1011$ $[N]_{обр} = 0{,}1011$;

для $N = -1011$ $[N]_{обр} = 1{,}0100$.

Дополнительный код числа N обозначим $[N]_{доп}$.

Пусть, как и выше, $N = a_1, a_2, a_3, ..., a_m$ и $\tilde{a}$ обозначает величину, обратную a (инверсию a), т. е. если $a = 1$, то $\tilde{a} = 0$, и наоборот. Тогда:

- при $N \geq 0$ $[N]_{доп} = 0, a_1, a_2, a_3, ..., a_m$;
- при $N \leq 0$ $[N]_{доп} = 1, \tilde{a}_1, \tilde{a}_2, \tilde{a}_3, ..., \tilde{a}_m + 0{,}00...1$.

Для того чтобы получить дополнительный код отрицательного числа, необходимо все его цифры инвертировать и затем к младшему разряду прибавить единицу. В случае возникновения переноса из первого после запятой разряда в знаковый разряд к числу следует прибавить единицу в младший разряд.

Например,

для $N = 1011$ $[N]_{доп} = 0{,}1011$;

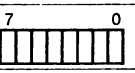
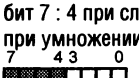
для $N = -1100$ $[N]_{доп} = 1{,}0100$;

для $N = -0000$ $[N]_{доп} = 10{,}0000 = 0{,}0000$ (1 исчезает). Неоднозначности в изображении 0 нет.

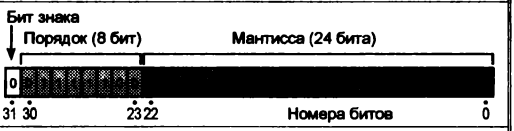
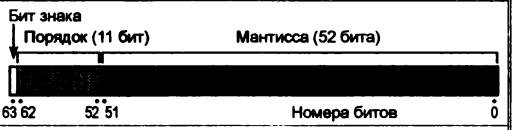
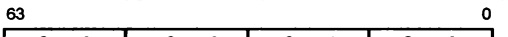
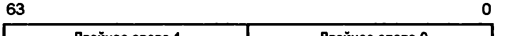
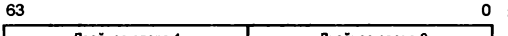
Эмпирическое правило: для получения дополнительного кода отрицательного числа необходимо все символы этого числа инвертировать, кроме последней (младшей) единицы и тех нулей, которые за ней следуют.

Данные, обрабатываемые в современных процессорах. Подводя итоги вышеизложенному, приведем типы данных (табл. 1.9), используемые в процессорах Intel (Pentium) и AMD (K6—K8).

Таблица 1.9. **Примеры некоторых реальных типов данных современных процессоров**

Вид данного	Английский эквивалент (или расшифровка)	Пояснения
Форматы данных, обрабатываемых блоками АЛУ 32-разрядных процессоров		
1. Двоичные и строковые		
Бит	Bit	Минимальная единица информации. Бит в памяти задается базой (адресом слова) и смещением (номером бита в слове)
Битовое поле	Bit Field	Группа до 32 смежных бит, располагающихся не более чем в 4 байтах
Битовая строка	Bit String	Набор смежных бит длиной до 4 Гбит
Байт	Byte 8 бит	
Строки байт, слов и двойных слов	Bit String, Byte String, Word String, Double Word String	Длиной до 4 Гбайт
Числа без знака	Unsigned Byte/ Word/Double Word/ QuadWord	Без знака — байт/слово/двойное/учетверенное слово (8/16/32/64 бит)
Целые числа со знаком	Integer Byte/ Word/Double Word/ QuadWord	Со знаком — байт/слово/двойное/учетверенное слово. Единичное значение самого старшего бита (знак) является признаком отрицательного числа, которое хранится в дополнительном коде
2. Двоично-десятичные числа (BCD — Binary Coded Decimal)		
8-разрядные упакованные	Packed BCD	Содержат два десятичных разряда в одном байте 
8-разрядные неупакованные	Unpacked BCD	Содержат один десятичный разряд в байте (значения бит 7 : 4 при сложении и вычитании несущественно, при умножении и делении они должны быть нулевыми) 

Продолжение табл. 1.9

Вид данного	Английский эквивалент (или расшифровка)	Пояснения
Двоично-десятичные 80-битовые упакованные числа		18 десятичных разрядов и знак
Форматы данных, обрабатываемых блоками ПЗ и векторными (MMX/XMM)		
1. Действительные числа в формате с ПЗ		
Формат IEEE-754, одинарная точность	Single Precision 32 бит — 24 бит мантисса, 8 бит порядок	
Формат IEEE-754, двойная точность	Double Precision 64 бит — 52 бит мантисса, 11 бит порядок	
Повышенная точность	Extended Precision	80 бит — 64 бит мантисса, 15 бит порядок
2. Упакованные целые числа, со знаком и без (64 разряда — MMX)		
Восемь упакованных байт	Packed byte (B)	
Четыре упакованных слова	Packed word (W)	
Два упакованных двойных слова	Packed doubleword (D)	
Учетверенное слово	Quadword (Q)	
3. Упакованные вещественные числа (64 разряда — 3DNow!)		
2 × 32 бита. Формат IEEE-754	Два упакованных двойных слова с ПЗ одинарной точности	
4. Упакованные вещественные и целые числа (128 разрядов — SSE, AMD64)		
Формат SSE	Четыре числа одинарной точности с ПЗ	

Окончание табл. 1.9

Вид данного	Английский эквивалент (или расшифровка)	Пояснения
Формат SSE2	Два двойной точности с ПЗ	
« «	Целое на 128 бит	
« «	Два целых (2 × 64 бита)	
« «	Четыре целых (4 × 32 бита)	
« «	Восемь коротких с (8 × 16 бит)	
Формат SSE2	16 байт (16 × 8 бит)	
Указатели*		
Длинный указатель (48 бит)		16-битовый селектор (или сегмент) и 32-битовое смещение
Короткий указатель		32-битовое смещение
Простой указатель		32 бит, единственный тип указателя для 8086 и 80286. 16-битовый селектор (или сегмент) и 16-битовое смещение

* Указатель (пойнтер, pointer) — данное, содержащее адрес некоторой переменной в оперативной памяти. В языках программирования обычно присутствует функция `PTP(X)`, определяющая адрес переменной `X`.

Разновидности «машинной арифметики»

Обычная двоичная арифметика. Двоичные операции выполняются над двоичными числами в арифметико-логических устройствах (АЛУ) компьютеров. Базовой операцией является с л о ж е н и е, которое непосредственно выполняется электронной схемой сумматора (см. рис. 2.11—2.13). Вычитание также выполняется как сложение, однако вычитаемое (отрицательное число) представлено в дополнительном или другом коде. Умножение реализуется как серия сложений со сдвигами (перемещение цифр на более старшую или младшую позицию), деление — вы-

читания со сдвигами. В каждый момент в операции участвуют только два числа. Вот четыре возможные операции сложения сочетаний двоичных цифр:

Первая цифра	0	0	1	1
Вторая цифра	0	1	0	1
Результат	0	1	1	10

Заметим, что перенос единицы в следующую позицию (в старшую сторону) происходит только в случае сложения двух единиц.

Рассмотрим примеры двоичного суммирования совместно с их десятичными эквивалентами:

1 + 1 = 2	2 + 2 = 4	6 + 3 = 9	10 + 5 = 15	37 + 22 = 59
1	10	110	1010	100101
1	10	+11	+101	+10110
10	100	1001	1111	111011

Вот, далее, четыре возможные операции вычитания двоичных цифр:

Первая цифра	0	1(0)	1	1
Вторая цифра	0	1	0	1
Результат	0	1	1	0

Отметим, что заем единицы в старшем разряде требуется только тогда, когда единица вычитается из нуля.

Рассмотрим примеры двоичного вычитания совместно с их десятичными эквивалентами:

2 – 1 = 1	5 – 3 = 2	10 – 2 = 8	15 – 9 = 6	43 – 18 = 25
10	101	1010	1111	101011
–1	–10	–10	1001	–10010
10	100	1001	1111	11001

Следует обратить внимание на аналогию в правилах выполнения арифметических действий в двоичной и десятичных системах счисления: если при сложении двух двоичных чисел (точнее, представленных в двоичной системе счисления) сумма

цифр окажется больше единицы, то возникает перенос в старший разряд; если уменьшаемая цифра меньше вычитаемой, то нужно сделать заем единицы в старшем разряде.

Как уже известно, дополнительный код используется для вычитания чисел в компьютерах и позволяет эту операцию свести к сложению чисел. Правила выполнения вычитания с дополнительным числом следующие. Чтобы число *A* вычесть из числа *B*, достаточно сложить *B* с дополнительным числом к *A* и отбросить перенос в соседний старший разряд. Например, чтобы вычесть 623 из 842, достаточно сложить 842 с 377 и, отбросив перенос, получим 219 (842 – 623 = 219).

Таким образом, важнейшее преимущество двоичной арифметики заключается в том, что она позволяет все арифметические действия свести к одному — сложению, а это значительно упрощает устройство процессора ЭВМ.

Двоично-десятичная арифметика. В этом случае для записи одного десятичного разряда используется четыре двоичных бита (тетрада). Иногда встречается название, пришедшее из англоязычной литературы — нибл. С помощью четырех битов можно закодировать шестнадцать цифр. Лишние комбинации в двоично-десятичном коде являются запрещенными. Соответствие двоично-десятичного кода и десятичных цифр приведено в табл. 1.8.

Суммирование двоично-десятичных чисел можно производить по правилам обычной двоичной арифметики, а затем выполнять двоично-десятичную коррекцию. При этом осуществляется проверка каждой тетрады на допустимые коды и если в какой-либо тетраде обнаруживается запрещенная комбинация, это — наличие переполнения и необходимость произвести двоично-десятичную коррекцию.

Двоично-десятичная коррекция заключается в дополнительном суммировании числа `0110 – шесть` (число запрещенных комбинаций) с тетрадой, в которой произошло переполнение или перенос в старшую тетраду. Вот два примера:

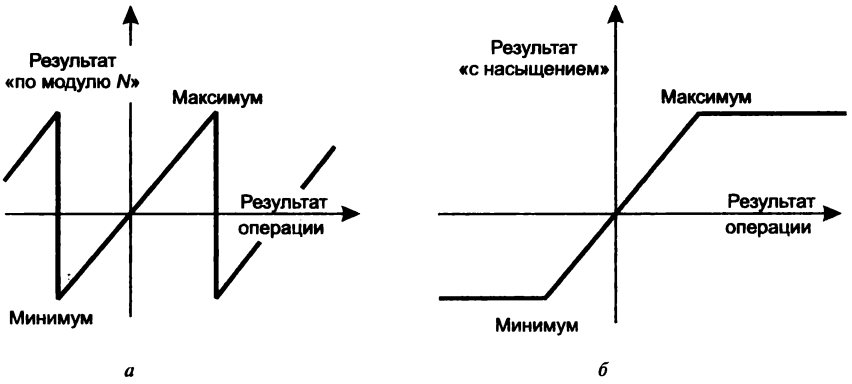
1) 18 + 13 = 31		2) 19 + 19 = 38	
Сложение тетрад	Коррекция	Сложение тетрад	Коррекция
+ 0001 1000 0001 0011 0010 1011	+ 0010 1011 0000 0110 0011 0001	+ 0001 1001 0001 1001 0011 0010	+ 0011 0010 0000 0110 0011 1000

Циклическая арифметика (WrapAround). Если при сложении двух чисел (например, в 8-разрядном сумматоре) возникает единица переполнения:

$$\begin{array}{r} 11111110_2 \\ +\ 00000011_2 \\ \hline 100000001_2 \\ \hline \varnothing 00000001_2 \end{array},$$

которая не помещается в разрядной сетке (в данном примере — 8 бит), то происходит «сбрасывание» этого промежуточного результата до остатка от его деления (в данном примере до 1) на максимальное число в разрядной сетке (здесь — $2^8 = 256$). Просьба к читателю перевести оба операнда в десятичную форму и удостовериться в истинности данного утверждения.

Такой эффект возникает в «циклической» (или «модульной») арифметике — сложение, умножение и пр. по модулю N (mod N), когда результат образуется путем получения остатка от деления «истинного» результата операции на основание модуля — число N (на рис 1.5, *а* N = «Максимум» — «Минимум»).



а *б*

Рис. 1.5. Диаграммы к определению арифметики циклической (*а*) и с насыщением (*б*)

Saturation arithmetic — SArith («арифметика насыщения») — версия арифметики, в которой результат операции ограничен интервалом между максимальным и минимальным значениями. Если результат больше максимума, он устанавливается («clamped») в максимум, если меньше минимума — то в мини-

мум. Дальнейшее суммирование с максимумом или вычитание из минимума не меняет результата (рис. 1.5, *б*).

Например, если интервал допустимых значений находится между –100 и +100, то следующие операции дают такие результаты (см. также табл. 1.10):

$60 + 43 = 100$; $(60 + 43) - 150 = -50$; $43 - 150 = -100$;
$60 + (43 - 150) = -40$;
$10 \times 11 = 100$; $99 \times 99 = 100$; $30 \times (5 - 1) = 100$;
$30 \times 5 - 30 \times 1 = 70$.

Таблица 1.10. **Примеры пределов для арифметики с насыщением**

Размер данного	Десятичное представление		Шестнадцатеричное	
	Нижний предел	Верхний предел	Нижний предел	Верхний предел
Байт со знаком	-128	+127	80_{16}	$7F_{16}$
Байт без знака	0	255	0	FF_{16}
Слово со знаком	-32768	+32767	8000_{16}	$7FFF_{16}$
Слово без знака	0	65535	0	$FFFF_{16}$

Векторные операции над упакованными данными («упакованная арифметика» — packed arithmetic). Принцип упакованных данных создает предпосылки для их параллельной обработки (архитектуры MMX, 3DNow!, SSE). Возьмем в качестве примера единицу звуковой информации — 16-разрядный элемент трека CD (см. рис. 1.1). Четыре элемента можно разместить в 64-разрядном регистре процессора. При выполнении «упакованной» команды из регистров *i* и *j* параллельно выбирается по элементу, над ними одновременно выполняется арифметическая или логическая операция (например, «+»), и результат записывается в регистр *k* (рис. 1.6, *а* — MMX-команда `PADDUS [W]`, упакованное сложение беззнаковых слов с насыщением).

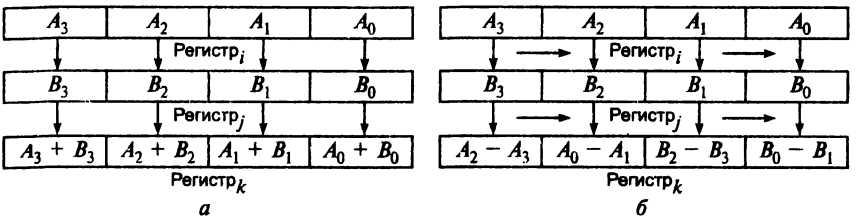


Рис. 1.6. Примеры векторных операций:
а — команда `PADDUS`; *б* — `HSUBPS`

«Горизонтальная» и «вертикальная» и арифметика. В то время как команды обычной векторной арифметики можно было бы отнести к более или менее строго вертикальным (между регистрами) операциям (рис. 1.6, *а*), в архитектуре SSE3 была введена возможность работать горизонтально (внутри регистра). Были добавлены команды сложения/вычитания по отношению к наборам (векторам) значений, расположенных в пределах единственного регистра, которые улучшают эффективность цифровой обработки сигналов и трехмерных преобразований (рис. 1.6, *б* — SSE3-команда HSUBPS — Horizontal-Subtract-Packed-Single — горизонтальное вычитание упакованных слов одинарной длины). Например, при обработке звукового сигнала команду HASUBPS можно использовать для получения относительного кодирования (см. рис. 1.2, *б*) из абсолютного (см. рис. 1.2, *а*).

Необходимо заметить, что наряду с рассмотренными здесь арифметическими операциями, в системах команд компьютеров присутствуют также и логические операции (описываются ниже), в частности, операции на рис. 1.6 могли бы быть поразрядными логическими (см. табл. 1.21).

Двоичное кодирование мультимедиа-информации

С 80-х гг. быстро развивается технология обработки на компьютерах графической информации. Компьютерная графика широко используется в моделировании при научных исследованиях, в компьютерных тренажерах, компьютерной анимации, деловой графике, играх и т. д.

В дальнейшем в связи с быстрым возрастанием аппаратных возможностей персональных компьютеров массовый пользователь получил возможность обрабатывать видеоинформацию.

Графическая информация на экране дисплея представляется в виде изображения, которое формируется из точек (пикселей). В современных компьютерах разрешающая способность (количество точек на экране дисплея), а также количество цветов зависит от видеоадаптера и может меняться программно.

Цветные изображения могут представляться в различных режимах: 16 цветов, 256 цветов, 65 536 цветов (high color), 16 777 216 цветов (true color) — табл. 1.11. Очевидно, что количество бит на точку (пиксель), например, режима true color равно:

$I = \log_2 65\,536 = 16$ бит = 2 байта.

Таблица 1.11. **Характеристики различных стандартов представления графики**

Разрешение	16 цветов	256 цветов	65 536 цветов	16 777 216 цветов
640 × 480	150 Кбайт	300 Кбайт	600 Кбайт	900 Кбайт
800 × 600	234,4 Кбайт	468,8 Кбайт	937,5 Кбайт	1,4 Мбайт
1024 × 768	384 Кбайт	768 Кбайт	1,5 Мбайт	2,25 Мбайт
1280 × 1024	640 Кбайт	1,25 Мбайт	2,5 Мбайт	3,75 Мбайт

Наиболее распространенной разрешающей способностью экрана является разрешение 800 на 600 точек, т. е. 480 000 точек.

Рассчитаем необходимый для режима true color объем видеопамяти:

V = 2 байта × 480 000 = 960 000 байт = 937,5 Кбайт.

Аналогично рассчитывается объем видеопамяти, необходимый для хранения битовой карты изображений при других видеорежимах.

В видеопамяти памяти компьютера хранится битовый план (bit map), являющийся двоичным кодом изображения, отсюда он считывается процессором (не реже 50 раз в секунду) и отображается на экран.

Двоичное кодирование звуковой информации. С начала 90-х гг. персональные компьютеры получают широкие возможности для работы со звуковой информацией. Каждый компьютер, имеющий звуковую плату, может сохранять звук в виде файлов и воспроизводить его. С помощью специальных программных средств (редакторов аудиофайлов) открываются широкие возможности по созданию, редактированию и прослушиванию звуковых файлов. В дальнейшем создаются программы распознавания речи и появляется возможность голосового управления компьютером.

Различные звуковые карты могут обеспечить как 8-, так и 16-битовые выборки. При замене непрерывного звукового сигнала его дискретным представлением в виде ступенек 8-битовые карты позволяют закодировать 256 различных уровней дискретизации звукового сигнала, соответственно 16-битовые — 65 536 уровней.

Частота дискретизации аналогового звукового сигнала (количество выборок в секунду) также может принимать различные значения (5,5, 11, 22 и 44 кГц). Таким образом, качество звука в дискретной форме может быть очень плохим (качество радиотрансляции) при 8 битах и 5,5 кГц и весьма высоким (качество аудиоCD) при 16 битах и 44 кГц.

Можно оценить объем моноаудиофайла с длительностью звучания 1 с при среднем качестве звука (16 бит, 22 кГц). Для этого 16 бит на одну выборку необходимо умножить на 22 000 выборок в секунду, что дает в результате 43 Кбайт.

Сжатие информации. Объемы обрабатываемой и передаваемой информации быстро возрастают. Это связано с появлением все более сложных прикладных процессов, развитием новых информационных служб, использованием изображений и звука.

Сжатие данных (data compression) — процесс, обеспечивающий уменьшение объема данных. Сжатие позволяет резко уменьшить объем памяти, необходимой для хранения данных, сократить (до приемлемых размеров) время их передачи. Особенно эффективно сжатие изображений. Сжатие данных может осуществляться как программным, так и аппаратным или комбинированным методами.

Сжатие текстов связано с более компактным расположением байтов, кодирующих символы. Определенные результаты дает статистическое кодирование, в котором наиболее часто встречающиеся символы имеют коды наименьшей длины. Здесь также используется счетчик повторений пробелов. Что же касается звука и изображений, то объем представляющей их информации зависит от выбранного шага квантования и числа разрядов аналого-цифрового преобразования. В принципе, здесь используются те же методы сжатия, что и при обработке текстов. Если сжатие текстов происходит без потери информации, то сжатие звука и изображения почти всегда приводит к ее некоторой потере. Сжатие широко используется при архивировании данных.

Сжатие изображений (image compression) — процесс минимизации данных, составляющих изображение. Минимизация количества информации, предоставляющей изображение или видеофильм прежде всего, осуществляется при выборе шага квантования и разрядности кодов. При этом, естественно, происходит определенная (допустимая) потеря информации. Затем происходит сжатие изображения, представленного дискретным сигналом.

Сжатие изображения осуществляется в несколько этапов:

- изображение делится на блоки пикселей, каждый из которых подвергается обработке, устраняющей избыточность;
- осуществляется кодирования с переменной длиной кодов, что исключает длинные цепочки нулей и единиц в последовательностях битов;

- дополнительное сжатие движущегося изображения за счет сравнения каждого изображения с предыдущим, чтобы сохранять только изменившуюся его часть.

Допускается потеря той информации, которая в решении поставленной задачи считается несущественной. Например, можно при обработке изображений удалить из аналогового сигнала частоты, которые находятся вне спектра, воспринимаемого глазом человека (до 10 000 цветов, 256 оттенков серого цвета). Нередко допускается игнорирование цвета каждого второго пикселя либо группа пикселей заменяется одним со средним значением цвета. Осуществляется также групповое кодирование. Его сущность заключается в кодировании групп одинаковых пикселей (например, небо без облаков на картине).

Размер файла сжатого дискретного неподвижного изображения зависит от четырех параметров: площади изображения, квадрата разрешения, числа бит, необходимых для представления пикселя и коэффициента сжатия. В видеофильме к этому еще добавляется число образующих его неподвижных изображений. Выбор коэффициентов сжатия — компромисс между пропускной способностью системы (скоростью переноса файлов) и качеством восстанавливаемого изображения. Чем выше коэффициент сжатия, тем ниже это качество. При этом следует иметь в виду, что при очень высокой разрешающей способности и большом коэффициенте сжатия можно получить изображение с низкой разрешающей способностью. Поэтому выбор указанных параметров обосновывается технико-экономическим анализом и алгоритмом сжатия. Что касается качества изображения, то оно зависит от конкретной поставленной задачи. Например, в системах телеконференций основной объем необходимой информации содержится в речи, тогда как качество изображения может играть вторую роль.

В зависимости от скорости сжатия изображений выполняемые процессы подразделяются на два класса. К первому относится сжатие неподвижных изображений, которое может выполняться в фоновом режиме, с любой возможной скоростью. Второй класс образуют алгоритмы сжатия движущихся изображений, которые должны выполняться в реальном времени по мере получения данных.

Существует немало технологий сжатия/восстановления изображений. Наиболее популярная из них предложена *Объединенной группой экспертов в области фотографии (JPEG)* и позволяет

сократить размеры графического файла в 10—20 раз. Благодаря специальным процессорам и алгоритмам удается также сжимать видеосюжеты.

Кодирование видеоинформации. В связи с большим объемом информации, содержащейся в видеопотоке (до 6 Мбайт/с), для записи информации в ЭВМ обычно применяют кодирование со сжатием потока данных на входе с использованием алгоритмов семейства MPEG/JPEG (табл. 1.12).

Таблица 1.12. **Характеристики представления видеоинформации в различных форматах**

№	Формат	Тип данных (размер изображения)	Длительность записи (CD/DVD), мин
1	VCD	288 × 384	63
2	S-VCD	480 × 576	32
3	DVD	576 × 720	59
4	VHS	288 × 384	—
5	S-VHS	540 × 720	—
6	Internet High Speed	193 × 144	—

Стандарт MPEG (Motion Picture Expert Group) включает несколько компонентов: системного потока, описывающего структуру смешанного аудио- и видеопотока, а также MPEG-video и MPEG-audio.

В случае MPEG-video сжатие достигается за счет четырех факторов.

1. Использование составляющих YUV вместо обычных RGB (красный, зеленый, синий).

Вместо элементарных цветов кодируется яркость (luminance, Y) и цветность (chrominance, U & V), причем цветность «прорежена» по вертикали и горизонтали в 2 раза по сравнению с яркостью (децимация). При этом вместо сильно коррелированных сигналов RGB получаются практически некоррелированные YUV, и за счет децимации достигается двукратное сжатие.

2. Дискретно-косинусное преобразование с последующим квантованием.

При этом квадраты пикселей (8 × 8) подвергаются двумерному дискретно-косинусному преобразованию (DCT), которое родственно преобразованию Фурье, различие заключается в наборе базисных функций (в преобразовании Фурье — это синусы

и косинусы, в DCT — косинусы). Это преобразование переводит пространственное представление сигнала в частотное. Результат преобразования подвергается квантованию, т. е. огрублению точности, при этом коэффициент квантования для более высоких пространственных частот выбирается более высоким, чем для низких, с учетом особенностей восприятия. При этом высокие пространственные частоты передаются с меньшей точностью, чем низкие частоты. При квантовании многие пространственные частоты не кодируются и не передаются.

3. Устранение временной избыточности с компенсацией движения.

Это означает, что для ликвидации избыточности, заключающейся в большой корреляции между соседними кадрами, передается разность между ними. Кадры видеопотока разбиваются на несколько типов — *Intra* (*I*), которые кодируются полностью, *Predicted* (*P*), для которых кодируется различие с предыдущим *I*- или *P*-кадром, и *Bidirectional* (*B*), для которых в качестве *опорных* (*reference*) используются *I*- и/или *P*-кадры, между которыми он находится. Обычно *I*-кадры следуют 1 или 2 раза в секунду, и между двумя опорными кадрами лежат 2—4 B-кадра. Типичная последовательность кадров (GOP — Group of Pictures) имеет вид: ***IBBPBBPBBPBBIBBP***. В общем случае вид последовательности выбирается кодером и может зависеть или нет от содержания кадров. Поскольку изображение на соседних кадрах обычно сдвинуто, применяется компенсация движения, т. е. кодируется отклонение («разность») от некоторого сдвинутого опорного изображения. Кодирование выполняется макроблоками (16 × 16 — яркость, 8 × 8 — цветность), для каждого макроблока находится свой вектор движения.

4. Квазиоптимальное кодирование.

Коэффициенты, полученные после DCT, векторы движения и все остальное кодируются кодами переменной длины. Это кодирование называют ***квазиоптимальным***, поскольку кодовая таблица не строится заново для каждого конкретного случая, а выбрана при разработке стандарта на основе анализа типичных видеопоследовательностей.

MPEG-1 проектировался из расчета на поток 120 Кбайт/с размером 288 × 354 при 25 кадрах в секунду, хотя он не ограничен этим и допускает существенно больший поток при произвольном размере кадра.

MPEG-2 проектировался с учетом опыта использования MPEG-1 и ориентируется на вещание, так как содержит средства для маскирования ошибок.

В случае MPEG-audio исходный сигнал подвергается многоканальной фильтрации. Далее амплитуды сигналов в каждой полосе сравниваются для нахождения полос, подлежащих кодированию с учетом эффекта маскирования слабого сигнала сильным. Далее амплитуда сигнала в полосе квантуется и кодируется.

Типы и структуры данных

Типы данных (табл. 1.13). Классификация информационных единиц, обрабатываемых на ЭВМ, включает следующие аспекты:

- типы данных, или совокупность соглашений о программно-аппаратурной форме представления и обработки, а также ввода, контроля и вывода элементарных данных;
- структуры данных — способы композиции простых данных в агрегаты и операции над ними;
- форматы файлов — представление информации на уровне взаимодействия операционной системы с прикладными программами.

Таблица 1.13. **Типы и структуры данных в некоторых системах программирования и управления данными**

	Характеристика информации	Система — язык программирования, СУБД, ИПС						
		Algol	Cobol	PL/1	FoxBase/ Clipper	Adabas/ Natural	Oracle/ SQL	STAIRS, IRBIS, ISIS
Тип данных	Целое короткое (2 байта)	—	—	—	—	—	Smallint	—
	Целое нормальное (4 байта)	Integer	Computational	Int	N(x)	N(x)	Int	—
	Целое длинное (8 байт)	—	—	Double	—	—	—	—
	Действительное нормальное (4 байта)	Real	Computational	Float	N(x.y)	N(x.y)	Float Real	—
	Действительное двойное (8 байт)	—	—	—	—	—	Float Double	—
	Двоичное	—	—	Binary	—	B(x)	—	—

Окончание табл. 1.13

	Характеристика информации	Система — язык программирования, СУБД, ИПС						
		Algol	Cobol	PL/1	FoxBase/ Clipper	Adabas/ Natural	Oracle/ SQL	STAIRS, IRBIS, ISIS
Типы данных	Десятичное упакованное (2 цифры на байт)	—	PIC(9)	Decimal	—	P(x)	—	—
	Десятичное распакованное (1 цифра на байт)	—	PIC(X)	—	N(x)	U(x)	—	—
	Логическое	Boolean	—	+	Logical	—	—	—
	Символьное	—	PIC(A)	Char	C(x)	A(x)	Char	+
	Длинный текстовый или бинарный объект (BLOB)	—	—	—	Memo	—	VarGrafic VarChar	—
	Дата	—	—	—	Date	—	Date	—
	Время	—	—	—	—	—	Time	—
Структуры	Массивы	Array	—	Dim	Dimen-tion	VAR(n)	—	—
	Записи (структуры)	—	+	+	+	+	+	—
	Множественные (векторные) поля записи	—	—	—	—	MU	—	+
	Групповые поля записи	—	+	+	—	GR	—	+
	Повторяющиеся группы в записи	—	—	—	—	PE	—	—
	Текстовые поля (параграфы, предложения, слова)	—	—	—	—	—	—	+

Ранние языки программирования (ЯП), а точнее, системы программирования (СП) — Fortran, Algol, будучи ориентированы исключительно на вычисления, не содержали развитых систем типов и структур данных.

В ЯП Algol символьные величины и переменные вообще не предусматривались, в некоторых реализациях строки (символы

в апострофах) могли встречаться только в операторах печати данных.

Типы числовых данных ЯП Algol: `Integer` (целое число), `Real` (действительное) — различаются диапазонами изменения, внутренними представлениями и применяемыми командами процессора ЭВМ (соответственно арифметика с фиксированной и плавающей точкой). Нечисловые данные представлены типом `BOOLEAN` — логические, имеющие диапазон значений {`true`, `false`}.

Позже появившиеся ЯП (СП) Cobol, PL/1, Pascal вводят новые типы данных:

- символьные (цифры, буквы, знаки препинания и пр.);
- числовые символьные для вывода;
- числовые двоичные для вычислений;
- числовые десятичные (цифры 0--9) для вывода и вычислений.

Разновидности числовых данных здесь соответствуют внутреннему представлению и машинным (или эмулируемым) командам обработки. Кроме того, вводятся числа двойного формата (2 машинных слова), для обработки которых также необходимо наличие в процессоре (или эмуляция) команд обработки чисел двойной длины (точности).

Уместно привести пример представления числовой информации в различных перечисленных формах. Пусть задано число $135_{10} = 207_8 = 87_{16} = 100000111_2$, тогда:

- внутренняя стандартная форма представления (тип `Binary` для обработки в двоичной арифметике) сохраняется (100000111_2). Объем — 1 байт, или 8 двоичных разрядов;
- внутренняя форма двоично-десятичного представления (тип `Decimal`, каждый разряд десятичного числа представляется двоично-десятичной, в 4 бита, комбинацией). Представление 135 есть $001\ 011\ 101_2$. Объем — 2,5 байта, 12 двоичных разрядов;
- символьное представление (тип `Alphabetic`, для вывода) — каждый разряд представляется байтом в соответствии с кодом ASCII (табл. приложения 2). Представление 135 есть — $00110001\ 00110011\ 00110101_2$. Объем — 3 байта.

Некоторые системы программирования (Fortran IV, например) поддерживают операции над комплексными числами вида $Z = A + Bi$ (где A, B — действительные коэффициенты, i — мнимая единица). Очевидно, для размещения таких чисел необхо-

дим как минимум двойной расход оперативной памяти (по одному слову для размещения действительной и мнимой частей при обычной точности и по два слова при двойной точности). Кроме того, очевидно, что процессоры обычных универсальных ЭВМ вряд ли поддерживают операции над такими числами, в связи с этим операции над ними требуют написания соответствующих подпрограмм *или эмуляции комплексной арифметики.*

Появление систем управления базами данных и систем программирования для разработки ИС приводит к появлению ряда других типов данных:

- *дата и время*;
- *бинарные* (Binary Large Object — BLOB) и *текстовые объекты* без внутренней структуры (интерпретация возлагается на прикладные программы).

Понятие типа данных ассоциируется также с допустимыми значениями переменной и операциями над ними, например, данные типа время (`ЧЧ:ММ:СС`) или дата (`ГГ/ММ/ДД`) предполагают определенные диапазоны значений каждого из разрядов, а также машинные или эмулируемые операции (сложение/вычитание дат и/или моментов времени). Основной причиной «проблемы 2000 г.» («Problem Y2K») являлась не столько двухразрядная запись года в базах данных, сколько встроенные в огромное количество программ (часто недокументированных) операции над данными типа `Date – ГГ/ММ/ДД`.

Структуры данных. В языке Algol были определены два типа структур: элементарные данные и массивы (векторы, матрицы, тензоры, состоящие из арифметических или логических переменных). Основным нововведением, появившимся первоначально в Cobol, (затем PL/1, Pascal и пр.) являются агрегаты данных (структуры, записи), представляющие собой именованные комплексы переменных разного типа, описывающих некоторый объект или образующих некоторый достаточно сложный документ.

Рассмотренные выше экзотические типы данных (комплексные числа), очевидно, занимают промежуточное положение между элементарными переменными и массивами (структурами).

Термин *запись* подразумевает наличие множества аналогичных по структуре агрегатов, образующих *файл* (картотеку), содержащих данные по совокупности однородных объектов, элементы данных образуют поля, среди которых выделяются элементарные и групповые (агрегатные).

Появление СУБД и АИПС приводит к появлению новых разновидностей структур:

- множественные поля данных;
- периодические групповые поля;
- текстовые объекты (документы), имеющие иерархическую структуру (документ, сегмент, предложение, слово).

1.3. Логические основы и элементы ЭВМ

Начало исследований в области формальной логики было положено работами Аристотеля в IV в. до нашей эры. Однако строго формализованный подход к проблеме впервые был предложен Дж. Булем. В честь него алгебру высказывания называют булевой (булевской) алгеброй, а логические значения — булевыми (булевскими). Основу математической логики составляет алгебра высказываний. Алгебра логики используется при построении основных узлов ЭВМ (дешифратор, сумматор, шифратор).

Алгебра логики оперирует с высказываниями. Под высказыванием понимают повествовательное предложение, относительно которого можно утверждать, истинно оно или ложно. Например, выражение «Расстояние от Москвы до Киева больше, чем от Москвы до Тулы» истинно, а выражение «$5 < 2$» — ложно.

Высказывания (логические переменные) принято обозначать буквами латинского алфавита (иногда — с индексами): A, B, C, ..., X, Y, a, b, c, ..., x, y, z, (x_1, x_2, ..., x_i, ...) и т. д. Если высказывание C истинно, это обозначается как $C = 1$ ($C =$ `t`, `true`), а если оно ложно, то $C = 0$ ($C =$ `f`, `false`).

Логические операции и базовые элементы компьютера

В алгебре высказываний над высказываниями можно производить определенные логические операции, в результате которых получаются новые высказывания. Истинность результирующих высказываний зависит от истинности исходных и использованных для их преобразования логических операций.

Схемные элементы ЭВМ. Преобразование информации в ЭВМ осуществляется элементами (схемами) двух классов:

- комбинационными;
- последовательностными (схемами с памятью).

Состояние выходов комбинационных схем однозначно определяется состояниями входов в данный момент времени. Состояние выходов в последовательностных схемах определяется не только состоянием входов, но и внутренними состояниями, имевшими место в предыдущие моменты времени.

Комбинационные схемы являются техническим аналогом булевых функций. Подобно тому, как сложная булева функция может быть получена суперпозицией более простых функций, так и комбинационная схема может строиться из более простых схем.

Существует следующее определение — систему логических элементов, с помощью которых путем суперпозиции можно представить любую сколь угодно сложную комбинационную схему, называют *функционально полной*. Известны различные функционально полные системы элементов, но наибольшее распространение получили системы, использующие логические операции, выражаемые предлогами НЕ, И, ИЛИ.

Логический элемент компьютера — это часть электронной схемы, которая реализует элементарную логическую функцию. Логическими элементами компьютеров являются электронные схемы «И», «ИЛИ», «НЕ», «И-НЕ», «ИЛИ-НЕ» или другие (называемые также *вентилями*), а также *триггер*. Можно показать, что с помощью этих схем можно реализовать любую логическую функцию, описывающую работу устройств компьютера. Обычно у вентилей бывает от двух до восьми входов и один или два выхода.

Каждый логический элемент имеет свое условное обозначение, которое выражает его логическую функцию, но не указывает на то, какая именно электронная схема в нем реализована. Работу логических элементов описывают с помощью таблиц истинности.

Рассмотрим логические операции и соответствующие им элементы логических схем.

Конъюнкция. Соединение двух (или нескольких) высказываний в одно с помощью союза И (OR) называется операцией логического умножения, или конъюнкцией. Эту операцию принято обозначать знаками «∧, &» или знаком умножения «×». Сложное высказывание $A \wedge B$ истинно только в том случае, когда истинны оба входящих в него высказывания. Истинность такого высказывания задается табл. 1.14.

Логическая схема «И» реализует конъюнкцию двух или более логических значений. Условное обозначение на структур-

Таблица 1.14. **Таблицы истинности конъюнкции и логической суммы высказываний**

Конъюнкция			Дизъюнкция			
A	*B*	$A \wedge B$	*A*	*B*	$A \vee B$	*A* хог *B*
0	0	0	0	0	0	0
0	1	0	0	1	1	1
1	0	0	1	0	1	1
1	1	1	1	1	1	0

ных диаграммах схемы «И» с двумя входами представлено на рис. 1.7, *а*.

Единица на выходе схемы «И» будет тогда и только тогда, когда на всех входах будут единицы. Когда хотя бы на одном входе будет нуль, на выходе также будет нуль.

Связь между выходом z этой схемы и входами x и y описывается соотношением $z = x \,\&\, y$ (читается как «x и y»). Операция конъюнкции на структурных схемах обозначается знаком «&».

Дизъюнкция. Объединение двух (или нескольких) высказываний с помощью союза ИЛИ (OR) называется операцией логического сложения, или дизъюнкцией. Эту операцию обозначают знаками «|, $\vee$» или знаком сложения «+». Сложное высказывание $A \vee B$ истинно, если истинно хотя бы одно из входящих в него высказываний (см. табл. 1.14).

В последнем столбце табл. 1.14 размещены результаты модифицированной операции ИЛИ — Исключающее ИЛИ (XOR). Отличается от обычного ИЛИ последней строкой (см. также рис. 1.7, *е*).

Схема «ИЛИ» реализует дизъюнкцию двух или более логических значений. Когда хотя бы на одном входе схемы «ИЛИ» будет единица, на ее выходе также будет единица.

Условное обозначение на структурных схемах схемы «ИЛИ» с двумя входами представлено на рис. 1.7, *б*. Знак «1» на схеме происходит от классического обозначения дизъюнкции как «≥» (т. е. значение дизъюнкции равно единице, если сумма значений операндов больше или равна 1). Связь между выходом z этой схемы и входами x и y описывается соотношением $z = x \vee y$ (читается как «x ИЛИ y»).

Инверсия. Присоединение частицы НЕ (NOT) к некоторому высказыванию называется операцией отрицания (инверсии) и обозначается $\overline{A}$ (или $\neg A$). Если высказывание A истинно, то B ложно, и наоборот (табл. 1.15).

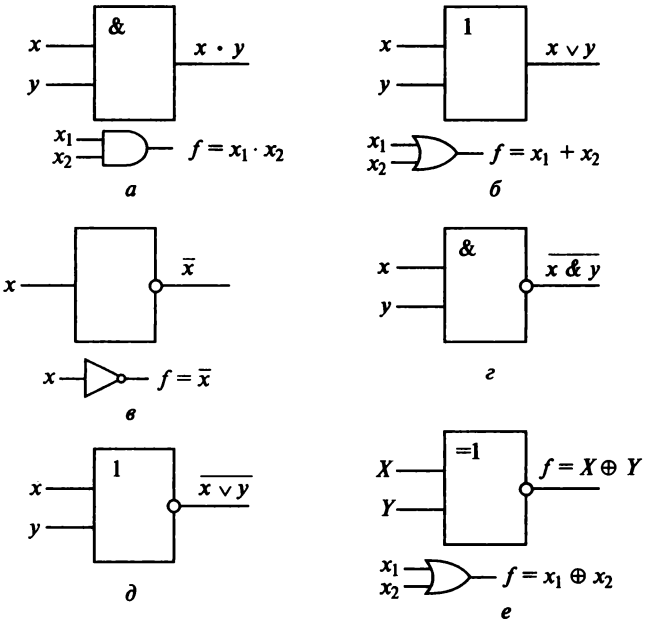


Рис. 1.7. Схемные логические элементы вычислительных машин: *а* — «И»; *б* — «ИЛИ»; *в* — «НЕ»; *г* — «И-НЕ»; *д* — «ИЛИ-НЕ»; *е* — «Исключающее ИЛИ» (сложение по модулю 2)

Таблица 1.15. **Таблица истинности отрицания**

A	$\bar{A}$
0	1
1	0

Схема «НЕ» (инвертор) реализует операцию отрицания. Связь между входом x этой схемы и выходом z можно записать соотношением $z = \bar{x}$, где $\bar{x}$ читается как «НЕ x» или «Инверсия x».

Если на входе схемы «0», то на выходе «1», и наоборот. Условное обозначение на структурных схемах инвертора — на рис. 1.7, *в*.

Вентили. Кроме схемных элементов, соответствующих перечисленным логическим операторам, в состав логических схем входят комбинированные связки, именуемые *вентилями,* например, следующие.

Схема «И-НЕ» состоит из элемента «И» и инвертора и осуществляет отрицание результата схемы «И» (табл. 1.16). Связь между выходом z и входами x и y схемы записывают как $\overline{x \wedge y}$, или «Инверсия x И y». Условное обозначение на структурных схемах схемы «И-НЕ» с двумя входами приведено на рис. 1.7, *г*.

Таблица 1.16. **Таблица истинности схем «И-НЕ», «ИЛИ-НЕ»**

Инверсия x И y			Инверсия x ИЛИ y		
x	y	$\overline{x \wedge y}$	x	y	$\overline{x \vee y}$
0	0	1	0	0	1
0	1	1	0	1	0
1	0	1	1	0	0
1	1	0	1	1	0

Схема «ИЛИ-НЕ» состоит из элемента «ИЛИ» и инвертора и осуществляет отрицание результата схемы «ИЛИ» (табл. 1.16). Связь между выходом z и входами x и y схемы записывают как $\overline{x \vee y}$, или «Инверсия x ИЛИ y». Условное обозначение на структурных схемах схемы «ИЛИ-НЕ» с двумя входами представлено на рис. 1.7, *д*.

Схема «Исключающее ИЛИ» (рис. 1.7, *е*) соответствует «сложению по модулю два» (см. также табл. 1.14).

Следует отметить, что помимо операций И, ИЛИ, НЕ в алгебре высказываний существует ряд и других операций. Например, операция эквивалентности (эквиваленции) $A \sim B$ ($A \equiv B$, или A eqv B) (табл. 1.17).

Таблица 1.17. **Таблицы истинности операций эквивалентности и импликации**

Эквивалентность			Импликация		
A	B	$A \sim B$	A	B	$A \to B$
0	0	1	0	0	1
0	1	0	0	1	1
1	0	0	1	0	0
1	1	1	1	1	1

Другим примером может служить логическая операция импликации или логического следования ($A \to B$, A IMP B), иначе говоря, «ЕСЛИ A, то B» (табл. 1.17).

Высказывания, образованные с помощью логических операций, называются сложными. Истинность сложных высказываний можно установить, используя таблицы истинности. Например, истинность сложного высказывания $\overline{A} \wedge \overline{B}$ определяется табл. 1.18.

Таблица 1.18. **Таблица истинности высказывания $\overline{A} \wedge \overline{B}$**

A	B	$\overline{A}$	$\overline{B}$	$\overline{A} \wedge \overline{B}$
0	0	1	1	1
0	1	1	0	0
1	0	0	1	0
1	1	0	0	0

Высказывания, у которых таблицы истинности совпадают, называются *равносильными*. Для обозначения равносильных высказываний используют знак «=» ($A = B$). Рассмотрим сложное высказывание $(A \wedge B) \vee (\overline{A} \wedge \overline{B})$ — табл. 1.19.

Таблица 1.19. **Таблица истинности выражения $(A \wedge B) \vee (\overline{A} \wedge \overline{B})$**

A	$\overline{A}$	B	$\overline{B}$	$A \wedge B$	$\overline{A} \wedge \overline{B}$	$(A \wedge B) \vee (\overline{A} \wedge \overline{B})$
0	1	0	1	0	1	1
0	1	1	0	0	0	0
1	0	0	1	0	0	0
1	0	1	0	1	0	1

Если сравнить эту таблицу с таблицей истинности операции эквивалентности высказываний A и B (см. табл. 1.17), то можно увидеть, что высказывания $(A \wedge B) \vee (\overline{A} \wedge \overline{B})$ и $A \sim B$ тождественны, т. е. $(A \sim B) = (A \wedge B) \vee (\overline{A} \wedge \overline{B})$.

В алгебре высказываний можно проводить тождественные преобразования, заменяя одни высказывания равносильными им другими высказываниями.

Свойства операций. Исходя из определений дизъюнкции, конъюнкции и отрицания, устанавливаются свойства этих операций и взаимные распределительные свойства. Приведем примеры некоторых из этих свойств:

- коммутативность (перестановочность):
  $A \wedge B = B \wedge A$,
  $A \vee B = B \vee A$;

- закон идемпотентности:
  $A \wedge A = A$, $A \vee A = A$;
- двойное отрицание:
  $\overline{\overline{A}} = A$;
- сочетательные (ассоциативные) законы:
  $A \vee (B \vee C) = (A \vee B) \vee C = A \vee B \vee C$,
  $A \wedge (B \wedge C) = (A \wedge B) \wedge C = A \wedge B \wedge C$;
- распределительные (дистрибутивные) законы:
  $A \wedge (B \vee C) = (A \wedge B) \vee (A \wedge C)$,
  $A \vee (B \wedge C) = (A \vee B) \wedge (A \vee C)$;
- поглощение:
  $A \vee (A \wedge B) = A$,
  $A \wedge (A \vee B) = A$;
- склеивание:
  $(A \wedge B) \vee (\overline{A} \wedge B) = B$,
  $(A \vee B) \wedge (\overline{A} \vee B) = B$;
- операция переменной с ее инверсией:
  $A \wedge \overline{A} = 0$,
  $A \vee \overline{A} = 1$;
- операция с константами (0 — `false`, 1 — `true`):
  $A \wedge 1$, $A \vee 1 = 1$,
  $A \wedge 0 = 0$, $A \vee 0 = A$;
- законы де Моргана:

$\overline{A \wedge B} = \overline{A} \vee \overline{B}$ (условно его можно назвать 1-й);
$\overline{A \vee B} = \overline{A} \wedge \overline{B}$ (2-й) — описывает результаты отрицания переменных, связанных операциями И, ИЛИ.

Высказывания, образованные с помощью нескольких операций логического сложения, умножения и отрицания, называются сложными. Истинность всякого сложного высказывания устанавливается с помощью таблиц истинности. Сложные высказывания, истинные (`true`) для любых значений истинности входящих в них простых высказываний, называются тождественно-истинными. Наоборот, тождественно-ложными являются формулы, принимающие значение `false` для любых значений входящих в него простых высказываний.

В табл. 1.20 приведено доказательство истинности дистрибутивного закона. Аналогичным образом могут быть доказаны и другие тождества.

Таблица 1.20. **Доказательство истинности дистрибутивного закона**

A	B	C	$B \vee C$	$A \wedge (B \vee C)$	$A \wedge B$	$A \wedge C$	$(A \wedge B) \vee (A \wedge C)$
0	0	0	0	0	0	0	0
0	0	1	1	0	0	0	0
0	1	0	1	0	0	0	0
0	1	1	1	0	0	0	0
1	0	0	0	0	0	0	0
1	0	1	1	1	0	1	1
1	1	0	1	1	1	0	1
1	1	1	1	1	1	1	1

На рис. 1.8, *а—е* приведены иллюстрации к основным логическим операциям и их композициям (так называемые диаграммы Эйлера — Венна) — области истинности каждого из высказываний и их объединения (дизъюнкции), пересечения (конъюнкции) и других операций. «Первый» из законов де Моргана иллюстрируется рис. 1.8, *д*, *е*.

Операции над битовыми строками. В некоторых современных ЯП реализованы операции и/или функции обработки битовых строк (машинных слов, их частей или групп, которые могут содержать числовые, символьные, мультимедийные и пр. данные). Наиболее распространенными являются побитовые операции, прямо вытекающие из рассмотренных ранее операций над логическими данными (табл. 1.21).

Таблица 1.21. **Операнды и результаты некоторых побитовых операций**

x	y	$x \,\&\, y$	$x \vee y$	x IMP y	x EQV y	x XOR y
0	0	0	0	1	1	0
0	1	0	1	1	0	1
1	0	0	1	0	0	1
1	1	1	1	1	1	0

Логические операции.

NOT. Поразрядный оператор NOT (НЕ), или дополнение, является одноместной операцией, которая выполняет логическое отрицание на каждом бите, формируя дополнение данного би-

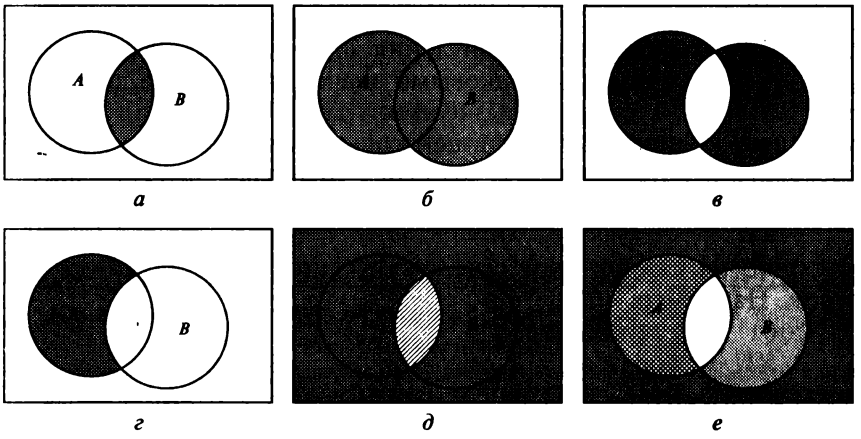


Рис. 1.8. Некоторые примеры диаграмм Эйлера — Венна:
а — конъюнкция высказываний *A* и *B* (AND); *б* — дизъюнкция высказываний *A* и *B* (OR); *в* — исключающая дизъюнкция (XOR); *г* — разность высказываний (*A* — *B*); *д* — иллюстрация к законам де Моргана (дополнение пересечения высказываний); *е* — иллюстрация к законам де Моргана (объединение дополнений)

нарного значения. Биты, содержавшие бывшие «0», устанавливаются в «1», и наоборот. Например:

```
1000 = NOT 0111
```

Во многих языках программирования (включая C), поразрядный оператор NOT обозначается как «~» (тильда). Этот оператор не следует путать с «логическим отрицанием» («!» — восклицательный знак), который обрабатывает все значение как отдельное булевское.

OR. Побитовый оператор OR (ИЛИ, «включающее ИЛИ») обрабатывает две битовые строки равной длины и создает третью (той же длины), где каждый бит является результатом операции «включающее ИЛИ» над каждой парой входных битов. Например:

```
0111 = 0101 OR 0011
```

В ЯП C поразрядный оператор OR обозначается символом «|» (вертикальная черта). Его булевский аналог обозначается как «||».

XOR («исключающее ИЛИ») выполняет логическую операцию XOR на каждой паре соответствующих битов двух строк одинаковой длины. Например:

```
0110 = 0101 XOR 0011
```

В ЯП C поразрядный оператор XOR обозначается как «^» («крышка»).

Ассемблерные программисты иногда используют операцию XOR как краткую команду обнуления значения регистра. Выполнение XOR над двумя одинаковыми значениями всегда дает нуль в результате, и во многих архитектурах эта операция требует меньшего количества циклов центрального процессора, чем последовательность операций, которые загружают нулевое значение в регистр.

AND. Поразрядный AND (И) выполняет логическую операцию AND на каждой паре соответствующих битов двух битовых строк. Например:

```
0001 = 0101 AND 0011
```

В ЯП C поразрядный оператор AND обозначается как «&» (амперсанд). Булевский аналог записывается как «&&» (два амперсанда).

Битовые сдвиги (bit shifts). Эти операции также осуществляются на битовом уровне и являются унарными. В этих операциях биты сдвигаются в регистре (влево или вправо). Поскольку регистры компьютера имеют ограниченный размер, некоторые биты будут «выдвинуты из регистра» с одной из его сторон, и аналогичное число бит должно быть «вдвинуто в регистр» с другой. Основное различие в операциях битовых сдвигов заключается в том, как именно они обрабатывают эти «вдвиги/выдвиги».

Арифметический сдвиг. При арифметическом сдвиге аннулируются биты, которые сдвинуты из любого конца регистра. Однако при арифметическом сдвиге влево с правой стороны вдвигаются нули, а при сдвиге вправо знаковый разряд остается в крайнем правом разряде, сохраняя знак операнда (рис. 1.9, *а*, *б*). Арифметический сдвиг влево на n разрядов эквивалентен умножению на 2^n (если не происходит переполнения), в то время как арифметический сдвиг вправо на n эквивалентен делению на 2^n.

Логический сдвиг. При логическом сдвиге аннулируются биты, которые сдвинуты, и их место занимают нули (в любом конце регистра) — рис. 1.9, *в*, *г*. Поэтому, логический и арифметический сдвиги влево оказываются тождественными. Однако логический сдвиг вправо вставляет биты со значением «0» вместо копии знакового разряда. Следовательно, логический сдвиг является более подходящим для двоичных чисел без знака, в то время как арифметический сдвиг — для двоичных со знаком.

Циклический сдвиг или разрядное вращение. В этой операции биты, «выдвигаемые» из регистра в любую сторону,

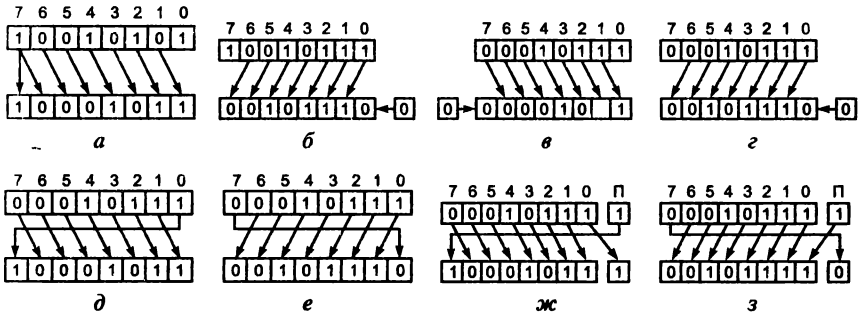


Рис. 1.9. Битовые сдвиги:
а — арифметический сдвиг вправо; *б* — арифметический сдвиг влево; *в* — логический сдвиг вправо; *г* — логический сдвиг влево; *д* — правый циклический сдвиг (вращение); *е* — левый циклический сдвиг; *ж* — правый циклический сдвиг через перенос; *з* — левый циклический сдвиг через перенос

«вдвигаются» в регистр с другой его стороны (рис. 1.9, *д*, *е*). Эта операция используется, если необходимо сохранить все существующие биты, и часто применяется в цифровой криптографии.

Циклический сдвиг «через перенос». В отличие от обычного сдвига (или сдвига «не через перенос») здесь считается, что два конца регистра отделены флажком переноса («п» на рис. 1.9, *ж*, *з*). Бит, который «вдвинут» (с любой стороны регистра), — старое значение флажка переноса, а бит, который «выдвинут» (с противоположной стороны) становится новым значением флажка переноса.

Единичный сдвиг через перенос может моделировать логический или арифметический сдвиг позиции, устанавливая флажок переноса заранее. Например, если флажок переноса содержит «0», то правый сдвиг через перенос соответствует логическому сдвигу вправо, а если флажок переноса содержит копию знакового разряда, то это арифметический сдвиг. Поэтому некоторые микроконтроллеры имеют только команды циклического сдвига. Циклический сдвиг через перенос особенно полезен, если операция осуществляется над числами, большими разрядности процессора (двойной или более длины). Если число хранится в двух регистрах, то бит, который «выдвинут» из первого регистра, должен быть «вдвинут» во второй. В данном случает этот бит сохраняется во флажке переноса в течение первого сдвига и готов к участию во втором сдвиге без какой-либо дополнительной подготовки.

Битовые манипуляции. Кроме перечисленных команд, существует большая группа операций, в том или ином виде реализованная в различных процессорах и объединяемая под общим названием «битовые манипуляции/жонглирование» (bit manipulation, bit twiddling, bit bashing). Эти операции связаны с задачами обнаружения и коррекции ошибок, алгоритмами шифрации данных и оптимизации, обработки мультимедийных данных, биоинформатики и пр. Например, в ЦП Motorolla 68000 это команды `BSET` (установка битов в «1»), `BCLR` (установка битов в «0») и `BTST` (переустановка нулевых битов); в систем команд SSE4 это `POPCNT` («Population Count» — подсчет количества битов, например, установленных в «1») и т. д.

Команды группы ABM (Advanced Bit Manipulation Instruction Set Architecture) поддерживают операции над частями слов (подсловами, subwords) размером в 8, 16, 32 или 64 бита (в дальнейшем — б и т г р у п п ы, БГ). Обзор некоторых команд этого типа приведен в табл. 1.22, 1.23. Здесь обычно P_1 — входной регистр, P_2 — выходной, P_3 — управляющий (ключевой, конфигурационный). Различаются следующие классы команд: перестановка бит и/или БГ, сжатие и расширение БГ, а также сравнение бит и/или БГ.

Таблица 1.22. **Обзор команд группы ABM (перестановки)**

Команда	Обозначение	Описание	Длительность в циклах АЛУ
Butterfly Inverse Butterfly Перестановки	`bfly` `ibfly`	Команды `bfly` (перестановка «бабочка») и `ibfly` (перестановка «обратная бабочка») осуществляют перестановку битов P_1. Исполнительные цепи спроектированы таким образом (сеть Бенеша, см. рис. 1.10, *а*, *б*), что единственное исполнение обеих инструкций позволяет получить все $n!$ перестановок n бит	1
Group Группирование	`grp`	БГ (например, байт) P_1 переупорядочивается в выходном регистре R_1 в соответствии с указателем, находящимся в P_{13}. Например, биты P_1, которым соответствуют «1» в R_3, перемещаются влево, а те, которым соответствуют «0», — вправо (рис. 1.10, *в*)	3
Mix Перемешивание	`mix`	БГ входных регистров P_1 и P_{13} разбиваются на пары и поочередно выбираются для помещения в P_2. Порядок выборки определяется модификацией команды (`mix.r` или `mix.l`, рис. 1.10, *г*)	1

Окончание табл. 1.22

Команда	Обозначение	Описание	Длительность в циклах АЛУ
Check, Excheck	`check` `excheck`	Команды `check` и `excheck` выполняют перестановку БГ (длиной в 1, 2 или 4 бита) в P_1 и P_{13}. БГ в каждом регистре разбиваются на пары и выбираются в P_2, причем `check` выбирает левый элемент каждой пары из P_1 (рис. 1.10, *д*) и правый из `excheck` — в обратном порядке (рис. 1.10, *е*)	1
Mux	`mux`	Выполняет 5 перестановок БГ: обратная перестановка бит (рис. 1.10, *ж*), перестановка частей регистра (рис. 1.10, *з*), заполнение всех байтов P_2 одинаковым кодом из P_1 и т. д.	1

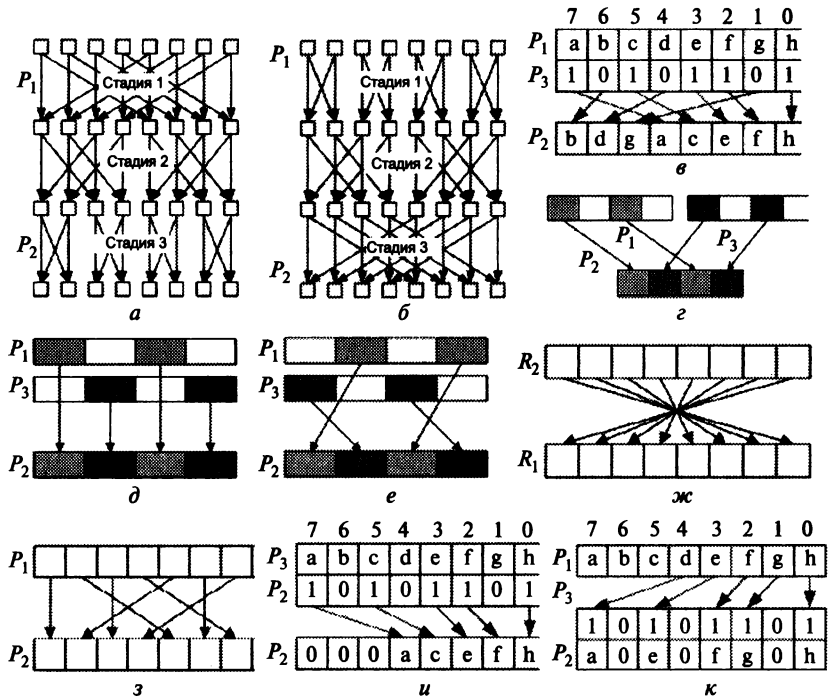


Рис. 1.10. Битовые манипуляции (АВМ):
а — восьмиразрядная перестановка `butterfly`; *б* — `inverse butterfly`; *в* — восьмиразрядная операция `grp`; *г* — операция `mix.2.l` (64-битовые слова); *д* — операция `check.2`; *е* — `excheck.2`; *ж* — операция `mux.1.rev` operation; *з* — `mux.1.mix`; *и* — 8-разрядная операция `pex`; *к* — `pdep`

Таблица 1.23. **Обзор команд группы ABM (сжатие-расширение строк)**

Команда	Модификация	Обозначение	Описание	Длительность в циклах АЛУ
Parallel Extract Параллельное сжатие	Static Variable	`pex` `pex.v`	Биты/БГ P_1, помеченные «1» в P_{13}, извлекаются и помещаются в правую часть P_{12}. Оставшиеся биты слева заполняются нулями (рис. 1.10, *и*)	1—3
Parallel Deposit Параллельное расширение	Static Variable	`pdep` `pdep.v`	Распределяет выровненную вправо (в P_1) последовательность БГ по позициям (в P_{12}), помеченным «1» в R_3. Оставшиеся позиции в P_2 заполняются нулями (рис. 1.10, *к*)	1—3
Population Count Подсчет «населения»		`popcount`	Подсчет числа битов P_1, установленных в «1»	1
Dot Product Скалярное произведение		`dotprod`	Скалярное произведение векторов, представляющих собой битовые строки	1
Bit Matrix Multiply Перемножение битовых матриц	$n \times n$	`bmm.1rR`	Умножение битовой матрицы размерности $n \times n$ на n-мерный битовый вектор. Операция эквивалентна n операциям dotprod	1 ($n = 64$)
	$k \times n$	`bmm.2rR`	Умножение $2n$-битовой матрицы на kn-битовую матрицу	1
	$1 \times n$	`bmm.2r`	Перемножение двух n-битовых матриц	1

Впервые функциональный (исполнительный) блок ABM появляется в процессорах архитектуры K10 (AMD) — см. рис. 3.35.

Синтез и оптимизация схем

При построении схемы, реализующей произвольную таблицу истинности, каждый выход анализируется (и строится схема) отдельно. Для реализации таблицы истинности с помощью логических элементов «И» достаточно рассмотреть только те строки таблицы истинности, которые содержат логические «1» в выходном сигнале. Строки, содержащие в выходном сигнале логиче-

ский «0», в построении схемы не участвуют. Каждая строка, содержащая в выходном сигнале логическую «1», реализуется схемой логического «И» с количеством входов, совпадающим с количеством входных сигналов в таблице истинности. Входные сигналы, описанные в таблице истинности логической «1», подаются на вход этой схемы непосредственно, а входные сигналы, описанные в таблице истинности логическим «0», подаются на вход через инверторы. Объединение сигналов с выходов схем, реализующих отдельные строки таблицы истинности, производится с помощью схемы логического «ИЛИ». Количество входов в этой схеме определяется количеством строк в таблице истинности, в которых в выходном сигнале присутствует логическая «1».

Рассмотрим конкретный пример. Пусть необходимо реализовать таблицу истинности, приведенную в табл. 1.24.

Таблица 1.24. **Таблица истинности некоторой логической функции**

Входы				Выходы	
x_1	x_2	x_3	x_4	y_1	y_2
0	0	0	0	0	0
0	0	0	1	1	0
0	0	1	0	0	0
0	0	1	1	0	0
0	1	0	0	0	0
0					0
0	1	1	1	0	0
1	0	0	0	0	0
1	0	1	0	0	0
1	0	1	1	0	0
					0
1	1	0	1	0	0
1	1	1	0	0	0

Для построения схемы, реализующей сигнал y_1, достаточно рассмотреть строки, выделенные светлой штриховкой. Эти строки реализуются сборкой (микросхемой) S_2 на рис. 1.11. Каждая строка реализуется своей схемой «И», затем выходы этих схем объединяются. Для построения схемы, реализующей сигнал y_2, достаточно рассмотреть строки, выделенные более темной штриховкой. Эти строки реализуются сборкой S_3. Инвертирование входов схемы осуществляется сборкой S_1.

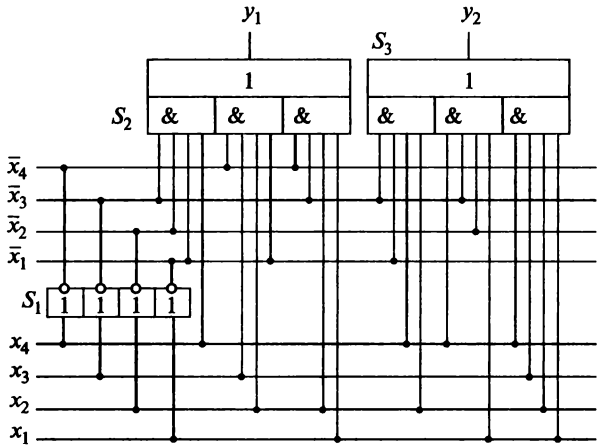


Рис. 1.11. Схемная реализация таблицы истинности (табл. 1.24)

В частности, в данном случае

$$y_1 = (\bar{x}_1 \wedge \bar{x}_2 \wedge \bar{x}_3 \wedge x_4) \vee (\bar{x}_1 \wedge x_2 \wedge x_3 \wedge \bar{x}_4) \vee$$

$$\vee (x_1 \wedge x_2 \wedge \bar{x}_3 \wedge \bar{x}_4) \text{ и т. д.}$$

Преобразования логических формул. Равносильные преобразования логических формул имеют то же назначение, что и преобразования формул в обычной алгебре. Они служат для упрощения формул или приведения их к определенному виду путем использования основных законов алгебры логики.

Под *упрощением формулы*, не содержащей операций импликации и эквиваленции, понимают *равносильное преобразование*, приводящее к формуле, которая по сравнению с исходной либо содержит меньшее число операций конъюнкции и дизъюнкции и не содержит отрицаний неэлементарных формул, либо содержит меньшее число вхождений переменных.

Некоторые преобразования логических формул похожи на преобразования формул в обычной алгебре (вынесение общего множителя за скобки, использование переместительного и сочетательного законов и т. п.), тогда как другие преобразования основаны на свойствах, которыми не обладают операции обычной алгебры (использование распределительного закона для конъюнкции, свойств поглощения и склеивания, законов де Моргана и др.).

Приведем несколько примеров, иллюстрирующих методы, применяемые при упрощении логических формул:

1) $\overline{x \vee y} \wedge (x \wedge \bar{y}) = \bar{x} \wedge \bar{y} \wedge (x \wedge \bar{y}) = x \wedge \bar{x} \wedge \bar{y} \wedge \bar{y} =$

$$= 0 \wedge \bar{y} \wedge \bar{y} = 0$$

(законы алгебры логики применяются в следующей последовательности: закон де Моргана, сочетательный закон, правило конъюнкции переменной с ее инверсией и правило операций с константами);

2) $$\bar{x} \wedge y \vee \overline{x \vee y} \vee x = \bar{x} \wedge y \vee \bar{x} \wedge \bar{y} \vee x =$$

$$= \bar{x} \wedge (y \vee \bar{y}) \vee x = \bar{x} \vee x = 1$$

(применяется закон де Моргана, выносится за скобки общий множитель, используется правило операций переменной с ее инверсией);

3) $$x \wedge \bar{y} \vee \bar{x} \wedge y \wedge z \vee x \wedge z =$$

$$= x \wedge \bar{y} \vee \bar{x} \wedge y \wedge z \vee x \wedge z \wedge (y \vee \bar{y}) =$$

$$= x \wedge \bar{y} \vee \bar{x} \wedge y \wedge z \vee x \wedge y \wedge z \vee x \wedge \bar{y} \wedge z =$$

$$= (x \wedge \bar{y} \vee x \wedge \bar{y} \wedge z) \vee (\bar{x} \wedge y \wedge z \vee x \wedge y \wedge z) = x \wedge \bar{y} \vee y \wedge z$$

(вводится вспомогательный логический сомножитель $(y \vee \bar{y})$, комбинируются два крайних и два средних логических слагаемых и используется закон поглощения);

4) $$x \vee \overline{y \wedge \bar{z}} \vee \overline{\bar{x} \vee y \vee \bar{z}} = x \vee \bar{y} \vee \bar{\bar{z}} \vee \bar{\bar{x}} \wedge \bar{y} \wedge \bar{\bar{z}} =$$

$$= x \vee \bar{y} \vee z \vee x \wedge \bar{y} \wedge z = x \vee z \vee (\bar{y} \vee x \wedge \bar{y} \wedge z) = x \vee z \vee \bar{y}$$

(к отрицаниям неэлементарных формул применяется закон де Моргана, используются двойное отрицание и склеивание);

5) $$x \wedge \bar{y} \vee x \wedge y \wedge z \vee x \wedge \bar{y} \wedge z \vee x \wedge \overline{y \wedge z} =$$
$$= x \wedge (\bar{y} \vee y \wedge z \vee \bar{y} \wedge z \vee \overline{y \wedge z}) =$$
$$= x \wedge ((\bar{y} \vee \bar{y} \wedge z) \vee (y \wedge z \vee \overline{y \wedge z})) =$$
$$= x \wedge (\bar{y} \vee \bar{y} \wedge z \vee 1) = x \wedge 1 = x$$

(общий множитель x выносится за скобки, комбинируются слагаемые в скобках — первое с третьим и второе с четвертым, к дизъюнкции $y \wedge z \vee \overline{y \wedge z}$ применяется правило операции переменной с ее инверсией);

6) $$(x \wedge \bar{y} \vee z) \wedge (\bar{x} \vee y) \vee \bar{z} =$$
$$= x \wedge \bar{y} \wedge \bar{x} \vee x \wedge \bar{y} \wedge y \vee z \wedge \bar{x} \vee z \wedge y \vee \bar{z} =$$
$$= 0 \vee 0 \vee z \wedge \bar{x} \vee z \wedge y \vee \bar{z} = z \wedge \bar{x} \vee (z \vee \bar{z}) \wedge (y \vee \bar{z}) =$$
$$= z \wedge \bar{x} \vee 1 \wedge (y \vee \bar{z}) = z \wedge \bar{x} \vee y \vee \bar{z} = (z \wedge \bar{x} \vee \bar{z}) \vee y =$$
$$= (z \vee \bar{z}) \wedge (\bar{x} \vee \bar{z}) \vee y = 1 \wedge (\bar{x} \vee \bar{z}) \vee y = \bar{x} \vee \bar{z} \vee y$$

(используются распределительный закон для дизъюнкции, правило операции переменной с ее инверсией, правило операций с константами, переместительный закон и распределительный закон для конъюнкции);

7) $$x \wedge y \wedge (\bar{x} \wedge z \vee \overline{x \wedge y \wedge z} \vee z \wedge t) =$$
$$= x \wedge y \wedge (\bar{x} \wedge z \vee \overline{x \wedge y} \vee \bar{z} \vee z \wedge t) =$$
$$= x \wedge y \wedge (\bar{x} \wedge z \vee x \wedge y \vee \bar{z} \vee z \wedge t) =$$
$$= x \wedge y \vee x \wedge y \wedge \bar{z} \vee x \wedge y \wedge z \wedge t = x \wedge y$$

(используются закон де Моргана, двойное отрицание и поглощение).

Минимизация логического выражения. Целью минимизации логического выражения, представляющего заданную логическую функцию, является уменьшение стоимости ее реализации (количества используемых логических элементов). Общая схема процесса реализации логической функции такова. Для нее составляется сумма произведений, или *дизъюнктивная совершенная нормальная форма*. Затем полученное выражение минимизируют до эквивалентной *минимальной суммы произведений*. Чтобы определить критерий минимизации, вводится понятие *стоимости, или величины,* логического выражения.

Обычно при оценке стоимости выражения учитывается общее количество вентилей и их входных значений (входных линий), необходимых для реализации выражения в форме, приведенной на рис. 1.12.

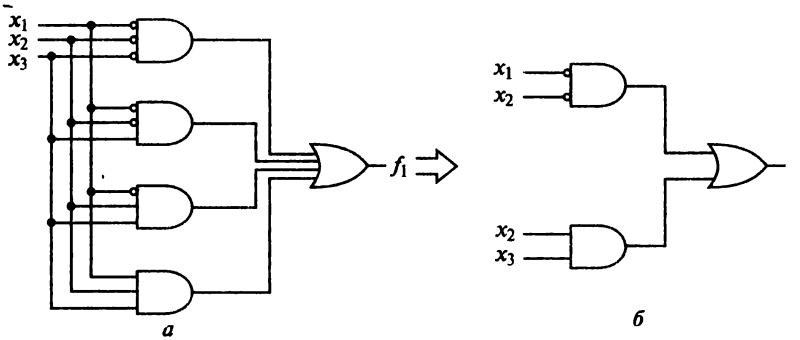


Рис. 1.12. Минимизация логических схем

Стоимость бóльшей схемы (рис. 1.12, *а*) равна 21:

5 вентилей плюс 16 входных значений.

Инверсия входных значений при подсчете игнорируется. Стоимость более простого выражения (рис. 1.12, *б*) равна 9:

3 вентиля плюс 6 входных значений.

Теперь можно определить критерий минимизации: *сумма произведений считается минимальной, если не существует эквивалентного ей выражения меньшей стоимости.*

Стратегия упрощения заданного выражения заключается в следующем. Прежде всего термы-произведения разбиваются на пары, различающиеся единой переменной, которая в одном терме стоит со знаком $\neg$ ($\bar{x}$), а во втором — без него (x). Затем в каждой паре общее произведение двух переменных выносится за скобки, а в скобках остается терм $\neg x + x$, всегда равный 1. Вот что мы получим, применив эту процедуру к первому выражению для функции f_1 (на рис. 1.12, *а*),

$$f_1 = (\bar{x}_1 \wedge \bar{x}_2 \wedge \bar{x}_3) \vee (\bar{x}_1 \wedge \bar{x}_2 \wedge x_3) \vee (\bar{x}_1 \wedge x_2 \wedge x_3) \vee (x_1 \wedge x_2 \wedge x_3) =$$
$$= \bar{x}_1 \wedge \bar{x}_2 \wedge (\bar{x}_3 \vee x_3) \vee (\bar{x}_1 \wedge x_1) \wedge x_2 \wedge x_3 =$$
$$= \bar{x}_1 \wedge \bar{x}_2 \wedge 1 \vee 1 \wedge x_2 \wedge x_3 =$$
$$= \bar{x}_1 \wedge \bar{x}_2 \vee x_2 \wedge x_3.$$

Это выражение минимально. Соответствующая ему логическая схема приведена на рис 1.12, *б*.

Другие схемные элементы ЭВМ

Триггеры (от *англ.* trigger — защелка, спусковой крючок) — электронная схема, широко применяемая в регистрах компьютера для запоминания одного разряда двоичного кода. Триггер имеет два устойчивых состояния, одно из которых соответствует двоичной единице, а другое — двоичному нулю.

Для обозначения этой схемы в английском языке часто употребляется термин *flip-flop*, что в переводе означает «хлопанье». Это звукоподражательное название электронной схемы указывает на ее способность мгновенно переходить («перебрасываться») из одного состояния в другое и обратно.

Самый распространенный тип триггера — так называемый *RS*-триггер (*S* и *R*, соответственно, от *англ.* set — установка и reset — сброс). Он имеет два симметричных входа *S* и *R* и два симметричных выхода *Q* и $\overline{Q}$, причем выходной сигнал *Q* является логическим отрицанием сигнала $\overline{Q}$. Условное обозначение *RS*-триггера приводится на рис. 1.13, *а*.

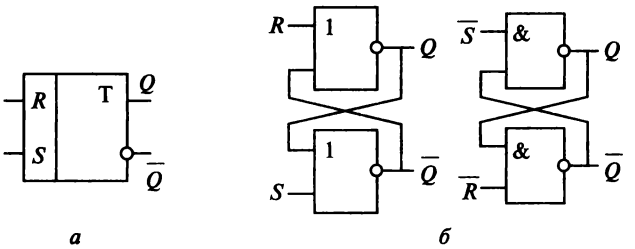


Рис. 1.13. *RS*-триггер (*а*), его реализация (*б*)

На каждый из двух входов *S* и *R* могут подаваться входные сигналы в виде кратковременных импульсов (рис. 1.13, *а*). На рис. 1.13, *б* показана реализация триггера на базе вентилей «ИЛИ-НЕ» и «И-НЕ».

Перечислим возможные комбинации значений входов *R* и *S* триггера, используя его схему и таблицу истинности схемы «ИЛИ-НЕ» (табл. 1.25).

1. Если на входы триггера подать $S = 1$, $R = 0$, то (независимо от состояния) на выходе *Q* верхнего вентиля появится «0». После этого на входах нижнего вентиля окажется $R = 0$, $Q = 0$ и выход $\overline{Q}$ станет равным «1».

Таблица 1.25. **Таблица истинности для *RS*-триггера**

S	R	Q	$\overline{Q}$
0	0	Без изменений	
0	1	1	0
1	0	0	1
1	1	Не определено	

2. При подаче «0» на вход S и «1» на вход R на выходе $\overline{Q}$ появится «0», а на Q — «1».

3. Если на входы R и S одновременно подан логический «0», то состояние Q и $\overline{Q}$ не меняется.

4. Состояние триггера при $R = 1$ и $S = 1$ считается неопределенным, так как после снятия таких сигналов триггер не переходит однозначно в нужное состояние. Поэтому на состояние входов налагается условие $R \wedge S = 0$.

Поскольку триггер может запомнить только один разряд двоичного кода, для запоминания байта нужно 8 триггеров, для запоминания килобайта, соответственно, $8 \times 2^{10} = 8192$ триггера. Современные микросхемы памяти содержат миллионы триггеров.

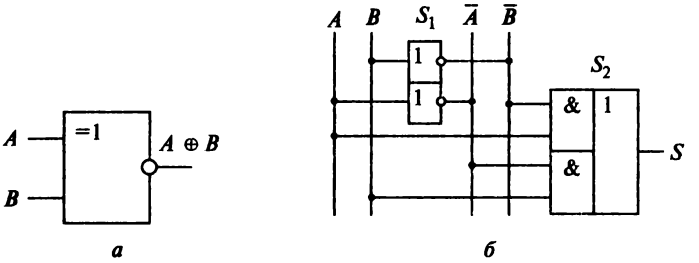
Кроме *RS*-триггеров известны также *JK*-, *T*- и *D*-триггеры. *JK*-триггер содержит схемные дополнения, которые снимают неопределенность состояния при подаче «1» на оба входа.

T-триггер имеет единственный вход (T), при подаче «1» на который осуществляется «переброс» схемы. *T*-триггеры могут использоваться для создания двоичных *счетчиков* (например, счетчик адреса команд, обеспечивающий последовательную выборку слов из оперативной памяти).

D-триггер имеет информационный вход D и вход синхронизации C. Состояние на выходе *D*-триггера отражает информацию, поступившую на его информационный вход в течение воздействия синхросигнала.

Рассмотрим далее одноразрядные сумматоры.

Сумматор по модулю 2 является простейшим среди них и реализует операцию A xor B (Исключающее ИЛИ, см. табл. 1.14). Обозначение на схемах и реализация сумматора по модулю 2 приводятся на рис. 1.14.



а *б*

Рис. 1.14. Сумматор по модулю 2:
а — обозначение на схемах; *б* — реализация

Полусумматор. Вспомним, что при сложении двоичных чисел образуется сумма в данном разряде и при этом возможен перенос в старший разряд. Обозначим слагаемые (*A*, *B*), перенос (*P*), сумму (*S*) и рассмотрим соответствующую данной операции табл. 1.26.

Таблица 1.26. **Таблица сложения одноразрядных двоичных чисел с учетом переноса в старший разряд (полусумматор)**

Слагаемые		Перенос	Сумма
A	*B*	*P*	*S*
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Из этой таблицы следует, что перенос может быть реализован с помощью операции логического умножения

$$P = A \wedge B.$$

Получим формулу для вычисления суммы. Значения суммы почти всегда совпадают с результатом операции логического сложения (кроме случая, когда на вход подаются две единицы, а на выходе должен получиться нуль).

Нужный результат достигается, если результат логического сложения умножить на инвертированный перенос. Таким образом, для определения суммы можно применить выражение (см. сложение по модулю 2):

$$S = A \vee B \wedge \overline{A \wedge B}.$$

На основе полученных логических выражений можно построить схему полусумматора из базовых логических элементов.

Из логической формулы для суммы следует, что на выходе должен стоять элемент логического умножения «И», который имеет два входа. На один из входов подается результат логического сложения исходных величин, т. е. на него должен подаваться сигнал с элемента логического сложения «ИЛИ».

На второй вход требуется подать результат инвертированного логического умножения исходных сигналов $\overline{A \wedge B}$, т. е. на второй вход подается сигнал с элемента «НЕ», на вход которого поступает сигнал с элемента логического умножения «И».

Данная схема называется полусумматором, так как реализует суммирование одноразрядных двоичных чисел без учета переноса из младшего разряда (рис. 1.15).

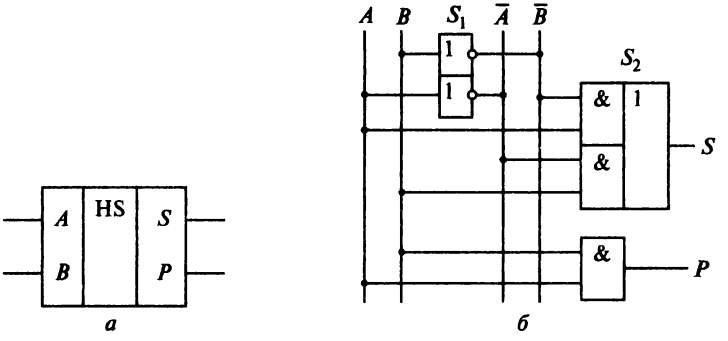


Рис. 1.15. Полусумматор:
а — обозначение на схемах; *б* — реализация

Полный одноразрядный сумматор. Полный одноразрядный сумматор должен иметь три входа: a_i, b_i — слагаемые и p_{i-1} — перенос из предыдущего разряда и два выхода: сумма S_i и перенос p_i. Порядок функционирования схемы приводится в табл. 1.27.

Последовательность построения полного сумматора такая же, как и полусумматора. Перенос реализуется с помощью следующей формулы:

$$p_i = (a_i \wedge b_i) \vee (a_i \wedge p_{i-1}) \vee (b_i \wedge p_{i-1}).$$

Логическое выражение для вычисления суммы в полном сумматоре принимает следующий вид:

$$S_i = (a_i \vee b_i \vee p_{i-1}) \wedge p_{i-1} \vee (a_i \wedge b_i \wedge p_{i-1}).$$

Таблица 1.27. **Таблица сложения для полного одноразрядного сумматора**

Слагаемые		Входящий перенос	Выходящий перенос	Сумма
a_i	b_i	p_{i-1}	p_i	S_i
0	0	0	0	0
0	0	1	0	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	1	0
1	1	1	1	0
1	1	1	1	1

В соответствии с принципами построения схемы по произвольной таблице истинности (см. табл. 1.24, рис. 1.11) можно построить схему полного двоичного одноразрядного сумматора (рис. 1.16, *б*).

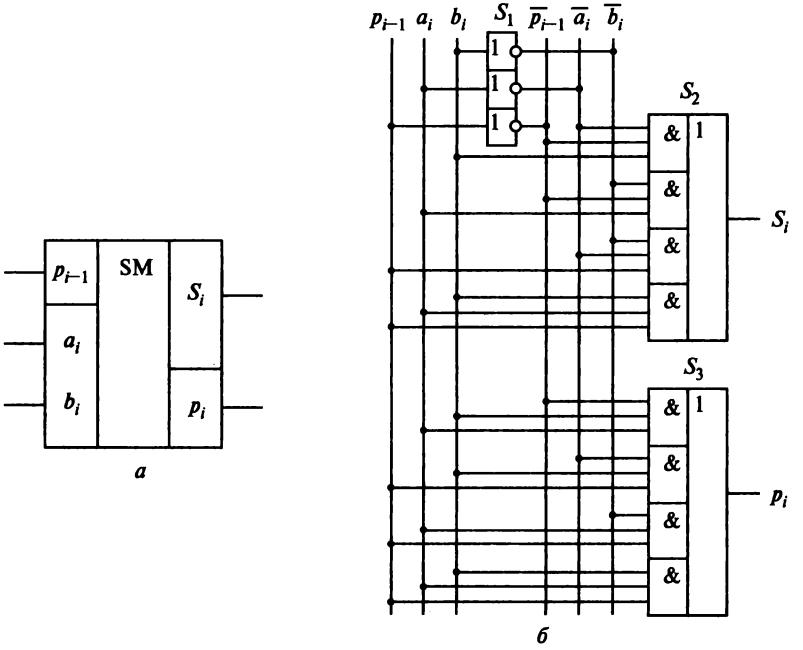


Рис. 1.16. Одноразрядный сумматор:
а — обозначение на схемах; *б* — реализация на логических сборках

1.4. Технологии электронных схем

Основой электронных технологий в настоящее время являются полупроводники (semiconductors) — вещества, электропроводность которых увеличивается с ростом температуры и является промежуточной между проводимостью металлов и изоляторов.

Наиболее часто используемыми в электронике полупроводниками являются кремний и германий. На их основе путем внедрения примесей в определенных точках кристаллов создаются разнообразные полупроводниковые элементы, к которым, в первую очередь, относятся:

- проводники, коммутирующие активные элементы;
- вентили, выполняющие логические операции;
- транзисторы (полупроводниковые триоды), предназначенные для усиления, генерирования и преобразования электрического тока;
- резисторы, обеспечивающие режимы работы активных элементов;
- приборы с зарядовой связью (ПЗС), предназначенные для кратковременного хранения электрического заряда и используемые в светочувствительных матрицах видеокамер;
- диоды и др.

В настоящее время используется несколько технологий построения логических элементов:

- транзисторно-транзисторная логика (ТТЛ, TTL);
- логика на основе комплементарных МОП-транзисторов (КМОП, CMOS);
- логика на основе сочетания комплементарных МОП- и биполярных транзисторов (BiCMOS).

Кроме того, различают:

- положительную логику, или систему высоких потенциалов;
- отрицательную логику, или систему низких потенциалов;
- смешанную.

При положительной логике напряжение высокого уровня соответствует логической «1», а при отрицательной логике — «0».

Логические элементы, функционирующие в системе высоких потенциалов, дуальны элементам, работающим в системе низких потенциалов. Например, в системе высоких потенциалов элемент реализует функцию «ИЛИ-НЕ», а в системе низких потенциалов — «И-НЕ».

Рассмотрим рис. 1.17, на котором достаточно упрощенно представлены транзисторные сборки «И» (последовательно включенные транзисторы) и «ИЛИ» (параллельное включение). Входные и выходные сигналы «1» представляются высоким уровнем напряжения на коллекторе транзистора (практически равным напряжению питания). Сигналу «0», наоборот, соответствует низкий уровень выходного напряжения.

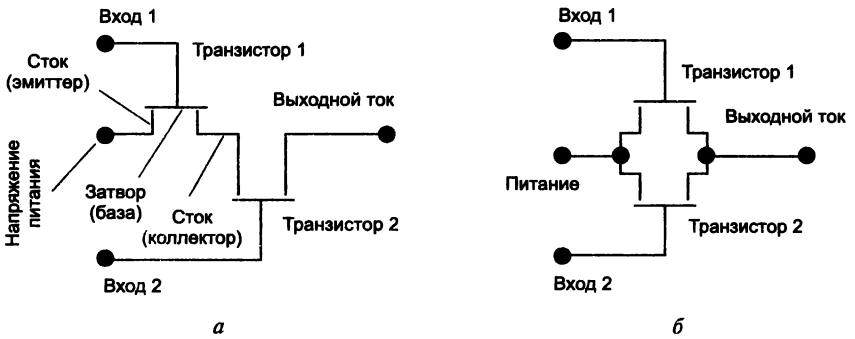


Рис. 1.17. Пример реализации сборок «И» (*а*) и «ИЛИ» (*б*)

Поскольку, например, в большинстве современных персональных компьютеров напряжение питания составляет 3,3 В (в более ранних версиях, до Pentium — 5 В), то выходная «1» задается напряжением 3,3 В.

На рис. 1.18 приводится иллюстрация так называемого «закона/правила Мура», с высокой точностью демонстрирующего удвоение за 18—24 мес. количества транзисторов в процессорах. Основой этой закономерности является объективный процесс увеличения плотности упаковки элементов микросхем (рис. 1.19).

Ключевыми выражениями при описании микросхемных элементов (рис. 1.19) являются такие, как «технология 130 нм», «технологический процесс 0,5 мкм» и т. д. Это означает, что размеры транзисторов или других элементов (узлов, node) соответственно не превышают 130 нанометров (1 нм = 10^{-9} м) либо же 0,5 микрон (1 мкм = 10^{-6} м) — рис. 1.20.

В процессоре Intel 4004 (1971 г.) использовалась *технология 10 мкм*; в процессоре Pentium II (1998 г.) — *технология 0,25 мкм*; в процессорах Intel Atom и AMD Shanghai (2008 г.) — *нанотехнологии 0,045 мкм* (45 нм) (см. также табл. 3.3 и 3.6).

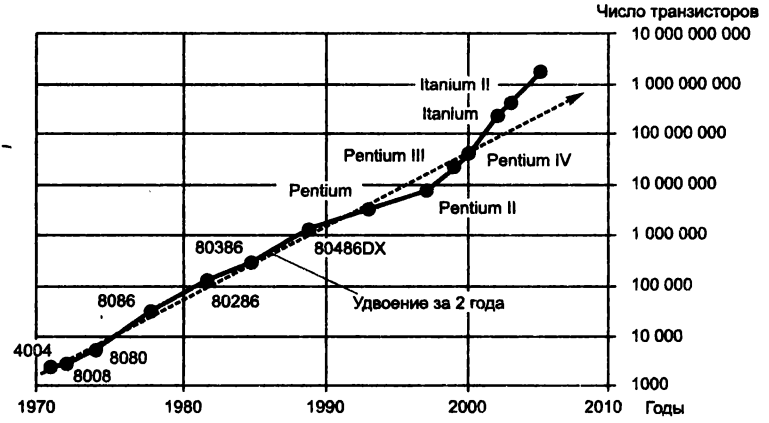


Рис. 1.18. Правило Мура (количество транзисторов в интегральной схеме удваивается каждые 18 мес.)

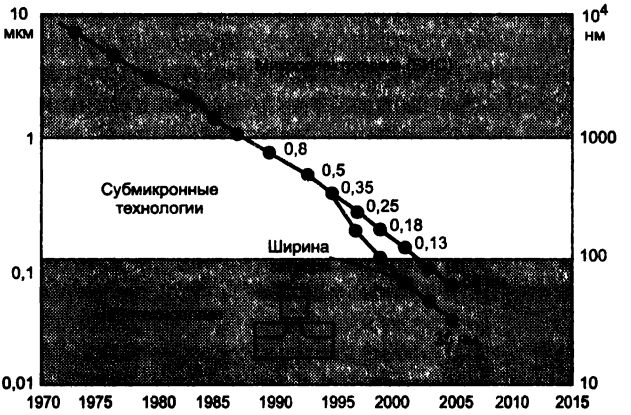
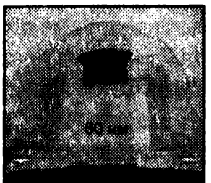


Рис. 1.19. Динамика изменений размеров схемных элементов



а

б

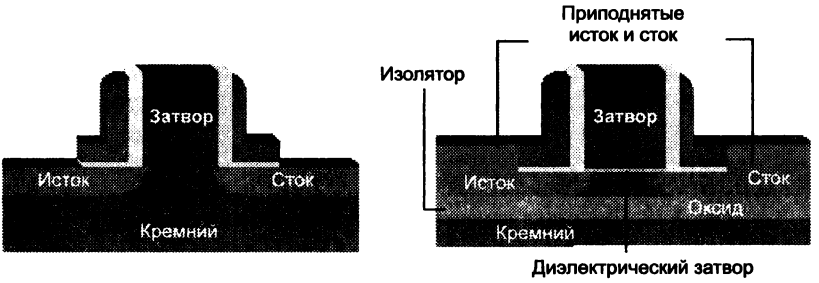
Рис. 1.20. Нанотехнологии наглядно:
а — транзистор (90 нм); *б* — вирус гриппа (100 нм)

Микропроцессоры

Microprocessor — процессор, выполненный в одной либо нескольких взаимосвязанных интегральных схемах.

Процессор полностью собирается на одном чипе из кремния. Электронные цепи создаются в несколько слоев, состоящих из различных веществ, например, диоксид кремния может играть роль изолятора, а поликремний — проводника.

В частности, транзистор представляет собой простейшее устройство, размещающееся на поверхности кремниевой пластины и функционирующее как электронный ключ (рис. 1.21, *а*). Обычно он содержит три вывода — исток (эмиттер), сток (коллектор) и затвор (база). Заметим, что в ламповых элементах соответствующие электроды именовались — катод, анод, сетка. Источник и сток образуются путем внедрения в поверхность кремния определенных примесей, а затвор содержит материал, именуемый полисиликоном. Ниже затвора расположен слой диэлектрика, изготовленного из диоксида кремния. Данная структура получила название «кремний-на-изоляторе» (silicon-on-insulator — SOI). Когда к транзистору приложено напряжение, затвор «открыт», и транзистор пропускает ток. Если напряжение снято, затвор «закрыт» и тока нет.



а *б*

Рис. 1.21. Обычный транзистор (*а*), терагерц-транзистор (*б*)

Традиционная технология. Технология микропроцессоров в простейшем случае включает следующие обязательные этапы производства:

- выращивание кремниевых заготовок и получение из них пластин;

- шлифование кремниевых пластин;
- нанесение защитной пленки диэлектрика (SiO_2);
- нанесение фоторезиста;
- литографический процесс;
- травление;
- диффузию;
- металлизацию.

Все перечисленные этапы используются для того, чтобы на кремниевой основе создать сложную структуру полупроводниковых планарных транзисторов (CMOS-транзисторов) и связать их должным образом между собой.

Процесс изготовления любой микросхемы начинается с выращивания кремниевых монокристаллических болванок цилиндрической формы (*кремниевых заготовок*). Это лишенный примесей монокристалл.

В дальнейшем из таких монокристаллических заготовок нарезают круглые пластины, *«таблетки»* (*waffer — вафля, облатка*), толщина которых составляет приблизительно от 0,2 до 1,0 мм, а диаметр — от 5 см (ранние технологии) до 20 см (современные технологии), поверхность которых отполировывается до зеркального блеска, а затем покрывается тончайшим слоем оксидной пленки (SiO_2), выполняющей функцию диэлектрика и защитной пленки при дальнейшей обработке кристалла кремния.

После того как кремниевая основа покроется защитной пленкой диоксида кремния, необходимо удалить эту пленку с тех мест, которые будут подвергаться дальнейшей обработке. Удаление пленки осуществляется посредством травления, а для того, чтобы в результате травления оксидная пленка удалялась избирательно, на поверхность пленки наносят слой фоторезиста (состава, чувствительного к воздействию света). Облученные области становятся растворимыми в кислотной среде.

Процесс нанесения фоторезиста и его дальнейшее облучение ультрафиолетом по заданному рисунку называется фотолитографией. Для засветки нужных участков слоя фоторезиста используется шаблон-маска, который содержит рисунок одного из слоев будущей микросхемы. Свет, проходя сквозь такой шаблон, засвечивает только нужные участки поверхности слоя фоторезиста. После облучения фоторезист подвергается проявлению, в результате которого удаляются ненужные участки слоя.

По мере возрастания плотности размещения транзисторов, формируемых в кристалле, литографический процесс усложня-

ется. Минимальная толщина линии, получаемая в процессе литографии, определяется размером пятна, в который удается сфокусировать лазерный луч. Поэтому при производстве современных микропроцессоров для облучения используют ультрафиолетовое излучение. Для производства микросхем по 130-нанометровому технологическому процессу используется глубокое ультрафиолетовое излучение (Deep UltraViolet — DUV) с длиной волны 248 нм. На подходе литографический процесс с длиной волны 13 нм, получивший название EUV-литографии (Extreme UltraViolet — сверхжесткое ультрафиолетовое излучение). Обычная литографическая технология позволяет наносить шаблон с минимальной шириной проводников 100 нм, а EUV-литография делает возможной печать линий гораздо меньшей ширины — до 30 нм.

После засвечивания слоя фоторезиста осуществляется травление (etching) с целью удаления пленки диоксида кремния. После процедуры травления, т. е. когда оголены нужные области чистого кремния, удаляется оставшаяся часть фотослоя, и на кремниевой основе остается рисунок, выполненный диоксидом кремния.

Процесс внедрения примесей осуществляется посредством диффузии — равномерного внедрения атомов примеси в кристаллическую решетку кремния. Для диффузии легирующей примеси применяется ионная имплантация, которая завершается созданием необходимого слоя полупроводниковой структуры, в котором сосредоточены десятки миллионов транзисторов.

Осуществить требуемую разводку в пределах того же слоя, где расположены сами транзисторы, нереально — неизбежны пересечения между проводниками, потому для соединения транзисторов друг с другом применяют несколько слоев металлизации, т. е. слоев с металлическими проводниками, причем, чем больше транзисторов насчитывается в микросхеме, тем больше слоев металлизации используется (см. рис. 1.24, *б*).

Для соединения транзисторов друг с другом прежде всего необходимо создать проводящие контакты стоков, истоков и затворов. Для этого по маске в нужных местах вытравливается слой диоксида кремния, и соответствующие окна заполняются атомами металла. Для создания очередного слоя на полученном рисунке схемы выращивается дополнительный тонкий слой диоксида кремния. После этого наносится слой проводящего металла

и еще один слой фоторезиста. Ультрафиолетовое излучение пропускается сквозь вторую маску и высвечивает соответствующий рисунок на фоторезисте. Затем опять следуют этапы растворения фоторезиста и травления металла. В результате в новом слое образуются нужные проводящие полоски, напоминающие рельсы, а для межслойных соединений, т. е. соединений слоев друг с другом, в слоях оставляются окна, которые затем заполняются атомами металла. К примеру, при 0,25-микронном технологическом процессе для осуществления разводки используется пять дополнительных слоев.

Процесс нанесения слоев заканчивается, когда схема собрана полностью. Поскольку за один раз на одной «таблетке» создается несколько десятков процессоров, на следующем этапе они разделяются на матрицы (dice), которые тестируются. Если на ранних этапах развития технологий отбраковывалось более 50 % схем, сейчас процент выхода выше, но никогда не достигает 100 %.

Прошедшая тестирование матрица помещается в керамический прямоугольный футляр, из которого выходят *«ножки»*, *микроразъемы* (pin grid arrays — PGA) интерфейса процессора, с помощью которых процессор помещается и закрепляется в *гнезде* (*socket*) на системной плате компьютера (иногда интерфейс оформляется в виде линейного разъема — *slot*). Количество контактов — от 169 (Socket 1, процессор Intel 80486) до 940 (Socket 940, AMD Opteron). В последнем случае часть соединений зарезервирована для последующего расширения возможностей — размещения на плате процессора кэш-памяти уровня 3 (L3-cache), соединения с другими процессорами (для многопроцессорных систем) и пр.

В настоящее время используется технология микроразъемов (micro pin grid array — μPGA), существенно снижающая физические размеры интерфейса процессора.

В новом поколении процессоров используются такие нововведения, как SOI-транзисторы (Silicon On Isolator — «кремний на изоляторе»), в которых за счет дополнительного слоя оксида снижаются емкость и токи утечки, а также транзисторы с двумерными затворами и другие новшества, позволяющие повысить быстродействие транзисторов при одновременном уменьшении их геометрических размеров.

Чипы памяти DRAM изготовляются на основе технологии, сходной с изготовлением процессора, — кремниевая основа с

нанесенными примесями обрабатывается с маской, которая образует множество пар «транзистор—емкость», каждая из которых размещает 1 бит информации. Стоимость этих схем гораздо ниже, чем процессоров, поскольку они состоят из однородных повторяющихся структур, а также дешевле схем SRAM, поскольку в последних содержится в 2 раза больше транзисторов (каждый бит здесь содержится в триггере, который требует по меньшей мере два транзистора).

Терагерц-технологии. Основная стратегия поставщиков микросхем всегда заключалась в уменьшении размера транзистора (схемного элемента) и повышении плотности упаковки на кристалле. В конечном итоге критическими факторами стали энергопотребление и разогрев платы.

В конце 2002 г. Intel Corporation объявила, что ее инженеры разработали инновационную структуру транзисторов и новые материалы, позволяющие снизить потребление энергии и выделение тепла. Новые структуры получили название Intel TeraHertz transistor (терагерц-транзисторы), в связи с их способностью переключаться со скоростью выше триллиона раз в секунду. Предполагается, что новая технология позволит увеличить плотность в 25 раз, использовать «технологию 20 нм» (элемент схемы в 250 раз меньше толщины человеческого волоса) и разместить на кристалле до миллиарда транзисторов.

Терагерц-транзистор отличается от обычного (см. рис. 1.21, *а*) тремя важными особенностями (см. рис. 1.21, *б*):

- источник и сток образуются из более толстых слоев в кремниевой пластине, что уменьшает электрическое сопротивление, потребление электроэнергии и тепловыделение;
- ниже источника и стока помещается сверхтонкий слой изолятора. Это обеспечивает более высокие интенсивности тока в открытом состоянии транзистора и увеличивает скорость переключения. Кроме того, изолятор понижает утечки тока при закрытом транзисторе (в 10 тыс. раз по сравнению с SOI). Это уменьшает вероятность случайного переключения под влиянием блуждающих тепловых электронов и повышает надежность схемы;
- химическое соединение, расположенное между затвором, источником, стоком, заменяется на новый материал «high-k gate dielectric» (оксид алюминия или титана), для нанесения которого используется технология наращивания слоя по одной молекуле.

Диэлектрико-металлические затворы транзисторов. Использование затвора из диэлектриков с высокой диэлектрической постоянной (High-k Gate Dielectrics) и металлических электродов затворов транзисторов (Metal Gate Electrodes) было впервые представлено в процессоре Intel Penryn (технология 45 нм) и позволило уменьшить размеры транзисторов и снизить энергопотребление.

В обычном транзисторе снижение толщины слоя диоксида кремния необходимо для уменьшения размера и увеличения плотности размещения транзисторов на кристалле. Однако при достижении определенного предела возникает утечка тока под воздействием «туннельного эффекта» — когда электроны покидают транзистор и рассеиваются, что понижает надежность и увеличивает рассеяние мощности. Поэтому уменьшение размеров ниже данного предела становится нецелесообразным.

Диэлектрик (high-k dielectric или материал с высокой диэлектрической постоянной) в новой технологии замещает слой диоксида кремния в транзисторе и позволяет снизить токи утечки в технологии 45 нм в 5 раз по сравнению с технологией 65 нм.

Относительная легкость использования оксидов кремния в транзисторах ограничивала в течение многих лет применение других материалов при производстве микропроцессоров. Аналогично, традиционная технология использования поликремния для затвора существенно проще, чем внедрение других, возможно более эффективных веществ в процесс производства (рис. 1.22, *а*).

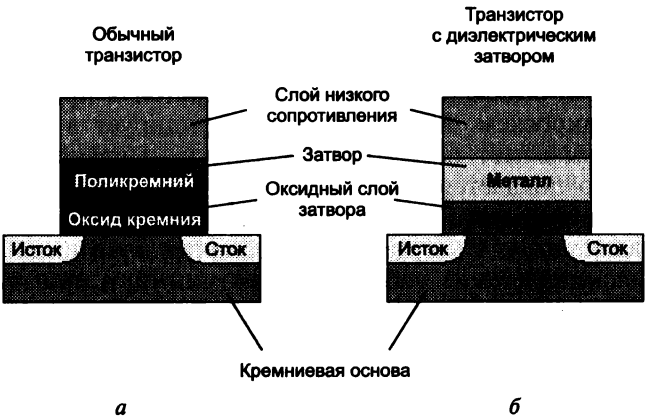


Рис. 1.22. Обычный транзистор (*а*); транзистор с диэлектрическим затвором (*б*)

Использование металлического затвора в процессорах Penryn «сломало» эту традицию; эта технология позволяет улучить эффективность и снизить токи неконтролируемой утечки, поскольку проводимость металлического затвора существенно выше (рис. 1.22, *б*).

Технология медных проводников. Транзисторы на поверхности чипа — сложная комбинация из кремния, металлов и микродобавок, точно расположенных, чтобы образовать миллионы крохотных переключателей. Поскольку создавались все мéньшие и быстрые транзисторы, упакованные все плотнее, их соединение между собой стало превращаться в проблему.

Для установления соединений длительное время использовался алюминий, однако к середине 1990-х гг. стало очевидным, что скоро будут достигнуты технологические и физические пределы существующей технологии. Относительно высокое удельное сопротивление алюминия при уменьшении диаметра проводников приводит к потерям и перегреву схем. Однако длительное время никому не удавалось создать конкурентоспособный чип с медными проводниками.

Основное преимущество медных соединений в данном случае заключается в том, что медь обладает меньшей удельной проводимостью по сравнению с алюминием. При уменьшении площади сечения проводников (с уменьшением размера транзисторов) увеличивается и сопротивление проводников. Кроме того, медные проводники способны выдерживать значительно большую плотность тока, чем алюминиевые, и к тому же обладают более высокой устойчивостью к разрушению под воздействием тока, что позволяет продлить время жизни микросхемы.

Наряду с рассмотренными преимуществами медь обладает рядом свойств, создающих немало сложностей в процессе производства микросхем. Медь легко диффундирует в глубь кристалла, что вызывает порчу микросхемы и, в отличие от алюминия, плохо поддается травлению, поэтому технологии создания медных и алюминиевых внутрислойных соединений в корне различаются. В случае использования алюминия травлению по маске подлежит собственно алюминий, а при применении меди травлению подлежит оксидная пленка, в результате этого образуются бороздки, которые впоследствии заполняются медью. Эта технология получила название *Damascus*, или *узорная инкрустация*. Поэтому процесс изготовления микросхем с использованием алюминиевых соединений технологически не-

совместим с аналогичным процессом с использованием медных соединений.

В сентябре 1998 г. IBM объявила о разработке нового технологического процесса, включающего создание медных проводников на чипе (Damascene процесс — 0,18 мкм CMOS 7SF). Создание каждого нового слоя начинается с получения оксидной пленки, которая покрывается слоем фоторезиста. Далее, посредством литографического процесса, в оксидной пленке вытравливаются бороздки и углубления требуемой формы. Эти бороздки и углубления необходимо заполнить медью. Но прежде, для предотвращения нежелательной диффузии меди, они заполняются тонким слоем антидиффузионного вещества (diffusing barrier), изготовленного из устойчивого материала — титана или нитрида вольфрама. Толщина такой антидиффузионной пленки — всего 10 нм. Микроскопическая начальная пленка меди размещается выше, чтобы удерживать медный слой, который затем наносится на весь чип (рис. 1.23).

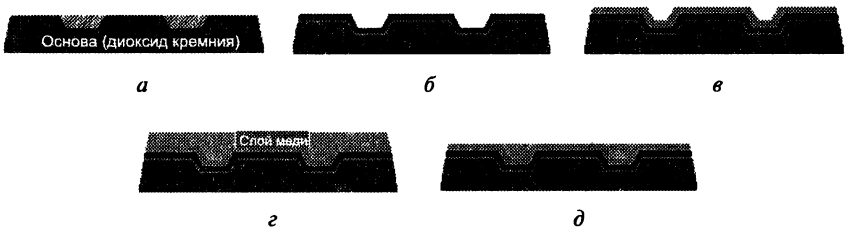
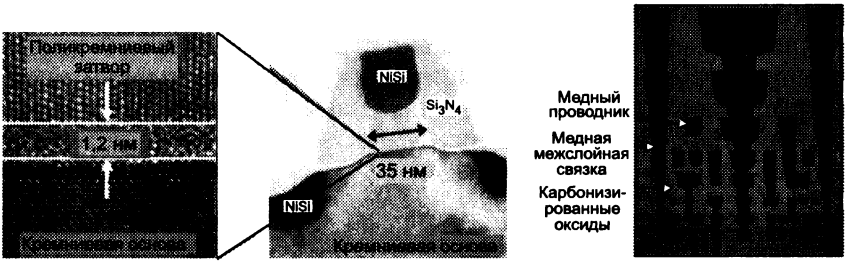


Рис. 1.23. Технология медных проводников:
а — вытравливание соединений путем фотолитографии; *б* — нанесение защитного слоя; *в* — нанесение микроскопической пленки меди; *г* — нанесение рабочего слоя меди; *д* — удаление избыточного металла

Для осаждения меди используют гальванизацию из раствора медного купороса Cu_2SO_4, причем сама пластина, на которую осаждаются ионы меди Cu^{++}, выступает в роли катода. При гальванизации необходимо, чтобы медь равномерно осаждалась по всей пластине, поэтому подбирают такую плотность электролита, чтобы минимизировать разницу тока в центре и по краям и тем самым обеспечить равномерность осаждения меди. При электролизе происходит постепенное заполнение атомами меди вытравленных канавок, в результате этого образуются проводящие «рельсы». После заполнения медью канавок лишний слой

меди удаляется с пластины посредством шлифования, а затем наносится очередной слой оксидной пленки и проводится формирование следующего слоя. В результате образуется многослойная система.

Технологический процесс 65 нм. Intel довела данную технологию до стадии промышленного производства к концу 2005 г. В 65-нм процессе Intel использует УФ-литографию с длиной волны 193 нм, комбинируемую с технологией фазового сдвига. При этом удалось уменьшить до 35 нм эффективную ширину затвора транзисторов (рис. 1.24, *а*), что приблизительно на 30 % меньше, чем при производстве по технологии 90 нм.



а *б*

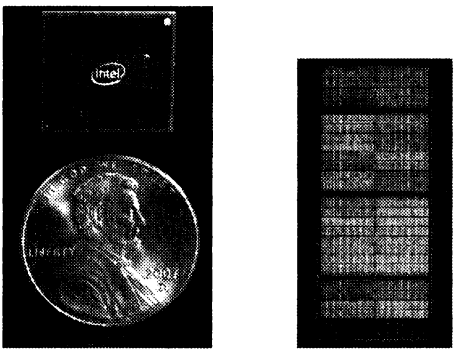
Рис. 1.24. Транзисторы поколения 65 нм (*а*); восемь слоев медных соединений (*б*)

Остались прежними в новом процессе и используемые для создания транзисторов материалы. Дополнительные усилия были направлены на борьбу с токами утечки. Появившаяся в 90-нм технологическом процессе технология напряженного кремния обрела в 65-нм технологии свою усовершенствованную версию — при сохранении толщины изоляционного слоя затвора на уровне 1,2 нм примерно на 15 % увеличилась деформация каналов транзисторов. Это дало четырехкратное уменьшение токов утечки, которое в конечном итоге создает возможность примерно 30%-ного увеличения частоты срабатывания транзисторов без возрастания их тепловыделения.

И последнее изменение — увеличение числа слоев медных соединений. В новом процессе их восемь, что на один больше, чем в ядрах, выпускаемых по 90-нм процессу (рис. 1.24, *б*). Благодаря этому Intel надеется упростить проектирование будущих кристаллов.

Технологический процесс 45 нм. Для того чтобы реализовать этот техпроцесс, разработчикам пришлось изменить одну из фундаментальных характеристик — материалы, применяемые в производстве. Вместо используемого с 1960-х гг. оксида кремния в качестве диэлектрика взят силицид гафния — изолятор с высокой диэлектрической проницаемостью (high-k). Затвор транзистора выполнен из металла вместо поликристаллического кремния (см. рис. 1.22). Благодаря этому удалось решить ключевую проблему — миниатюризация транзисторов привела к невозможности дальнейшего эффективного использования оксида кремния в качестве диэлектрика. Попутно существенно (в 5—10 раз) были снижены токи утечек. Важно, что для производства по 45-нм процессу также применяются 300-мм кремниевые пластины и 193-нм УФ-литография, что означает отсутствие необходимости модернизаций фабричных конвейеров.

По технологии 45 нм фирма Intel выпускает процессоры Wolfdale, Kentsfield, Atom (рис. 1.25, *а*), AMD — процессорные ядра Shanghai, Propus, Regor, Deneb (серия Stars/«Звезды»). Процесс 45 нм не является технологическим пределом — в настоящее время появились образцы изделий, выпущенных по технологиям 32 и даже 22 нм. На рис. 1.25, *б* представлен кристалл статической (SRAM) оперативной памяти, изготовленной по технологии 32 нм (2007 г., фирма IBM с партнерами). Память реализована по традиционной схеме (6 транзисторов на триггер, см. рис. 4.9), и плата содержит более чем 1,6 млрд транзисторов.



а *б*

Рис. 1.25. Образцы нанотехнологий:
а — процессор Intel Atom N270 (45 нм, размер 13 × 14 мм, диаметр монеты — 19 мм); *б* — чип оперативной памяти SRAM (технология 32 нм)

Печатные платы

Плата, или *printed circuit board*, — изоляционная пластина, на которой устанавливаются и соединяются друг с другом электронные элементы, перечисленные выше, и приборы меньшей степени интеграции — отдельные транзисторы, резисторы, конденсаторы и др.

Печатная плата изготавливается из пластмассы, гетинакса, текстолита либо другого изолятора (керамика). На плате с одной либо обеих сторон размещаются интегральные схемы, резисторы, диоды и другие полупроводниковые приборы. Для их соединения на поверхности платы наносятся тонкие электропроводящие полоски. Печатная плата может быть двух- либо многослойной.

Существует несколько технологий монтажа элементов (в том числе и интегральных схем) на печатных платах. Наиболее старая из них — монтаж в сквозные отверстия. Здесь элементы создаваемой схемы устанавливаются с одной стороны платы. Вслед за этим появился способ укладки интегральных схем прямо на поверхности этой платы. Вначале интегральные схемы припаивались к печатным платам. Теперь все чаще они приклеиваются без использования припоя. Малая высота интегральных схем, монтируемых на поверхность, позволяет устанавливать их на обеих сторонах платы.

Печатные платы перестают быть только плоскими. Происходит переход от двух измерений к криволинейным поверхностям и созданию печатных дорожек на геометрически изогнутых формах. Все это связано с тем, что по мере усложнения электронных компонентов становится все трудней размещать плоские платы в корпусы, удовлетворяющие требованиям потребителя. Для изготовления основы трехмерных печатных плат используется пластмасса, пригодная для литья.

Для решения задач трассировки соединений на печатных платах применяются средства автоматизированного проектирования (САПР) — программы-трассировщики.

1.5. Алгоритмы и программы

Понятие *алгоритма* является одним из основных в современной науке и практике. Еще на самых ранних ступенях развития математики (Древний Египет, Вавилон, Греция) в ней

стали рассматриваться различные вычислительные процессы чисто механического характера. С их помощью искомые величины ряда задач вычислялись последовательно из исходных величин по определенным правилам и инструкциям. Со временем все такие процессы в математике получили название алгоритмов (алгорифмов).

Алгоритм есть совокупность четко определенных правил, процедур или команд, обеспечивающих решение поставленной задачи за конечное число шагов.

Термин *алгоритм* происходит от имени средневекового узбекского математика Аль-Хорезми, который еще в IX в. (825 г.) дал правила выполнения четырех арифметических действий в десятичной системе счисления. Процесс выполнения арифметических действий был назван *алгоризмом*.

С 1747 г. вместо слова *алгоризм* стали употреблять *алгорисмус*, смысл которого состоял в комбинировании четырех операций арифметического исчисления — сложения, вычитания, умножения, деления.

К 1950 г. *алгорисмус* стал *алгорифмом*. Смысл алгорифма чаще всего связывался с алгорифмами Евклида — процессами нахождения наибольшего общего делителя двух многочленов, наибольшей общей меры двух отрезков и т. п.

Способы записи алгоритмов

Алгоритм должен быть понятен (доступен) пользователю и/или машине. Доступность пользователю означает, что он обязан отображаться посредством конкретных формализованных изобразительных средств, понятных пользователю. В качестве таких изобразительных средств используются следующие способы их записи:

- словесный;
- формульный;
- табличный;
- операторный;
- графический;
- язык программирования.

При словесном способе записи содержание последовательных этапов алгоритма описывается в произвольной форме на естественном языке.

Формульный способ основан на строго формализованном аналитическом задании необходимых для исполнения действий.

Табличный способ подразумевает отображение алгоритма в виде таблиц, использующих аппарат реляционного исчисления и алгебру логики для задания подлежащих исполнению взаимных связей между данными, содержащимися в таблице.

Операторный способ базируется на использовании для отображения алгоритма условного набора специальных операторов: арифметических, логических, печати, ввода данных и т. д.; операторы снабжаются индексами и между ними указываются необходимые переходы, а сами индексированные операторы описываются чаще всего в табличной форме.

Графическое отображение алгоритмов в виде блок-схем — весьма наглядный и распространенный способ. Графические символы, отображающие выполняемые процедуры, стандартизованы. Наряду с основными символами используются и вспомогательные, поясняющие процедуры и связи между ними.

Алгоритмы могут быть записаны и в виде команд какого-либо языка программирования. Если это макрокоманды, то алгоритм читаем и пользователем-программистом, и вычислительной машиной, имеющей транслятор с соответствующего языка.

Приведем пример словесного представления алгоритма на примере нахождения произведения n натуральных чисел ($c = n! = 1 \times 2 \times 3 \times 4 \times \ldots \times n$).

Этот процесс может быть записан в виде следующей системы последовательных указаний (пунктов):

1. Полагаем c равным единице и переходим к следующему пункту.

2. Полагаем i равным единице и переходим к следующему пункту.

3. Полагаем c равным $c = c \times i$ и переходим к следующему указанию.

4. Проверяем, равно ли i числу n. Если $i = n$, то вычисления прекращаем. Если $i < n$, то увеличиваем i на единицу и переходим к пункту 3.

Классификация и свойства алгоритмов

Алгоритмы, в соответствии с которыми решение поставленных задач сводится к арифметическим действиям, называются *численными алгоритмами.*

Алгоритмы, в соответствии с которыми решение поставленных задач сводится к логическим действиям, называются *логическими алгоритмами.* Примерами логических алгоритмов могут служить алгоритмы поиска минимального числа, поиска пути на графе, поиска пути в лабиринте и др.

Алгоритмом является последовательность четких однозначных указаний, которые, будучи применены к определенным имеющимся данным, обеспечивают получение требуемого результата. *Данными* называют все величины, участвующие в решении задачи. Данные, известные перед выполнением алгоритма, являются начальными, *исходными данными.* Результат решения задачи — это конечные, *выходные данные.*

Каждое указание алгоритма предписывает исполнителю выполнить одно конкретное законченное действие. Исполнитель не может перейти к выполнению следующей операции, не закончив полностью выполнения предыдущей. Предписания алгоритма надо выполнять последовательно одно за другим, в соответствии с указанным порядком их записи. Выполнение всех предписаний гарантирует правильное решение задачи.

Поочередное выполнение команд алгоритма за конечное число шагов приводит к решению задачи, к достижению цели. Разделение выполнения решения задачи на отдельные операции (выполняемые исполнителем по определенным командам) — важное свойство алгоритмов, называемое *дискретностью.*

Для того чтобы алгоритм мог быть выполнен, нельзя включать в него команды, которые исполнитель не в состоянии исполнить. У каждого исполнителя имеется свой перечень команд, которые он может выполнить. Совокупность команд, которые могут быть выполнены исполнителем, называется *системой команд исполнителя.*

Каждая команда алгоритма должна определять однозначное действие исполнителя. Такое свойство алгоритмов называется *определенностью (или точностью) алгоритма.*

Алгоритм, составленный для конкретного исполнителя, должен включать только те команды, которые входят в его систему команд. Это свойство алгоритма называется *понятностью.* Алго-

ритм не должен быть рассчитан на принятие каких-либо самостоятельных решений исполнителем, не предусмотренных составлением алгоритма.

Еще одно важное требование, предъявляемое к алгоритмам, — *результативность (или конечность)* алгоритма. Оно означает, что исполнение алгоритма должно закончиться за конечное число шагов.

Поскольку разработка алгоритмов — процесс творческий, требующий умственных усилий и затрат времени, предпочтительно разрабатывать алгоритмы, обеспечивающие решения всего класса задач данного типа. Например, если составляется алгоритм решения кубического уравнения $ax^3 + bx^2 + cx + d = 0$, то он должен быть *вариативен*, т. е. обеспечивать возможность решения для любых допустимых исходных значений коэффициентов a, b, c, d. Про такой алгоритм говорят, что он удовлетворяет требованию *массовости*. Свойство массовости не является необходимым свойством алгоритма. Оно, скорее, определяет качество алгоритма; в то же время свойства точности, понятности и конечности являются необходимыми (иначе это не алгоритм).

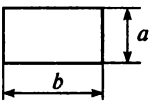
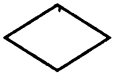
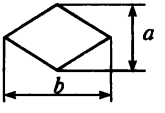
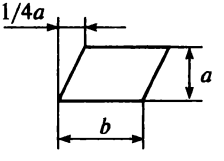
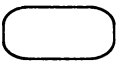
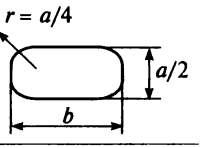
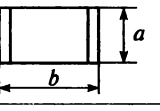
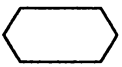
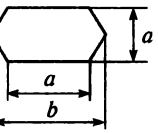
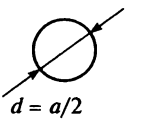
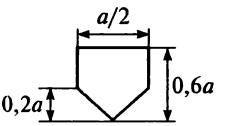
Запись алгоритмов в виде блок-схем

Алгоритмы можно записывать по-разному. Форма записи, состав и количество операций алгоритма зависят от того, кто будет исполнителем этого алгоритма. Если задача решается с помощью ЭВМ, алгоритм решения задачи должен быть записан в понятной для машины форме, т. е. в виде программы.

Схема алгоритма — графическое представление алгоритма, дополняемое элементами словесной записи. Каждый пункт алгоритма отображается на схеме некоторой геометрической фигурой или блоком. При этом правило выполнения схем алгоритмов регламентирует ГОСТ 19.002—80 «Единая система программной документации» (табл. 1.28).

Блоки на схемах соединяются линиями потоков информации. Основное направление потока информации идет сверху вниз и слева направо (стрелки могут не указываться), снизу вверх и справа налево — стрелка обязательна. Количество входящих линий для блока не ограничено. Выходящая линия — одна, за исключением логического блока.

Таблица 1.28. **Основные элементы блок-схем**

№	Символ	Наименование	Содержание	Размеры по ГОСТ 19.003—80 (ЕСПД): $a = 10, 15, 20$ мм; $b = 1{,}5a$
1	$a := b$ $b := c$	Блок вычислений	Вычислительные действия или последовательность действий	
2		Логический блок	Выбор направления выполнения алгоритма в зависимости от некоторого условия	
3		Блок ввода-вывода данных	1. Общие обозначения ввода (вывода) данных (вне зависимости от физического носителя). 2. Вывод данных, носителем которых является документ	
4		Начало (конец)	Начало или конец алгоритма, вход в программу или выход из нее	
5		Процесс пользователя (подпрограмма)	Вычисление по стандартной программе или подпрограмме	
6		Блок модификации	Функция выполняет действия, изменяющие пункты (например, заголовок цикла) алгоритма	
7		Соединитель	Указание связи прерванными линиями между потоками информации в пределах одного листа	
8		Межстраничное соединение	Указание связи между информацией на разных листах	

Базовые структуры алгоритмов

Это определенный набор блоков и стандартных способов их соединения для выполнения типичных последовательных действий. К основным структурам относятся следующие — линейные, разветвляющиеся, циклические (рис. 1.26).

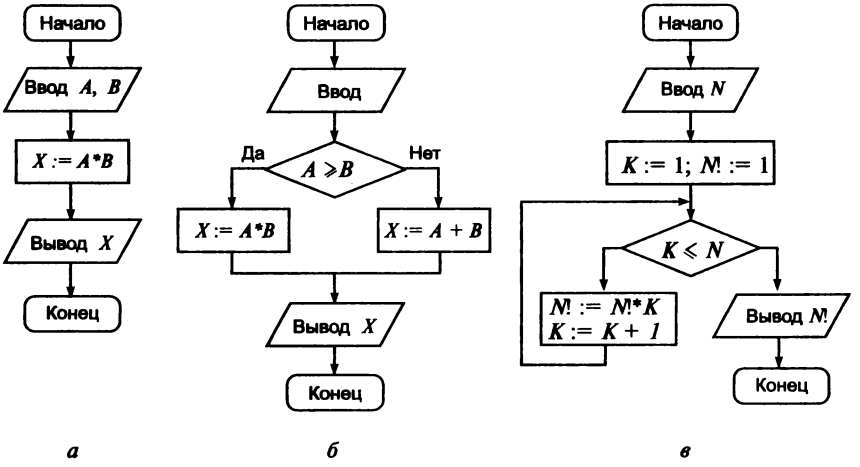


Рис. 1.26. Примеры структур алгоритмов:
а — линейный алгоритм; *б* — алгоритм с ветвлением; *в* — алгоритм с циклом

Линейными называются алгоритмы, в которых действия осуществляются последовательно друг за другом. Стандартная блок-схема линейного алгоритма приводится на рис. 1.26, *а* (вычисление суммы двух чисел — *A* и *B*).

Разветвляющимся называется алгоритм, который, в отличие от линейных алгоритмов, содержит условие, в зависимости от истинности или ложности которого выполняется та или иная последовательность команд. Таким образом, команда ветвления состоит из условия и двух последовательностей команд.

Примером может являться разветвляющийся алгоритм, изображенный в виде блок-схемы (рис. 1.26, *б*). Аргументами этого алгоритма являются две переменные *A, B*, а результатом — переменная *X*. Если условие $A \geq B$ истинно, то выполняется операция $X := A \times B$, в противном случае выполняется $X := A + B$. В результате печатается то значение переменной *X*, которое она получает при выполнении одной из серий команд.

Циклическим называется алгоритм, в котором некоторая последовательность операций (тело цикла) выполняется многократно. Однако «многократно» не означает «до бесконечности». Организация циклов, никогда не приводящая к остановке в выполнении алгоритма, является нарушением требования его результативности — получения результата за конечное число шагов.

В цикл в качестве базовых входят — блок проверки условия и тело цикла. Перед операцией цикла осуществляется начальное присвоение значений тем переменным, которые используются в теле цикла. Если тело цикла расположено после проверки условий *P* (цикл с предусловием), то может случиться так, что при определенных условиях тело цикла не выполнится ни разу. Такой вариант организации цикла, управляемый предусловием, называется *цикл «ПОКА»/«WHILE»* (здесь условие — это условие на продолжение цикла).

Возможен другой случай, когда тело цикла выполняется, по крайней мере, один раз и будет повторяться до тех пор, пока не станет истинным условие. Такая организация цикла, когда его тело расположено перед проверкой условия, носит название цикла с постусловием, или *цикла «ДО»/«FOR»*. Истинность условия в этом случае — условие окончания цикла. Отметим, что возможна ситуация с постусловием и при организации цикла «ПОКА». Итак, цикл «ДО» завершается, когда условие становится истинным, а цикл «ПОКА» — когда становился ложным. Современные языки программирования имеют достаточный набор операторов, реализующих как цикл «ПОКА», так и цикл «ДО».

Рассмотрим пример алгоритма вычисления факториала, изображенный на рис. 1.26 (с циклом «ПОКА»). Переменная *N* получает значение числа, факториал которого вычисляется. Переменной *N*!, которая в результате выполнения алгоритма должна получить значение факториала, присваивается первоначальное значение 1. Переменной *K* также присваивается значение 1. Цикл будет выполняться, пока справедливо условие $N \geq K$. Тело цикла состоит из двух операций $N! = N! \times K$ и $K = K + 1$.

Циклические алгоритмы, в которых тело цикла выполняется заданное число раз, реализуются с помощью цикла со счетчиком. Цикл со счетчиком реализуется с помощью команды повторения.

Процесс решения сложной задачи довольно часто сводится к решению нескольких более простых подзадач. Соответственно при разработке сложного алгоритма он может разбиваться на от-

дельные алгоритмы, которые называются вспомогательными. Каждый такой вспомогательный алгоритм описывает решение какой-либо подзадачи.

Процесс построения алгоритма методом последовательной детализации состоит в следующем. Сначала алгоритм формулируется в «крупных» блоках (командах), которые могут быть непонятны исполнителю (не входят в его систему команд) и записываются как вызовы вспомогательных алгоритмов. Затем происходит детализация, и все вспомогательные алгоритмы подробно расписываются с использованием команд, понятных исполнителю.

Контрольные вопросы

1. Дайте классификацию информации.
2. Каковы преимущества цифровой информации по отношению к аналоговой?
3. Перечислите методы кодирования символов.
4. Перечислите методы кодирования численной информации.
5. Переведите число $32\,451_{10}$ в шестнадцатеричную и восьмеричную системы счисления.
6. Переведите число $32\,451_{16}$ в десятичную и восьмеричную системы счисления.
7. В чем заключаются особенности двоичной арифметики?
8. Подсчитайте произведение $1FA_{16}$ и $2BC_{16}$ по модулю 8.
9. Подсчитайте сумму 457_8 и 375_8 по модулю 3.
10. Перечислите логические элементы ЭВМ.
11. Что такое логические узлы ЭВМ?
12. Составьте таблицы истинности для левого $(\neg(A \wedge B))$ и правого $(\neg A \vee \neg B)$ выражений 1-го закона де Моргана. Проверьте их на соответствие.
13. Составьте таблицы истинности для левого $(\neg(A \vee B))$ и правого $(\neg A \vee \neg B)$ выражений 2-го закона де Моргана. Проверьте их на соответствие.
14. Последний столбец таблицы истинности для двухместных операций, очевидно, может содержать $16 = 2^4$ различных сочетаний «1» и «0». Следовательно, всего может быть определено 16 логических операций над двумя переменными, из которых нами рассмотрены только пять. Составьте таблицу истинности для одной из 9 оставшихся вне рассмотрения функций и попытайтесь построить логическое выражение для этой функции.
15. Перечислите базовые структуры алгоритмов и программ.

Глава 2
АРХИТЕКТУРА И СТРУКТУРА ЭЛЕКТРОННЫХ ВЫЧИСЛИТЕЛЬНЫХ МАШИН И СИСТЕМ

Прежде всего, следует определить основные объекты рассмотрения. Электронные вычислительные машины и системы включают:

- обычные вычислительные машины (однопроцессорные);
- суперкомпьютеры (многопроцессорные ЭВМ, иногда объединяются в один класс с вычислительными системами);
- вычислительные системы (обычно — комплексы ЭВМ), в том числе многопроцессорные машины.

Таким образом, в целом следует рассматривать все множество современных *вычислительных машин и систем*. На рис. 2.1 приводится перечень основных компонентов ЭВМ и ВС, рассматриваемых в настоящем учебнике.

Рассматривая архитектуру ЭВМ, вычислительных систем и суперкомпьютеров с общих позиций и абстрагируясь от деталей, можно воспользоваться следующей схемой (рис. 2.2), которая включает:

- *процессоры и системы памяти* — вычислительно-информационная среда;
- *средства коммутации и коммуникации* — коммуникационно-коммутационная среда.

Все эти компоненты активно присутствуют как в ЭВМ, так и в вычислительных сетях и системах (суперЭВМ).

Рассмотрим последовательно состав, структуру, архитектуру и классификацию ЭВМ и вычислительных комплексов.

В табл. 2.1 перечислены основные уровни архитектур ЭВМ, ВС, других компонентов систем и машин, о которых далее будет идти речь в настоящем учебнике.

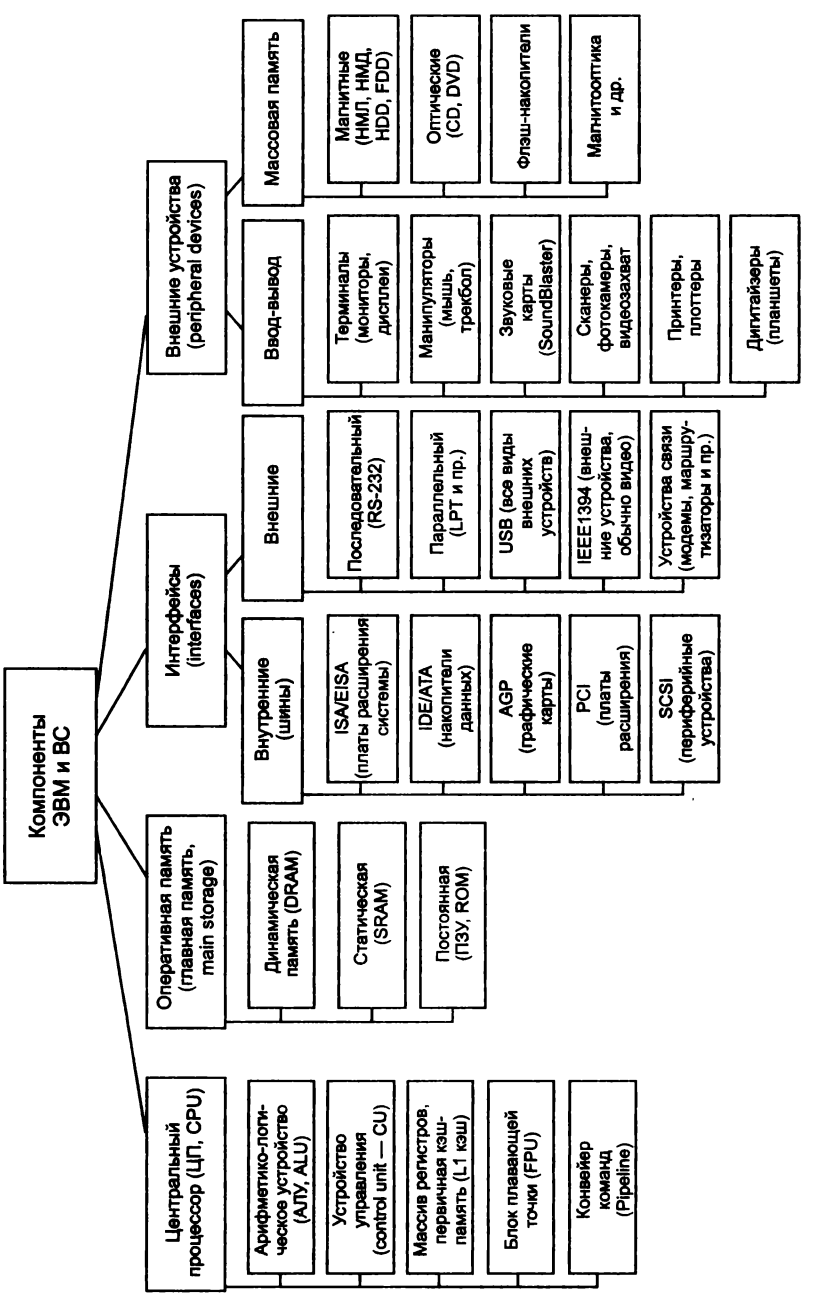


Рис. 2.1. Основные компоненты (блоки) ЭВМ и систем

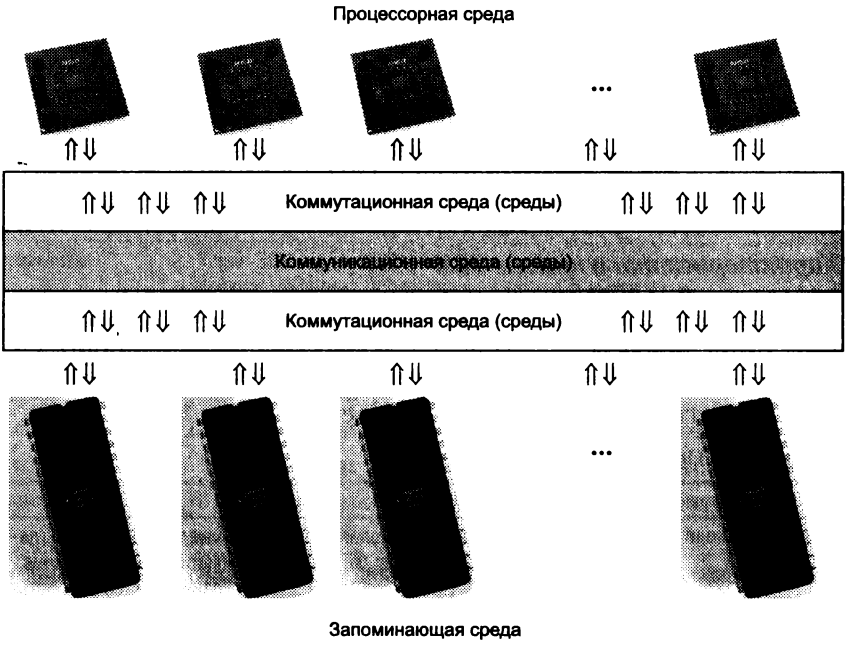


Рис. 2.2. Обобщенное представление об архитектурах ЭВМ и вычислительных систем

Таблица 2.1. **Разновидности архитектур вычислительных машин и устройств**

Тип архитектуры	Характеристика	Примеры архитектур
ЭВМ в целом	Взаимосвязь компонентов	Иерархическая, магистральная
Центральное устройство	Использование оперативной памяти	Принстонская, гарвардская
Вычислительная система	Взаимосвязь компонентов	Кластерная, массивно-параллельная
Микроархитектура процессора	Обработка микроопераций, кэш-память, конвейеры	AMD K7-K8, Intel P5-P6
Система команд	Команды процессора	IA-32, IA-64, AMD64
Микроархитектура чипсет	Взаимосвязь компонентов системной платы ПК	«Северный мост — Южный мост»

2.1. Классы вычислительных машин и систем

Электронная вычислительная машина (ЭВМ, компьютер) представляет собой комплекс технических средств, предназначенный для автоматической обработки информации в процессе решения вычислительных и информационных задач.

Характеристики и классы ЭВМ

Основные характеристики, используемые для классификации ЭВМ, приведены в табл. 2.2. В табл. 2.3 даны различные основания для классификации ЭВМ, некоторые из которых, а также ряд других более подробно изложены далее.

Таблица 2.2. **Основные характеристики ЭВМ**

Характеристика	Содержание
Быстродействие	Способность ЭВМ выполнять определенные типы операций (пересылка данных между регистрами, суммирование, сравнение) за единицу времени. Быстродействие ограничивается скоростью протекания переходных процессов в элементной базе и обычно характеризуется тактовой частотой внешнего генератора, которая согласует действия устройств и узлов
Производительность	Способность ЭВМ обрабатывать некоторый тестовый массив различных команд (так называемые смеси Гибсона) за единицу времени. Производительность во многом зависит от применяемых архитектурных решений
Разрядность машинного слова	Влияет на диапазон представимых в ЭВМ чисел, адресность системы команд, скорость пересылки данных
Максимально возможный размер адресного пространства	Определяет максимально доступный объем оперативной памяти (ОП), возможности по ее виртуальному расширению
Количество групп команд и команд в группах	Характеризует систему команд, их адресность и пр. характеристики
Количество способов адресации команд и данных	Включает прямую, косвенную и прочие типы адресации команд и данных
Тип используемого интерфейса (сопряжения) ядра ЭВМ с периферией	Шины или каналы связи, характеризующиеся разрядностью (шириной параллельного кода, передаваемого единовременно) и частотой, с которой эта передача осуществляется
Надежность	Измеряется средней наработкой на отказ (средним временем безотказной работы), скоростью восстановления и пр.
Стоимость	Очевидная экономическая характеристика
Потребляемая мощность	Электрическая мощность в ваттах (Вт), киловаттах (кВт)

Таблица 2.3. **Различные подходы к классификации ЭВМ**

Тип ЭВМ	Особенности класса
Физический способ представления информации	
Аналоговые (АВМ)	Информация представляется в виде непрерывно изменяющейся во времени аналоговой величины (напряжения или тока)
Цифровые (ЦВМ)	Информация представляется в виде специальных кодов, в принятой для данной ЭВМ системе исчисления
Гибридные	Такие ЭВМ, например, имеют аналоговый вход информации, затем следует цифровая обработка и аналоговый выход
Поколения ЭВМ	
I поколение	1950—1958, построены на лампах
II поколение	1959—1967 — на транзисторах и печатных платах
III поколение	1968—1978 — на микросхемах малой степени миниатюризации
IV поколение	1979—1993 — на микросхемах большой степени миниатюризации
V поколение	С 1994 — на микросхемах сверхбольшой степени миниатюризации
Назначение ЭВМ	
Вычислительные системы	Характеризуются относительно небольшими объемами входной и выходной информации и сложными алгоритмами ее обработки. Такие ЭВМ должны иметь высокую производительность и небольшое количество устройств ввода-вывода
Системы обработки данных	Характеризуются большим количеством внешних запоминающих устройств, способных хранить большой объем информации и сравнительно несложными алгоритмами обработки этой информации
Управляющие ЭВМ	Предназначены для управления в реальном масштабе времени объектами и производственными процессами, поэтому для связи с объектами управления ЭВМ снабжаются преобразователями, датчиками и т. д., которые устанавливаются в контуре управления
Способ организации вычислительного процесса	
Однопрограммные	Способны одновременно выполнять не более одной задачи
Многопрограммные	Многопрограммные ЭВМ могут работать в однопрограммном и мультипрограммных режимах. Эти режимы должны поддерживаться соответствующими операционными системами (ОС)
Область применения	
Универсальные	Ориентированы на решение широкого круга задач путем применения соответствующего программного обеспечения (ПО). Как правило, разработку таких программ осуществляют по заказам извне. Характерные особенности универсальных ЭВМ: наличие высокопроизводительного процессора; большой объем памяти; соответствующая ОС
Специализированные	Предназначены для решения определенного круга задач и не требуют, как правило, разработки нового программного обеспечения. Специализированные ЭВМ проще и дешевле универсальных ЭВМ

Рассмотрим основные классы ЭВМ.

Физическое представление обрабатываемой информации. Здесь выделяют:

- АВМ — аналоговые вычислительные машины, или вычислительные машины непрерывного действия, которые работают с информацией, представленной в непрерывной (аналоговой) форме, т. е. в виде непрерывного ряда значений какой-либо физической величины (чаще всего электрического напряжения);
- ЦВМ — цифровые вычислительные машины, или вычислительные машины дискретного действия, которые обрабатывают информацию, представленную в дискретной, а точнее, цифровой форме. В силу универсальности цифровой формы представления информации ЭВМ является универсальным инструментом обработки данных;
- ГВМ — гибридные вычислительные машины, или вычислительные машины комбинированного действия, работают с информацией, представленной как в цифровой, так и в аналоговой форме. Они совмещают в себе достоинства АВМ и ЦВМ. ГВМ целесообразно использовать для решения задач управления сложными быстродействующими техническими комплексами.

Поколения ЭВМ. Идея делить машины на поколения вызвана к жизни тем, что за время недолгой истории развития компьютерная техника прошла большой путь развития как в аспекте элементной базы (лампы, транзисторы, микросхемы и др.), так и с точки зрения изменения структуры, появления новых возможностей, расширения областей применения и характера использования (табл. 2.4).

К первому поколению обычно относят машины, созданные на рубеже 50-х гг. и базирующиеся на электронных лампах. Лампы потребляли значительное количество электроэнергии и выделяли много тепла (рис. 2.3, *а*). Эти ЭВМ были громоздкими и слишком дорогими и их могли приобретать только крупные корпорации и правительства.

Набор команд был ограничен, схемы арифметико-логического устройства и устройства управления достаточно просты, программное обеспечение практически отсутствовало. Показатели объема оперативной памяти и быстродействия были низкими. Для ввода-вывода использовались перфоленты, перфокарты,

Таблица 2.4. **Этапы развития компьютерных информационных технологий**

Параметр	Период, годы				
	50-е	60-е	70-е	80-е	Настоящее время
Цель использования компьютера	Научно-технические расчеты	Технические и экономические расчеты	Управление и экономические расчеты	Управление, предоставление информации	Телекоммуникации, информационное обслуживание
Режим работы компьютера	Однопрограммный	Пакетная обработка	Разделение времени	Персональная работа	Сетевая обработка
Интеграция данных	Низкая	Средняя	Высокая	Очень высокая	Сверхвысокая
Расположение пользователя	Машинный зал	Отдельное помещение	Терминальный зал	Рабочий стол	Произвольное мобильное
Тип пользователя	Инженеры-программисты	Профессиональные программисты	Программисты	Пользователи с общей компьютерной подготовкой	Малообученные пользователи
Тип диалога	Работа за пультом компьютера	Обмен перфоносителями и машинограммами	Интерактивный (через клавиатуру и экран)	Интерактивный с жестким меню	Интерактивный экранный типа «вопрос — ответ»

магнитные ленты и печатающие устройства. Быстродействие составляло порядка 10—20 тыс. операций в секунду.

Программы для этих машин писались на языке конкретной машины. Математик, составивший программу, садился за пульт управления машины, вводил и отлаживал программы и производил по ним счет. Процесс отладки был весьма длительным по времени.

Несмотря на ограниченность возможностей, эти машины позволяли выполнять сложнейшие расчеты, необходимые для прогнозирования погоды, решения задач атомной энергетики и др.

Опыт использования машин первого поколения показал, что существует огромный разрыв между временем, затрачиваемым на разработку программ, и временем счета. Эти проблемы начали преодолевать путем интенсивной разработки средств автоматизации программирования, создания систем обслуживающих

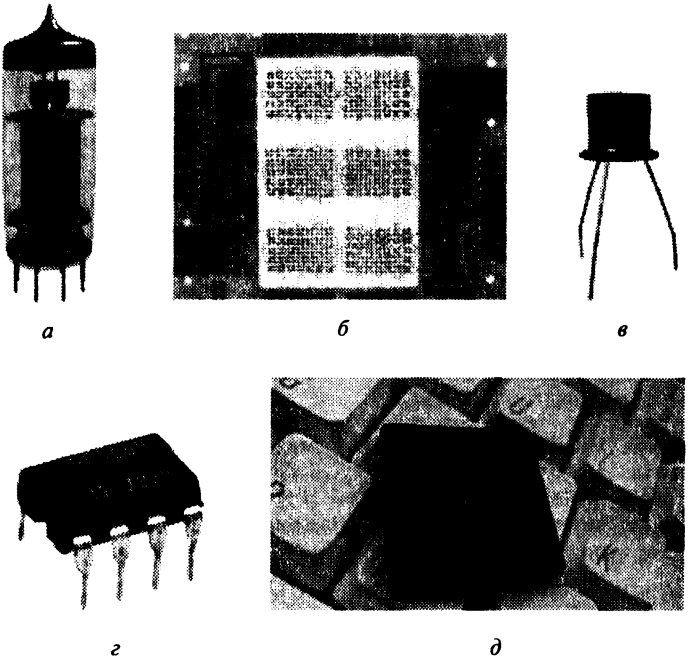


Рис. 2.3. Электронная лампа (*а*); память на магнитных сердечниках (*б*); транзистор (*в*); одна из первых интегральных схем (*г*); микропроцессор Motorola 68000 (*д*)

программ, упрощающих работу на машине и увеличивающих эффективность ее использования. Это, в свою очередь, потребовало значительных изменений в структуре компьютеров, направленных на то, чтобы приблизить ее к требованиям, возникшим из опыта эксплуатации компьютеров.

Отечественные машины первого поколения: МЭСМ (малая электронная счетная машина), БЭСМ, Стрела, Урал, М-20.

Второе поколение компьютерной техники — машины, сконструированные в 1955—65 гг. Характеризуются использованием в них как электронных ламп, так и дискретных транзисторных логических элементов (рис. 2.3). Их оперативная память была построена на магнитных сердечниках. В это время стал расширяться диапазон применяемого оборудования ввода-вывода, появились высокопроизводительные устройства для работы с магнитными лентами (НМЛ), магнитные барабаны (НМБ) и первые магнитные диски (табл. 2.5).

Таблица 2.5. **Основные характеристики отечественных ЭВМ второго поколения**

Параметр	Первая очередь					Вторая очередь			
	Раздан-2	БЭСМ-4	М-220	Урал-11	Минск-22	Урал-16	Минск-32	М-222	БЭСМ-6
Адресность	2	3	3	1	2	1	1 и 2	3	1
Форма представления данных	С плавающей запятой	С плавающей запятой	С плавающей запятой	С фиксированной запятой, символьная	С фиксированной запятой, символьная	С плавающей и фиксированной запятой, символьная	С плавающей и фиксированной запятой, символьная	С плавающей запятой, символьная	С плавающей запятой, символьная
Длина машинного слова (дв. разр.)	36	45	45	24	37	48	37	45	48
Быстродействие, оп/с	5 тыс.	20 тыс.	20 тыс.	14—15 тыс.	5 тыс.	100 тыс.	До 65 тыс.	27 тыс.	1 млн
ОЗУ, тип, емкость (слов)	Ферритовый сердечник, 2048	Ферритовый сердечник, 8192	Ферритовый сердечник, 4096—16 384	Ферритовый сердечник, 4096—16 384	Ферритовый сердечник, 8192	Ферритовый сердечник, 8192—65 536	Ферритовый сердечник, 16 384—65 636	Ферритовый сердечник, 16 384—32 768	Ферритовый сердечник, 32 768—131 071
ВЗУ, тип, емкость (слов)	НМЛ 120 тыс.	НМЛ 8 млн	НМЛ 16 млн	НМЛ 8 млн	НМЛ до 5 млн	НМЛ 12 млн НМБ 130 тыс.	НМЛ до 16 млн	НМЛ до 32 млн НМБ до 192 тыс.	НМЛ 32 млн НМБ 512 тыс.

Эти машины характеризуются быстродействием до сотен тысяч операций в секунду, емкостью памяти — до нескольких десятков тысяч слов.

Появляются языки высокого уровня, средства которых допускают описание всей необходимой последовательности вычислительных действий в наглядном, легко воспринимаемом виде.

Поскольку смысл программы, подготовленной на алгоритмическом языке, недоступен компьютеру, воспринимающему только язык своих собственных команд, специальные программы, которые называются трансляторами, должны переводить (транслировать) программу с алгоритмического на машинный язык.

Появился широкий набор библиотечных программ для решения разнообразных задач, а также мониторные системы, управляющие режимом трансляции и исполнения программ, из которых в дальнейшем выросли современные операционные системы.

Операционная система — важнейшая часть программного обеспечения компьютера, предназначенная для автоматизации планирования и организации процесса обработки программ, ввода-вывода и управления данными, распределения ресурсов, подготовки и отладки программ, других вспомогательных операций обслуживания.

Машинам второго поколения была свойственна программная несовместимость, которая затрудняла организацию крупных информационных систем. Поэтому в середине 60-х гг. наметился переход к созданию компьютеров, программно совместимых и построенных на микроэлектронной технологической базе.

Машины третьего поколения — это семейства машин с единой архитектурой, т. е. программно совместимых. В качестве элементной базы в них используются интегральные схемы, которые также называются микросхемами (см. рис. 2.3, *г*).

Машины третьего поколения создавались примерно после 60-х гг., имели развитые операционные системы и обладали возможностями мультипрограммирования, т. е. параллельного выполнения нескольких программ. Многие задачи управления памятью, устройствами и ресурсами стала брать на себя операционная система или же непосредственно сама машина.

Примеры машин третьего поколения — семейства IBM-360, IBM-370, PDP-11, VAX, ЕС ЭВМ (Единая система ЭВМ), СМ ЭВМ (Семейство малых ЭВМ) и др.

Быстродействие машин внутри семейства изменяется от нескольких десятков тысяч до миллионов операций в секунду. Емкость оперативной памяти достигает нескольких сотен тысяч слов.

Четвертое поколение — это основной контингент современной компьютерной техники, разработанной после 70-х гг.

Наиболее важный в концептуальном отношении критерий, по которому эти компьютеры можно отделить от машин третьего поколения, состоит в том, что машины четвертого поколения проектировались в расчете на эффективное использование современных высокоуровневых языков и упрощение процесса программирования для конечного пользователя.

В аппаратурном отношении для них характерно широкое использование интегральных схем в качестве элементной базы, а также наличие быстродействующих запоминающих устройств с произвольной выборкой емкостью в десятки мегабайт (см. рис. 2.3, *д*).

С точки зрения структуры машины этого поколения представляют собой многопроцессорные и многомашинные комплексы, работающие на общую память и общее поле внешних устройств. Быстродействие составляет до нескольких десятков миллионов операций в секунду, емкость оперативной памяти порядка 1—64 Мбайт.

Для них характерны:

- применение персональных компьютеров (ПК);
- телекоммуникационная обработка данных и компьютерные сети;
- широкое применение систем управления базами данных;
- элементы интеллектуального поведения систем обработки данных и устройств.

В компьютерах пятого поколения, предположительно, должен произойти качественный переход от обработки данных к обработке знаний.

Архитектура компьютеров будущего поколения будет содержать два основных блока. Один из них — это традиционный компьютер, однако лишенный связи с пользователем. Эту связь осуществляет интеллектуальный интерфейс. Будет также решаться проблема децентрализации вычислений с помощью компьютерных сетей.

Сферы применения и методы использования. Здесь ЭВМ можно разделить на следующие группы (рис. 2.4, табл. 2.6).

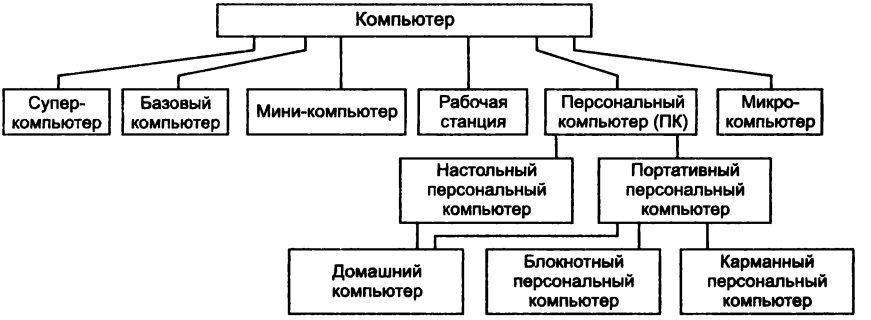
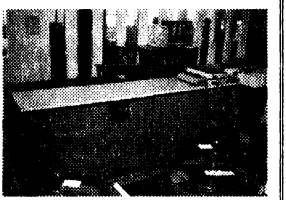
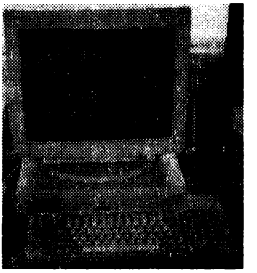
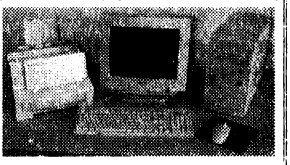


Рис. 2.4. Классификация по сферам применения и методам использования

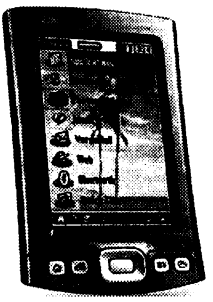
Таблица 2.6. **Классификация ЭВМ по сферам применения, методам использования и габаритным характеристикам**

Класс ЭВМ	Основное назначение	Основные технические данные	Общий вид ЭВМ
СуперЭВМ, суперкомпьютер, вычислительная система (ВС)	Высокоскоростное выполнение прикладных процессов	Имеет скалярные и векторные процессоры. Совместная работа процессоров основывается на различных архитектурах	 Roadrunner
Большие ЭВМ (мэйнфреймы — mainframe)	Обработка больших объемов данных крупных предприятий и организаций	Мультипроцессорная архитектура, позволяющая подключение нескольких сотен рабочих мест	 Honeywell-Bull DPS7 Mainframe
Мини-ЭВМ	Системы управления предприятиями	Однопроцессорная архитектура, разветвленная система периферийных устройств (ограниченные возможности, обработка слов меньшей длины и т. д.)	 PDP-11

Продолжение табл. 2.6

Класс ЭВМ	Основное назначение	Основные технические данные	Общий вид ЭВМ
Рабочие станции	Системы автоматизированного проектирования, системы автоматизации эксперимента, индустриальные процессы и др.	Высокое быстродействие процессора, емкость оперативного запоминающего устройства 32—64 Мбайт, специализированная система периферийных устройств	 Sun SPARCstation 1+
МикроЭВМ; настольный (desktop) персональный компьютер (ПК)	Индивидуальное обслуживание пользователей	Центральный блок с одним или несколькими процессорами, монитор, акустическая система, клавиатура, электронное перо с планшетом, устройство ввода информации, принтеры, жесткие диски, гибкие диски, магнитные ленты, оптические диски и пр.	 IBM PC-подобный ПК
Переносной ПК «наколенник» (laptop)	То же	Малогабаритный книжного размера портативный вариант стационарного персонального компьютера	 «Mid-range HP Laptop»
Блокнотный ПК, ноутбук (notebook)	» »	Модели могут иметь процессор Pentium, оперативную память до 96 Мбайт, жесткий диск до 9 Гбайт, встроенные компакт-диск и факс-модем, дисплей жидкокристаллический, время работы от собственного источника питания от 2 до 8 ч	 Sony VAIO SZ (Intel Core Duo)

Окончание табл. 2.6

Класс ЭВМ	Основное назначение	Основные технические данные	Общий вид ЭВМ
Карманный компьютер «наладонник» (palmtop)	Индивидуальное обслуживание пользователей	Оперативная память выполняет функцию долговременной памяти размером в несколько мегабайт. Жесткий диск отсутствует. Работает под управлением Windows CE, имеет интерфейс с другими компьютерами, встроенные интегрированные системы, жидкокристаллический дисплей	 PalmOne Tungsten T5

Суперкомпьютер (supercomputer) предназначен для высокоскоростного выполнения прикладных процессов. В 1976 г. корпорация Cray Research изготовила первый сверхбыстродействующий компьютер, давая начало новому классу компьютеров. Первоначально Cray Research предполагала, что потребность в таких компьютерах будет небольшой, однако она увеличивается и особенно в последние годы. Кроме этого, производители суперкомпьютеров постоянно улучшали показатель стоимость/производительность. Появился и получил большую популярность новый класс — супермини-компьютеры. Это уменьшенные по габаритам и более экономичные варианты суперкомпьютеров, нередко — настольного исполнения.

Суперкомпьютер может иметь один процессор, и тогда в нем одна последовательность команд работает с одним потоком данных. Вместе с этим большие скорости обработки данных можно получить лишь в многопроцессорных системах. Поэтому во всех последующих архитектурах степень параллельной обработки возрастает. Растет соответственно и число входящих в суперкомпьютер процессоров. В дополнение к обычным (скалярным) подключаются векторные процессоры. В первом случае обрабатываются скалярные величины, а во втором — векторные.

Внедрение суперкомпьютеров долго сдерживалось отсутствием развитого программного обеспечения. В настоящее время ситуация изменяется, появились языки, предназначенные для параллельной обработки, все больше предлагается эффективных

операционных систем. Суперкомпьютеры выпускаются значительным числом фирм. Корпорация IBM создала суперкомпьютер в одном кристалле интегральной схемы (ИС).

Базовый (большой) компьютер — mainframe — основной тип компьютера, используемый в больших информационных сетях, работает с большой скоростью и по производительности уступает суперкомпьютеру, но охватывает более широкий круг решаемых задач. С другой стороны, он превосходит мини-компьютер по скорости работы и сложности выполняемых прикладных процессов. Базовый компьютер обладает относительно большой оперативной памятью и предоставляет свои ресурсы через коммуникационную сеть большому числу пользователей. Вследствие сказанного, базовые компьютеры принимают на себя основные потоки обработки данных. Нередко под базовым компьютером понимают лишь центральную часть крупного компьютера, включающую процессоры и оперативное запоминающее устройство.

В связи с развитием архитектуры «клиент—сервер» базовые компьютеры стали нередко использоваться в качестве серверов. Интеграция средств распределенной обработки данных с большими базовыми компьютерами обеспечивает эффективную обработку данных в корпоративных сетях. Базовые компьютеры не только функционируют как крупные серверы, но обеспечивают автоматизацию процессов, протекающих в сети.

Мини-компьютер — minicomputer — компьютер с ограниченными возможностями обработки данных. По сравнению с базовым компьютером мини-компьютер работает со словами меньшей длины, имеет ограниченную оперативную память и относительно небольшое быстродействие. Поэтому мини-компьютер используется для решения более простых задач, чем базовый. Но, по сравнению с последним, мини-компьютер имеет небольшую стоимость, размеры и проще в эксплуатации. Термин «мини-компьютер» появился тогда, когда не было персональных компьютеров. Теперь же существуют такие персональные компьютеры, которые превосходят даже базовые компьютеры восьмидесятых годов. Поэтому рассматриваемый термин применяется все реже, уступая понятиям *рабочая станция* и *персональный компьютер.*

Рабочая станция — workstation — абонентская система, специализированная на выполнение определенных задач пользователя.

Первая рабочая станция, названная Сетевым изделием Станфордского университета (SUN), была создана корпорацией SUN Microsystems под девизом «сеть есть компьютер». Это связано с тем, что рабочая станция в своей основе предназначена для работы в информационной сети. Рабочая станция, нередко именуемая рабочим местом, создается на базе малого, но достаточно мощного настольного либо напольного компьютера. Для этого разрабатывается архитектура рабочей станции, подбираются необходимые устройства (процессоры, запоминающие устройства, графопостроители, принтеры и т. д.). Создается нужное программное обеспечение, станция включается в сеть. Она предназначается для любого специалиста (программиста, системотехника, менеджера, исследователя и др.). Рабочая станция занимает среднее место среди компьютеров и характеризуется *многозадачностью* — режимом, при котором пользователь может запускать несколько задач. Это позволяет выполнять группу прикладных процессов.

Важное значение в архитектуре рабочей станции имеет визуализация информации. Она заключается в создании условий для формирования и обработки изображений. Это позволяет пользователям не только удобно отображать на экране физические объекты, но также строить модели, манипулировать ими и наблюдать ход экспериментов в реальном масштабе времени.

Все большее распространение получают рабочие станции с комбинированным сервисом. Они имеют широкий набор устройств связи с внешней средой:

- измерительные приборы;
- устройства видеоввода и микрофоны;
- оптические диски;
- клавиатуры и сенсорные устройства.

В рабочих станциях часто используются графические акселераторы (платы, содержащие процессоры, специализирующиеся по обработке изображений). Акселераторы выполняют геометрические преобразования двухмерных изображений в трехмерные изображения, учитывая ряд сложных требований, таких, например, как движущиеся источники света на экранах, отображение особенностей структуры поверхностей объектов. Акселераторы повышают стоимость станций, но резко увеличивают скорость выполнения ими сложных прикладных процессов. Широкую известность получили рабочие станции корпораций SUN Microsystems и Silicon Graphics.

Микрокомпьютер (microcomputer) — устройство, созданное на основе одного либо нескольких микропроцессоров. Существует два подхода к определению микрокомпьютера. Первый из них заключается в том, что под микрокомпьютером понимается одна либо несколько сверхбольших интегральных схем. Для этого схемы должны содержать все логические элементы, необходимые для получения полноценного компьютера небольшой производительности. Во втором подходе микрокомпьютером называется любой компьютер, в котором основными компонентами являются микропроцессоры. В дальнейшем эти ЭВМ стали именовать персональными компьютерами (ПК). В этой связи под микрокомпьютером чаще всего понимают устройство, созданное на одной либо группе интегральных схем. Для этой цели нередко бескорпусные интегральные схемы группируются в одном корпусе. Что же касается высокопроизводительного персонального компьютера, то в нем может использоваться группа микрокомпьютеров. Микрокомпьютеры также широко используются в технологии производства и в разнообразной аппаратуре автоматического управления.

Персональный компьютер (ПК) — personal computer (PC) — недорогой компьютер, созданный на базе микропроцессора. ПК, или персональные электронные вычислительные машины (ПЭВМ), в ряду компьютеров характеризуются небольшими размерами и массовым производством. Это позволяет делать их широкодоступным товаром, обеспечивающим обработку различной информации. ПК предназначены для обработки текстов, звука и изображений.

Персональный компьютер для удовлетворения требованиям общедоступности и универсальности применения должен обладать такими качествами, как:

- малая стоимость, находящаяся в пределах доступности для индивидуального покупателя;
- автономность эксплуатации без специальных требований к условиям окружающей среды;
- гибкость архитектуры, обеспечивающая адаптируемость к разнообразным применениям в сфере управления, науки, образования и в быту;
- дружественность операционной системы и прочего программного обеспечения, обусловливающая возможность работы пользователя без специальной профессиональной подготовки;

- высокая надежность работы (более 5000 ч наработки на отказ).

В последние годы использование высокоскоростных 32- и 64-разрядных микропроцессоров и версий операционной системы UNIX привело к интеграции ПК с рабочими станциями. С другой стороны, создаются устройства, в которых объединяются функции персонального компьютера с телевизором и телефонным аппаратом. Такое устройство, например, предложено корпорацией Microsoft. Это — простой интерактивный персональный компьютер (Simple Interactive Personal Computer — SIPC). Он несложен в эксплуатации, легко соединяется с телевизионной или телефонной сетью.

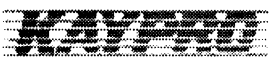
Среди всех перечисленных здесь типов ЭВМ задержимся на ПК или персональных электронных вычислительных машинах (ПЭВМ), которые в ряду компьютеров характеризуются небольшими размерами и массовым производством. Это позволяет делать их широкодоступным товаром, обеспечивающим обработку различной информации — текстов, звука, изображений и пр. В связи с их доминированием в сегодняшней ситуации остановимся на них подробнее.

Первые ПК

История ПК началась в 80-е гг. XX в., когда практически одновременно компании Motorola, Zilog и Intel выпустили на рынок достаточно мощные микропроцессоры — Intel 8086, Z80 и M68000.

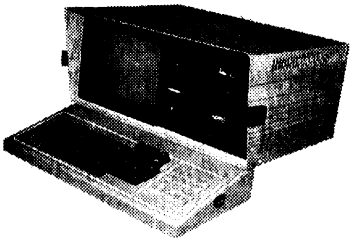
На этих микропроцессорах были построены первые микрокомпьютеры (ПК):

- Kaypro II (Zilog);
- Macintosh 128K (Motorola);
- IBM PC — XT/AT (Intel — INTegrated ELectronics).

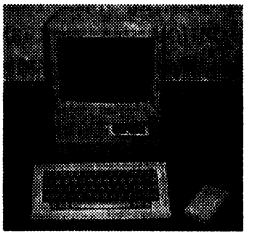


Kaypro II был представлен публике в августе 1982 г. (рис. 2.5, *а*). Несмотря на название, это была первая модель компании Э. Кея (Andrew F. Kay) Non-Linear Systems, Inc. (позже переименованной в Kaypro Corporation).

При весе более десяти килограмм Kaypro II позиционировался как переносная система. Возможность работы в полевых условиях была подтверждена во время ралли Париж—Дакар в



а



б

в

Рис. 2.5. Первые ПК:
а — Kaypro II; *б* — Macintosh 128K; *в* — IBM PC XT

1984 г., на котором организаторами использовалось десять компьютеров Kaypro II.

Технические характеристики этой модели близки к системе Osborne 1, выпускаемой фирмой А. Осборна, и HP-85 (Hewlett-Packard). Следует, однако, отметить, что при близости возможностей Kaypro II был почти вдвое легче Osborne 1, что для переносной системы имеет первоочередную важность, и на два года «моложе» HP-85.

Kaypro II оснащался процессором Zilog Z80 с тактовой частотой 2,5 МГц. Объем ОЗУ составлял 64 Кб, ПЗУ — 2 Кб. Встроенный дисплей размером 9" был способен работать в текстовом режиме 80 × 24, поддержка графического режима не предусматривалась. Клавиатура компьютера была довольно удобной, с курсорными и цифровыми клавишами. Устройством хранения данных служили два дисковода для односторонних 5,25" дисков двойной плотности емкостью 190 Кб. Работал Kaypro II под управлением ОС CP/M (Control Program for Microprocessor) и на момент выхода стоил 1795 долл.

Модель 128K — первый продукт в семействе компьютеров Macintosh (рис. 2.5, *б*). Корпорация Apple представила ее в январе 1984 г., развернув интенсивную мар-

кетинговую кампанию. «Если уж компьютеры так умны, — говорилось в одной из реклам, — то не лучше ли научить их общаться с человеком, чем учить людей общаться с компьютером?»

Macintosh 128K базировался на 32-разрядном микропроцессоре Motorola 68000 с тактовой частотой 8 МГц, имел 128 Кб ОЗУ, 64 Кб ПЗУ, односторонний флоппи-дисковод (400 Кбайт; 3,5"), встроенный 9" черно-белый экран с графическим разрешением 512 × 342 точки, оснащался, помимо клавиатуры, манипулятором «мышь» и весил 20 фунтов (8 кг).

Многое в новом компьютере было заимствовано из разработок исследовательского центра PARC компании Xerox, 15 специалистов которой в начале 1980-х гг. перешли на работу в Apple. Именно PARC были предложены система «окон» на экране, интерфейс, основанный на символических изображениях действий, устройство «мышь».

Хотя первый Macintosh стоил в 5 раз дороже, чем требовали спецификации проекта — 2495 долл. вместо 500 долл., продажи шли успешно, и за первые 9 мес. было реализовано 275 тыс. компьютеров.

«Мак», как вскоре окрестили поклонники новую машину, принес Apple грандиозный успех. Владельцы компьютеров других типов завидовали графическим средствам «Мака» и простоте обращения с ним. Программы, позволявшие использовать «окна» и «мышь» в компьютерах других моделей, раскупались нарасхват.

Однако очень скоро пользователи Macintosh 128K столкнулись с проблемой слишком малого объема оперативной памяти при том, что возможностей для расширения ОЗУ конструкцией модели предусмотрено не было. Многочисленные нарекания побудили корпорацию Apple усовершенствовать свой продукт, и уже осенью того же 1984 г. был представлен «Fat Mac» — модель Macintosh 512K — (Macintosh 128K был снят с производства в октябре 1985 г.).

IBM PC. На рис. 2.5, *в* изображена ПЭВМ IBM PC XT (eXtendet Tecnology — расширенная технология, по сравнению с IBM PC), не самая первая машина в серии (8 марта 1983 г.). Процессор Intel 8088 — 4,77 МГц, ОЗУ 256 Кбайт, ПЗУ 64 Кбайт, небольшой винчестер, два дисковода по 360 Кбайт и адаптер MDA (CGA); операционная система — CP/M-86, MS-DOS, Minix.

2.2. Узлы ЭВМ

Узлом ЭВМ называется совокупность функционально связанных элементов, предназначенных для выполнения определенных операций над двоичными словами. Узлы ЭВМ являются основными элементами реализации аппаратных функций ЭВМ (преобразование, передача, хранение и управление информацией). Они обеспечивают преобразование кодов, подсчет импульсов, сравнение кодов, сдвиг двоичных слов.

По выполняемым функциям узлы делятся на регистры, сумматоры (накапливающего типа), счетчики, дешифраторы, шифраторы, мультиплексоры, демультиплексоры, схемы сравнения кодов, программируемые логические матрицы (ПЛМ), аналого-цифровые и цифроаналоговые преобразователи (АЦП и ЦАП) и пр.

Регистр — узел ЭВМ, предназначенный для хранения двоичных слов и выполнения над ними некоторых логических операций. Регистр представляет собой совокупность триггеров, число которых соответствует числу разрядов в слове, и вспомогательных схем, обеспечивающих выполнение некоторых операций, таких как:

- установка регистра в 0 — сброс;
- прием слова;
- выдача слова;
- сдвиг слова влево или вправо на требуемое количество разрядов;
- преобразование последовательного кода в параллельный и наоборот;
- разрядные логические операции.

По способу приема и выдачи информации регистры делятся на:

- параллельные;
- сдвиговые;
- комбинированные.

Параллельные регистры. Используются для выполнения операций приема, хранения, выдачи и поразрядных логических операций над словами и представляют собой совокупность *RS*-, *D*- и *T*-триггеров, имеющих связанные входные и выходные цепи (рис. 2.6).

Сдвиговые регистры. Используются для выполнения следующих операций:

1) умножение на 2^k, если сдвиг осуществляется на k разрядов в сторону старших разрядов;

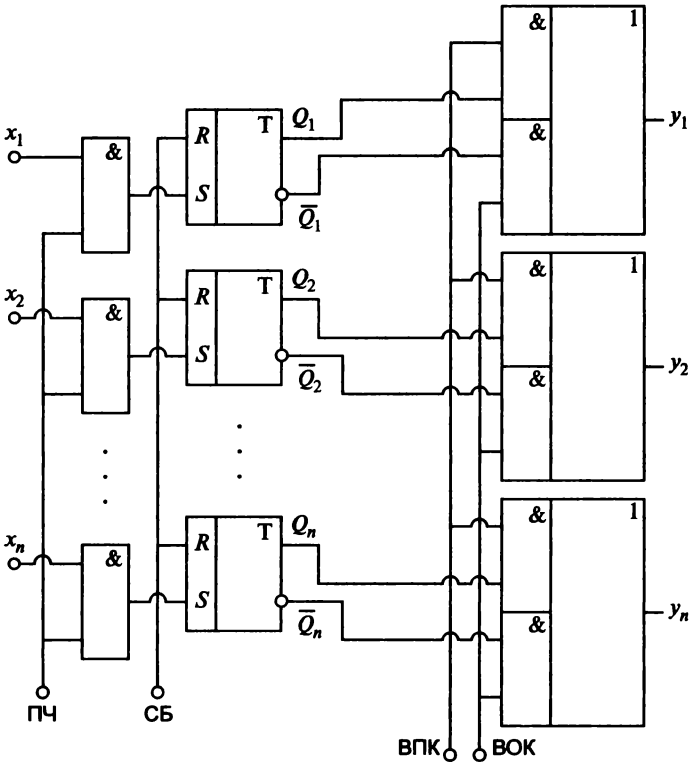


Рис. 2.6. Параллельный регистр:
ПЧ — прием числа; СБ — сброс регистра в 0; ВПК — выдача прямого кода; ВОК — выдача обратного кода

2) умножение на 2^{-k}, если сдвиг осуществляется на k разрядов в сторону младших разрядов;

3) преобразование кода из параллельного в последовательный и обратно.

Самая простая схема сдвигового регистра строится на D-триггерах и имеет вид, приведенный на рис. 2.7.

Прямой выход Q_i предыдущего разряда поступает на информационный вход D последующего разряда. Благодаря этому каждый синхросигнал устанавливает последующий триггер в состояние, в котором до этого находился предыдущий разряд, осуществляя тем самым сдвиг информации на разряд влево (или вправо).

Реверсивные регистры сдвига. Обеспечивают возможность сдвига информации как влево, так и вправо (рис. 2.8).

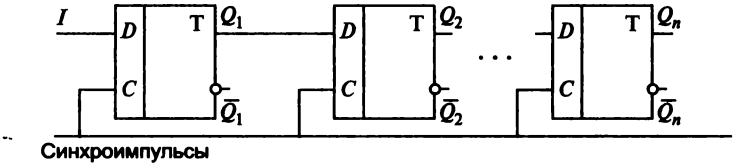


Рис. 2.7. Простейший сдвиговый регистр

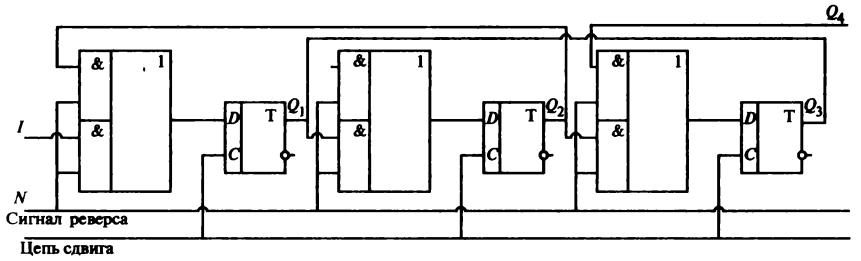


Рис. 2.8. Реверсивный регистр сдвига

Если сигнал на шине N равен «1», то потенциал на входе D i-го триггера определяется выходом Q_{i-1} триггера, стоящего слева от него $(i-1)$. Если сигнал равен «0», то потенциал на входе D i-го триггера определяется выходом Q_{i+1} триггера, стоящего справа от него $(i+1)$.

Счетчик — накопительный узел ЭВМ, предназначенный для подсчета числа импульсов, поступивших на его вход. По структуре различают счетчики:

- с последовательным переносом;
- сквозным переносом;
- параллельным переносом;
- групповым переносом.

В зависимости от алгоритма реализации выделяют счетчики:

- суммирующие;
- вычитающие;
- реверсивные;
- с предустановкой.

В зависимости от модуля счета счетчики бывают:

- двоичные;
- десятичные.

К характеристикам счетчиков относят:

- коэффициент пересчета (число состояний счетчика) M (количество импульсов, поступивших на вход счетчика, кото-

рые переводят его в исходное состояние). Между числом разрядов счетчика n и коэффициентом пересчета M существует соотношение $n \geq \log(M+1)$;

- время реакции (регистрации) — интервал времени $t_{рег}$ между поступлением входного сигнала и окончанием самого длительного переходного процесса в счетчике;
- разрешающую способность — минимальный допустимый период (или максимальная частота) следования входных сигналов, при котором счетчик работает без сбоев.

Счетчик с последовательным переносом. Строится на основе T-триггеров (рис. 2.9).

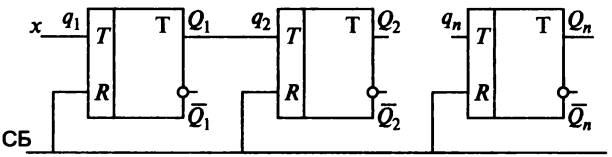


Рис. 2.9. Счетчик с последовательным переносом

Здесь $t_{рег} = n\tau_т$, где n — разрядность счетчика, $\tau_т$ — время переключения триггера. Чем больше разрядность слова, тем больше время регистрации, поэтому счетчик имеет наиболее низкое быстродействие.

Счетчик со сквозным переносом. Увеличить быстродействие счетчика можно за счет организации цепей сквозного переноса между разрядами счетчика (рис. 2.10). В этом случае $t_{рег} = \tau_т + (n-1)\tau_и$, где $\tau_и$ — время срабатывания схемы «И».

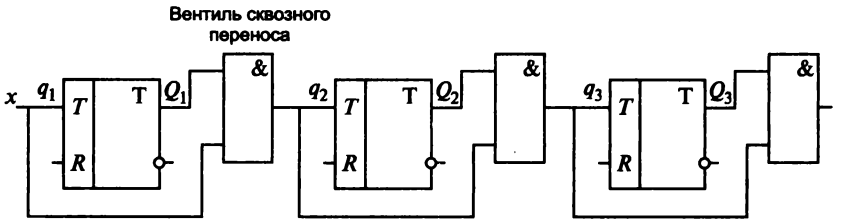


Рис. 2.10. Счетчик со сквозным переносом

Счетчик с параллельным переносом имеет максимальное быстродействие. Здесь $t_{рег} = \tau_т + \tau_и$.

Счетчик с групповым переносом. Из-за конечности коэффициентов объединения элементов «И» число разрядов в

счетчике с параллельным переносом не может быть очень большим (обычно не более 8 разрядов). Если же требуется больше разрядов, то целесообразно использовать групповой перенос, при котором переносы внутри группы формируются параллельно, а между группами или последовательно, или параллельно.

Вычитающий счетчик. Строится по принципу суммирующего счетчика, только подача сигналов осуществляется с инверсных выходов предыдущих разрядов.

Реверсивный счетчик. В зависимости от наличия сигнала сложения или вычитания ведет счет в прямом или обратном направлениях.

Счетчик с предустановкой. Используется, например, в качестве счетчика команд (СчАК) — см. рис. 2.23. Позволяет вначале переслать некоторый код в счетчик, а затем продолжить прерванный счет, начиная с этого кода, записанного в счетчике.

Пересчетная схема. Отличается от счетчиков способом снятия результата. В счетчиках показания снимаются в параллельном коде. В пересчетных схемах единичный сигнал формируется на выходе после подачи определенного количества импульсов на входе.

Сумматор — узел ЭВМ, выполняющий суммирование двоичных кодов чисел. Он является узлом преобразования информации. Различают комбинационные и накапливающие сумматоры.

В *комбинационных* сумматорах оба слагаемых подаются одновременно. При этом на выходах сумматоров фиксируется сумма, которая существует до тех пор, пока на входах действуют слагаемые.

В *накапливающих* сумматорах в начале подается 1-е слагаемое, которое запоминается сумматором. После подачи 2-го слагаемого в сумматоре образуется сумма, которая тоже запоминается.

В зависимости от способов обработки разрядов слагаемых различают сумматоры:

- последовательного действия (разряды обрабатываются последовательно один за другим, начиная с младшего);
- параллельного действия (все слагаемые обрабатываются одновременно, как правило, за один рабочий такт);
- последовательно-параллельного действия (одновременно обрабатывается группа разрядов, а между группами обработка идет последовательно).

В зависимости от способа реализации переносов различают сумматоры:

- с последовательным переносом;
- сквозным переносом;
- параллельным переносом;
- групповым переносом.

Многоразрядные сумматоры строятся как совокупность одноразрядных (см. рис. 1.14).

Комбинационный сумматор параллельного действия с последовательным переносом. Такие сумматоры строятся на основе композиции одноразрядных полных сумматоров. Обработка слагаемых происходит одновременно во всех разрядах. Сигнал переноса, который вырабатывается в младших разрядах, последовательно распространяется в цепях переноса к старшим разрядам. Например, схема вычисления суммы $S = (s_3\ s_2\ s_1\ s_0)$ двух двоичных трехразрядных чисел $A = (a_2\ a_1\ a_0)$ и $B = (b_2\ b_1\ b_0)$ может иметь вид, приведенный на рис. 2.11.

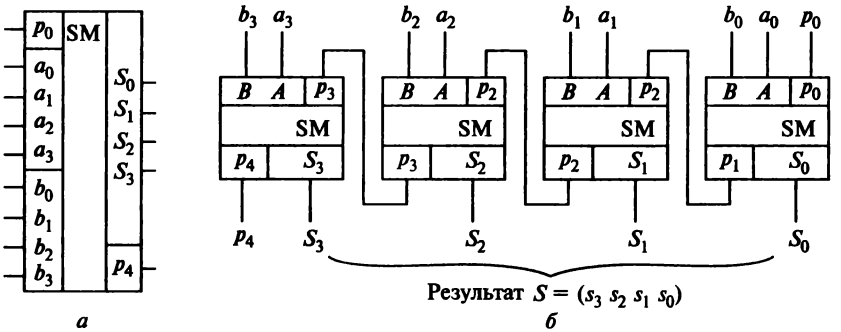


Рис. 2.11. Изображение полного двоичного многоразрядного сумматора на схемах (*а*); фрагмент (три разряда) принципиальной схемы многоразрядного сумматора (*б*)

Комбинационный сумматор параллельного действия с параллельным переносом. Применяется схема параллельного переноса (СПП), формирующая перенос каждого разряда сумматора независимо и параллельно (рис. 2.12). Сложность СПП уменьшается, если использовать зависимости между последующим (P_{i+1}) и предыдущим (P_i) переносами, которые выражаются так:

$$P_{i+1} = a_i \vee b_i \wedge P_i.$$

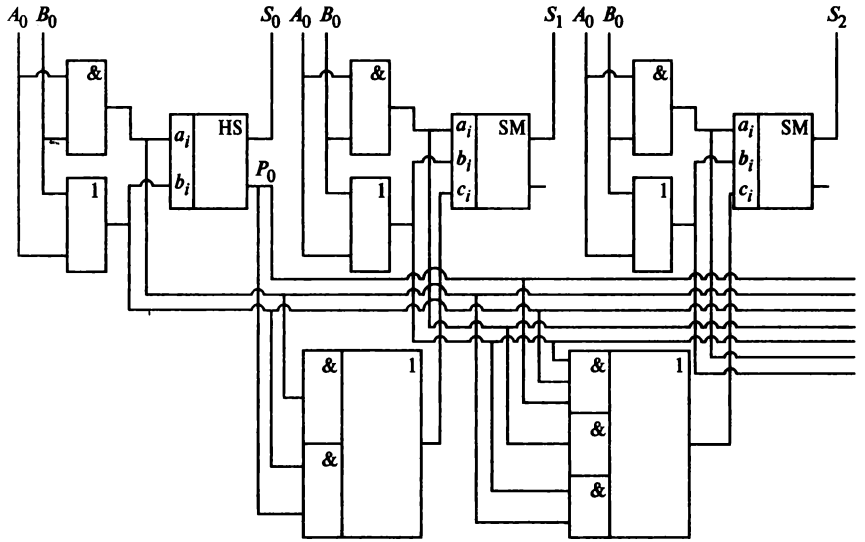


Рис. 2.12. Сумматор параллельного действия с параллельным переносом

Отсюда

$$P_1 = a_0 \vee b_0 \wedge P_0;\ \ P_2 = a_1 \vee b_1 \wedge P_1;\ \ P_3 = a_2 \vee b_2 \wedge P_2 \ \ldots$$

или

$$P_2 = a_1 \vee b_1 \wedge b_0 \wedge P_0;$$

$$P_3 = a_2 \vee (b_2 \wedge a_1) \vee (b_2 \wedge b_1 \wedge a_0) \vee (b_2 \wedge b_1 \wedge b_0 \wedge P_0);$$

...

$$P_{i+1} = a_i \vee (b_i \wedge a_{i-1}) \vee (b_i \wedge b_{i-1} \wedge a_{i-2}) \vee \ldots \vee (b_i \wedge b_{i-1} \wedge \ldots \wedge b_0 \wedge P_0).$$

Комбинационный сумматор параллельного действия со сквозным последовательным переносом (рис. 2.13). Повысить быстродействие сумматора можно за счет упрощения цепей распространения переносов, если один вход с одноразрядного сумматора выделяется для этих целей:

$$P_{i+1} = A_i \wedge B_i \vee (A_i \vee B_i) \wedge P_i,$$

где $x_i = A_i \wedge B_i$ — сигнал переноса, непосредственно обрабатывается в i-м разряде; $Y_i = A_i \vee B_i$ — признак распространения сигнала переноса через i-й разряд.

Комбинационный n-разрядный сумматор с групповым переносом. В таком сумматоре вводят обходные цепи распространения сигнала переноса, для этого сумматор разбива-

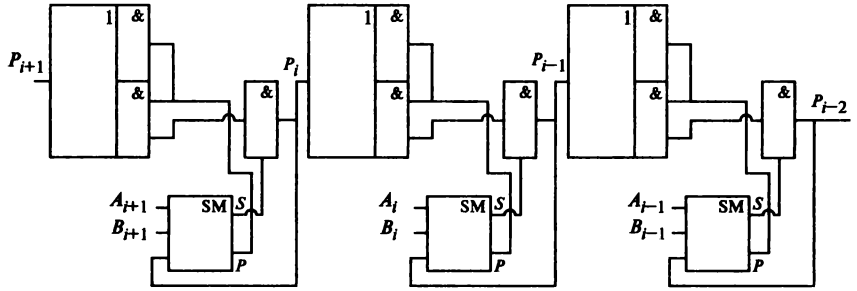


Рис. 2.13. Комбинационный сумматор параллельного действия со сквозным последовательным переносом

ется на группы разрядов равной длины, как правило, по 4 или 8 разрядов. Внутри каждой группы создается параллельный перенос, а между группами может быть параллельный или последовательный перенос. Сигнал переноса, поступающий на вход младшего разряда группы при наличии условий распространения переносов во всех разрядах группы, подается на вход следующей группы в обход данной группы. Внутри группы находится блок, который вырабатывает сигнал переноса, и схема, которая вырабатывает распространение переноса *P*. Такой блок называется схемой ускоренного переноса (СУП). В СУП выполняется параллельный перенос, а между группами может осуществляться параллельный или последовательный перенос.

Дешифратор — комбинационный узел, который предназначен для преобразования двоичного кода (x) на входе в управляющий сигнал (z) на одном из выходов (рис. 2.14, *а*). Если входов n, то выходных шин должно быть $N = 2^n$ (табл. 2.7, $n = 3$, $N = 8$). Если на вход дешифратора подается двоичный код, то на одном из выходов вырабатывается сигнал «1», а на остальных выходах сохраняется «0» (дешифратор преобразует код на входах в унитарный код на выходах).

Для трехвходового дешифратора можно записать логическое выражение выходов (см. также табл. 2.7):

$$z_0 = \bar{x}_3 \wedge \bar{x}_2 \wedge \bar{x}_1; \quad z_1 = \bar{x}_3 \wedge \bar{x}_2 \wedge x_1;$$
$$z_2 = \bar{x}_3 \wedge x_2 \wedge \bar{x}_1; \quad z_3 = x_3 \wedge \bar{x}_2 \wedge \bar{x}_1;$$
$$z_4 = \bar{x}_3 \wedge x_2 \wedge x_1; \quad z_5 = x_3 \wedge \bar{x}_2 \wedge x_1;$$
$$z_6 = x_3 \wedge x_2 \wedge \bar{x}_1; \quad z_7 = x_3 \wedge x_2 \wedge x_1.$$

По структурному построению дешифраторы делятся на линейные и многокаскадные.

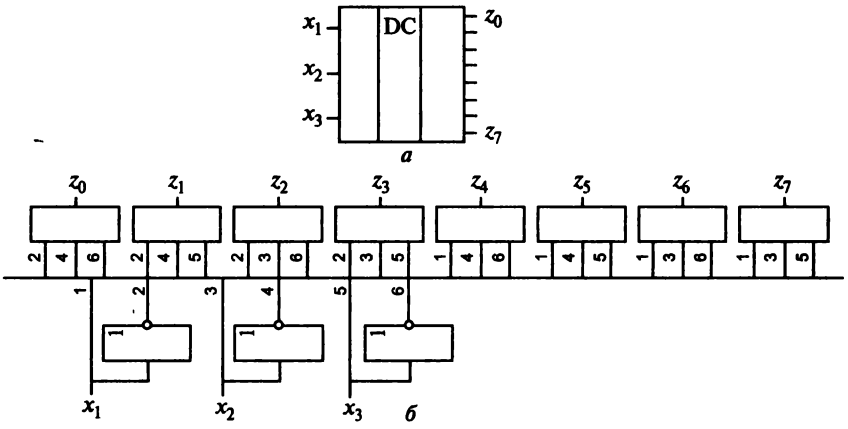


Рис. 2.14. Условное обозначение дешифратора в схемах (*а*); схема линейного дешифратора (*б*)

Таблица 2.7. **Пример таблицы состояний дешифратора**

x_1	x_2	x_3	z_0	z_1	z_2	z_3	z_4	z_5	z_6	z_7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

У *линейных дешифраторов* все переменные x_1, x_2, x_3 подаются одновременно (рис. 2.14, *б*). Схема дешифратора представляет собой набор из восьми трехвходовых элементов «И», на входы которых подаются все возможные комбинации прямых и инверсных значений входного кода (рис. 2.14, *б*).

Они обладают более высоким быстродействием, но более трех переменных одновременно подать нельзя, поэтому чаще применяются *многокаскадные* дешифраторы. Здесь количество элементов в каждом следующем разряде больше, чем в предыдущем. На вход первого каскада подается один слог, на вход следующего каскада — второй слог и результаты конъюнкций, произведенных в первом каскаде.

Простейший линейный дешифратор можно построить на *диодной матрице* (рис. 2.15, *а*). В этой схеме используется отрицательная логика. При подаче «1» на анод (коллектор) диода он закрывается. Если закрыты все три диода, подсоединенных к одной горизонтальной линии, то на этой линии появляется потенциал $-E$, соответствующий уровню «1».

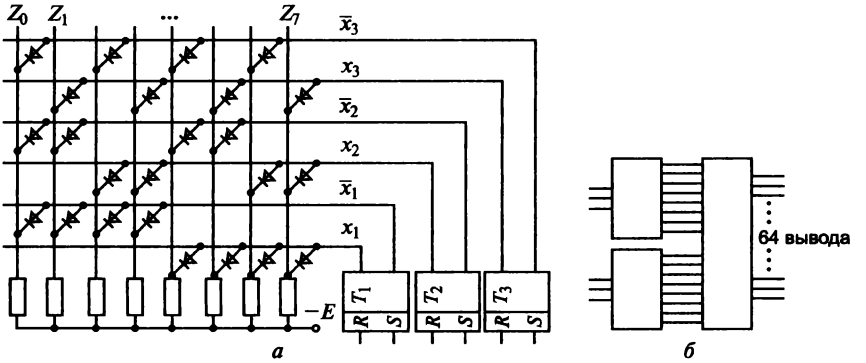


Рис. 2.15. Диодная матрица (*а*), многокаскадный дешифратор (*б*)

Многокаскадный дешифратор можно организовать так, как это изображено на рис. 2.15, *б*. Два линейных дешифратора обрабатывают по два слова. В последнем каскаде образуются конъюнкции выходного сигнала первого каскада. Многокаскадные дешифраторы обладают меньшим быстродействием.

Шифратор — это узел ЭВМ с несколькими входами и выходами, преобразующий сигнал на одном из входов в код этого входа (рис. 2.16, *а*). Шифратор выполняет функцию, обратную относительно дешифратора. Примером шифратора является клавиатура, преобразующая сигналы клавиш в код этой клавиши.

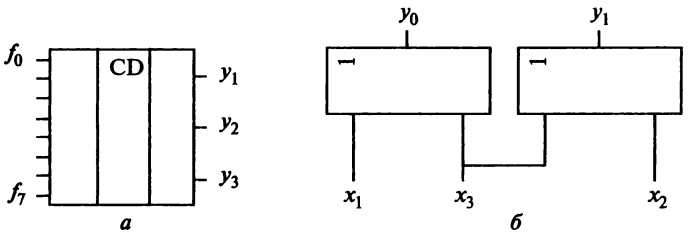


Рис. 2.16. Шифратор:
а — условное обозначение на схемах; *б* — схема шифратора

Для построения шифраторов могут использоваться схемы «ИЛИ», на которые подаются прямые значения входного сигнала (рис. 2.16, *б*).

Таблица истинности работы шифратора, который имеет 3 входа и 2 выхода, приводится в табл. 2.8.

Таблица 2.8. **Таблица истинности шифратора**

x_3	x_2	x_1	y_1	y_0
0	0	1	0	1
0	1	0	1	0
1	0	0	1	1

Мультиплексор — узел ЭВМ, осуществляющий передачу сигналов с одной из входных линий в выходную (рис. 2.17). Выбор выходной линии производится управляющим кодом, поступающим

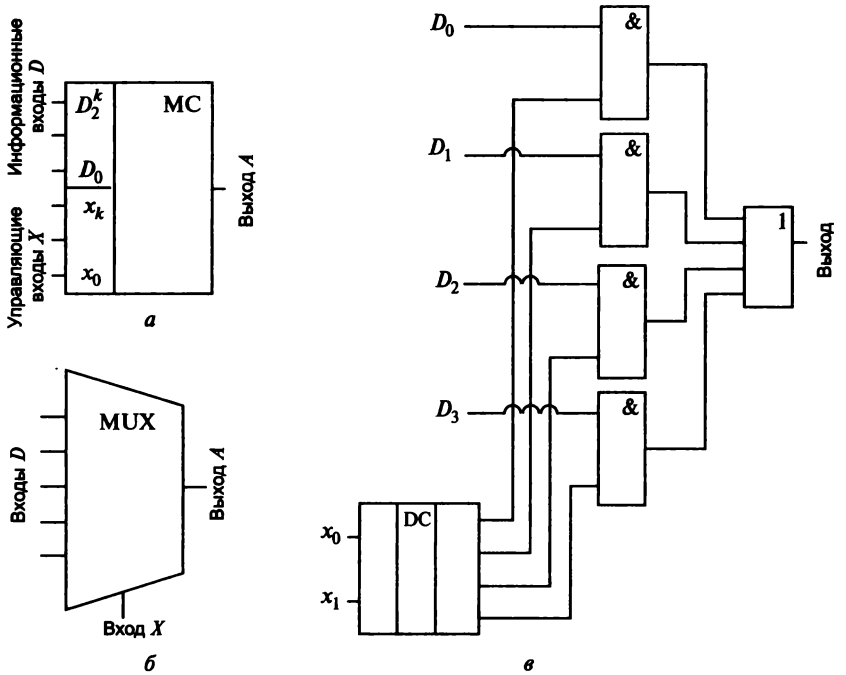


Рис. 2.17. Мультиплексор:
а, б — обозначения на схемах; *в* — схемная реализация

на входы мультиплексора, т. е. в мультиплексорах различают управляющие и информационные входы. Если управляющих кодов k, то информационных кодов 2^k. Мультиплексор обеспечивает временное объединение каналов и является основным узлом, реализующим аппаратную функцию передачи данных.

Демультиплексор выполняет функцию, обратную функции мультиплексора, и используется для временного разделения данных, поступающих от одного источника, по каналам. Это узел ЭВМ, осуществляющий передачу информации, поступающей на общий вход, на одну из выходных линий (рис. 2.18). Выбор линии выхода производится кодом, поступающим на управляющие входы демультиплексора, т. е. он имеет одну информационную линию и несколько управляющих.

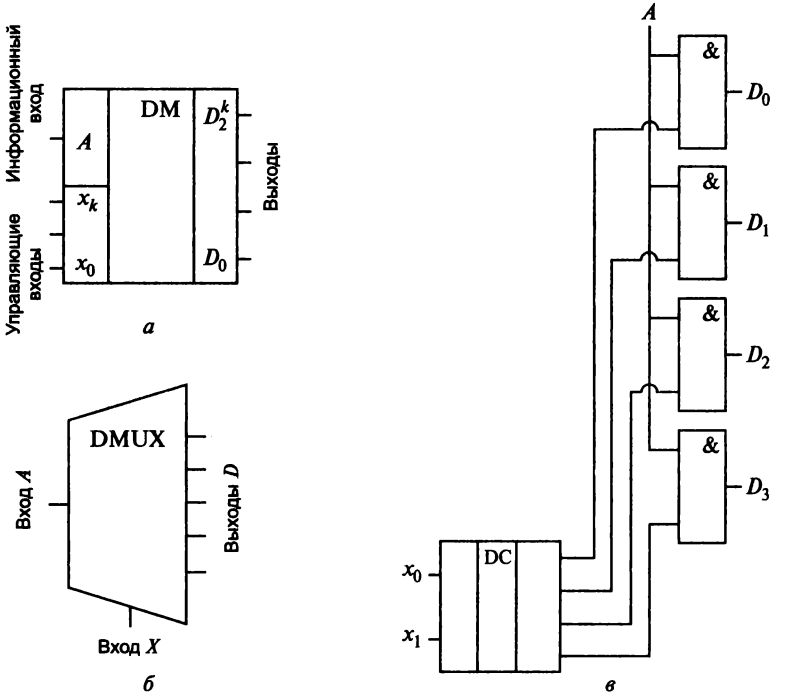


Рис. 2.18. Демультиплексор:
а, б — обозначения на схемах; *в* — схемная реализация

Схема сравнения чисел (цифровой компаратор) — узел ЭВМ, предназначенный для выдачи выходных сигналов «равно» (E),

«больше» (G), «меньше» (L) в зависимости от соотношения сравниваемых кодов A и B (рис. 2.19). Для того чтобы синтезировать n-разрядную схему сравнения, вначале надо составить таблицу истинности для i-го разряда (табл. 2.9).

Таблица 2.9. **Таблица истинности сравнения для i-го разряда компаратора**

a_i	b_i	g_i	l_i	e_i
0	0	0	0	1
0	1	0	1	0
1	0	1	0	0
1	1	0	0	1

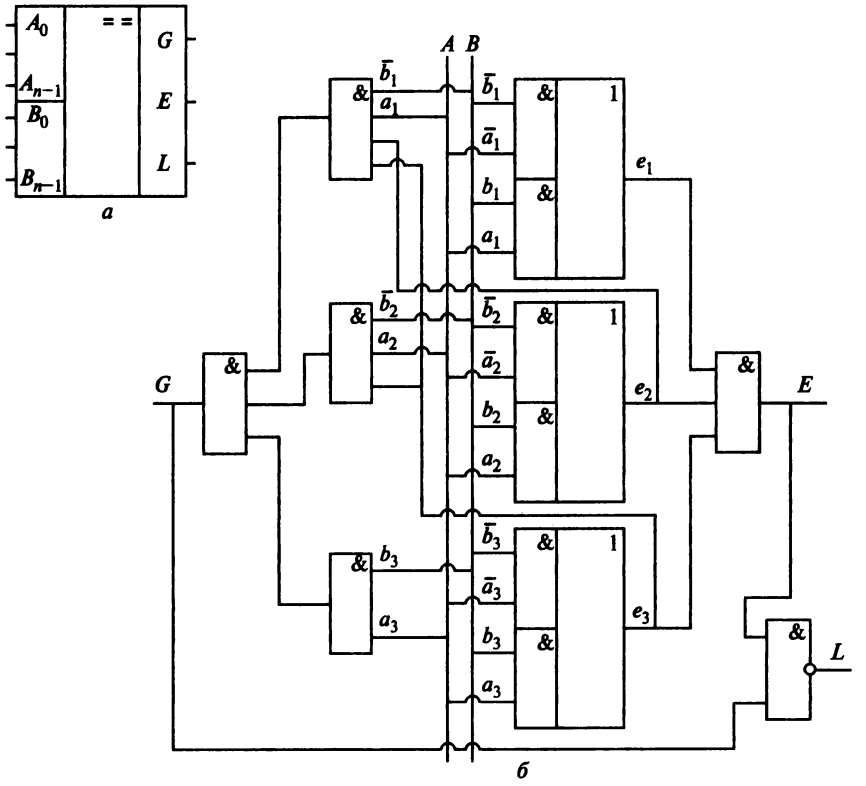


Рис. 2.19. Компаратор:
а — обозначение на схемах; *б* — схемная реализация

Значения выходов i-го разряда, очевидно, равны:

$$g_i = a_i \wedge \overline{b}_i; \quad l_i = \overline{a}_i \wedge b_i; \quad e_i = \overline{a}_i \wedge \overline{b}_i \wedge a_i \wedge b_i,$$

т. е. $g_i = 1$, если $a_i = 1$ и $b_i = 0$; $l_i = 1$, если $a_i = 0$ и $b_i = 1$; $e_i = 1$, если $a_i = b_i$.

Значение E, равное «1», означает равенство двух n-разрядных кодов, и оно вычисляется как конъюнкция одноразрядных функций равнозначностей e_i:

$$E = e_1 \wedge e_2 \wedge \ldots \wedge e_n.$$

Функция G определяется в соответствии с правилами формирования переноса в параллельном сумматоре, только продвижение переносов осуществляется от старших к младшим разрядам:

$$G = g_n \vee (g_{n-1} \wedge e_n) \vee (g_{n-2} \wedge e_{n-1} \wedge e_n) \vee \ldots$$

$$\ldots \vee g_1 \wedge (e_2 \wedge e_3 \wedge \ldots \wedge e_{n-1} \wedge e_n).$$

Отношение L определяется как совместное наступление событий $\overline{E}$ и $\overline{G}$:

$$L = \overline{E} \wedge \overline{G}.$$

Программируемые логические матрицы (ПЛМ). ПЛМ — узел ЭВМ, предназначенный для реализации системы булевых функций. ПЛМ — это комбинационная схема с регулярной структурой, которая реализуется обычно в виде интегральной схемы.

В ней входы x_1, x_2, x_3, ..., x_n и выходы y_1, y_2, y_3, ..., y_n связаны двумя матрицами и логическими элементами «НЕ», «ИЛИ», «И».

Структурно ПЛМ состоит из двух матриц M_1 и M_2 (рис. 2.20). M_1 (матрица «И») формирует k промежуточных конъюнкций от n входных переменных (или их инверсий), а M_2 (матрица «ИЛИ») — m дизъюнкций от k конъюнкций.

При построении матриц M_1 и M_2 на пересечении горизонтальных и вертикальных линий включаются транзисторы. В матрице M_1 входные сигналы и их инверсии коммутируются через транзисторы с горизонтальными линиями z_k, образуя логическое произведение входов:

$$z_k = x_1 \wedge x_2 \wedge x_3 \wedge \ldots \wedge x_n.$$

Количество функций z_k будет зависеть от числа логических объектов, формирующих вертикальные линии.

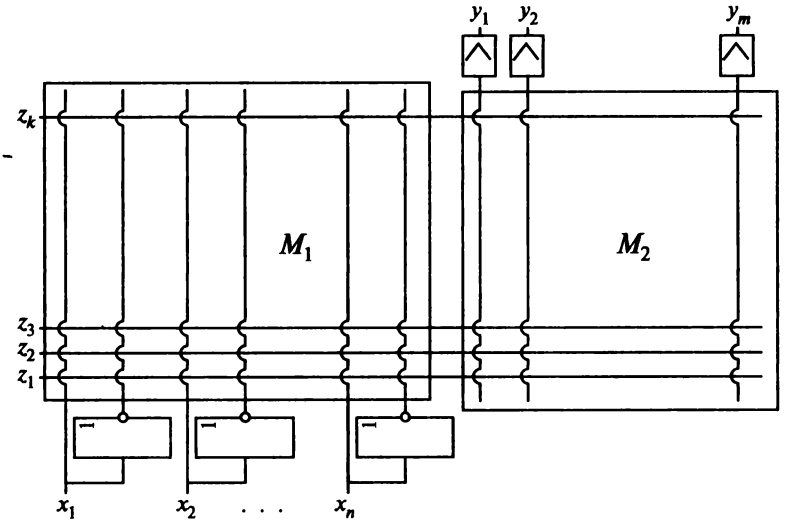


Рис. 2.20. Структура программируемой логической матрицы (ПЛМ)

Далее, выходные сигналы y_m в матрице M_2, соответствующие функциям,

$$y_m = z_1 \vee z_2 \vee z_3 \vee \ldots \vee z_k.$$

Информация заносится в ПЛМ путем установки связей между горизонтальными и вертикальными линиями. Этот процесс называется программированием матриц.

Цифроаналоговый преобразователь (ЦАП, DAC) предназначен для преобразования числа, представленного n-разрядным двоичным кодом в выходное напряжение — пропорциональную аналоговую величину. Схемы ЦАП строятся с использованием операционных усилителей (ОУ). В основу преобразования положено ступенчатое изменение коэффициента передачи ОУ пропорционально коду числа на входе путем коммутации сопротивлений на входе ОУ.

В состав ЦАП входят (рис. 2.21):

- регистр *RG*, предназначенный для хранения входного преобразуемого двоичного кода;
- ОУ с сопротивлением обратной связи R_o;
- матрица сопротивлений R_1, R_2, ..., R_n ($R_i = 2R_{i+1}$);
- источник стабильного напряжения $U_{оп}$;
- транзисторные ключи.

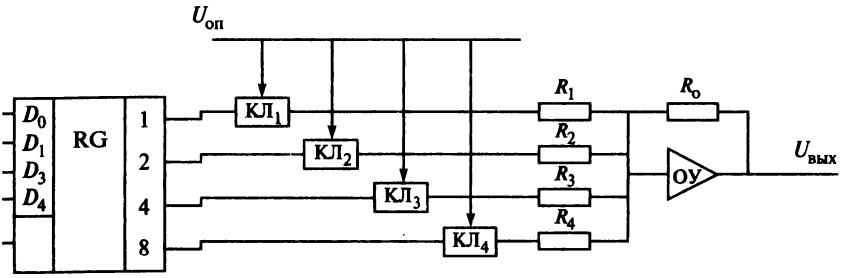


Рис. 2.21. Принципиальная схема цифроаналогового преобразователя

Выходное напряжение ЦАП равно сумме напряжений включенных разрядов с учетом их веса:

$$U_{вых} = \sum_{i=1}^{n} b_i \frac{R_o}{R_i} U_{оп},$$

где b_i (0 или 1) — значение i-го разряда регистра RG.

Точность преобразования ЦАП определяется:

- стабильностью опорного напряжения $U_{оп}$;
- точностью изготовления сопротивлений R_i;
- количеством преобразуемых двоичных разрядов (n);
- точностью операционного усилителя.

Аналого-цифровой преобразователь (АЦП, ADC) основывается на операциях дискретизации сигнала по времени и квантовании по уровню (рис. 2.22). В процессе дискретизации через определенные интервалы времени измеряются мгновенные значения непрерывного сигнала.

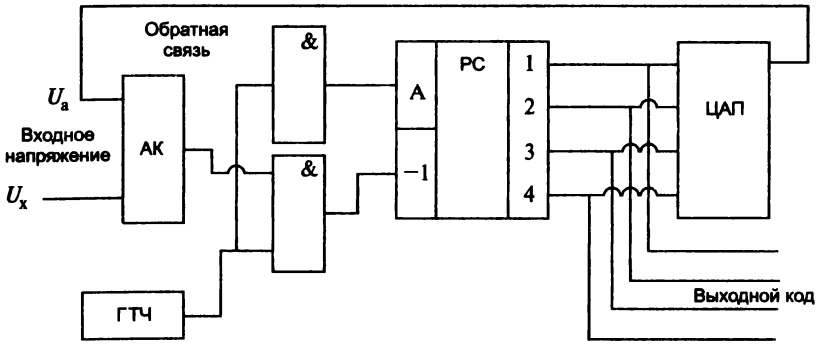


Рис. 2.22. АЦП с обратной связью

Суть операции квантования состоит в создании множества уровней, смещенных относительно друг друга на величину шага квантования.

По принципу действия АЦП делятся на два класса:

- прямого преобразования (без обратных связей);
- уравновешивающие (с обратными связями).

Рассмотрим в качестве примера АЦП поразрядного уравновешивания с использованием реверсивного счетчика, в состав которого входят (см. рис. 2.22):

- ГТЧ — генератор тактовой частоты;
- РС — реверсивный счетчик, на выходе которого формируется цифровое представление входного сигнала;
- ЦАП — цифроаналоговый преобразователь — преобразует выходной код счетчика в напряжение обратной связи U_a;
- АК — амплитудный компаратор, который сравнивает напряжение с выхода ЦАП с преобразуемым входным напряжением U_x.

Компаратор переключает реверсивный счетчик:

- при $U_x - U_a \geq 0$ счетчик ведет счет в прямом направлении;
- при $U_x - U_a < 0$ счет идет в обратном направлении.

2.3. Базовые представления об архитектуре ЭВМ

Структура компьютера — это совокупность его функциональных элементов и связей между ними. Элементами могут быть самые различные устройства — от основных логических узлов компьютера до простейших схем. Структура компьютера графически представляется в виде структурных схем, с помощью которых можно дать описание компьютера на любом уровне детализации.

Архитектурой компьютера считается его представление на некотором общем уровне, включающее описание пользовательских возможностей программирования, системы команд, системы адресации, организации памяти и т. д. Архитектура определяет принципы действия, информационные связи и взаимное соединение основных логических узлов компьютера: *процессора, оперативного запоминающего устройства (ОЗУ, ОП), внешних ЗУ и периферийных устройств*. Общность архитектуры разных компьютеров обеспечивает их совместимость с точки зрения пользователя.

Принципы фон Неймана

В основу архитектуры большинства компьютеров положены следующие общие принципы, сформулированные в 1945 г. американским ученым Джоном фон Нейманом в отчете по ЭВМ EDVAC:

- ***принцип программного управления.*** Из него следует, что программа состоит из набора команд, которые выполняются процессором автоматически друг за другом в определенной последовательности. Выборка программы из памяти осуществляется с помощью *счетчика команд* (СчАК). Этот регистр процессора последовательно увеличивает хранимый в нем адрес очередной команды. Если после выполнения команды следует перейти не к следующей, а к какой-то другой, используются команды *условного или безусловного переходов,* которые заносят в счетчик команд номер ячейки памяти, содержащей следующую команду;
- ***принцип однородности памяти*** — программы и данные хранятся в одной и той же памяти. Поэтому компьютер не различает, что хранится в данной ячейке памяти — число, текст или команда. Над командами можно выполнять такие же действия, как и над данными. Например, программа в процессе своего выполнения также может подвергаться переработке, что позволяет задавать в самой программе правила получения некоторых ее частей (так в программе организуется выполнение циклов и подпрограмм);
- ***принцип адресности.*** Структурно основная память состоит из перенумерованных ячеек; процессору в произвольный момент времени доступна любая ячейка. Отсюда следует возможность давать имена областям памяти так, чтобы к запомненным в них значениям можно было впоследствии обращаться или менять их в процессе выполнения программ с использованием присвоенных имен.

Компьютеры, построенные на этих принципах, относятся к типу фон-неймановских. Существуют и другие классы компьютеров, принципиально отличающиеся от них, — нефон-неймановские.

Например, в ассоциативных компьютерах может не выполняться принцип программного управления, поскольку каждая команда здесь содержит адрес следующей (т. е. они могут

работать без *счетчика команд*, указывающего на выполняемую команду программы).

По прошествии более 60 лет большинство компьютеров так и имеют «фон-неймановскую архитектуру», причем принципы фон Неймана реализованы в следующем виде:

- оперативная память (ОП) организована как совокупность *машинных слов (МС) фиксированной длины или разрядности* (имеется в виду количество двоичных единиц или бит, содержащихся в каждом МС). Например, ранние ПЭВМ имели разрядность 8, затем появились 16-разрядные, а затем — 32- и 64-разрядные машины. В свое время существовали также 45-разрядные (М-20, М-220), 35-разрядные (Минск-22, Минск-32) и др. машины;
- ОП образует единое адресное пространство, адреса МС возрастают от младших к старшим;
- в ОП размещаются как данные, так и программы, причем в области данных одно слово, как правило, соответствует одному числу, а в области программы — одной команде (машинной инструкции — минимальному и неделимому элементу программы);
- команды выполняются в *естественной последовательности* (по возрастанию адресов в ОП), пока не встретится *команда управления* (условного/безусловного перехода, или ветвления — `branch`), в результате которой естественная последовательность нарушится;
- ЦП может произвольно обращаться к любым адресам в ОП для выборки и/или записи в МС чисел или команд.

Функциональные блоки (агрегаты, устройства)

В то время как логические элементы и узлы во многом универсальны и могут использоваться в самых различных сочетаниях для решения разнообразных задач, блоки (агрегаты) ЭВМ представляют собой комплексы элементов (узлов), ориентированные на узкий круг задач (операций). Такие агрегаты, как АЛУ, процессор, банк памяти, внешние устройства (НГМД и пр.), обязательно включают в свой состав (кроме механического, оптического, электромагнитного и иного оборудования) логические элементы и узлы, используемые для хранения информации, ее обработки и управления этими процессами.

Центральное устройство (ЦУ) представляет основную компоненту ЭВМ и, в свою очередь, включает ЦП — центральный процессор (central processing unit — CPU) и ОП — оперативную (главную) память или оперативное запоминающее устройство — ОЗУ (синонимы — Main Storage, Core Storage, Random Access Memory — RAM).

Процессор непосредственно реализует операции обработки информации и управления вычислительным процессом, осуществляя выборку машинных команд и данных из оперативной памяти, их выполнение и запись результатов в ОП, включение и отключение ВУ. Основными блоками процессора являются:

- устройство управления (УУ) с интерфейсом процессора (системой сопряжения и связи процессора с другими узлами машины);
- арифметико-логическое устройство (АЛУ);
- процессорная память (внутренний кэш).

Оперативная память предназначена для временного хранения данных и программ в процессе выполнения вычислительных и логических операций.

Арифметико-логическое устройство (АЛУ). Arithmetic and Logical Unit (ALU) — часть процессора, выполняющая арифметические и логические операции над данными.

АЛУ реализует набор простых операций. Арифметической операцией называют процедуру обработки данных, аргументы и результат которой являются числами (сложение, вычитание, умножение, деление). Логической операцией именуют процедуру, осуществляющую построение сложного высказывания (операции И, ИЛИ, НЕ). АЛУ состоит из регистров, сумматора с соответствующими логическими схемами и блока управления выполняемым процессом. Устройство работает в соответствии с сообщаемыми ему кодами операций, которые должны быть выполнены над переменными, помещаемыми в регистры.

Внешние устройства (ВУ). ВУ обеспечивают эффективное взаимодействие компьютера с окружающей средой — пользователями, объектами управления, другими машинами.

В специализированных управляющих ЭВМ (технологические процессы, связь, ракеты и пр.) внешними устройствами ввода являются датчики (температуры, давления, расстояния и пр.), устройствами вывода — манипуляторы (гидро-, пневмо-, сервоприводы рулей, вентилей и др.).

В универсальных ЭВМ (человеко-машинная обработка информации) в качестве ВУ выступают терминалы, принтеры и др. устройства.

Интерфейсы (каналы связи) служат для сопряжения центральных узлов машины с ее внешними устройствами.

Однотипные ЦУ и устройства хранения данных могут использоваться в различных типах машин. Известны примеры того, как фирмы, начавшие свою деятельность с производства управляющих машин, совершенствуя свою продукцию, перешли к выпуску систем, которые в зависимости от конфигурации ВУ могут исполнять роль как универсальных, так и управляющих машин (машины Hewlett-Packard — HP и Digital Equipment Corporation — DEC).

Абстрактное центральное устройство

Перечислим основные понятия и рассмотрим структуру и функции абстрактного центрального устройства ЭВМ (рис. 2.23), арифметико-логическое устройство (АЛУ) (arithmetic and logic unit — ALU) которого предназначено для обработки целых чисел и битовых строк.

Команда, инструкция (instruction) — описание операции, которую нужно выполнить. Каждая команда характеризуется форматом, который определяет ее структуру. Типичная команда содержит:

- код операции (КОП), характеризующий тип выполняемого действия;
- адресную часть (АЧ), которая в общем случае включает:
  - номера (адреса) индексного (ИР) и базисного (БР) регистров;
  - адреса операндов — A1, A2 и т. д.

Цикл процессора — период времени, за который осуществляется выполнение команды исходной программы в машинном виде; состоит из нескольких *тактов.*

Такт работы процессора — промежуток времени между соседними импульсами (tick of the internal clock) *генератора тактовых импульсов,* частота которых есть *тактовая частота процессора. Такт процессора (такт синхронизации)* — квант времени, в течение которого осуществляется элементарная операция — выборка, сравнение, пересылка данных.

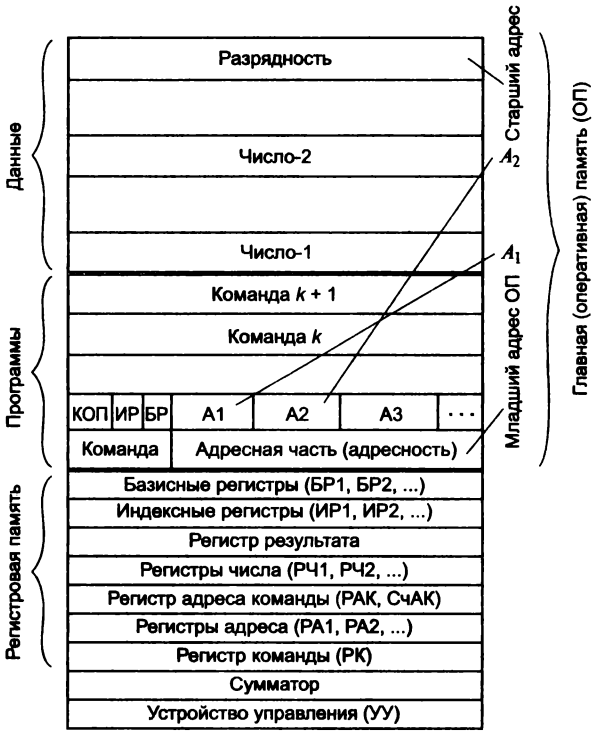


Рис. 2.23. Структура простейшего центрального устройства ЭВМ

Выполнение *короткой команды* — арифметика с ФТ (фиксированной запятой — ФЗ), логическая операция — занимает как минимум пять тактов (см. также рис. 3.1):

- выборка команды (`Fetch`);
- расшифровка кода операции/декодирование (`Instruction Decode`);
- вычисление адреса и выборка данных из памяти (`Address Generate, Load`)
- выполнение операции (`Execute`);
- запись результата в память (`Write Back, Store`).

Процедура, соответствующая каждому такту, реализуется определенной логической цепью (схемой) процессора, обычно именуемой м и к р о к о м а н д о й.

Регистры — устройства, предназначенные для временного хранения данных ограниченного размера (регистровое запоми-

нающее устройство — РЗУ). Важной характеристикой регистра является высокая скорость приема и выдачи данных. Регистр состоит из разрядов, в которые можно быстро записывать, запоминать и считывать слово, команду, двоичное число и т. д. Обычно регистр имеет ту же разрядность, что и машинное слово.

Регистр, обладающий способностью перемещать содержимое своих разрядов, называют *сдвиговым*. В этих регистрах за один такт хранимое слово поразрядно сдвигается на одну позицию.

Регистры общего назначения — РОН, регистры сверхоперативной памяти или регистровый файл — РФ (General Purpose Registers) — общее название для регистров, которые временно содержат данные, передаваемые в память или принимаемые из нее.

Регистр команды (РК, Instruction Register — IR) служит для размещения текущей команды, которая находится в нем в течение текущего цикла процессора.

Регистр (РАК), счетчик (СчАК) адреса команды (program counter — PC) — регистр, содержащий адрес текущей команды.

Регистр адреса (числа) — РА(Ч) — содержит адрес одного из операндов выполняемой команды (регистров может быть несколько).

Регистр числа (РЧ) содержит операнд выполняемой команды, этих регистров также несколько.

Регистр результата (РР) предназначается для хранения результата выполнения команды.

Сумматор — регистр, осуществляющий операции сложения (логического и арифметического двоичного) чисел или битовых строк, представленных в *прямом или обратном коде*. Регистр, хранящий промежуточные данные, часто именуют *аккумулятором*.

Существуют и другие регистры, не отмеченные на схеме, например *регистр состояния* — Status Register (SR) или регистр флагов. Типичным содержанием SR является информация об особых результатах завершения команды (ноль, переполнение, деление на ноль, перенос и пр.). УУ использует информацию из SR для исполнения условных переходов (например, «в случае переполнения перейти по адресу 4170»). Ниже более подробно будут рассмотрены *регистры процессора i8086*.

Цикл выполнения *короткой команды* может выглядеть следующим образом.

1. В соответствии с содержимым СчАК (адрес очередной команды) УУ извлекает из ОП очередную команду и помещает ее в РК. Некоторые команды УУ обрабатывает самостоятельно, без привлечения АЛУ (например, по команде «перейти по адресу 2478» величина 2478 сразу заносится в СчАК, и процессор переходит к выполнению следующей команды.

2. Осуществляется расшифровка (декодирование) команды.

3. Адреса А1, А2 и пр. помещаются в регистры адреса.

4. Если в команде указаны ИР или БР, то их содержимое используется для модификации РА — фактически выбираются числа или команды, смещенные в ту или иную сторону по отношению к адресу, указанному в команде.

5. По значениям РА осуществляется чтение чисел (строк) и помещение их в РЧ.

6. Выполнение операции и помещение результата в РР.

7. Запись результата по одному из адресов (если необходимо).

8. Увеличение содержимого СчАК на единицу (переход к следующей команде).

Очевидно, что за счет увеличения числа регистров возможно *распараллеливание, перекрытие* операций. Например, при считывании команды СчАК можно автоматически увеличить на 1, подготовив выборку следующей команды. После расшифровки текущей команды РК освобождается и в него может быть прочитана следующая команда. При выполнении операции возможна расшифровка следующей команды и т. д. Все это является предпосылкой построения так называемых конвейерных структур (*pipeline*). Однако все это хорошо только при последовательном (естественном) порядке выполнения команд. Появление переходов (особенно по условию, не определенному ранее) нарушает эту картину (в частности, увеличение СчАК на 1, упомянутое выше, оказывается недействительным). Поэтому современные процессоры пытаются предсказывать переходы в программе *(branch prediction)*.

Архитектуры ЭВМ

Архитектура «звезда». Здесь процессор (ЦУ) (рис. 2.24, *а*) соединен непосредственно с ВУ и управляет их работой (ранние модели машин). Этот тип также именуется классическая архитектура (фон Неймана) — одно арифмети-

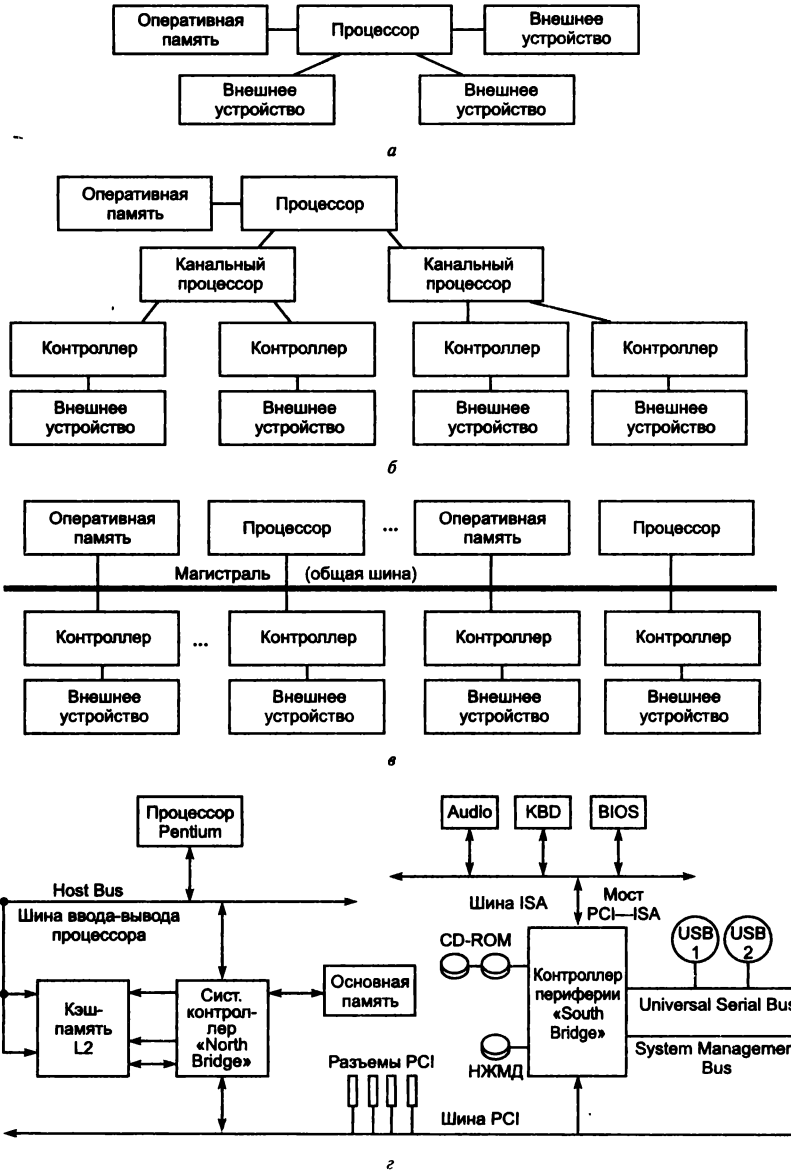


Рис. 2.24. Основные классы архитектур ЭВМ:
а — централизованная; *б* — иерархическая; *в* — магистральная; *г* — общая структура персонального компьютера (архитектура Triton 430 TX — Northbridge/Southbridge)

ко-логическое устройство (АЛУ), через которое проходит поток данных, и одно устройство управления (УУ), через которое проходит поток команд — программа. Это однопроцессорный компьютер.

Принстонская и гарвардская архитектуры. Архитектура фон Неймана часто ассоциируется с принстонской архитектурой, которая характеризуется использованием общей оперативной памяти для хранения программ и данных.

Альтернативная — гарвардская архитектура (название связано с компьютером «Марк-I» (1950 г.), в котором использовалась отдельная память для команд) характеризуется физическим разделением памяти команд (программ) и памяти данных. Каждая память соединяется с процессором отдельной шиной, что позволяет одновременно с чтением-записью данных при выполнении текущей команды производить выборку и декодирование следующей команды.

Гарвардская архитектура появляется в современных процессорах, когда в кэш-памяти ЦП выделяется память команд (I-Cache) и память данных (D-Cache).

Иерархическая архитектура (рис. 2.24, *б*) — ЦУ соединено с периферийными процессорами (вспомогательными процессорами, каналами, канальными процессорами), управляющими в свою очередь контроллерами, к которым подключены группы ВУ (системы IBM 360-375, ЕС ЭВМ);

Магистральная структура (общая шина — unibas, рис. 2.24, *в*). Процессор (процессоры) и блоки памяти (ОП) взаимодействуют между собой и с ВУ (контроллерами ВУ) через внутренний канал, общий для всех устройств (машины DEC, IBM PC-совместимые ПЭВМ). Физически *магистраль* представляет собой многопроводную линию с гнездами для подключения электронных схем. Совокупность линий магистрали разделяется на отдельные группы — шину адреса, шину данных и шину управления.

К этому типу архитектуры относится также архитектура персонального компьютера (ПК). Конечно, реальная структура ПК (рис. 2.24, *г*) отличается от теоретических схем — в ней используется несколько разновидностей шинных интерфейсов, которые соединяются между собой мостами — контроллерами памяти (Northbridge) и периферийных устройств (Southbridge).

2.4. Классы и архитектуры вычислительных систем и суперкомпьютеров

Вычислительная система (ВС) — совокупность взаимосвязанных и взаимодействующих процессоров или ЭВМ, периферийного оборудования и программного обеспечения, предназначенная для сбора, хранения, обработки и распределения информации.

Создание ВС преследует следующие основные цели:

- повышение производительности системы за счет ускорения процессов обработки данных;
- повышение надежности и достоверности вычислений;
- предоставление пользователям дополнительных сервисных услуг и т. д.

Отличительной особенностью ВС по отношению к классическим ЭВМ является наличие в ней нескольких вычислителей, реализующих *параллельную обработку*.

Параллелизм выполнения операций существенно повышает быстродействие системы; он может также значительно повысить и надежность (при отказе одного компонента системы его функции может взять на себя другой), и достоверность функционирования системы, если операции будут дублироваться, а результаты их выполнения сравниваться.

Если не вдаваться в подробности, ВС прежде всего можно разделить на:

- многомашинные;
- многопроцессорные.

Многомашинная вычислительная система

Здесь несколько процессоров, входящих в вычислительную систему, не имеют общей оперативной памяти, а имеют каждый свою (локальную). Каждый компьютер в многомашинной системе имеет классическую архитектуру, однако эффект от применения такой вычислительной системы может быть получен только при решении задач, имеющих очень специальную структуру: она должна разбиваться на столько слабо связанных подзадач, сколько компьютеров в системе.

Многопроцессорная архитектура. Наличие в компьютере нескольких процессоров означает, что параллельно может быть организовано много потоков данных и много потоков команд. Таким образом, параллельно могут выполняться несколько фраг-

ментов одной задачи. Преимущество в быстродействии многопроцессорных и многомашинных вычислительных систем перед однопроцессорными очевидно.

Архитектура с параллельными процессорами. Здесь несколько АЛУ работают под управлением одного УУ. Это означает, что множество данных может обрабатываться по одной программе, т. е. по одному потоку команд. Высокое быстродействие такой архитектуры можно получить только на задачах, в которых одинаковые вычислительные операции выполняются одновременно на различных однотипных наборах данных.

Уровни и средства комплексирования. Логические и физические уровни

В создаваемых ВС стараются обеспечить несколько путей передачи данных, что позволяет достичь необходимой надежности функционирования, гибкости и адаптируемости к конкретным условиям работы. Эффективность обмена информацией определяется скоростью передачи и возможными объемами данных, передаваемыми по каналу взаимодействия. Эти характеристики зависят от средств, обеспечивающих взаимодействие модулей, и уровня управления процессами, на котором это взаимодействие осуществляется. Сочетание различных уровней и методов обмена данными между модулями ВС наиболее полно представлено в универсальных суперЭВМ и больших ЭВМ, в которых сбалансированно использовались основные методы достижения высокой производительности. В этих машинах предусматривались следующие уровни комплексирования (рис. 2.25):

1) прямого управления (процессор—процессор);
2) общей оперативной памяти;
3) комплексируемых каналов ввода-вывода;
4) устройств управления внешними устройствами (УВУ);
5) общих внешних устройств.

На каждом из этих уровней используются специальные технические и программные средства, обеспечивающие обмен информацией.

Уровень прямого управления служит для передачи коротких однобайтовых приказов-сообщений. Последовательность взаимодействия процессоров сводится к следующему. Процессор-инициатор обмена по интерфейсу прямого управления (ИЛУ) переда-

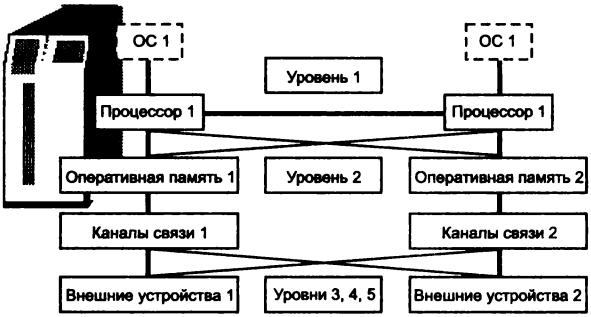


Рис. 2.25. Уровни комплексирования машин в вычислительную систему

ет в блок прямого управления байт-сообщение и подает команду «прямая запись». У другого процессора эта команда вызывает прерывание, относящееся к классу внешних. В ответ он вырабатывает команду «прямое чтение» и записывает передаваемый байт в свою память. Затем принятая информация расшифровывается и по ней принимается решение. После завершения передачи прерывания снимаются, и оба процессора продолжают вычисления по собственным программам. Видно, что уровень прямого управления не может использоваться для передачи больших массивов данных, однако оперативное взаимодействие отдельными сигналами широко используется в управлении вычислениями. У ПЭВМ типа IBM PC этому уровню соответствует комплексирование процессоров, подключаемых к системной шине.

Уровень общей оперативной памяти (ООП) является наиболее предпочтительным для оперативного взаимодействия процессоров. В этом случае ООП эффективно работает при небольшом числе обслуживаемых абонентов.

Уровень комплексируемых каналов ввода-вывода предназначается для передачи больших объемов информации между блоками оперативной памяти, сопрягаемых в ВС. Обмен данными между ЭВМ осуществляется с помощью адаптера «канал—канал» (АКК) и команд «чтение» и «запись». Адаптер — это устройство, согласующее скорости работы сопрягаемых каналов. Обычно сопрягаются селекторные каналы (СК) машин как наиболее быстродействующие. Скорость обмена данными определяется скоростью самого медленного канала. Скорость передачи данных по этому уровню составляет несколько Мбайт в секунду. В ПЭВМ данному уровню взаимодействия соответствует подключение периферийной аппаратуры через контроллеры и адаптеры.

Уровень устройств управления внешними устройствами (УВУ) предполагает использование встроенного в УВУ двухканального переключателя и команд «зарезервировать» и «освободить». Двухканальный переключатель позволяет подключать УВУ одной машины к селекторным каналам различных ЭВМ. По команде «зарезервировать» канал — инициатор обмена имеет доступ через УВУ к любым накопителям на дисках НМД или на магнитных лентах НМЛ. На самом деле УВУ магнитных дисков и лент — совершенно различные устройства. Обмен канала с накопителями продолжается до полного завершения работ и получения команды «освободить». Только после этого УВУ может подключиться к конкурирующему каналу. Только такая дисциплина обслуживания требований позволяет избежать конфликтных ситуаций.

На ***четвертом уровне*** с помощью аппаратуры передачи данных (АПД) (мультиплексоры, сетевые адаптеры, модемы и др.) имеется возможность сопряжения с каналами связи. Эта аппаратура позволяет создавать сети ЭВМ.

Пятый уровень предполагает использование *общих внешних устройств*. Для подключения отдельных устройств используется автономный двухканальный переключатель.

Пять уровней комплексирования получили название *логических* потому, что они объединяют на каждом уровне разнотипную аппаратуру, имеющую сходные методы управления. Каждое из устройств может иметь логическое имя, используемое в прикладных программах. Этим достигается независимость программ пользователей от конкретной физической конфигурации системы. Связь логической структуры программы и конкретной физической структуры ВС обеспечивается операционной системой по указаниям — директивам пользователя, при генерации ОС и по указаниям диспетчера-оператора вычислительного центра. Различные уровни комплексирования позволяют создавать самые различные структуры ВС.

Второй логический уровень позволяет создавать многопроцессорные ВС. Обычно он дополняется и первым уровнем, что позволяет повышать оперативность взаимодействия процессоров. Вычислительные системы сверхвысокой производительности должны строиться как многопроцессорные. Центральным блоком такой системы является быстродействующий коммутатор, обеспечивающий необходимые подключения абонентов (процессоров и каналов) к общей оперативной памяти.

Уровни 1, 3, 4, 5 обеспечивают построение разнообразных машинных комплексов. Особенно часто используется третий в комбинации с четвертым. Целесообразно их дополнять и первым уровнем.

Пятый уровень комплексирования используется в редких специальных случаях, когда в качестве внешнего объекта используется какое-то дорогое уникальное устройство. В противном случае этот уровень малоэффективен. Любое внешнее устройство — это недостаточно надежное устройство точной механики, а значит, выгоднее использовать четвертый уровень комплексирования, когда можно сразу управлять не одним, а несколькими внешними устройствами, включая и резервные.

Чтобы дать более полное представление о многопроцессорных вычислительных системах, помимо высокой производительности необходимо назвать и другие отличительные особенности. Прежде всего это необычные архитектурные решения, направленные на повышение производительности (работа с векторными операциями, организация быстрого обмена сообщениями между процессорами или организация глобальной памяти в многопроцессорных системах и др.).

Несмотря на то, что классическим является *многомашинный* вариант ВС, в ВС может быть только один компьютер, но агрегированный с многофункциональным периферийным оборудованием (стоимость периферийного оборудования часто превосходит стоимость центральных устройств компьютера). В компьютере может быть как несколько процессоров (тогда имеет место также классический многопроцессорный вариант ВС), так и один процессор (если не брать в расчет специализированные процессоры, входящие в состав периферийных устройств).

Следует отметить, что здесь возникает проблема масштабируемости — использование нескольких процессоров обычно не приводит к пропорциональному приросту производительности (рис. 2.26).

Если, например, одиночный процессор (1 × П) простаивает 20 % своего времени, ожидая данные из оперативной памяти, то 2 × П будет простаивать 33 % времени, а 4 × П — 50 %. В пересчете на общую производительность, если 1 × П-система работает со скоростью 100 %, то 2 × П-система — со скоростью 167 % (вместо ожидаемых 200 %), а 4 × П-система — со скоростью 250 % (вместо 400 %).

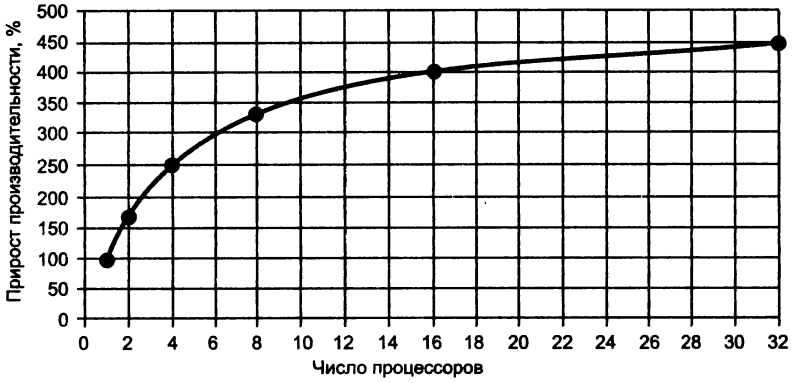


Рис. 2.26. Типичная кривая зависимости производительности многопроцессорной системы от числа процессоров

В принципе любая многопроцессорная система должна быть хорошо сбалансирована — слишком быстрая память обходится чересчур дорого; слишком медленная — сводит эффект от установки нескольких процессоров к минимуму.

Классификация архитектур вычислительных систем с параллельной обработкой данных (М. Флинн, 1966 г.)

В 1966 г. М. Флинном (Flynn) была предложена классификация архитектур ЭВМ и вычислительных систем, в основу которой положено понятие потока, или последовательности элементов (команд или данных), обрабатываемых процессором. Соответствующая система классификации, основанная на рассмотрении числа потоков команд и потоков данных, приводит к четырем базовым классам (табл. 2.10, рис. 2.27).

Кратко опишем характерные особенности каждой из архитектур.

Архитектура ОКОД охватывает все однопроцессорные и одномашинные варианты систем — все ЭВМ классической структуры попадают в этот класс. Здесь параллелизм вычислений обеспечивается путем конвейеризации и распараллеливания потока микрокоманд между исполнительными устройствами (см. рис. 3.1).

Архитектура ОКМД предполагает создание структур векторной или матричной обработки. Системы этого типа обычно

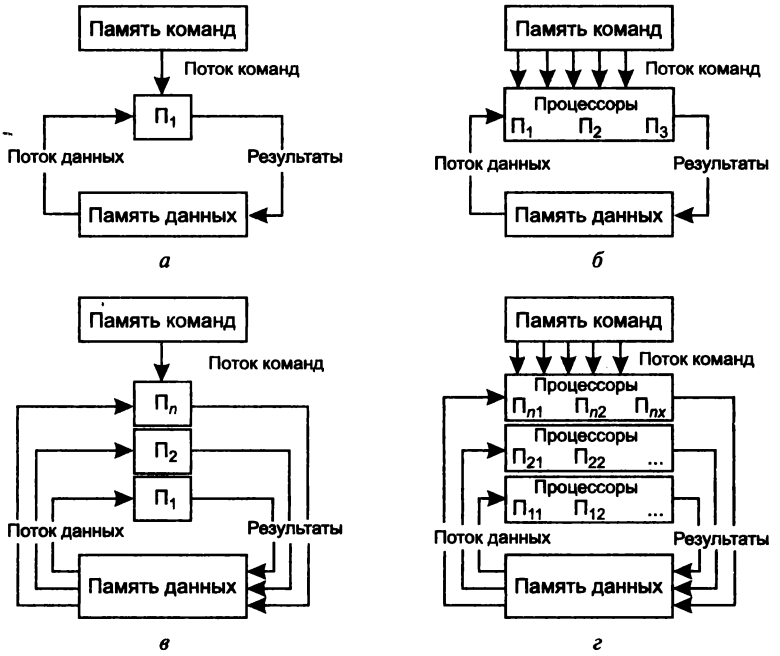


Рис. 2.27. Классификация Флинна:
а — SISD; *б* — MISD; *в* — SIMD; *г* — MIMD

Таблица 2.10. **Классификация Флинна**

Поток данных	Поток команд	
	Одиночный	Множественный
Одиночный	SISD — Single Instruction stream/Single Data stream (Одиночный поток Команд и Одиночный поток Данных — ОКОД)	MISD — Multiple Instruction stream/Single Data stream (Множественный поток Команд и Одиночный поток Данных — МКОД)
Множественный	SIMD — Single Instruction stream/Multiple Data stream (Одиночный поток Команд и Множественный поток Данных — ОКМД)	MIMD — Multiple Instruction stream/Multiple Data stream (Множественный поток Команд и Множественный поток Данных — МКМД)

строятся как однородные, т. е. процессорные элементы, входящие в систему, идентичны, и все они управляются одной и той же последовательностью команд. Однако каждый процессор обрабатывает свой поток данных. Под эту схему хорошо подходят задачи обработки матриц или векторов (массивов), задачи реше-

ния систем линейных и нелинейных, алгебраических и дифференциальных уравнений, задачи теории поля и др.

Третий тип архитектуры (МКОД) предполагает построение своеобразного процессорного конвейера, в котором результаты обработки передаются от одного процессора к другому по цепочке. Прототипом таких вычислений может служить схема любого производственного конвейера. В современных ЭВМ по этому принципу реализована схема совмещения операций, в которой параллельно работают различные функциональные блоки, и каждый из них делает свою часть в общем цикле обработки команды. В ВС этого типа конвейеры должны образовывать группы процессоров.

Архитектура МКМД предполагает, что все процессоры системы работают по своим программам с собственным потоком команд. В простейшем случае они могут быть автономны и независимы. Такая схема использования ВС часто применяется во многих крупных вычислительных центрах для увеличения пропускной способности центра.

Другие подходы к классификации ВС

Наличие большого разнообразия систем, образующих класс МКМД (MIMD), делает классификацию Флинна не полностью адекватной. Действительно и 4-процессорный SX-5 компании NEC и 1000-процессорный Cray T3E попадают в класс MIMD. Это заставляет искать другие основания классификации.

Классификация Джонсона. Е. Джонсон предложил проводить классификацию MIMD-архитектур на основе структуры памяти и реализации механизма взаимодействия и синхронизации между процессорами.

По структуре оперативной памяти существующие вычислительные системы делятся на две большие группы: либо это системы с общей памятью, прямо доступной всем процессорам, либо это системы с распределенной памятью, каждая часть которой доступна только одному процессору. Одновременно с этим и для межпроцессорного взаимодействия существуют две альтернативы — через разделяемые переменные или с помощью механизма передачи сообщений. Исходя из таких предположений, можно получить четыре класса MIMD-архитектур, уточняющих систематику Флинна (табл. 2.11).

Таблица 2.11. **Классификация Джонсона для систем MIMD по Флинну**

Обмен данными	Память	
	Общая	Распределенная
Общие данные	GMSV — General Memory-Shared variables (Общая память — разделяемые переменные) Класс 1. «Системы с разделяемой памятью»	DMSV — Distributed Memory, Shared variables (Распределенная память — разделяемые переменные) Класс 2. «Гибридная архитектура»
Передача данных	GMMP — General Memory, Message propagation (Общая память — передача сообщений)	DMMP — Distributed Memory, Message propagation (Распределенная память — передача сообщений) Класс 3. «Архитектуры с передачей сообщений»

Опираясь на такое деление, Джонсон вводит следующие наименования для некоторых классов:

- вычислительные системы, использующие общую разделяемую память для межпроцессорного взаимодействия и синхронизации, он называет *системами с разделяемой памятью*, например CRAY Y-MP (по его классификации это класс 1);
- системы, в которых память распределена по процессорам, а для взаимодействия и синхронизации используется механизм передачи сообщений, называются *архитектурами с передачей сообщений*, например NCube (класс 3);
- системы с распределенной памятью и синхронизацией через разделяемые переменные, как в BBN Butterfly, называются *гибридными архитектурами* (класс 2).

В качестве уточнения классификации автор отмечает возможность учитывать вид связи между процессорами: общая шина, переключатели, разнообразные сети и т. п.

Классификация Базу. По мнению А. Базу (A. Basu), любую параллельную вычислительную систему можно однозначно описать последовательностью решений, принятых на этапе ее проектирования, а сам процесс проектирования представить в виде дерева. Корень дерева — это вычислительная система (рис. 2.28) и последующие ярусы дерева, фиксируя уровень параллелизма, метод реализации алгоритма, параллелизм инструкций и способ управления, последовательно дополняют друг друга, формируя описание системы.

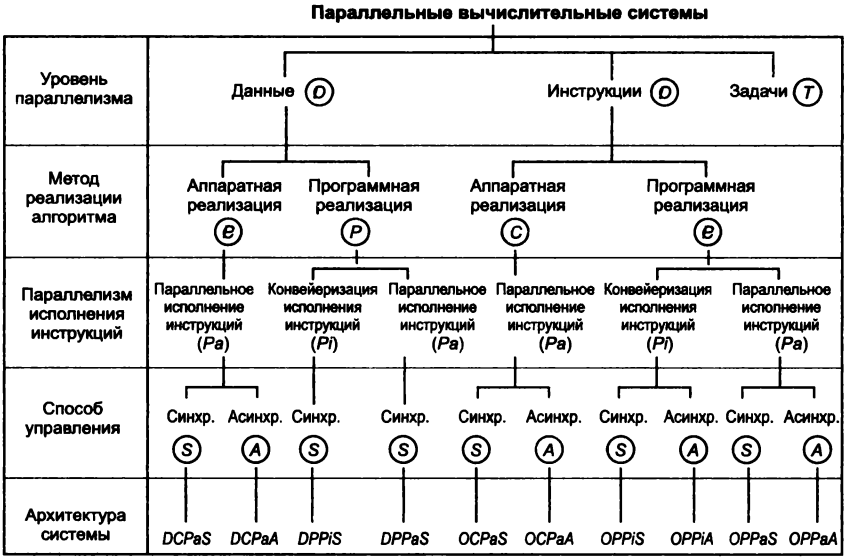


Рис. 2.28. Классификация Базу

На первом этапе определяется, какой уровень параллелизма использует вычислительная система. Одна и та же операция может одновременно выполняться над целым набором данных, определяя параллелизм на уровне данных (обозначено *D* на рис. 2.28). Способность выполнять более одной операции одновременно говорит о параллелизме на уровне команд (*O*). Если же компьютер спроектирован так, что целые последовательности команд могут быть выполнены одновременно, то будем говорить о параллелизме на уровне задач (*T*).

Второй уровень в классификационном дереве фиксирует метод реализации алгоритма. С появлением сверхбольших интегральных схем (СБИС) стало возможным реализовывать аппаратно не только простые арифметические операции, но и алгоритмы целиком. Например, быстрое преобразование Фурье, перемножение матриц и другие относятся к классу тех алгоритмов, которые могут быть эффективно реализованы в СБИС. Данный уровень классификации разделяет системы с аппаратной реализацией алгоритмов (*C* на рис. 2.28) и системы, использующие традиционный способ программной реализации (*P*).

Третий уровень конкретизирует тип параллелизма, используемого для обработки инструкций машины, — конвейеризация

инструкций (P_i) или их независимое (параллельное) выполнение (P_a). В большей степени этот выбор относится к компьютерам с программной реализацией алгоритмов, так как аппаратная реализация всегда предполагает параллельное исполнение команд. Отметим, что в случае конвейерного исполнения имеется в виду лишь конвейеризация самих команд, разбивающая весь цикл обработки на выборку команды, дешифрацию, вычисление адресов и т. д. (возможная конвейеризация вычислений на данном уровне не принимается во внимание).

Последний уровень данной классификации определяет способ управления, принятый в вычислительной системе: синхронный (S) или асинхронный (A). Если выполнение команд происходит в строгом порядке, определяемом только сигналами таймера и счетчиком команд, то говорят о синхронном способе управления. Если же для инициации команды определяющими являются такие факторы, как, например, готовность данных, то машина попадает в класс с асинхронным управлением. Наиболее характерными представителями систем с асинхронным управлением являются компьютеры *data-driven* и *demand-driven*.

Классификация Дункана. Р. Дункан определяет тот набор требований, на который может опираться искомая классификация.

Из класса параллельных машин должны быть исключены те, в которых параллелизм заложен лишь на самом низком уровне, включая:

- конвейеризацию на этапе подготовки и выполнения команды (instruction pipelining), т. е. частичное перекрытие таких этапов, как дешифрация команды, вычисление адресов операндов, выборка операндов, выполнение команды и сохранение результата;
- наличие в архитектуре нескольких функциональных устройств, работающих независимо, в частности, возможность параллельного выполнения логических и арифметических операций;
- наличие отдельных процессоров ввода-вывода, работающих независимо и параллельно с основными процессорами.

Причины исключения перечисленных выше особенностей ее автор объясняет следующим образом. Если рассматривать компьютеры, использующие только параллелизм низкого уровня, наравне со всеми остальными, то, во-первых, практически все существующие системы будут классифицированы как «параллельные» (что заведомо не будет позитивным фактором для

классификации), и, во-вторых, такие машины будут плохо вписываться в любую модель или концепцию, отражающую параллелизм высокого уровня.

Классификация должна быть согласованной с классификацией Флинна, показавшей правильность выбора идеи потоков команд и данных.

Классификация должна описывать архитектуры, которые однозначно не укладываются в систематику Флинна, но тем не менее относятся к параллельным архитектурам (например, векторно-конвейерные).

Учитывая вышеизложенные требования, Дункан дает неформальное определение параллельной архитектуры, причем именно неформальность дала ему возможность включить в данный класс компьютеры, которые ранее не вписывались в систематику Флинна. Итак, *параллельная архитектура* — это такой способ организации вычислительной системы, при котором допускается, чтобы множество процессоров (простых или сложных) могло бы работать одновременно, взаимодействуя по мере надобности друг с другом. Следуя этому определению, все разнообразие параллельных архитектур Дункан систематизирует так, как это показано на рис. 2.29.

По существу систематика очень простая: процессоры системы работают либо синхронно, либо независимо друг от друга, либо в архитектуру системы заложена та или иная модификация идеи MIMD. На следующем уровне происходит детализация в рамках каждого из этих трех классов. Дадим небольшое поясне-

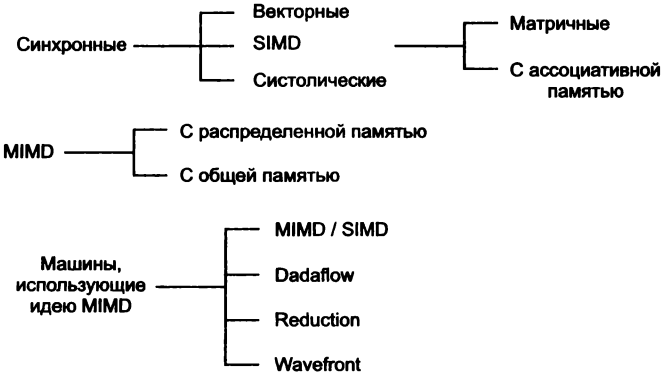


Рис. 2.29. Классификация Дункана

ние лишь к тем из них, которые на сегодняшний день не столь широко известны.

Систолические архитектуры (их чаще называют систолическими массивами) представляют собой множество процессоров, объединенных регулярным образом (например, система WARP). Обращение к памяти может осуществляться только через определенные процессоры на границе массива. Выборка операндов из памяти и передача данных по массиву осуществляется в одном и том же темпе. Направление передачи данных между процессорами фиксировано. Каждый процессор за интервал времени выполняет небольшую инвариантную последовательность действий.

Гибридные MIMD/SIMD-архитектуры, вычислительные системы *dataflow*, *reduction* и *wavefront* осуществляют параллельную обработку информации на основе асинхронного управления, как и MIMD-системы. Но они выделены в отдельную группу, поскольку все имеют ряд специфических особенностей, которыми не обладают системы, традиционно относящиеся к MIMD.

MIMD/SIMD — типично гибридная архитектура. Она предполагает, что в MIMD-системе можно выделить группу процессоров, представляющую собой подсистему, работающую в режиме SIMD (например, PASM, Non-Von). Такие системы отличаются относительной гибкостью, поскольку допускают реконфигурацию в соответствии с особенностями решаемой прикладной задачи.

Остальные три вида архитектур используют нетрадиционные модели вычислений. *Dataflow*-машины используют модель, в которой команда может выполняться сразу же, как только вычислены необходимые операнды. Таким образом, последовательность выполнения команд определяется зависимостью по данным, которая может быть выражена, например, в форме графа.

Модель вычислений, применяемая в *reduction*-машинах, иная и состоит в следующем: команда становится доступной для выполнения тогда и только тогда, когда результат ее работы требуется другой, доступной для выполнения команде в качестве операнда.

Архитектура *wavefront array* объединяет в себе идею систолической обработки данных и модель вычислений, используемую в *dataflow*-машинах. В данной архитектуре процессоры объединяются в модули и связи, по которым процессоры могут взаимодействовать друг с другом, фиксируются. Однако, в противоположность ритмичной работе систолических массивов, данная

архитектура использует асинхронный механизм связи с подтверждением (*handshaking*), из-за этого «фронт волны» вычислений может менять свою форму по мере перемещения по всему множеству процессоров.

Классификация Кришнамарфи. Е. Кришнамарфи для классификации параллельных вычислительных систем предлагает использовать четыре характеристики, похожие на характеристики классификации А. Базу (рис. 2.30):

- степень гранулярности;
- способ реализации параллелизма;
- топологию и природу связи процессоров;
- способ управления процессорами.

Принцип построения классификации достаточно прост. Для каждой степени гранулярности рассматриваются все возможные способы реализации параллелизма. Для каждого полученного таким образом варианта устанавливаются все комбинации топологии связи и способов управления процессорами. В результате получается дерево (см. рис. 2.30), в котором каждый ярус соответствует своей характеристике, каждый лист представляет отдельную группу компьютеров в данной классификации, а путь от вершины дерева однозначно определяет значения указанных выше характеристик.

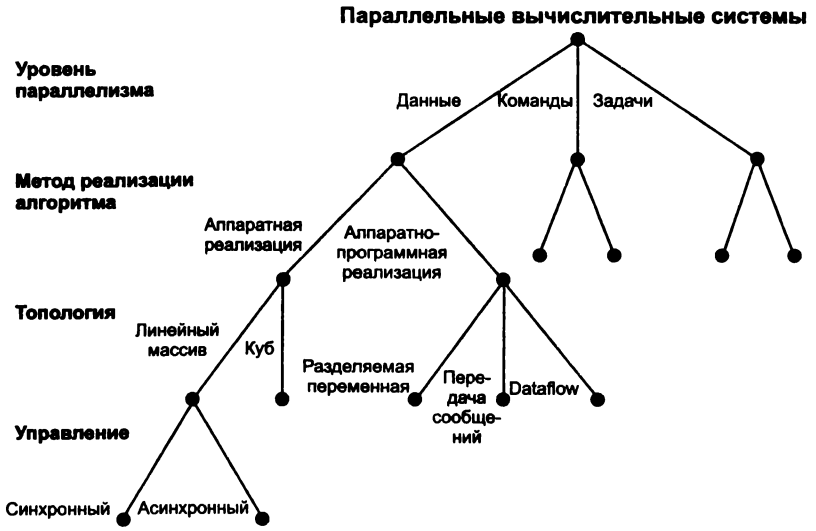


Рис. 2.30. Классификация Кришнамарфи

Первые два уровня практически повторяют классификацию А. Базу. Третий уровень классификации (топология и природа связи процессоров) тесно связан со вторым. Если был выбран аппаратный способ реализации параллелизма, то надо рассмотреть топологию связи процессоров (матрица, линейный массив, тор, дерево, звезда и т. п.) и степень связности процессоров между собой (сильная, слабая или средняя), которая определяется относительной долей накладных расходов при организации взаимодействия процессоров. В случае комбинированной реализации параллелизма, помимо топологии и степени связности, надо дополнительно учесть механизм взаимодействия процессоров: передача сообщений, разделяемые переменные или принцип *dataflow* (по готовности операндов).

Наконец, последний, четвертый уровень — способ управления процессорами, определяет общий принцип функционирования всей совокупности процессоров вычислительной системы: синхронный, *dataflow* или асинхронный.

На основе выделенных четырех характеристик нетрудно определить место наиболее известных классов архитектур в данной систематике.

Векторно-конвейерные компьютеры:

- гранулярность — на уровне данных;
- реализация параллелизма — аппаратная;
- связь процессоров — простая топология со средней связностью;
- способ управления — синхронный.

Классические мультипроцессоры:

- гранулярность — на уровне задач
- реализация параллелизма — комбинированная;
- связь процессоров — простая топология со слабой связностью и использованием разделяемых переменных;
- способ управления — асинхронный.

Матричные процессоры:

- гранулярность — на уровне данных;
- реализация параллелизма — аппаратная;
- связь процессоров — двумерные массивы с сильной связностью;
- способ управления — синхронный.

Систолические массивы:

- гранулярность — на уровне данных;
- реализация параллелизма — аппаратная;

- связь процессоров — сложная топология с сильной связностью;
- способ управления — синхронный.

Архитектура типа wavefront:

- гранулярность — на уровне данных;
- реализация параллелизма — аппаратная;
- связь процессоров — двумерная топология с сильной связностью;
- способ управления — *dataflow*.

Архитектура типа dataflow:

- гранулярность — на уровне команд;
- реализация параллелизма — комбинированная;
- связь процессоров — простая топология с сильной либо средней связностью и использованием принципа *dataflow*;
- способ управления — асинхронно-*dataflow*.

Несмотря на то, что классификация Е. Кришнамарфи построена только на четырех признаках, она позволяет выделить и описать такие «нетрадиционные» параллельные системы, как систолические массивы, машины типа *dataflow* и *wavefront*. Однако эта же простота является и основной причиной ее недостатков: некоторые архитектуры нельзя однозначно отнести к тому или иному классу, например компьютеры с архитектурой гиперкуба и ассоциативные процессоры. Для более точного описания таких машин потребуется ввести еще целый ряд характеристик, таких, как размещение задач по процессорам, способ маршрутизации сообщений, возможность реконфигурации, аппаратная поддержка языков программирования и другие. Вместе с тем ясно, что эти признаки формализовать гораздо труднее, поэтому есть опасность вместо ясности внести в описание лишь дополнительные трудности.

Классификация Скилликорна. Д. Скилликорн разработал подход, ориентированный как на описание свойств многопроцессорных систем, так и некоторых нетрадиционных архитектур, в частности *dataflow*- и *reduction*-машины.

Предлагается рассматривать архитектуру любого компьютера как абстрактную структуру, состоящую из четырех компонентов:

- *процессор команд* (IP — Instruction Processor) — функциональное устройство, работающее как интерпретатор команд (в системе, вообще говоря, может отсутствовать);

- *процессор данных* (DP — Data Processor) — функциональное устройство, работающее как преобразователь данных, в соответствии с арифметическими операциями;
- *иерархия памяти* (IM — Instruction Memory, DM — Data Memory) — запоминающее устройство, в котором хранятся данные и команды, пересылаемые между процессорами;
- *переключатель* — абстрактное устройство, обеспечивающее связь между процессорами и памятью.

Функции процессора команд во многом схожи с функциями устройств управления последовательных машин и, согласно Д. Скилликорну, сводятся к следующим:

- на основе своего состояния и полученной от DP информации IP определяет адрес команды, которая будет выполняться следующей;
- осуществляет доступ к IM для выборки команды;
- получает и декодирует выбранную команду;
- сообщает DP команду, которую надо выполнить;
- определяет адреса операндов и посылает их в DP;
- получает от DP информацию о результате выполнения команды.

Функции процессора данных делают его во многом похожим на арифметическое устройство традиционных процессоров:

- DP получает от IP команду, которую надо выполнить;
- получает от IP адреса операндов;
- выбирает операнды из DM;
- выполняет команду;
- запоминает результат в DM;
- возвращает в IP информацию о состоянии после выполнения команды.

В терминах таким образом определенных основных частей компьютера структуру традиционной фон-неймановской архитектуры можно представить в следующем виде — рис. 2.31.

Это один из самых простых видов архитектуры, не содержащих переключателей. Для описания параллельных вычислительных систем автор зафиксировал четыре типа переключателей, без какой-либо явной связи с типом устройств, которые они соединяют:

- 1—1 — переключатель такого типа связывает пару функциональных устройств;

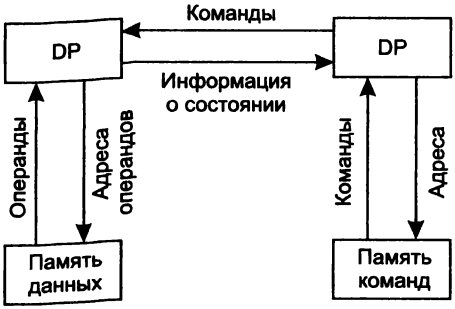


Рис. 2.31. Представление фон-неймановской архитектуры по Скилликорну

- *n*—*n* — переключатель связывает *i*-е устройство из одного множества устройств с *i*-м устройством из другого множества, т. е. фиксирует попарную связь;
- 1—*n* — переключатель соединяет одно выделенное устройство со всеми функциональными устройствами из некоторого набора;
- *n* × *n* — каждое функциональное устройство одного множества может быть связано с любым устройством другого множества и наоборот.

Примеров подобных переключателей можно привести очень много. Так, все матричные процессоры имеют переключатель типа 1—*n* для связи единственного процессора команд со всеми процессорами данных. В компьютерах семейства Connection Machine каждый процессор данных имеет свою локальную память, следовательно, связь будет описываться как *n*—*n*. В то же время каждый процессор команд может связаться с любым другим процессором, поэтому данная связь будет описана как *n* × *n*.

Классификация Д. Скилликорна состоит из двух уровней. На первом уровне она проводится на основе восьми характеристик:

- количества процессоров команд (IP);
- числа запоминающих устройств (модулей памяти) команд (IM);
- типа переключателя между IP и IM;
- количества процессоров данных (DP);
- числа запоминающих устройств (модулей памяти) данных (DM);
- типа переключателя между DP и DM;
- типа переключателя между IP и DP;
- типа переключателя между DP и DP.

Используя введенные характеристики и предполагая, что рассмотрение количественных характеристик можно ограничить только тремя возможными вариантами значений: 0, 1 и *n* (т. е. больше одного), можно получить 28 классов архитектур.

В классах 1—5 находятся компьютеры типа *dataflow* и *reduction*, не имеющие процессоров команд в обычном понимании этого слова. Класс 6 — это классическая фон-неймановская последовательная машина. Все разновидности матричных процессоров содержатся в классах 7—10. Классы 11 и 12 отвечают компьютерам типа MISD классификации Флинна и на настоящий момент, по-видимому, не заполнены. Классы с 13-го по 28-й занимают всевозможные варианты мультипроцессоров, причем в классах 13—20 находятся машины с достаточно привычной архитектурой, в то время, как архитектура классов 21—28 пока выглядит экзотично.

Классификация Хендлера. В основу классификации В. Хендлер закладывает явное описание возможностей параллельной и конвейерной обработки информации вычислительной системой. При этом он намеренно не рассматривает различные способы связи между процессорами и блоками памяти и считает, что коммуникационная сеть может быть нужным образом сконфигурирована и будет способна выдержать предполагаемую нагрузку (рис. 2.32).

Предложенная классификация базируется на различии между тремя уровнями обработки данных в процессе выполнения программ:

- уровень выполнения программы: опираясь на счетчик команд и некоторые другие регистры, устройство управле-

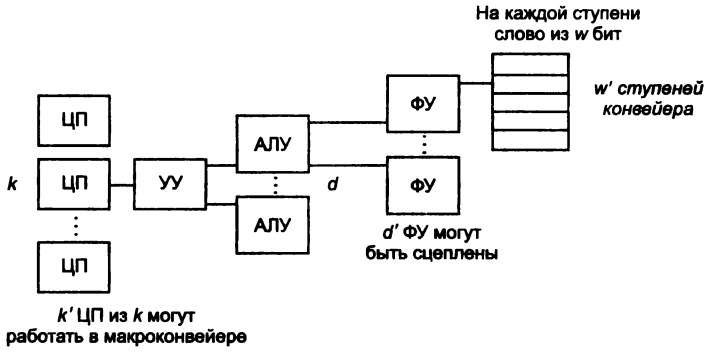


Рис. 2.32. Классификация Хендлера

ния (УУ) производит выборку и дешифрацию команд программы;
- уровень выполнения команд: арифметико-логическое устройство компьютера (АЛУ) исполняет команду, выданную ему устройством управления;
- уровень битовой обработки: все элементарные логические схемы процессора (ЭЛС) разбиваются на группы, необходимые для выполнения операций над одним двоичным разрядом.

Таким образом, подобная схема выделения уровней предполагает, что вычислительная система включает какое-то число процессоров, каждый со своим устройством управления. Каждое устройство управления связано с несколькими арифметико-логическими устройствами, исполняющими одну и ту же операцию в каждый конкретный момент времени. Наконец, каждое АЛУ объединяет несколько элементарных логических схем, ассоциированных с обработкой одного двоичного разряда (число ЭЛС есть ничто иное, как длина машинного слова). Если на какое-то время не рассматривать возможность конвейеризации, то число устройств управления k, число арифметико-логических устройств d в каждом устройстве управления и число элементарных логических схем w в каждом АЛУ составят тройку для описания данной вычислительной системы C:

$$t(C) = (k, d, w).$$

Классификация Хокни. Р. Хокни (английский специалист в области параллельных вычислительных систем) разработал свой подход к классификации, введенной им для систематизации компьютеров, попадающих в класс MIMD по систематике Флинна.

Как отмечалось ранее (см. классификацию Флинна), класс MIMD чрезвычайно широк, причем наряду с большим числом компьютеров он объединяет и целое множество различных типов архитектур. Хокни, пытаясь систематизировать архитектуры внутри этого класса, получил иерархическую структуру, представленную на рис. 2.33.

Основная идея классификации состоит в следующем. Множественный поток команд может быть обработан двумя способами: либо одним конвейерным устройством обработки, работающем в режиме разделения времени для отдельных потоков, либо каждый поток обрабатывается своим собственным устройством.

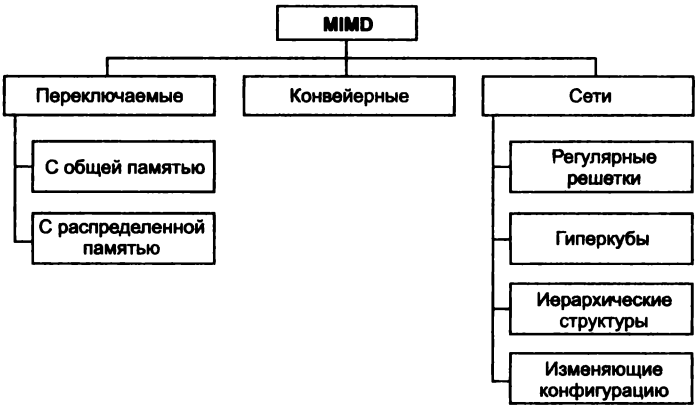


Рис. 2.33. Системы архитектуры MIMD (Флинн) в интерпретации Хокни

Первая возможность используется в MIMD-компьютерах, которые автор называет *конвейерными* (например, процессорные модули в Denelcor HEP). Архитектуры, использующие вторую возможность, в свою очередь опять делятся на два класса:

MIMD-компьютеры, в которых возможна прямая связь каждой пары процессоров, которая реализуется с помощью переключателя;

MIMD-компьютеры, в которых прямая связь каждого процессора возможна только с ближайшими соседями по сети, а взаимодействие удаленных процессоров поддерживается специальной системой маршрутизации через процессоры-посредники.

Далее, среди MIMD-машин с переключателем Хокни выделяет те, в которых вся память распределена среди процессоров как их локальная память (например, PASM, PRINGLE). В этом случае общение самих процессоров реализуется с помощью очень сложного переключателя, составляющего значительную часть компьютера. Такие машины носят название MIMD-машин *с распределенной памятью*. Если память — это разделяемый ресурс, доступный всем процессорам через переключатель, то такие MIMD-машины являются системами *с общей памятью* (CRAY X-MP, BBN Butterfly). В соответствии с типом переключателей можно проводить классификацию и далее: простой переключатель, многокаскадный переключатель, общая шина.

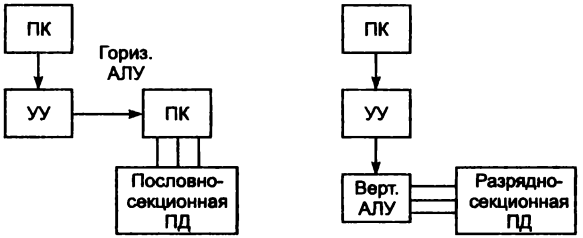
Многие современные вычислительные системы имеют как общую разделяемую память, так и распределенную локальную. Такие системы являются *гибридными* MIMD с переключателем.

При рассмотрении MIMD-машин с *сетевой структурой* считается, что все они имеют распределенную память, а дальнейшая классификация проводится в соответствии с топологией сети: звездообразная сеть (ICAP), регулярные решетки разной размерности (Intel Paragon, CRAY T3D), гиперкубы (NCube, Intel iPCS), сети с иерархической структурой, такой, как деревья, пирамиды, кластеры (Cm, CEDAR) и, наконец, сети, изменяющие свою конфигурацию.

Заметим, что если архитектура компьютера спроектирована с использованием нескольких сетей с различной топологией, то по аналогии с гибридными MIMD с переключателями их стоит назвать *гибридными сетевыми* MIMD, а использующие идеи разных классов — просто *гибридными* MIMD. Типичным представителем последней группы, в частности, является компьютер Connection Machine 2, имеющий на внешнем уровне топологию гиперкуба, каждый узел которого является кластером.

Классификация Шора. Классификация Дж. Шора, появившаяся в начале 70-х гг., интересна тем, что представляет собой попытку выделения типичных способов компоновки вычислительных систем на основе фиксированного числа базисных блоков: устройства управления, арифметико-логического устройства, памяти команд и памяти данных. Дополнительно предполагается, что выборка из памяти данных может осуществляться словами, т. е. выбираются все разряды одного слова, и/или битовым слоем — по одному разряду из одной и той же позиции каждого слова (иногда эти два способа называют горизонтальной и вертикальной выборками соответственно). Конечно же, при анализе данной классификации надо делать скидку на время ее появления, так как предусмотреть невероятное разнообразие параллельных систем настоящего времени было в принципе невозможно. Итак, согласно классификации Шора все компьютеры разбиваются на шесть классов, которые так и называются: *машина типа I, II* и т. д.

Машина типа I — это вычислительная система, которая содержит устройство управления, арифметико-логическое устройство, память команд и память данных с пословной выборкой (рис. 2.34, *а*). Считывание данных осуществляется выборкой всех разрядов некоторого слова для их параллельной обработки в арифметико-логическом устройстве. Состав АЛУ специально не оговаривается, что допускает наличие нескольких функциональных устройств, быть может, конвейерного типа. По этим сообра-



а *б*

Рис. 2.34. Машины типов I (*а*) и II (*б*) по классификации Шора

жениям в данный класс попадают как классические последовательные машины (IBM 701, PDP-11, VAX 11/780), так и конвейерные скалярные (CDC 7600) и векторно-конвейерные (CRAY-1).

Если в *машине типа I* осуществлять выборку не по словам, а содержимого одного разряда из всех слов, то получим *машину типа II* (рис. 2.34, *б*). Слова в памяти данных по-прежнему располагаются горизонтально, но доступ к ним осуществляется иначе. Если в *машине I* происходит последовательная обработка слов при параллельной обработке разрядов, то в *машине II* — последовательная обработка битовых слоев при параллельной обработке множества слов.

Структура *машины II* лежит в основе ассоциативных компьютеров (например, центральный процессор машины STARAN), причем фактически такие компьютеры имеют не одно арифметико-логическое устройство, а множество сравнительно простых устройств поразрядной обработки. Другим примером служит матричная система ICL DAP, которая может одновременно обрабатывать по одному разряду из 4096 слов.

Если объединить принципы построения *машин I* и *II*, то получим *машину типа III* (рис. 2.35, *а*). Эта машина имеет два арифметико-логических устройства — горизонтальное и вертикальное, и модифицированную память данных, которая обеспечивает доступ как к словам, так и к битовым слоям. Впервые идею построения таких систем в 1960 г. выдвинул У. Шуман, называвший их *ортогональными* (если память представлять как матрицу слов, то доступ к данным осуществляется в направлении, «ортогональном» традиционному — не по словам (строкам), а по битовым слоям (столбцам)). В принципе, как машину STARAN, так и ICL DAP можно запрограммировать на выполнение функций машины III, но поскольку они не имеют отдельных АЛУ для

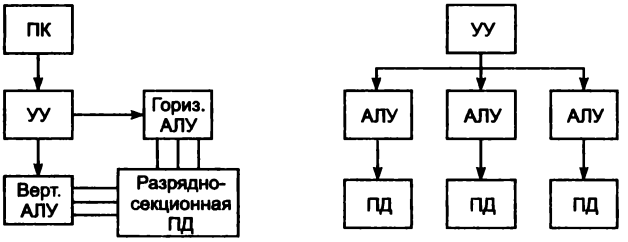


Рис. 2.35. Машины типов III (*а*) и IV (*б*) по классификации Шора

обработки слов и битовых слоев, отнести их к данному классу нельзя. Полноправными представителями машин класса III являются вычислительные системы семейства OMEN-60 фирмы Sanders Associates, построенные в прямом соответствии с концепцией ортогональной машины.

Если в *машине I* увеличить число пар «арифметико-логическое устройство — память данных» (иногда эту пару называют *процессорным элементом*), то получим *машину типа IV* (рис. 2.35, *б*). Единственное устройство управления выдает команду за командой сразу всем процессорным элементам. С одной стороны, отсутствие соединений между процессорными элементами делает дальнейшее наращивание их числа относительно простым, но с другой — сильно ограничивает применимость машин этого класса. Такую структуру имеет вычислительная система PEPE, объединяющая 288 процессорных элементов.

Если ввести непосредственные линейные связи между соседними процессорными элементами *машины IV*, например в виде матричной конфигурации, то получим схему *машины типа V* (рис. 2.36, *а*). Любой процессорный элемент теперь может обращаться к данным как в собственной памяти, так и в памяти непо-

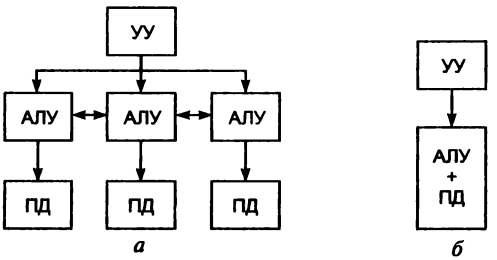


Рис. 2.36. Машины типов V (*а*) и VI (*б*) по классификации Шора

средственных соседей. Подобная структура характерна, например, для классического матричного компьютера ILLIAC IV.

Заметим, что все *машины типа I—V* придерживаются концепции разделения памяти данных и арифметико-логических устройств, предполагая наличие шины данных или какого-либо коммутирующего элемента между ними. *Машина типа VI* (рис. 2.36, *б*), названная *матрицей с функциональной памятью* (или памятью с встроенной логикой), представляет собой другой подход, предусматривающий распределение логики процессора по всему запоминающему устройству. Примерами могут служить как простые ассоциативные запоминающие устройства, так и сложные ассоциативные процессоры.

Примеры некоторых архитектур вычислительных систем

Рассмотрим далее примеры конкретных архитектур, а именно:
- симметричную многопроцессорную;
- асимметричную многопроцессорную (мультипроцессорную);
- массивно-параллельную;
- гибридную с неоднородным доступом к памяти;
- параллельную с векторными процессорами;
- кластерную.

Типичная конфигурация многопроцессорной системы предполагает, что процессоры совместно используют общую ОП, что дает возможность создать масштабируемую многопроцессорную (мультипроцессорную) вычислительную систему, на фоне которой обычная ЭВМ выглядит как однопроцессорная (унипроцессорная).

Авторы предлагают читателю самостоятельно отнести указанные типы систем к тем или иным классам из вышерассмотренных классификаций.

Симметричная мультипроцессорная обработка Symmetric MultiProcessing (SMP). SMP — архитектура суперкомпьютера, в которой группа процессоров работает с общей оперативной памятью (рис. 2.37).

Память является способом передачи сообщений между процессорами, при этом все вычислительные устройства при обращении к ней имеют равные права и одну и ту же адресацию для всех ячеек памяти.

Рис. 2.37. Симметричная мультипроцессорная архитектура

Работой управляет единственная копия операционной системы. Для ускорения обработки каждый процессор может также иметь собственную кэш-память. Задания между процессами распределяются непосредственно при выполнении прикладного процесса. Нагрузка между процессорами динамически выравнивается, а обмен данными между ними происходит с большой скоростью. Достоинство этого подхода состоит в том, что каждый процессор видит всю решаемую задачу в целом. Но поскольку для взаимодействия используется лишь одна шина, то возникают повышенные требования к ее пропускной способности. Соединение посредством шины применяется при небольшом (4—8) числе процессоров.

В подобных системах возникает проблема организации *когерентности многоуровневой иерархической памяти.*

Когерентность кэшей означает, что все процессоры получают одинаковые значения одних и тех же переменных в любой момент времени. Действительно, поскольку кэш-память принадлежит отдельному компьютеру, а не всей многопроцессорной системе в целом, данные, попадающие в кэш одного компьютера, могут быть недоступны другому. Чтобы избежать этого, следует провести синхронизацию информации, хранящейся в кэш-памяти процессоров.

Возможность увеличения числа процессоров в SMP ограничена из-за использования общей памяти. Более того, по той же причине все процессоры должны располагаться в одном корпусе. Между тем, преимуществом SMP является то, что она может работать с прикладными программами, разработанными для однопроцессорных систем.

Данная конфигурация, в которой все ЦП имеют одинаковую скорость доступа к общей ОП, известна так же, как система с однородным доступом к памяти (Uniform Memory Access — UMA) или сильносвязная распределенная система памяти.

Асимметричная мультипроцессорная обработка ASymmetric MultiProcessing (ASMP) — архитектура суперкомпьютера, в которой каждый процессор имеет свою оперативную память.

В ASMP используются три схемы (рис. 2.38). В любом случае процессоры взаимодействуют между собой, передавая друг другу сообщения, т. е. как бы образуя скоростную локальную сеть. Передача сообщений может осуществляться через общую шину (рис. 2.38, *а*, см. также *MPP-архитектуру*) либо благодаря межпроцессорным связям. В последнем случае процессоры связаны либо непосредственно (рис. 2.38, *б*), либо через друг друга (рис. 2.38, *в*). Непосредственные связи используются при небольшом числе процессоров.

Каждый процессор имеет свою оперативную память, расположенную рядом с ним. Благодаря этому, если это необходимо, процессоры могут располагаться в различных, но рядом установленных корпусах. Группа устройств в одном корпусе именуется

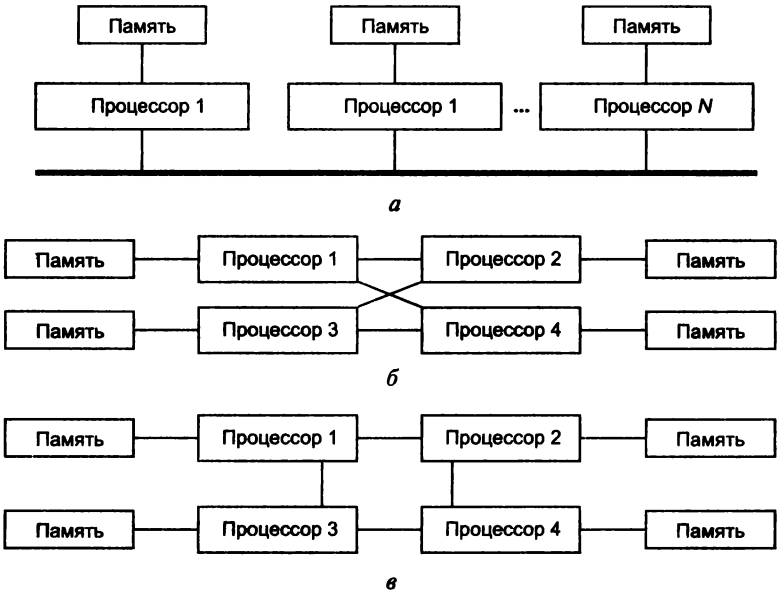


Рис. 2.38. Схемы асимметричной многопроцессорной архитектуры

кластером. Пользователь, обращаясь к кластеру, может работать сразу с группой процессоров. Такое объединение увеличивает скорость обработки данных и расширяет используемую оперативную память. Резко возрастает также отказоустойчивость, ибо кластеры осуществляют резервное дублирование данных. Созданная таким образом система называется кластерной. В этой системе в соответствии с ее структурой может функционировать несколько копий операционной системы и несколько копий прикладной программы, которые работают с одной и той же базой данных (БД), решая одновременно разные задачи.

MPP-архитектура (massive parallel processing) — массивно-параллельная архитектура (рис. 2.38, *а*). В этом случае система строится из отдельных модулей, каждый из которых содержит:

- процессор;
- локальный банк оперативной памяти (ОП);
- два коммуникационных процессора (маршрутизатора, рутера — router): один — для передачи команд, другой — для передачи данных (или сетевой адаптер);
- жесткие диски и/или другие устройства ввода-вывода.

По своей сути, такие модули представляют собой полнофункциональные компьютеры. Доступ к банку ОП из данного модуля имеют только процессоры из этого же модуля. Модули соединяются специальными коммуникационными каналами. Пользователь может определить логический номер процессора, к которому он подключен, и организовать обмен сообщениями с другими процессорами.

Гибридная архитектура (NUMA). Главная особенность гибридной архитектуры NUMA (*nonuniform memory access*) — неоднородный доступ к памяти.

Гибридная архитектура воплощает в себе удобства систем с общей памятью и относительную дешевизну систем с раздельной памятью. Суть этой архитектуры — в методе организации памяти, а именно: память является физически распределенной по различным частям системы, но логически разделяемой, так что пользователь видит единое адресное пространство. Система состоит из однородных базовых модулей (плат), состоящих из небольшого числа процессоров и блока памяти. Модули объединены с помощью высокоскоростного коммутатора. Поддерживается единое адресное пространство, аппаратно поддерживается доступ к удаленной памяти, т. е. к памяти других модулей. При этом доступ к локальной памяти осуществляется в несколько раз

быстрее, чем к удаленной. По существу архитектура NUMA является MPP-архитектурой (массивно-параллельной), где в качестве отдельных вычислительных элементов берутся SMP-узлы.

Пример структурной схемы компьютера с гибридной сетью (рис. 2.39): четыре процессора связываются между собой с помощью кроссбара в рамках одного SMP-узла. Узлы связаны сетью типа «бабочка» (Butterfly).

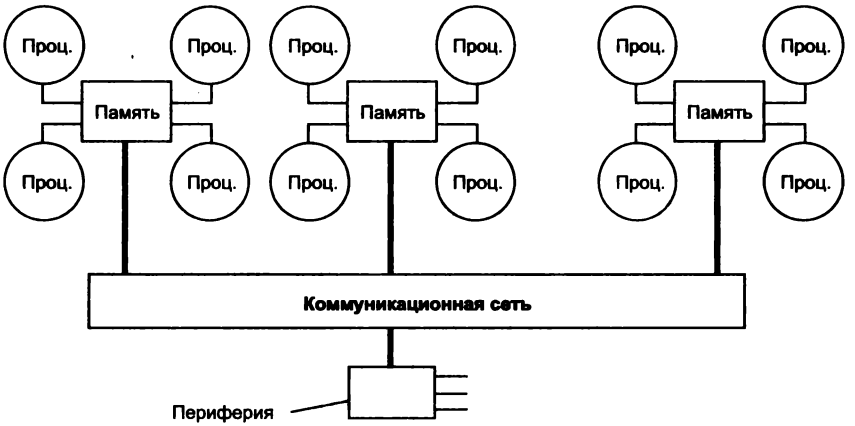


Рис. 2.39. Гибридная архитектура

Известны также гибридные структуры с коммутатором (рис. 2.40). Здесь каждый процессор работает со своей памятью, но модули устройств памяти связаны друг с другом с помощью коммутатора (рис. 2.40, *а*). Коммутаторы могут включаться также между группами процессоров (ПР) и модулей памяти (П). Здесь сообщения между процессорами и памятью передаются через несколько узлов (рис. 2.40, *б*).

Архитектура развивалась такими компаниями, как Burroughs, Convex Computer (в последующем HP), SGI, Sequent и Data General в период 1990-х гг. Разработанные здесь технологии в дальнейшем воплотились в многих Unix-подобных ОС, а также в определенной степени — в Windows NT.

Эффективность как UMA, так и NUMA ограничивается пропускной способностью шины памяти и временами задержки как шины, так и самой памяти. Это значит, что с увеличением числа процессоров после определенного предела производительность перестает возрастать линейно (см. также рис. 2.26). Этот «предел

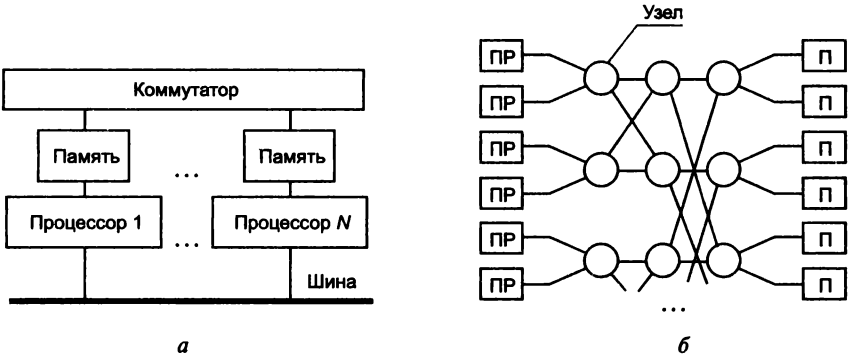


Рис. 2.40. Гибридная архитектура с коммутатором (*а*); многокаскадная коммутация (*б*)

роста» зависит от выполняемых приложений и, как правило, не превосходит 24—68 процессоров.

PVP-архитектура. PVP (Parallel Vector Process) — параллельная архитектура с векторными процессорами. Основным признаком PVP-систем является наличие *векторно-конвейерных процессоров*, в которых предусмотрены команды однотипной обработки векторов независимых данных, эффективно выполняющиеся на конвейерных функциональных устройствах. Как правило, несколько таких процессоров (1—16) работают одновременно с общей памятью (аналогично SMP) в рамках многопроцессорных конфигураций. Несколько таких узлов могут быть объединены с помощью коммутатора (аналогично MPP).

Кластерная архитектура. Кластер, как правило, состоит из двух или более узлов, которые связаны интерфейсами. Распределенные данные, которые доступны кластеру, находятся в накопителях информации. Каждый узел кластера содержит следующие основные компоненты:

- центральный процессор (ПЦ — основное звено обработки информации), обменивающийся данными с оперативной памятью;
- оперативную память (ОП), как и обычно, предназначенную для хранения программ и данных;
- интерфейсы, обеспечивающие связь узлов;
- накопители данных (диски, ленты и пр.).

В любой кластерной архитектуре ЦП используется более или менее одинаковым образом, однако методы конфигурирования

компонентов — узлов, памяти и интерфейсов — существенно различаются. В качестве узлов кластера могут выступать серверы, рабочие станции или обычные персональные компьютеры. Преимущество кластеризации для повышения работоспособности становится очевидным в случае сбоя какого-либо узла; при этом другой узел кластера может взять на себя нагрузку неисправного узла, и пользователи не заметят прерывания в доступе. Возможности масштабируемости кластеров позволяют многократно увеличивать производительность приложений для большего числа пользователей.

Типы кластеров:

Тип I. Машина строится целиком из стандартных деталей, которые продают многие продавцы компьютерных компонентов (низкие цены, простое обслуживание, аппаратные компоненты доступны из различных источников).

Тип II. Система включает эксклюзивные или не широко распространенные детали. Этим можно достичь очень хорошей производительности, однако при более высокой стоимости.

Отметим, что границы между этими типами кластеров до некоторой степени размыты, и часто существующий кластер может иметь такие свойства или функции, которые выходят за рамки перечисленных типов. Более того, при конфигурировании большого кластера, используемого как система общего назначения, приходится выделять блоки, выполняющие все перечисленные функции.

Связь процессоров в кластерной системе. Архитектура кластерной системы (способ соединения процессоров друг с другом) определяет ее производительность в большей степени, чем тип используемых в ней процессоров. Критическим параметром, влияющим на величину производительности такой системы, является расстояние между процессорами. Так, соединив вместе 10 персональных компьютеров, можно получить систему для проведения высокопроизводительных вычислений. Проблема, однако, будет состоять в нахождении наиболее эффективного способа соединения стандартных средств друг с другом, поскольку при увеличении производительности каждого процессора в 10 раз производительность системы в целом в 10 раз не увеличится.

Рассмотрим пример построения симметричной 16-процессорной системы, в которой все процессоры были бы равноправны. Наиболее естественным представляется соединение в виде

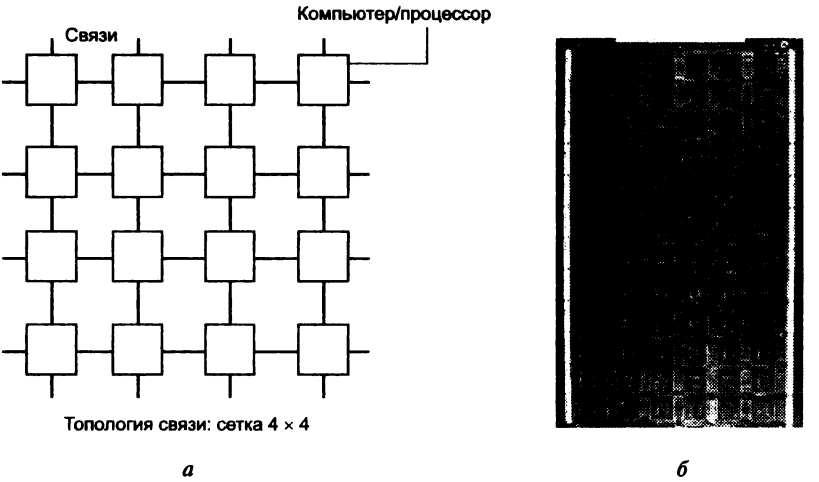


Рис. 2.41. Схема соединения процессоров в виде плоской решетки (*а*), процессорная плата векторного компьютера CRAY YMP (*б*)

плоской решетки, где внешние концы могут использоваться для подсоединения внешних устройств (рис. 2.41).

При таком типе соединения максимальное расстояние между процессорами окажется равным 6 (количество связей между процессорами, отделяющих самый ближний процессор от самого дальнего). Однако, оказывается, что если в системе максимальное расстояние между процессорами больше 4, то такая система не может работать эффективно. Поэтому, при соединении 16 процессоров друг с другом плоская схема является неэффективной. Для получения более компактной конфигурации необходимо использовать фигуры, имеющие максимальный объем при минимальной площади поверхности.

В трехмерном пространстве таким свойством обладает шар. Но поскольку необходимо построить узловую систему, то вместо шара приходится использовать куб (если число процессоров равно 8, рис. 2.42, *а*) или гиперкуб, если число процессоров больше 8. Размерность гиперкуба будет определяться в зависимости от числа процессоров, которые необходимо соединить. Так, для соединения 16 процессоров потребуется четырехмерный гиперкуб (рис. 2.42, *б*). Для его построения следует взять обычный трехмерный куб, сдвинуть в одном направлении и, соединив вершины, получить гиперкуб размерности 4.

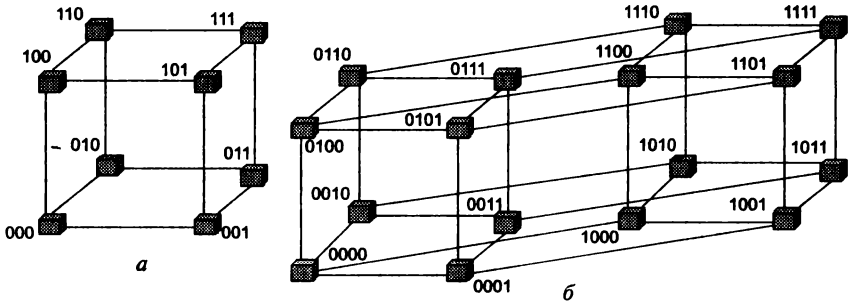


Рис. 2.42. Топологии связи:
а — трехмерный куб; *б* — четырехмерный гиперкуб

Эффективной считается архитектура с топологией «толстого дерева» (fat-tree). Процессоры локализованы в листьях дерева, в то время как внутренние узлы дерева скомпонованы во внутреннюю сеть (рис. 2.43). Поддеревья могут общаться между собой, не затрагивая более высоких уровней сети.

Поскольку способ соединения процессоров друг с другом больше влияет на производительность кластера, чем тип используемых в ней процессоров, то может оказаться более рентабельным создать систему из большего числа дешевых компьютеров, чем из меньшего числа дорогих. В кластерах, как правило, используются операционные системы, стандартные для рабочих станций, чаще всего свободно распространяемые — Linux, FreeBSD.

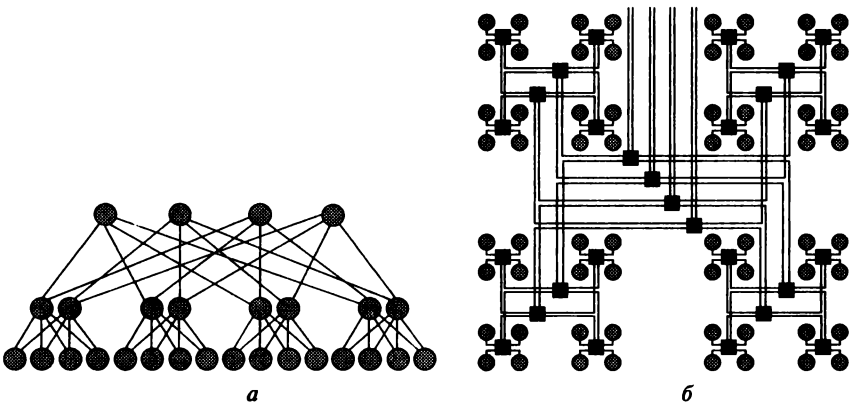
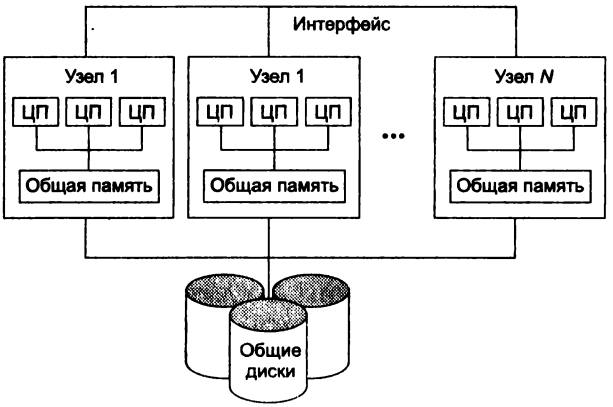


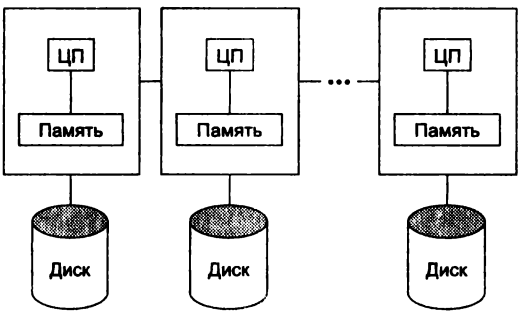
Рис. 2.43. Кластерная архитектура «Fat-tree»:
а — вид «сбоку»; *б* — вид «сверху»

Доступ к внешней памяти (накопителям) в кластерных структурах. Кластерные архитектуры используют различные методы доступа к накопителям информации, каждый из которых использует специфическую схему распределения ресурсов, наилучшую для решаемых задач. Тип доступа к внешней памяти может быть независимым от использования ОП, например, кластер типа SMP может быть снабжен как однородным, так и неоднородным доступом к дисковой памяти.

Однородный доступ к дисковой памяти (Uniform Disk Memory Access — UDMA). При UDMA (рис. 2.44, *а*), затраты на доступ к дискам одинаковы для различных узлов.



а



б

Рис. 2.44. Разновидности архитектур доступа к дисковой памяти: *а* — однородный доступ (UDMA); *б* — неоднородный (NUDMA)

Кластер на рис. 2.44, *а* состоит из нескольких SMP-узлов. Совместно используемая дисковая система такого типа часто применяется при организации соединения по каналам SCSI или Fibre Channel с большим количеством дисков.

Преимущества UDA:

- высокая доступность данных; даже если некоторые узлы выходят из строя, доступ к данным не нарушается;
- хорошая масштабируемость.

Неоднородный доступ к дискам (Non-Uniform DiskMemory Access — NUDMA). В таких системах дисковая память подсоединяется непосредственно к узлам, и для каждого узла такой диск является локальным. Для всех других узлов доступ к «чужому» диску должен быть обслужен программными средствами поддержки виртуальных дисков через каналы связи между узлами. Это означает, что затраты на такой доступ возрастают, как в связи с пониженным приоритетом «чужого» процессора, так и за счет задержек коммутации и перегрузки каналов связи.

Преимущества неоднородного доступа к дискам:

- количество узлов не ограничено возможностями системы коммуникации с дисками;
- общий объем дисковой памяти может быть неограниченно увеличен путем добавления узлов.

Созвездия. Для определенных кластерных конфигураций в последнее время был предложен термин созвездие (constellation).

Рассмотрим рис. 2.45. Каждый узел (node) кластера есть независимый компьютер с одним или более (N) процессоров. Если

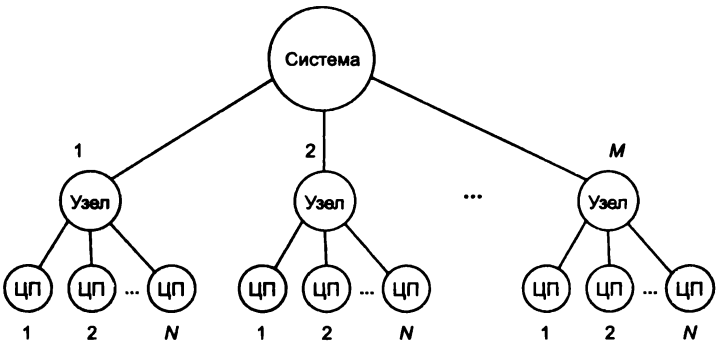


Рис. 2.45. Конфигурации вычислительных систем: кластеры и созвездия

в системе всего M узлов, причем численность узла меньше этого количества ($N < M$), то имеет место *кластерная* конфигурация, в противном случае ($N > M$) имеем дело с *созвездием*.

Система Columbia (кластер из 20 машин SGI Altix, каждая по 512 процессоров) считается типичным примером созвездия.

Транспьютеры и транспьютероподобные системы. Транспьютер — это микроэлектронный прибор, объединяющий на одном кристалле микропроцессор, быструю память, интерфейс внешней памяти и каналы ввода-вывода (порты шин, «линки»), предназначенные для подключения аналогичных приборов, что интегрирует вычислительную и коммутационно-коммуникационную функции. Прибор спроектирован таким образом, чтобы максимально облегчить построение параллельных вычислительных систем. При соединении транспьютерных элементов между собой требуется минимальное число дополнительных интегральных схем. Связь между транспьютерами осуществляется путем непосредственного соединения порта одного прибора с портом другого. Это позволяет создавать сети с различными топологиями с большим числом элементов.

Транспьютер представляет собой микропроцессор, в состав которого входят (рис. 2.46):

- ЦПУ с сокращенным набором команд (RISC);
- 64-разрядный сопроцессор (FPU) плавающей арифметики с высокой пиковой производительностью, работающий параллельно с ЦПУ;
- внутрикристальное ОЗУ;
- 32-разрядная шина памяти;
- четыре последовательные двунаправленные линии связи (*link*), обеспечивающие взаимодействие транспьютера с внешним миром, работающих параллельно;
- таймер;
- генераторы системных управляющих сигналов «*инициализация*», «*анализ*», «*ошибка*», управляющие загрузкой и анализом состояния транспьютера, сигнализирующие об ошибках;
- интерфейс внешних событий (*event*), обеспечивающий асинхронную связь внутреннего процесса и внешнего события.

Транспьютеры размещаются на транспьютерных модулях (TRAM или TPAM) — дочерних платах, содержащих транспьютер, ОЗУ, переключатели для выбора режимов и интерфейс,

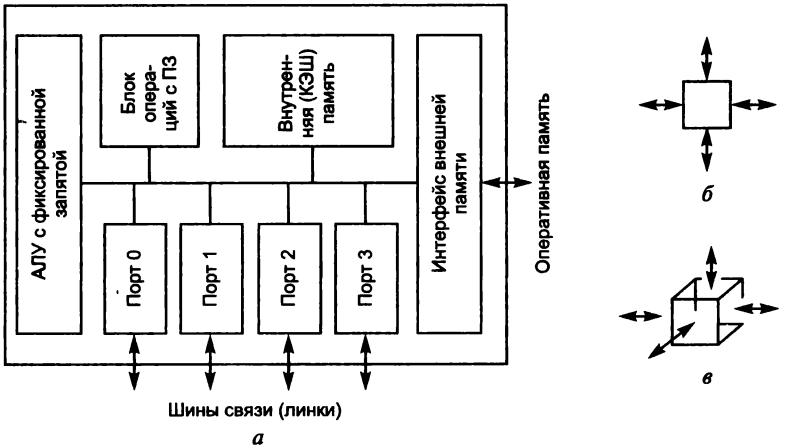


Рис. 2.46. Транспьютер:
а — структура; *б* — возможность соединения в плоскую решетку (рис. 2.41); *в* — возможность соединения в трех-, четырехмерный куб (рис. 2.42)

включающий гнезда/штекеры питания, четыре линии связи, линии внешних событий и системных управляющих сигналов. В зависимости от состава ТРАМ может иметь разные физические размеры, которые стандартизованы и пронумерованы. Так, наименьший по размеру ТРАМ имеет номер 1, следующий — 2 и т. д.

ТРАМы в свою очередь размещаются на объединительных платах, которые либо непосредственно включаются в некоторый компьютер, либо соединенные вместе составляют сетевой компьютер. Известно два типа объединительных плат, подключаемых к компьютеру (вычислительные транспьютерные платы):

- загружаемые по линии связи платы общего назначения, начальная загрузка которых осуществляется программой главного компьютера по линии связи, соединяющей главный компьютер и транспьютер (корневой транспьютер), специально выделенный для взаимодействия с главным компьютером;
- загружаемые из ПЗУ платы, предназначенные для автономных, встроенных систем.

Изначально транспьютеры производила фирма Inmos. В настоящее время ряд зарубежных фирм пошел по пути создания транспьютероподобных микропроцессоров, имеющих гораздо большую вычислительную мощность, чем транспьютер фирмы

Inmos (например, фирма Texas Instruments выпустила сигнальный процессор TMS320C40 с производительностью 50 Mflops).

Возникновение высокопроизводительных параллельных вычислительных систем на базе транспьютеров и транспьютероподобных микропроцессоров в свое время потребовало создания новых эффективных операционных систем.

Кластерные, массивно-параллельные и матричные системы различных производителей

Развитие сетевых технологий привело к появлению недорогих, но эффективных коммуникационных решений. Это и предопределило появление кластерных вычислительных систем, фактически являющихся одним из направлений развития компьютеров с массовым параллелизмом. Классические суперкомпьютеры, использующие специализированные процессоры таких фирм, как, например, Cray, NEC (векторно-параллельные или массивно-параллельные), обычно недешевы, поэтому и стоимость подобных систем не сравнима со стоимостью систем, находящихся в массовом производстве.

Вычислительные системы, создаваемые из массово-выпускаемых компонентов, стали притягательной альтернативой традиционным суперкомпьютерным системам. При выполнении многих прикладных задач такие ВС, даже с небольшим или средним (до 128—256) числом вычислительных модулей, показывают производительность, не уступающую или даже превосходящую производительность традиционных суперкомпьютеров как с распределенной, так и с разделяемой памятью. Наряду с этим, эти ВС обладают рядом преимуществ, среди которых: более низкая стоимость, короткий цикл разработки и возможность оперативно использовать наиболее эффективные вычислительные и коммуникационные компоненты из имеющихся на рынке во время создания системы. Поэтому неудивительно, что ведущие фирмы-разработчики высокопроизводительной техники приступили к созданию кластерных систем.

Отечественные суперкомпьютеры семейства МВС. Создание и использование суперкомпьютеров в мире относятся к факторам стратегического значения и входят в первую десятку приоритетов «ключевых» технологий развитых стран. В настоящее время в США эксплуатируются суперкомпьютеры с производительно-

стью более 10^{15} операций в секунду, что дает возможность ускорения работ в области фундаментальных исследований, решения прикладных задач народнохозяйственного и оборонного комплекса. США, Япония и страны Западной Европы резко ограничивают возможность приобретения Россией мощных вычислительных систем. Кроме того, зарубежные вычислительные системы такого уровня непомерно дороги и их эксплуатационное освоение затруднено.

Проблема создания конкретных суперкомпьютеров требуемого уровня мощности для наиболее крупных российских вычислительных центров решается на основе сбалансированного целенаправленного сочетания закупок новейших комплектующих изделий, создания на этой основе отечественных суперкомпьютерных систем, их интеграции в информационно-вычислительные сети и необходимых усилий в области применения, т. е. в разработке прикладных программ и методов математического моделирования. Эта концепция реализована в мультипроцессорной вычислительной системе МВС-100, которая создана в кооперации научно-исследовательских институтов Российской академии наук и промышленности (головные организации НИИ «Квант» РАСУ и ИПМ РАН). Система построена на основе микропроцессоров с быстродействием порядка 100 миллионов операций в секунду, межпроцессорное взаимодействие осуществляется с помощью транспьютеров. Установки МВС-100 с суммарной производительностью более 50 миллиардов операций в секунду в течение нескольких лет успешно эксплуатируются в вычислительных центрах РАН (в Москве, Екатеринбурге, Новосибирске, Владивостоке) и в отраслевых ВЦ. При этом решены сложные прикладные задачи качественно нового уровня из области аэродинамики для самолетостроения и создания реактивных двигателей, ядерной физики, управления динамическими системами, распознавания изображений при навигации движущихся объектов, сейсмогеологоразведки, нефтедобычи, метеорологии, биоинженерии и др. Показана возможность эффективного распараллеливания вычислений и обработки данных.

В сложившейся кооперации проведены работы по созданию системы нового поколения — МВС-1000 на микропроцессорах Alpha с технологическими нормами 0,35 мкм и быстродействием до 1—2 млрд оп./с. В течение 1998 г. на эксплуатируемых установках этого типа отрабатывалось программное обеспечение и решен ряд новых сложных реальных вычислительных задач.

К настоящему времени введена в действие система с производительностью 200 млрд оп./с для Межведомственного суперкомпьютерного центра (Миннауки, Минобразования, РАН, РФФИ); предполагается дальнейшее наращивание мощности. Освоен режим телекоммуникационного доступа к МВС, в том числе по Internet, с обеспечением требований защиты информации. Предстоят дальнейшие разноплановые работы по техническому и математическому освоению созданных систем, развитию их программного обеспечения. Эти работы входят в соответствующие целевые научно-технические программы федерального и ведомственного уровня.

Массово-параллельные масштабируемые системы МВС предназначены для решения прикладных задач, требующих большого объема вычислений и обработки данных. Суперкомпьютерная установка системы МВС представляет собой мультипроцессорный массив, объединенный с внешней дисковой памятью и устройствами ввода-вывода информации под общим управлением персонального компьютера или рабочей станции.

МВС-1000 — система 3-го поколения, основана на использовании микропроцессоров Alpha 21164 (разработка фирмы DEC-Compaq; выпускается также заводами фирм Intel и Samsung) с производительностью до 1—2 млрд операций в секунду и присоединенной оперативной памятью объемом 0,1—2 Гбайт.

Мультипроцессорный массив системы с блоками вторичного электросилового питания и вентиляцией располагается в стойках размером 550 × 650 × 2200 мм промышленного стандарта; вес заполненной стойки — 220 кг, потребляемая мощность до 4 кВт.

В основном исполнении системы межпроцессорный обмен структурно аналогичен используемому в системе МВС-100 и реализуется в двух модификациях: на базе «транспьютероподобного» связного микропроцессора TMS320C44 (фирма Texas Instruments), имеющего четыре канала с пропускной способностью каждого 20 Мбайт/с либо на базе связного микропроцессора SHARC ADSP 21060 (фирма Analog Devices), имеющего шесть внешних каналов с пропускной способностью каждого 40 Мбайт/с.

Исполнение МВС-1000К отличается использованием для межпроцессорного обмена коммутационной сети MYRINET (фирма Myricom, США) с пропускной способностью канала в дуплексном режиме 2 × 160 Мбайт/с. Кроме того, предусмотрено подключение к каждому процессору памяти на жестком диске с объемом 2—9 Гбайт.

В стандартной стойке располагается до 64 процессоров системы МВС-1000 или до 24 процессоров системы МВС-1000К. Предусмотрены средства системного объединения стоек для установок с большим числом процессоров.

В программном обеспечении МВС используются в числе прочего:

- языки Fortran и C (C++), дополнительные средства описания параллельных процессов;
- программные средства PVM и MPI (общепринятые для систем параллельной обработки);
- средства реализации многопользовательских режимов и удаленного доступа.

Примеры кластерных решений IBM. В начале 2000 г. IBM создала Linux-кластер из установленных в стойке серверов IBMxSeries, интегрировав их с соответствующими сетями, системами управления (аппаратное и программное обеспечение) и необходимыми услугами. После выпуска в 2001 г. кластера 1300 IBM представила кластер 1350 на процессорах Intel Xeon (табл. 2.12).

Таблица 2.12. **Пример конфигурации кластера 1350**

Класс	Число узлов кластера	Скорость процессора, ГГц	Память системы, Гбайт	Внутренняя память, Гбайт	Соединение кластера, Мбит/с
Начальный	8	2,0	0,512	18	10/100 Ethernet
Средний	32	2,4	1	18	10/100 Ethernet
Профессиональный	128	2,8	1	36	Gigabit Ethernet
Высокопроизводительный	64	2,8	1	36	Myrinet-2000

Стандартным вычислительным узлом для кластера 1350 является IBMxSeries 335. Это позволяет одному или двум процессорам Intel Pentium 4 (Xeon) с быстрой динамической памятью и диском размещаться в стандартном корпусе размером «1U». Символ 1U обозначает 1,75 дюйма высоты в стандартном 19-дюймовом корпусе. X335 имеет встроенный сервисный процессор и два слота для соединения с другими компонентами системы.

Головные узлы, узлы управления и узлы запоминающих устройств обеспечивают особые функции для управления кластером (как обеспечение загрузки, управление устройствами, внешний

ввод-вывод и т. д.). Сервер 2U IBM xSeries 345, основанный на процессорах Xeon, в кластере 1350 используется как узел управления и хранения данных и может быть также использован как вычислительный узел. Коммутаторы применяются для межпроцессорного соединения в параллельном программировании и для различных функций управления.

Для параллельного программирования в качестве межпроцессорного соединения обычно используется сеть Myrinet (см. ниже, гл. 4). Пропускная способность канала составляет приблизительно 200 Мбайт/с в каждом направлении со временем задержки 6—8 мкс.

Терминальные серверы обеспечивают удаленный доступ к консолям ОС узлов через последовательную сеть. Дополнительные функциональные возможности добавляются посредством клавиатуры, мыши, монитора.

Коммерческий программный пакет может включать в себя WebSphere, DB2, MySQL и т. д. HPC пакет может включать MPICH, PVM, Maui Scheduler, математические библиотеки, трансляторы, профилировщики и т. д.

Операционная система Linux инсталлирована на каждом узле кластера. Кластер 1350 запускается под Red Hat Linux. В дальнейшем планируется ставить ОС SuSE (4Q02).

Большинство сложившихся систем управления, называемых xCAT, было разработано IBM для сборки кластеров на основе требований заказчика. xCAT поддерживает все требуемые функции, включая функции удаленного контроля. Отметим, что xCAT использует сервисный процессор xSeries и что xCAT не является открытым программным продуктом. Продукт поставляется свободно с кластерным пакетом IBM, включая исходные тексты.

Управление системами кластера для Linux (CSM) — это лицензионный программный продукт IBM. Он обеспечивает функции управления системами, сходными по форме с программами поддержки параллельных систем (Parallel System Support Programs — PSSP) для AIX-систем уровня поддержки на RS/6000 SP. CSM — это стандартный программный продукт для кластера 1350.

CSM для Linux включает технологию, извлеченную из PSSP, и сейчас доступную на AIX для управления кластерами, собранными из серверов xSeries и запускаемых под Linux, серверами IBM pSeries, управляемых AIX, или комбинацией обеих операционных систем.

Другие программные продукты, как взятые из открытого доступа, так и лицензионные, могут быть выбраны и адаптированы к нуждам заказчика и инсталлированы в виде части полной системы всего кластерного решения. Образцы этого ПО включают Portable Batch Scheduler (PBS) и Maui Scheduler, взятые из открытого доступа. Другие образцы включают MPICH для параллельного программирования, математические библиотеки, инструментарий для параллельной отладки и повышения производительности и много других приложений от независимых продавцов.

Примеры кластерных решений HP. Слияние HP и Compaq обеспечило HP прочное положение лидера по продаже Linux-систем, соответствующих лучшим индустриальным стандартам на базе архитектур IA-32 и IA-64. Данная технология дополнена мощной поддержкой разработок ядра Linux на базе семейства Itanium, а также разработок с открытым кодом в целом.

Поддержка ОС Linux со стороны HP охватывает всю линейку серверов HP, основанных на архитектуре Intel (IA-32 и IA-64), включая все серверы промышленного стандарта HP ProLiant, сверхплотную блейд-архитектуру, рабочие станции HP, настольные компьютеры Evo, отдельные портативные компьютеры, серверы ProLiant для применения в качестве межсетевых экранов и даже портативные устройства iPAQ. HP также продолжает поддерживать технологию ОС Linux для архитектуры AlphaServer, разработанную компанией Compaq. Это открыло путь для современных разработок ОС Linux на базе семейства Itanium. HP поддерживает на своих серверах дистрибутивы Red Hat и SuSE, планируя осуществлять поддержку дистрибутивов операционной системы UnitedLinux после ее выпуска. HP предлагает заказчикам возможность предварительно установить любую ОС Linux на выбранные серверы ProLiant и рабочие станции Evo. Услуги по глобальному развертыванию позволяют управлять предварительной установкой операционной системы в любой точке мира.

Программное обеспечение HP может поддерживать большинство современных средств разработки и настройки производительности для кластерных решений на базе системы Linux. При выборе этих средств действуют ограничения, связанные с типами процессоров и межузловых соединений. В число программных продуктов входят:

- компилятор Intel C++ Compiler для Linux;
- компилятор Intel Fortran Compiler для Linux;

- библиотека Intel Math Kernel Library;
- Intel Vtune Performance Analyzer — средство оптимизации программного кода.

Примеры кластерных решений SGI. Компания Silicon Graphics (SGI) была создана в 1981 г. Основным направлением работы компании в течение многих лет было создание высокопроизводительных графических рабочих станций. В настоящее время ее интересы распространяются на рынок высокопроизводительных вычислений как для технических, так и для коммерческих приложений. В частности она концентрирует свои усилия на разработке и внедрении современных технологий визуализации вычислений, трехмерной графики, обработки звука и мультимедиа.

Седьмого января 2003 г. компания SGI представила новое семейство 64-разрядных Linux-серверов и суперкластеров, названных SGI Altix 3000. Система SGI Altix 3000 использует процессоры Intel Itanium 2 и основана на архитектуре глобальной разделяемой памяти SGI Numaflex, которая является реализацией архитектуры неоднородного доступа к памяти (NUMA). NUMAflex появилась в 1996 г. и с тех пор использовалась в известной серии серверов и суперкомпьютеров SGI Origin, основанных на процессорах MIPS и 64-разрядной операционной системе IRIX. Дизайн NUMAflex позволяет помещать процессор, память, систему ввода-вывода, соединительные провода, графическую подсистему в модульные компоненты, иначе называемые *блоками* или *кирпичиками*. Эти кирпичики могут комбинироваться и конфигурироваться с большой гибкостью, чтобы удовлетворять потребности клиента в ресурсах и рабочей нагрузке. Используя этот дизайн третьего поколения, компания SGI смогла создать систему SGI Altix 3000 на основе традиционных блоков ввода-вывода (IX- и PX-блоки), хранения данных (D-блоки) и соединительных компонентов (маршрутизирующие блоки/R-блоки). Основным отличием этой новой системы является процессорный блок (C-блок), который содержит процессоры Itanium 2.

Ключевой особенностью системы является использование каскадируемых коммутаторов в маршрутизирующих элементах. Каскадируемые коммутаторы обеспечивают системе малые времена задержки или замедление доступа к памяти, несмотря на модульную конструкцию. Это критично для машин, использующих архитектуру неоднородного доступа к памяти (NUMA). Задержки всегда были проблемой в архитектуре NUMA, так как

память распределяется между узлами, а не сосредоточена в одном месте. Каскадируемые коммутаторы используют каталогизируемую схему памяти для отслеживания данных, находящихся в разных кэшах. В результате меньшие объемы данных пересылаются между частями памяти, что выливается в понижение задержек по сравнению с традиционными системами, основанными на шинах.

В недавних тестах SPECfp_rate_base2000 система SGI Altix 3000 (1 ГГц) показала мировой рекорд производительности в операциях с плавающей точкой для 64-процессорного сервера со значением 862. Наиболее близкий результат для 64-процессорных систем с единым образом операционной системы показал сервер HP Superdome (875 МГц) со значением 267 — меньше трети производительности системы SGI. По сравнению с 32-процессорными системами SGI Altix 3000 показал производительность, в 1,8 раз большую, чем IBM eServer p690 (1,3 ГГц), и в 3,5 раза большую, чем HP Superdome (750 МГц). 32-процессорная система SGI получила 443 очка, IBM eServer p690 — 251, HP Superdome — 128. Результаты 32-процессорного сервера SGI Altix 3000 демонстрируют превосходство на 300 % по критерию цена/производительность по сравнению с IBM eServer p690.

SMP Power Challenge фирмы Silicon Graphics. Разработка процессора R10000 позволила компании перейти к объединению своих серверов Challenge (на базе процессора R4000) и PowerChallenge (на базе процессора R8000) в единую линию изделий. Благодаря повышенной производительности этого процессора на целочисленных операциях и плавающей точке, обе линии продуктов могут быть объединены без потери производительности.

Серверы Silicon Graphics работают под управлением операционной системы IRIX (ОС UNIX реального времени), построенной в соответствии с требованиями стандартов SVID (System V Interface Definition) и XPG4. Она поддерживает возможность работы нескольких машин на одном шлейфе SCSI (multi-hosted SCSI), 4-кратное зеркалирование и 128-кратное расщепление дисковых накопителей. На платформе поддерживаются многие продукты компаний Oracle, Informix и Sybase.

Компьютеры Challenge DM/L/XL ориентированы в первую очередь на коммерческие применения, а Power Challenge L/XL — на работу с плавающей запятой. Системы Challenge DM относятся к среднему классу.

Power Challenge относится к классу симметричных мультипроцессорных ЭВМ (SMP-системы), базирующихся на поколении суперскалярных процессоров MIPS R8000 фирмы Silicon Graphics. Отличительными особенностями этих систем являются:

- масштабируемость суперкомпьютинга;
- использование большой динамической памяти (адресация у Power Challenge до 16 Гбайт — в 2 раза выше, чем у Cray T90/C90/J90);
- 64-разрядная архитектура (в отличие от машин фирм IBM, HP, Sun и Thinking Machines) как у машин Cray и Convex;
- бинарная совместимость со всем семейством компьютеров SGI, включая рабочие станции Indy.

Матричные процессоры. Наиболее распространенными из систем класса SIMD являются матричные системы, которые лучше всего приспособлены для решения задач, характеризующихся параллелизмом независимых объектов или данных. Организация систем подобного типа на первый взгляд достаточно проста. Они имеют общее управляющее устройство, генерирующее поток команд и большое число процессорных элементов, работающих параллельно и обрабатывающих каждая свой поток данных. Таким образом, производительность системы оказывается равной сумме производительностей всех процессорных элементов. Однако на практике, чтобы обеспечить достаточную эффективность системы при решении широкого круга задач, необходимо организовать связи между процессорными элементами с тем, чтобы наиболее полно загрузить их работой. Именно характер связей между процессорными элементами и определяет разные свойства системы.

Solomon. Одним из первых матричных процессоров был Solomon (рис. 2.47). Система Solomon содержит 1024 процессорных элемента, соединенных в виде матрицы 32×32. Каждый процессорный элемент матрицы включает в себя процессор, обеспечивающий выполнение последовательных поразрядных арифметических и логических операций, а также оперативное ЗУ емкостью 16 Кбайт. Длина слова — переменная от 1 до 128 разрядов. Разрядность слов устанавливается программно. По каналам связи от устройства управления передаются команды и общие константы. В процессорном элементе используется так называемая многомодальная логика, которая позволяет каждому процессорному элементу выполнять или не выполнять общую операцию в зависимости от значений обрабатываемых данных. В каж-

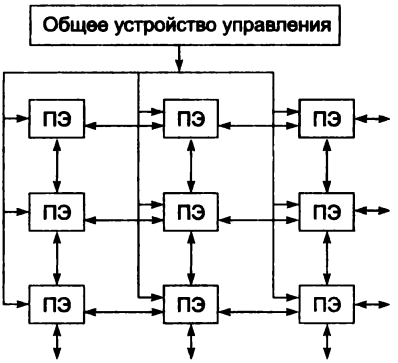


Рис. 2.47. Структура матричной вычислительной системы Solomon

дый момент все активные процессорные элементы выполняют одну и ту же операцию над данными, хранящимися в собственной памяти и имеющими один и тот же адрес.

Идея многомодальности заключается в том, что в каждом процессорном элементе имеется специальный регистр на четыре состояния — регистр моды. Мода (модальность) заносится в этот регистр от устройства управления. При выполнении последовательности команд модальность передается в коде операции и сравнивается с содержимым регистра моды. Если есть совпадения, то операция выполняется. В других случаях процессорный элемент не выполняет операцию, но может в зависимости от кода пересылать свои операнды соседнему процессорному элементу. Такой механизм позволяет выделить строку или столбец процессорных элементов, что очень полезно при операциях над матрицами. Взаимодействуют процессорные элементы с периферийным оборудованием через внешний процессор.

Дальнейшим развитием матричных процессоров стала система Illias-4, разработанная фирмой Burrouth. Первоначально система должна была включать в себя 256 процессорных элементов, разбитых на группы, каждый из которых должен управляться специальным процессором. Однако по различным причинам была создана система, содержащая одну группу процессорных элементов и управляющий процессор. Если в начале предполагалось достичь быстродействия в 1 млрд операций в секунду, то реальная система работала с быстродействием 200 млн операций в секунду. Эта система в течение ряда лет считалась одной из самых высокопроизводительных в мире.

Рис. 2.48. Система ПС-2000 (СССР)

Матричная система ПС-2000 (СССР). В начале 80-х годов в СССР была создана система ПС-2000, которая также являлась матричной (рис. 2.48). Основой этой системы является мультипроцессор ПС-2000, состоящий из решающего поля и устройства управления мультипроцессором. Решающее поле строится из одного, двух, четырех или восьми устройств обработки, в каждом из которых — 8 процессорных элементов. Мультипроцессор из 64 процессорных элементов обеспечивает быстродействие, равное 200 млн операций в секунду на коротких операциях.

В 1972—1975 гг. в Москве в Институте проблем управления АН СССР (ИПУ РАН) была предложена структура и архитектура ПС-2000 — мультипроцессора с одним потоком команд и многими потоками данных. Ее авторам удалось найти оригинальное структурное решение, которое соединило относительную простоту управления одним потоком команд с высокой гибкостью программирования высокопараллельной обработки информации. Найденные в ИПУ РАН структурные решения впервые сориентировали конструкторов на проектирование для таких задач недорогих высокопараллельных компьютеров с высокой производительностью в расчете на единицу стоимости. Предварительные исследования и расчеты подтвердились. Производительность серийных комплексов ПС-2000 достигла 200 млн операций в секунду.

В 1980 г. Госкомиссия приняла опытные образцы и санкционировала серийное производство ЭГВК. Сразу восемь экземпляров ЭГВК ПС-2000, демонстрировавшихся перед комиссией на геофизических задачах (пакет программ в составе СОС-ПС (НПО «Геофизика», Москва)), давали суммарную производительность около миллиарда операций в секунду. Столь высокая производительность проблемно-ориентированных ЭГВК ПС-2000 достигалась на хорошо распараллеливаемых задачах, которые характерны для многих практических применений. При решении таких задач на ЭГВК ПС-2000 достигался рекордный «гражданский» показатель «производительность/стоимость».

С 1981 по 1988 гг. Северодонецким приборостроительным заводом Министерства приборостроения и средств автоматиза-

ции СССР было выпущено около 180 ЭГВК ПС-2000, мультипроцессоров ПС-2000 — 242 шт. Высокопроизводительные и недорогие вычислительные комплексы ПС-2000 работали в различных областях народного хозяйства во всех регионах страны и на специальных объектах.

Отечественное компьютеростроение впервые в мире большим тиражом выпустило высокопроизводительную многопроцессорную вычислительную систему.

В состав ЭГВК ПС-2000 (рис. 2.49) входит собственно мультипроцессор, мониторная подсистема (МПС) и от одной до четырех подсистем внешней памяти (СВП). Мультипроцессор включает в себя 1, 2, 4 или 8 устройств обработки (УО), каждое из которых содержит 8 процессорных элементов (ПЭ), обрабатывающих множество потоков данных по программам, находящимся в общем устройстве управления (ОУУ).

Мониторная подсистема на базе малой ЭВМ СМ-2М взяла на себя функции операционной системы, а также трансляцию, редактирование текстов, счет по вспомогательным программам, управление СВП и средствами отображения. При работе с физическими объектами в реальном времени возможно подключение потоков информации к мультипроцессору как через СВП, так и через специальные высокоскоростные каналы.

Мультипроцессор ПС-2000 ориентирован на высокопроизводительную обработку больших массивов информации по хорошо

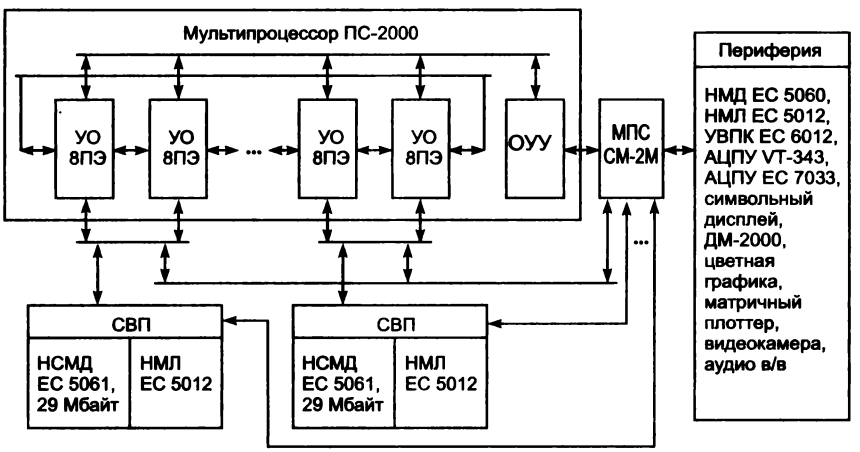


Рис. 2.49. Структура ЭГВК ПС-2000

распараллеливаемым регулярным алгоритмам. Он обеспечивает однозадачный режим работы с одним потоком команд и многими потоками данных (SIMD-архитектура). Особенностью SIMD-архитектуры ПС-2000 является наличие значительных объемов регистровой памяти, в которой и протекают массовые вычисления и межпроцессорные обмены, а также выполняется адресация распределенной оперативной памяти.

Мультипроцессор ПС-2000 (рис. 2.50) состоит из структурированной совокупности однотипных ПЭ и общего устройства управления. Конструктивно восемь ПЭ объединяются в УО. Каждый ПЭ содержит арифметико-логическое устройство с набором регистров общего назначения *S*, память *M*, устройство локальной адресной арифметики *L*, устройство активации ПЭ — *T*, фрагменты регулярного *B* и магистрального каналов. ОУУ содержит арифметико-логическое устройство с набором регистров общего назначения *W*, память данных *H*, адресную арифметику, память микрокоманд *G*.

Устройство *S* за 0,32 мкс выполняет операции с фиксированной запятой над 24-разрядными регистровыми операндами, в

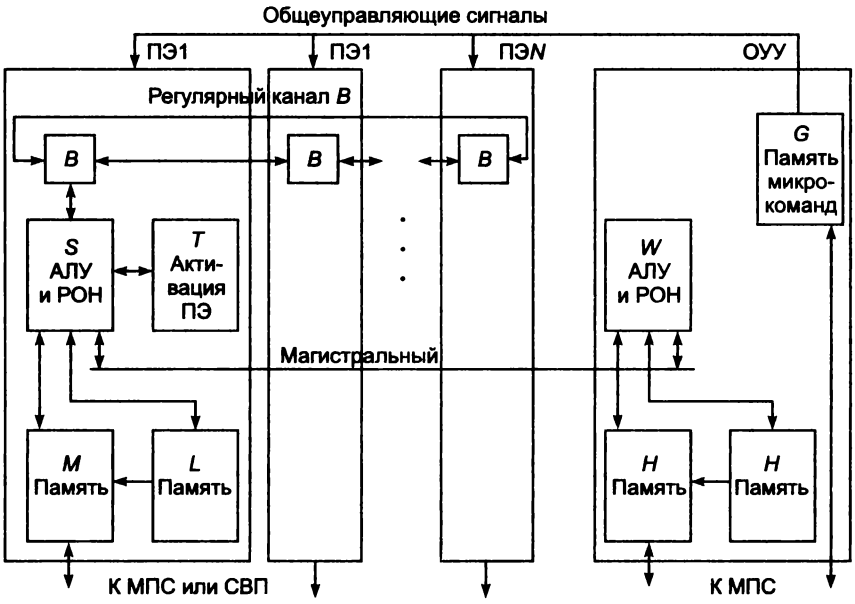


Рис. 2.50. Структура мультипроцессора ПС-2000

нем также имеется аппаратная поддержка, обеспечивающая выполнение операций с плавающей запятой (сложение/вычитание за 0,96 мкс, умножение за 1,6 мкс). Объем памяти M и H — 16 384 24-разрядных слова каждая, операции считывания или записи выполняются за 0,96 мкс. Объем памяти *G* — 16 384 64-разрядных слова. Время выдачи микрокоманды — 0,32 мкс, время выполнения команды ветвления от 1,28 до 1,92 мкс.

Все это позволяет мультипроцессору ПС-2000 с 64 ПЭ работать с эффективной производительностью 200 млн операций в секунду при выполнении расчетов с фиксированной запятой и 50 млн операций в секунду при выполнении одновременно нескольких вычислительных задач, содержащих операции с плавающей запятой. Таким образом, 64-процессорный ПС-2000, имея тактовую частоту 3 МГц, для пользователя работает с частотой 200 МГц.

Таков эффект высокопараллельной хорошо сбалансированной архитектуры. На современной элементной базе архитектура ПС-2000 при тактовой частоте 200 МГц могла бы обеспечить производительность 20 млрд операций в секунду при многократном снижении габаритов и энергопотребления.

Модульное конструктивное построение, неприхотливая элементная база, не требующая специальных условий охлаждения, и система программирования с гибкой системой настроек, позволяющая писать программы, не зависящие от числа ПЭ в ЭГВК ПС-2000, обеспечили высокую живучесть и ремонтопригодность мультипроцессора ПС-2000, что позволило в условиях экспедиций обеспечить работу ЭГВК ПС-2000 более 20 ч в сутки.

Областью широкого использования ЭГВК ПС-2000 стала геофизика, которая объективно нуждалась в компьютерах такого класса. Быстрорастущие стеллажи из внушительных катушек магнитных лент (размером с большую тарелку каждая) с записями данных сейсморазведки, годами безмолвно хранивших нефтяные и газовые секреты, предопределили успех ПС-2000. Уже в 70-е годы сейсмическая разведка настолько успешно «просвечивала» и записывала на ленты километровые глубины недр Родины, что буквально с головой засыпала катушками вычислительные центры. В год удавалось расшифровывать лишь несколько процентов того, что поступало за один сезон разведки.

Для обработки данных сейсмической разведки месторождений нефти и газа во ВНИИ геофизики (Москва) при участии

ИПУ РАН была создана система промышленной обработки геофизической информации СОС-ПС (В. М. Крейсберг). В отрасли успешно эксплуатировалось около 90 экспедиционных геофизических вычислительных комплексов ЭГВК ПС-2000, обеспечивающих углубленную обработку значительной части данных сейсморазведки нефти и газа.

На базе нескольких комплексов ПС-2000 были созданы высокопроизводительные (до 1 млрд операций в секунду) системы обработки гидроакустической и телеметрической информации в реальном масштабе времени. Каждая система содержит три-четыре ЭГВК ПС-2000, соединенных в единый конвейер, а для быстрого ввода и вывода гидроакустической, спутниковой информации для таких систем создавались специализированные высокоскоростные каналы.

Телеметрический вычислительный комплекс центра управления космическими полетами (ЦУП) использовал с 1986 г. вплоть до 1997 г. года систему предварительной обработки телеметрической информации на базе ЭГВК ПС-2000, связанную в единый комплекс с центральной системой обработки на базе многопроцессорного вычислительного комплекса «Эльбрус-2». Высокий параллелизм обработки информации в ПС-2000 позволил реализовать новые алгоритмы обработки телеметрической информации. Первые комплексы ПС-2000 поступили в ЦУП в 1982 г., последние — в 1988 г. Всего было задействовано восемь 32-процессорных комплексов. К одной центральной системе «Эльбрус-2» подключена пара 32-процессорных ЭГВК ПС-2000 для обработки восьми полных потоков телеметрии. С целью дублирования параллельно работали два телеметрических комплекса, а на динамических участках полета космических объектов — три.

Рейтинг суперкомпьютеров

В табл. 2.13 приведены краткие сведения (по состоянию на июль 2012 г.) о десятке наиболее мощных суперкомпьютеров, заимствованные из рейтинга Top500.

В табл. 2.14 приведены также данные о количестве применений тех или иных типов архитектур, коммуникационно-коммутационных (сетевых) средств и типов процессоров в компьютерах, вошедших в рейтинг TOP500.

Таблица 2.13. **Рейтинг суперкомпьютеров (первые 10 из 500, по состоянию на июль 2012 г.)**

Место (ранг)	Компьютер	Производитель	Тип процессоров	Количество процессоров	Интерфейсы (внутренняя коммуникация)	Производительность (Тфлопс)		Расположение системы	Страна, год
						максимальная R_{max}	пиковая R_{peak}		
1	Sequoia — BlueGene/Q	IBM	POWER BQC 16C 1,60 ГГц	1 572 864	Собственная разработка	16 324,75	20 132,66	DOE/NNSA/LLNL	США, 2011
2	K computer	Fujitsu	SPARC64 VIIIfx 2,0	88 128×8 (705024)	Tofu interconnect	10 510,0	11 280,4	RIKEN Advanced Institute for Computational Science (AICS)	Япония, 2011
3	Mira — BlueGene/Q	IBM	POWER BQC 16C 1,60 ГГц	786 432	Собственная разработка	8162,38	10 066,33	DOE/SC/Argonne National Laboratory	США, 2012
4	SuperMUC — iDataPlex DX360M4	IBM	Xeon E5-2680 8C 2,70 ГГц	147 456	Infiniband FDR	2897,00	3185,05	Leibniz Rechenzentrum	ФРГ, 2012
5	Tianhe-1A — NUDT YH MPP	NUDT	Xeon X5670 6C 2,93 ГГц	186 368	NVIDIA 2050	2566,0	4701,0	National Supercomputing Center in Tianjin	КНР, 2010
6	Jaguar — Cray XT5-HE	Cray Inc.	Opteron 6-core 2,6 ГГц	224 162	InfiniBand	1759,0	2331,0	DOE/SC/Oak Ridge National Laboratory	США, 2009
7	Fermi — BlueGene/Q	IBM	POWER BQC 16C 1,60 ГГц	163 840	Собственная разработка	1725,49	2097,15	CINECA	Италия, 2012
8	JuQUEEN — BlueGene/Q	IBM	Power BQC 16C 1,60 ГГц	131 072	Собственная разработка	1380,39	1677,72	Forschungszentrum Juelich (FZJ)	ФРГ, 2012
9	Curie thin nodes — Bullx B510	Bull	Xeon E5-2680 8C 2,700 ГГц	77 184	Infiniband QDR	1271,00	2984,30	CEA/TGCC-GENCI	Франция, 2012
10	Nebulae — Dawning TC3600 Blade System	Dawning	Xeon X5650 6C 2,66 ГГц	120 640	Infiniband QDR, NVIDIA 2050	1271,0	2984,3	National Supercomputing Centre in Shenzhen (NSCS)	КНР, 2010

Таблица 2.14. **Статистика использования архитектурных, сетевых и процессорных решений в 500 наиболее мощных суперкомпьютерах**

Наименование	Число	Процент	Сумма R_{max} (Гфлопс)	Сумма R_{peak} (Гфлопс)	Число процессоров
Архитектура					
Кластерная	407	81,4	66 633 050,71	99 141 890,55	7 169 077
Массивно-параллельная	93	18,6	56 784 736	72 729 253,98	6 258 868
Тип сети					
Gigabit Ethernet	195	39	15250361,5	30 013 028,43	2 724 612
infiniband QDR	105	21	21 247 612,68	31 389 647,46	1 902 852
Infiniband	67	13,4	10 086 122,86	15 071 915,5	1 017 162
Собственная разработка	46	9,2	53 204 689	64 251 548,36	4 781 064
Infiniband FDR	20	4	5 833 998,87	6 776 313,4	312 914
Proprietary	16	3,2	7 280 017	10 993 634	1 366 496
Прочие (NUMAlink, Quadrics, SP Switch, Fat Tree)	49	9,8	8 845 039	2 080 980	1 303 005
Тип процессоров					
Xeon 5600-series (Westmere-EP)	244	48,8	27 721 388,35	51 089 444	3 480 974
Xeon 5500-series (Nehalem-EP)	54	10,8	5 865 550,86	8 601 941,12	723 198
Intel Xeon E5	45	9	13 525 247,64	17 139 955,64	773 986
Opteron 6100-series «Magny-Cours»	33	6,6	6 829 613,9	9 422 817	1 015 940
Power BQC	20	4	34 495 831	42 362 471,4	3 309 568
Xeon 5400-series «Harpertown»	19	3,8	3 123 368,3	4 312 137,61	353 873
POWER7	16	3,2	4 116 318	5 564 367,16	182 736
Opteron 6200 Series «Interlagos»	14	2,8	5 555 924,73	7 400 548	811 546
Opteron Quad Core	11	2,2	1 545 432	2 029 365,8	227 104
POWER6	8	1,6	749 603	984 217,6	52 352
PowerPC 450	7	1,4	2 176 911	2 618 162,4	770 048
PowerPC 440	4	0,8	430 406	539 033,2	192 512
SPARC64 IXfx	3	0,6	1 294 100	1 407 910	95 232
Xeon 5300-series «Clovertown»	3	0,6	334 480	434 227,2	38 320
Xeon 5500-series (Nehalem-EX)	3	0,6	1 224 940	1 463 569	161 408
Opteron Six Core	2	0,4	1 033 900	1 335 240	128 400
Прочие (ShenWei, NEC, PowerPC 970)	13	2,6	13 233 477	14 656 776	1 100 848

Контрольные вопросы

1. Что такое поколения ЭВМ?
2. Охарактеризуйте ЭВМ по областям применения.
3. Назовите основные классы и подклассы вычислительных машин и дайте их сравнительную характеристику.
4. Дайте общую характеристику и определите область использования суперЭВМ и мэйнфреймов.
5. Когда и на основании чего фон Нейман предложил новые принципы создания компьютеров?
6. Что такое процессор и АЛУ?
7. Что такое регистры? Назовите некоторые важные регистры и опишите их функции.
8. Перечислите основные типы архитектуры ЭВМ.
9. Что такое вычислительные системы и каковы их разновидности?
10. Чем многомашинные ВС отличаются от многопроцессорных?
11. Охарактеризуйте одиночный поток команд — одиночный поток данных (ОКОД).
12. Охарактеризуйте одиночный поток команд — множественный поток данных (ОКМД).
13. Охарактеризуйте множественный поток команд — одиночный поток данных (МКОД).
14. Охарактеризуйте множественный поток команд — множественный поток данных (МКМД).
15. На какие классы подразделяются многопроцессорные параллельные ВС?
16. Что такое кластеры и какими преимуществами они обладают?
17. Охарактеризуйте принципы функционирования машин типа *wavefront* и *reduction*.

Глава 3
ПРОЦЕССОРЫ: МИКРОАРХИТЕКТУРЫ И ПРОГРАММИРОВАНИЕ

Процессор и оперативная память образуют центральное устройство (ЦУ) ЭВМ. Процессором является функционально полная совокупность устройств, которая регулирует, управляет и контролирует соответствующий рабочий процесс (в ЭВМ — процесс обработки данных). Сопряжение процессора, памяти и внешних устройств осуществляется через систему интерфейсов, реализующих коммутационно-коммуникационные функции.

3.1. Общее представление о структуре и архитектуре процессоров

Системы команд

Основные команды ЭВМ классифицируются вкратце следующим образом (подробнее см. далее, описание команд для i8086) — по функциям (выполняемым операциям), направлению приема-передачи информации, адресности. Основные признаки классификации и типы команд помещены в табл. 3.1.

Очевидна связь таких параметров ЦУ, как длина адресного пространства, адресность, разрядность. Увеличение разрядности позволяет увеличить адресность команды и длину адреса (т. е. объем памяти, доступной данной команде, например, в 32-разрядной машине, можно адресовать до 4 Гбайт ОП). Увеличение адресности, в свою очередь, приводит к повышению быстродействия обработки (за счет снижения числа требуемых команд).

В трехадресной машине, например, сложение двух чисел требует одной команды (извлечь число по A1, число по A2, сложить и записать результат по A3). В двухадресной машине необходи-

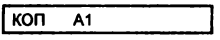
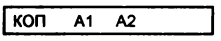
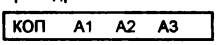
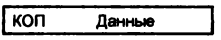
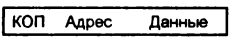
Таблица 3.1. **Классы команд**

Тип команд	Выполняемые действия (или другие признаки)		
Команды пересылки	Пересылка данных между двумя регистрами или между регистром и ячейкой памяти. В некоторых процессорах реализуется пересылка между двумя ячейками памяти, а также групповая пересылка содержимого нескольких регистров в память или их загрузка из памяти		
Команды ввода и вывода	Реализуют пересылку данных из регистра процессора (ОП) во внешнее устройство или прием данных из внешнего устройства в регистр (ОП)		
Команды обработки данных (O1 — первый операнд, O2 — второй)	Короткие операции (один такт)	Логические	Логическое сложение (для каждого бита O1 и O2 осуществляется операция ИЛИ)
			Логическое умножение (для каждого бита O1 и O2 осуществляется операция И)
			Инверсия (в O1 все единицы заменяются на нули и наоборот)
			Сравнение логическое (если O1 = O2, то некий *флаг* или *регистр* устанавливается в «1», иначе — в «0»)
		Арифметические	Сложение операндов
			Вычитание (сложение в обратном коде)
			Сравнение арифметическое (если O1 > O2, или O1 = O2, или O1 < O2, то некий *флаг* или *регистр* устанавливается в «1», иначе — в «0»)
		Команды сдвига	Осуществляют арифметические, логические и циклические сдвиги адресуемых операндов на один или несколько разрядов
	Длинные операции (несколько тактов)	Сложение/вычитание с плавающей запятой	
		Умножение/деление с фиксированной и плавающей запятой	
Операции управления	Безусловный переход (ветвление, branch)	Загружает в СчАК новое содержимое, являющееся адресом следующей выполняемой команды	
	Вызов подпрограммы	Производится путем безусловной передачи управления с сохранением адреса возврата управления	
	Условный переход (conditional branch)	Производит загрузку в PC нового содержимого, если выполняются определенные условия	
	Команды организации программных циклов	Условный переход в зависимости от значения содержимого заданного регистра, который используется как счетчик циклов	
	Команды прерывания	Переход к одной из программ обслуживания исключений и прерываний	

Продолжение табл. 3.1

Тип команд	Выполняемые действия (или другие признаки)	
Операции управления	Команды изменения признаков	Запись-чтение содержимого регистра состояния, в котором хранятся признаки, а также изменение значений отдельных признаков
	Команды управления процессором	Команды останова, отсутствия операции и ряд команд, определяющих режим работы процессора или его отдельных блоков
Тип выборки и пересылок данных	Регистр—регистр	O1 и O2 размещаются в регистрах АЛУ
	Память—регистр (регистр—память)	Один из операндов размещается в ОП
	Память—память	O1 и O2 размещены в ОП
Адресация	Прямая	Операнд выбирается из ячейки/слова памяти, адрес которой содержится в команде
	Регистровая	Операнд выбирается из регистра, номер (имя) которого указано в команде
	Косвенно-регистровая	Операнд выбирается из ячейки/слова памяти, адрес которой содержится в регистре, указанном в команде
	Косвенно-регистровая со смещением	Операнд выбирается из ячейки памяти, адрес которой является суммой содержимого, указанного в команде регистра и заданного в команде смещения (смещение может быть положительным или отрицательным числом)
	Косвенно-регистровая с индексированием и смещением	Операнд выбирается из ячейки памяти, адрес которой является суммой содержимого указанного в команде регистра, индексного регистра и заданного в команде смещения. Иногда имеются специальные индексные регистры, иногда в качестве индексного используется регистр, номер или имя которого указывается в команде. Частным случаем этого способа является индексная адресация, когда адрес образуется суммированием специального индексного регистра и заданного в команде смещения
	Относительная	Операнд выбирается из ячейки памяти, адрес которой является суммой текущего содержимого СчАК и заданного в команде смещения (числа со знаком). Во многих процессорах этот способ адресации используется не для адресации операнда, а для формирования адреса, к которому переходит программа при ветвлении. При этом сформированный таким образом адрес загружается в СчАК, обеспечивая выборку требуемой следующей команды
	Непосредственная	Операнд непосредственно содержится в поступившей команде, размещаясь следом за кодом операции (КОП)

Окончание табл. 3.1

Тип команд	Выполняемые действия (или другие признаки)	
Адресность	Одноадресные 	A1 в зависимости от модификации команды может обозначать либо адрес ячейки (регистра), в которой хранится одно из чисел, участвующих в операции, либо адрес ячейки (регистра), куда следует поместить результат
	Двухадресные 	A1 — это обычно адрес ячейки (регистра), где хранится 1-е из чисел, участвующих в операции, и куда после завершения операции должен быть записан результат; A2 — обычно адрес ячейки (регистра), где хранится второе участвующее в операции число
	Трехадресные 	A2 и A3 — адреса ячеек (регистров), где расположены, соответственно, 1- и 2-е числа, участвующие в операции, A1 — адрес ячейки (регистра), куда следует поместить результат выполнения операции
	Безадресные 	Содержит только код операции, а информация для нее должна быть заранее помещена в определенные регистры машины или содержаться в адресной части
	Комбинированные 	Один или более адресов адресной части предназначены для размещения данных (непосредственный операнд)
Другие признаки	Операции с фиксированной запятой (ФЗ) или точкой (ФТ)	Арифметические и логические операции над числами, обычно занимающими одно машинное слово (пример — системы команд процессоров i8086 — i80486, или x86)
	Операции с плавающей запятой (ПЗ) или точкой (ПТ)	Арифметические операции над числами, представленными в виде «мантисса—порядок» (пример — операции сопроцессора i80487 или x87)
	Десятичная арифметика	Реализация команд, обрабатывающих тетрады бит и выполняющих соответствующие арифметические операции
	Символьная обработка	Команды обработки байт памяти как символов ASCII (сравнение, сортировка и пр.)
	Обработка чисел большой длины	Обработка машинных слов длины, например, 64 или 128 (256) байт
	Векторные операции	Команды SIMD (Single Instruction — Multiple Data) или ОКМД (Одна Команда — Множество Данных) включают MMX, 3DNow!, SSE, SSE2, SSE3, SSSE3
	Команды индексной арифметики	Изменение содержания индексных регистров (в некоторых машинах — ячеек ОП), что используется для обращения к последовательным элементам массива

мы две команды (первая — извлечь число по A1 и поместить в РЧ (или сумматор), вторая — извлечь число по A1, сложить с содержимым РЧ и результат записать по A2). Легко видеть, что одноадресная машина потребует три команды. Поэтому неудивительно, что основная тенденция в развитии ЦУ ЭВМ состоит в увеличении разрядности.

Классы процессоров

В зависимости от набора и порядка выполнения команд процессоры подразделяются на четыре класса, отражающих также хронологию развития ЭВМ.

CISC (Complex Instruction Set Computer) — классическая архитектура процессоров, которая начала свое развитие в 1940-х гг. с появлением первых компьютеров и в которой ЦП использует микропрограммы для выполнения большого набора разноформатных команд с использованием многочисленных способов адресации, для этого требуется наличие сложных электронных цепей для декодирования и исполнения. В течение длительного периода производители компьютеров разрабатывали и воплощали в изделиях все более сложные и полные системы команд.

Типичным примером CISC являются процессоры Intel x86 (в частности, семейство Pentium). Они выполняют более 200 команд разной степени сложности, которые имеют размер от 1 до 15 байт, и обеспечивают более 10 различных способов адресации. Такое многообразие выполняемых команд и способов адресации позволяет программисту реализовать наиболее эффективные алгоритмы решения различных задач. Однако при этом существенно усложняется структура процессора, особенно его устройства управления, что приводит к увеличению размеров и стоимости кристалла, снижению производительности.

В то же время анализ работы процессоров показал, что в течение примерно 80 % времени выполняется лишь 20 % общего набора команд. Поэтому была поставлена задача оптимизации выполнения небольшого по числу, но часто используемых команд. В середине 70-х это привело многих производителей компьютеров к пересмотру своих позиций и к разработке ЦП с ограниченным набором команд.

RISC (Redused Instuction Set Computer) — архитектура отличается использованием ограниченного набора команд фиксиро-

ванного формата. Первый процессор RISC был создан корпорацией IBM в 1979 г. и имел шифр IBM 801.

Современные RISC-процессоры обычно реализуют около 100 команд, имеющих фиксированный формат длиной 4 байта. Также значительно сокращается число используемых способов адресации. Обычно в RISC-процессорах все команды обработки данных выполняются только с регистровой или непосредственной адресацией. При этом для сокращения количества обращений к памяти RISC-процессоры имеют увеличенный объем внутреннего РЗУ — от 32 до нескольких сотен регистров (в CISC-процессорах число регистров общего назначения обычно составляет 8—16). В результате процессор на 20—30 % реже обращается к оперативной памяти, что также повышает скорость обработки данных. Упростилась топология процессора, выполняемого в виде одной интегральной схемы, сократились сроки ее разработки, она стала дешевле.

Обращение к памяти в RISC-процессорах используется только в операциях загрузки данных в РЗУ или пересылки результатов из РЗУ в память. При этом используется небольшое число наиболее простых способов адресации — косвенно-регистровая, индексная и некоторые другие. В результате существенно упрощается структура процессора, сокращаются его размеры и стоимость, значительно повышается производительность. Начиная с процессора Pentium, корпорация Intel начала внедрять элементы RISC-технологий в свои изделия.

В то время, как в процессоре CISC для выполнения одной команды необходимо в большинстве случаев десять тактов и более, процессоры RISC близки к тому, чтобы выполнять по одной команде в каждом такте. Следует также иметь в виду, что благодаря своей простоте процессоры RISC не патентуются. Это также способствует их быстрой разработке и широкому производству.

Процессор MISC работает с минимальным набором длинных команд и характеризуется небольшим набором чаще всего встречающихся команд. Вместе с этим принцип команд VLIW обеспечивает выполнение группы команд за один цикл работы процессора. Порядок выполнения команд распределяется таким образом, чтобы в максимальной степени загрузить маршруты, по которым проходят потоки данных. Таким образом, архитектура MISC объединила вместе суперскалярную (многопоточную) и VLIW концепции. Компоненты процессора просты и работают с высокими скоростями.

VLIW (Very Large Instruction Word) — архитектура, которая появилась относительно недавно (в 1990-х гг.). Ее особенностью является использование очень длинных команд (до 128 бит и более), отдельные поля которых содержат коды, обеспечивающие выполнение различных операций.

Специальный компилятор планирования перед выполнением прикладной программы проводит ее анализ и по множеству ветвей последовательности операций определяет группу команд, которые могут выполняться параллельно. Каждая такая группа образует одну сверхдлинную команду. Это позволяет решать две важные задачи. Во-первых, в течение одного такта выполнять группу коротких («обычных») команд, а во-вторых — упростить структуру процессора. Этим технология VLIW отличается от суперскалярности (здесь отбор групп одновременно выполняемых команд происходит непосредственно в ходе выполнения прикладной программы, а не заранее, из-за этого усложняется структура процессора и замедляется скорость его работы).

Процессоры типа VLIW выпускают фирмы Transmeta, Intel и Hewlett-Packard. К VLIW-типу можно отнести и ожидавшийся в 2002 г. процессор Elbrus 2000 — E2k, объявленный российской компанией «Эльбрус».

3.2. Технологии повышения производительности процессоров и эффективности ЭВМ

Конвейерная обработка команд

Как уже говорилось выше, обработка команды, или цикл процессора, может быть разделена на несколько основных этапов (микрокоманд), которых как минимум пять (выборка, декодирование, чтение исходных данных, выполнение, запись результата).

Каждая операция требует для своего выполнения времени, равного такту генератора процессора (tick of the internal clock). Отметим, что к длинным операциям (плавающая точка) это не имеет отношения — там другая «бухгалтерия». Очевидно, что при тактовой частоте в 100 МГц быстродействие составит 20 миллионов операций в секунду.

Все этапы команды задействуются только один раз и всегда в одном и том же порядке — одна за другой (рис. 3.1, *а*). Это, в частности, означает, что если логическая схема первой микроко-

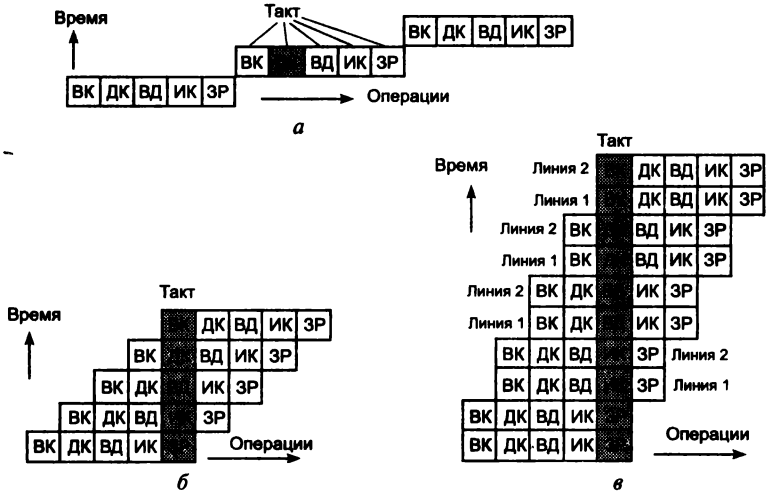


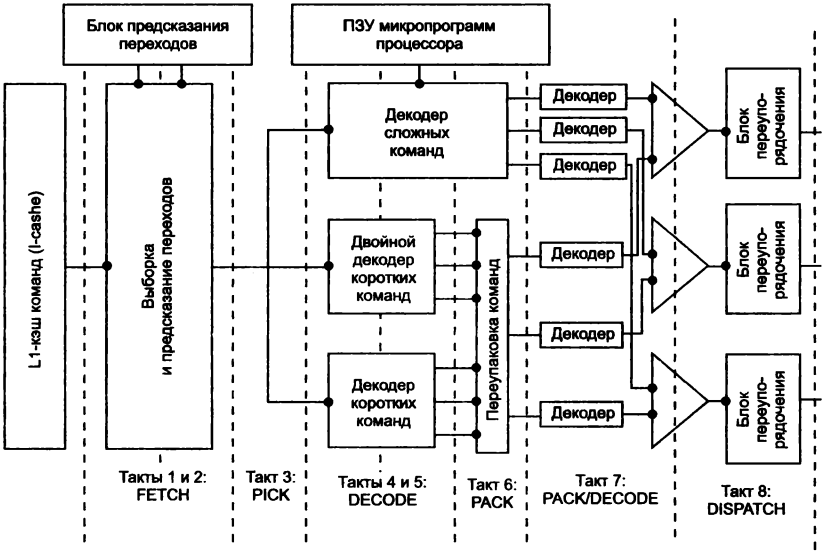
Рис. 3.1. Временны́е диаграммы выполнения команд:
а — без конвейеризации; *б* — пятиступенчатый конвейер; *в* — суперскалярный конвейер; ВК — выборка команды (Fetch); ДК — декодирование команды (Decode); ВД — выборка данных (Load); ИК — исполнение команды (Execute); ЗР — запись результата (Store)

манды выполнила свою работу и передала результаты второй, то для выполнения текущей команды она больше не понадобится, и, следовательно, может приступить к выполнению следующей команды программы.

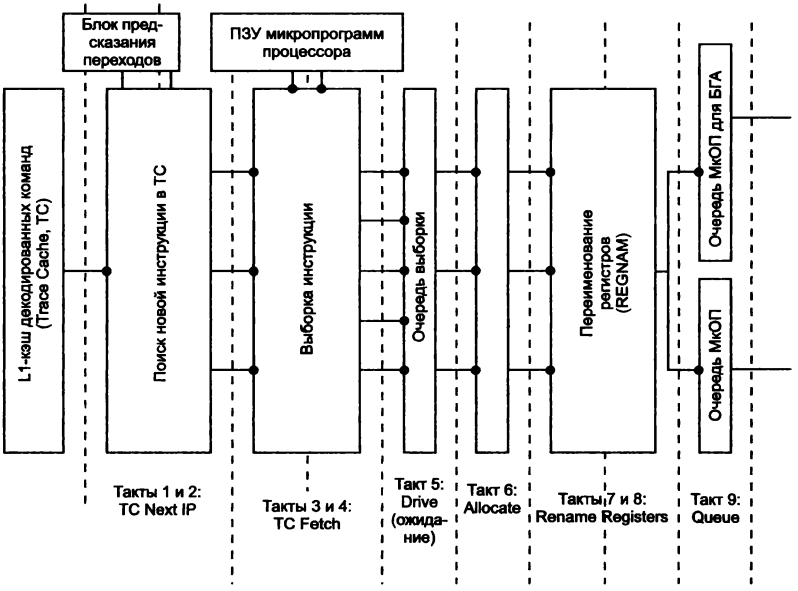
Такая технология обработки команд носит название конвейерной (pipeline) обработки. Каждая часть устройства называется ступенью (стадией) конвейера, а общее число ступеней — длиной линии конвейера.

Конвейеризация осуществляет многопоточную параллельную обработку команд, так что в каждый момент одна из команд считывается, другая декодируется и т. д., и всего в обработке одновременно находится пять команд. Таким образом, на выходе конвейера на каждом такте процессора появляется результат обработки одной команды (одна команда в один такт) — рис. 3.1, *б*.

Приведенный пример процессора (5 микроопераций) является гипотетическим — в реальных ЦП конвейер обработки команд сложнее и включает большее количество ступеней (см. рис. 3.2). Причина увеличения длины конвейера заключается в том, что многие команды являются довольно сложными и не могут быть выполнены за один такт процессора, особенно при

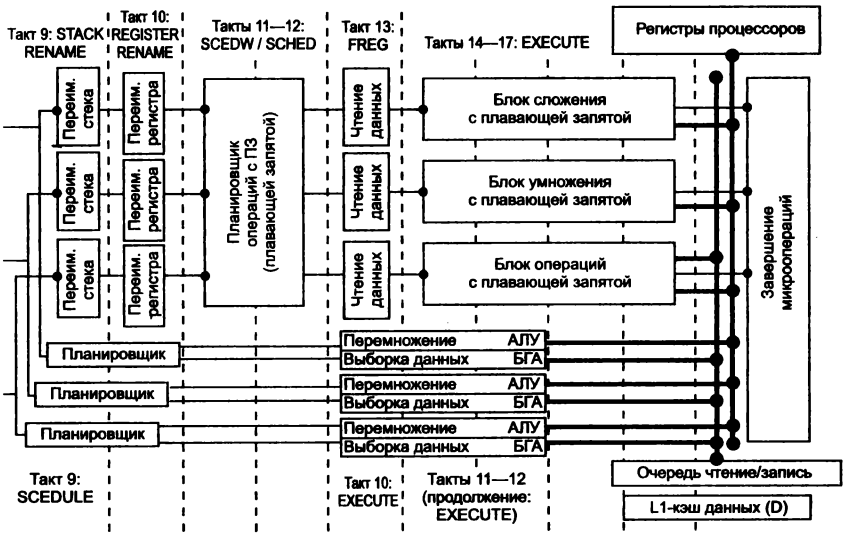


а

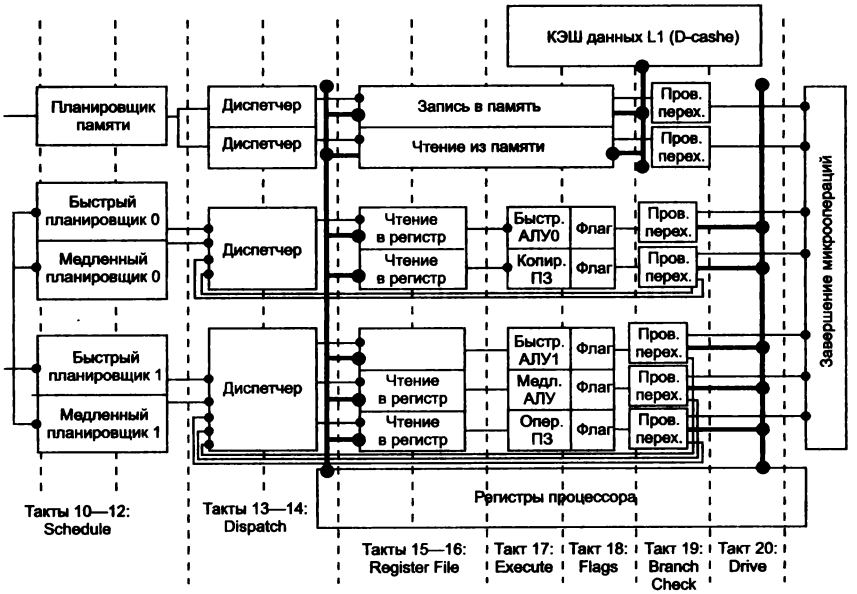


б

Рис. 3.2. Примеры конвейеров AMD K8 (*а*) и Intel Netburst (*б*) (начало)



а



б

Рис. 3.2. (продолжение)

высоких тактовых частотах. Поэтому каждая из упомянутых пяти стадий обработки команд в свою очередь может состоять из нескольких ступеней конвейера.

С ростом числа линий конвейера и увеличением числа ступеней на линии (табл. 3.2) увеличивается пропускная способность процессора при неизменной тактовой частоте. Наоборот, чем больше ступеней насчитывается в конвейере, тем меньшая работа выполняется за такт и тем выше можно поднимать частоту процессора.

Таблица 3.2. **Характеристики конвейеров процессоров Intel и AMD**

Процессор	i80486	Pentium	Pentium Pro	Pentium MMX	Pentium II	Pentium IV	AMD Athlon
Число линий	1	2	3	2	3	3	3+3
Длина линии	5	5	14	6	14	20	17

Суперскаляризация. Процессоры с несколькими линиями конвейера получили название суперскалярных. Pentium — первый суперскалярный процессор Intel. Здесь две линии, что позволяет ему при одинаковых частотах быть вдвое производительней i80486, выполняя сразу две инструкции за такт (рис. 3.1, *в*).

Во многих вычислительных системах, наряду с *конвейером команд*, используются *конвейеры данных*. Сочетание этих двух конвейеров дает возможность достичь очень высокой производительности на определенных классах задач, особенно если используется несколько различных конвейерных процессоров, способных работать одновременно и независимо друг от друга.

На рис. 3.2 приведены примеры конвейеров процессоров архитектуры AMD K8 — 17 ступеней (*а*) и Intel Netburst — 20 ступеней (*б*).

Операции над вещественными числами (с плавающей запятой)

Сопроцессоры. Для расширения вычислительных возможностей центрального процессора — выполнения арифметических операций, вычисления основных математических функций (тригонометрических, показательных, логарифмических) и т. д. — в состав ЭВМ добавляется математический сопроцессор. Применение сопроцессора повышает производительность

вычислений в сотни раз. В разных поколениях процессоров он назывался по-разному — FPU (Floating Point Unit — блок чисел/операций с плавающей точкой — БПЗ) или NPX (Numeric Processor eXtension — числовое расширение процессора).

Для процессоров 386 и ниже сопроцессор был отдельной микросхемой, подключаемой к локальной нише основного процессора. В любом случае сопроцессор исполняет только свои специфические команды, а всю работу по декодированию инструкций и доставке данных осуществляет ЦП.

Блоки операций с плавающей запятой. С программной точки зрения сопроцессор и процессор выглядят как единое целое. В современных (486+) процессорах БПЗ располагается на одном кристалле с центральным процессором.

Начиная с 80486 восемь регистров для ПЗ (именуемых `ST(0)—ST(7)`) встраивают в центральный процессор. Каждый регистр имеет ширину 80 бит и хранит числа в формате стандарта ПЗ расширенной точности (IEEE floating-point standard — см. табл. 1.6).

Эти регистры доступны в стековом порядке. Имена (номера) регистров устанавливаются относительно вершины стека (`ST(0)` — вершина стека, `ST(1)` — следующий регистр ниже вершины стека, `ST(2)` — второй после вершины стека и т. д.). Вводимые данные таким образом всегда сдвигаются «вниз» от вершины стека, а текущая операция совершается с содержимым вершины стека.

Увеличение разрядности систем

В «романтические» 1980-е годы соответствие между типом ЭВМ и ее разрядностью имело простейший вид:

- микроЭВМ — 8 разрядов;
- мини-ЭВМ — 16 разрядов;
- большие ЭВМ — 32 разряда;
- сверхбольшие (супер) ЭВМ — 64 разряда.

В процессе развития микропроцессоров Intel рубежи в 16 и 32 разряда (IA-32) были преодолены довольно быстро, а в районе 2004 г. произошел переход и на 64-разрядные архитектуры в процессорах Intel и AMD.

Преимущества 64-битовой архитектуры микропроцессоров главным образом относятся к памяти. Если взять два идентич-

ных микропроцессора, и один из них будет 32-битовым, а другой — 64-битовым, то последний сможет адресовать намного бóльший объем памяти, чем 32-битовый (2^{64} против 2^{32}). Известны следующие архитектуры на 64 разряда (64-bit architecture).

IA-64. Спецификация IA-64 означает «Архитектура Intel, 64 бита», но связь с IA-32 — только по названию. Архитектура IA-64 не совместима непосредственно с набором команд IA-32. Здесь появляется полностью отличный набор команд, а также используются принципы VLIW вместо выполнения вне естественного порядка. IA-64 — архитектура, используемая линией процессоров Itanium.

Усовершенствования:

- в 16 раз увеличено количество РОН и ПЗ (теперь по 128);
- механизм переименования/ротации регистров, чтобы сохранять значения в регистрах при вызове функций.

AMD64. Набор команд AMD64, первоначально названный x86-64, в значительной степени построен на основе IA-32 и таким образом обеспечивает наследственность семейства x86. При расширении набора команд AMD воспользовалась возможностью, чтобы очистить часть его от ряда «устаревших» команд — наследия «16-разрядных времен».

Усовершенствования:

- в 2 раза увеличено количество РОН и SSE (теперь по 16);
- РОН — теперь действительно регистры общего назначения и ничем больше не ограничены.

Архитектура использована в ЦП Athlon 64, Athlon 64 X2, Athlon 64 FX, Opteron, Turion 64, Turion 64 X2, Sempron («Palermo», «Manila»).

EM64T (Extended Memory 64-bit Technology, или Intel 64) — набор команд (ранее известный как Yamhill), объявленный Intel в феврале 2004 г., в подражание AMD64. EM64T в целом совместим с кодами, написанными для AMD64, хотя и имеет ряд недостатков сравнительно с AMD64.

Intel начала использовать набор EM64T, начиная с ЦП Xeon (ядро Nocona) в конце 2004 г., а затем вышла с ним на рынок настольных ПК в начале 2005 г. (Pentium IV, версия E0).

EMT/Intel 64 используется в ЦП архитектуры Intel NetBurst — Xeon («Nocona»), Celeron D («Prescott» и далее), Pentium 4 («Prescott» и далее), Pentium D, Pentium Extreme Edition и архитектуры Intel Core — Xeon («Woodcrest»), Intel Core 2.

Поскольку AMD64 и EMT64 почти не различаются, для ссылки на них используются нейтральные названия — x86-64, x86_64 (Linux и Apple's Mac OS X), x64 (Microsoft и Sun Microsystems).

Векторная обработка (SIMD-команды)

В классификации Г. Флинна (см. табл. 2.10) имеется рубрика SIMD — поток данных, обрабатываемых одной командой. Процессоры, реализующие такую обработку, именуют потоковыми процессорами. Могут быть определены как однопотоковые (Single-streaming processor — SSP), так и многопотоковые процессоры (Multi-Streaming Processor — MSP).

Типичными представителями класса SIMD считаются матричные процессоры — ILLIAC IV, ICL DAP, Goodyear Aerospace MPP, Connection Machine 1 и т. п. В таких системах единое управляющее устройство контролирует множество процессорных элементов. Каждый процессорный элемент получает от устройства управления в каждый фиксированный момент времени одинаковую команду и выполняет ее над своими локальными данными. В последовательных расширениях системы команд x86, выполненных Intel и AMD, все более полно используются принципы обработки одной командой вектора (потока) данных (см. также Приложение 3).

MMX (MultiMedia eXtension) — архитектура системы команд (57 команд для ФЗ), непосредственно предназначенных для задач мультимедиа, связи и графических приложений, которые часто используют сложные алгоритмы, исполняющие одинаковые операции на большом количестве типов данных (байты, слова и двойные слова). Анализ участков таких программ с большим объемом вычислений показал, что такие приложения имеют следующие общие свойства, определившие выбор системы команд и структуры данных:

- небольшая разрядность целочисленных данных (например, 8-разрядные пиксели для графики или 16-разрядное представление речевых сигналов);
- небольшая длина циклов, но большое число их повторений;
- большой объем вычислений и значительный удельный вес операций умножения и накопления;
- существенный параллелизм операций в программах.

Это и определило новую структуру данных и расширение системы команд. При этом было достигнуто общее повышение производительности на 10—20 %, а в программах обработки мультимедиа — до 60 %.

В процессоре Pentium MMX (1996 г.) появляются 8 новых 64-разрядных «регистров» с именами `MM0`—`MM7` (или же `MMn`), в действительности, эти новые «регистры» были только псевдонимами для регистров стека ПЗ x87 (рис. 3.3). В связи с тем, что каждый из регистров стека содержит 80 бит, старшие 16 бит регистров стека оказываются неиспользованными в MMX (рис. 3.3). Поэтому в режиме MMX они заполняются всеми единицами, что позволяет отличать данные формата с плавающей запятой от MMX-данных.

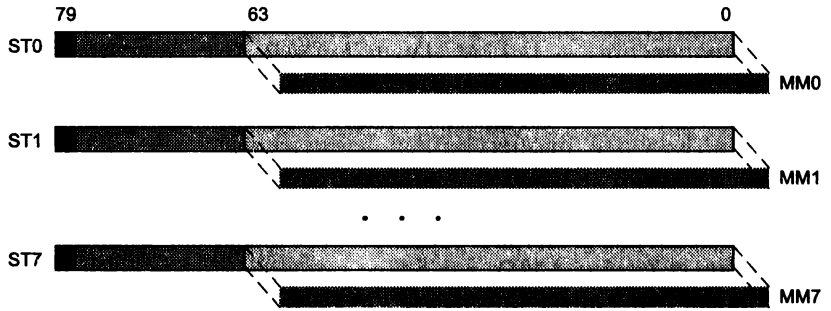


Рис. 3.3. Стек регистров с ПЗ и MMX-регистры

Преимущество совмещения `MMn` с регистрами FPU состоит в том, что одни и те же команды и структуры данных различных операционных систем (при обработке прерываний и вызове подпрограмм) могут использоваться как для сохранения содержания регистров ПЗ, так и регистров 3DNow!.

Каждый из регистров `MMn` предназначен для целых чисел на 64 бита. Однако главным понятием набора команд MMX является концепция упакованных типов данных (см. табл. 1.9, рис. 1.6).

MMX-команды используются в ЦП, начиная с Pentium MMX (AMD K6) и имеют следующий синтаксис:

```
instruction [src, dest]
```

Здесь `instruction` — имя команды, `dest` — выходной, а `src` — входной операнды.

Большинство команд имеют суффикс, который определяет тип данных и используемую арифметику:

- US (unsigned saturation) — арифметика с насыщением, данные без знака;
- S или SS (signed saturation) — арифметика с насыщением, данные со знаком. Если в суффиксе нет ни S, ни SS, используется циклическая арифметика (wraparound);
- B, W, D, Q (см. табл. 1.9) указывают тип данных. Если в суффиксе есть две из этих букв, первая соответствует входному операнду, а вторая — выходному.

Архитектура 3DNow! впервые реализована в процессорах AMD K6-2 (май 1998 г.). Технология 3DNow! включает 21 дополнительную команду, новые типы данных и использует регистры MMn (MM0—MM7) для поддержки высокопроизводительной обработки 3D-графики и звука.

В то время как архитектура MMX предполагает целочисленную арифметику, векторные команды 3DNow! параллельно обрабатывают две пары 32-разрядных вещественных операндов одинарной точности (см. табл. 1.9).

Процессор может выполнять две 3DNow! команды за такт и, следовательно, 4 операции над числами с ПЗ одновременно. Технология 3DNow! предполагает наличие в ЦП для выполнения векторных MMX и 3DNow! операций нескольких устройств (пара умножителей, сумматоров и т. д.). Все команды 3DNow! имеют длительность исполнения 2 такта и полностью конвейеризированы.

SSE (или SIMD-FP) — система команд Streaming SIMD Extensions — SIMD-расширение, предложенное Intel в 1999 г. в Pentium III (ядро Katmai), отсюда вариант названия — KNI (Katmai New Instructions). Это 70 новых команд, в том числе:

- 50 команд предназначаются для повышения эффективности операций с ПЗ, с этой целью в ЦП встроены 128-битовые регистры — восемь регистров, названные XMM0—XMM7 (в AMD64 число SSE/XMM регистров было увеличено от 8 до 16). В результате операции с ПЗ могут совершаться за один цикл процессора;
- 12 команд (New Media) дополняют ранее введенные 57 команд MMX для ФЗ;
- оставшиеся 8 команд (New Cacheability) повышают производительность кэш-памяти L1 при работе с мультимедийными данными.

SSE — набор команд, которые обрабатывают только значения с ПЗ, подобно 3DNow!. Поскольку здесь используются более длинные регистры, чем в 3DNow!, SSE может упаковать два числа ПЗ в каждый регистр (см. табл. 1.9). Первоначальная версия SSE была ограничена только числами одинарной точности, подобно 3DNow!.

SSE2 — введенный с Pentium IV набор команд является существенным развитием SSE, оперирует с теми же самыми регистрами и обратно совместим с SSE процессора Pentium III. В расширении SSE2 операции со 128-битовыми регистрами могут выполняться не только как с четверками вещественных чисел двойной точности, но и как с парами вещественных чисел двойной точности, с шестнадцатью однобайтовыми целыми и пр. (табл. 1.9). SSE2 представляет собой симбиоз MMX и SSE и позволяет работать с любыми типами данных, вмещающимися в 128-битовые регистры.

SSE3 — набор команд, также известный как Prescott New Instructions (PNI), является третьей версией команд SSE для IA-32. Intel использует SSE3 с начала 2004 г. в ЦП Pentium IV Prescott. В апреле 2005 г. AMD также включает SSE3 в ЦП Athlon 64 (версия E — ядра Venice и San Diego). SSE3 содержит 13 дополнительных по отношению к SSE2 команд. Самое существенное новшество — «горизонтальная арифметика» (см. рис. 1.6).

Динамическое исполнение (dynamic execution technology)

Динамическое исполнение — технология обработки данных процессором, обеспечивающая более эффективную работу процессора за счет манипулирования данными, а не просто линейного исполнения списка инструкций.

Предсказание ветвлений. С большой точностью (более 90 %) процессор предсказывает, в какой области памяти можно найти следующие инструкции. Это оказывается возможным, поскольку в процессе исполнения инструкции процессор просматривает программу на несколько шагов вперед.

Это обеспечивает значительное повышение производительности. Например, программный цикл, состоящий из пересылки, сравнения, сложения и перехода в 80486 DX, выполняется за шесть тактов синхронизации, а в — Pentium за два (команды пе-

ресылки и сложения, а также сравнения и перехода сочетаются и предсказывается переход).

Внеочередное выполнение (выполнение вне естественного порядка — out-of-order execution). Процессор анализирует поток команд и составляет график исполнения инструкций в оптимальной последовательности, независимо от порядка их следования в тексте программы, просматривая декодированные инструкции и определяя, готовы ли они к непосредственному исполнению или зависят от результата других инструкций. Далее процессор определяет оптимальную последовательность выполнения и исполняет инструкции наиболее эффективным образом.

Переименование (ротация) регистров (register rename). Чтобы избежать пересылок данных между регистрами в соответствующей команде изменяется адрес регистра, содержащего данные, участвующие в следующей операции. Поэтому вместо пересылки данных в регистр-источник осуществляется трактовка регистра с данными как источника.

Выполнение по предположению (спекулятивное — speculative). Процессор выполняет инструкции (до пяти инструкций одновременно) по мере их поступления в оптимизированной последовательности (спекулятивно). Поскольку выполнение инструкций происходит на основе предсказания ветвлений, результаты сохраняются как предположительные («спекулятивные»). На конечном этапе порядок инструкций восстанавливается.

На рис. 3.4 и 3.5 представлены варианты спекулятивного выполнения:

- *предикация (predication)* — одновременное исполнение нескольких ветвей программы вместо предсказания переходов (выполнения наиболее вероятного);
- *опережающее чтение данных (speculative loading)*, т. е. загрузка данных в регистры с опережением, до того, как определилось реальное ветвление программы (переход управления).

Эти возможности осуществляются комбинированно — при компиляции и выполнении программы.

Предикация. Обычный компилятор транслирует оператор ветвления (например, `IF-THEN-ELSE`) в блоки машинного кода, расположенные последовательно в потоке. Обычный процессор, в зависимости от исхода условия, исполняет один из этих базовых блоков, пропуская все другие. Более развитые процессоры пытаются прогнозировать исход операции и предварительно выполняют предсказанный блок. При этом в случае ошибки много

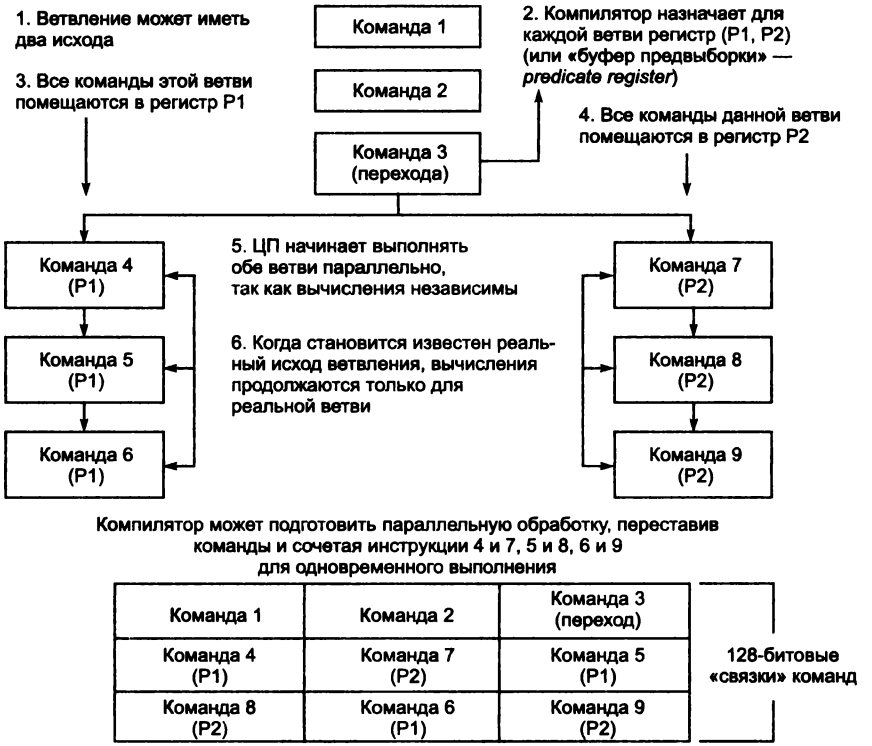


Рис. 3.4. Предикация или одновременное предварительное выполнение всех альтернативных ветвей оператора условного перехода

тактов тратится впустую. Сами блоки зачастую весьма малы — две или три команды, — а ветвления встречаются в коде в среднем каждые шесть команд. Такая структура кода делает крайне сложным его параллельное выполнение.

При использовании предикации компилятор, обнаружив оператор ветвления в исходной программе, анализирует все возможные ветви (блоки) и помечает их метками или *предикатами (predicate)*. После этого он определяет, какие из них могут быть выполнены параллельно (из соседних, независимых ветвей).

В процессе выполнения программы ЦП выбирает команды, которые взаимно независимы и распределяет их на параллельную обработку. Если ЦП обнаруживает оператор ветвления, он не пытается предсказать переход, а начинает выполнять все возможные ветви программы.

Рис. 3.5. Опережающее считывание данных в регистры ЦП из памяти (speculative loading)

Таким образом, могут быть обработаны все ветви программы, но без записи полученного результата. В определенный момент процессор наконец «узнает» о реальном исходе условного оператора, записывает в память результат «правильной ветви» и отменяет остальные результаты.

В то же время, если компилятор не «отметил» ветвление, процессор действует как обычно — пытается предсказать путь ветвления и т. д. Испытания показали, что описанная технология позволяет устранить более половины ветвлений в типичной программе, и, следовательно, уменьшить более чем в 2 раза число возможных ошибок в предсказаниях.

Опережающее чтение (предварительная загрузка данных, чтение по предположению) разделяет загрузку данных в регистры и их реальное использование, избегая ситуации, когда

процессору приходится ожидать прихода данных, чтобы начать их обработку.

Прежде всего, компилятор анализирует программу, определяя команды, которые требуют приема данных из оперативной памяти. Там, где это возможно, он вставляет команды опережающего чтения и парную команду контроля опережающего чтения (speculative check). В то же время компилятор переставляет команды таким образом, чтобы ЦП мог их обрабатывать параллельно.

В процессе работы ЦП встречает команду опережающего чтения и пытается выбрать данные из памяти. Может оказаться, что они еще не готовы (результат работы блока команд, который еще не выполнился). Обычный процессор в этой ситуации выдает сообщение об ошибке, однако система откладывает «сигнал тревоги» до момента прихода процесса в точку «команда проверки опережающего чтения». Если к этому моменту все предшествующие подпроцессы завершены и данные считаны, то обработка продолжается, в противном случае вырабатывается сигнал прерывания.

Возможность располагать команду предварительной загрузки до ветвления очень существенна, так как позволяет загружать данные задолго до момента использования (напомним, что в среднем каждая шестая команда является командой ветвления).

Многократное декодирование команд

В то время как традиционный процессор линейно переводит команды в тактовые микрокоманды и последовательно их выполняет, ЦП с многократным декодированием сначала преобразует коды исходных команд программы в некоторые вторичные псевдокоды (предварительное декодирование, или предекодирование), которые затем более эффективно исполняет ядро процессора. Эти преобразования могут содержать несколько этапов (см. рис. 3.2). В качестве примеров рассмотрим 2- —3-ступенчатые декодеры (рис. 3.6).

Декодирование команд CISC/RISC в VLIW (рис. 3.6, *а*). Эти технологии использованы в мобильных процессорах Crusoe (фирма Transmeta) и некоторых ЦП Intel — архитектуры IA-64 и EPIC (Explicitly Parallel Instruction Computing — вычисления с явной параллельностью инструкций).

В частности, в Crusoe на входе процессора — программы, подготовленные в системе команд Intel x86, однако внутренняя

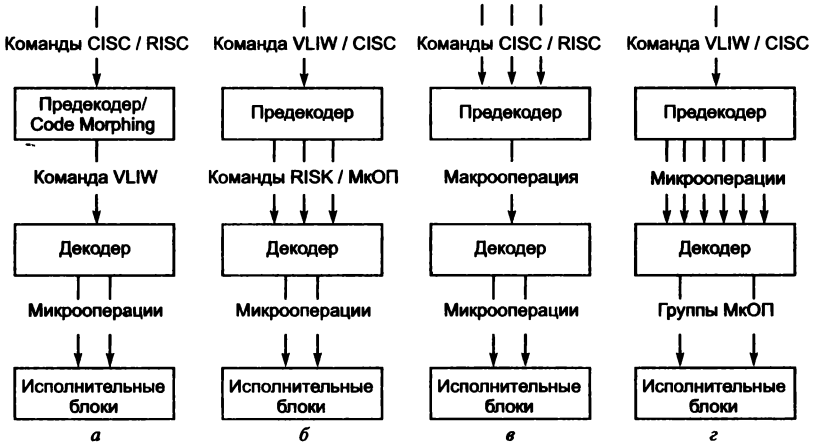


Рис. 3.6. Многократные декодеры и смежные технологии:
а — преобразование CISC/RISC в VLIW; *б* — преобразование VLIW/CICS в RISC; *в* — макрослияние; *г* — микрослияние

система команд VLIW не имеет ничего общего с командами x86 и разработана для быстрого выполнения при малой мощности, используя обычную CMOS-технологию. Окружающий уровень программного называют программным обеспечением модификации кодов (Code Morphing software — CMS, или CM), здесь осуществляется динамический перевод команд x86 в команды VLIW.

Программное обеспечение модификации кодов переводит сразу блоки команд x86, записывая результат в кэш перевода. Этот подход к выполнению кода x86 устраняет миллионы транзисторов, заменяя их программным обеспечением. Процессор Crusoe содержит не более $^1/_4$ числа логических транзисторов, требуемых для достижения подобной производительности только аппаратными средствами, и имеет следующие преимущества:

- аппаратный компонент значительно меньше, быстрее и энергетически эффективнее, чем обычные чипы;
- аппаратные средства полностью отделены от команд x86, давая возможность использовать в проектах наиболее современные и эффективные тенденции в проектировании электроники, не загружая программное обеспечение проблемами совместимости;
- программное обеспечение CM может развиваться отдельно от аппаратных средств, и обновление программной части

процессора может быть выполнено независимо от аппаратных версий чипа;
- упрощение аппаратуры позволило уменьшить габариты процессоров и потребление энергии (эти процессоры иногда называют «холодными»).

Аналогичные приемы используются в процессорах Intel Itanium — здесь при компиляции готовятся пакеты (связки, bundles) команд (по 3 команды в 128-битовом пакете — рис. 3.4). Тем самым, компилятор выполняет в данном случае функции СМ.

Декодирование команд CISC VLIW в RISC. Указанные выше достоинства RISC-архитектуры привели к тому, что во многих современных CISC-процессорах используется RISC-ядро, выполняющее обработку данных. При этом поступающие сложные и разноформатные команды предварительно преобразуются в последовательность простых RISC-операций, быстро выполняемых этим процессорным ядром (рис. 3.6, *б*). Таким образом, работают, например, современные модели процессоров Pentium и K7, которые по внешним показателям относятся к CISC-процессорам. Использование RISC-архитектуры является характерной чертой многих современных процессоров.

Макрослияние (macrofusion). В процессорах предыдущих поколений каждая выбранная команда отдельно декодируется и выполняется. Макрослияние позволяет объединять типичные пары последовательных команд (например, сравнение, сопровождающееся условным переходом) в единственную внутреннюю команду-микрооперацию (МкОП, micro-op) в процессе декодирования (рис. 3.6, *в*). В дальнейшем две команды выполняются как одна МкОП, сокращая полный объем работы процессора.

Микрослияние (micro-op fusion). В современных доминирующих процессорах команды x86 (macro-ops) обычно расчленяются на МкОП прежде, чем передаются на конвейер процессора. Микрослияние группирует и соединяет МкОП, уменьшая их число (рис. 3.6, *г*). Исследования показали, что слияние МкОП вкупе с выполнением команд в измененном порядке может уменьшить число МкОП более чем на 10 %. Данная технология использована в системах Intel Core, а ранее апробировалась в ПЦ мобильных систем Pentium M.

В процессорах AMD K8 конвейер строится на том, что работа с потоком МкОП происходит тройками инструкций (AMD называет их л и н и я м и — line). Конвейер K8 обрабатывает именно линии, а не x86-инструкции или отдельные микрооперации.

Технология Hyper-Threading (HT)

Здесь реализуется разделение времени на аппаратном уровне, разбивая физический процессор на два логических процессора, каждый из которых использует ресурсы чипа — ядро, кэш-память, шины, исполнительное устройство (рис. 3.7). Благодаря HT многопроцессная операционная система использует один процессор как два и выдает одновременно два потока команд. Смысл технологии заключается в том, что в большинстве случаев исполнительные устройства процессора далеки от полной загруженности. От передачи на выполнение вдвое большего потока команд повышается загрузка исполнительных устройств.

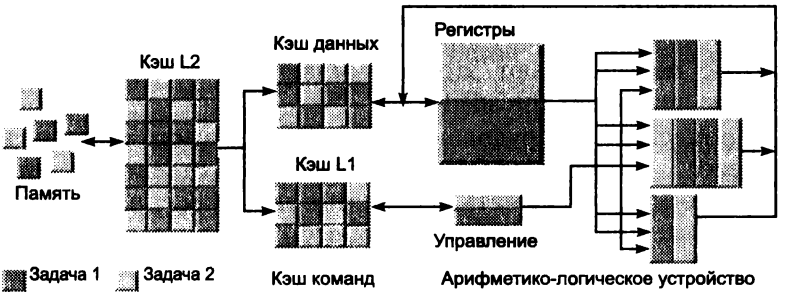


Рис. 3.7. Технология Hyper-Threading (HT)

Специалисты Intel оценивают повышение эффективности в 30 % HT-процессоров при использовании многопрограммных ОС и обычных прикладных программ.

Многоядерные процессоры

В октябре 1989 г., рассматривая будущее «через призму Закона Мура», специалисты Intel в статье «Microprocessors Circa 2000» («Процессоры 2000-х гг.») предсказали, что многоядерные процессоры могут выйти на рынок вскоре после начала столетия. Пятнадцать лет спустя их предсказания оправдались — развитие процессоров в этом направлении стало приоритетной задачей как для Intel, так и для конкурирующей AMD.

Многоядерная архитектура предполагает размещение двух или более основных вычислительных агрегатов в пределах единственного процессора. Этот многоядерный процессор имеет

единственный интерфейс с системной платой, но операционные системы «видят» каждое из его ядер как дискретный логический процессор со всеми связанными ресурсами. Это отличает их от технологии Hyper-Threading (где отдельные процессы выполняются единственным ядром), и существующие ресурсы используются более эффективно.

Разделяя вычислительную нагрузку, выполняемую единственным ядром в традиционных процессорах между многими ядрами, многоядерный процессор может выполнить бóльшую работу в пределах отдельного цикла ЭВМ. Чтобы реализовать это увеличение эффективности, соответствующее программное обеспечение должно поддерживать это распараллеливание. Эти функциональные возможности называют «параллелизмом уровня подпроцесса (нити)» или «threading». Приложения и операционные системы, которые поддерживают это, упоминаются как мульти-подпроцессные или «multi-threaded».

Процессор, оборудованный параллелизмом уровня подпроцесса, может выполнить полностью отдельные «нити» кода. Это может означать или один подпроцесс, выполняемый из приложения, и второй подпроцесс, выполняемый из операционной системы, или параллельные подпроцессы, выполняемые из единственного приложения. Параллелизм уровня подпроцесса приносит особенный выигрыш для многих мультимедийных приложений, потому что многие из их операций способны к выполнению параллельно.

Если при этом используется режим Hyper-Threading, процессоры двойного ядра будут способны обработать четыре программных подпроцесса одновременно более эффективно имеющимися ресурсами, которые иначе, возможно, остались бы незанятыми (рис. 3.8).

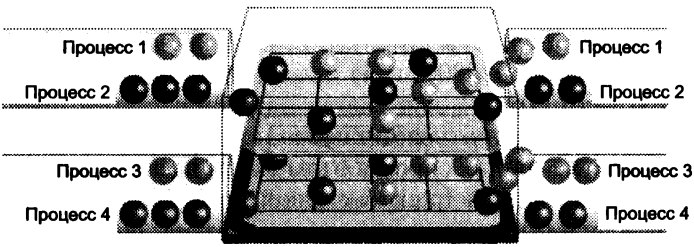


Рис. 3.8. Выполнение процессов Hyper-Threading (HT) на двухъядерном процессоре

Перспективные типы процессоров ЭВМ

Ассоциативные процессоры. Существующие в настоящее время алгоритмы прикладных задач, системное программное обеспечение и аппаратные средства преимущественно ориентированы на традиционную адресную обработку данных. Данные должны быть представлены в виде ограниченного количества форматов (например, массивы, списки, записи, см. табл. 1.13), должна быть явно создана структура связей между элементами данных посредством указателей на адреса элементов памяти, при обработке этих данных должна быть выполнена совокупность операций, обеспечивающих доступ к данным по указателям. Такой подход обусловливает громоздкость операционных систем и систем программирования, а также служит препятствием к созданию вычислительных средств с архитектурой, ориентированной на более эффективное использование параллелизма обработки данных.

Ассоциативный способ обработки данных позволяет преодолеть многие ограничения, присущие адресному доступу к памяти, за счет задания некоторого критерия отбора и проведение требуемых преобразований только над теми данными, которые удовлетворяют этому критерию. Критерием отбора может быть совпадение с любым элементом данных, достаточным для выделения искомых данных из всех данных. Поиск данных может происходить по фрагменту, имеющему большую или меньшую корреляцию с заданным элементом данных.

Исследованы и в разной степени используются несколько подходов, различающихся полнотой реализации модели ассоциативной обработки. Если реализуется только ассоциативная выборка данных с последующим поочередным использованием найденных данных, то говорят об ассоциативной памяти или памяти, адресуемой по содержимому. При достаточно полной реализации всех свойств ассоциативной обработки используется термин «ассоциативный процессор» (рис. 3.9).

Ассоциативные системы относятся к классу SIMD и включают большое число операционных устройств, способных одновременно по командам управляющего устройства вести обработку нескольких потоков данных. В ассоциативных вычислительных системах информация на обработку поступает от ассоциативных запоминающих устройств (АЗУ), характеризующихся тем, что информация в них выбирается не по определенному адресу, а по ее содержанию.

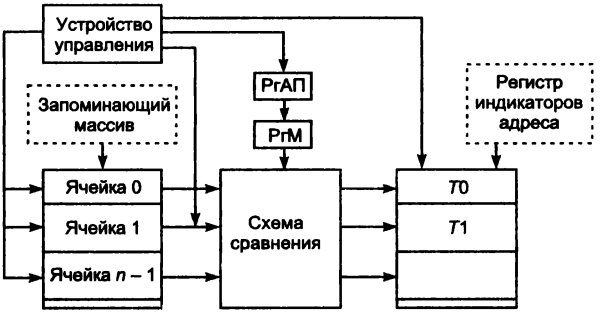


Рис. 3.9. Пример структуры ассоциативного процессора

Клеточные и ДНК-процессоры. В настоящее время в поисках реальной альтернативы полупроводниковым технологиям создания новых вычислительных систем ученые обращают все большее внимание на биотехнологии, или биокомпьютинг, который представляет собой гибрид информационных, молекулярных технологий, а также биохимии. Биокомпьютинг позволяет решать сложные вычислительные задачи, пользуясь методами, принятыми в биохимии и молекулярной биологии, организуя вычисления с помощью живых тканей, клеток, вирусов и биомолекул. Наибольшее распространение получил подход, где в качестве основного элемента (процессора) используются молекулы дезоксирибонуклеиновой кислоты. Центральное место в этом подходе занимает так называемый ДНК-процессор. Кроме ДНК в качестве биопроцессора могут быть использованы также белковые молекулы и биологические мембраны.

ДНК-процессоры. Так же, как и любой другой процессор, ДНК-процессор характеризуется структурой и набором команд. В нашем случае структура процессора — это структура молекулы ДНК.

Ричард Липтон из Принстона первым показал, как, используя ДНК, кодировать двоичные числа и решать проблему удовлетворения логического выражения. Суть ее в том, что, имея некоторое логическое выражение, включающее n логических переменных, нужно найти все комбинации значений переменных, делающих выражение истинным. Задачу можно решить только перебором $2n$ комбинаций. Все эти комбинации легко закодировать с помощью ДНК, а дальше действовать по методике Адлемана. Липтон предложил также способ взлома шифра DES (американский криптографический), трактуемого как своеобразное логическое выражение.

В конце февраля 2002 г. появилось сообщение, что фирма Olympus Optical претендует на первенство в создании коммерческой версии ДНК-компьютера, предназначенного для генетического анализа.

Компьютер, построенный Olympus Optical, имеет молекулярную и электронную составляющие. Первая осуществляет химические реакции между молекулами ДНК, обеспечивает поиск и выделение результата вычислений. Вторая — обрабатывает информацию и анализирует полученные результаты.

Пока до практического применения компьютеров на базе ДНК еще очень далеко. Однако в будущем их смогут использовать не только для вычислений, но и как своеобразные нанофабрики лекарств. Поместив подобное «устройство» в клетку, врачи смогут влиять на ее состояние, исцеляя человека от самых опасных недугов.

Клеточные компьютеры. Клеточные компьютеры представляют собой самоорганизующиеся колонии различных «умных» микроорганизмов, в геном которых удалось включить некую логическую схему, которая могла бы активизироваться в присутствии определенного вещества.

Главным свойством компьютера такого рода является то, что каждая их клетка представляет собой миниатюрную химическую лабораторию. Если биоорганизм запрограммирован, то он просто производит нужные вещества. Достаточно вырастить одну клетку, обладающую заданными качествами, и можно легко и быстро вырастить тысячи клеток с такой же программой.

Основная проблема, с которой сталкиваются создатели клеточных биокомпьютеров, — организация всех клеток в единую работающую систему. На сегодняшний день практические достижения в области клеточных компьютеров напоминают достижения 20-х годов в области ламповых и полупроводниковых компьютеров.

Хотя до практического использования биокомпьютеров все же еще очень далеко, и они вряд ли будут рассчитаны на широкие массы пользователей, предполагается, что они найдут достойное применение в медицине и фармакологии, а также с их помощью станет возможным объединение информационных и биотехнологий.

Нейронные процессоры. Одно из наиболее перспективных направлений разработки принципиально новых архитектур вычислительных систем тесно связано с созданием компьютеров нового поколения на основе принципов обработки информации, заложенных в искусственных нейронных сетях (НС).

Первые практические работы по искусственным нейросетям и нейрокомпьютерам начались еще в 40—50-е годы. Под нейронной сетью обычно понимают совокупность элементарных преобразователей информации, называемых «нейронами», которые определенным образом соединены друг с другом каналами обмена информации «синаптическими связями».

Одним из основных достоинств нейровычислителя является то, что его основу составляют относительно простые, чаще всего однотипные элементы, имитирующие работу нейронов мозга. Каждый нейрон характеризуется своим текущим состоянием по аналогии с нервными клетками головного мозга, которые могут быть возбуждены или заторможены. Он обладает группой синапсов — однонаправленных входных связей, соединенных с выходами других нейронов, а также имеет аксон — выходную связь данного нейрона, с которой сигнал (возбуждения или торможения) поступает на синапсы следующих нейронов. Общий вид нейрона приведен на рис. 3.10, *а*.

Каждый синапс характеризуется величиной синаптической связи или ее весом w_i и по физическому смыслу эквивалентен электрической проводимости. Текущее состояние нейрона определяется как взвешенная сумма его входов:

$$Y = \sum_{i=1}^{n} x_i w_i .$$

Выход нейрона есть функция его состояния: $y = f(s)$, которая называется активационной. Известны различные виды таких

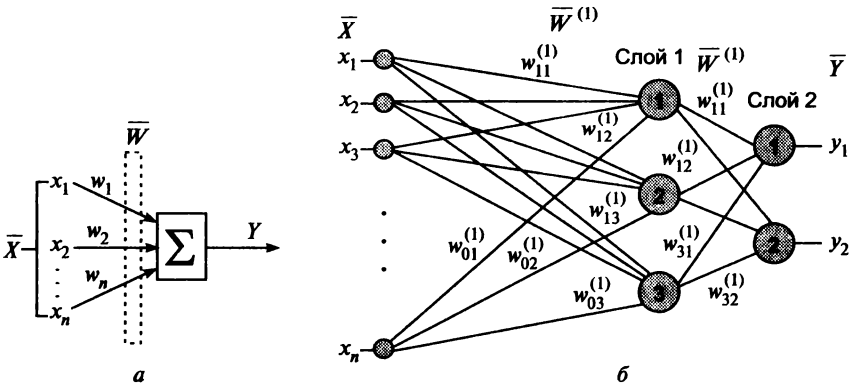


Рис. 3.10. Нейрон (*а*), нейросеть (*б*)

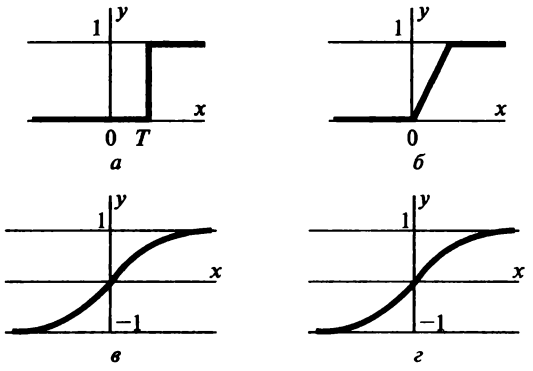


Рис. 3.11. Типовые активационные функции:
а — единичная пороговая функция; *б* — линейный порог (гистерезис); *в* — сигмоид (гиперболический тангенс); *г* — логистический сигмоид

функций, некоторые из которых представлены на рис. 3.11. Одной из наиболее распространенных является нелинейная функция с насыщением, так называемая сигмоидальная (логистическая) функция

$$f(x) = \frac{1}{1 + e^{-ax}}.$$

Состояния нейронов изменяются в процессе функционирования и составляют кратковременную память нейросети. Каждый нейрон вычисляет взвешенную сумму пришедших к нему по синапсам сигналов и производит над ней нелинейное преобразование. При пересылке по синапсам сигналы умножаются на некоторый весовой коэффициент. В распределении весовых коэффициентов заключается информация, хранимая в ассоциативной памяти НС. При обучении и переобучении НС ее весовые коэффициенты изменяются. Однако они остаются постоянными при функционировании нейросети, формируя долговременную память.

НС может состоять из одного слоя, из двух слоев, из трех и большего числа, однако, как правило, для решения практических задач более трех слоев в НС не требуется.

Вообще говоря, под термином «нейрокомпьютер» подразумевается довольно широкий класс вычислителей. Это происходит по той простой причине, что формально нейрокомпьютером можно считать любую аппаратную реализацию нейросетевого алгоритма — от простой модели биологического нейрона до системы распознавания символов или движущихся целей.

Процессоры с многозначной (нечеткой) логикой. Идея построения процессоров с нечеткой логикой (fuzzy logic, «фаззи») основывается на нечеткой математике. Основанные на этой теории различные компьютерные системы, в свою очередь, существенно расширяют область применения нечеткой логики.

Подходы нечеткой математики дают возможность оперировать входными данными, непрерывно меняющимися во времени, и значениями, которые невозможно задать однозначно, такими, например, как результаты статистических опросов. В отличие от традиционной формальной логики, известной со времен Аристотеля и оперирующей точными и четкими понятиями типа истина и ложь, да и нет, ноль и единица, нечеткая логика имеет дело со значениями, лежащими в некотором (непрерывном или дискретном) диапазоне (рис. 3.12).

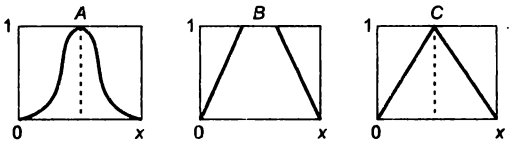


Рис. 3.12. Различные типы функций принадлежности

Функция принадлежности элементов к заданному множеству также представляет собой не жесткий порог «принадлежит — не принадлежит», а плавную сигмоиду, проходящую все значения от нуля до единицы. Теория нечеткой логики позволяет выполнять над такими величинами весь спектр логических операций — объединение, пересечение, отрицание и др. (рис. 3.13).

Задачи с помощью нечеткой логики решаются по следующему принципу (рис. 3.14):

1) численные данные (показания измерительных приборов, результаты анкетирования) *фаззируются* (переводятся в нечеткий формат);

2) обрабатываются по определенным правилам;

3) *дефаззируются* и в виде привычной информации подаются на выход.

В 1986 г. в AT&T Bell Labs создавались процессоры с «прошитой» нечеткой логикой обработки информации. В начале 90-х компания Adaptive Logic (США) выпустила кристалл, сделанный по аналогово-цифровой технологии. Он позволит сократить сроки конструирования многих встроенных систем управления ре-

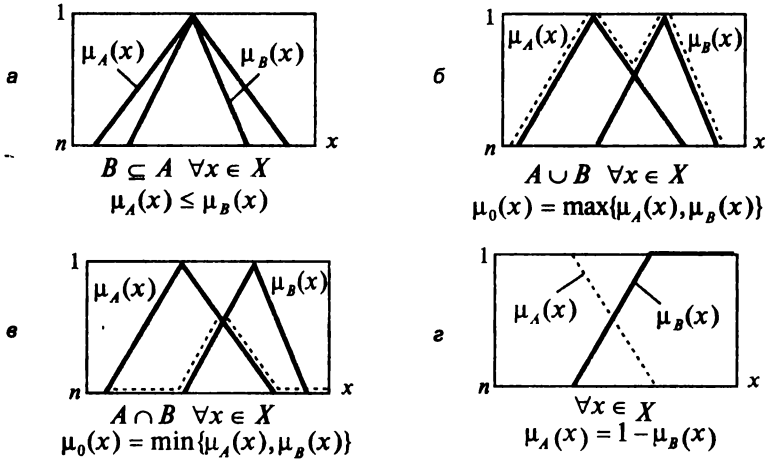


Рис. 3.13. Операции включения (*а*), объединения (*б*), пересечения (*в*) и дополнения (*г*) нечетких множеств

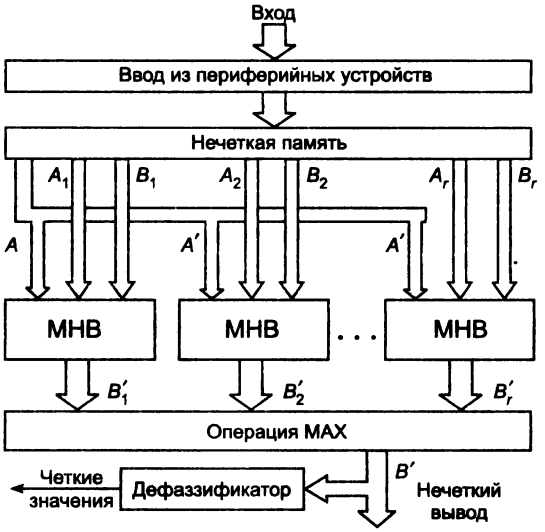


Рис. 3.14. Архитектура нечеткого компьютера: МНВ — механизм нечеткого вывода

ального времени, заменив собой традиционные схемы нечетких микроконтроллеров. Аппаратный процессор нечеткой логики второго поколения принимает аналоговые сигналы, переводит

их в нечеткий формат, затем, применяя соответствующие правила, преобразует результаты в формат обычной логики и далее — в аналоговый сигнал. Все это осуществляется без внешних запоминающих устройств, преобразователей и какого бы ни было программного обеспечения нечеткой логики.

В Европе и США ведутся интенсивные работы по интеграции нечетких команд в ассемблеры промышленных контроллеров встроенных устройств (чипы Motorola 68HC11. 12. 21). Такие аппаратные средства позволяют в несколько раз увеличить скорость выполнения приложений и компактность кода по сравнению с реализацией на обычном ядре. Кроме того, разрабатываются различные варианты нечетких сопроцессоров, которые контактируют с центральным процессором через общую шину данных, концентрируют свои усилия на размывании/уплотнении информации и оптимизации использования правил (продукты Siemens Nixdorf).

Квантовый компьютер — еще одно гипотетическое вычислительное устройство, однако использующее при работе квантово-механические эффекты, такие как квантовый параллелизм. Это позволяет преодолеть некоторые ограничения классических компьютеров.

Идея квантовых вычислений, впервые высказанная Ю. И. Маниным и Р. Фейнманом базируется на том, что квантовая система из L двухуровневых квантовых элементов (кубитов) имеет 2^L линейно независимых состояний, и таким образом, квантовое вычислительное устройство размером L кубит может выполнять параллельно 2^L операций.

Упрощенная схема вычисления на квантовом компьютере выглядит так: берется система кубитов, на которую записывается начальное состояние (исходные данные). Затем состояние системы или ее подсистем изменяется посредством базовых квантовых операций. В конце измеряется значение, и это результат работы компьютера.

Оказывается, что для построения любого вычисления достаточно двух базовых операций. Квантовая система дает результат, только с некоторой вероятностью являющийся правильным. Но за счет небольшого увеличения операций в алгоритме можно сколь угодно приблизить вероятность получения правильного результата к единице.

С помощью базовых квантовых операций можно симулировать работу обычных логических элементов, из которых сделаны клас-

сические компьютеры. Поэтому любую задачу, которая решена сейчас, квантовый компьютер решит и почти за такое же время.

Квантовый компьютер отличается от обычных ЭВМ, которые работают по следующей схеме: n бит памяти хранят состояние и каждый такт времени изменяются процессором. В квантовом случае, система из n кубитов находится в состоянии, являющемся суперпозицией всех базовых состояний, поэтому изменение системы касается *всех* 2^n базовых состояний одновременно. Теоретически новая схема может работать намного быстрее классической. Например, алгоритм Шора позволяет разложить натуральное число n на простые множители за полиномиальное от $\log(n)$ время (для обычного компьютера полиномиальный алгоритм неизвестен).

Другие технологии

Технологии «невыполнимых битов» (No-eXecute bit). Бит «NX» (63-й бит адреса) позволяет операционной системе определить, какие страницы адреса могут содержать исполняемые коды, а какие — нет. Попытка обратиться к NX-адресу как к исполняемой программе вызывает событие «нарушение защиты памяти», подобное попытке обратиться к памяти «только для чтения» или к области размещения ОС. Этим может быть запрещено выполнение программного кода, находящегося в некоторых страницах памяти, таким образом предотвращая вирусные или хакерские атаки. С теоретической точки зрения, здесь осуществляется виртуальное назначение «Гарвардской архитектуры» — разделение памяти для команд и для данных.

Обозначение «NX-bit» используется AMD, Intel использует выражение «XD-bit» (eXecute Disable bit).

Технологии энергосбережения.

On Now PC — способ управления энергопотреблением системы, который заключается в значительном уменьшении потребления электрической энергии, но так, чтобы система в любой момент времени была готова к работе без перезагрузки ОС (например, как готов телевизор, включаемый с помощью удаленного пульта). Система при включении остается способной реагировать на внешние события: нажатие кнопки пользователем, сигнал из сети. Это обеспечивается тем, что небольшая часть системы остается постоянно включенной.

Интеллектуальное управление электропитанием (Intel Intelligent Power Capability) — уменьшение потребления энергии путем включения именно тех логических блоков, которые требуются в данный момент.

Enhanced Intel Speed STep (EIST) — идентичен механизму, осуществленному в процессорах Intel мобильных ПК, который позволяет процессору уменьшать его тактовую частоту, когда не требуется высокая загрузка, таким образом значительно сокращая нагрев центрального процессора и потребление мощности.

Технологии виртуализации, или возможности запуска при помощи специального программного обеспечения (менеджера виртуальных машин) одного или нескольких экземпляров операционных систем, называемых гостевыми, внутри другой ОС, называемой хост-системой (см., напр., [16]). Для ускорения, или во многих случаях, вообще возможности работы гостевых операционных систем разрабатываются средства аппаратной поддержки виртуализации со стороны процессоров:

- VT-x (Intel Virtualization Technology), AMD-V (AMD Virtualization Technology) — технологии аппаратной поддержки виртуализации в процессорах Intel и AMD;
- VT-d (Intel Virtualization Technology for Directed I/O), AMD-vi — технологии виртуализации ввода-вывода, позволяющие гостевым машинам напрямую использовать сетевые адаптеры, графические и дисковые контроллеры и другие аналогичные устройства.

Технология Turbo Boost (от *англ.* turbo boost — турборазгон, турбоускорение) — предложена Intel для автоматического увеличения тактовой частоты процессора свыше номинальной, если при этом не превышаются ограничения мощности, температуры и тока в составе расчетной мощности (TDP). При этом достигается увеличение производительности однопоточных и многопоточных приложений. Аналогичная технология от AMD получила название «Turbo Core».

3.3. Микроархитектуры процессоров

Микроархитектура (логическая структура) микропроцессора, т. е. конфигурация составляющих микропроцессор логических схем и связей между ними, определяется функциональным назначением. Одни и те же функции можно выполнить в микро-

процессорах со структурой, отличающейся набором, количеством и порядком работы логических блоков. Логические блоки типового микропроцессора с развитой архитектурой показаны на рис. 3.15, *а*.

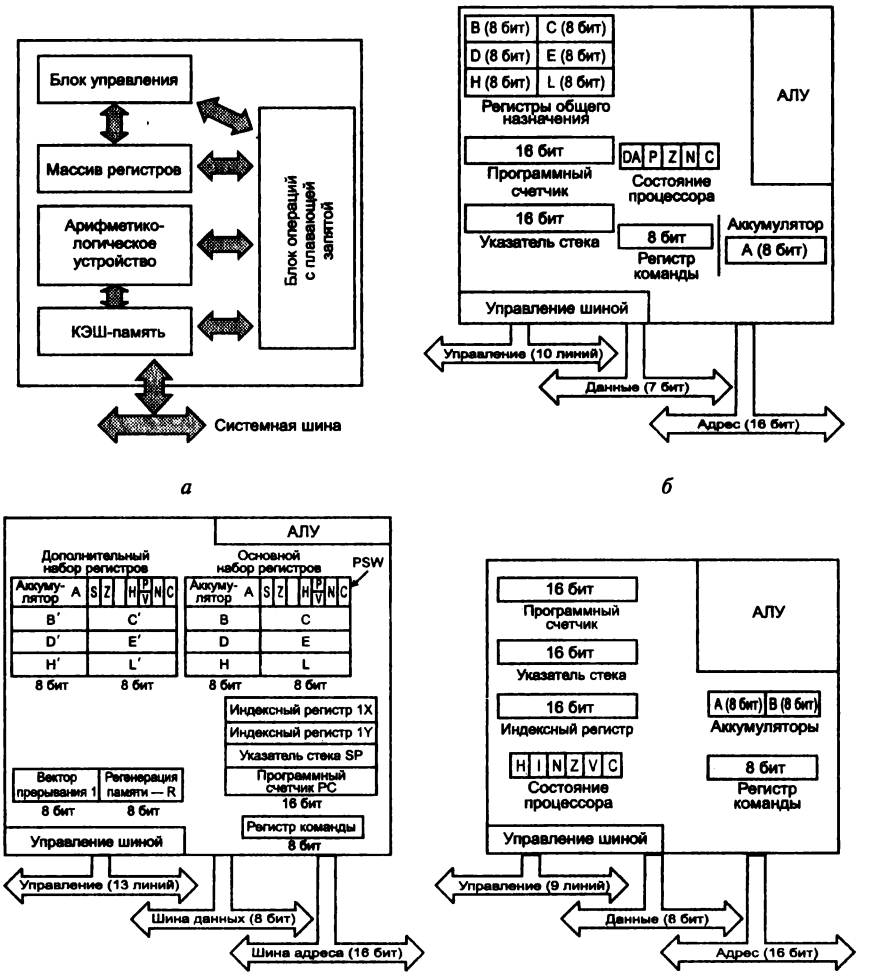


Рис. 3.15. Общая логическая структура микропроцессора (*а*); микропроцессор Intel 8080 (*б*); микропроцессор Z80 фирмы Zilog (*в*); микропроцессор MC6800 фирмы Motorola (*г*)

Первые микропроцессоры

Рассмотрим основные характеристики первых микропроцессоров, ассоциированных с первыми ПК (см. рис. 2.5).

МП Intel 8080 был представлен 1 апреля 1974 г. Благодаря использованию технологии *n*-МОП 6 мкм, на кристалле было размещено 6 тыс. транзисторов. Тактовая частота процессора была доведена до 2 МГц, а длительность цикла команд составила 2 мкс. Объем памяти, адресуемой процессором, — 64 Кбайт. За счет использования 40-выводного корпуса удалось разделить шину адреса (ША) и шину данных (ШД); общее число микросхем, требовавшихся для построения системы в минимальной конфигурации, сократилось до 6 (рис. 3.15, *б*).

В блок РОН (РФ) были введены указатель стека, активно используемый при обработке прерываний, а также два программно недоступных регистра для внутренних пересылок. Блок РОН был реализован на микросхемах статической памяти. Исключение аккумулятора из РФ и введение его в состав АЛУ упростило схему управления внутренней шиной. Новое в архитектуре МП — использование многоуровневой системы прерываний по вектору. Такое техническое решение позволило довести общее число источников прерываний до 256.

В i8080 появился механизм прямого доступа в память (ПДП, DMA) (как ранее в мэйнфреймах IBM System 360 и др.). ПДП открыл возможность для применения в микроЭВМ таких сложных устройств, как накопителей на магнитных дисках и лентах, дисплеев на ЭЛТ, которые превратили микроЭВМ в полноценную вычислительную систему.

Процессор Z80, разработка фирмы Zilog, помимо расширенной системы команд, одного номинала питания и способности исполнять программы, написанные для i8080, имел ряд архитектурных особенностей (рис. 3.15, *в*).

Регистровая архитектура определяется наличием достаточно большого регистрового файла внутри МП. Команды получают возможность обратиться к операндам, расположенным в одной из двух запоминающих сред: оперативной памяти или регистрах. К любому регистру можно обратиться непосредственно, поскольку регистры представлены в виде массива запоми-

нающих элементов — регистрового файла. Типичным является выполнение арифметических операций только в регистре, при этом команда содержит два операнда (оба операнда в регистре или один операнд в регистре, а второй в оперативной памяти).

Микропроцессор MC6800 Motorola также имел ряд существенных особенностей (рис. 3.15, *г*). Прежде всего, кристалл MC6800 требовал для работы одного номинала питания, а система команд оказалась весьма прозрачной для программиста. МП содержал два аккумулятора, и результат операции АЛУ мог быть помещен в любой из них. Но самым ценным качеством структуры МС 6800 было автоматическое сохранение в стеке содержимого всех регистров процессора при обработке прерываний (Z80 требовалось для этого несколько команд `PUSH`). Процедура восстановления РОН из стека тоже выполнялась аппаратно.

Архитектуры процессоров Intel (рис. 3.16, табл. 3.3)

Intel 4004 (1971 г.). Этот прибор послужил отправной точкой абсолютно новому классу полупроводниковых устройств. Чип представлял 4-разрядный процессор с классической архитектурой ЭВМ гарвардского типа, насчитывал 2300 транзисторов и работал на тактовой частоте 750 кГц (длительность цикла команды 10,8 мкс).

Чип имел адресный стек, содержащий счетчик команд и три регистра стека типа LIFO (Last In — First Out — «последним поступил — первым выводится», специальная память с ограниченной емкостью с указанной дисциплиной обслуживания); блок регистров общего назначения — РОН (регистры сверхоперативной памяти или регистровый файл); 4-разрядное параллельное АЛУ; аккумулятор; регистр команд; дешифратор команд; схему управления; схему связи с периферийными устройствами. Эти функциональные узлы объединялись 4-разрядной внутренней шиной данных. Блок РОН состоял из шестнадцати 4-разрядных регистров, которые можно было использовать и как восемь 8-разрядных регистров. Такая организация РОН сохранилась и в последующих МП фирмы Intel. Система команд содержала 46 универсальных инструкций. Цикл команды МП состоял из 8 тактов задающего генератора (так как чип монтировался в корпусе с

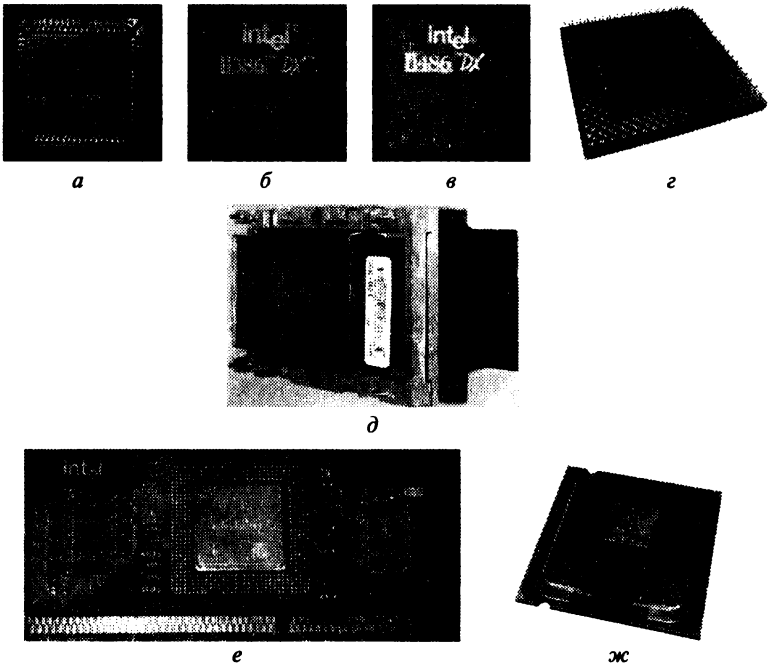


Рис. 3.16. Процессоры Intel:
а — i8086; *б* — i80386; *в* — i80486; *г* — Pentium MMX, интерфейс Socket 7; *д* — Itanium; *е* — Celeron, упаковка Single Edge Processor Package (SEPP)/Slot 1; *ж* — Core 2 Duo (интерфейс LGA775)

16 выводами и имел узкий интерфейс с «внешним миром», то приходилось применять мультиплексирование шины адреса и данных, причем 12-разрядный адрес выдавался порциями по 4 разряда, а прием команды требовал еще двух тактов, на выполнение самой инструкции затрачивалось всего три такта). Адресуемая память команд достигала 4 Кбайт (для сравнения: объем памяти мини-ЭВМ в начале 70-х гг. редко превышал 16 Кбайт).

Недостатки в скором времени были устранены в МП 4040, где количество РОН было доведено до 24, причем они были разделены на две области, выбираемые с помощью специальных команд, т. е. процессор теперь мог обращаться к двум блокам памяти команд емкостью 4 Кбайт, и за каждым из них можно было закрепить свою область регистров (8 РОН были всегда доступны для использования). Можно было разрабатывать самостоятельные программные модули, способные взаимодейство-

вать через общую часть регистрового файла. Но самое главное — это обработка одноуровневых прерываний, что превратило МП в прибор, применяемый в системах реального масштаба времени. «Остановка» создала возможность синхронизации МП с некоторыми внешними событиями. 60 инструкций, 8 Кбайт памяти команд, обработка прерываний стали достоинством этого МП и вывели его на первое место на рынке МП. Тем не менее специальных команд работы со стеком пока не было.

Intel 8008 (1972 г.). Первый 8-разрядный МП. Чип содержал уже 3500 транзисторов, работал на частоте 500 кГц при длительности машинного цикла 20 мкс (10 периодов задающего генератора) и в отличие от предшественников имел архитектуру ЭВМ принстонского типа (компьютер с единой памятью для команд и данных — архитектура Джона фон Неймана). В нем допускалось применение комбинаций постоянной и оперативной памяти. Значительные изменения (кроме увеличения разрядности) произошли и в регистровом файле. Из-за ограниченных возможностей применяемой технологии в качестве блока РОН была применена динамическая память, которая требовала производить регенерацию (были введены дополнительные аппаратные средства). МП большинство команд выполнял за 1—3 машинных цикла. Для работы с медленно действующими устройствами был введен сигнал готовности (*ready*). Система команд содержала в общем 65 инструкций и отличалась значительным количеством команд условного перехода, логических команд и команд сдвигов. Теперь МП мог адресоваться к памяти объемом 16 Кбайт. Однако объем и организация стека остались прежними, реализация операций со стеком по-прежнему возлагалась на программиста, узкий интерфейс с «внешним миром» требовал применения около 20 схем средней интеграции для сопряжения процессора с памятью и устройствами ввода-вывода.

Intel 8080 (1974 г.). Этот микропроцессор практически по всем параметрам разительно отличался от своих предшественников. Он работал с тактовой частотой 2 МГц, цикл команды — 2 мкс. Адресуемый объем памяти достиг 64 Кбайт, был внедрен эффективный механизм обработки прерываний, в результате использования корпуса с 40 выводами удалось разделить адресную и информационную шины процессора.

МП был окружен целым семейством новых микросхем (БИС контроллера ПДП, контроллера прерываний и др.). В результате этого проектирование микроЭВМ на базе семейства БИС значи-

Таблица 3.3. **Характеристики процессоров Intel**

Тип процессора	Архитектура	Год выпуска	Кодовое наименование	Количество транзисторов, млн	Ядро, мм^2	L1-кэш, Кбайт	L2-кэш, Кбайт	Размер минимальной структуры, мкм	Тактовая частота шины, МГц	Тактовая частота процессора, МГц	Потребляемая мощность, Вт	Интерфейс
8086		1978		0,029					4.77—8	4.77—8		
80286		1982		0,130				3,0	6—20	6—20		
80386DX	IA-32	1985		0,27				1,5	16—33	16—33		
80486DX		1989		1,2		8		1,0	25—50	25—50		
80486DX2		1992				8		0,8	25—40	50—80		
80486DX4		1994				16			25—40	75—120		
Pentium	P5	1993	P5	3,1	294	2×8	Внешн.	0,8	60—66	60—66	14—16	Socket 4
		1994—1995	P54	3,3	148	16	Внешн.	0,6	50—66	75—120	8—12	Socket 5.7
		1996—1996	P54C	3,3	83—91	16	Внешн.	0,35	66	133—200	11—15	Socket 7
MMX		1996—1997	P55C	4,5	146—128	2×16	Внешн.	0,28	66	166—233	13—17	Socket 7
PRO	P6	1995—1997	P6	5,5	306—195	2×8	256—1 Мбайт	0,60—0,35	60—66	150—200	37,9	Socket 8
Pentium II		1997	Klamath	7,6	203	2×16	512	0,35	66	233—300	34—43	Slot1
		1998	Deschutes	7,5	131—118	2×16	512	0,25	66—100	266—450	18—27	Slot1
Pentium III	P6	1999	Katmai	9,5	123	32	512	0,25	100—133	450—600	28—34	Slot1
		1999—2000	Coppermine	28,1	106—90	32	256	0,18	100	650—1,33 ГГц	14—37	Slot1/Socket 370
		2001—2002	Tualatin	44,0	95—80	32	256	0,13	133	1,0—1,4 ГГц	27—32	S 370

Продолжение табл. 3.3

Тип процессора	Архитектура	Год выпуска	Кодовое наименование	Количество транзисторов, млн	Ядро, мм^2	L1-кэш, Кбайт	L2-кэш, Кбайт	Размер минимальной структуры, мкм	Тактовая частота шины, МГц	Тактовая частота процессора, МГц	Потребляемая мощность, Вт	Интерфейс
Pentium M (мобильный)	P6/ Centrino	2003	Banias, Dothan	140	100—84	32	1024—2048	0,13—0,09	400 —533	800—2,26 ГГц	5—27	Socket M
Pentium IV	Netburst (IA-32e)	2000—2001	Willamette	42,0	217	8+12	256	0,18	400	1,3—2,0 ГГц	48—66	Socket 423/478
		2002—2004	Northwood	55,0	146—131	8+12	512	0,13	400—800	1,6—3,4 ГГц	46—82	Socket 478
		2004—2005	Prescott	125,0	122	16+12	1024	0,09	533—800	2,66—3,8 ГГц	89—115	Socket 478/LGA775
		2005	Prescott 2M	169	135	12+16	2048	0,09	800—1066	2,8—3,73	84—118	LGA775
		2005—2006	Cedar Mill	188,0	81	12+16	2048	0,065	800	3,0—3,8	80—86	LGA775
Pentiun IV Extreme Edition	Intel Netburst		Gallatin, Irwindale				512—1024	0,13—0,09	800—1066	3,2 —3,73 ГГц		Socket 478, LGA 775
Pentium D (двухъядерные)		2005	Smithfield (2×Prescott)	230,0	206	12+6×2	2×1,0 Мбайт	0,09	533—800	2,8 —3,2 ГГц	115—130	LGA775
		2006	Presler (2×Cedar Mill)	376,0	162	800	2×2,0 Мбайт	0,065	800—1066	3,4 ГГц	95—130	LGA775
Intel Core/Duo	Intel Core	2006	Yonah	151,0	90	32+32	2 Мбайта общих	0,065	533—667	1,06 —2,33 ГГц	14—49	Socket-M (PGA/BGA)

Продолжение табл. 3.3

Тип процессора	Архитектура	Год выпуска	Кодовое наименование	Количество транзисторов, млн	Ядро, мм^2	L1-кэш, Кбайт	L2-кэш, Кбайт	Размер минимальной структуры, мкм	Тактовая частота шины, МГц	Тактовая частота процессора, МГц	Потребляемая мощность, Вт	Интерфейс
Core 2/Duo	Core 2	2006	Conroe	291	143	32×2	1024 и более	0,065—0,045	533—1333	1,06—3,2 ГГц	5,5—130	LGA 775
		2006	Alendale	167	111	32×2	2—4 Мбайт	0,065	800—1066	1,8—2,66 ГГц	45—65	LGA775
		2006	Merom (мобильный)	291,0	143	32 +32	2—4 Мбайта общих	0,065	533— 800	1,06 —2,4 ГГц	5—35	Socket-M
Core 2/ Extreme		2007	Penryn/XE	205×4			3—6 Мбайт	0,045 (45 нм)	800 MT/c	1,1—2,8 ГГц	35	
Core 2 Quad		2007	Kentsfield	2×291	2×143	4×32	2 ×4096 KiB	65 нм	1066 MT/c	2400—2667	95—105	LGA775
		2008	Yorkfield-4M/6M	2×410	2×82	4×32	2 ×2048—2 ×6144 KiB	45 нм	1333 MT/c	2333—3000	95	LGA775
Core 2 Quad Extreme		2006—2007	Kentsfield XE	2×291	2×143	4×32	2 ×4096 KiB	65 нм	1066—1333 MT/c	2667—3000	130	LGA775
		2007—2008	Yorkfield XE	2×410	2×107	4×32	2 ×6144 KiB	45 нм	1600 MT/c	3000—3200 МГц	136—150	LGA775
Core i5-2100T	Core iX	2010	Sandy Bridge ×2	995	216	64 Кб ×2	256 Кб ×2	32 нм	5000	2,5 ГГц	35	Socket LGA1156
Core i7-870		2011	Lynnfield ×4	731	263	64 Кб ×4	256 Кб ×4	45 нм	2500	2,93 ГГц / до 3,6 ГГц (Turbo Boost)	95	LGA1156

Окончание табл. 3.3

Тип процессора	Архитектура	Год выпуска	Кодовое наименование	Количество транзисторов, млн	Ядро, мм^2	L1-кэш, Кбайт	L2-кэш, Кбайт	Размер минимальной структуры, мкм	Тактовая частота шины, МГц	Тактовая частота процессора, МГц	Потребляемая мощность, Вт	Интерфейс
Z500-Z540	Атом	2008	Silverthorne	47	25	32 + 24	512	0,045 (45 нм)	400 МТ/с	0,8—1,866 ГГц	0,65—2,4	441-ball μFCBGA
N270-N230		2008	Diamondville	47	25	32 + 24	512	0,045 (45 нм)	533 МТ/с	1,6 ГГц	2,5—4,0	441-ball μFCBGA
Xeon	P5, P6, Netburst	1998	Ядро Pentium II	См. Pentium II			512—1,0 Мбайт	0,18	100	400		Slot2
Xeon	P5, P6, Netburst	1999—2000	Tanner	См. Pentium III			512—2,0 Мбайт	0,13	100—133	500—733		Slot2
		2001	Foster	См. Pentium IV			512—1,0 Мбайт	0,09—0,65		1,4—1,7 ГГц		Socket 603/604
Celeron	P5, P6, Netburst	1998	Covington	7,5	131	32	Нет	0,25	66	266—300	16—18	Slot 1
		1998—2000	Mendocino	19,0	154	32	128	0,25	66	300—533	19—26	Socket 370/ Slot 1
Celeron	P5, P6, Netburst	2000	Coppermine	28,1	105/90	32	128	0,18	100	533—1,1 ГГц	11—33	Socket-370
		2002	Willamette	42,0	217	8	128	0,18	400	1,7—1,8 ГГц	63—66	S 478
		2002—2004	Nordwood	55,0	131	8	128	0,13	400	2,0—2,8 ГГц	59—68	S 478
Celeron D	Netburst	2004—2006	Prescott	140,0	120	16	256	0,09	533	2,133—3,33 ГГц	73—84	S 478/LGA775
		2004/2006	Cedar Mill	188,0	81	16	512	0,065	533	3,33 ГГц	86	LGA775
Itanium	IA-64	1999	Merced/ Itanic	30,0—220		2—4 Мбайт L3		0,18	733—800	800—1,0 ГГц		PAC418
Itanium 2		2003	Madison	410,0		6,0 Мбайт L3		0,13		1,5 ГГц		PAC611
Itanium (двухъядерный)		2006	Montecito	1720,0	596	16+16 Кбайт L1 1 Мбайт+256 Кбайт L2 24 Мбайт L3		0,09	533—400	1,4—1,6 ГГц	75—104	PAC611

тельно упростилось. МП стал стандартом де-факто, но были и недостатки (например, уже имелись МП других фирм, которые оказались более прозрачными для программистов). Далее была выпущена серия периферийных контроллеров. Но самое значительное достижение состояло в том, что компания создала системное программное обеспечение (ПО) — однопользовательскую операционную систему (ОС) ISIS II и ОС реального времени iRMX-80 (мощнейшая в то время программная поддержка своих изделий). Фирма в 1976 г. приступила к выпуску одноплатных микроЭВМ серии iSBC на базе своих МП-комплектов.

Intel 8086 (объявлен 8 июня 1978 г.) — первое поколение. МП содержал 29 тыс. транзисторов (рис. 3.16). Высокое быстродействие элементов обеспечило тактовую частоту 5 МГц, а 16-разрядная архитектура и машинный цикл 200 нс — производительность, превышающую аналогичный параметр 8080 на порядок. Именно стратегия эволюционного, а не революционного развития, выбранная фирмой Intel была верна и давала свои плоды. Программная совместимость была исключительно важной характеристикой, которая объединяла 86-й кристалл с его предшественниками. Структура МП была полностью перестроена, он как бы был разделен на два одновременно работающих функциональных блока. Это операционный блок (EU — Execution Unit) и блок интерфейса (BIO — Bus Interface Unit). В результате исполнение одной команды совмещалось во времени с выборкой следующей команды или данных из памяти. Таким образом, из универсальных ЭВМ МП позаимствовали еще одно техническое решение — реализацию принципов параллелизма. В ЦП появился небольшой буфер команд, что давало дополнительную экономию времени при обращениях к памяти.

Адресация 1 Мбайт оперативной памяти (благодаря 20 адресным линиям) и ее сегментация могут быть отнесены к одним из наиболее существенных достижений. Сегментация памяти и большое число уровней прерываний были ориентированы на работу системы в многозадачном режиме. Однако механизм защиты памяти пока реализован не был, и это в ряде случаев существенно усложняло разработку программного обеспечения.

Наряду с поддержкой ввода-вывода по каналу ПДП дополнительно обеспечивалась адресация до 64 К портов программно-управляемого ввода-вывода.

МП 8086 мог работать в двух режимах: минимальном (рассчитанном на использование в небольших системах без приме-

нения БИС контроллера шины) и максимальном (ориентированном на применение МП в сложных системах с использованием БИС контроллера шины). В систему команд входило 147 инструкций, позволяющих решать задачи управления практически любой сложности. Появились операции умножения и деления 16-разрядных чисел со знаком и без знака, команды обработки массивов данных, программно-управляемые прерывания и др., что превратило чип в универсальный прибор, который мог успешно применяться как для построения сложных контроллеров, так и в качестве ЦП ЭВМ общего назначения. Кроме того, МП вышел в мощном сопровождении средств поддержки (вспомогательных БИС, средств разработки и отладки аппаратуры и системного ПО и т. д.).

Наряду с этим фирмой Intel был выпущен процессор 8088 с 8-разрядной внешней шиной данных. Из-за применения экономичных 8-разрядных микросхем появился ПК с МП 8088 (фирма IBM, ПК класса XT — Extended Technology — расширенная технология, тактовая частота 4,77 МГц). На базе 8086 были выпущены младшие модели 25 и 30 семейства PS/2.

Процессор i8088 также относится в настоящее время к процессорам первого поколения.

Использование чипов 8086 в IBM PC предопределило дальнейшее развитие корпорации Intel как разработчика и изготовителя универсальных процессоров общего назначения. Был изготовлен 16-разрядный арифметический сопроцессор 8087, который позволил превратить ПК еще и в достаточно мощный инструмент для решения задач вычислительного характера.

Intel 80286 (1 февраля 1982 г.) — второе поколение. В этом процессоре было введено большое количество новшеств. В 1984 г. фирма IBM использовала этот МП в PC AT (AT — Advanced Technology — улучшенная технология). Вот основные новшества этого чипа:

- адресное пространство составляло 16 Мбайт (вместо 1 Мбайт у предшественников), так как использовалась 24-разрядная шина адресов;
- поддержка виртуальной памяти (это позволяло использовать внешнюю память для имитации большой реальной внутренней памяти емкостью до 1 Гбайт);
- аппаратная мультизадачность (эта архитектурная новинка позволяла в ПК одновременно выполнять несколько задач с большой скоростью переключения с одной на другую);

- повышенное быстродействие (4 МГц, однако вскоре рабочая частота была повышена до 8 МГц и стала стандартной, хотя производители клонов эту частоту довели до 10; 12,5; 16 и 20 МГц);
- встроенная система управления памятью и средства ее защиты (открывали широкие возможности использования МП в многозадачных средах);
- дополнение системы команд 16 новыми инструкциями;
- размещение в одном кристалле контроллеров прерываний и ПДП, а также таймера и системного генератора.

МП мог работать в двух режимах — *реальном* (МП действовал как 8086, что обеспечивало совместимость с DOS и существующим ПО) и *защищенном* (в этом режиме МП реализовывал режим виртуальной памяти, аппаратную мультизадачность и адресацию к большему пространству памяти).

Операционная система MS DOS может работать только в реальном режиме. Другие операционные системы, например, OS/2 (предложенная фирмой IBM — альтернатива DOS) и UNIX (XENIX и AIX), могут использовать защищенный режим и, следовательно, расширенные возможности 80286. Многие новшества, введенные в этот чип, впоследствии переходили от поколения к поколению МП фирмы Intel. Имелись и определенные нерешенные проблемы, связанные с многозадачностью, повышением производительности, совершенствованием тракта процессор—память и устройства управления памятью (для эффективного функционирования ПК под управлением многозадачных ОС).

Архитектура IA-32. Intel 80386 (17 октября 1985 г.) — третье поколение. Архитектура IA-32 (иногда называемая x86-32) является архитектурой 32-битовой системы команд семейства микропроцессоров Intel (предыдущая архитектура была 16-битовой) и появляется с Intel80386SX.

Данный МП был процессором для ЭВМ общего назначения. Размещение на кристалле 275 тыс. транзисторов дало возможность полностью реализовать 32-разрядную архитектуру.

Главные особенности:

- обеспечивает 32-разрядный ввод-вывод, 32-битовую адресацию основной памяти (адресуемая реальная память — до 4 Гбайт, т. е. $2^{32} = 4\,294\,967\,296$ байт) и емкостью до 64 Тбайт (64×2^{40} байт) виртуальной памяти;
- рабочая тактовая частота равнялась 33 МГц;

- в МП были встроены система управления памятью и защиты (регистры преобразования адреса, механизмы защиты оперативной памяти, улучшенные аппаратные средства поддержки многозадачных ОС), средства работы с виртуальной памятью со страничной организацией памяти (в устройство менеджера памяти — MMU (*memory management unit*) помимо блока сегментации — SU (*segmentation unit*) был включен блок управления страницами — PU (Paging Unit), благодаря этому относительно просто реализовывались процессы свопинга (перестановка сегментов из одного места памяти в другое).

Предварительная выборка команд, буфер для команд (внутренняя кэш-память) 16 байт, конвейер команд и механизмы выполнения функций преобразования адреса значительно уменьшили среднее время выполнения команды (3—4 млн команд в секунду, что в 6—8 раз превышало аналогичный показатель 8086). Как и раньше, новый чип был совместим со своими предшественниками на уровне объектных кодов.

Наиболее существенной особенностью 80386 было использование кэш-памяти, значительно повышающей производительность системы (еще один атрибут универсальных ЭВМ, который начал применяться в МП-системах). Для управления этой памятью был разработан специальный контроллер, с помощью которого формировался двухходовый множественный ассоциативный кэш (обеспечивал буфер емкостью до 32 Кбайт и высокий коэффициент удачных обращений). Но математический сопроцессор был еще автономным на отдельном кристалле (80387).

Реализованы три режима работы 80386: реальный, защищенный и виртуальный МП 8086. Процессор 80386 как бы включает в себя три разных процессора.

В рабочем режиме — Real Mode (реальный режим — стартовый) — он ведет себя как 8086, т. е. тот же 8086 с расширенным набором команд и имеющий доступ к первому Мбайту памяти (при этом возможности 80386 используются не полностью, но на ней могут выполняться все программы, написанные для 8086/8088 и причем значительно быстрее).

Защищенный режим (Protected Mode) 80386 соответствует аналогичному режиму 80286, имеет доступ к 16 Мбайт памяти и расширенному набору команд, а также имеет возможность использовать систему мультипрограммирования (в основном могут выполняться несколько прикладных программ, чаще работаю-

щих в среде Windows и поддерживающих защищенный режим). Если заменить ОС на OS/2 (разработка специально для защищенного режима), то появится возможность полностью использовать функциональные возможности этого режима.

Последний, третий режим 80386 — Virtual Real Mode. В этом режиме он одновременно заменяет некоторое количество параллельно работающих 8086/8088, т. е. одновременно могут быть задействованы несколько программ, которые выполняются соответствующими процессорами 8086/8088. Здесь нет ограничения 1 Мбайт на память. Ядром многозадачности является основная программа, переключающая процессор в виртуальный режим и контролирующая текущие процессы выполнения различных программ (например, система Windows).

80386 внутренне одновременно оперирует 32 битами и имеет внешний 32-битовый интерфейс, но к тому времени большинство устройств и микросхем были 16-битовыми и не могли использовать эту возможность МП. Intel повторила опыт МП 8088 (8086 — 16 бит внутренние и внешние, а 8088 — 16 бит внутренние и 8 бит внешние) и создала 80386 с 16-битовым интерфейсом (он получил название 80386 SX), который оказался меньше и дешевле. Полноразрядный 80386 получил название 80386 DX.

Intel 80486 (10 апреля 1989 г.) — четвертое поколение. Здесь в результате повышения степени интеграции в 1,2 млн транзисторов открылась возможность реализовать на одном кристалле не только кэш-память, но и математический сопроцессор. Для кэш-памяти использовался более эффективный четырехвходовый статический буфер, который, будучи размещенным в чипе, мог работать на тактовой частоте МП (намного быстрее, чем ОП). Здесь применялся «групповой режим» — самый скоростной режим доступа к шине, обеспечивающий быстрое заполнение кэш-памяти МП. Интегрированное в чип МП 8 Кбайт кэш-памяти, управляемой через контроллер, называется внутренней (Internal Cache), т. е. Level 1 (L1). Имеется также внешняя кэш-память (External Cache), т. е. Level 2 (L2). 80 % команд могут выполняться за один такт (применяется конвейерная обработка). Этот прибор так же, как и предыдущие МП, функционировал в трех режимах и был ориентирован на многозадачные среды. Производительность в задачах вычислительного характера возросла в 3—4 раза (за счет интеграции МП и сопроцессора).

В зависимости от режима 80486 работает с различной частотой.

Intel 80486 DX, основное отличие — отсутствие внутреннего математического сопроцессора, рабочие частоты — 25 и 33 МГц.

Intel 80486 DX2 (март 1992 г.) — усовершенствованный вариант 80486 DX (он на внешнем уровне представляет собой МП с тактовой частотой 25 или 33 МГц, который, однако, на внутреннем уровне работает с тактовой частотой 50 или, соответственно, 66 МГц). Тогда команды, которые не используют передачу данных на внешнюю шину, выполняются почти в 2 раза быстрее (это на практике означает повышение эффективности от 50 до 95 %). Эти МП обеспечили новую технологию, при которой скорость работы внутренних блоков МП в 2 раза выше скорости остальной части системы (появилась возможность объединения высокой производительности МП с внутренней тактовой частотой 50 (66) МГц и эффективной по стоимости системой на 25/33 МГц).

Intel 80486 DX4 (март 1994 г.). Процессор 486DX с внутренним утроением частоты совместим с 486DX, кэш объемом 16 Кб, 1,6 млн транзисторов, технология 0,6 мкм, 75 МГц, 53 млн операций в секунду, 100 МГц, 70 млн операций в секунду.

Микроархитектура P5. Представленный в 1993 г., ЦП Pentium (рис. 3.17) был преемником линий 486 Intel и позиционировался как процессор пятого поколения.

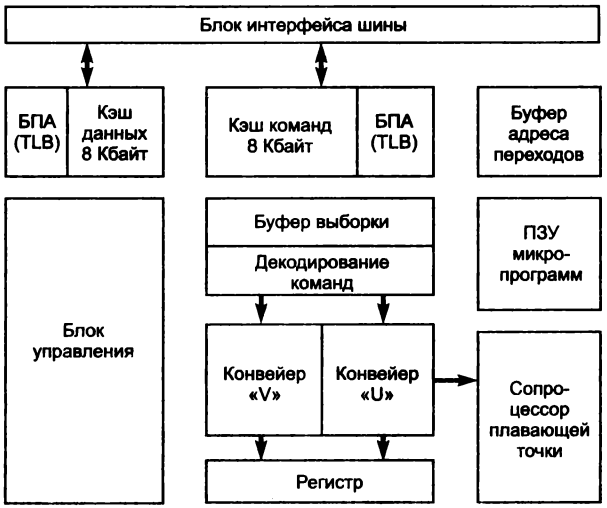


Рис. 3.17. Основные компоненты процессора Pentium

Первый Pentium имел внутреннее кодовое наименование P5, использовал конвейеризацию (суперскаляризацию) и производился с использованием процесса 0,8 мкм. Затем появились ЦП P54, производившиеся по технологии 0,6 мкм и имевшие внутреннюю тактовую частоту, отличную от частоты первичной системной шины (FSB) (оказалось намного более трудным увеличить частоту шины, чем процессора). В дальнейшем вышли ЦП P54C, где использовался технологический процесс 0,35 мкм — чистый процесс CMOS, в отличие от «биполярного процесса» CMOS, который применялся в более ранних Pentium.

Основные архитектурные отличия, которые привели к существенному повышению эффективности Pentium по сравнению с ЦП 486, состояли в следующем:

- суперскалярная архитектура — Pentium имел два конвейера: первый («U-конвейер») мог обрабатывать любые команды, в то время как второй («V-конвейер») — только простые команды. Использование более чем одного конвейера характерно для RISC-процессоров, и это стало началом применения Intel RISC-методов на семействе процессоров Pentium;
- шина данных на 64 бита — каждое обращение к памяти позволяло получать вдвое большее количество информации, чем это было в предыдущих чипах.

Следующий процессор P55C Pentium MMX базировался на ядре P5 и процессе изготовления 0,35 мкм. Здесь также был удвоен размер до 32 Кбайт и расширен набор команд, чтобы оптимизировать обработку мультимедиа-данных.

Микроархитектура P6. P6 — шестое поколение x86 архитектуры процессора Intel, первоначально осуществленной в дизайне Pentium Pro (рис. 3.18), представленного в 1995 г. в качестве преемника исходного Pentium P5.

Pentium Pro (1 ноября 1995 г.) имеет три конвейера, каждый из которых включает 14 ступеней. Для постоянной загрузки имеется высокоэффективный четырехвходовый кэш команд и высококачественная система предсказания ветвлений на 512 входов. Дополнительно для повышения производительности была применена буферная память (кэш) второго уровня емкостью 256 Кбайт, расположенная в отдельном чипе и смонтированная в корпусе ЦП. В результате стала возможной эффективная разгрузка пяти исполнительных устройств: два блока целочисленной арифметики; блок чтения (load); блок записи (store);

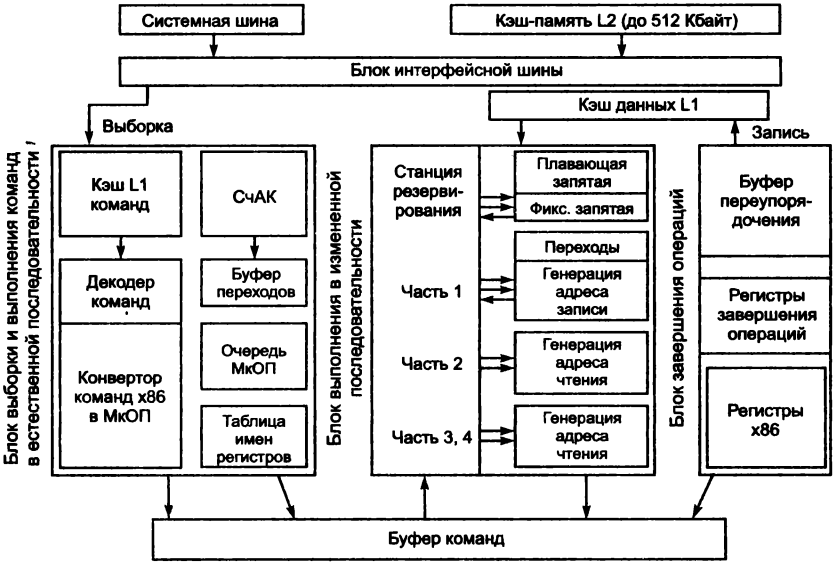


Рис. 3.18. Процессор Pentium Pro

FPU (Floating-Point Unit — устройство арифметических операций с плавающей точкой).

Pentium P55 (Pentium MMX), 8 января 1997 г. Pentium MMX — версия Pentium с дополнительными возможностями. Технология MMX должна была добавить/расширить мультимедийные возможности компьютеров. MMX объявлен в январе 1997 г., тактовая частота 166 и 200 МГц, в июне того же года появилась версия 233 МГц. Технологический процесс 0,35 мкм, 4,5 млн транзисторов.

Pentium II (7 мая 1997 г.). Процессор представляет собой модификацию Pentium Pro с поддержкой возможностей MMX. Была изменена конструкция корпуса — кремниевую пластину с контактами заменили на картридж, увеличена частота шины и тактовая частота, расширены MMX-команды. Первые модели (233—300 МГц), производились по технологии 0,35 мкм, следующие — по 0,25 мкм. Модели с частотой 333 МГц выпущены в январе 1998 г. и содержали 7,5 млн транзисторов. В апреле того же года появились версии 350 и 400 МГц, а в августе — 450 МГц. Все P2 имеют кэш второго уровня объемом 512 Кбайт. Есть также модель для ноутбуков — Pentium II PE, и для рабочих станций — Pentium II Xeon 450 МГц.

Pentium III (26 февраля 1999 г.). P3 — один из самых мощных и производительных процессоров Intel, но в своей конструкции он мало чем отличается от P2, увеличена частота и добавлено около 70 новых команд (SSE). Первые модели объявлены в феврале 1999 г., тактовые частоты — 450, 500, 550 и 600 МГц. Частота системной шины 100 МГц, 512 Кбайт кэша второго уровня, технологический процесс 0,25 мкм, 9,5 млн транзисторов. В октябре 1999 г. также выпущена версия для мобильных компьютеров, выполненная по 0,18-мкм технологии с частотами 400, 450, 500, 550, 600, 650, 700 и 733 МГц. Для рабочих станций и серверов существует P3 Хеоп, ориентированный на системную логику GX с объемом кэша второго уровня 512 Кбайт, 1 Мбайт или 2 Мбайт.

Pentium M (здесь «M» означает «мобильный», mobile) — последний процессор микроархитектуры Intel P6. Это 32-разрядные одноядерные процессоры с системой команд x86, вышедшие в марте 2003 г. и входящие в платформу Intel Centrino. Максимальное энергопотребление Pentium M находится на уровне 3—25 Вт, и процессоры предназначаются для мобильных ПК. Особенности архитектуры:

- видоизмененное ядро Pentium III (конкретно — Pentium III Tualatin);
- используется интерфейс с шиной, разработанный для Pentium IV;
- усовершенствованный декодер команд и блок предсказания переходов;
- поддержка SSE2;
- больший объем кэш-памяти;
- технология энергосбережения SpeedStep 3 (предусматривающая большее число стадий «сна», чем предшествующие версии). Например, Pentium M (1,6 ГГц) в зависимости от загрузки переключается на частоты 600, 800, 1000, 1200, 1400 и 1600 МГц, при этом энергопотребление может изменяться от 5 до 27 Вт.

Banias. Поскольку линия процессоров «M» была первоначально разработана Центром Исследований и Разработок Intel (Intel's Research & Development Center) в Израиле, первый Pentium M получил кодовое имя «Banias» (античное наименование Голанских Высот), а в дальнейшем именовался Pentium M 705. Процессор изготовлялся по технологии 130 нм, рассчитан на частоты от 1,3 до 1,7 ГГц, использовал FSB на 400 MT/s,

и имел L1-кэш размера 1 MiB. Средняя энергоемкость ядра (Thermal Design Power — TDP) составляла 24,5 Вт.

Dothan. Усовершенствованная версия Pentium M (известная как Dothan, по названию другого античного израильского поселения) была выпущена в мае 2004 г. Pentium M был один из первых процессоров Intel, которым были присвоены «номера процессоров» («processor number», см. рис. 3.39) вместо частотного рейтинга, и основные образцы изделия известны как Pentium M 710 (1,4 ГГц), 715 (1,5 ГГц), 725 (1,6 ГГц), 735 (1,7 ГГц), 745 (1,8 ГГц), 755 (2,0 ГГц) и 765 (2,1 ГГц). Процессор не поддерживал ни возможностей hyperthreading, ни набор команд SSE3.

Эта серия процессоров Pentium M была выпущена на основе технологии 90 нм, с удвоенным размером вторичного кэша. Размер кристалла, на котором размещалось 140 млн транзисторов (в основном используемых для реализации массивной кэш-памяти на 2 MiB), составлял 84 мм^2. Энергопотребление снизилось до 21 Вт. Тесты, выполненные независимыми исследователями, показали, что Dothan превышает Banias по эффективности примерно на 10—20 % в большинстве ситуаций.

В 2005 г. был выпущен переработанный Dothan, предназначенный для чипсета Sonoma, поддерживающий FSB на 533 MT/s и опцию XD (версия Intel дляNX bit). Это были изделия с номерами 730 (1,6 ГГц), 740 (1,73 ГГц), 750 (1,86 ГГц), 760 (2,0 ГГц) 770 (2,13 ГГц) и 780 (2,26 ГГц), которые имели энергопотребление 27 Вт и кэш L2 на 2 MiB.

Частота процессоров к июлю 2005 г. достигла диапазона от 1,0 до 2,26 ГГц. Модели 718 (1,3 ГГц), 738 (1,4 ГГц), 758 (1,5 ГГц), 778 (1,60 ГГц) являются низковольтными (1,116 В) с тепловыделением 10 Вт, тогда как 723 (1,0 ГГц), 733 (1,1 ГГц) и 753 (1,2 ГГц) — ультранизковольтные (0,940 В), тепловыделение — 5 % Вт.

Несмотря на то что Intel позиционировала Pentium M исключительно как мобильный продукт, производители системных плат, такие как AOpen, DFI и MSI, начали поставлять изделия, совместимые с Pentium M, ориентируясь на рабочие станции, серверы и энтузиастов-пользователей. ASUS даже выпустила адаптер (CT-479), позволяющий устанавливать Pentium M на некоторые системные платы ASUS, спроектированные для Pentium IV (Socket 478). Фирма Shuttle Inc. предлагает настольные ПК на Pentium M, характеризующиеся как малошумные изделия с низким энергопотреблением.

Архитектура P6 продержалась три поколения — от Pentium Pro до Pentium III — и характеризовалась малым энергопотреблением, хорошей общей производительностью и относительно высоким отношением «число команд/число циклов» (instructions per cycle — IPC).

Архитектура IA-64. Данная архитектура была объявлена Intel в мае 1999 г. Типичным представителем архитектуры является ЦП Itanium. Процессоры IA-64 располагают мощными вычислительными ресурсами, включая 128 регистров для ФЗ, 128 регистров ПЗ и 64 регистра предикации наряду с множеством регистров специального назначения (рис. 3.19). Команды должны группироваться для параллельного выполнения различными функциональными модулями. Набор команды оптимизирован, чтобы обеспечить вычислительные потребности криптографии, видеокодирования и других функций, которые все более необходимы следующим поколениям серверов и рабочих станций. В процессорах IA-64 также поддерживаются и развиваются MMX-технологии и SIMD-расширения.

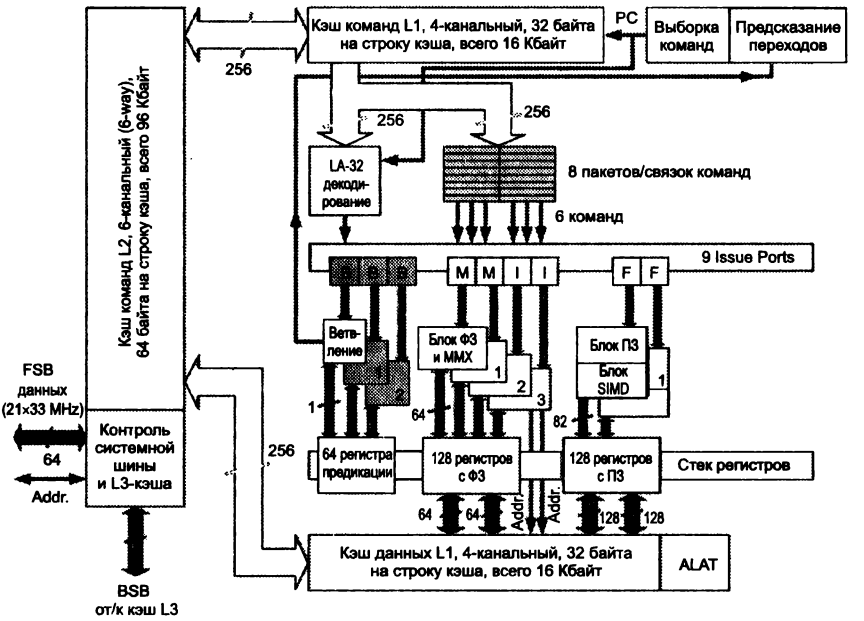


Рис. 3.19. Процессор Itanium

Архитектура IA-64 не является ни 64-битовой версией архитектуры Intel IA-32, ни адаптацией предложенной Hewlett-Packard архитектуры PA-RISC на 64 бита, а представляет собой полностью оригинальную разработку (табл. 3.4). IA-64 — это компромисс между CISC и RISC, попытка сделать их совместимыми (существуют два режима декодирования команд — VLIW и CISC, и ЦП автоматически переключается в необходимый режим исполнения).

Таблица 3.4. **Основные различия архитектур IA-32 и IA-64**

Характеристики	
Архитектура x86	***Архитектура IA-64***
Использование сложных команд переменной длины, обрабатываемых по одной	Использование простых команд одинаковой длины, сгруппированных по 3
Переупорядочивание и оптимизация команд в процессе исполнения	Переупорядочивание и оптимизация в процессе компиляции
Попытки предсказания переходов (ветвлений)	Выполнение нескольких последовательностей команд одновременно без предсказания ветвлений
Считывание данных из памяти (загрузка) по мере необходимости, в первую очередь проверяя кэш	Загрузка данных прежде, чем они потребуются

Основные инновационные технологии IA-64 — длинные слова команд (long instruction words — LIW), предикаты команд (instruction predication), устранение ветвлений (branch elimination), предварительное чтение данных (speculative loading) и другие ухищрения для того, чтобы «извлечь больше параллелизма» из кода программ.

Основная проблема архитектуры IA-64 заключается в отсутствии встроенной совместимости с x86 кодом, что не позволяет процессорам IA-64 эффективно работать с программным обеспечением, разработанным за последние 20—30 лет. Intel оборудует свои процессоры IA-64 (Itanium, Itanium 2 и т. д.) декодером, который преобразует инструкции x86 в команды IA-64. Декодер не является самым эффективным как по способу реализации, так и по принципу построения, ведь аппаратная поддержка инструкций x86 работает значительно быстрее. Поэтому

Itanium и Itanium 2 характеризуются низкой производительностью в приложениях x86.

Montecito — кодовое наименование двухъядерных процессоров Intel Itanium 2, которые впервые были выпущены 18 июля 2006 г. под официальным наименованием «Dual-Core Intel Itanium 2 processor». По данным Intel, ЦП Montecito позволяет удвоить производительность по сравнению с предшествующим одноядерным Itanium 2 при снижении энергопотребления на 20 %.

Особенности архитектуры (рис. 3.20):

- два ядра на кристалл;
- многопоточность (multi-threading) до 2 потоков на ядро, 4 потоков на кристалл;
- для каждого ядра выделяется L1-кэш по 16 Кбайт для программ и данных;
- для каждого ядра кэш L2 размером 1 Мбайт для команд и 256 Кбайт для данных;
- кэш L3 по 12 Мбайт на каждое ядро;

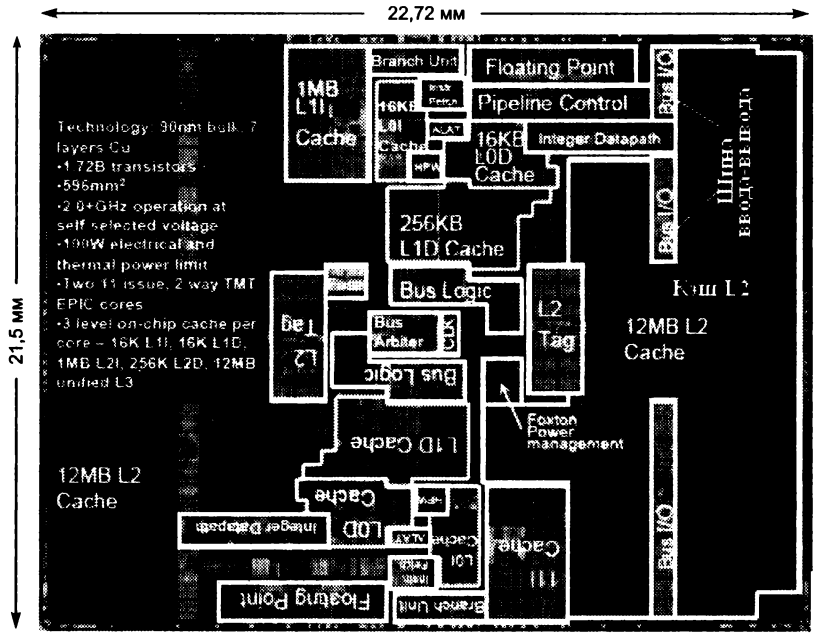


Рис. 3.20. Двухъядерный процессор «Dual-Core Intel Itanium 2 processor» (Montecito)

- 1,72 млрд транзисторов, в том числе:
  — схемы логики ядра — 57 млн, по 28,5 млн на ядро;
  — кэши ядер — по 106,5 млн;
  — кэш L3 (24 Мбайта) — 1550 млн;
  — схемы шин и ввода-вывода — 6,7 млн транзисторов;
- размер кристалла — 27,72 × 21,5 мм, или 596 мм^2.
- технологический процесс 90 нм;
- более низкое энергопотребление и тепловыделение по сравнению с более ранними ЦП Itanium, несмотря на большее число транзисторов и тактовую частоту (75—104 Вт). Это достигается в основном за счет использования различных типов транзисторов — при низкой нагрузке используются более медленные и слаботочные приборы, а с повышением нагрузки подключаются более мощные и сильноточные транзисторы;
- управление энергопотреблением Demand Base Switching — технология, поддерживаемая совместно с ОС;
- улучшенная компенсация ошибок в кэш-памяти (кодовое наименование — Pellston technology, официальное — Cache Safe Technology);
- технология виртуализации, поддерживающая работу нескольких копий ОС одновременно (кодовое наименование Silvervale technology, официальное — Intel Virtualization Technology);
- улучшенная полоса пропускания первичной шины (FSB), утроенная по сравнению с предшествующими образцами (до 21 Гбайт/с).

NetBurst. Это название Intel дала новой архитектуре, которая последовала за микроархитектурой P6. Концепция NetBurst должна была улучшить производительность, повысить эффективность выполнения команд вне естественного порядка и позволить создать процессор, который может достигнуть намного более высоких частот и более высокой производительности относительно микроархитектур P5 и P6, при обеспечении обратной совместимости (рис. 3.21, а также см. рис 3.2, *б*).

Первый представитель архитектуры NetBurst — ЦП 7-го поколения Pentium IV (ядро Willamette, конец 2000 г. — рис. 3.22) — являлся самым большим шагом развития к архитектуре IA-32. Одно из самых важных изменений — внутренний конвейер процессора (гиперконвейер или Hyper Pipeline), включающий 20 стадий против 10 для микроархитектуры P6 и способствующий ра-

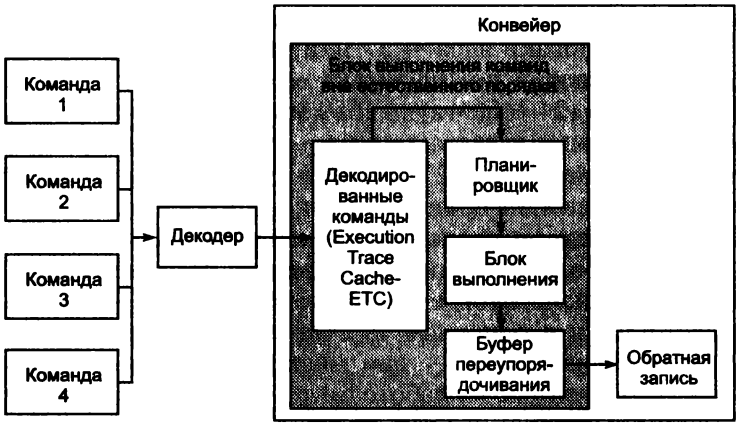


Рис. 3.21. Микроархитектура Netburst

боте процессора на значительно более высоких частотах, чем у его предшественников.

Микроархитектура NetBurst предусматривает только один декодер (в противоположность трем для P6), а блок выполнения команд вне естественной последовательности включает кэш-память декодированных команд (execution trace cache — ETC), которая хранит расшифрованные микрооперации.

Pentium IV заранее превращает x86-инструкции в микрооперации, которые записываются в Trace Cache. Имеются также несколько буферов, используемых, чтобы демпфировать поток микроопераций. Способность ядра выполнять инструкции не в порядке поступления остается ключевым фактором в обеспечении возможности параллелизма (см. также рис. 3.2).

Pentium IV Prescott (февраль 2004 г.). В начале февраля 2004 г. Intel анонсировала четыре новых процессора Pentium IV (2,8; 3,0; 3,2 и 3,4 ГГц), основанных на ядре Prescott, которое включает ряд нововведений. Вместе с выпуском четырех новых процессоров Intel представила процессор Pentium IV 3.4 EE (Extreme Edition), основанный на ядре Northwood и имеющий 2 Мбайт кэш-памяти третьего уровня, а также упрощенную версию Pentium IV 2.8 A, основанную на ядре Prescott с ограниченной частотой шины (533 МГц).

Prescott выполнен по технологии 90 нм, что позволило уменьшить площадь кристалла, причем число транзисторов было увеличено более чем в 2 раза. В то время как ядро Northwood

имеет площадь 145 мм^2 и на нем размещено 55 млн транзисторов (рис. 3.22, *б*), ядро Prescott имеет площадь 122 мм^2 и содержит 125 млн транзисторов (рис. 3.22, *в*).

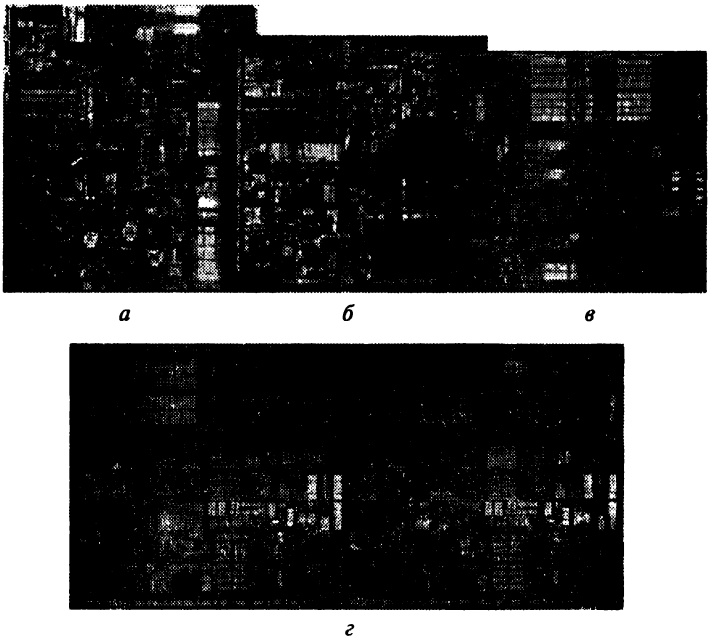


Рис. 3.22. Ядра процессоров Pentium IV:
а — Willamette, 0,18 мкм; *б* — North0wood, 0,13 мкм; *в* — Prescott, 0,09 мкм; *г* — Smithfield (2 × Prescott 1M)

Перечислим некоторые отличительные особенности процессора:

- новые SSE-команды. Intel представила в Prescott новую технологию SSE3, которая включает 13 новых потоковых команд, которые увеличат производительность некоторых операций как только программы начнут их использовать. SSE3 является не просто расширением SSE2, так как добавляет новые команды, но и позволяет облегчить и автоматизировать процесс оптимизации готовых приложений средствами компилятора. Другими словами, разработчику программного обеспечения не надо будет переписывать код программы, необходимо будет только перекомпилировать ее;

- увеличенный объем кэш-памяти. Одним из важнейших с точки зрения производительности дополнений можно считать увеличенный до 1 Мбайт кэш второго уровня. Объем кэш-памяти первого уровня также был увеличен до 16 Кбайт;
- улучшенная предвыборка данных. Ядро Prescott имеет улучшенный механизм предвыборки данных;
- улучшенный Hyperthreading. В новую версию включено множество новых особенностей, способных оптимизировать многопоточное выполнение различных операций. Единственный недостаток новой версии заключается в необходимости перекомпиляции программного обеспечения и обновления операционной системы;
- увеличенная длина конвейера. Для увеличения рабочей частоты будущих процессоров ядро Prescott имеет увеличенную с 20 до 31 ступени длину конвейера. Увеличение длины конвейера негативно сказывается на производительности в случае неправильного предсказания ветвлений. Для компенсации увеличения длины конвейера была улучшена технология предсказания ветвлений.

Cedar Mill. «Последним из могикан» модельного ряда Pentium IV был ЦП Cedar Mill, выпущенный в начале 2006 г. и представлявший собой прямую переделку ядра серии 600 на технологию 65 нм, без какого-либо качественного развития.

Выпуск ядра Prescott, для которого Intel использовала технологический процесс 90 нм, вскрыл ряд труднопреодолимых проблем. Первоначально NetBurst была объявлена специалистами Intel как архитектура с существенным запасом производительности, который со временем можно будет реализовать посредством постепенного наращивания тактовой частоты. Однако на практике оказалось, что увеличение тактовой частоты процессора влечет за собой неприемлемое возрастание тепловыделения и энергопотребления. Причем, происходящее параллельно развитие технологии производства полупроводниковых транзисторов не позволяло эффективно бороться с ростом электрических и тепловых характеристик.

В результате третье поколение процессоров с архитектурой NetBurst (Prescott) осталось в истории процессоров как одно из самых «горячих» (процессоры, построенные на этом ядре, могли потреблять и, соответственно, выделять до 160 Вт, получив прозвище «кофеварки», «coffee heater») при том, что их тактовая час-

тота не поднялась выше 3,8 ГГц. Высокие тепловыделение и энергопотребление вызвали множество смежных проблем. Процессоры Prescott требовали использования специальных материнских плат с усиленным стабилизатором напряжения и особых систем охлаждения.

Проблемы с высокими тепловыделением и энергопотреблением были бы не столь заметны, если бы не то обстоятельство, что при всем при этом процессоры Prescott не смогли продемонстрировать высокой производительности, благодаря которой можно было бы закрыть глаза на упомянутые недостатки. Заданный конкурирующими процессорами AMD Athlon 64 уровень быстродействия оказался для Prescott практически недостижимым, в результате этого данные ЦП стали восприниматься как провал Intel.

Поэтому не вызвало особого удивления, когда оказалось, что преемник NetBurst будет основываться на принципе эффективного энергопотребления, принятом в мобильной микроархитектуре Intel и воплощенном в семействе процессоров Pentium M.

Тем не менее перед появлением архитектур Core и Core 2 были выпущены двухъядерные процессоры Pentium D (Smithfield и Presler — рис. 3.23).

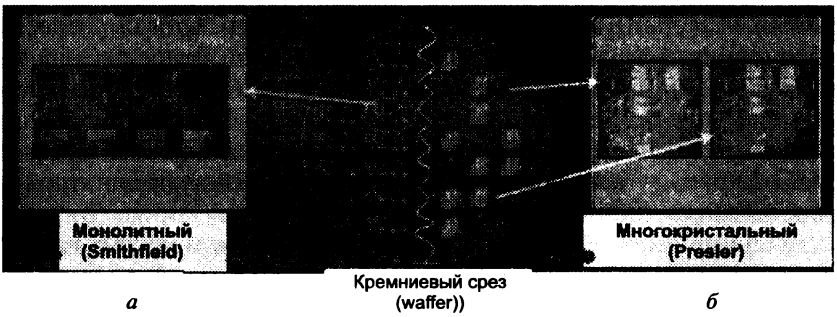


Рис. 3.23. Варианты изготовления двухъядерных процессоров: *а* — Smithfield; *б* — Presler

Pentium D. Появление многоядерных процессоров для настольных ПК, как ожидается, прекратит гонку за тактовую частоту между Intel и AMD, которая длилась в течение нескольких лет (дальнейший рост частоты будет, вероятно, по-прежнему происходить по экспоненциальному закону). Вместо этого, пока закон Мура продолжает выполняться, следует ожидать, что увеличение

числа транзисторов в ЦП перевоплотится, например, в увеличение числа ядер процессора, как это и произошло в Pentium D.

Pentium D построен по микроархитектуре NetBurst, как и все модели Pentium 4. Pentium D стал первым двухъядерным процессором, предназначенным для ПК архитектуры x86, хотя и не первым двухъядерным процессором этой архитектуры, — незадолго до этого AMD выпустила двухъядерные ЦП Opteron. Двухъядерные процессоры других архитектур существовали и ранее, например IBM PowerPC-970MP.

Smithfield (26 мая 2005 г.). По существу, ядро ЦП Smithfield — пара кристаллов Prescott 1M (90 нм), связанных вместе (рис. 3.22, *г*). Каждое ядро имеет собственную кэш-память L1 (16 Кбайт для данных + 12 тысяч операций), а также L2 (1024 Кбайт), к которому может обратиться другое ядро через специальную интерфейсную шину. Площадь кристалла — 206 мм^2, 230 млн транзисторов. Максимальная TDP — 130 Вт, номинальное напряжение питания 1,4 В.

Даже при том, что новые двухъядерные чипы используют то же самое гнездо LGA775 как Pentium 4 Prescott, необходима новая системная плата, чтобы разместить их, поскольку они требуют поддержки от чипсета платы. Могут быть использованы более новые чипсеты 945 и 955X, ранее известные под кодовыми названиями «Lakeport» и «Glenwood» соответственно. Они по существу обеспечивают те же самые возможности, что и более ранние 915 и 925X, плюс поддержка двухъядерных процессоров. Первый — для использования с основным процессором Smithfield — разделяется на две версии: 945P Express и поддерживающий графику 945G Express — в то время как второй предназначен для процессоров Extreme Edition.

Первые чипы Pentium D, представленные в мае 2005 г. были построены на технологии Intel 90 нм и имели номера моделей в ряду 800. В 2005 г. вышли три чипа Pentium D Smithfield, модели: 805 (2,66 ГГц), 820 (2,8 ГГц), 830 (3,0 ГГц), 840 (3,2 ГГц). Эффективная частота системной шины (FSB) — 800 МГц (для моделей 820, 830, 840), 533 МГц (для модели 805).

Presler. В январе 2006 г. был выпущен образец Pentium D с номерами 900 и кодовым наименование «Presler», изготовленный на технологическом процессе Intel 65 нм.

Чипы Presler включают пару ядер Cedar Mill. Однако, в отличие от предыдущего Pentium D Smithfield, здесь два ядра физически разделены. Включение двух дискретных кристаллов в

единый пакет обеспечивает гибкость производства, позволяя использовать тот же самый кристалл как для одноядерного Cedar Mill, так и для двухъядерного ЦП Presler. Кроме того, производственные расходы улучшаются, поскольку при обнаружении дефекта выбраковывается только один кристалл, а не двухъядерный пакет (см. рис. 3.22).

Новая технология позволила увеличить не только тактовую частоту, но также и число транзисторов на кристалле. Как следствие, Presler имеет 376 млн транзисторов сравнительно с 230 млн для Smithfield. В то же самое время размер кристалла был уменьшен с 206 до 162 мм^2.

Размер кэша L1 (для каждого ядра) — 16 Кбайт (для данных) + + 12 тысяч операций, кэша L2 (для каждого ядра) — 2048 Кбайт. Эффективная частота системной шины (FSB) 800 МГц, номинальное напряжение питания 1,25—1,4 В. Максимальная TDP 130 Вт, разъем — LGA775.

Размещение нескольких ядер центрального ЦП на одном кристалле имеет то преимущество, что кэш-память может работать при намного более высокой частоте.

Следует иметь в виду, что Pentium D сможет эффективно реализовать свои преимущества только при выполнении приложений, которые были написаны специально для многоядерных ЦП или мультипроцессорных систем, например, обработка трехмерных изображений или видеокодирование (декодирование) или же в ситуациях, если пользователь выполняет программу с интенсивным использованием ЦП, которая позволяет каждому ядру обрабатывать часть команд приложения. Так как большинство офисных приложений и игр по состоянию на 2005 г. обычно используют единственную «нить» (подпроцесс, поток), для них Pentium D не будет существенно отличаться от старших ЦП Pentium 4, работающих на той же частоте.

Модели процессоров — 920 (2,8 ГГц), 930 (3,0 ГГц), 940 (3,2 ГГц), 950 (3,4 ГГц), 960 (3,6 ГГц).

Считается, что Pentium D был последним процессором, несущим фирменный знак Pentium — основного изделия Intel, начиная с 1993 г.

Микроархитектура Intel Core. На Форуме Развития (Intel Development Forum) в марте 2006 г. Intel обнародовала подробности новой микроархитектуры Intel Core, являющейся преемником NetBurst и мобильных процессоров Pentium M, которая рассматривается в качестве основы для планируемых к выпуску

многоядерных процессоров, ориентированных на серверные, настольные и мобильные компьютеры. Микроархитектура Intel Core была разработана той же группой инженеров Intel, которая спроектировала высокоэкономичные процессоры Pentium M.

На самом деле, первые процессоры Intel Core фактически появились в январе 2006 г. в составе технологий мобильных ЦП Centrino Duo. Процессор Core Duo (предварительное наименование «Yonah»), выпущенный тогда, был первым мобильным двухъядерным процессором Intel, основывавшимся на технологии 65 нм. Новая микроархитектура базируется на модифицированной версии ядра Yonah (рис. 3.24).

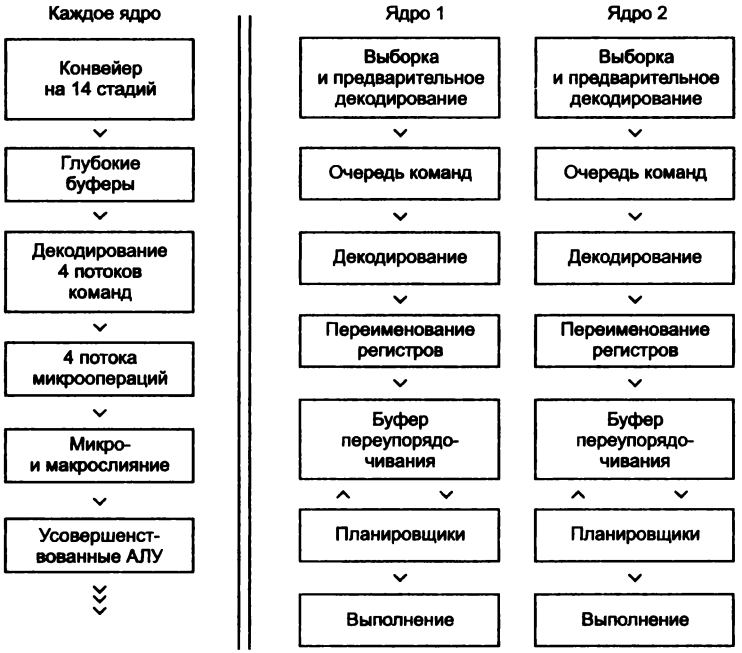


Рис. 3.24. Процессоры Intel Core

Основываясь на принципах энергетической экономичности (важнейшая черта мобильных процессоров Intel Pentium M) и существующих технологиях Intel Pentium IV, многоядерная архитектура также включает ряд важных усовершенствований:

- расширенное динамическое выполнение (Intel Wide Dynamic Execution) позволяет каждому «широ-

кому» ядру выполнять до четырех полных команд одновременно, используя эффективный 14-стадийный конвейер — спекулятивное выполнение с изменением порядка инструкций, усовершенствованный алгоритм предсказания переходов, уменьшающий количество неверных предсказаний;

- интеллектуальное управление электропитанием (Intel Intelligent Power Capability);
- интеллектуальное управление кэш-памятью (Intel Advanced Smart Cache) — совместное использование L2-кэша, что позволяет уменьшить потребление мощности путем переключения трафика между модулями памяти ядер процессора; например, если одно из ядер не занято, второе может использовать полный (двойной) кэш;
- интеллектуальный доступ к памяти (Intel Smart Memory Access) — еще одна особенность, которая улучшает системную производительность, уменьшая латентность памяти и таким образом улучшая скорость передачи данных к подсистеме памяти;
- улучшенная цифровая обработка мультимедиа (Intel Advanced Digital Media Boost) — теперь многие из 128-битовых команд SSE, SSE 2 и SSE3 смогут выполняться в пределах только одного цикла процессора. Это фактически удваивает скорость выполнения этих команд, которые широко используются в мультимедийных и графических приложениях.

Кроме того, Intel Core также предусматривает микрослияние (micro-op fusion) и макрослияние (macrofusion) — см. рис. 3.6.

Yonah — первоначально был кодовым наименованием первого поколения процессоров Intel, изготовленных по технологии 65 нм для мобильных систем, основанных на Banias/Dothan микроархитектуре Pentium M, включая технологию защиты LaGrande. Эффективность потоковой обработки (SIMD) была повышена посредством добавления команд SSE3 и улучшения реализации наборов команд SSE и SSE2, поскольку ранее выполнение целочисленных операций замедлялось повышенной латентностью кэша. Кроме того, Yonah предполагает поддержку NX bit.

Процессор Intel Core Duo состоит из двух ядер на одном кристалле, включает совместно используемый L2-кэш объема 2 MiB. Предусматривается также отключение одного из ядер при снижении нагрузки для уменьшения энергопотребления.

Intel Core Solo (процессоры предназначаются для мобильных ПК) размещается на таком же двухъядерном кристалле, что и Core Duo, но задействует только одно активное ядро. Здесь обычно используются кристаллы с одним дефектным ядром, кроме того, вообще оказывается дешевле устанавливать двухъядерный кристалл с одним отключенным ядром, нежели налаживать отдельный выпуск одноядерных кристаллов. Intel уже использовала ранее такую стратегию в процессорах 486, где процессор 486SX представлял собой 486DX, в котором блок операций с ПЗ не прошел приемный контроль и потому был отключен.

Процессоры Core Duo. Core Duo содержит 151 млн транзисторов, с учетом совместного L2-кэша на 2 MiB. Исполнительное ядро Yonah содержит конвейер на 12 стадий, предполагаемая максимальная частота 2,33—2,50 ГГц. Коммуникации между кэшем L2 и обоими ядрами обрабатываются арбитражным блоком, который контролирует доступ как к L2-кэшу, так и к первичной шине (FSB, см. рис. 4.13).

Процессоры Core не лишены и ряда недостатков:

- относительно большие значения задержек при обращении к памяти, поскольку весь трафик традиционно проходит через контроллер памяти чипсета (NorthBridge). В то же время, например, в процессорах AMD K8, этот контроллер внедрен в процессор;
- невысокая эффективность операций с ПЗ, поскольку в каждом ядре небольшое количество исполнительных блоков ПЗ;
- отсутствует поддержка 64-разрядной системы команд (EM64T);
- такая же или даже более низкая приведенная эффективность («на 1 ватт») для однопроцессных приложений (сравнительно с предшествующими ЦП).

Sossaman — ЦП, производный от Yonah, был выпущен 14 марта 2006 г. как двухъядерный Xeon (Dual-Core Xeon LV). Sossaman почти идентичен Yonah, за исключением поддержки двухразъемной конфигурации (dual-socket configurations), для 4-х ядер, кроме того, здесь реализована 36-разрядная адресация (PAE mode). ЦП Sossaman предназначен для серверов, однако как и Yonah, не поддерживает EM64T, что для рыночного сектора серверов является серьезным недостатком.

В то время как первые процессоры Intel Core планировались исключительно для мобильных систем, объявленные Intel вес-

ной 2006 г. для выпуска ЦП ориентированы на все секторы рынка. Вот их кодовые имена:

- «Merom» — мобильные системы (35—40 Вт);
- «Conroe» — рабочие станции (65—80 Вт);
- «Woodcrest» — серверы (40—80 Вт).

Таким образом, новая архитектура осуществила воссоединение настольных и мобильных линий в изделиях Intel.

Процессоры Core 2 являются 8-м поколением процессоров, производимых фирмой Intel. Первые образцы Core 2 были официально выпущены 27 июля 2006 г. Как и ЦП Intel Core, которые они должны заместить, Core 2 включает Duo (двухъядерные) и Solo (одноядерные) модели. Новые линии продуктов включают также модели Extreme (высокопроизводительный сектор) и Quad (четырехъядерные). Основные процессоры имеют кодовые имена «Conroe» (для настольных ПК) и «Merom» (портативные модели), их варианты именуются «Kentsfield» (четырехъядерные Conroe) и «Penryn» (45 нм Merom). Хотя серверные процессоры «Woodcrest» также базируются на микроархитектуре Core, они фигурируют на рынке под маркой «Xeon», а не «Core 2».

Рассмотрим процессы в ЦП Conroe (рис. 3.25), который является «прямым наследником» мобильного процессора Pentium M и «дальним» процессоров архитектуры P6 (см. рис. 3.18), отличаясь от них разрядностью шин и количеством исполнительных устройств:

- поскольку это процессор гарвардской архитектуры (раздельные кэши первого уровня — кэш команд или I-cache и кэш данных или D-cache), выполнение программы начинается с того, что соответствующие команды попадают в I-cache;
- из I-cache команды, до которых дошел процессор, извлекаются в блок выборки команд (Instruction Fetch Unit). Этому блоку приходится отслеживать ход выполнения программы с учетом условных и безусловных переходов. В случае безусловного перехода он переключается на новое место в кэше, в случае же условного перехода, пользуясь данными блока предсказания переходов (Branch Prediction Unit — BPU), определяет, по какому пути пойдет дальше программа;
- взятый блоком выборки участок памяти делится на отдельные x86-команды, которые помещаются в очередь блока выбранных команд (Instruction Fetch Buffer), откуда затем передаются на декодеры команд (Instruction Decoders), где

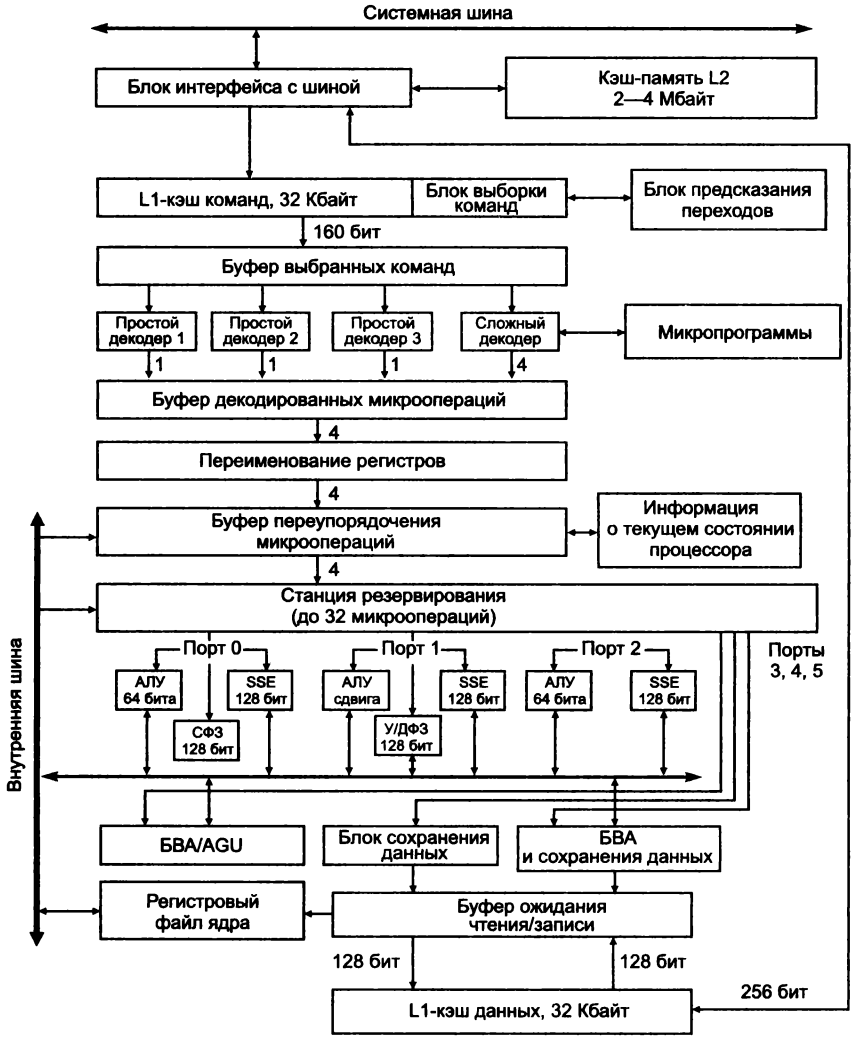


Рис. 3.25. Микроархитектура процессоров Conroe

переводятся в микрооперации (МкОП). Элементарными командами, которые удается конвертировать в одну МкОП, занимаются простые декодеры, остальными — сложный декодер, подставляющий вместо x86-команд последовательности из нескольких МкОП, при необходимости заимствуя

их из специальной области памяти — микропрограмм процессора;

- полученные из x86-кода МкОП проходят еще одну очередь, призванную сгладить неравномерность декодирования команд, и попадают в блок переименования регистров, в нем МкОП выдается разрешение на пользование теми или иными ресурсами процессора. В частности, определяется, из какого внутреннего регистра МкОП должна брать необходимые ей данные и куда их сохранять. Получив необходимые допуски, МкОП попадают в буфер внеочередного выполнения (ReOrder Buffer — ROB);
- буфер представляет собой таблицу, в которой указано, какие команды сейчас выполняются процессором, какие у них исходные данные и какие получаются результаты. По мере выполнения команди соответствующие записи в таблице обновляются. Когда результат становится известен, ROB «публикует» его, соответственно изменяя видимые пользовательской программе регистры процессора или разрешая записать вычисленное число в оперативную память, после этого стирает информацию о команде из таблицы;
- сделав отметку в ROB, МкОП передается на станцию резервирования (Reservation Station). Здесь МкОП стоят в очередях, ожидая, пока не дойдут все необходимые им данные (пока не завершится операция чтения из оперативной памяти или не будет вычислена предыдущая команда) и не освободится подходящее им исполнительное устройство процессора. Причем порядок, в котором они поступили в Reservation Station, не играет роли. Помимо возможности внеочередного выполнения кода, позволяющей во время непредвиденного простоя занять процессор каким-либо другим делом, Reservation Station помогает решить и другую задачу: благодаря тому, что к ней подключено сразу несколько функциональных устройств, на выполнение могут быть запущены до 5—6 МкОП одновременно;
- пройдя очередь в Reservation Station, МкОП отправляется вместе с готовыми данными на выполнение в одно из соответствующих исполнительных устройств — это один из самых коротких этапов конвейера: обычно он занимает один такт (на выполнение команды требуется свыше 14 тактов), в сложнейших случаях — от 4 до 7 тактов. Полученный результат вычислений возвращается в ROB, где дожидается

либо своей очереди на выписку, либо возникновения ошибки в ходе вычисления очередной МкОП, а такой сбой приводит, как правило, к сбросу конвейера со всеми ранее полученными результатами.

В отличие от ЦП архитектуры NetBurst (таких как the Pentium IV и Pentium D), дизайн Core 2 не основывается на повышении эффективности исключительно за счет достижения максимальных тактовых частот, а использует другие возможности, включая улучшение кэш-памяти и увеличение числа ядер. Энергопотребление таких процессоров существенно ниже, чем для настольных процессоров Pentium (65 Вт для Core 2 сравнительно с 130 Вт для Pentium 4 Prescott).

Процессоры Intel Core 2 поддерживают такие опции, как EM64T, Virtualization Technology, Execute Disable Bit и SSE3. Кроме того, Core 2 вводит технологии LaGrande Technology, Enhanced SpeedStep Technology и Active Management Technology (iAMT2).

Conroe. Процессоры выпускаются по технологии 65 нм и ориентированы на настольные ПК, заменяя Pentium 4 и Pentium D. Intel заявляет, что Conroe демонстрирует 40%-ное повышение эффективности при 40%-ном снижении энергопотребления сравнительно с Pentium D. Все ЦП Conroe имеют кэш L2 объемом 4 Мбайт, однако у моделей E6300 и E6400 половина кэша отключена (только 2 Мбайта используемого кэша L2).

Более дешевые ЦП моделей E6300 (1,86 ГГц) и E6400 (2,13 ГГц), оба с первичной шиной (FSB) на 1066 MT/с были выпущены первыми. Традиционно процессоры одного ряда с меньшим кэшем просто представляют собой экземпляры, не прошедшие полный приемный контроль, в которых часть кэша поэтому отключена.

Следует отметить, что процессоры Conroe имеют высокий резерв по тактовой частоте — модель на 1,86 ГГц вполне может быть «разогнана» (overclocked) до значений более чем 3,0 ГГц при установке на качественной системной плате, которая поддерживает высокую скорость FSB.

Модели более высокого ряда (E6600 и E6700 Core 2 Duo) имеют заявленную тактовую частоту в 2,4 и 2,67 ГГц соответственно. Они имеют 1066 MT/s первичную шину, 4 Мбайта разделяемого кэша L2 и электрическую мощность 65 Вт. Проведенные тесты показали, что E6700 и E6600 стабильно работают при частотах до 4 ГГц (при воздушном охлаждении) и даже при 5,4 ГГц (при охлаждении жидким азотом).

Предполагается, что процессоры с номерами, оканчивающиеся на «50» будут иметь FSB на 1333 МГц.

Conroe XE (Core 2 Extreme) предназначен для замены процессоров Pentium 4 Extreme Edition и двухъядерного Pentium Extreme Edition. Core 2 Extreme заявлен с частотой 2,93 ГГц и FSB на 1066 MT/s, электрическая мощность 75—80 Вт и при полной нагрузке температура не превышает 45 °C, а использование технологий SpeedStep позволяет почти уравнять среднюю температуре ЦП с окружающей атмосферой.

Как и Core 2 Duo, он имеет 4 Мбайта разделяемого кэша L2. «Разгон» процессора показал, что Core 2 Extreme работает при 3,6 ГГц при штатном охлаждении и без повышения напряжения питания, 4,1 ГГц требует усиленного воздушного охлаждения и повышения напряжения, более 5,5 ГГц — только при охлаждении жидким азотом.

Merom — первый процессор Core 2 для мобильных систем, предполагается, что его энергетическая эффективность будет на 20 % выше, чем у мобильных Core Duo (Yonah). Объявлена мощность в 35 Вт для стандартной версии и 5 Вт для версии с очень низким напряжением питания (Ultra Low Voltage — ULV).

Merom является первым мобильным процессором Intel, в котором реализовано 64-разрядное расширение EM64T 64-bit extensions, скорость системной шины — 667 MT/s. Следующая версия предполагает FSB на 800 MT/s и новый интерфейс Socket P.

Процессоры Merom имеют номера «T5x00» и «T7x00» (для Core 2 Duo), причем T5200 заявлен с частотой 1,60 ГГц; T5500 — 1,66; T5600 — 1,83; T7200 — 2,0; T7400 — 2,16, а T7600 — с частотой 2,33 ГГц. Модели T5x00 выпускаются с общим кэшем L2 на 2 Мбайта, а T7x00 — на 4 Мбайта.

Intel Penryn Core 2. Процессор построен по архитектуре Core 2, но является первым в серии, выпущенным по технологии 45 нм с использованием инновационной технологии транзисторов с металлическим затвором и диэлектрическим изолятором (см. рис. 1.22).

Предполагается, что он заменит ряд моделей Merom, в том числе версии на два или четыре ядра в одном кристалле.

В частности, четырехядерные процессоры Intel Core Quad включают 582 млн транзисторов, тогда как четырехъядерный Penryn — 820 млн на меньшей площади кристалла. Такое повышение числа транзисторов связано с реализацией в каждом ядре кэша L2 на 6 Мбайт (150 % по отношению к Core 2) — рис. 3.26.

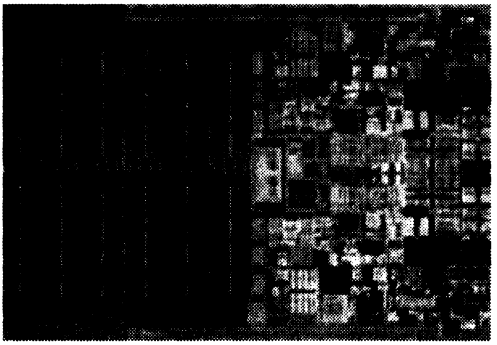


Рис. 3.26. Микроснимок двухъядерного процессора Penryn

Другие особенности:

- Penryn использует новый более быстрый блок деления, который обрабатывает 4 бита в такт вместо 2 бит для Conroe. Это убыстряет в 2 раза операции деления, извлечение корня осуществляется быстрее в 4 раза;
- используется новый режим управления питанием — Deep Power Down, в котором предусмотрено состояние ЦП Deep sleep (тактовый генератор останавливается, кэш-память отключена и напряжение питания падает до минимума);
- в то время как в процессорах Core 2 Duo введена концепция независимого «засыпания» ядер, когда одно из ядер работает в полную нагрузку, второе является «горячим резервом», в Penryn сделан следующий шаг — недоиспользуемое ядро «жертвует» свою частоту и мощность питания полностью загруженному ядру;
- наконец, здесь внедрены «Penryn Instructions», или набор команд SSE4, добавляющий новые возможности мультимедиа-обработки, позволяя повысить на 40 % производительность тех программ, которые будут его активно использовать.

Kentsfield. ЦП Kentsfield, выпущенный в ноябре 2006 г., был первым четырехъядерным (quad core CPU) процессором Intel Core 2, предназначенным для настольных ПК. Могут быть выделены более мощные модели (Core 2 Extreme) и обычные образцы (Core 2 Quad). Все они укомплектованы двумя кэшами L2 по 4 Мбайт. Серия Core 2 Quad Q6600, с частотой 2,4 ГГц, начала выпускаться в январе 2007 г., а модели Extreme QX6850 Kentsfield — в июле 2007 г.

Аналогично ЦП Pentium D, процессор Kentsfield конструктивно включает два раздельных кремниевых кристалла (каждый эквивалентен одному Core 2 duo), размещенных на отдельном керамическом мультипроцессорном модуле (MCM). Это оказывается более дешевым решением, однако при этом снижается скорость обмена данными между ЦП и «Северным мостом» чипсета, сравнительно с раздельным размещением ЦП, как это реализовано, например, в AMD Quad FX platform. Максимальная рассеиваемая мощность достигает от 95 до 130 Вт, что примерно вдвое выше, чем для Core 2 Duo с аналогичной частотой.

Многоядерные ЦП Kentsfield наиболее подходят для таких приложений, которые легко расщепляются на несколько параллельных потоков обработки (например, аудио-/видеокодирование, сжатие данных, трехмерная графика).

Yorkfield представляет собой четырехъядерный ЦП на двух кремниевых кристаллах, с двумя общими блоками кэша L2 по 6 Мбайт. Собственно, кэш-памятью занята большая часть площади ядра. Это уже не первый раз, когда Intel использует возможности новых техпроцессов по уплотнению транзисторов в ядре одного и того же размера для наращивания объемов кэша, что вполне оправдано, поскольку многие приложения повышают эффективность при увеличение L2. Предусмотрены также версии с блоками кэша по 3 Мбайта.

Wolfdale — кодовое название серий E5000 Pentium Dual Core и E7000/E8000 серий Core 2 Duo ЦП для настольных ПК, которые аналогичны Penryn и Yorkfield XE и должны сменить изделия Conroe. Выпуск был начат в январе 2008 г. ЦП изготовляются по технологии 45 нм и на одном кристалле размещаются два ядра ЦП. Модель E7200, работающая на частоте 2,53 ГГц, имеет кэш L2 на 3 Мбайта и поддерживает скорость FSB (первичная процессорная шина) в 1066 MT/c (миллионов передач в секунду), остальные модели, работающие на частотах 2,66—3,33 ГГц, имеют 6 Мбайт общего кэша L2 и скорость FSB в 1333 MT/c. Изделия серии E5200 работают на частоте 2,5 ГГц, имеют кэш L2 на 2 Гбайта и предназначены для замены процессоров Pentium Dual core.

В июне 2009 г. компания объявила об упразднении многообразия вариантов данной торговой марки (например, Core 2 Duo, Core 2 Quad, Core 2 Extreme) в пользу трёх ключевых наименований: Core i3, Core i5 и Core i7.

Nehalem — микроархитектура процессоров компании Intel, представленная для ядра Bloomfield в исполнении LGA 1366 и

для ядра Lynnfield в исполнении LGA 1156. Микропроцессоры продаются под торговой маркой Core i7 и Core i5 соответственно.

Процессоры Nehalem содержат не менее 731 млн транзисторов, что на 10 % меньше, чем у процессоров Yorkfield. Но площадь кристалла значительно увеличилась по сравнению с предшественником — с 214 до 263 мм^2.

Архитектура Nehalem построена на базе Core, но содержит такие кардинальные изменения, как:

- встроенный контроллер памяти, поддерживающий 2 или 3 канала DDR3 SDRAM или 4 канала FB-DIMM;
- новая шина QPI, пришедшая на смену шине FSB (только в процессорах для LGA 1366; процессоры для LGA 1156 используют шину DMI);
- возможность выпуска процессоров со встроенным графическим процессором (в бюджетных решениях на базе двухядерных процессоров);
- в отличие от Kentsfield и Yorkfield, которые состоят из двух кристаллов по 2 ядра на каждом, все 4 ядра Bloomfield находятся на одном кристалле;
- кэш 3-го уровня;
- поддержка SMT (организация 2-х логических ядер из 1 физического).

Первые процессоры Nehalem основаны на том же 45-нм техпроцессе, что и Penryn.

Микроархитектура Intel Atom. Intel Atom — торговое название линейки ЦП от Intel, которые поддерживают системы команд x86 и x86-64 CPUs, изготовляются по технологии 45 нм CMOS (high-k metal gate, см. рис. 1.22, *б*) и предназначены для использования в ультрамобильных ПК, смартфонах и других портативных экономичных изделиях (рис. 3.27).

В апреле 2008 г. Intel официально объявила, что процессоры с кодовыми именами Silverthorne (серия Atom Z, процессоры Z500—Z540) и Diamondville (Atom N, изделия N270 и 230) базируются на одной микроархитектуре. Более дорогой энергосберегающий ЦП Silverthorne предполагается использовать в мобильных Internet-устройствах (Mobile Internet Devices — MID), в то время как Diamondville — в таких изделиях, как настольные системы (Nettops) и бюджетные ноутбуки (Netbooks).

Intel и Lenovo, в частности, совместно объявили о MID на основе Atom под названием IdeaPad U8, который весит менее 280 г, имеет 4,8" сенсорный экран и обеспечивает как лучшую порта-

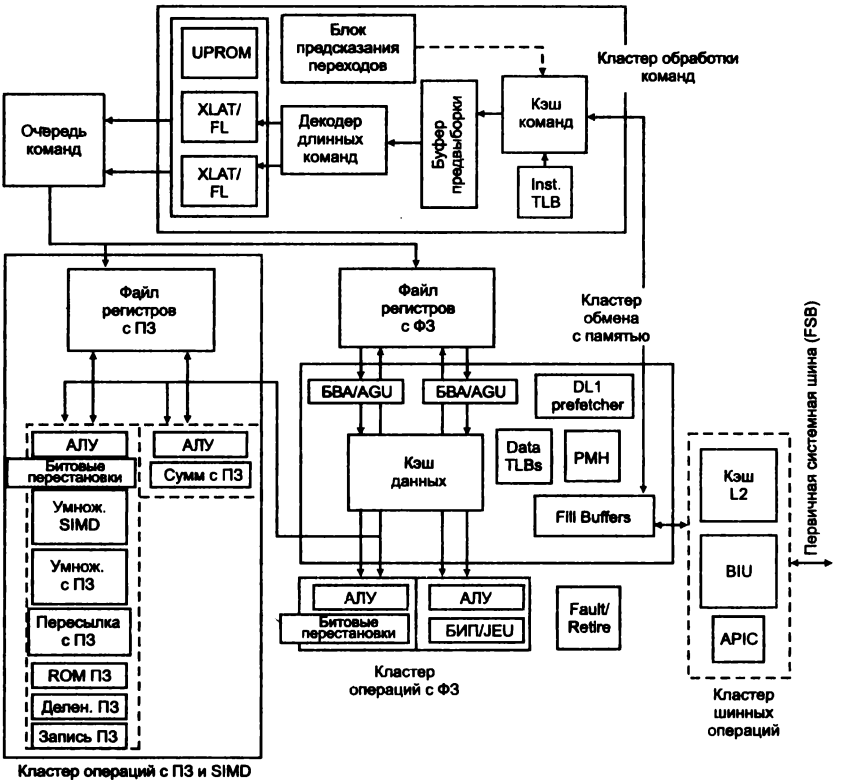


Рис. 3.27. Микроархитектура Intel Atom

тивность, чем типичный «Netbook», так и лучшую эффективность в Internet, чем мобильный телефон или карманный ПК (PDA).

Процессоры Atom имеют кэш L1 объемом 56 Кбайт, из которых 32 отведено под кэш инструкций, а 24 — под данные. Процессоры могут исполнять 32-битовый код и поддерживают наборы команд расширений MMX, SSE, SSE2, SSE3 и SSSE3. Что касается 64-битового кода (x86-64), то его поддерживает только ядро Diamondville и только в модели Atom 230. В основном процессоры Atom являются одноядерными, однако они поддерживают технологию многопоточности (Hyper-Threading), которая позволяет исполнять два параллельных потока команд.

В сентябре 2008 г. Intel объявила о выпуске двухъядерного ЦП данной архитектуры (неофициальное кодовое имя — Dual Diamondville) под маркой Atom 330, который работает на частоте

1,6 ГГц и потребляет 8 Вт. Частота FSB — 533 МГц, на каждое из ядер приходится по 512 Кбайт кэша L2.

Intel Atom выполняет две команды на цикл (подобно Pentium в 1993 г.). Как и в других ЦП, команда x86 расщепляется на более простые операции перед исполнением, но гораздо в меньшей степени, чем это происходит в ЦП Intel P6 или Intel P68. В ЦП Atom внутренние микрооперации (МкОП) выполняют как считывание, так и запись в память в сочетании с операциями АЛУ. Это ближе к уровню x86 и оказывается более эффективным, чем МкОП в предшествующих проектах. При этом ПЦ обходится парой АЛУ и не использует таких механизмов, как внеочередное исполнение, опережающее исполнение, и ротация регистров.

Процессоры Xeon и Celeron. Перечисленные выше архитектуры ЦП Intel имеют следующие модификации, предназначенные как для серверов и рабочих станций (Xeon), так и для недорогих настольных ПК (Celeron).

Процессоры Xeon. В июне 1998 г. Intel начала выпускать ЦП Pentium II Xeon, работающий на частоте 400 МГц. Технически Xeon представлял собой комбинацию технологий Pentium Pro и Pentium II и был разработан, чтобы предложить повышенную эффективность, требуемую в критических приложениях для рабочих станций и серверов. Используя интерфейс Slot 2, Xeon были почти вдвое больше размером, чем Pentium II, прежде всего из-за увеличенной кэш-памяти L2.

В первых образцах чип снабжался кэш-памятью L2 на 512 Кбайт или 1 Мбайт. Первый вариант был предназначен для рынка рабочих станций, второй — для серверов. Версия на 2 Мбайт вышла позже — в 1999 г.

Основное усовершенствование сравнительно с Pentium II заключалось в том, что кэш-память L2 работала на частоте ядра центрального процессора в отличие от конфигураций на основе Slot 1, которые ограничивали кэш L2 половиной частоты ЦП, что позволяло Intel использовать более дешевую память Burst SRAM в качестве кэша, вместо того, чтобы применять обычную SRAM.

Другое ограничение, которое удалось преодолеть посредством Slot 2, был «двухпроцессорный предел». При использовании архитектуры SMP (симметрический мультипроцессор) процессор Pentium II оказался неспособен поддерживать системы с более чем двумя центральными процессорами, в то время как системы, основанные на Pentium II Xeon, могли объединять четыре, восемь или более процессоров.

Вскоре после выхода Pentium III весной 1999 г. был выпущен Pentium III Xeon (кодовое имя Tanner). Это был базовый Pentium Xeon с добавлением набора команд SSE. Нацеленный на рынок серверов и рабочих станций, Pentium III Xeon первоначально выпускался на 500 МГц и с кэш-памятью L2 на 512 Кбайт (или 1,0—2,0 Мбайт). Осенью 1999 г. Xeon начал выпускаться с ядром Cascade (0,18 мкм) со скоростями, увеличивающимися от начальных 667 МГц до 1 ГГц к концу 2000 г.

Весной 2001 г. выпущен первый Xeon на основе Pentium IV со скоростями 1,4, 1,5 и 1,7 ГГц. Базирующийся на ядре Foster, он был идентичен стандарту Pentium IV, за исключением разъема microPGA Socket 603. Pentium IV Xeon поддерживался чипсетом i860, который подобен i850, однако предусматривает двухпроцессорные системы и позволяет увеличить максимальный размер памяти до 4 Гбайт. Через год была выпущена мультипроцессорная версия, позволяющая собирать 4- и 8-процессорные системы и включающая кэш-память L3 на 512 Кбайт или 1 Мбайт.

Процессоры Celeron. В попытке обратиться к сектору рынка дешевых ПК (до того времени — область AMD и Cyrix, выпускавших клоны и поддерживавших архитектуру Socket 7), в апреле 1998 г. Intel начала производство процессоров семейства Celeron.

Celeron — упрощенный вариант P2 для дешевых компьютеров. Основные отличия этих процессоров — в объеме кэша второго уровня и частоте шины. Первые выпущенные в апреле и июне 1998 г. Celeron на 266 и 300 МГц не имели кэша вообще при частоте шины 66 МГц и выполнены в конструктиве Slot 1. Следущие модели имели 128 Кбайт кэша и выпускались как для Slot 1, так и для Socket 370 (PPGA), в их названии присутствует буква A (например, Celeron 333A).

Основанный на той же самой микроархитектуре P6, как и Pentium II, и изготовляемый на основе аналогичного процесса 0,25 мкм, первоначальный Celeron предлагал законченный пакет последних технологий, включая поддержку графики AGP, интерфейс жесткого диска ATA-33, а также SDRAM и ACPI. Эти процессоры работали с любыми чипсетами Intel Pentium II, поддерживавшими системную шину на 66 МГц — включая 440LX, 440BX и новый 440EX — и определенно предназначенными для основного рынка ПК.

Covington. Первые ЦП Celeron (Covington) по существу представляли собой Pentium II, изготовленные без кэша L2.

В результате, имея частоту в 266 или 300 МГц (что существенно выше, чем для Pentium MMX), ЦП Celeron были намного медленнее чем части, для замены которых они были предназначены.

В отличие от Pentium II с его упаковкой Single Edge Cartridge (SEC), начальный Celeron не имел никаких защитных пластмассовых оболочек вокруг карты процессора, который Intel назвал Single Edge Processor Package (SEPP) и который был совместим со Slot 1, что позволяло использовать существующие системные платы (см. рис. 1.23, *б*).

Mendocino. Начиная с частоты 300, все ЦП Celeron начали оборудоваться кэш-памятью L2 объемом 128 Кбайт, расположенной на кристалле, работающей на полной скорости центрального процессора и поддерживающей внешнюю связь через шину 66 МГц.

Процессоры Celeron от 300 до 466 МГц выпускались в двух версиях — SEPP и PPGA (plastic pin grid array). Первый рассматривался как господствующая версия (совместимый с существующей архитектурой Slot 1), в то время как последний не был совместим ни с Socket 370, ни Socket 7 или Slot 1.

Coppermine. Весной 2000 г. общая картина процессорных конфигураций и интерфейсов усложнилась с появлением первых процессоров Celeron, основанных на ядре Pentium III Coppermine (0,18 мкм). Они производились, используя еще один форм-фактор — дешевую упаковку FC-PGA (flip-chip pin grid array). Это, казалось, указывало на «начало конца» как PPGA, так и Slot 1, с последующим выпуском чипов Pentium III, также поддерживающих FC-PGA. Процессоры Pentium III в FC-PGA имели два разъема RESET (Сброс) и требовали спецификаций VRM 8.4. В дальнейшем существующие системные платы на Slot 370 стали называть «платами наследства», в то время как новые платы на Slot 370, поддерживающие новый форм-фактор FC-PGA, — «гибкими системными платами».

Первые ЦП Celeron, основанные на Coppermine, имели частоту 566 МГц, затем было произведено множество приращений вплоть до 766 МГц, и в начале 2001 г. вышла версия на 800 МГц, позволяющая использовать FSB на 100 МГц. К моменту выхода последнего процессора Celeron на базе Coppermine (осень 2001 г.) скорости достигли 1,1 ГГц.

Celeron Tualatin. Первый ЦП Celeron, использовавший ядро Tualatin Intel по технологии 0,13 мкм, был выпущен в начале 2002 г. и показывал частоту 1,2 ГГц. Предполагалось, что по-

тенциал Tualatin сможет продвинуть семейство Celeron к частоте 133 МГц для FSB, однако даже версия на 1,3 МГц ограничивалась частотой FSB в 100 МГц и использованием модулей памяти стандарта PC100.

В дальнейшем стало ясно, что для преодоления этого узкого места FSB необходима частота не менее чем в 1,5 ГГц. Поэтому был произведен переход к процессорам Celeron на базе ядра Pentium IV Willamette, обеспечивающего частоту 1,8 ГГц и более.

Процессоры Celeron класса NetBurst. Первый Celeron на базе Pentium IV был выпущен в мае 2002 г. Базирующиеся на ядре Willamette (0,18 мкм) и часто называемые Celeron 4, они имели кэш L2 128 Кбайт, а не 256 или 512 Кбайт, но в остальном очень походили на прототип. Их эффективность значительно пострадала по причине меньших размеров кэшей и вообще это первое поколение (Celeron 4) не было хорошо принято рынком.

К концу 2002 г. Celeron был модернизирован с использованием ядра Northwood (0,13 мкм), позволяя достигнуть частоты 2,0 ГГц (хотя Northwood и использует L2-кэш на 512 Кбайт, Intel ограничилась кэшем 128 Кбайт для ЦП Celeron).

Последний и самый быстрый Celeron, основанный на Northwood, был выпущен весной 2004 г. и имел частоту 2,8 ГГц.

Процессоры Cyrix

Длительное время компания Intel занимала комфортабельное положение в качестве основного производителя процессоров для ПК. Начиная с выпуска процессоров серии 486 (1989 г.), компания Cyrix вместе с «сотоварищем», также давним производителем клонов Intel, компанией Advanced Micro Devices (AMD), представляли наиболее серьезную угрозу доминированию Intel.

В начале 1990-х гг. AMD и Cyrix выпускали собственные версии 486DX, но из их продуктов наиболее известны клоны 486DX, первый — копия 486DX2-66 (представленный Intel в 1992 г.) и второй — повышающий до 80 МГц внутреннюю скорость. В основе 486DX2-80 была системная шина на 40 МГц, и в отличие от чипов Intel DX2, которые работали на напряжении 5 В, он использовал 3 В. AMD и Cyrix позже предложили версии своих 40 МГц 80486 процессоров с тройным коэффициентом умножения частоты (множителем), которые работали при 120 МГц. AMD

и Cyrix предложили функцию управления мощностью, начиная с процессоров с двойным множителем, которую Intel в итоге использовала в своем DX4, запущенном несколько лет спустя.

Хотя Intel прекратила выпускать 486 после DX4-100, AMD и Cyrix продолжали его развитие. В 1995 г. AMD предложила четырехкратный множитель 5x86, 33 МГц 486DX, который работал с внутренней частотой 133 МГц. AMD продвигала на рынке этот чип как сопоставимый с новым Intel Pentium 75, и поэтому компания назвала его 5x86-75. Но это был процессор 486DX во всех отношениях, включая к тому же кэш первого уровня (встроенный в процессор) на 16 Кбайт, который Intel применяла, начиная с DX4. Cyrix продолжала работать со своим собственным 5x86, названным M1sc, но этот чип сильно отличался от чипа компании AMD. Фактически, M1sc предложил характеристики, подобные Pentium, несмотря на то, что он был разработан для использования в системных платах для i80486. Работая на частотах 100 МГц и 120 МГц, чип включал 64-битовую внутреннюю шину, шестистадийный конвейер (против пятистадийного у DX4) и технологию предсказания переходов.

Важно помнить, однако, что Cyrix 5x86 появился после того, как Intel внедрила Pentium, так что эти особенности были более полезны в модернизации 486-го, чем для применения в новых системах.

В середине 1999 г. произошли важные события на рынках высоких информационных технологий. В августе Cyrix наконец-то покинула рынок процессоров для настольных ПК, когда National Semiconductor продала права на ее процессоры 80x86 основанному в Тайване производителю чипов VIA Technologies. Высоко интегрированная номенклатура продуктов MediaGX осталась за National Semiconductor, чтобы быть частью нового семейства Geode решений «компьютер-на-чипе», — компании, развивающейся на рынке клиентских устройств.

Другим важным фактом был запуск компанией AMD нового процессора Athlon (раннее кодовое название «K7») и захват технологического превосходства над Intel, которая была вынуждена объявить о задержке выпуска его 0,18-микронного Pentium III «Coppermine».

Cyrix 6x86. Обнародованный в октябре 1995 г., 6x86 был первым совместимым с Pentium процессором, позволявшим проникнуть на рынок и добиться сотрудничества с IBM Microelectronics Division (рис. 3.28). Распространение 6x86 первона-

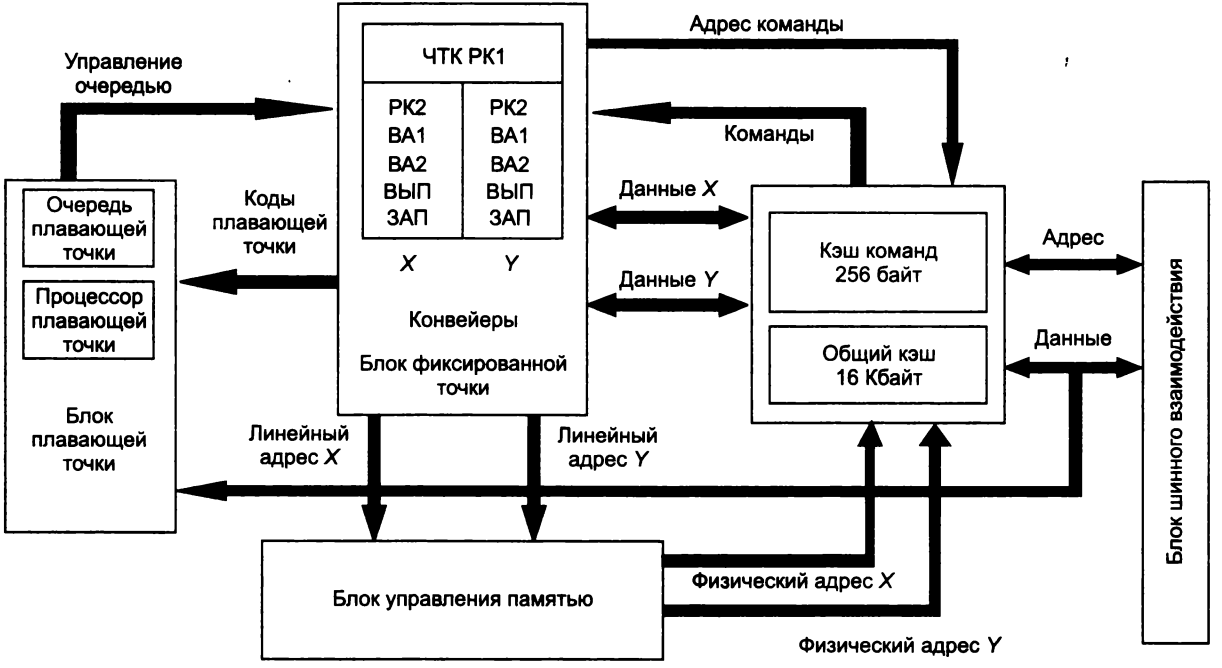


Рис. 3.28. Схема функционирования процессора Cyrix 6x86:
ЧТК — чтение команды; РК1 — расшифровка команды, стадия 1; РК2, стадия 2; ВА1 — вычисление адреса, стадия 1; ВА2, стадия 2; ВЫП — выполнение; ЗАП — запись результата

чально было медленным, поскольку Cyrix установила слишком высокие цены, ошибочно думая, что, так как эффективность процессора была сопоставима с Intel, его цена могла быть такой же. Как только Cyrix пересмотрела свои позиции, чип стал оказывать значительное влияние в доле соответствующего сектора рынка как высокоэффективная альтернатива серии Pentium.

Начиная с 6x86 процессоры Cyrix были способны к уровню производительности, эквивалентному чипу Pentium, но при более низкой частоте. Для оценки производительности используется Processor Performance Rating — P-рейтинг (обозначение P100+, например, символизирует производительность, эквивалентную Pentium с частотой 100 МГц). Процессоры Cyrix (как и AMD) традиционно работают на более низких частотах, чем численное значение их P-рейтинга без заметного снижения производительности. Например, P133+ (P-рейтинг равен 133) работает на частоте 110 МГц, в то время как P150+ и P166+ работают на 120 и 133 МГц соответственно.

Превосходство 6x86 вытекало из усовершенствований архитектуры чипа, которая позволила 6x86 получать доступ к внутреннему кэшу и регистрам в одном цикле частоты (Pentium обычно задействует два или больше циклов для доступа к кэшу). Кроме того, первичный кэш 6x86-го был объединен, вместо того, чтобы включить две отдельные секции — 8 Кбайт для команд и данных. Эта объединенная модель была в состоянии хранить команды и данные в любом соотношении, обеспечивая «вероятность попадания» кэша в пределах 90 %. ЦП содержит 3,5 млн транзисторов, первоначально изготовленных по технологии пяти 0,5-микронных слоев. Интерфейс — Socket 7. Напряжение питания ядра — 3,3 В.

Характеристики 6x86 были подобны Pentium — суперскалярная архитектура, 80-битовый блок плавающей точки, 16-Кбайт кэш первого уровня и System Management Mode (SMM). Однако он имеет множество важных отличий.

Во-первых, это *суперконвейер* т. е. семь, вместо пяти, стадий (чтение команды, две стадии расшифровки, две стадии вычисления адреса, выполнение, запись результата — см. рис. 3.28).

Другие новые характеристики — удаление зависимости данных, предсказание переходов, выполнение команд вне естественного порядка (возможность более быстрым командам выходить из очереди конвейера, не нарушая процесс выполнения программы). Процессор 6x86 обращается с кодом команд без

преобразования, что полностью оптимизирует набор команд 80x86 CISC. Это используется как для 16-, так и для 32-битового кодов. Pentium также делает это, но здесь происходит преобразование CISC-команд к RISC (или микрокомандам) прежде, чем они входят в конвейеры.

Однако процессоры 6x86 сталкивались с множеством проблем, особенно перегревом, низкой производительностью при работе с плавающей запятой и несовместимостью с Windows NT. Это неблагоприятно воздействовало на успех процессора, поэтому конкуренция с Pentium оказалась недолгой и закончилась с запуском Intel Pentium MMX.

Cyrix MediaGX. Введение процессора MediaGX в феврале 1997 г. определило первую новую архитектуру PC в десятилетии и определило новый сегмент рынка — дешевый «Основной ПК». Рост этого рынка был бурным, и технология процессора Cyrix и новшество уровня системы были ключевыми элементами.

Чем больше процессов, которые обрабатываются в центральном процессоре ПК непосредственно, тем выше общая производительность системы. В традиционных компьютерных разработках центральный процессор обрабатывает данные на частоте в мегагерцы, в то время как шина, которая перемещает данные в (и от) другие компоненты, работает только на половинной (или даже меньшей) скорости. Это означает, что движение данных к (и от) центральному процессору занимает больше времени. Cyrix устранила это узкое место введением технологией MediaGX (рис. 3.29).

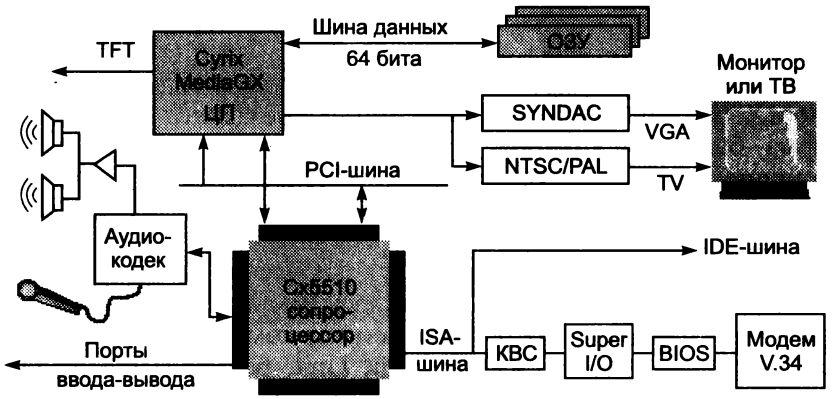


Рис. 3.29. Архитектура Cyrix MediaGX

Архитектура MediaGX объединяет в блок процессора графические и звуковые функции, интерфейс PCI и диспетчер памяти, таким образом устраняя потенциальные конфликты в системе и проблемы конфигурации конечного пользователя. Она состоит из двух чипов — процессора MediaGX и сопроцессора MediaGX Cx5510. Процессор использует особое гнездо, требующее специально разработанной материнской платы.

MediaGX — x86-совместимый процессор, который непосредственно соединяет на шине PCI и память EDO DRAM по выделенной 64-битовой шине данных. Техника сжатия, используемая на шине данных, устраняет потребность в кэше второго уровня. Есть объединенный (16 Кбайт) кэш первого уровня на центральном процессоре — того же объема, что и на стандартном чипе Pentium.

Графика обрабатывается специальным конвейером непосредственно в центральном процессоре, и контроллер монитора находится также на главном процессоре. Нет никакой видеопамяти, буфера кадров, сохраняемых в главной памяти (традиционная Unified Memory Architecture — UMA), вместо этого используется собственная Cyrix Display Compression Technology (DCT). Операции с данными VGA выполняются аппаратными средствами ЭВМ, но регистры VGA управляются программами Cyrix — Virtual System Architecture (VSA).

Сопутствующий чип MediaGX Cx5510 содержит аудиоконтроллер и также использует программы VSA, чтобы эмулировать возможности стандартных звуковых карт. Этот чип соединяет процессор MediaGX через шину PCI с шиной ISA, а также с IDE и портами ввода-вывода, т. е. выполняет традиционные функции чипсета.

Cyrix 6x86MX. Ответом Cyrix на технологию Intel MMX был 6x86MX, запущенный в середине 1997 г., незадолго до того, как фирма была приобретена компанией National Semiconductor. Компания осталась верной формату Socket 7 для своего нового чипа, это поддерживало на необходимом уровне затраты производителей системы и в конечном счете потребителей, продлевая жизнь существующего чипа и системных плат.

Архитектура нового чипа оставалась по существу той же , что и у его предшественника, с дополнением команд MMX, некоторыми улучшениями к Floating Point Unit, большим (64 Кбайт) универсальным кэшем первого уровня и расширенным блоком управления памятью.

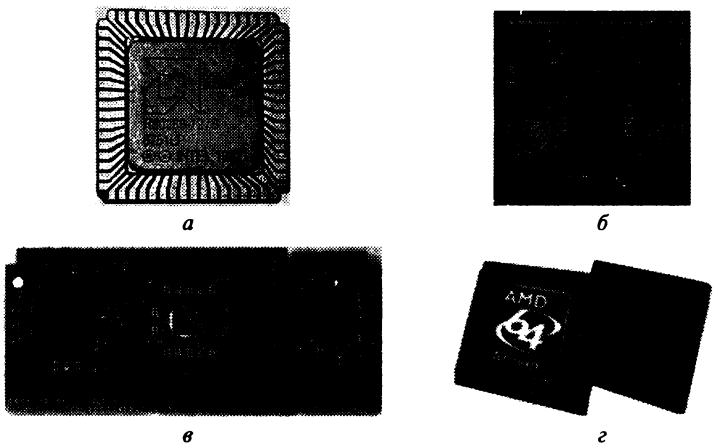
Cyrix MII. Процессор MII — развитие 6x86MX, работает на более высоких частотах. К лету 1998 г. 0,25-микронные процессоры MII-300 и MII-333 производились на новых производственных мощностях компании National Semiconductor в шт. Мэн, нацеленных на развитие технологии 0,22 мкм, продвигаясь к своей конечной цели — 0,18 мкм в 1999 г.

Микроархитектуры AMD

AMD работала в сфере персональных компьютеров на протяжении всей истории отрасли (рис. 3.30). Компания поддерживала производство каждого поколения процессоров ПЭВМ, начиная с 8088, применяемых в первых ПК IBM-PC, до современных процессоров 7- —8-го поколений AMD (табл. 3.5, 3.6, рис. 3.30).

Таблица 3.5. **Оценка числа произведенных процессоров, млн шт.**

Год	2005	2006	2007
Всего процессоров	225,6	235,2	263,8
Производство Intel	183,3	180,4	197,8
Производство AMD	42,4	54,8	66,0



а *б* *в* *г*

Рис. 3.30. Процессоры AMD:
а — AMD 80286 (1982 г.); *б* — Sempron 3000+ (Socket A); *в* — Athlon (формат Slot A); *г* — Athlon 64 (Socket 754)

Таблица 3.6. **Основные характеристики процессоров AMD**

Тип процессора	Архитектура	Год выпуска	Кодовое наименование	Количество транзисторов, млн	Ядро, мм	L1-кэш, Кбайт	L2-кэш, Кбайт	Размер минимальной структуры, мкм	Тактовая частота шины, МГц	Тактовая частота процессора, МГц	Потребляемая мощность, Вт	Интерфейс
AMD K5	K5	1996	SSA/5	4,3	271—161	8+16	Внешн.	0,5—0,35	50—66	75—100	11—15	Socket 5/7
		1996	Godot	4,3	181	8+16	Внешн.	0,35	60—66	90—115	12—16	Socket 5/7
AMD K6	K6	1997	Nx686 (Model 6)	8,8	162	32+32	Внешн.	0,36 CMOS	66	166—300	13—28	Socket 7
		1998	Little Foot	8,8	88	32+32	Внешн.	0,25	66	200—300	13—28	Socket 7
K6-II		1998—2001	Chompers	9,3	81	64	Внешн.	0,25	66—100	266—550	15—30	Super7 (321 p)
K6 III		1999	Sharptooth	21,3	118	64	256	0,25	100	400—450	18—30	Super7
Athlon	K7	1999	Argon	22,0	184	128	512	0,25	200	500—700	36—54	Slot A (575 p)
		2000	Pluto	22,0	102	128	512	0,18	200	550—950	31—62	Slot A
		2000—2001	Thunderbird	37,0	120	64+64	256	0,18	200	700—1,4 ГГц	38—72	Socket A/ Slot A
Duron		2000—2001	Spitfire	25,0	100	64(I) + 64(D)	64—128	0,18	200	600—950	27—41	Socket A (Socket 462)
		2001—2002	Morgan	25,18	100	128	64	0,18	200	900—1,3 ГГц	44—60	S 462
		2003	Applebred	37,2	85	128	64	0,13	266	1,4—1,8	57	S 462
Athlon XP/MP		2001—2002	Palomino	37,5	136	128	256	0,18	266	1,4—1,7	62—72	Socket 462
		2002	Thoroughbred	37,2	85	128	256	0,13	266	1,4—2,25	49—74	S 462
		2003—2004	Barton	54,3	101	64+64	512	0,13	266—400	1,86—2,33	66—77	Socket A

Продолжение табл. 3.6

Тип процессора	Архитектура	Год выпуска	Кодовое наименование	Количество транзисторов, млн	Ядро, мм	L1-кэш, Кбайт	L2-кэш, Кбайт	Размер минимальной структуры, мкм	Тактовая частота шины, МГц	Тактовая частота процессора, МГц	Потребляемая мощность, Вт	Интерфейс
Sempron	K7	2004	Thorton	54,3	101	128	256	0,13	333	1,5—2,0	62	S 754/S 939
		2004	Thoroubred	37,2	85	128	256	0,13	333	1,5—2,0	62	S 462
		2005	Winchester	68,5	04	128	128	0,09	400	1,6	62	S 754
		2005	Palermo	68—75	84	84+64	128—256	0,39	400	1,6—1,8	59—64,0	Socket A/ Socket 754
		2006	Manila	103	81	128	128—256	0,09	400	1,6—2,0	35—62	AM2
Athlon 64	K8	2003—2004	Clawhammer	105,9	193	128	512—1024	0,13	400	1,8—2,4	89	S 754
		2004	Newcastle	68,5	144	128	512	0,09	400	1,6—2,4	89	S 754
		2004	Winchester	68,5	84	128	512	0,09	400	1,8—2,2	67	S 939
		2005	Venice	76	84	128	512	0,09	400	2,0—2,4	16—89	S 754
		2005	San Diego	114	115	128	512—1024	0,09	400	2,2	89	S 939
		2006	Orleans	129	125	128	512	0,09	400	1,8—2,4	35—62	AM2
		2006	Manchester	154	147	128	512	0,09	400	2,0—2,2	67	S 939
Opteron		2003	Sledgehammer			64+64	1024	0,13	800/HT	1,4—2,4	55—95	S 940
		2005	Venus, Troy, Athens			64+64	1024	0,09	1000/HT	1,6—3,0	55—85	S 940
Athlon 64 X2	K8 2-ядерные	2006	Manchester	154	147	64×2	512×2	0,09	667—800	2,0—2,4	69—110	S 939
		2005	Toledo	233	199	64×2	512×2	0,09	1000/HT	2,0—2,4	89—110	S 939
		2006	Windsor	243	220	128×2	512×2	0,09	1000/HT	2,0—2,6	35—89	AM2
		2006	Brisbane	153,6	100	128×2	1024×2	0,09		2,0—2,8	65—89	AM2

Окончание табл. 3.6

Тип процессора	Архитектура	Год выпуска	Кодовое наименование	Количество транзисторов, млн	Ядро, мм	L1-кэш, Кбайт	L2-кэш, Кбайт	Размер минимальной структуры, мкм	Тактовая частота шины, МГц	Тактовая частота процессора, МГц	Потребляемая мощность, Вт	Интерфейс
Turion 64/X2 (мобильный)	K8	2005—2008	Lancaster, Richmond, Taylor, Tyler			64+64	256—1024	0,09—0,065	800 HT	1600 —2400	31—35	Socket 754, Socket S1
Opteron Quadrcore	K10	2007	Barselona, Budapest	463	283	64+64	256—512 + L3 (2048)	65	1000 HT	1700 —2500	95	Socket F
Opteron Quadcore	K10	2008—2009	Shanghai			128	512 +L3 (6 Мбайт)	45 нм	HT 3.0	2,3—2,7 ГГц	55—105	Socket F/1207
Phenom	K10.5	2008—2009	Shanghai	758	243	128	512 +L3 (6 Мбайт)	45 нм	HT 3.0	2,3—2,7 ГГц	55—105	Socket F/1207
		2008—2009	Deneb (4 ядра)	738	258	128×4	4×512 KiB+ +L3 (6 MiB)	45 нм	HT 4,0 ГГц	2,6—3,0 ГГц	95—140	AM2+, AM3
		2009	Propus				4×512 KiB	45	HT 4,0 ГГц	2,1—2,8 ГГц	45—95	Socket AM3
		2010	Thuban (6 ядер)	904	346	128×6	6×512 KiB+ +L3 (6 Мбайт)	45	HT 4,0 ГГц	2,6—3,3 ГГц; до 3,7 ГГц (Turbo Core)	45—95	Socket AM2+, Socket AM3
Athlon/ Phenom ×2 (2 ядра)	K 10	2007—2008	Kuma/Rana				2 ? 512 Кбайт + L3 (2 Мбайт)	65 нм	HT 2,8— 4,2 ГГц	1,9—2,4 ГГц	65—95	AM2+
		2009	Regor				2×1 MiB	45 нм	HT 4,0 ГГц	2,0 ГГц	45—95W	Socket AM3

AMD K5. Длительное время AMD (Advanced Micro Devices), подобно Cyrix, производила центральные процессоры 286, 386 и 486, которые были основаны на разработках Intel. K5 был первым независимо созданным x86 процессором, на который AMD возлагала большие надежды.

K5 был разработан компанией AMD как конкурент процессору Intel Pentium. Он был представлен в 1995 г., более чем на год позже Pentium, кроме того, AMD не удалось выпустить K5, работающие на первоначально запланированной частоте. Процессор содержал 4,3 млн транзисторов и обладал хорошей совместимостью с x86, но не поддерживал набор инструкций MMX. ЦП включал пять блоков ФЗ, поддерживающих внеочередное выполнение, один блок ПЗ, сравнимый по производительности с двумя такими же в Pentium.

Под маркой K5 выпускалось два варианта процессоров SSA/5 и 5k86. SSA/5 работал на частотах от 75 до 100 МГц, 5k86 — от 90 до 133 МГц. AMD использовала так называемый рейтинг производительности (PR rating) для маркировки процессоров. Этот рейтинг показывал, какому процессору Pentium эквивалентен данный K5 по производительности. Например, 116-МГц процессор «5k86» был маркирован «K5 PR166», поскольку AMD считала его производительность эквивалентной Pentium-166.

Проект K5 был одной из возможностей компании AMD перехватить техническое лидерство у Intel. Но хотя при разработке использовались верные дизайнерские концепции, инженерное воплощение оказалось недостаточным. Низкая тактовая частота процессора частично объясняется трудностями с производственными мощностями, которые испытывала компания в то время, однако вчетверо больший, чем у Pentium, буфер предсказания переходов не показывал лучшую производительность, блок ПЗ был менее производительным, чем у Pentium, и т. д. Из за опоздания с выходом на рынок и недостаточной производительности K5 так и не завоевал признания у производителей компьютеров.

Архитектура AMD K6. Однако покупка компанией AMD основанного в Калифорнии конкурента весной 1996 г., кажется, создала возможность лучше подготовиться к своей следующей атаке на Intel.

Процессор K6 начал жизнь как Nx686, будучи переименованным после приобретения NextGen. Серия MMX-совместимых

процессоров K6 была запущена в середине 1997 г., за несколько недель до Cyrix 6x86MX, и сразу была одобрена пользователями.

Изготовленный по 5-слойной технологии 0,35 мкм, K6 был почти на 20 % меньше, чем Pentium Pro и при этом содержал на 3,3 млн транзисторов больше (8,8 против 5,5 млн). Большинство этих дополнительных транзисторов находилось в кэше первого уровня на 64 Кбайт (32 Кбайт кэш команд и 32 Кбайт кэш данных). Это равносильно четырем Pentium Pro или двум Pentium MMX и Pentium II.

ЦП K6 поддерживал технологию MMX Intel, включая 57 новых x86 команд, разработанных для развития мультимедийного программного обеспечения. Как и Pentium Pro, K6 был многим обязан классическим технологиям RISC. Используя суперскалярную микроархитектуру AMD RISC86, чип декодировал каждую x86-инструкцию в ряд более простых команд, которые могли быть обработаны, используя типичные принципы RISC — такие, как выполнение команд вне естественного порядка, переименование регистров, предсказание переходов, спекулятивное исполнение, опережающая выборка данных.

ЦП K6 начинал с версий 166, 200 и 233 МГц. Уровень его производительности был очень схож с Pentium Pro соответствующих частот, у которого максимальный размер кэша 2-го уровня достигал 512 Кбайт. Общая черта с чипом Cyrix MX — работа с плавающей запятой — была областью относительной слабости (но в несколько меньшей степени) по сравнению с Pentium Pro или Pentium II.

AMD K6-2. Процессоры AMD K6-2 с 9,3 млн транзисторов производились по 0,25-микронной технологии AMD. Процессор был упакован в 100 МГц Super7-совместимую, 321-контактную керамическую плату (ceramic pin grid array package — CPGA).

K6-2 включает инновационную эффективную микроархитектуру RISC86, большой (64 Кбайт) кэш первого уровня (двухпортовый кэш данных на 32 Кбайт, кэш команд на 32 Кбайт с дополнительным предрасшифровывающим кэшем на 20 Кбайт), а также улучшенный модуль работы с плавающей запятой. Эффективная производительность при его запуске в середине 1998 г. была оценена в 300 МГц, к началу 1999 г. самым быстрым из доступных процессоров была версия 450 МГц.

Трехмерные возможности K6-2 представляли другое важное достижение. Они были воплощены в AMD технологии 3DNow!, как новый набор из 21 команды, который дополнял стандартные

команды MMX, уже включенные в архитектуру K6, что ускоряло обработку трехмерных приложений.

AMD K6-III. В феврале 1999 г. AMD объявила о начале выпуска процессора K6-III на 400 МГц под кодовым названием «Sharptooth» и опробовала версию на 450 МГц. Ключевой особенностью этого нового процессора была инновационная разработка — трехуровневый кэш (рис. 3.31).

Традиционно процессоры ПК использовали два уровня кэша:

- кэш первого уровня (L1), который обычно помещался в кристалле (on-die);
- кэш второго уровня (L2), который мог располагаться либо вне ЦП, на материнской плате или слоте, либо непосредственно на чипе ЦП (on-chip).

Общее эмпирическое правило при проектировании подсистемы кэша — чем больше и быстрее кэш, тем выше производительность (ядро центрального процессора может быстрее получить доступ к инструкциям и данным).

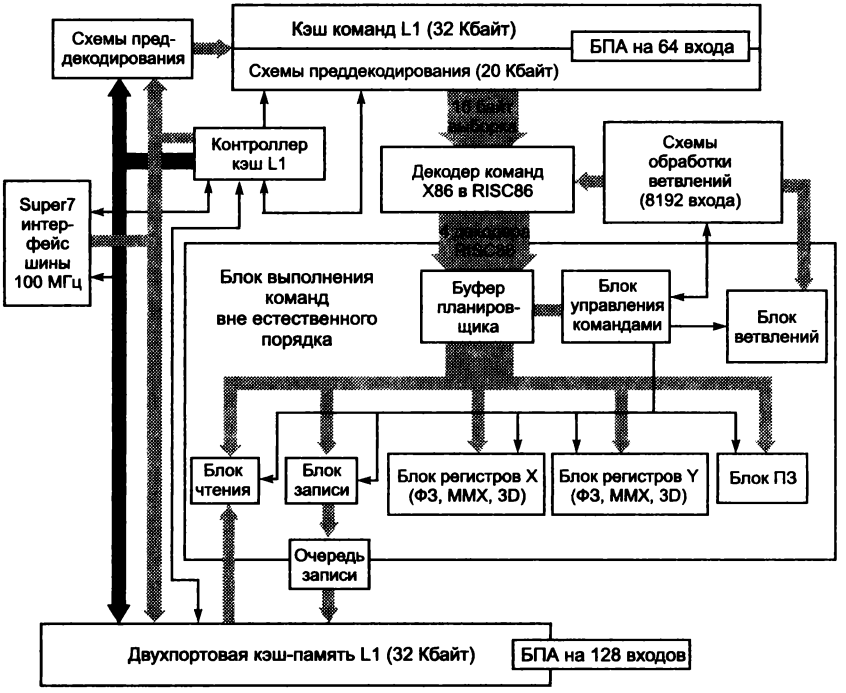


Рис. 3.31. Процессор K6

Признавая выгоды большого и быстрого кэша в удовлетворении потребностей приложений, все более требовательных к производительности ПК, «трехуровневый кэш» компании AMD вводил архитектурные новшества кэша, разработанные для увеличения производительности ПК на основе платформы Super7:

- внутренний L2-кэш (256 Кбайт), работающий на полной скорости процессора AMD-K6-III и дополняющий кэш L1 (64 Кбайт), который был стандартен для всего семейства процессоров AMD-K6;
- многопортовый внутренний кэш, позволяющий одновременное 64-битовое чтение и запись как кэшу L1, так и L2;
- первичная процессорная шина (100 МГц), обеспечивающая соединение с резидентной кэш-памятью на системной плате, расширяемой от 512 до 2048 Кбайт.

Проект многопортового внутреннего кэша процессора AMD-K6-III позволил как кэшу L1 (64 Кбайт), так и кэшу L2 (256 Кбайт) выполнять одновременное 64-битовое чтение и запись операций за один такт процессора. В дополнение к этому многопортовому проекту кэша ядро процессора AMD-K6-III было в состоянии получить доступ к кэшам L1 и L2 одновременно, что увеличивало общую пропускную способность центрального процессора.

Процессоры AMD K7. К этому типу относятся ЦП Athlon, Athlon XP/MP, Sempron. K7 — первый из семейства микропроцессоров x86 7-го поколения, в котором присутствуют конструктивные решения, до сих пор не применявшиеся в процессорах архитектуры x86 и сулящие выигрыш в быстродействии даже при одинаковых тактовых частотах (рис. 3.30, *б, в,* рис. 3.32):

- многократные декодеры. Если Pentium II (Katmai) разбивает CISC x86-команды на более мелкие и быстрые RISC-команды, которые затем можно параллельно исполнять, то K7 оперирует блоками x86-инструкций, которые AMD называет МакОП (macroOPS, mOPs). Некоторые такие блоки могут содержать одну x86-инструкцию, а большинство содержит две. Например, «прочитать данные в регистр и инвертировать их». Конвейер декодирования инструкций может обрабатывать до 3 МакОП за цикл, после этого они идут в модуль контроля инструкций (ICU). Всего в обороте одновременно могут находиться до 72 x86-команд. С одной стороны, такие блоки, несомненно, больше небольших RISC-команд, с другой стороны, за счет такого подхода

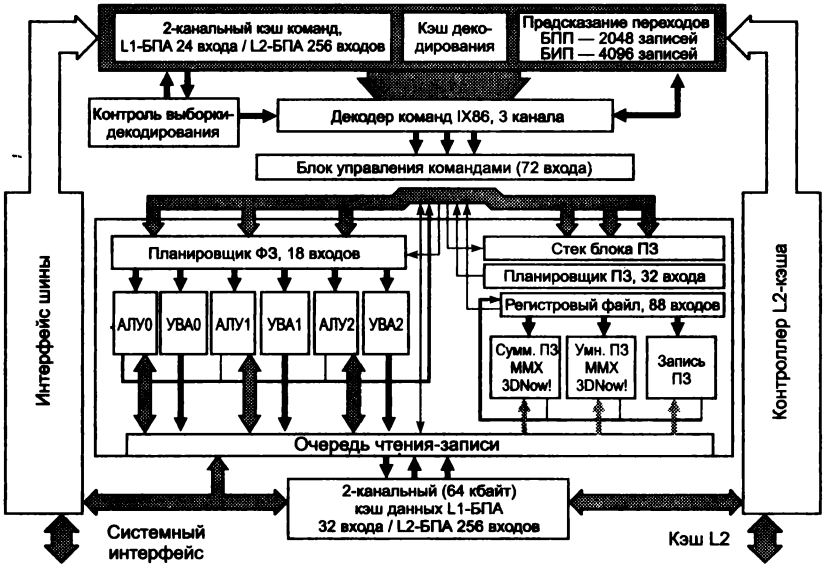


Рис. 3.32. Архитектура AMD K7 (Athlon)

ядро может непосредственно работать с x86-инструкциями вместо того, чтобы эмулировать их через RISC-команды;

- блок контроля команд. Как только МакОП расшифрована за цикл до трех МакОП посылаются на блок управления инструкциями (Instruction Control Unit — ICU). Это буфер, который управляет выполнением каждой МакОП в целом, осуществляет переименование регистра для операндов, управляет любыми условиями исключения и действиями команды, посылает МакОП планировщику исполнения. ICU рассчитан на 72 инструкции против 20 у Pentium III; увеличивая этот буфер, удалось добиться того, чтобы дешифратор команд не простаивал из-за его переполнения;
- благодаря наличию трех конвейерных блоков исполнения целочисленных команд (АЛУ0—АЛУ2), K7 может выполнять три целочисленные инструкции параллельно;
- новая архитектура узла вычислений с плавающей точкой (FPU). K7 содержит три узла вычислений с плавающей точкой, любой из которых способен принимать на вход инструкции каждый такт работы процессора. При этом один узел служит исключительно для выполнения команды `FSTORE`

(запись в память числа с ПЗ) и назначение этого узла — обеспечивать обмен между регистрами и памятью в то время как процессор выполняет другие инструкции. Остальные два — блок сложения (adder) и блок умножения (multiplier). Оба блока полностью конвейеризованы (fully pipelined);
- 200-мегагерцовая системная шина;
- кэш L1 увеличился в 2 раза — до 128 Кбайт;
- модернизируемый кэш L2 размещен по примеру Pentium II в картридже, а не интегрирован в кристалл, как для K6-3. K7 может включать кэш L2 размером от 512 Кбайт в дешевых моделях до 8 Мбайт в серверах;
- расширенный набор команд 3DNow! — модернизирован добавлением 24 новых команд к исходным 21 инструкциям 3DNow! (19 команд, чтобы улучшить целочисленные математические вычисления MMX и передачу данных в Internet-приложениях, и 5 команд DSP-расширения для обычных модемов, модемов ADSL, систем Dolby Digital и MP3).

AMD Athlon. Выпуск процессора Athlon летом 1999 г. был наиболее удачным ходом AMD. Это позволило им гордиться тем, что они произвели первый процессор седьмого поколения (у него было достаточно много радикальных архитектурных отличий от Pentium II/III, и K6-III, чтобы заслужить название процессора следующего поколения), и это означало также, что они вырвали технологическое лидерство у Intel.

Древнегреческое слово Athlon означает «трофей» или «игры». Athlon — процессор, с помощью которого AMD надеялась увеличить реальное конкурентоспособное присутствие в корпоративном секторе, помимо ее традиционного преимущества на потребительском рынке и рынке трехмерных игр. Ядро размещается на кристалле в 102 мм^2 и содержит приблизительно 22 млн транзисторов.

Atlon использует разъем Slot A компании AMD, который является механически совместимым с системными платами с разъемом Slot 1, но использует другой электрический интерфейс, подразумевая, что центральные процессоры Athlon не будут работать с обычной Slot 1 системной платой. Slot A разработан, чтобы электрически соединяться с системной шиной на 200 МГц, основанной на шинном протоколе Alpha EV6, предоставляя, таким образом, существенное преимущество в производительности перед инфраструктурой Slot 1. Так же, как обеспе-

чение ее собственного оптимизированного чипсет-решения (чипсет AMD-750, см. рис. 4.26), компания работает над тем, чтобы вынуждать сторонних поставщиков чипов способствовать поставке ее собственных Athlon-оптимизированных решений.

Athlon первоначально выпускался в диапазонах скорости 500, 550 и 600 МГц и немного позднее — 650 МГц (все изготовлены по технологии 0,25 мкм). К концу 1999 г. AMD еще более повысил частоту — его ядро K75 (750 МГц), является первым процессором, построенным с использованием алюминиевой 6-слойной технологии 0,18 мкм компании AMD.

Утверждение о том, что это был самый быстрый x86-совместимый ЦП тысячелетия, спорно, поскольку Intel быстро ответила объявлением Pentium III (800 МГц). Однако AMD вскоре вернула лидерство 2000 г. выпуском версий на 800 и 850 МГц и даже преуспела в опережении Intel в преодолении барьера 1 ГГц буквально на несколько недель.

Thunderbird. В середине 2000 г. была выпущена улучшенная версия Athlon с кодовым названием «Thunderbird».

Технология 0,18 мкм, кэш-память 2-го уровня (L2) размером в 256 Кбайт расположена на плате процессора и работает на полной частоте процессора (первые процессоры Athlon имели кэш L2, работавшую на меньших частотах, например при частоте в 1 ГГЦ, память L2 работала на 330 МГц).

Интерфейсы — 462-контактный Socket A и Slot A. Частоты от 0,75 до 1 ГГц. Размещение 256 кбайт памяти на кристалле привело к увеличению его размера до 120 мм^2 (102 мм^2 для ядра). Однако, он меньше исходного (0,25-micron) K7 Athlon, который занимает 184 мм^2. Добавление 256 Кбайт к L2-кэшу на кристалле весьма увеличивает число транзисторов. ЦП Thunderbird включает 37 млн транзисторов, т. е. 15 млн добавились для размещения кэша L2.

Осенью 2000 г. был выпущен чипсет AMD760 (см. далее), обеспечивающий поддержку для памяти DDR SDRAM PC1600 (200 МГц FSB) и PC2100 (266 МГц FSB). Другие особенности — AGP 4-x, 4 порта USB, адресация памяти 8 Гбайт на 4 DIMM и поддержка ATA-100. С этого момента процессоры Athlon выпускались только для разъемов Socket A. Последние из процессоров Athlon/Thunderbird были выпущены летом 2001 г., достигнув частоты 1,4 ГГц.

Duron. В середине 2000 г. был выпущен процессор Duron, предназначенный для дома и офиса (вариант Athlon, ядро —

Spitfire). Название происходит от латинского «durare» — «вечный», «длительный». Кэш-память L1 (128 Кбайт, по 64 Кбайта на код и на данные) и L2 (64 Кбайт) размещается на плате. Первичная системная шина работает на частоте 200 МГц. Поддерживается улучшенная технология 3DNow! Технология 0,18 мкм, частоты 600, 650, 1000, 1200 МГц. Интерфейс — 462-контактный разъем Socket A.

Содержит 25 млн транзисторов и имеет площадь 100 кв. мм, использует высокопроизводительную 100 МГц DDR (Double Data Rate, фактически данные передаются на частоте 200 МГц) системную шину EV6, все подсистемы работают на полной частоте ядра, напряжение питания — 1,5 В.

Palomino (серия Athlon XP, «XP» — «Extra Performance», сверхпроизводительность). Процессор выполнен по технологии 0,18 мкм с использованием медных проводников на плате (вместо алюминия), содержит 37,5 млн транзисторов на кристалле в 128 мм^2. Достигнуто понижение на 20 % энергопотребления сравнительно с Thunderbird. Введен ряд новшеств, в совокупности именуемых AMD как «QuantiSpeed Architecture»:

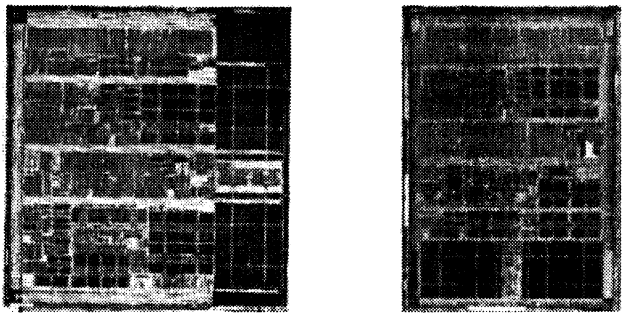
- введение дополнительного буфера — буфера быстрого преобразования адреса (БПА, TLB — Transition Lookaside Buffer). Это дополнительная кэш-память, расположенная между L1 и L2. В частности, TLB содержит данные, которые используются для перевода виртуальных адресов в физические и наоборот. Вероятность того, что процессор найдет необходимые данные в TLB (или TLB hit-rate), оказывается достаточно высокой;
- поддержка SSE-технологии Intel. В Palomino добавлены еще 52 новые команды SIMD по отношению к ранее имевшимся. Удвоено количество исходных 21 SIMD-команд, реализующих «3DNow!», и получена технология «Enhanced 3DNow!» («3DNow! Professional»);
- использование технологии упаковки OPGA (organic PGA) для замещения CPGA (ceramic PGA), которая использовалась ранее. Применение пластмасс вместо керамики технологичнее, платы оказываются легче и обладают лучшими тепловыми свойствами. Кроме того, можно плотнее размещать навесные элементы, что уменьшает наводки и помехи. OPGA размещаются на уже известном разъеме Socket A.

Morgan. Morgan представляет модификацию процессоров AMD, первоначально представлял ядро Palomino с удаленными

3/4 кэша L2 (64 Кбайт вместо 256 Кбайт). Размер кристалла — 106 мм^2, число транзисторов — 25,18 млн. Напряжение питания было изменено от 1,6 до 1,75 В.

Thoroughbred. Летом 2002 г. AMD начала поставлять первый процессор с технологией 0,13 мкм и медными соединителями. Площадь кристалла — 80 мм^2 (у его предшественников — 128 мм^2). Питание — 1,65 В, размеры кэша на кристалле — 128 Кбайт для L1 и 256 Кбайт для L2, разъем — Socket A. Эквивалентная производительность Athlon XP — 2400+ или 2600+.

Однако ядро Thoroughbred рассматривать как простую переделку Palomino с учетом новых норм технологического процесса все же не совсем верно. Thoroughbred по своей внутренней структуре значительно отличается от Palomino, в этом можно убедиться хотя бы по внешнему виду процессорных ядер (рис. 3.33).



а *б*

Рис. 3.33. Процессорные ядра Athlon XP:
а — Palomino; *б* — Thoroughbred

Barton. В феврале 2003 г. AMD начинает выпускать первый процессор, основанный на обновленном ядре Athlon XP (кодовое наименование «Barton»). В то время как этот чип Athlon 3000+ имел все особенности предыдущего процессора Athlon XP, новое ядро позволило организовать более эффективный дизайн памяти, чем в любых предыдущих процессорах Athlon. Это связано с тем, что здесь были объединены лучшие характеристики предшественников — кэш L2, работающий на полной скорости (Thunderbird и его последователи), и больший размер кэша в 512 Кбайт исходного процессора Athlon.

Вследствие большего кэша здесь существенно увеличивается размер кристалла, который включает 56 млн транзисторов

и измеряется 101 мм^2. Это представляет существенное возрастание по сравнению с ЦП Thoroughbred (54,3 млн транзисторов на 84 мм^2).

Однако ядро Barton предусматривает более быстрый доступ к главной памяти — в то время как более ранние ЦП Athlon XP характеризуются частотой FSB 266 МГц (133 × 2), здесь частота FSB составляет 333 МГц (166 × 2).

В итоге Athlon XP 3000+ показывает более высокую эффективность сравнительно с более ранним Athlon XP 2800+, несмотря на меньшую тактовую частоту (2,167 ГГц против 2,250).

Появление ЦП Athlon 64 осенью 2003 г. подразумевало сход со сцены процессоров Barton, и последний процессор Athlon XP (модель 3200+) вышел не более чем через три месяца после появления первых изделий Barton.

Sempron. Летом 2004 г. AMD объявила о выходе ЦП семейства Sempron. Первоначально задуманный как преемник успешного ЦП Duron и прямой конкурент процессору Celeron D (Intel, 90 нм), диапазон применения Sempron фактически перекрыл диапазон Athlon AMD XP и поставил фирмы, выпускающие настольные и мобильные ПК перед выбором — либо Sempron, либо Athlon 64.

Все первые ЦП базировались на технологии AMD 130 нм. Наиболее мощные образцы (3100+) выпускаются в формате интерфейса Socket 754 (Athlon 64 — в формате Socket 939). Другие участники семейства — от 2 ГГц (2800+) до 1,5 ГГц (2200+) —используют Socket A.

В дальнейшем Sempron предполагается перевести на технологию 90 нм и интерфейс Socket 939.

Архитектура K8. Эта архитектура используется во всех современных серверных, настольных и мобильных процессорах AMD (Opteron, Athlon 64 и Athlon 64 X2). Первым из процессоров K8 являлся Hammer (середина 2000 г.).

Одним из главных новшеств K8 является 64-разрядная архитектура x86-64 ISA. Примером 64-разрядных процессоров (IA-64) является Intel Itanium (см. рис. 3.19). Однако между 64-разрядными архитектурами процессоров Itanium и K8 мало общего. Itanium — процессор, несовместимый с системой команд x86, тогда как K8, напротив, таковым является.

Стратегия AMD на 64 бита (x86-64) заключается в следующем — за основу взято производительное x86-ядро и расширен

набор инструкций для возможности адресации 64-битового пространства памяти.

Особенности архитектуры x86-64 (AMD64):

- обратная совместимость с инструкциями x86;
- 8 новых 64-разрядных РОН плюс 64-разрядные версии прежних 8 РОН x86 (доступны лишь в 64-разрядном «длинном» режиме — табл. 3.7);
- поддержка SSE и SSE2 помимо восьми новых регистров SSE2;
- увеличен объем адресуемой памяти для приложений, работающих с большими объемами данных (доступно лишь в «длинном» режиме);
- высокая производительность 32-разрядных приложений, плюс поддержка появляющихся 64-битовых приложений, хороший вариант переходного процессора.

Таблица 3.7. **Режимы процессоров K8**

Режим	Подрежим	Назначение	Адресуемая память, Гбайт	Операционная система	Примечания
«Преемственный» (Legacy Mode)	Нет	Работа со всеми 16- или 32-битовыми x86-приложениями	4	32-разрядная	Используются только 32 разряда в 64-разрядных регистрах. Дополнительные 64-разрядные регистры не задействованы. Перекомпиляция ПО не требуется
«Длинный» (Long Mode)	Полный (64 разряда)	Работа с 64-разрядными приложениями (инструкции x86-64)	Более 4	64-разрядная	Используются 64-разрядные основные и дополнительные регистры. Требуется перекомпиляция старых программ
	Совместимости (Compatibility Mode)	Запуск 32-разрядных программ в 64-разрядной ОС	2 в 32-битовой ОС, 4 в 64-битовой ОС		Используются только 32 разряда в 64-разрядных регистрах. Дополнительные 64-разрядные регистры не задействованы. Перекомпиляция ПО не требуется

Основные недостатки:

- процессор продолжает поддерживать архитектуру x86, которая достаточно устарела;

- новые РОН можно использовать лишь в 64-разрядном режиме, что не позволяет повысить производительность 32-разрядных приложений посредством улучшения архитектуры системы команд.

Для реализации возможности работы как с 32-, так и с 64-битовыми приложениями процессоры K8 поддерживают два режима работы — Long Mode и Legacy Mode. В режиме Long Mode также предусмотрено два подрежима — 64-битовый и Compability mode (совместимый) — табл. 3.7.

Некоторые прочие особенности K8 (рис. 3.34):

- контроллер памяти интегрирован в сам процессор. Традиционно он располагается в северном мосте чипсета на системной плате. Собственно говоря, контроллер памяти — это основной функциональный блок «северного моста» (в чипсетах Intel он так и называется — MCH, Memory Controller Hub). Преимущество такого решения очевидно — контроллер памяти работает на частоте процессора, а следовательно, обладает низкой латентностью, которая будет тем меньше, чем больше частота, на которой работает процессор;

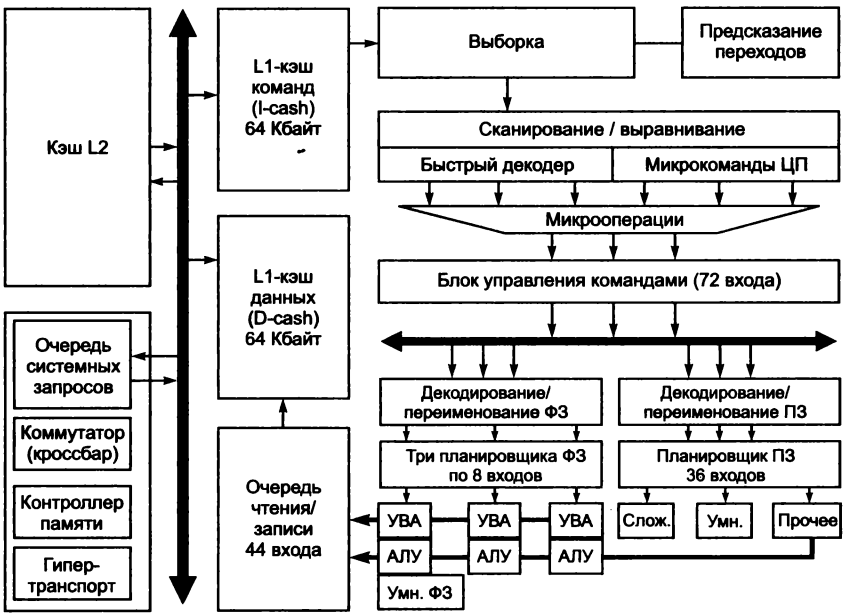


Рис. 3.34. Архитектура AMD K8

- встроенный порт («линк») шины HyperTransport — универсальной шины межчипового соединения — элемент транспьютерной архитектуры (см. рис. 2.46). В процессорах K8 Opteron может быть до 3—4 линков HT, что позволяет комбинировать их в 3- и 4-мерные кластерные структуры (см. рис. 2.41, а также рис. 2.42);
- архитектура K8 разработана с перспективой создания многоядерных процессоров и многопроцессорных систем: если ЦП Intel Xeon может продемонстрировать лишь 11%-ное увеличение производительности при переходе к двум процессорам, то в случае с Opteron оно составляет 24 %;
- усовершенствован блок предсказания переходов — для увеличения точности он содержит историю 16 000 переходов, а также 2000 адресов назначения.

Декодирование команд здесь происходит по схеме, близкой к рис. 3.6, *г*. Исполнение инструкций на конвейере K8 начинается с блока выборки инструкций (см. рис. 3.2, *а*, рис. 3.34). За один такт блок выбирает из кэша 16 байт данных и выделяет из них от одной до трех инструкций x86 — сколько в выбранных данных поместилось. Поскольку средняя длина команды x86 составляет 5—6 байт, то, как правило, блоку удается выбрать три команды за такт.

На втором такте конвейера выбранные команды распределяются по трем блокам декодирования инструкций. Самые сложные команды отправляются в декодер сложных команд (VectorPath), другие — в декодеры простых команд (DirectPath).

Исходные x86-инструкции на завершающих этапах работы декодера K7/K8 переводятся в макрооперации, или МакОПы (mOPs). Большинству x86-инструкций соответствует одна МакОП, некоторые преобразуются в 2 или 3, а наиболее сложные, например деление или тригонометрические, — в последовательность из нескольких десятков МакОП. Макрооперации имеют фиксированную длину и регулярную структуру.

Условно можно считать, что в определенный момент МакОп может «расщепляться» на две микрооперации (МкОП, μОП). Как правило, в K7 и в K8 МакОП содержит две МкОП — одну для АЛУ (ALU) (или блока ПЗ — FPU), другую — для УВА (устройства вычисления адреса, AGU — Address Generation Unit).

За счет конвейеризации возможны ситуации, когда одновременно в разных блоках процессора будут выполняться до двух десятков команд — и в K7, и в K8 имеется десять исполнитель-

ных устройств — три ALU, три FPU, три AGU и отдельный блок умножения.

Подобно тому, как объединение двух отдельных МкОП в одну МакОП дает явные преимущества, точно так же дела обстоят и с самими МакОП — практически везде они выступают не в виде самостоятельных единиц, а в виде группы. Группу образуют три МакОП, которые одновременно запускаются на параллельные каналы.

Вся дальнейшая работа идет не с одиночными, а с «тройками» МакОП («линиями», line). Такая «линия» воспринимается центральным управляющим блоком процессора ICU как единое целое: все основные действия выполняются именно над «линиями», в первую очередь — выделение внутренних ресурсов.

Сгенерированные линии от декодеров по одной за такт поступают в блок управления командами — Instructions Control Unit (ICU), где подготовленные к исполнению линии накапливаются в специальной очереди (24 линии).

Из очереди в 24 линии (по три МакОП в каждой) ICU выбирает в наиболее удобной для исполнения последовательности одну-три МакОП и пересылает их либо на АЛУ, либо на блок ПЗ в зависимости от типа микрооперации. В случае АЛУ микрооперации сразу же попадают в очередь планировщика (шесть элементов по три МакОП), который подготавливает необходимые для исполнения микрооперации ресурсы, дожидается их готовности и только потом отправляет. Причем при исполнении одной МакОП на самом деле может происходить исполнение сразу двух действий (МкОП).

Подготовка данных в планировщике занимает (в идеальном случае) один такт, исполнение — от одного (подавляющее большинство инструкций) до трех (при обращении к оперативной памяти) и даже пяти (64-разрядное умножение) тактов.

С блоком ПЗ (FPU) все обстоит сложнее. Для начала вышедшие из ICU МкОП проходят две стадии по подготовке их операндов. Затем накапливаются в планировщике FPU (двенадцать элементов по три МакОП), который по аналогии со своим целочисленным аналогом дожидается, пока данные для этих МакОП будут готовы, а исполнительные устройства освободятся, и распределяет накопленные МакОП по трем исполнительным устройствам. Но в отличие от целочисленной части конвейера (где содержатся по три одинаковых блока АЛУ и УВА), исполнительные устройства ПЗ специализированы — каждое производит

только свой специфический набор действий над числами с плавающей запятой. Время выполнения — два такта на переименование и отображение регистров, один такт на планирование и ожидание операндов, четыре такта на собственно исполнение.

Hammer. Процессор, первоначально имевший кодовое имя «Sledgehammer», а в дальнейшем «Hammer», был выпущен в середине 2000 г.

В процессоре Hammer используется улучшенная конвейеризация, предполагающая 12-стадийный конвейер вычисления с ФТ и 17-стадийный конвейер для ПТ. Это связано с необходимостью упаковки-распаковки данных и команд при их передаче и декодировании. Одним из главных новшеств процессора Hammer является 64-разрядная архитектура x86-64 ISA.

Athlon 64. Осенью 2003 г. вышли две модели процессора AMD — Athlon 64 для массового рынка и Athlon 64 FX-51 для мультимедийных и профессиональных приложений (архитектура K8). В системе обозначений AMD Athlon 64 имеет эквивалентную частоту 3200+, при физической частоте 2 ГГц (FX-51 чуть выше — 2,2 ГГц).

Процессор приспособлен как для 32-, так и для 64-битовых приложений. Поддерживаются системы ОЗУ типов PC3200, PC2700, PC2100 или PC1600 DDR SDRAM. Установлены наибольшие по размерам для данного момента модули кэш-памяти на процессоре — 64 Кбайт L1 — кэш команд; 64 Кбайт L1 — кэш данных. Общая кэш-память может достигать 1152 Кбайт.

В то время как Athlon 64 использует 64-битовый одноканальный контроллер памяти, FX-51 включает 128-битовый двухканальный контроллер памяти на чипе.

Первоначально размер кэша 2-го уровня — 512 Кбайт, как и для Athlon XP. Затем он был увеличен до 1 Мбайта, в связи с чем здесь число транзисторов достигло 106 млн по сравнению с 54 млн для Athlon XP.

Процессор выпускается по технологии 0,13 мкм, разработанной в филиале AMD, Дрезден, Германия. Система команд AMD64, x86-64 — расширение AMD для команд Intel x86 — несовместима с Intel IA-64 архитектурой, например поддерживаемой в серверных процессорах Itanium. Однако AMD поддерживает полную совместимость с 32-битовыми процессорами ПЭВМ, выпускаемыми как Intel, так и AMD.

В то же время, когда AMD объявила Athlon 64, фирма Microsoft заявила о выпуске бета-версии Windows XP 64-Bit

Edition для 64-битовых процессоров, которая может работать естественно как на процессорах AMD Athlon 64 (ПЭВМ), так и AMD Opteron (рабочие станции). Хотя здесь и поддерживаются все существующие 32-битовые приложения, 32-битовые драйверы устройств должны быть обновлены и перетранслированы.

Интересно, что Intel в начале 2004 г. объявила, что будущие версии процессора Prescott (Socket T) будут включать 64-битовые расширения x86, совместимые с AMD 64-битовой архитектурой.

Athlon 64 X2. AMD снова оказалась впереди Intel, продемонстрировав действующий экспериментальный образец двухъядерного процессора летом 2004 г., и поэтому Intel вызвала всеобщее удивление, все же выйдя первой на рынок с двухъядерным процессором весной 2005 г. Однако, несмотря на то, что AMD 64 X2 был только короткое время позади Pentium Extreme Edition и Pentium D по датам выхода на рынок, он значительно опережал их по показателям эффективности.

Athlon 64 X2 включает все возможности, заложенные в единственном ядре дизайна Athlon 64 (такие, как HyperTransport и Enhanced Virus Protection — EVP). Когда ЦП работает под ОС Windows XP (SP2), EVP интерпретирует области системной памяти как «только данные», так что любой находящийся здесь фрагмент кода может быть либо прочитан, либо записан, но не может быть выполнен как код программы. Тем самым EVP действует как профилактическая мера против обычных злонамеренных вирусов, локализуя и обезвреживая их.

Когда процессоры X2 были впервые выпущены, они показали бóльшую производительность, чем самые быстрые в то время одноядерные процессоры (4000+), а именно 4200+, 4400+, 4600+ и 4800 +. Двухъядерные системы работают медленнее, чем единственное ядро для некоторых процессов, что затрудняет точное сравнение. Однако специалисты AMD считают, что X2 обеспечивает 80%-ное превышение эффективности над системами с единственным ядром (при той же самой частоте) при выполнении приложений с обработкой мультимедийной информации.

Основная архитектура ядра X2 по существу та же, как и у Athlon 64. Различие в том, что новые чипы, размещаемые на единственном кристалле в 199 мм^2, причем каждый содержит более чем 233 млн транзисторов, изготовлены по технологии AMD 90 нм.

Степень интеграции двойных ядер больше, чем в системах Intel — два ядра AMD способны к прямой связи без использова-

ния внешней платы и чипсета и фактически совместно используют объединенный контроллер памяти.

Таким образом, спецификации первоначально объявленного диапазона Athlon 64 X2 были эквивалентны таковым из существующих ЦП на 3500+, 3700+, 3800+ и 4000+, с изменением кэша L2 и тактовой частоты. Модели с 512 Кбайт кэша на ядре базируются на двойном ядре «Winchecter», в то время как версии кэша L2 на 1 Мбайт используют дизайн «Toledo». К лету 2005 г. диапазон был расширен с появлением нового чипа (3800 +).

Кроме скорости, другим существенным преимуществом, который системы AMD имеют по сравнению с Intel, является совместимость и, следовательно, возможность обновления существующих систем. Принимая во внимание, что ЦП Intel требуют новый чипсет и VRM, чтобы справиться с увеличением мощности (при этом сокращается напряжение питания ядра от 1,50 до 1,35 В), AMD спроектировала двухъядерный чип, соответствующий тому же самому разъему с 939 штырьками, что и у одноядерных процессоров. При этом достигается совместимость с существующими системными платами для Athlon 64 (90 нм) и только требуется модификация BIOS.

Turion 64 — семейство 64-битовых мобильных (с низким энергопотреблением) процессоров. Данные процессоры, включая и Turion 64 X2, являются ответом AMD на линейку мобильных процессоров компании Intel — Pentium M и Intel Core.

Процессоры Turion 64 (кроме Turion 64 X2) совместимы с интерфейсом Socket 754 компании AMD и включают от 512 до 1024 Кбайт кэша 2-го уровня, 64-битовый одноканальный контроллер памяти, интегрированный на ядро, и шину HyperTransport (800 МГц). Основной акцент при позиционировании и продвижении данного процессора на рынке делается на его энергосберегающие функции, такие как PowerNow! и Cool & Quiet.

Lancaster (выпущен 10 марта 2005 г.) изготовлен по технологии 90 нм (SOI), кэш L1 составляет 64 + 64 Кбайт (команды и данные), кэш L2 — 512 или 1024 КБ, работающий на скорости ядра. Процессор поддерживает технологии MMX, Extended 3DNow!, SSE — SSE3, AMD64, PowerNow!, NX Bit.

Напряжение питания ядра: 1,00—1,45 В, потребление энергии (TDP) — максимум 25/35 Вт, диапазон частот — 1600, 1800, 2000, 2200, 2400 МГц, интерфейс — Socket 754, системная шина — HyperTransport (800 МГц, HT800).

Richmond (1 сентября 2006 г.), технология изготовления 90nm (SOI), L1-кэш: 64 + 64 KiB (команды и данные), L2-кэш: 512 KiB (полная скорость ядра). Поддерживается стандартный список технологий (см. выше).

Напряжение питания ядра — 1,00—1,45 В, энергопотребление (TDP) — максимум 31 Вт, частоты — 2000, 2200 МГц, интерфейс Socket S1, HyperTransport (HT800).

Turion 64 X2 — двухъядерный мобильный 64-разрядный процессор AMD. Он конкурирует с процессорами Intel Core и Core 2. Процессор Turion 64 X2 был представлен компанией AMD 17 мая 2006 г. после нескольких задержек. Процессор устанавливается в разъем Socket S1 и использует память DDR2. В Turion 64 X2 используются более совершенные энергосберегающие технологии по сравнению с предыдущими процессорами компании.

Первые модели Turion 64 X2 производятся с использованием 90 нм SOI процесса компании IBM (Taylor). В дальнейшем осуществляется переход на 65 нм процесс (Tyler), вероятнее всего на основе технологии напряженного кремниево-германиевого процесса, который был недавно совместно разработан исследователями IBM и AMD и который является более совершенным по сравнению с другими 65 нм процессами.

Taylor (17 мая 2006 г.), технология 90 нм (SOI), кэш первого уровня: 64 (данные) + 64 (инструкции) Кбайт в каждом ядре; кэш второго уровня: 256 или 512 КБ в каждом ядре, работает на скорости ядра, поддерживает стандартные технологии. Контроллер памяти — двухканальный DDR2, 667 МГц. Интерфейс Socket S1, HyperTransport (800 МГц, HT800), энергопотребление (TDP): 31, 33, 35 Вт максимум, частота: 1600, 1800, 2000 МГц.

Tyler (первая половина 2007 г.), технология 65 нм (SOI), кэш первого уровня: 64 + 64 Кбайт (данные + инструкции) в каждом ядре, кэш второго уровня: 512 Кбайт в каждом ядре, работает на скорости ядра. Контроллер памяти — двухканальный DDR2, 800 МГц, поддерживает стандартные технологии. Интерфейс Socket S1, HyperTransport, энергопотребление (TDP): 35 Вт максимум.

K9. Об архитектуре AMD K9 в анналах истории не сохранилось достоверных упоминаний... Есть сведения, что проект AMD Greyhound (2001—2003 гг.) имел кодовое название K9, кроме того, утверждают, что под этой вывеской первоначально объявлялся Athlon 64 X2.

Существует, кроме этого, легенда о том, что «брэнд K9» был отвергнут разработчиками в связи с его сомнительным звучанием на английском языке — K9 (K-nine) произносится как ['keɪ'naɪn], т. е. аналогично латинскому «canina» («собачий»). Так это или не так, но следующим заметным шагом в AMD-архитектурах стала архитектура K10.

Архитектура K10 представляет собой следующее поколение архитектур AMD. Впервые о существовании новой микроархитектуры заявил в апреле 2006 г. Henri Richard, исполнительный вице-президент AMD, директор департамента маркетинга и продаж. Рассмотрим характеристики микроархитектуры K10 (рис. 3.35, 3.36).

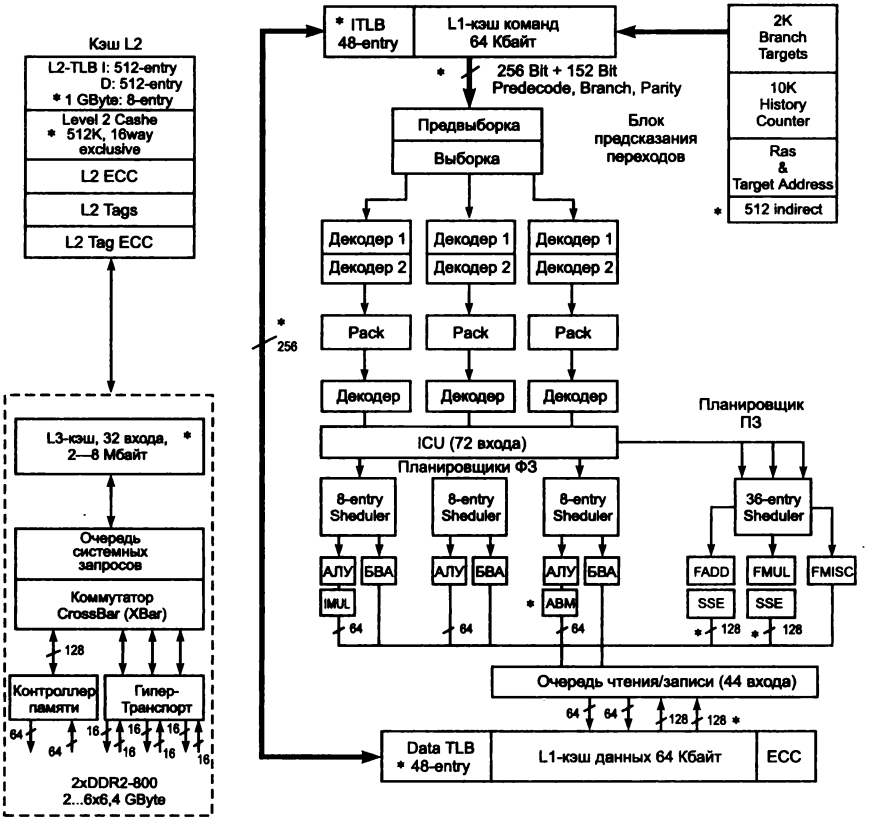


Рис. 3.35. Микроархитектура K10 (символом «*» отмечены количественные или качественные отличия от K8)

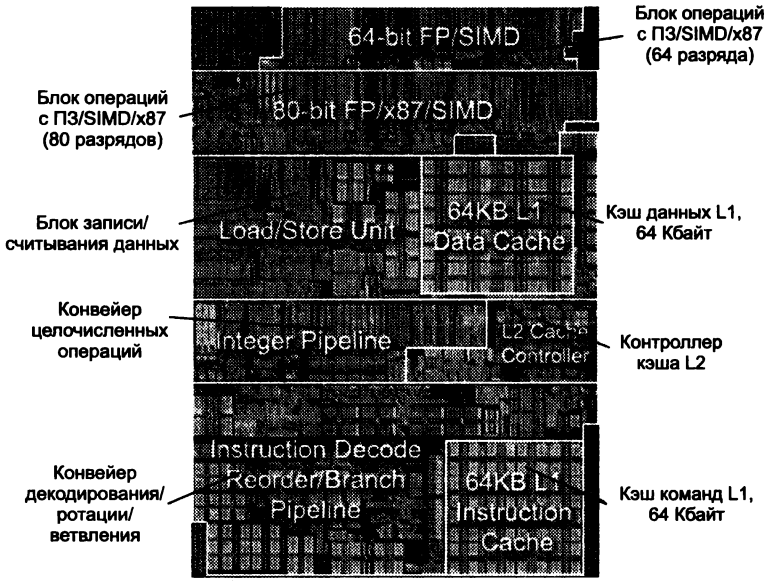


Рис. 3.36. Одиночное ядро процессора архитектуры K10

Интерфейсы:

- Socket AM2+ для процессоров Athlon 64 X2, Phenom X2 и Phenom X4, а также для одноядерных ЦП Opteron и Socket F+ для процессоров Phenom FX, ориентированных на 4-процессорные платформы AMD Quad FX, а также для многоядерных ЦП Opterons, с поддержкой HyperTransport 3.0 и использованием банков ОП DDR2 DIMM;
- обратная совместимость с системными платами, укомплектованными разъемами Socket AM2 и Socket F.

Добавление и расширение набора команд:

- новые команды обработки битовых строк: подсчет ведущих нулей и битов, установленных в единицу — `LZCNT` (Leading Zero Count) и `POPCNT` (Population Count);
- новые команды SSE (названы SSE4a): комбинированного маскирования-сдвига (`EXTRQ/INSERTQ`) и потоковой записи данных в ОП (`MOVNTSD/MOVNTSS`). Эти инструкции отсутствуют в Intel-SSE4;
- поддержка SSE-операций по загрузке невыравненных данных в ОП (ранее требовалось 16-байтовое выравнивание);

- добавлен блок ABM (Advanced bit manipulation), предназначенный для поддержки программных средств мультимедийной обработки (например, Microsoft Visual Studio 2008, а также улучшения генерации кода Visual C++ — см. рис. 1.10).

Усовершенствование конвейера исполнения (см. также рис. 3.34):

- блоки операций SSE на 128 разрядов;
- более «широкий» интерфейс с кэшем данных L1 (запись 128 бит за цикл, сравнительно с 64 битами на цикл для K8);
- меньшие задержки для операций целочисленного деления;
- блок косвенного предсказания переходов (indirect branch predictor) на 512 входов, более емкий стек возвратов (return stack) и буфер точек перехода (branch target buffer), размер удвоен по сравнению с K8;
- оптимизатор регистрового стека, связанный с увеличением/уменьшением указателя стека;
- ускорение исполнения инструкций `CALL` и `RET-Imm` а также команд пересылки (`MOV`) из регистров SIMD в РОН.

Интеграция новых технологий в кристалл ЦП:

- 4-ядерные процессоры (Quad-core);
- Dynamic Independent Core Engagement (DICE) или Enhanced PowerNow! — разделение управления энергопотреблением между ядром ЦП и контроллером памяти (Northbridge), что позволяет им управлять питанием независимо;
- технология CoolCore, позволяющая отключать питание от незагруженных цепей ядра ЦП.

Усовершенствования в системах памяти:

- уменьшение задержек памяти:
  - чтение команд и запись данных, связанные с перестановкой команд, выполняются ранее других аналогичных операций;
  - более активная стратегия предвыборки команд (выбор 32 байтов вместо 16 для K8);
  - использование памяти DRAM для предвыборки при буферизованном считывании;
  - буферизованная пакетная обратная запись в ОП для снижения вероятности конфликтов;
- изменение в иерархии памяти:
  - предвыборка команд непосредственно в кэш L1 (L2 для семейства K8);

  - кэш-память L3 (частично ассоциативная, 32 входа, объем не менее 2 MiB), разделяемая между ядрами ЦП (каждое из которых имеет 512 KiB эксклюзивного кэша L2), размещенными на одном кристалле;
  - расширяемая кэш-память L3 (например, 6 MiB для процессора Shanghai, технология 45 нм);
- усовершенствования в управлении адресным пространством:
  - два независимых 64-разрядных контроллера памяти, каждый со своим физическим адресным пространством. Это создает возможность лучшего использования доступной пропускной способности при случайных обращениях к памяти, инициируемых многопроцессными задачами. В K8 («interleaved» design) два канала данных были связаны с единственным адресным пространством;
  - буфер TLB (Tagged Lookaside Buffers) большего размера (1 GiB) и новый страничный TLB (128 входов, 2 MiB);
  - адресация памяти в 48 разрядов, что теоретически обеспечивает доступ к 256 TiB;
  - зеркалирование (mirroring) памяти, улучшенная адресация строк ОП (RAS);
  - технология AMD-V (virtualization, или вложенные таблицы страничной адресации), что снижает время переключения на 25 %.

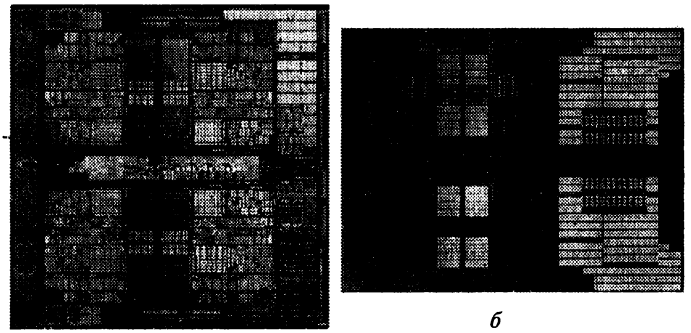
Усовершенствования в интерфейсах:

- поддержка режима повторных пересылок в HyperTransport;
- обеспечение для HyperTransport 3.0 восьми связей «точка—точка» для каждого процессорного разъема.

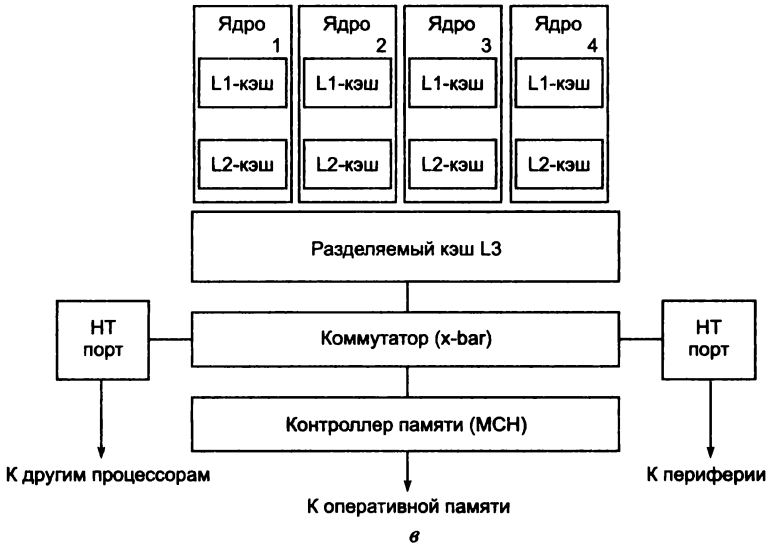
Другие усовершенствования:

- автоматическая модуляция частоты тактового генератора;
- повышение скважности импульсов тактового генератора;
- поддержка сопроцессоров через разъемы HTX или свободные гнезда ЦП посредством интерфейса HyperTransport.

Barcelona. В августе 2006 г. при запуске Socket F (известного так же, как Socket 1207) для двухъядерного процессора Opteron AMD объявила, что закончено проектирование четырехъядерного процессора на основе Opteron. Новый 4-ядерный процессор (Quad-Core Opteron) выпущен AMD в конце 2007 г. и является конкретной реализацией архитектуры K10. Первоначально спроектирован для серверов и рабочих станций и имеет кодовое название «Barcelona». В основе лежат процессоры



а *б*



в

Рис. 3.37. Четырехъядерные процессоры AMD K10: *а* — общий вид кристалла Barcelona (65 нм); *б* — Shanghai (45 нм); *в* — блок-схема Barcelona

Opteron 2348 и 2350 с частотами ядра в 1,9 и 2,0 ГГц, с возможным дальнейшим ее повышением (рис. 3.37).

Как и двухъядерные архитектуры, 4-ядерные на некоторых сложных приложениях (параллельные процессы, интенсивные вычисления с ПЗ, обработка/кодирование графики или видео, 3-мерный рендеринг и пр.) демонстрируют существенное (многократное) повышение производительности, в то время как для более простых это повышение несущественно или вообще отсутствует.

Выполненное AMD тестирование показало повышение быстродействия на 65—70 % сравнительно с двухъядерными процессорами Opteron (Santa Rosa, Opteron 2222 — 3,0 ГГц).

В ядре AMD Barcelona существенно усилен блок SSE ЦП Opteron, что дает по различным методам измерения более чем удвоение эффективности. Следующая таблица (табл. 3.8) дает сравнение характеристик Barcelona с предшественниками.

Таблица 3.8. **Сравнительные характеристики ЦП Barcelona**

Характеристика	Более ранние ЦП	Barcelona
Длина обрабатываемых слов в SSE, бит	64	128
Скорость выборки команд, байт/цикл	16	32
Скорость кэша данных, бит/цикл	2 × 64	2 × 128
Скорость обмена L2-кэша и контроллера памяти, бит/цикл	64	128
Емкость планировщика ПЗ	36 × 64-битовых операций	36 × 128-битовых операций

Процессоры Barcelona используют три уровня кэш-памяти — L1 и L2 (64 и 512 Кбайт соответственно выделены каждому ядру, как это и было у ЦП Opterons и Athlon), тогда как L3-кэш (от 2 Мбайт) совместно используется всеми ядрами.

ЦП Barcelona будет выпускаться по технологии AMD 65 нм SOI (кремний на изоляторе), позволяющей использовать более низкие напряжение питания и рассеиваемую мощность. Первые образцы ЦП Barcelona потребляют около 95 Вт. Здесь используется улучшенная технология PowerNow! (позволяющая отдельным ядрам оперировать на той тактовой частоте, которая соответствует вычислительной нагрузке в данный момент).

AMD прилагает большие усилия по обеспечению обратной совместимости, обеспечивая возможность установки новых процессоров на существующие системные платы с интерфейсами AM2 (при несущественном обновлении BIOS), хотя только интерфейс AM2+ позволяет раскрыть все возможности новых процессоров.

Shanghai — серверные процессоры, замещающие Barcelona (рис. 3.37, *б*) и повторяющие ее архитектуру. С переходом на техпроцесс 45 нм здесь осуществляется не только снижение стоимости производства микрочипов, снижение их энергопотребления, но также и возможность повышения рабочей частоты

процессоров. При одинаковой с решениями Barcelona рабочей частоте процессоры Shanghai превосходят их по производительности в среднем на 35 %, и в то же время потребляют на 35 % меньше электроэнергии. Это достигается как повышением объема кэш-памяти третьего уровня с 2 до 6 Мбайт, так и повышением числа выполняемых за один такт инструкций.

Объем кэш-памяти 1—2 уровней остается таким же, как в Barcelona AMD (128 Кбайт L1 и 512 Кбайт L2). Кроме того, здесь реализована технология AMD-V (повышенная эффективность виртуализации) и поддерживается система оперативной памяти DDR2-800.

Рабочая частота ЦП варьируется в пределах от 2,3 до 2,7 ГГц. Устройства поддерживают разъем Socket F (1207) и имеют интегрированный контроллер оперативной памяти стандарта DDR2.

Ядро Deneb (Shanghai) представляет собой 45нм процессор поколения K10.5. Состоит из ≈758 млн транзисторов и имеет площадь в 243 мм^2 (против 463 млн и 283 мм^2 соответственно у 65нм Barcelona и 731 млн и 246 мм^2 у Intel Nehalem).

Основная цель — повышение частот процессорной линейки Phenom, снижение TDP, а также себестоимости производства. Первые процессоры на ядре Deneb выпущены AMD 8 января 2009 г. под именем Phenom II X4 (модели 920 и 940 Black Edition).

Propus. Представляет собой аналог процессора Deneb, но без кэша L3.

Кроме того, к архитектуре K10.5 относятся также процессоры:

- Thuban (технология 45 нм SOI), количество транзисторов: 904 млн, площадь процессора: 346 мм^2, шесть ядер;
- Zosma (45 нм SOI) — четыре ядра (Thuban с двумя отключенными ядрами);
- Heka (45 нм SOI) — три ядра (Deneb с одним отключенным ядром);
- Callisto (45 нм SOI) — два ядра (Deneb с двумя отключенными ядрами).

Phenom — процессоры AMD для настольных ПК, основанные на микроархитектуре K10. Первыми на рынке появятся четырехъядерные процессоры Agena FX и Agena. Процессоры Phenom FX в исполнении Socket F+ предназначены для установки в системы класса Quad FX с двумя процессорными разъемами.

Четырехъядерные процессоры Phenom X4 обладают схожими с Phenom FX техническими характеристиками: частоты лежат в

диапазоне 2,2—2,4 ГГц, объем кэша второго уровня равен 4 × 512 Кбайт, объем кэша третьего уровня равен 2 Мбайт, мощность — не более 89 Вт. Все четырехъядерные процессоры AMD поколения K10 будут иметь частоту шины HyperTransport 3.0 порядка 3,2—3,6 ГГц. Эти же процессоры смогут работать и в более старых материнских платах, при этом частота шины HyperTransport будет понижена до версий 1.0—2.0 (см. табл. 4.3).

Переходя к двухъядерным процессорам Phenom X2 (Kuma), следует отметить, что их частотные характеристики будут гораздо выше соответствующих характеристик четырехъядерных. В частности, частоты ядра достигнут 2,8 ГГц, а частота шины HyperTransport 3.0 — до 4,2 ГГц.

На частотах до 2,6 ГГц процессоры Phenom X2 сохраняют энергопотребление не более 65 Вт, на частоте 2,8 ГГц — до 89 Вт. AMD планирует выпустить процессоры Phenom X2 с пониженным энергопотреблением (45 Вт). Частоты будут охватывать диапазон 1,9—2,3 ГГц, частота шины HyperTransport 3.0 — от 2,8 до 3,4 ГГц.

Для нижних ценовых диапазонов AMD представляет семейство Athlon 64 X2, в которое войдут 2-ядерные Rana, и Sempron (одноядерные Spica).

Двухъядерные процессоры Rana, в которых по сравнению с Kuma отсутствует кэш-память L3, сохраняют торговую марку Athlon X2. Наличие кэша третьего уровня дает процессорам право носить имя Phenom. Единственный пока представитель ядра Rana будет работать на частоте 2,2 ГГц, иметь 2 × 512 Кбайт кэша второго уровня, энергопотребление 65 Вт. Частота шины HyperTransport 3.0 составляет 3,2 ГГц.

AMD Fusion — кодовое название проектов процессоров следующего (после K10) поколения, продукт сотрудничества между AMD и ATI, начатого в 2006 г.

Архитектура Fusion представляет собой гетерогенный многоядерный микропроцессор, комбинирующий ядра обычных процессоров (центральный процессор — ЦП, CPU) и графической обработки (графический процессор — ГП, GPU).

Термин «Fusion» (от *англ.* — плавление, сплав) отражает тенденцию интеграции в процессоры AMD функций, ранее исполнявшихся другими модулями чипсета (в архитектуре K8 это были контроллер памяти и порты/линки HyperTransport, в K10 — графический процессор и т. д.).

Процессоры серии Fusion будут базироваться на новой модульной методологии проектирования, именуемой «M-SPACE», которая предоставляет возможность построения различных сочетаний процессорных ядер в изделиях, настраиваемых на различные приложения в рамках единой архитектуры. При этом, например, ядро ГП может быть переработано или изменено, без необходимости перестройки ядер ЦП.

Bobcat — процессор архитектуры, промежуточной между K10 и Fusion. Представляет собой существенно упрощенное ядро x86, позволяющее осуществлять обработку программ x86 при энергопотреблении между 1 и 10 Вт. Предполагается, что Bobcat будет использоваться в устройствах UMPC, OLPC, КПК и других малогабаритных изделиях.

Bulldozer — кодовое название, данное ядру ЦП архитектуры Fusion, опирающегося на методологии проектирования, ядро которого будет иметь энергоемкость от 10 до 100 Вт. Ядро Bulldozer предполагается устанавливать на общем чипе с одним или более ядрами графической обработки, поддерживающими стандарты DirectX (предполагается, что это будут процессоры Radeon). Кроме того, планируется архитектура с кодовым именем Sandtiger, предусматривающая объединение от 8 до 16 ядер Bulldozer и предназначенная для серверов и высокопроизводительных вычислений (HPC applications). В ядрах Bulldozer также будет реализована следующая итерация Streaming SIMD Extensions (SSE), со 170 новыми командами, названными SSE5.

Предполагается также, что ядра Bulldozer в многоядерных архитектурах будут наделены способностями переключения незанятых исполнительных элементов (арифметико-логические устройства, ALU и блоки операций с ПЗ, FPU) от одного ядра к другому для выравнивания загрузки элементов процессора.

Процессоры IBM POWER и PowerPC

POWER (**P**erformance **O**ptimization **W**ith **E**nhanced **R**ISC) — серия микропроцессоров, поддерживающих RISC-систему команд, разработанных в фирме IBM. Под тем же именем фигурирует семейство ЭВМ на основе RISC-архитектуры. Процессоры POWER используются как ЦП во многих серверах производства IBM, мини-компьютерах, рабочих станциях и суперЭВМ (табл. 3.9).

Таблица 3.9. **Характеристики некоторых процессоров IBM POWER и PowerPC**

Тип процессора	Год выпуска	Количество транзисторов, млн	Ядро, мм^2	L1-кэш, Кбайт	L2-кэш, Кбайт	Размер минимальной структуры, мкм	Тактовая частота процессора, МГц	Потребляемая мощность, Вт
POWER 4 (два ядра) POWER 4+	2001—2002	174	414 —267	64×2	1,4 Мбайт×2+ L3 (32 Мбайта)	0,18—0,13 (SOI)	1,1— 1,3 ГГц	125
POWER 5	2005	276	389		1,8 Мбайт+ L3 (36 Мбайт)	0,13 (SOI)	1,5—1,9 ГГЦ	
PowerPC 601	1991	2,8	121	32+32	Внеш.	0,5	50, 66, 80	
603	1994	1,6	85,4	4+4	Внеш.	0,5	80	2,5
603e 603ev	1995	2,8	98—79		Внеш.	0,5 —0,35	80, 100, 166, 200	3,2
604	1995	5,1	148	32+32	Внеш.	0,35	107, 180, 200, 332	
620	1995	7	311	32+32	Внеш. (до 128 Мбайт)	0,5		
750 750cxl	1998 2001			32+32	256	0,25; 0,22; 0,18	200, 500, 600	
970FX	2005	52	113—66	64+32	512—2 Мбайта	0,13—0,09	1,8 ГГц	42

Архитектура POWER (рис. 3.38) во многих отношениях представляет собой традиционную RISC-архитектуру. Она сохраняет наиболее важные особенности RISC: фиксированную длину команд, архитектуру «регистр—регистр», простые способы адресации, большой регистровый файл, а также трехоперандный формат инструкций. Однако POWER имеет и несколько дополнительных свойств, отличающих ее от других RISC-архитектур.

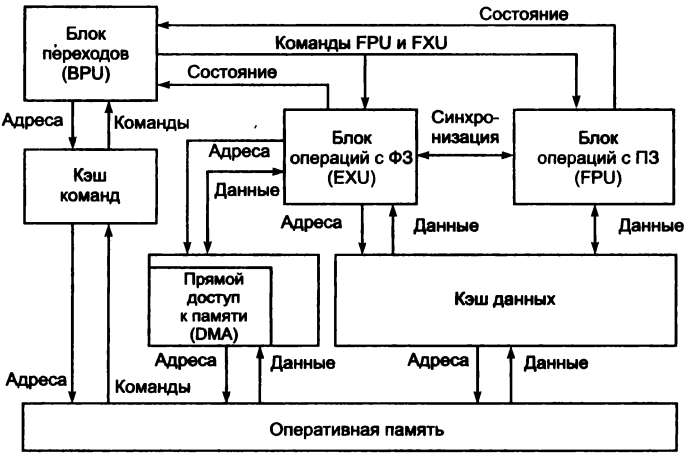


Рис. 3.38. Обобщенная схема архитектур POWER и PowerPC

Набор команд POWER изначально был основан на идее суперскалярной обработки. Команды распределяются по трем независимым исполнительным устройствам (блокам) — переходов, целочисленной и вещественной арифметики. Для сокращения времён выполнения были введены команды группового (векторного) чтения и записи данных, которые обеспечивали пересылку содержимого нескольких регистров в память (и обратно) с помощью единственной команды. Архитектура переходов POWER была организована с учетом их предварительного просмотра и методики свертывания.

Проект POWER восходит к давнишним процессорам IBM 801, которые во всех учебниках приводятся в качестве примера «подлинной RISC-архитектуры».

В 1974 г. IBM приступила к проекту, целью которого было создать крупную телефонную сеть с цифровым компьютерным управлением, в которой должно обрабатываться не менее

300 звонков в секунду. По оценкам специалистов, для поддержки ответов в реальном масштабе времени требовалось около 20 тыс. машинных команд на каждый звонок, вследствие этого скорость процессора должна была достигать 12 Mips (млн команд в секунду). Эти требования были чрезвычайно высокими для тех времен, однако было ясно, что здесь следовало использовать упрощенную конструкцию, поддерживающую только операции ввода-вывода, ветвления, суммирования «регистр—регистр» и пересылки данных между ОП и регистрами.

Хотя в 1975 г. этот проект телефонной сети был закрыт, в результате были разработаны многообещающие процессоры 801. Эти процессоры активно использовались в продуктах фирмы IBM, однако не были широко известны до появления IBM PC/RT в середине 1980-х гг. (не показавшей, однако, высокой производительности).

Основными недостатками 801 являлись:

- все команды выполнялись за один процессорный цикл, это исключало применение операций с ПЗ;
- хотя декодер команд предусматривал конвейер, в одноцикловой архитектуре не удавалось достигнуть суперскалярного эффекта.

На следующем этапе развития этих процессоров («America» Project, 1985—1986 гг.) IBM использовала новые алгоритмы, позволяющие осуществлять за один цикл умножение и деление чисел двойной точности. Кроме того, блок операций с ПЗ выделен из декодера и блоков ФЗ, и поэтому декодер может пересылать для исполнения команды на блоки ПЗ (FPU) и ФЗ (ALU) одновременно.

В дальнейшем здесь использовался комплексный декодер команд, способный одновременно осуществлять выборку одной команды, декодирование другой и пересылку двух команд в АЛУ и БПЗ (черты суперскалярного процессора, одного из первых в истории отрасли).

Процессор содержал 32 × 32-битовых целочисленных регистров и 32 × 64-битовых регистров чисел с ПЗ, блок обработки переходов также включал ряд внутренних регистров, в том числе СчАК.

Другой интересной чертой архитектуры была виртуальная адресация, отображавшая все адреса в памяти на 52-битовое пространство. Тем самым приложения могли совместно

использовать 32-битовое адресное пространство, с выделением каждым из них 32-битового блока памяти.

POWER и RS/6000. Первый компьютер фирмы IBM, использовавший архитектуру POWER, был выпущен в 1990 г. Архитектура получила наименование «RISC System/6000» или RS/6000. RS/6000 включали два типа машин: рабочие станции (POWERstation) и серверы (POWERserver). Процессор RS/6000, называвшийся RIOS (позднее — RIOS I или POWER1), состоял из 11 чипов — блок операций с ФЗ, блок ПЗ, кэш команд, 4 кэша данных, контроллер памяти, 2 блока ввода-вывода и тактовый генератор.

Реализация RIOS на одном чипе, RSC (для «RISC Single Chip»), была разработана для дешевых моделей RS/6000, первые ЭВМ на базе RSC были выпущены в 1992 г.

POWER2 (развитие RIOS/POWER1) был выпущен в 1993 г., и основным новшеством было включение дополнительного кэша на 256 KiB, блоков операций с ПЗ (128-разрядная арифметика) и ФЗ. Кроме того, был добавлен ряд новых команд:

- загрузка 4-словных операндов. При этом два рядом расположенных числа двойной точности загружаются в два смежных регистра операций с ПЗ;
- аппаратурная реализация извлечения квадратного корня;
- преобразования чисел ПЗ в формат ФЗ.

В 1996 г. была выпущена реализация POWER2 на одном чипе — P2SC («POWER2 Super Chip»).

POWER3 был выпущен в 1998 г. Здесь был реализован 64-разрядный набор команд POWER, включая альтернативные инструкции набора команд (бывшие в то время). ЦП включал также 2 блока ПЗ, 3 блока ФЗ, 2 декодера команд и 2 блока записи-выборки данных. Все последующие ЦП POWER реализуют полные наборы 64-разрядных команд PowerPC и POWER.

POWER3 и последующие процессоры серии POWER реализуют 64-разрядную архитектуру PowerPC. POWER3 и выше не поддерживают какие-либо старые команды POWER, которые были удалены из набора команд, когда был выпущен PowerPC или какие-либо из расширения POWER2, например `lfq` или `stfq`.

POWER4. IBM представила процессор POWER4 (первый из GIGA-Series) в 2001 г. Это был полностью 64-разрядный ЦП, поддерживающий все 64-битовые команды PowerPC, а также AS/400-расширение, и использовавшийся как в системах RS/6000 так и в AS/400, заменяя процессоры POWER3 и RS64. Был разра-

ботан также новый набор команд, получивший название PowerPC 2.00 ISA, в котором было добавлено несколько новых инструкций. Количество команд в PowerPC 2.00 ISA превышало 100, причем многие их них являлись модификациями друг друга.

POWER4 уникален уже тем, что в нем даже один кристалл представляет собой мультипроцессорную систему — в одном корпусе содержится два 64-разрядных микропроцессора. Архитектуру кристалла POWER4 отличает несколько современных решений — суперскалярная структура, внеочередное исполнение команд, большая кэш-память на кристалле, специализированный порт для основной памяти, а также высокоскоростные линки для объединения микропроцессоров в системы с архитектурой распределенной разделяемой памяти.

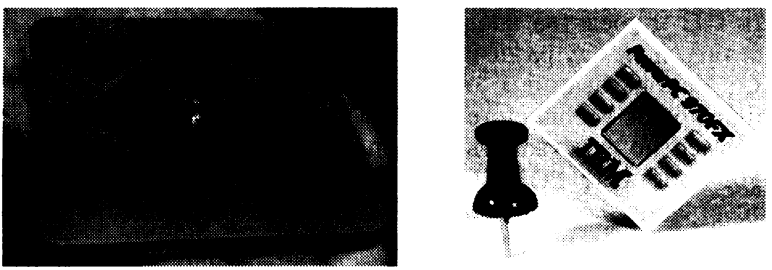
Каждый процессор POWER4 имеет два конвейерных блока для работы с 64-разрядными операндами с плавающей точкой, выбирающих на исполнение по пять команд каждый, и два блока для работы с памятью. Процессоры содержат раздельную кэш-память команд и данных 1-го уровня объемом по 64 Кбайт каждая. Кроме того, имеется разделяемая (общая) кэш-память 2-го уровня на кристалле (объемом 1,4 Мбайт) и внешняя кэш-память 3-го уровня (32 Мбайт). Совместный доступ к внешней кэш-памяти выполняется по технологии DSI (Distributed Switch Interconnect). Для создания мультипроцессорных конфигураций имеются специальные линки с высокой пропускной способностью. Наряду с параллелизмом на уровне команд процессор использует параллелизм на уровне потоков (нитей, thread).

POWER4 изготавливался по технологии 0,18 мкм SOI с несколькими слоями медной металлизации на кристалле площадью около 400 мм^2. Базовое напряжение питания POWER4 равнялось 1,5 В. Тактовая частота кристалла, содержащего 174 млн транзисторов, могла составлять 1,1 или 1,3 ГГц. Такие параметры обеспечивал технологический процесс CMOS-8S2, представляющий собой дальнейшее развитие известного процесса CMOS-8.

Одной из отличительных особенностей POWER4 стало наличие кэш-памяти 2-го уровня, разделяемой двумя процессорами кристалла, а также внешними процессорами других кристаллов через интерфейсы (линки) по 128 бит, работающие на тактовой частоте более 500 МГц, что обеспечивает пропускную способность свыше 10 Гбайт/с. При объединении четырех кристаллов и их специальном размещении проводники линков могут быть

достаточно короткими и прямыми, что важно при работе на высоких тактовых частотах.

Физически кэш-память 2-го уровня объемом около 1,4 Мбайт состоит из нескольких одинаковых блоков, доступ к которым выполняется через коммутатор с пропускной способностью на уровне 100 Гбайт/с. Протокол когерентности обеспечивает размещение данных, поступивших по линкам, в том блоке кэш-памяти, который последним использовался для размещения данных. Порт кристалла POWER4, предназначенный для подключения кэш-памяти 3-го уровня объемом до 32 Мбайт, имеет ширину 128 разрядов для каждого из двух направлений пересылки данных, обеспечивая пропускную способность на уровне 13—14 Гбайт/с. Скорость передачи данных между кэш-памятью 3-го уровня и основной памятью может достигать 12,8 Гбайт/с. Каждый сдвоенный процессор POWER4 упакован в керамический мультипроцессорный модуль (размером 4,5 × 4,5 дюйма), наподобие изображенного на рис. 3.39, *а*.



а *б*

Рис. 3.39. Керамический модуль POWER5 MCM с четырьмя процессорами и четырьмя модулями L3-кэша по 36 Мбайт (*а*); PowerPC 970FX (*б*)

POWER5 был выпущен в 2004 г., в него добавлен ряд новых команд, с учетом которых получен новый набор команд PowerPC 2.02 ISA. Другие основные отличия:

- двухъядерный процессор, который поддерживает многопоточную (multithreading) обработку данных (2 потока), так что он воплощает в себе 4 логических (с точки зрения операционных систем) процессора;
- частота более 1,9 ГГц;
- контроллер памяти на чипе;
- бóльший объем кэша L2;
- улучшенное управление энергопотреблением;

- технологии Hypervisor (virtualization technology) и eFuse (hardware re-routing around faults).

POWER5 — это девятое поколение 64-разрядной RISC-архитектуры IBM. И хотя в ней использованы многие решения, появившиеся еще в POWER4, разработчики нового процессора подчеркивают, что его нельзя рассматривать как всего лишь модификацию предшественника. Дело в том, что значительно изменилась конструкция самого кристалла, что позволило создавать более эффективные суперскалярные комплексы. В POWER5 реализована одновременная многопоточность, при которой процессорное ядро может запрограммировать порядок параллельного выполнения команд из нескольких потоков.

Технологии виртуализации предусматривают создание логических разделов (logical partitioning) и выделение микроразделов (Micro-Partitioning). Для каждого ЦП могут быть созданы до 10 логических разделов (LPAR — logical partition), и в результате наиболее крупные системы могут поддерживать до 256 копий независимо выполняющихся ОС. Оперативная память, мощность ЦП и средства ввода-вывода могут динамически перераспределяться между разделами.

С использованием виртуальной векторной архитектуры (Virtual Vector Architecture — ViVA) несколько ЦП POWER5 могут совместно действовать как один векторный процессор.

На кристалле POWER5 реализовано 276 млн транзисторов, которые занимают площадь 389 мм^2. Изготовляется по технологии 0,13 мкм с применением медных проводников и «кремний-на-изоляторе» (SOI), что позволяет достичь большей производительности и снизить энергопотребление.

Для сравнения: у кристалла POWER4, изготовлявшегося по технологии 0,18 мкм, площадь равна 414 мм^2, а у его модификации POWER4+, выпущенной в конце 2002 г., она уменьшилась до 267 мм^2 благодаря переходу на технологию 0,13 мкм. На кристалле POWER5 размещены два одинаковых процессорных ядра и общая кэш-память 2-го уровня (L2) объемом 1,875 Мбайт, выполненная в виде трех отдельных блоков, у каждого из которых имеется свой отдельный контроллер (для POWER4 объем кэш-памяти 2-го уровня составлял 1,5 Мбайт). Физический адрес данных определяет, в каком блоке кэш-памяти 2-го уровня находятся данные. Каждое из процессорных ядер может независимо обращаться к любому из трех контроллеров кэш-памяти 2-го уровня. Тактовая частота POWER5 составляет от 1,5 до 1,9 ГГц.

Одно из главных новшеств в конструкции кристалла по сравнению с POWER4 — это интегрированная кэш-память 3-го уровня объемом 36 Мбайт. Стоит отметить, что в предыдущем поколении процессора кэш-память на 32 Мбайт располагалась вне кристалла. Благодаря переносу L3 ближе к процессорному ядру при отсутствии нужных данных в кэш-памяти 2-го уровня процессору намного реже придется обращаться за пределы кристалла, за счет чего в SMP-системе снижается интенсивность обмена данными между кристаллами. В результате, если системы на базе POWER4 не могли масштабироваться до числа процессоров свыше 32 (это приводило к резкому увеличению задержек из-за увеличения межпроцессорного трафика), то POWER5 обеспечивает построение 64-процессорных конфигураций. Кроме того, переход на технологию 0,13 мкм позволил конструкторам POWER5 интегрировать в него также контроллер памяти, что дополнительно сократило латентность считывания данных из памяти.

Структура конвейера команд POWER5 осталась полностью идентичной той, что применялась в POWER4, причем не изменились и величины задержек.

Четыре кристалла POWER5 (восемь процессорных ядер) вместе с четырьмя кристаллами кэш-памяти 3-го уровня упаковываются в многокристальный модуль MCM (MultiChip Module) размером 95 × 95 мм (см. рис. 3.39, *а*), который в итоге объединяет восемь процессоров. Может использоваться также двухкристальный модуль Double chip Module (DCM).

Многослойный керамический корпус MCM содержит магистрали, соединяющие микросхемы между собой, а также с модулями кэш-памяти и высокоскоростным коммутатором для связи с удаленными процессорами. Четыре микросхемы POWER4, образующие восьмипроцессорную конфигурацию, расположены в MCM под углом 90° друг относительно друга, что позволяет минимизировать длину шин расширения, соединяющих микросхемы. Шины расширения связывают между собой и модули MCM. В этих шинах используется специальная технология волновой конвейеризации (wave pipelining), обеспечивающая очень низкие величины задержек. Пропускная способность каждой шины превосходит 8 Гбайт/с, и соответственно MCM с четырьмя шинами расширения, ведущими к другим MCM, будет иметь суммарную пропускную способность свыше 32 Гбайт/с. Такую пропускную способность обменов между MCM будет иметь 32-процессорная SMP-конфигурация из четырех модулей MCM.

Шины расширения, кроме собственно межмодульных шин, включают выделенные шины для организации ввода-вывода и создания NUMA-конфигураций.

Для объединения нескольких MCM используется смешанная инфраструктура из шин и распределенного коммутатора — каждый модуль имеет четыре логические шины, позволяющие построить кольцо из четырех MCM. Соединяющая MCM восьмибайтовая шина работает на половине тактовой частоты процессора и обеспечивает пропускную способность 4 Гбайт/с. Для соединения шин применяется усовершенствованная версия распределенного коммутатора, разработанного для POWER4. Число шин увеличилось, что повысило суммарную пропускную способность по сравнению с POWER4.

При необходимости пара ядер может быть отключена, так что оставшиеся будут совместно использовать возможности системной шины и кэш-памяти L3.

POWER6 объявлен 21 мая 2007 г. Это двухъядерный чип, работающий на частоте 4,7 ГГц и изготовленный по технологии 65 нм. Здесь используется более совершенная технология межъядерной коммуникации и, кроме того, энергопотребление не превышает показателей POWER5 при двукратном повышении производительности. В процессорах добавлены возможности VMX, а также используется новое поколение ViVA (ViVA-2).

POWER7 разрабатывается в настоящее время IBM и является первым из Peta-Series. Предполагается, что он будет выпущен около 2010 г., но уже выбран DARPA как возможный процессор для проекта Peta-Flop SuperComputer (суперкомпьютер на 1000 Гфлопс).

PowerPC. В 1991 г. IBM пришла к выводу, что необходимо расширить рынок для POWER путем развития и модификации архитектуры процессоров. С этой целью она пригласила к сотрудничеству Apple для разработки семейства микропроцессоров POWER-архитектуры на одном чипе. Apple, в свою очередь, обратилась к Motorola (как к своему давнему поставщику ЦП) с предложением присоединиться к альянсу, который в дальнейшем получил название «AIM alliance» (от Apple, IBM, Motorola).

601. В результате к 1993 г. была разработана архитектура PowerPC, видоизменение POWER (первый чип назывался PowerPC 601 и базировался на RSC). Устройство имело 32-разрядную внутреннюю структуру и размещалось на кристалле площадью 121 мм^2, общее количество транзисторов в нем составля-

ло 2,8 млн. Были выпущены модели с тактовыми частотами 50, 66, 80 МГц.

Три исполнительных устройства (см. рис. 3.36) работали параллельно и в значительной степени независимо. Благодаря суперскалярной архитектуре за один такт могло одновременно выполняться до трех команд. Процессор оперировал с 8-, 16-, 32-разрядными целыми, а также с 32- и 64-разрядными числами с плавающей запятой. На кристалле также располагались раздельная кэш-память команд и данных объемом 32 Кбайт (16+16) и устройство управления памятью. Процессор имел 64-разрядную шину данных и 32-разрядную шину адресов.

603. (1994 г.). Структура этого процессора копировала структуру PowerPC 601, но он был размещен на кристалле площадью 85,1 мм^2 и, будучи изготовлен по технологии 0,5 мкм, содержал 1,6 млн транзисторов. Потребляемая мощность процессора, работавшего на частоте 80 МГц, не превышала 2,5 Вт. Данное устройство во многом было упрощенной версией своего предшественника. Оно выполняло не более двух команд за такт, а размер кэш-памяти был уменьшен до 8 Кбайт (4+4).

В 1995 г. была выпущена улучшенная версия этого процессора под маркой PowerPC 603e. Процессор, производимый по технологии норм 0,5 мкм, размещался на кристалле площадью 98 мм^2 и содержал 2,8 млн транзисторов. Первоначально он выпускался с тактовыми частотами 80 и 100 МГц. На частоте 100 МГц он потреблял примерно 3,2 Вт. Данные кристаллы нашли широкое применение в качестве встраиваемых микропроцессоров. Их тактовые частоты составляли 100, 166 и 200 МГц.

Еще одним новшеством в этом процессоре стало появление трех энергосберегающих режимов, а также устройства динамического управления энергопотреблением. При изготовлении по технологии 0,35 мкм площадь кристалла уменьшается до 79 мм^2.

602. Одновременно с PowerPC 603e был выпущен ЦП для портативных компьютеров — PowerPC 602. Он был выполнен по технологии 0,5 мкм и на кристалле площадью 50 мм^2 было расположено до 1 млн транзисторов. Шины адреса и данных устройства были мультиплексированы (сначала процессор выводил на шину адрес и специальным сигналом информировал об этом все устройства, а затем по той же шине передавал данные). Доступ к памяти организован в пакетном режиме, что делает снижение производительности процессора не столь ощутимым.

К трем исполнительным устройствам добавлен блок вычисления адреса (БВА, AGU — Address Generation Unit) доступа к памяти. Тем не менее архитектура процессора была упрощена по сравнению с PowerPC 603. Так, устройство для операций вещественной арифметики могло обрабатывать только 32-разрядные числа (ранее — 64-разрядные). Количество выполняемых за один такт операций сократилось до одной, что уменьшает размер блока декодирования и упрощает схему предсказания переходов. Кроме того, были исключены сложные графические и строковые операции, благодаря чему упрощается БВА. Операции пересылки были оптимизированы для выполнения за один такт. Объемы кэш-памяти команд и данных составляли теперь по 2 Кбайт (впоследствии 4 Кбайт) каждый. Процессор был снабжен быстрым механизмом защиты памяти и имел те же режимы энергосбережения, что и PowerPC 603.

603ev (1996 г.) — процессор с тактовой частотой 166, затем 180 и 200 МГц, выпущенный по технологии 0,35 мкм. В нем были усовершенствованы блоки операций деления и управления кэш-памятью.

604 (1995 г.) — последний 32-разрядный процессор этого семейства. Процессор производился по технологии 0,35 мкм и работал на тактовых частотах 100, 120, 133 и 150 МГц. Усовершенствованным вариантом этого процессора стал PowerPC 604e, выполненный с использованием того же технологического процесса и содержащий 5,1 млн транзисторов на кристалле площадью 148 мм^2. Были выпущены микросхемы, работающие на тактовых частотах 167, 180, 200, а позднее — 332 МГц. Внутренняя кэш-память была увеличена вдвое — по 32 Кбайт для команд и для данных.

620 (1995 г.) — первый 64-разрядный процессор семейства, который предназначался для рабочих станций и высокопроизводительных серверов. Этот кристалл имел шесть уже независимых исполнительных устройств и встроенную кэш-память 64 Кбайт (32+32), производился по технологии норм 0,5 мкм и на кристалле площадью 311 мм^2 содержал 7 млн транзисторов. В PowerPC 620 использовалась четырехконвейерная суперскалярная архитектура с шестью исполнительными устройствами, в числе которых было три блока целочисленной арифметики, один блок операций с плавающей точкой, блок загрузки/сохранения и блок переходов. За один такт процессор мог выполнять до четырех команд. Шинный интерфейс этого процессора включал унифи-

цированную внутреннюю поддержку кэш-памяти 2-го уровня объемом до 128 Мбайт.

750 (1998 г.) — первый микропроцессор IBM с медными соединениями (см. рис. 1.22). Одной из особенностей PowerPC 750 была схема кэширования, отличавшаяся от используемой в предыдущих версиях PowerPC. Микросхема имела выделенную шину, с помощью которой кэш-память 2-го уровня (емкостью 0,5; 1 или 2 Мбайт) подсоединялась непосредственно к кристаллу (Back Side Bus, см. рис. 4.13), а не через системную шину. При этом выделенная шина работала на удвоенной тактовой частоте (системной шины). Это допускало использование микросхем памяти типа SRAM 233 МГц. Кристаллы PowerPC 750, работающие на тактовых частотах от 200 до 500 МГц, были выполнены по технологиям 0,25 (PID8t) и 0,22 (PID8p) мкм.

Версия микропроцессора PowerPC 750CX отличается встроенной 2-входовой наборно-ассоциативной кэш-памятью объемом 256 Кбайт. Отметим, что 8-входовая наборно-ассоциативная кэш-память 1-го уровня для команд и данных имеет объем по 32 Кбайт каждая. Кристалл выполнен с учетом проектных норм 0,18 мкм, с шестислойной медной металлизацией. Рабочие тактовые частоты составляют 366, 400 и 466 МГц.

В феврале 2001 г. IBM сообщила о выпуске PowerPC 750CXe, который содержал 256 Кбайт встроенной кэш-памяти 2-го уровня и выпускался с использованием медных проводников и по технологии 0,18 мкм. IBM начала массовое производство PowerPC 750CXe с тактовыми частотами 400, 500, 600 МГц. Сейчас в эту линейку входят также микросхемы PowerPC 750CXr с тактовой частотой 533 МГц.

PowerPC 970FX (кодовое название Altair) — один из наиболее производительных микропроцессоров в линейке PowerPC, с тактовой частотой до 2,5 ГГц и пиковой производительностью до 10 GFLOPS. PowerPC 970FX (рис. 3.39, *б*) — не первый 64-разрядный микропроцессор данной архитектуры. Как уже отмечалось, первым был появившийся в 1998 г. PowerPC 620, но он оказался неудачным в коммерческом плане, и последующие 64-разрядные PowerPC-совместимые микропроцессоры в настольных системах никогда не применялись.

Считается, что PowerPC 970 представляет собой фактически упрощенную версию процессора POWER4. Последний содержит два процессорных ядра, 1,5 Мбайт кэш-памяти 2-го уровня, контроллеры кэш-памяти 3-го уровня и при площади 415 мм^2 (для

технологии 0,18 мкм) имеет 170 млн транзисторов. Переход на технологию 0,13 мкм в POWER4+ существа дела не меняет. По сравнению с POWER4 в PowerPC 970 резко уменьшилось число транзисторов (до 52 млн), площадь (113 мм^2 для технологии 0,13 мкм или 66 мм^2 при 90 нм) и энергопотребление (42 Вт при частоте 1,8 ГГц у PowerPC 970 против 125 Вт у POWER4 с тактовой частотой 1,3 ГГц). Были упразднены одно процессорное ядро и названные выше контроллеры, объем кэш-памяти 2-го уровня был уменьшен до 512 Кбайт. Одновременно была существенно увеличена длина конвейеров, чтобы легче было поднимать тактовую частоту. Наконец, в PowerPC 970 была добавлена поддержка мультимедийного расширения системы команд AltiVec.

Основу успеха PowerPC 970FX заложили отличные характеристики POWER4, в частности, рекордное для RISC-микропроцессоров максимальное число команд, выполняемых за один такт, включая одну команду перехода.

PowerPC 970 как и POWER4 — полностью 64-разрядный микропроцессор, имеющий 64-разрядные виртуальные адреса (64-разрядное «плоское» адресное пространство), 64-разрядные внутренние магистрали данных, 64-разрядные регистры общего назначения. Под адреса реальной памяти в обоих процессорах отводится 42 бита.

Все внутренние элементы PowerPC 970 — 64-разрядные, но регистры AltiVec-расширения и соответствующие пути данных — 128-разрядные. В PowerPC 970 имеется по 32 регистра ФЗ, AltiVec-регистра и регистра ПЗ плюс по 48 регистров каждого типа для переименования. Объем кэш-памяти команд 1-го уровня составляет 64 Кбайт, а в дополнение к ней имеется буфер предварительной выборки на 32 строки. PowerPC 970 может выбирать до восьми команд за такт.

Интегрированная кэш-память 2-го уровня имеет емкость 512 Кбайт. В кэш-память первого уровня может осуществляться предварительная выборка до восьми потоков данных. Шина кэш-памяти 2-го уровня работает на частоте микропроцессорного ядра. Если в кэш-памяти 1-го уровня используется контроль по четности, то в L2 — уже коды ECC.

Пропускная способность системной шины — еще один фактор повышения производительности PowerPC 970. Здесь реализованы две однонаправленные шины — для чтения и записи в память. Обе имеют ширину 32 разряда и работают на частоте, в 4 раза меньшей частоты процессорного ядра. Для 1,8 ГГц

в ядре, таким образом, получается 450 МГц у шин, а эффективная частота равна 900 МГц. Соответственно суммарная пропускная способность равна 7,2 Гбайт/с (по 3,6 Гбайт/с для чтения и записи).

Конвейеры в PowerPC 970 имеют различную длину. Длина конвейера ФЗ в PowerPC 970 составляет 16 ступеней (12 в POWER4). Конвейер загрузки регистров/записи в память имеет 17 ступеней, конвейер ПТ — 21 ступень, AltiVec-конвейеры — до 25 ступеней. Из 16 ступеней целочисленного конвейера девять приходятся на выборку и декодирование команд.

Современные высокопроизводительные микропроцессоры все чаще становятся многоядерными. В микросхеме POWER4 размещаются два процессорных ядра. Кроме того, иногда говорят о микропроцессорном ядре внутри процессора. Если забыть о длине конвейеров, микропроцессорное ядро в PowerPC 970 такое же, как в POWER4 (рис. 3.40), а кэш-память 2-го уровня является внешней по отношению к нему.

Ядро PowerPC 970 аналогично ядру POWER4. Эти процессоры реализуют широкий параллелизм исполнения команд и обладают относительно длинными конвейерами исполнительных устройств. Устройство выборки PowerPC 970 выбирает из кэша

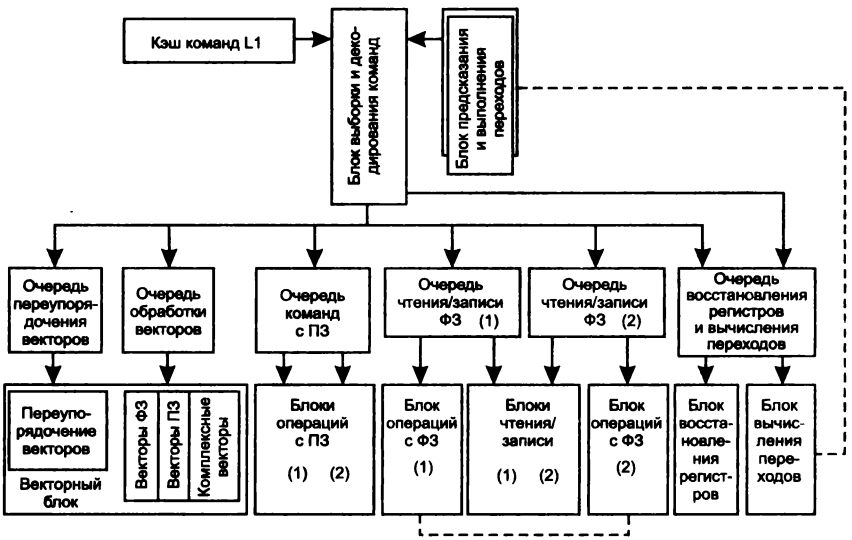


Рис. 3.40. Схема микропроцессорного ядра PowerPC 970FX

команд до восьми инструкций за такт, направляя их в буфер, откуда они считываются декодером, который также способен декодировать до восьми PowerPC-инструкций за такт. Кроме того, декодер преобразует PowerPC-инструкции во внутренние микрооперации (Internal OPerations — IOPs), более простые, подобно тому, как это делается в рассмотренных выше x86-процессорах. Большинство команд PowerPC транслируются в одну IOP, но некоторые команды (наподобие групповой записи из регистров в память) разделяются на две IOPs (процесс называется «to crack» — взламывать, расщеплять) или большее число (милликодирование). Далее IOPs организуются в группы по пять в каждой, которые затем направляются в очереди исполнительных устройств. Выборка, декодирование и формирование групп занимают девять стадий конвейера. Затем диспетчер распределяет IOPs из групп по шести очередям функциональных устройств, после этого начинается их выполнение (вне естественного порядка).

В PowerPC 970 насчитывается двенадцать исполнительных устройств — два блока ФЗ, два блока записи/чтения, блок переходов, блок регистра условий, два блока ПЗ и четыре векторных устройства. Однако векторные устройства не являются универсальными, и каждое из них способно выполнять только специфические для него команды.

Используется техника переименования регистров, всего на разных стадиях исполнения может находиться до двухсот IOPs. Длина конвейера устройства с плавающей точкой — одиннадцать стадий, ALU — пять, векторных устройств — тринадцать. За каждый такт может завершаться максимум пять команд. Используется двухуровневый механизм предсказания ветвлений. В процессоре имеется таблица истории ветвлений на 16 тысяч записей, как в Opteron, и в 4 раза больше, чем в Pentium 4/Athlon XP. Кроме того, существует дополнительная таблица на 16 тысяч записей, с каждой записью которой связан 11-битовый вектор, в котором записывается путь исполнения, выбранный для последних одиннадцати групп (состоящих из пяти IOPs). Таблица выбора (также на 16 тысяч записей) отслеживает эффективность первых двух схем для каждой инструкции ветвления, и по ее данным делается выбор в пользу той или иной схемы предсказания в каждом отдельном случае. Важное отличие комбинации PowerPC + AltiVec от аналогичных (Opteron + SSE2 или Pentium 4 + SSE2) состоит в том, что PowerPC с самого начала обладали «полноценными» FPU с тридцатью двумя 64-разрядными регистрами и им

не требуется расширение для чисел с такой точностью. AltiVec, как и SSE, работает с векторами чисел с максимальной точностью лишь 32 бита, в то время как SSE2 поддерживает 64-битовые числа. В противоположность этому Pentium 4 и в меньшей степени Opteron, из-за совместимости с ранними процессорами (вплоть до 8087), имеют весьма неэффективный блок плавающей точки в худших традициях CISC, и для его замены потребовалось создавать SSE2. В итоге в большинстве вычислительных задач PowerPC использует свой RISC-FPU, а Opteron и Pentium 4 — SSE2, что дает PowerPC несколько большую гибкость.

Как и любая современная RISC-архитектура, Power PC применяется в серверах различного уровня, однако наиболее широко используется в компьютерах Apple (с 1997 г. по настоящее время). В настоящий момент наблюдается смещение Power PC в рыночную нишу высокопроизводительных мультимедиа-процессоров.

Процессоры с набором команд Power PC поддерживаются многими операционными системами, включая MacOS, Darwin (ОС, лежащей в основе MacOS X), OpenBSD, FreeBSD, MorphOS и многими другими.

Номера процессоров

В течение долгих лет развитие ЦП показывало, что использование только тактовой частоты является недостаточным критерием для сопоставления ЦП.

AMD долго стремилась преуменьшить важность «сырой» тактовой скорости как окончательного критерия уровня производительности процессора и перешла к «номеру модели», который сопоставлял реальные скорости ее процессоров с оценками чипов Intel. До недавнего времени казалось, что Intel достаточно было ограничиваться тактовой частотой, по-видимому, потому что их чипы показывали более высокую частоту, чем равноценные ЦП AMD.

Процессоры Intel. Ситуация изменилась весной 2004 г., когда Intel также решила использовать более общие критерии для сопоставления процессоров в глазах пользователей. В дальнейшем, кроме рассмотрения чистой скорости часов, номера процессоров Intel, также примут во внимание такие важные особенности, как размер кэш-памяти, скорость FSB, технологический процесс и другие существенные архитектурные особенности.

Результатом явилась разработка системы номеров процессора с использованием комбинации марки процессора («семейство процессоров») и определенного номера из трех цифр («номер процессора» — 3 цифры — 7xx, 5xx или 3xx) — рис. 3.41.

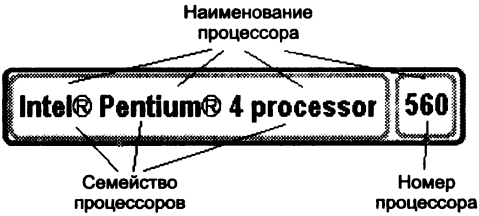


Рис. 3.41. Структура номера процессора

Это число да плюс «семейство процессоров» дают полное «наименование процессора». В пределах каждой последовательности определены номера процессора (например, 735, 560 или 320). Ссылка на тактовую скорость или на название процессора (как использовались в прошлом) заменяется номерами процессоров, которые теперь описывают более широкий набор особенностей. Семейства процессора могут также изменяться, чтобы отразить изменения в изделиях Intel.

Во время объявления системы обозначений примеры определенных семейств процессоров Intel выглядели следующим образом (табл. 3.10).

Таблица 3.10. **Примеры номеров процессоров Intel**

ПК	Процессоры	Номер
Настольные	Процессоры Pentium IV (включая процессоры Intel Pentium IV с поддержкой технологии Hyper-Threading)	5xx
	Процессоры Celeron D	3xx
Мобильные	Процессор Intel Pentium M (компонент Intel Centrino)	7xx
	Мобильный процессор Intel Pentium 4 processor (включая Mobile Intel Pentium 4 processor с поддержкой Hyper-Threading)	5xx
	Процессор Intel Celeron M	3xx

Более высокий номер в пределах семейства процессора может отмечать улучшение какой-либо характеристики ЦП, или изменения в архитектуре. Следует отметить, что в некоторых

случаях более высокий номер процессора может потенциально описывать улучшение одной из характеристик (не отражая ухудшение ряда других).

Номера процессоров предназначены, чтобы отразить различия в пределах некоторого семейства процессора (например, в пределах Intel Pentium IV) или в пределах последовательности (например, 550 против 540). Сами цифры не имеют никакого решающего значения, особенно в рамках семейства (например, 710 — не «лучше», чем 510 на том основании, что 7 больше 5!). Номера ассоциируются с различными семействами процессоров и таким образом представляют различные оценки для конечного пользователя.

Основные принципы интерпретации «номера процессора» комментируются в табл. 3.11.

Таблица 3.11. **Интерпретация «номера процессора»**

Что есть	Чего нет
Различает относительные особенности в пределах семейства процессоров	Способ сравнивать номера в рамках семейств процессора
Указывает на отдельные особенности или изменения в архитектуре	Мера эффективности
Будучи объединенным с маркой, помогает потребителю в выборе процессора	Единственный фактор в отборе процессора

Процессоры AMD. В частности, маркировка процессоров Athlon XP на ядре Thoroughbred расшифровывается следующим образом (рис. 3.42).

Система модельных номеров Opteron состоит из трех цифр — XYZ. Первая цифра (X) указывает на общее число процессоров,

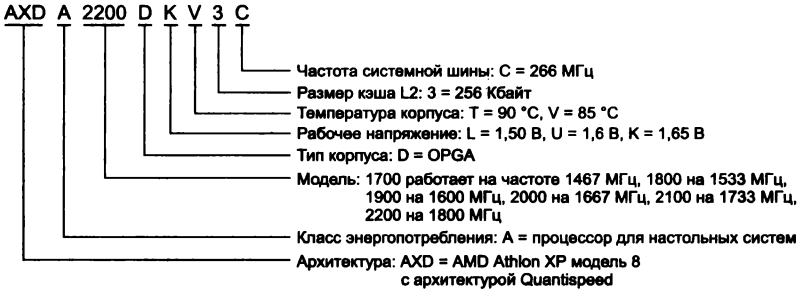


Рис. 3.42. Пример интерпретации номера процессора AMD

которое можно использовать в многопроцессорной системе (Opteron серии 200 может использоваться как в однопроцессорной, так и в двухпроцессорной конфигурациях, в то время как Opteron серии 800 — также и в 4- и в 8-процессорных системах).

Оставшиеся цифры (YZ) отражают относительную производительность (Opteron X44 быстрее Opteron X42, но причина здесь может заключаться в тактовой частоте, размере кэша или скорости шины HyperTransport).

Пример номеров ранних процессоров Opteron, между которыми нет никаких других отличий, помимо тактовой частоты, приводится в табл. 3.12.

Таблица 3.12. **Модельные номера AMD Opteron**

Название процессора	Тактовая частота, ГГц
Opteron 244	1,80
Opteron 242	1,60
Opteron 240	1,40

3.4. Системы команд x86. Макроассемблер

Рассмотрим несколько подробнее некоторые элементы набора команд x86 и программирования в этой среде (см. также Приложение 3). Система команд, обобщенно именуемая «x86», включает, как об этом уже упоминалось ранее, целочисленные операции, операции с ПЗ и SIMD-команды. Первоначальный набор команд IA-32 был расширен через какое-то время путем дополнения мультимедийных команд, однако коренное развитие IA-32 заключалось в переходе к 64 битам.

IA-32

IA-32 (иногда именуется x86-32) представляет собой набор команд-«долгожителей» ЦП Intel и определяет набор инструкций для микропроцессоров, работающих сегодня в подавляющем большинстве персональных компьютеров мира.

Этот набор команд впервые был представлен в микропроцессоре Intel 80386 (1985 г.) и остается основой большинства мик-

ропроцессоров ПК 20 лет спустя (процессоры, конечно, выполняют эти команды сегодня гораздо быстрее). В списках различных директив языков программирования IA-32 именуется еще иногда как «архитектура i386».

Аббревиатура означает «Intel Architecture, 32-bit» (Архитектура Intel, 32 бита), что отличает ее от предыдущих x86 процессоров (16 бит) и более поздней архитектуры на 64 бита — IA-64, известной также как «архитектура Itanium». «Живучесть» IA-32 частично объясняется полной обратной программной совместимостью.

Набор команд IA-32 обычно относится к типу CISC, хотя эта рубрика стала менее определенной по мере усовершенствований в микроархитектуре процессоров. Современные процессоры x86 (K7/8, NetBurst и более новые) часто упоминаются как процессоры «post-RISC».

Следующие поколения наборов команд на 64 бита. Преемниками IA-32 на 64 разряда являются два различных набора команд. Один из них действительно основывается на IA-32, но имеет отличающееся название (AMD64/EM64T), в то время как другой отказывается от IA-32 полностью, но имеет похожее название (IA-64).

Модели управления памятью

Известны две модели доступа к памяти в системе команд IA-32:

- реальный режим (Real mode) — процессор ограничен доступом не более чем к 1 Мбайт памяти;
- защищенный режим (Protected mode) — можно обратиться ко всей памяти (до 4 Гбайт).

Реальный режим. Ранняя ОС — MS-DOS — работала в реальном режиме, в то время как более поздние (OS/2, Windows, Linux и др.) обычно требуют защищенного режима. После включения питания (в процессе загрузки) процессор стартует в реальном режиме и затем начинает загружать программы в оперативную память из ПЗУ и с диска. Для того чтобы переключить процессор в защищенный режим, соответствующая программа может быть вставлена где-либо в последовательности команд начальной загрузки.

Защищенный режим. Кроме очевидной дополнительной адресуемости памяти выше предела 1 Мбайт для DOS, здесь имеется также ряд других преимуществ:

- защищенная память, которая препятствует взаимным помехам программ;
- виртуальная память, или выделение пространства на жестком диске так, чтобы программы использовали больше ОП, чем физически установлено на машине;
- переключение задач (многозадачный режим), когда несколько задач выполняются одновременно (путем быстрого и своевременного переключения активности процессора от одной задачи к другой).

Размер памяти в защищенном режиме обычно ограничивается 4 Гбайт, что, однако, не окончательный предел размера памяти в процессорах IA-32. Используя такие приемы, как страничная память и управление сегментами памяти, ОС IA-32 может обратиться более чем к 32-битовому адресному пространству, даже без переключения к 64 битам (один из этих приемов известен как физическое расширение адреса — Physical Address Extension).

Кроме того, можно использовать виртуальный режим 8086 — это подрежим работы в защищенном режиме — некий гибрид, который позволяет старым программам DOS выполняться под управлением супервизора защищенного режима, а также поддерживает одновременное выполнение программ в Protected mode и DOS. В то время как защищенный режим существовал уже в версиях 80286 на 16 битов, виртуальный появляется только в версии IA-32 защищенного режима.

Рассмотрим основные принципы функционирования процессоров i80x86 в реальном режиме.

Реальный режим процессоров i80x86

Оперативную память при работе в этом режиме можно разбить на логические блоки по 64 Кбайт, называемые сегментами, причем каждый сегмент может начинаться с адреса, кратного 16 байт. Таким образом, первый сегмент имеет начальный адрес 0, второй находится по адресу 16 (или 10_{16}) и т. д. Несколько близко расположенных сегментов могут перекрываться. Это удобно при организации совместного доступа к командам, дан-

ным разными программами. Доступ к каждой ячейке в памяти происходит путем указания значения регистра сегмента и смещения — адреса внутри этого блока.

Например, команда, подлежащая исполнению процессором в каждый данный момент времени, определяется из значений двух регистров — регистра `CS`, значение которого, будучи умноженное на 16, дает адрес начала сегмента команд, и регистра указателя команд `IP` (`Instruction Pointer`, СчАК или PC), указывающего положение соответствующей команды относительно начала сегмента команд.

Описание форматов команд, данных, структуры памяти и процессора далее производится с использованием ассемблерных представлений.

Оперативная память

Объем оперативной памяти i80x86 (здесь — 8086) — 2^{20} байт (1 Мбайт). Байты нумеруются, начиная с 0, и номер байта называется его *адресом*. Для ссылок на байты памяти используются 20-разрядные адреса — от 00000_{16} до $FFFFF_{16}$.

Байт содержит 8 разрядов (битов), каждый из которых может принимать значение 1 или 0. Разряды нумеруются справа налево от 0 до 7:

7	6	5	4	3	2	1	0

Байт — наименьшая адресуемая ячейка памяти — используется для хранения небольших целых чисел и символов. В i80x86 используются и более крупные ячейки — слова и двойные слова.

Слово — два соседних байта, размер — 16 бит (они нумеруются справа налево от 0 до 15). Используется для хранения целых чисел и адресов. Адресом слова считается адрес его первого байта (с меньшим адресом); этот адрес может быть четным и нечетным.

Двойное слово — четыре соседних байта (два соседних слова), размер — 32 бита. Адресом двойного слова считается адрес его первого байта. Используется для хранения «длинных» целых чисел и так называемых адресных пар (`сегмент:смещение`).

Регистры

Помимо ячеек оперативной памяти для хранения данных (кратковременного) могут использоваться регистры, входящие в состав процессора и доступные из машинной программы.

Все регистры имеют размер слова (16 битов), за каждым из них закреплено определенное имя (AX, SP и т. п.). По назначению и способу использования регистры можно разбить на следующие группы (расшифровка этих названий приводится в табл. 3.13):

- регистры общего назначения (AX, BX, CX, DX, BP, SI, DI, SP);
- сегментные регистры (CS, DS, SS, ES);
- счетчик команд (IP);
- регистр флагов (Flags).

Таблица 3.13. **Регистры (полурегистры) процессора i80x86**

Аббревиатура	Исходное название	Русскоязычный эквивалент	Старший полурегистр (разряды 15—8)	Младший полурегистр (разряды 7—0)
AX	Accumulator	Сумматор	AH	AL
BX	Base	База	BH	BL
CX	Counter	Счетчик	CH	CL
DX	Data	Данные	DH	DL
BP	Base pointer	Указатель базы	Нет	Нет
SI	Source index	Индекс источника	Нет	Нет
DI	Destination index	Индекс приемника	Нет	Нет
SP	Stack pointer	Указатель стека	Нет	Нет
CS	Code segment	Сегмент команд	Нет	Нет
DS	Data segment	Сегмент данных	Нет	Нет
SS	Stack segment	Сегмент стека	Нет	Нет
ES	Extra segment	Дополнительный сегмент	Нет	Нет
IP	Instruction pointer	Счетчик команд (СчАК)	Нет	Нет

Регистры общего назначения можно использовать во всех арифметических и логических командах. В то же время каждый

из них имеет определенную специализацию (некоторые команды «работают» только с определенными регистрами). Например, команды умножения и деления требуют, чтобы один из операндов находился в регистре AX (AX и DX, в зависимости от размера операнда), а команды управления циклом используют регистр CX в качестве счетчика цикла. Регистры BX и BP очень часто используются как базовые регистры, а SI и DI — как индексные. Регистр SP обычно указывает на вершину стека, аппаратно поддерживаемого в i80x86.

Регистры AX, BX, CX и DX конструктивно устроены так, что возможен независимый доступ к их старшей и младшей половинам; можно сказать, что каждый из этих регистров состоит из двух байтовых регистров, обозначаемых AH, AL, BH и т. д. (H — high, старший; L — low, младший) — табл. 3.13.

Таким образом, с каждым из этих регистров можно работать как с единым целым, так и с его частями, например — записать слово в AX, а затем считать только часть слова из регистра AH или заменить только часть в регистре AL и т. д. Такое устройство регистров позволяет использовать их как для работы с числами, так и со строками.

Все остальные регистры не делятся на полурегистры, поэтому прочитать или записать их содержимое (16 битов) можно только целиком.

Сегментные регистры CS, DS, SS и ES не могут быть операндами никаких команд, кроме команд пересылки и стековых команд. Эти регистры используются только для сегментирования адресов.

Счетчик команд IP всегда содержит адрес (смещение от начала программы) той команды, которая должна быть выполнена следующей (адрес начала программы хранится в регистре CS). Содержимое регистра IP можно изменить только командами перехода.

Флаги. Имеется особый регистр флагов (табл. 3.14). Флаг — это бит, принимающий значение «1» (флаг установлен) или «0» (флаг сброшен):

- флаги условий — автоматически меняются при выполнении команд и фиксируют те или иные свойства их результата (например, равен ли он нулю);
- флаги состояний — они меняются из программы и оказывают влияние на дальнейшее поведение процессора (например, блокируют прерывания) — табл. 3.15.

В i80x86 используются 9 флагов, каждому из них присвоено определенное имя (ZF, CF и т. д.).

Таблица 3.14. **Регистр флагов**

Флаги	x	x	x	x	OF	DF	IF	TF	SF	ZF	x	AF	x	PF	x	CF
Разряды	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Таблица 3.15. **Содержание регистра флагов**

Символ	Расшифровка	Содержание	Комментарий
Флаги условий			
CF	Carry flag	Флаг переноса	Принимает значение 1, если при сложении целых чисел появилась единица переноса, не укладывающаяся в разрядную сетку, или если при вычитании чисел без знака первое из них было меньше второго. В командах сдвига в CF заносится бит, вышедший за разрядную сетку. CF фиксирует также особенности команды умножения
OF	Overflow flag	Флаг переполнения	Устанавливается в 1, если при сложении или вычитании целых чисел со знаком получился результат, по модулю превосходящий допустимую величину (произошло переполнение мантиссы и она «залезла» в знаковый разряд)
ZF	Zero flag	Флаг нуля	Устанавливается в 1, если результат команды оказался равным 0
SF	Sign flag	Флаг знака	Устанавливается в 1, если в операции над числами со знаками получился отрицательный результат
PF	Parity flag	Флаг четности	Равен 1, если результат очередной команды содержит четное количество двоичных единиц. Учитывается обычно только при операциях ввода-вывода
AF	Auxiliary carry flag	Флаг дополнительного переноса	Фиксирует особенности выполнения операций над двоично-десятичными числами
Флаги состояний			
DF	Direction flag	Флаг направления	Устанавливает направление просмотра строк в строковых командах: при DF = 0 строки просматриваются «вперед» (от начала к концу), при DF = 1 — в обратном направлении

Окончание табл. 3.15

Символ	Расшифровка	Содержание	Комментарий
IF	Interrupt flag	Флаг прерываний	При IF = 0 процессор перестает реагировать на поступающие к нему прерывания, при IF = 1 блокировка прерываний снимается
TF	Trap flag	Флаг трассировки	При TF = 1 после выполнения каждой команды процессор делает прерывание (с номером 1), чем можно воспользоваться при отладке программы для ее трассировки

Регистры IA-32

В 32-разрядных (и выше) ЦП картина регистров изменяется. Как 386, так и все другие процессоры IA-32 имеют восемь 32-разрядных РОН (GPR) для использования прикладными программами. В процессорах AMD64 это число удваивается до 16. Есть 8 регистров стека с плавающей запятой (число также удвоено в AMD64). В более поздних процессорах были добавлены новые регистры для обеспечения выполнения различных наборов команд SIMD — типа MMX, 3DNow!, SSE, SSE2, SSE3 и SSSE3.

Есть также системные регистры, которые используются главным образом операционными системами, но не приложениями. Это — сегментные (segment), контрольные (control), отладочные (debug) и тестовые (test). Шесть сегментных регистров используются главным образом для управления памятью. Число контрольных, отладочных и тестовых регистров изменяется от модели к модели ЦП.

РОН. РОН x86 также являются специализированными и в свою очередь подразделяются на регистры данных и регистры адресации. Отметим, что в расширении AMD64 это разграничение ликвидируется, и РОН действительно здесь могут применяться для любых операций.

8- и 16-битовые участки регистров. Для программистов доступны также именуемые 8- и 16-битовые части регистров. Например, к младшим битам EAX можно обратиться, вызывая регистр AX. Некоторые из регистров на 16 бит могут быть далее подразделены на участки по 8 бит, например, старшие 8 бит AX могут быть вызваны под именем AH, а младшие биты — как AL. Точно

так же в EBX можно выделить регистр BX (16 бит), который в свою очередь содержит BH и BL (каждый по 8 бит).

Описание регистров приводится в табл. 3.16 (ср. с табл. 3.13).

Таблица 3.16. **Регистры IA-32**

Имя	Номер	Назначение регистра
		РОН данных*
EAX	000	Выделенный сумматор, который используется во всех основных вычислениях
ECX	001	Универсальный счетчик циклов, который имеет специальную интерпретацию для циклов
EDX	010	Регистр данных, который дополняет сумматор и хранит данные, необходимые для операции, которая выполняется сумматором
EBX	011	Используется как буферное ЗУ, но в режиме на 16 бит использовался как указатель (pointer)
		Адресные РОН**
ESP	100	Указатель стека, используется, чтобы хранить высший адрес стека
EBP	101	Указатель базы, используется, чтобы хранить адрес текущего фрейма стека. Может иногда применяться как буферное ЗУ
ESI	110	Индекс источника, обычно используемый в строковых операциях, чтобы загрузить данные из памяти на сумматор
EDI	111	Индекс назначения, обычно используемый в строковых операциях, чтобы записать данные из сумматора в память
EIP		Указатель адреса команды

* Могут использоваться как РОН, однако каждый имеет некоторую специализацию. Каждый из этих регистров также позволяет обращаться к именуемым участкам по 8 или 16 бит.

** Используются только для указания адреса. Они имеют именуемые 16-битовые участки (но не по 8 бит).

Представление чисел

Рассмотрим машинное представление целых чисел, строк и адресов. Представление двоично-десятичных чисел, используемых достаточно редко, не рассматривается. Что касается вещественных чисел, то в i80x86 нет команд вещественной арифметики (операции над этими числами реализуются программным путем или выполняются сопроцессором) и потому нет стандартного

представления вещественных чисел. Кроме того, рассматриваются некоторые особенности выполнения арифметических операций.

Как это принято в MASM, шестнадцатеричные числа записываются с буквой `h` в суффиксе («на конце»), двоичные — с буквой `b`, восьмеричные — с буквой `o` или `q`, десятичные — без каких-либо «дополнений».

Представление целых чисел. В общем случае под целое число можно отвести любое число байтов, однако система команд i80x86 поддерживает только числа размером в байт и слово и частично — размером в двойное слово.

В i80x86 делается различие между целыми числами без знака (неотрицательными) и числами со знаком. Это объясняется тем, что в ячейках одного и того же размера можно представить больший диапазон чисел без знаков, чем чисел со знаком, и если известно заранее, что некоторая числовая величина является неотрицательной, то выгоднее рассматривать ее как величину без знака, чем как величину со знаком.

Целые числа без знака в зависимости от их размера могут быть представлены в виде байта, слова или двойного слова. В виде байта представляются целые от 0 до 255 ($2^8 - 1$), в виде слова — целые от 0 до 65 535 ($2^{16} - 1$), в виде двойного слова — целые от 0 до 4 294 967 295 ($2^{32} - 1$). Числа записываются в двоичной системе счисления, занимая все разряды ячейки.

Например, число 130 записывается в виде байта `10000010b` (`82h`).

Числа размером в слово хранятся в памяти в «перевернутом» виде:

- младшие (правые) 8 битов числа размещаются в первом байте слова;
- старшие 8 битов — во втором байте (в шестнадцатеричной системе: две правые цифры — в первом байте, две левые цифры — во втором байте). Например, число `130` (`0082h`) в виде слова хранится в памяти так:

82	00

Однако в регистры числа загружаются в нормальном виде:

AX	00	82
	AH	AL

Перевернутое представление используется и при хранении в памяти целых чисел размером в двойное слово: в первом его байте размещаются младшие 8 битов числа, во втором байте — предыдущие 8 битов и т. д. Например, число 12345678h хранится в памяти так:

78	56	34	12

Другими словами, в первом слове двойного слова размещаются младшие (правые) 16 битов числа, а во втором слове — старшие 16 битов, причем в каждом из этих двух слов в свою очередь используется «перевернутое» представление.

Такой формат чисел объясняется тем, что в первых моделях i80x86 за один такт можно было считать из памяти только один байт, и все арифметические операции над многозначными числами начинаются с действий над младшими цифрами, поэтому из памяти в первую очередь надо считывать младшие цифры (если сразу нельзя считать все). Учитывая это, в первых i80x86 младшие цифры числа размещались перед старшими, а ради обратной совместимости такое представление сохранилось в последующих моделях.

Целые числа со знаком. Эти числа также представляются в виде байта, слова и двойного слова. В виде байта записываются числа от –128 до 127, слова — от –32 768 до 32 767, а двойные слова — от –2 147 483 648 до 2 147 483 647. При этом числа записываются в дополнительном коде.

Числа со знаком размером в слово и двойное слово записываются в памяти в «перевернутом» виде (при этом знаковый бит оказывается в последнем байте ячейки). Но в тексте на языке MASM эти числа, как и беззнаковые, записываются в нормальной форме.

Иногда число-байт необходимо расширить до слова, т. е. получить такое же по величине число, но размером в слово. Существует два способа такого расширения — без знака и со знаком. В любом случае исходное число-байт попадает во второй (до «переворачивания») байт слова, а первый байт заполняется по-разному:

- при расширении без знака в него записываются нулевые биты (12h превращается в 0012h);

- при расширении со знаком в первый байт записываются нули, если число-байт было неотрицательным, и записывается восемь двоичных единиц в противном случае (81h преобразуется в FF81h).

Аналогично происходит расширение числа-слова до двойного слова.

Арифметические операции

В i80x86 имеются команды сложения и вычитания целых чисел размером в слово и байт (см. табл. П3.1). Специальных команд для сложения и вычитания двойных слов нет, эти операции реализуются через команды сложения и вычитания слов. Сложение и вычитание беззнаковых чисел производится по модулю 2^8 для байтов и 2^{16} для слов (см. рис. 1.4, *а*).

Это означает, что если в результате сложения появилась единица переноса, не вмещающаяся в разрядную сетку, то она отбрасывается. Например, при сложении байтов 128 и 130 получается число 258 (100000010b) длиной 9 бит, поэтому левая двоичная единица отбрасывается и остается число 2 (10b), которое и является результатом сложения.

Ошибка здесь не фиксируется, но флаг переноса CF устанавливается в «1» (если переноса не было, в CF заносится «0»). Установить такое искажение суммы можно только последующим анализом флага CF.

Искажение результата происходит и при вычитании из меньшего числа большего. Здесь также не фиксируется ошибка, однако первому числу дается «заем единицы» (в случае байтов это число увеличивается на 256, для слов — на 2^{16}), после этого и производится вычитание. Например, вычитание байтов 2 и 3 сводится к вычитанию чисел 256 + 2 = 258 и 3, в результате этого получается неправильная разность — 255 (а не –1). Для того чтобы можно было обнаружить такую ситуацию, флаг переноса CF переключается на «1» (если же «заема» не было, в CF записывается «0»).

Сложение и вычитание целых чисел со знаком производится по тем же алгоритмам, что и для чисел без знака (в этом одно из достоинств дополнительного кода).

Например, сложение байтовых чисел 1 и –2 происходит так: берутся их дополнительные коды 1 и (256 – 2) = 254, вычисляется сумма этих величин 1 + 254 = 255, и она трактуется как число со знаком –1 (255 = 256 – 1).

Если при таком сложении возникла единица переноса, то она, как обычно, отбрасывается, а флаг CF получает значение «1». Однако в данном случае это отсечение не представляет интереса — результат операции будет правильным, например: 3 + (–2) = 3 + 254(mod 256) = 257(mod 256) = 1. Тем не менее здесь возможна иная неприятность — модуль суммы (ее мантисса) может превзойти допустимую границу и переместиться в знаковый разряд, «испортив» его. Например, при сложении байтовых чисел 127 и 2 получается значение 129 (100001001b), представляющее дополнительный код числа –127 (256 – 129).

Хотя результат здесь получился и неправильным, процессор не фиксирует ошибку, но заносит «1» в флаг переполнения OF (если «переполнения мантиссы» не было, в OF записывается «0»). Анализируя затем этот флаг, можно зафиксировать такую ошибку.

Что касается умножения и деления знаковых и чисел без знака, то они выполняются по разным алгоритмам, разными машинными командами. Однако и у этих операций есть ряд особенностей. При умножении байтов (слов) первый сомножитель должен находиться в регистре AL (AX), результатом же умножения является слово (двойное слово), которое заносится в регистр AX (регистры DX и AX). Тем самым при умножении сохраняются все цифры произведения. При делении байтов (слов) первый операнд (делимое) должен быть словом (двойным словом) и обязан находиться в регистре AX (регистрах DX и AX). Результатом деления являются две величины размером в байт (слово) — неполное частное (div) и остаток от деления (mod); неполное частное записывается в регистр AL (AX), а остаток — в регистр AH (DX).

Представление символов и строк

На символ отводится один байт памяти, в который записывается код символа — целое от 0 до 255. В i80x86 используется система кодировки ASCII (см. Приложение 3).

Строка (последовательность символов) размещается в соседних байтах памяти (в неперевернутом виде): код первого символа строки записывается в первом байте, код второго символа — во втором байте и т. п. Адресом строки считается адрес ее первого байта.

В i80x86 строкой считается также и последовательность слов (обычно это последовательность целых чисел). Элементы таких строк располагаются в последовательных ячейках памяти, но каждый элемент представлен в «перевернутом» виде.

Представление адресов

Адрес — это порядковый номер ячейки памяти, т. е. неотрицательное целое число, поэтому в общем случае адреса представляются так же, как и числа без знака. Однако в i80x86 есть ряд особенностей в представлении адресов.

Дело в том, что в i80x86 термином «адрес» могут обозначаться разные понятия:

- 16-битовое смещение (*offset*) — адрес ячейки, отсчитанный от начала сегмента (области) памяти, которому принадлежит эта ячейка. В этом случае под адрес отводится слово памяти, причем адрес записывается в «перевернутом» виде (как и числа-слова вообще);
- 20-битовый абсолютный адрес некоторой ячейки памяти. В силу ряда причин в i80x86 такой адрес задается не как 20-битовое число, а как пара `сегмент:смещение`, где `сегмент` (*segment*) — это первые 16 битов начального адреса сегмента памяти, которому принадлежит ячейка, а `смещение` (*offset*) — 16-битовый адрес этой ячейки, отсчитанный от начала данного сегмента памяти (величина 16 × сегмент + смещение дает абсолютный адрес ячейки).

Такая пара записывается в виде двойного слова: в первом слове размещается смещение, а во втором — сегмент, причем каждое из этих слов в свою очередь представлено в «перевернутом» виде. Например, пара `1234h:5678h` будет записана так:

78	56	34	12
Смещение		Сегмент	

Директивы определения данных

Для того чтобы в программе на MASM зарезервировать ячейки памяти под константы и переменные, необходимо воспользоваться директивами определения данных — с названиями DB (описывает данные размером в байт), DW (слово) и DD (двойное слово).

Директивы, или команды ассемблеру — это предложения программы, которыми ее автор сообщает какую-то информацию ассемблеру или предписывает что-либо сделать дополнительно, помимо перевода символьных команд на машинный язык.

В простейшем случае в директиве DB, DW или DD описывается одна константа, которой дается имя для последующих ссылок на нее. По этой директиве ассемблер формирует машинное представление константы (в частности, если надо, «переворачивает» ее) и записывает в очередную ячейку памяти. Адрес этой ячейки становится значением имени: все вхождения имени в программу ассемблер будет заменять на этот адрес.

Имена, указанные в директивах DB, DW и DD, называются именами переменных (в отличие от меток — имен команд).

В MASM числа записываются в нормальном (неперевернутом) виде в системах счисления с основанием 10, 16, 8 или 2.

Примеры:

```
A DB 162     ; описать константу-байт 162 и дать ей имя A
B DB 0A2h    ; такая же константа, но с именем B
C DW -1      ; константа-слово –1 с именем C
D DW 0FFFFh  ; такая же константа-слово, но с именем D
E DD -1      ; –1 как двойное слово
```

Константы-символы описываются в директиве DB двояко: указывается либо код символа (целое от 0 до 255), либо сам символ в кавычках (одинарных или двойных); в последнем случае ассемблер сам заменит символ на его код. Например, следующие директивы эквивалентны (2A — код звездочки в ASCII):

```
CH DB 02Ah
CH DB '*'
CH DB "*"
```

Константы-адреса, как правило, задаются именами. Так, по директиве

```
ADR DW CH
```

будет выделено слово в памяти, которому дается имя ADR и в которое будет помещен адрес (смещение), соответствующий имени CH. Если такое же имя описать в директиве DD, то ассемблер автоматически добавит к смещению имени его сегмент и запишет смещение в первую половину двойного слова, а сегмент — во вторую половину.

По любой из директив DB, DW и DD можно описать переменную, т. е. отвести ячейку, не дав ей начального значения. В этом случае в правой части директивы указывается вопросительный знак:

```
F DW ? ; отвести слово и дать ему имя F, ничего в этот байт
         не записывать.
```

В одной директиве можно описать сразу несколько констант и/или переменных одного и того же размера, с этой целью их следует перечислить через запятую. Они размещаются в соседних ячейках памяти. Пример:

```
G DB 200, -5, 10h, ?, 'F'
```

Имя, указанное в директиве, считается именующим первую из констант.

Для ссылок на остальные в MASM используются выражения вида <имя> + <целое>; например, для доступа к байту с числом –5 надо указать выражение G + 1, для доступа к байту с 10h — выражение G + 2 и т. д.

Если в директиве DB перечислены только символы, например:

```
S DB 'a','+','b',
```

тогда эту директиву можно записать короче, заключив все эти символы в одни кавычки:

```
S DB 'a+b'
```

И наконец, если в директиве описывается несколько одинаковых констант (переменных), то можно воспользоваться конструкцией повторения

```
k DUP(a,b,...,c),
```

которая эквивалентна повторенной *k* раз последовательности `a, b, ..., c`.

Например, директивы

```
V1 DB 0,0,0,0,0
V2 DW ?,?,?,?,?,?,?,?,?,'a',1,2,1,2,1,2,1,2,
```

можно записать более коротко таким образом:

```
V1 DB 5 DUP(0)
V2 DW 9 DUP(?), 'a', 4 DUP(1,2).
```

Представление команд. Модификация адресов

Структура команд. Исполнительные адреса. Машинные команды i80x86 занимают от 1 до 6 байтов. Код операции (КОП) занимает один или два первых байта команды. В i80x86 достаточно много различных операций, поэтому для них не хватает 256 различных КОП, которые можно представить в одном байте. В связи с этим некоторые операции объединяются в группу и им дается один и тот же КОП, во втором же байте команды этот КОП уточняется. Кроме того, во втором байте указываются типы и способ адресации операндов. Остальные байты команды указывают на операнды.

Команды могут иметь от 0 до трех операндов, у большинства команд — один или два операнда. Размер операндов — байт или слово (редко — двойное слово). Операнд может быть указан в самой команде (это так называемый непосредственный операнд) либо может находиться в одном из регистров i80x86 и тогда в команде указывается этот регистр, либо может находиться в ячейке памяти и тогда в команде тем или иным способом указывается адрес этой ячейки. Некоторые команды требуют, чтобы операнд находился в фиксированном месте (например, в регистре `AX`), тогда операнд явно не указывается в команде. Результат выполнения команды помещается в регистр или ячейку памяти, из которого (которой), как правило, берется первый

операнд. Например, большинство команд с двумя операндами реализуют действие

```
op1 := op1 - op2,
```

где op1 — регистр или ячейка; а op2 — непосредственный операнд, регистр или ячейка.

Адрес операнда разрешено модифицировать по одному или двум регистрам. В первом случае в качестве регистра-модификатора разрешено использовать регистр BX, BP, SI или DI (и никакой иной). Во втором случае один из модификаторов обязан быть регистром BX или BP, а другой — регистром SI или DI; одновременная модификация по BX и BP или SI и DI недопустима.

Регистры BX и BP обычно используются для хранения базы (или базового, начального адреса) некоторого участка памяти (например, массива) и потому называются *базовыми регистрами*, а регистры SI и DI часто содержат индексы элементов массива и потому называются *индексными регистрами*.

Однако такое «распределение ролей» необязательно, и, например, в SI может находиться база массива, а в BX — индекс элемента массива.

В MASM адреса в командах записываются в виде одной из следующих конструкций:

```
A, A[M] или A[M1][M2],
```

где A — адрес; M — регистр BX, BP, SI или DI; M1 — регистр BX или BP, а M2 — регистр SI или DI. Во втором и третьем варианте A может отсутствовать, в этом случае считается, что A = 0.

При выполнении команды процессор прежде всего вычисляет так называемый *исполнительный* (эффективный) адрес — сумму адреса, заданного в команде, и текущих значений указанных регистров-модификаторов, причем все эти величины рассматриваются как неотрицательные, и суммирование ведется по модулю 2^{16} ([r] означает содержимое регистра r):

```
A : Аисп = A
A[M] : Аисп = A+[M] (mod 2^16)
A[M1][M2]: Аисп = A+[M1]+[M2] (mod 2^16).
```

Полученный таким образом 16-разрядный адрес определяет так называемое *смещение* — адрес, отсчитанный от начала некоторого сегмента (области) памяти. Перед обращением к памяти

процессор добавляет к смещению начальный адрес этого сегмента (он хранится в некотором сегментном регистре) и в результате получается окончательный 20-разрядный адрес, по которому происходит реальное обращение к памяти.

Форматы команд

В i80x86 форматы машинных команд достаточно разнообразны. Для примера приведем лишь основные форматы команд с двумя операндами (см. также табл. 3.1).

Формат «регистр—регистр» (2 байта):

КОП	d	w	11	reg1	reg2
7 2	1	0	76	53	20

Команды этого формата описывают обычно действие `reg1 := reg1 - reg2` или `reg2 := reg2 - reg1`. Поле КОП первого байта указывает на операцию (–), которую надо выполнить.

Бит `w` определяет размер операндов, а бит `d` указывает, в какой из регистров записывается результат:

w = 1	— слова	d = 1	reg1 := reg1 – reg2
= 0	— байты	= 0	reg2 := reg2 – reg1

Во втором байте два левых бита фиксированы (для данного формата), а трехбитовые поля `reg1` и `reg2` указывают на регистры, участвующие в операции, согласно следующим правилам:

reg	w = 1	w = 0
000	AX	AL
001	CX	CL
010	DX	DL
011	BX	BL
100	SP	AH
101	BP	CH
110	SI	DH
111	DI	BH

Формат «регистр—память» (2—4 байта):

КОП		w	mod	reg	mem	Адрес (0—2 байта)

Эти команды описывают операции `reg := reg – mem` или `mem := mem – reg`. Бит `w` первого байта определяет размер операндов (см. ранее), а бит `d` указывает, куда записывается результат: в регистр (`d = 1`) или в ячейку памяти (`d = 0`). Трехбитовое поле `reg` второго байта указывает операнд-регистр (см. выше), двухбитовое поле `mod` определяет, сколько байт в команде занимает операнд-адрес (00 — 0 байтов, 01 — 1 байт, 10 — 2 байта), а трехбитовое поле `mem` указывает способ модификации этого адреса. В следующей таблице указаны правила вычисления исполнительного адреса в зависимости от значений полей `mod` и `mem` (`a8` — адрес размером в байт, `a16` — адрес размером в слово):

mod / mem	00	01	10
000	[BX]+[SI]	[BX]+[SI]+a8	[BX]+[SI]+a16
001	[BX]+[DI]	[BX]+[DI]+a8	[BX]+[DI]+a16
010	[BP]+[SI]	[BP]+[SI]+a8	[BP]+[SI]+a16
011	[BP]+[DI]	[BP]+[DI]+a8	[BP]+[DI]+a16
100	[SI]	[SI]+a8	[SI]+a16
101	[DI]	[DI]+a8	[DI]+a16
110	a16	[BP]+a8	[BP]+a16
111	[BX]	[BX]+a8	[BX]+a16

Замечания. Если в команде не задан адрес, то он считается нулевым.

Если адрес задан в виде байта (`a8`), то он автоматически расширяется со знаком до слова (`a16`). Случай `mod = 00` и `mem = 110` указывает на отсутствие регистров-модификаторов, при этом адрес должен иметь размер слова (адресное выражение `[BP]` ассемблер транслирует в `mod = 01` и `mem = 110` при `a8 = 0`). Случай `mod = 11` соответствует формату «регистр—регистр».

Формат «регистр—непосредственный операнд» (3—4 байта):

КОП	s	w	11	КОП'	reg	Непосред. операнд (1—2 б.)

Команды этого формата описывают операции `reg := reg – – immed` (`immed` — непосредственный операнд). Бит `w` указывает на размер операндов, а поле `reg` – на регистр-операнд (см. ранее). Поле `КОП` в первом байте определяет лишь класс операции

(например, класс сложения), уточняет же операцию поле КОП' из второго байта. Непосредственный операнд может занимать 1 или 2 байта в зависимости от значения бита w, при этом операнд-слово записывается в команде в «перевернутом» виде. Ради экономии памяти в i80x86 предусмотрен случай, когда в операции над словами непосредственный операнд может быть задан байтом (на этот случай указывает «1» в бите s при w = 1), и тогда перед выполнением операции байт автоматически расширяется (со знаком) до слова.

Формат «память—непосредственный операнд» (3—6 байтов):

КОП	s	w	mod	КОП"	mem	Адрес (0–2 б.)	Непоср. оп (1–2 б.)

Команды этого формата описывают операции типа mem := mem – immed. Смысл всех полей — тот же, что и в предыдущих форматах.

Помимо рассмотренных в i80x86 используются и другие форматы команды с двумя операндами; так, предусмотрен специальный формат для команд, один из операндов которых фиксирован (обычно это регистр AX). Имеют свои форматы и команды с другим числом операндов.

Запись команд в MASM

Из сказанного ясно, что одна и та же операция в зависимости от типов операндов записывается в виде различных машинных команд, например, в i80x86 имеется 28 команд пересылки байтов и слов. В то же время в MASM все эти «родственные» команды записываются единообразно — например, все команды пересылки имеют одну и ту же символьную форму записи:

```
MOV op1,op2 (op1:=op2)
```

Анализируя типы операндов, ассемблер автоматически выбирает подходящую машинную команду.

В общем случае команды записываются в MASM следующим образом:

```
МЕТКА: МНЕМОКОД ОПЕРАНДЫ; КОММЕНТАРИЙ
```

Метка с двоеточием, а также точка с запятой и комментарий могут отсутствовать. Метка играет роль имени команды, ее можно использовать в командах перехода на данную команду. Комментарий не влияет на смысл команды, а лишь поясняет ее. Операнды, если есть, перечисляются через запятую. Основные правила записи операндов следующие.

Регистры указываются своими именами, например:

```
MOV AX,SI ;оба операнда — регистры.
```

Непосредственные операнды задаются константными выражениями (их значениями являются константы-числа), например:

```
MOV BH,5          5 — непосредственный операнд
MOV DI,SIZE X     SIZE X (число байтов, занимаемых
                  переменной X) — непосредственный
                  операнд
```

Адреса описываются адресными выражениями (например, именами переменных), которые могут быть модифицированы по одному или двум регистрам; например, в следующих командах первые операнды задают адреса:

```
MOV X,AH
MOV X[BX][DI],5
MOV [BX],CL.
```

При записи команд в символьной форме необходимо внимательно следить за правильным указанием типа (размера) операндов, чтобы не было ошибок. Тип обычно определяется по внешнему виду одного из них, например:

```
MOV AH,5          Пересылка байта, так как AH — байтовый
                  регистр
MOV AX,5          Пересылка слова, так как AX — 16-битовый
                  регистр-операнд (5 может быть байтом и
                  словом, по нему нельзя определить размер
                  пересылаемой величины)
MOV [BX],300      Пересылка слова, так как число 300 не
                  может быть байтом
```

Если по внешнему виду можно однозначно определить тип обоих операндов, тогда эти типы должны совпадать, иначе ассемблер зафиксирует ошибку. Примеры:

```
MOV DS,AX    Оба операнда имеют размер слова
MOV CX,BH    Ошибка: регистры CX и BH имеют разные
             размеры
MOV DL,300   Ошибка: DL — байтовый регистр, а число 300
             не может быть байтом
```

Возможны ситуации, когда по внешнему виду операндов нельзя определить тип ни одного из них, как, например, в команде

```
MOV [BX],5.
```

Здесь число 5 может быть и байтом, и словом, а адрес из регистра BX может указывать и на байт памяти, и на слово. В подобных ситуациях ассемблер фиксирует ошибку. Чтобы избежать ее, надо уточнить тип одного из операндов с помощью оператора PTR:

```
MOV BYTE PTR [BX],5      Пересылка байта
MOV WORD PTR [BX],5      Пересылка слова
```

Операторы

Это разновидность выражений языка MASM, аналогичных функциям.

Оператор PTR (вычисляет адрес аргумента) может использоваться для изменения типа переменной, заданной при ее описании. Пусть, например, X есть имя переменной размером в слово

```
X DW 999,
```

тогда при необходимости записать в байтовый регистр AH значение только первого байта этого слова, нельзя воспользоваться командой вида

```
MOV AH,X,
```

так как ее операнды имеют разный размер. Эту команду следует записать несколько иначе:

```
MOV AH,BYTE PTR X.
```

Здесь конструкция `BYTE PTR X` означает адрес `X`, но уже рассматриваемый не как адрес слова, а как адрес байта. (Напомним, что с одного и того же адреса может начинаться байт, слово и двойное слово; оператор `PTR` уточняет, ячейка какого именно размера имеется в виду.)

Если в символьной команде, оперирующей со словами, указан непосредственный операнд размером в байт, как, например, в команде

```
MOV AX,80h,
```

то возникает некоторая неоднозначность — неизвестно, что именно будет записано в регистр `AX` — число `0080h` (+128) или `0FF80h` (–128). В подобных ситуациях ассемблер формирует машинную команду, где операнд-байт расширен до слова, причем расширение происходит со знаком, если операнд был записан как отрицательное число, и без знака — в остальных случаях. Например:

```
MOV AX,-128 ; => MOV AX,0FF80h (A:=-128)
MOV AX,128 ; => MOV AX,0080h (A:=+128)
MOV AX,80h ; => MOV AX,0080h (A:=+128).
```

Сегментирование

Сегменты памяти. Сегментные регистры. Первые модели i80x86 имели оперативную память объемом 2^{16} байтов (64 Кбайт) и потому использовали 16-битовые адреса. В последующих моделях память была увеличена до 2^{20} байт (1 Мбайт = 1000 Кбайт), что потребовало уже 20-битовые адреса. Однако в этих i80x86 ради преемственности были сохранены 16-битовые адреса — именно такие адреса хранятся в регистрах, указываются в командах и образуются в результате модификации по базовым и индексным регистрам.

Проблема совместимости решается с помощью сегментирования (неявного базирования) адресов (рис. 3.43). В i80x86 абсолютный (20-битовый) адрес `A` любой ячейки памяти можно представить как сумму 20-битового начального адреса (базы) `B` сегмента, которому принадлежит ячейка, и 16-битового смещения `D` — адреса этой ячейки, отсчитанного от начала сегмента:

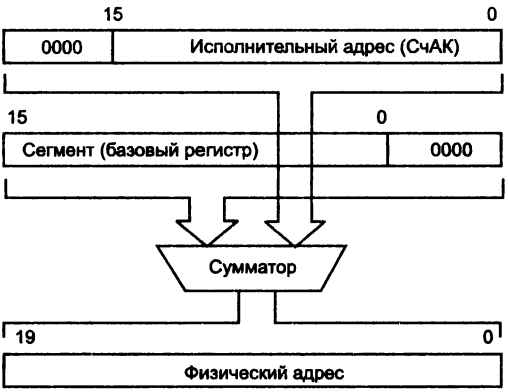


Рис. 3.43. Модификация адресов при сегментной адресации

`A = B + D`. (Неоднозначность выбора сегмента не играет существенной роли, главное — чтобы сумма `B` и `D` давала нужный адрес). Адрес `B` заносится в некоторый регистр `S`, а в команде, где должен быть указан адрес `A`, вместо него записывается пара из регистра `S` и смещения `D` (в MASM такая пара, называемая адресной парой или указателем, записывается как `S:D`). Процессор же устроен так, что при выполнении команды он прежде всего по паре `S:D` вычисляет абсолютный адрес `A` как сумму содержимого регистра `S` и смещения `D` и только затем обращается к памяти по адресу `A`. Заменяя в командах абсолютные адреса на адресные пары, удается адресовать всю память 16-битовыми адресами (смещениями).

В качестве регистра `S` разрешается использовать не любой регистр, а только один из четырех регистров, называемых сегментными: `CS`, `DS`, `SS` и `ES`. В связи с этим одновременно можно работать с четырьмя сегментами памяти: начало одного из них загружается в регистр `CS` и все ссылки на ячейки этого сегмента указываются в виде пар `CS:D`, начало другого заносится в `DS` и все ссылки на его ячейки задаются в виде пар `DS:D` и т. д. Если одновременно надо работать с большим числом сегментов, тогда следует своевременно сохранять содержимое сегментных регистров и записывать в них начальные адреса пятого, шестого и других сегментов.

Отметим, что используемые сегменты могут быть расположены в памяти произвольным образом: они могут не пересекаться, а могут пересекаться и даже совпадать. Какие сегменты памяти

использовать, в каких сегментных регистрах хранить их начальные адреса — все это — дело автора программы.

Как и все регистры i80x86, сегментные регистры имеют размер слова. Поэтому возникает проблема — как удается разместить в них 20-битовые начальные адреса сегментов памяти? Решение следующее — поскольку все эти адреса кратны 16 (см. ранее), то в них младшие 4 бита (последняя шестнадцатеричная цифра) всегда нулевые, а потому эти биты можно не хранить явно, а лишь подразумевать. Именно так и происходит — в сегментном регистре всегда хранятся только первые 16 битов (первые четыре шестнадцатеричные цифры) начального адреса сегмента (эта величина называется номером сегмента или просто сегментом). При вычислении же абсолютного адреса $A_{абс}$ по паре `S:D` процессор сначала приписывает справа к содержимому регистра `S` четыре нулевых бита (другими словами, умножает на 16) и лишь затем прибавляет смещение `D`, причем суммирование ведется по модулю 2^{20}:

$$A_{абс} = 16 \times [S] + D \pmod{2^{20}}.$$

Если, например, в регистре `CS` хранится величина `1234h`, тогда адресная пара `1234h:507h` определяет абсолютный адрес, равный `16 × 1234h + 507h = 12340h + 507h = 12847h`.

Сегментные регистры по умолчанию. Согласно описанной схеме сегментирования адресов, замену абсолютных адресов на адресные пары надо производить во всех командах, имеющих операнд-адрес. Однако был разработан способ, позволяющий избежать указания этих пар в большинстве команд, суть которого состоит в том, что по умолчанию устанавливается, какой именно сегментный регистр соответствует данному сегменту памяти, а в командах задается только смещение. Явно не указанный сегментный регистр автоматически подставляется в команду, и только при необходимости нарушить эти умолчания следует полностью указывать адресную пару. Список умолчаний приводится в табл. 3.17.

С учетом такого распределения ролей сегментных регистров машинные программы обычно строятся так: все команды программы размещаются в одном сегменте памяти, начало которого заносится в регистр `CS`, а все данные размещаются в другом сегменте, начало которого заносится в регистр `DS`; если нужен стек, то под него отводится третий сегмент памяти, начало которого

Таблица 3.17. **Сегментные регистры по умолчанию**

Регистр	Умолчания	Комментарий
CS	Указывает на начало области памяти, в которой размещены команды программы (эта область называется сегментом команд или сегментом кодов), и потому при ссылках на ячейки этой области регистр CS можно не указывать явно, он подразумевается по умолчанию	Абсолютный адрес очередной команды, подлежащей выполнению, всегда задается парой CS:IP; в счетчике команд IP всегда находится смещение этой команды относительно адреса из регистра CS
DS	Указывает на сегмент данных (область памяти с константами, переменными и другими величинами программы)	Во всех ссылках на этот сегмент регистр DS можно явно не указывать, так как он подразумевается по умолчанию
SS	Указывает на стек — область памяти, доступ к которой осуществляется по принципу «последним записан — первым считан»	Все ссылки на стек, в которых явно не указан сегментный регистр, по умолчанию сегментируются по регистру SS
ES	Считается свободным, он не привязан ни к какому сегменту памяти и его можно использовать по своему усмотрению	Чаще всего он применяется для доступа к данным, которые не поместились или сознательно не были размещены в сегменте данных

записывается в регистр SS. После этого практически во всех командах можно указывать не полные адресные пары, а лишь смещения, так как сегментные регистры в этих парах будут восстанавливаться автоматически.

Здесь, правда, возникает такой вопрос: как по смещению определить, на какой именно сегмент памяти оно указывает? В общих чертах ответ такой — ссылки на сегмент команд могут содержаться только в командах перехода, а ссылки практически во всех других командах (кроме строковых и стековых) — это ссылки на сегмент данных. Например, в команде пересылки

```
MOV AX,X,
```

имя X воспринимается как ссылка на данное, а потому автоматически восстанавливается до адресной пары DS:X. В команде же безусловного перехода по адресу, находящемуся в регистре BX,

```
JMP BX,
```

абсолютный адрес перехода определяется парой CS:[BX].

Если в ссылке на какую-то ячейку памяти не указан явно сегментный регистр, то этот регистр берется по умолчанию.

Явно же сегментные регистры надо указывать, только если по каким-то причинам регистр по умолчанию не подходит. Если, например, в команде пересылки нам надо сослаться на стек (скажем, надо записать в регистр AH байт стека, помеченный именем X), тогда уже будет недостаточно предположения о том, что по умолчанию операнд команды MOV будет сегментироваться по регистру DS, и потому следует явно указать иной регистр — в данном случае регистр SS, так как именно он указывает на стек:

```
MOV AH,'SS:X.
```

Однако такие случаи встречаются редко и потому в командах, как правило, указывается только смещение.

Отметим, что в MASM сегментный регистр записывается в самой команде непосредственно перед смещением (именем переменной, меткой и т. п.), однако на уровне машинного языка ситуация несколько иная. Предусмотрено четыре специальные однобайтовые команды, именуемые префиксами замены сегмента (обозначаемые как CS:, DS:, SS: и ES:). Они ставятся перед командой, операнд-адрес которой должен быть просегментирован по регистру, отличному от регистра, подразумеваемого по умолчанию. Например, приведенная ранее символическая команда пересылки — это на самом деле две машинные команды:

```
SS:
MOV AH,X.
```

Сегментирование, базирование и индексирование адресов. Поскольку сегментирование адресов — это разновидность модификации адресов, то в i80x86 адрес, указываемый в команде, в общем случае модифицируется по трем регистрам — сегментному, базовому и индексному. В целом модификация адреса производится в два этапа. Сначала учитываются только базовый и индексный регистры (если они, конечно, указаны в команде), причем вычисление здесь происходит в области 16-битовых адресов; полученный в результате 16-битовый адрес называется *исполнительным (эффективным) адресом*. Если в команде не предусмотрено обращение к памяти (например, она загружает адрес в регистр), то на этом модификация адреса заканчивается и используется именно исполнительный адрес (он загружается в регистр).

Если же нужен доступ к памяти, тогда на втором этапе исполнительный адрес рассматривается как смещение и к нему прибавляется (умноженное на 16) содержимое сегментного регистра, указанного явно или взятого по умолчанию и в результате вычисляется *абсолютный (физический)* 20-битовый адрес, по которому фактически происходит обращение к памяти (см. рис. 3.22).

Отметим, что сегментный регистр учитывается только непосредственно перед обращением к памяти, а до этого работа ведется только с 16-битовыми адресами. Если учесть к тому же, что сегментные регистры, как правило, не указываются в командах, то можно считать, что i80x86 работает с 16-битовыми адресами.

Как уже сказано, если в ссылке на ячейку памяти не указан сегментный регистр, то он определяется по умолчанию. Это делается по следующим правилам:

- в командах перехода адрес перехода сегментируется по регистру `CS` и только по нему, так как абсолютный адрес команды, которая должна быть выполнена следующей, всегда определяется парой `CS:IP` (попытка изменить в таких командах сегментный регистр будет безуспешной). Сегментирование по регистру `CS` касается именно адреса перехода, а не адреса той ячейки, где он может находиться. Например, в команде безусловного перехода по адресу, находящемуся в ячейке `X`:

  ```
  JMP X,
  ```

  имя `X` сегментируется по регистру `DS`, а вот адрес перехода, взятый из ячейки `X`, уже сегментируется по регистру `CS`;
- адреса во всех других командах, кроме строковых (`STOS`, `MOVS`, `SCAS` и `CMPS`), сегментируются по умолчанию, а именно:
  - — по регистру `DS`, если среди указанных регистров-модификаторов нет регистра `BP`;
  - — по регистру `SS`, если один из модификаторов — регистр `BP`.

  Таким образом, адреса вида `A`, `A[BX]`, `A[SI]`, `A[DI]`, `A[BX][SI]` и `A[BX][DI]` сегментируются по регистру `DS`, а адреса `A[BP]`, `A[BP][SI]` и `A[BP][DI]` — по регистру `SS`, т. е. адреса трех последних видов используются для доступа к ячейкам стека;

- в строковых командах STOS, MOVS, SCAS и CMPS, имеющих два операнда-адреса, на которые указывают индексные регистры SI и DI, один из операндов (на который указывает SI) сегментируется по регистру DS, а другой (на него указывает DI) — по регистру ES.

Программные сегменты. Директива ASSUME. Рассмотрим, как сегментирование проявляется в программах на MASM. Для того чтобы указать, что некоторая группа операторов программы образует единый сегмент памяти, они оформляются как программный сегмент: перед ними ставится директива SEGMENT, после них — директива ENDS, причем в начале обеих этих директив должно быть указано одно и то же имя, играющее роль имени сегмента. Программа же в целом представляет собой последовательность таких программных сегментов, в конце которой указывается директива конца программы END, например:

```
DT1 SEGMENT; программный сегмент с именем DT1
A DB 0
B DW ?
DT1 ENDS
;
DT2 SEGMENT; программный сегмент DT2
C DB 'hello'
DT2 ENDS
;
CODE SEGMENT; программный сегмент CODE
ASSUME CS:CODE, DS:DT1, ES:DT2
PROG: MOV AX,DT2
MOV DS,AX
MOV BH,C
...
CODE ENDS
END PROG; конец текста программы.
```

Предложения программного сегмента ассемблер размещает в одном сегменте памяти (в совокупности они не должны занимать более 64 Кбайт), начиная с ближайшего свободного адреса, кратного 16. Номер (первые 16 битов начального адреса) этого сегмента становится значением имени сегмента. В MASM это

имя относится к константным выражениям, а не адресным, поэтому в команде

```
MOV AX,DT2,
```

второй операнд является непосредственным, поэтому в регистр AX будет записано начало (номер) сегмента DT2, а не содержимое начальной ячейки этого сегмента.

Имена же переменных (A, B, C) и метки (PROG) относятся к адресным выражениям, и им ставится в соответствие адрес их ячейки относительно «своего» сегмента:

- имени A соответствует адрес 0;
- имени B — адрес 1;
- имени C — адрес 0, а метке PROG — адрес 0.

Все ссылки на предложения одного программного сегмента ассемблер сегментирует по умолчанию по одному и тому же сегментному регистру, по какому именно — устанавливается специальной директивой ASSUME. В приведенном примере эта директива определяет, что все ссылки на сегмент CODE, если явно не указан сегментный регистр, должны сегментироваться по регистру CS, все ссылки на DT1 — по регистру DS, а все ссылки на DT2 — по регистру ES.

Встретив в тексте программы ссылку на какое-либо имя (например, на имя C в команде MOV AX,C), ассемблер определяет, в каком именно программном сегменте оно описано (здесь — в DT2), затем по информации из директивы ASSUME «узнает», какой сегментный регистр поставлен в соответствие этому сегменту (в данном случае — это ES), и далее образует адресную пару из данного регистра и смещения имени (здесь — ES:0), которую и записывает в формируемую машинную команду. При этом ассемблер учитывает используемое в i80x86 соглашение о сегментных регистрах по умолчанию — если в адресной паре, построенной им самим или явно заданной в программе, сегментный регистр совпадает с регистром по умолчанию, то в машинную команду заносится лишь смещение. Если, скажем, встретится команда

```
MOV CX,B,
```

тогда по имени B ассемблер построит пару DS:1, но если операнд — адрес команды MOV — по умолчанию сегментируется по регистру DS, то записывать этот регистр в машинную команду излишне, и ассемблер записывает в нее только смещение 1.

Таким образом, директива ASSUME избавляет программистов от необходимости выписывать полные адресные пары не только тогда, когда используются сегментные регистры по умолчанию (как в случае с именем B), но тогда, когда в машинной команде следовало бы явно указать сегментный регистр (как в случае с именем C). В MASM сегментный регистр в ссылке на имя требуется указывать лишь тогда, когда имя должно по каким-либо причинам сегментироваться по регистру, отличному от того, что поставлен в соответствие всему сегменту, в котором это имя описано.

Однако все это справедливо только при соблюдении следующих условий:

- директива ASSUME должна быть указана перед первой командой программы. В противном случае ассемблер, «просматривающий» текст программы сверху вниз, не будет знать, как сегментировать имена из команд, расположенных до этой директивы, и потому зафиксирует ошибку;
- в директиве ASSUME следует каждому сегменту ставить в соответствие сегментный регистр: если ассемблеру встретится ссылка на имя из сегмента, которому не соответствует никакой сегментный регистр, то он зафиксирует ошибку. Правда, в обоих случаях можно избежать ошибки, но для этого в ссылках необходимо явно указывать сегментный регистр.

Начальная загрузка (инициализация) сегментных регистров. Директива ASSUME сообщает ассемблеру о том, по каким регистрам он должен сегментировать имена, из каких сегментов, и гарантирует, что в этих регистрах будут находиться начальные адреса этих сегментов. Однако загрузку этих адресов в регистры сама директива не осуществляет.

Сделать такую загрузку — обязанность самой программы, с загрузки сегментных регистров и должно начинаться выполнение программы. Поскольку в i80x86 нет команды пересылки непосредственного операнда в сегментный регистр (а имя, т. е. начало, сегмента — это непосредственный операнд), то такую загрузку приходится делать через какой-то другой, несегментный, регистр (например, AX):

```
MOV AX,DT1 ;AX := начало сегмента DT1
MOV DS,AX ;DS := AX.
```

Аналогично загружается регистр ES.

Загружать регистр CS в начале программы не надо: он, как и счетчик команд IP, загружается операционной системой перед тем, как начинается выполнение программы (иначе нельзя было бы начать ее выполнение). Что же касается регистра SS, используемого для работы со стеком, то он может быть загружен так же, как и регистры DS и ES, однако в MASM предусмотрена возможность загрузки этого регистра еще до выполнения программы.

Ссылки вперед. «Встречая» в команде ссылку назад — имя, которое описано в тексте программы до этой команды, ассемблер уже имеет необходимую информацию об имени и потому может правильно оттранслировать эту команду. Но если встретится ссылка вперед или имя, которое не было описано до команды и которое, вероятно, будет описано позже, то ассемблер в большинстве случаев не сможет правильно оттранслировать такую команду. Например, не «зная», в каком программном сегменте будет описано это имя, ассемблер не может установить, по какому именно сегментному регистру надо сегментировать имя, и потому не может определить, надо или же нет размещать перед соответствующей машинной командой префикс замены сегмента и, если надо, то какой именно.

В подобной ситуации ассемблер действует следующим образом:

- если в команде встретилась ссылка вперед, то он делает некоторое предположение относительно этого имени и уже на основе этого предположения формирует машинную команду;
- если затем (когда встретится описание имени) окажется, что данное предположение было неверным, тогда ассемблер пытается исправить сформированную ранее машинную команду. Однако это не всегда удается: если правильная машинная команда должна занимать больше места, чем машинная команда, построенная на основе предположения (например, перед командой надо на самом деле вставить префикс замены сегмента), тогда ассемблер фиксирует ошибку (как правило, это ошибка номер 6: `Phase error between passes`).

Какие же «предположения» делает ассемблер, встречая ссылку вперед?

Во всех командах, кроме команд перехода, ассемблер «предполагает», что имя будет описано в сегменте данных и потому сегментируется по регистру DS. Это следует учитывать при со-

ставлении программы: если в команде встречается ссылка вперед на имя, которое описано в сегменте, на начало которого указывает сегментный регистр, отличный от DS, то перед таким именем автор программы должен написать соответствующий префикс. Пример:

```
CODE SEGMENT
ASSUME CS:CODE
X DW ?
BEG: MOV AX,X ;здесь вместо CS:X можно записать просто X
MOV CS:Y,AX   ;здесь обязательно надо записать CS:Y
...
Y DW ?
CODE ENDS.
```

Команды перехода (ветвления программы)

В систему команд i80x86 входит обычный для ЭВМ набор команд перехода — безусловные и условные переходы, переходы с возвратами и др. Однако в i80x86 эти команды имеют некоторые особенности, которые здесь и рассматриваются.

Абсолютный адрес команды, которая должна быть выполнена следующей, определяется парой CS:IP, поэтому выполнение перехода означает изменение этих регистров, обоих или только одного (IP):

- если изменяется только счетчик команд IP, то такой переход называется внутрисегментным или ближним (управление остается в том же сегменте команд);
- если меняются оба регистра CS и IP, то это межсегментный или дальний переход (начинают выполняться команды из другого сегмента команд).

По способу изменения счетчика команд переходы делятся на абсолютные и относительные:

- если в команде перехода указан адрес (смещение) той команды, которой надо передать управление, то это *абсолютный переход*;
- если в команде указана величина (сдвиг), которую надо добавить к текущему значению регистра IP, чтобы получился адрес перехода, тогда это будет относительный переход, при этом сдвиг может быть *положительным* и *отрицательным*, так что возможен переход вперед и назад.

По величине сдвига относительные переходы делятся на короткие (сдвиг задается байтом) и длинные (сдвиг-слово).

Абсолютные переходы делятся на прямые и косвенные:

- при *прямом* переходе адрес перехода задается в самой команде;
- при *косвенном* — в команде указывается регистр (ячейка памяти), в котором (которой) находится адрес перехода.

Безусловные переходы. В MASM все команды безусловного перехода обозначаются одинаково:

```
JMP op,
```

но в зависимости от типа операнда ассемблер формирует разные машинные команды.

Внутрисегментный относительный короткий переход.

```
JMP i8 (IP:=IP+i8).
```

Здесь `i8` обозначает непосредственный операнд размером в 1 байт, который интерпретируется как знаковое целое от –128 до 127. Команда прибавляет это число к текущему значению регистра `IP`, получая в нем адрес (смещение) той команды, которая должна быть выполнена следующей. Регистр `CS` при этом не меняется.

Необходимо учитывать следующую особенность регистра `IP`. Выполнение любой команды начинается с того, что в `IP` заносится адрес следующей за ней команды, и только затем выполняется собственно команда. Для команды относительного перехода это означает, что операнд `i8` прибавляется не к адресу этой команды, а к адресу команды, следующей за ней, поэтому, к примеру, команда `JMP 0` — это переход на следующую команду программы.

При написании машинной программы сдвиги для относительных переходов приходится вычислять вручную, однако MASM избавляет от этого неприятного занятия: в MASM в командах относительного перехода всегда указывается метка той команды, на которую надо передать управление, и ассемблер сам вычисляет сдвиг, который он и записывает в машинную команду. Отсюда следует, что в MASM команда перехода по метке воспринимается не как абсолютный переход, а как относительный.

По короткому переходу можно передать управление только на ближайшие команды программы, отстоящие от команды, следующей за командой перехода, до 128 байтов назад или до 127 байтов вперед.

Внутрисегментный относительный длинный переход используется для перехода на более дальние команды.

```
JMP i16 (IP:=IP+i16).
```

Здесь `i16` обозначает непосредственный операнд размером в слово, который рассматривается как знаковое целое от –32 768 до 32 767. Этот переход аналогичен короткому переходу.

Отметим, что, встретив команду перехода с меткой, которой была помечена одна из предыдущих (по тексту) команд программы, ассемблер вычисляет разность между адресом этой метки и адресом команды перехода и по этому сдвигу определяет, какую машинную команду относительного перехода (короткую или длинную) надо сформировать. Но если метка еще не встречалась в тексте программы, т. е. происходит переход вперед, тогда ассемблер, не зная еще адреса метки, не может определить, какую именно машинную команду относительного перехода формировать, поэтому он на всякий случай выбирает команду длинного перехода. Однако эта машинная команда занимает 3 байта, тогда как команда короткого перехода — 2 байта, и если автор программы на MASM стремится к экономии памяти и знает заранее, что переход вперед будет на близкую метку, то он должен сообщить об этом ассемблеру, чтобы тот сформировал команду короткого перехода. Такое указание делается с помощью оператора `SHORT`:

```
JMP SHORT L.
```

Для переходов назад оператор `SHORT` не нужен — «зная» адрес метки, ассемблер сам определит вид команды относительного перехода.

Внутрисегментный абсолютный косвенный переход.

```
JMP r16 (IP:=[r])
```

или

```
JMP m16 (IP:=[m16]).
```

Здесь r16 обозначает любой 16-битовый регистр общего назначения, а m16 — адрес слова памяти. В этом регистре (слове памяти) должен находиться адрес, по которому и будет произведен переход. Например, по команде JMP BX осуществляется переход по адресу, находящемуся в регистре BX.

Межсегментный абсолютный прямой переход.

```
JMP seg:ofs (CS := seg, IP := ofs).
```

Здесь seg — начало (первые 16 битов начального адреса) некоторого сегмента памяти, а ofs — смещение в этом сегменте. Пара seg:ofs определяет абсолютный адрес, по которому делается переход. В MASM эта пара всегда задается конструкцией FAR PTR <метка>, которая указывает, что надо сделать переход по указанной метке, причем эта метка — «дальняя», из другого сегмента. Отметим, что ассемблер сам определяет, какой это сегмент, и сам подставляет в машинную команду его начало, т. е. seg.

Межсегментный абсолютный косвенный переход.

```
JMP m32 (CS := [m32 + 2], IP := [m32]).
```

Здесь под m32 понимается адрес двойного слова памяти, в котором находится пара seg:ofs, задающая абсолютный адрес, по которому данная команда должна выполнить переход. Напомним, что в i80x86 величины размером в двойное слово хранятся в «перевернутом» виде, поэтому смещение ofs находится в первом слове двойного слова m32, а смещение seg — во втором слове (по адресу m32+2).

Команды межсегментного перехода используются тогда, когда команды программы размещены не в одном сегменте памяти, а в нескольких (например, если команд так много, что в совокупности они занимают более 64 Кбайт, т. е. больше максимального размера сегмента памяти). При переходе из одного такого сегмента в другой необходимо менять не только счетчик команд IP, но и содержимое регистра CS, загружая в последний начальный адрес второго сегмента. Такое одновременное изменение обоих этих регистров и делают команды межсегментного перехода.

Проблемы команд перехода. При записи в MASM команд перехода следует учитывать, что они могут восприниматься неоднозначно.

Вопрос — как воспринимать команду JMP A:

- как переход по метке A;
- как переход по адресу, хранящемуся в ячейке с именем A?

Кроме того, какой это переход — внутрисегментный или межсегментный? Ответ зависит от того, как описано имя A, и от того, когда описано имя A — до команды перехода или после нее.

Пусть A описано до команды перехода (*ссылка назад*). Если именем A помечена некоторая команда текущего сегмента команд (т. е. A — метка), тогда ассемблер формирует машинную команду внутрисегментного относительного перехода. Если же A — имя переменной, тогда ассемблер формирует машинную команду косвенного перехода — внутрисегментного, если A описано в директиве DW, или межсегментного, если A описано в директиве DD.

В случае же, если имя A описано после команды перехода (*ссылка вперед*), ассемблер всегда формирует машинную команду внутрисегментного относительного длинного перехода. С учетом этого имя A обязательно должно метить команду из текущего сегмента команд, иначе будет зафиксирована ошибка. Если такая трактовка ссылки вперед не удовлетворяет автора программы, тогда он обязан с помощью оператора SHORT или PTR уточнить тип имени A:

```
JMP SHORT A     ; внутрисегментный короткий переход по метке;
JMP WORD PTR A  ; внутрисегментный косвенный переход;
JMP DWORD PTE A; межсегментный косвенный переход.
```

Отметим, что переход по метке A из другого сегмента команд всегда должен указываться с помощью FAR PTR (независимо от того, описана метка A до команды перехода или после нее):

```
JMP FAR PTR A   ; межсегментный переход по метке.
```

Условные переходы. Практически во всех командах условного перехода проверяется значение того или иного флага (например, флага нуля ZF) и, если он имеет определенное значение, выполняется переход по адресу, указанному в команде. Значение флага должно быть установлено предыдущей командой, например командой сравнения

```
CMP op1,op2,
```

которая вычисляет разность op1 – op2, однако результат никуда не записывает, а только меняет флаги, на которые будет реагировать команда условного перехода.

В MASM команды условного перехода имеют следующую форму:

```
Jxx op,
```

где xx — одна или несколько букв, в сокращенном виде отражающих проверяемое условие (обычно в предположении, что перед этой командой находится команда сравнения).

Примеры некоторых мнемоник:

JE — переход «по равно» (jump if equal);
JNL — переход «по не меньше» (jump if not less).

Особенностью всех машинных команд условного перехода является то, что они реализуют внутрисегментный относительный короткий переход, т. е. добавляют к счетчику команд IP свой операнд, рассматриваемый как число со знаком от –128 до 127. В MASM этот операнд всегда должен записываться как метка, которую ассемблер заменит на соответствующий сдвиг.

Такая особенность команд условного перехода вызывает неудобство при переходах на «дальние» команды. Например, если надо сделать переход при A < B на команду, помеченную меткой L и расположенную далеко от команды перехода, то приходится использовать команду длинного безусловного перехода:

```
MOV AX,A
CMP AX,B ; сравнение A и B
JNL M    ; не меньше --> M (обход команды JMP)
JMP L    ; меньше --> L (длинный переход)
M: ...
```

Команды управления циклом

В i80x86 есть несколько команд, упрощающих программирование циклов с заранее известным числом повторений. Применение этих команд требует, чтобы к началу цикла в регистр CX было занесено число шагов цикла. Сами команды размещаются в конце цикла, они уменьшают значение CX на 1 и, если CX еще не

равно 0, передают управление на начало цикла. Например, найти S — сумму элементов массива X из 10 чисел-слов можно так:

```
MOV AX,0    ; начальное значение суммы (накапливается в AX)
MOV SI,0    ; начальное значение индексного регистра
MOV CX,10   ; число повторений цикла
L: ADD AX,X[SI]  ; AX:=AX+X[i]
ADD SI,2         ; SI:=SI+2
LOOP L           ; CX:=CX-1 ; if CX<>0 then goto L
MOV S,AX         ; S:=AX.
```

Помимо команды LOOP есть еще две «циклические» команды — LOOPZ и LOOPNZ (они имеют синонимичные названия LOOPE и LOOPNE), которые, кроме регистра CX, проверяют еще и флаг нуля ZF; например, команда LOOPZ «выходит» из цикла, если CX = 0 или ZF = 1. Эту команду можно, например, использовать при поиске в массиве первого нулевого элемента, где должно быть предусмотрено два условия выхода из цикла: либо будет найден нулевой элемент (ZF = 1, если перед LOOPZ поставить команду сравнения очередного элемента с 0), либо будет исчерпан весь массив (CX = 0).

Отметим, что все эти «циклические» команды реализуют короткий относительный переход, как и команды условного перехода, поэтому их можно использовать только для циклов с небольшим числом команд.

В MASM есть еще две команды перехода — CALL (переход с возвратом) и RET (возврат из подпрограммы).

Строковые операции

В i80x86 под строкой понимается последовательность соседних байтов или слов. В связи с этим все строковые команды имеют две разновидности:

- для работы со строками из байтов (в мнемонику операций входит буква B);
- для работы со строками из слов (в мнемонику входит W).

Имеются следующие операции над строками:

- пересылка элементов строк (в память, из памяти, память—память);
- сравнение двух строк;

- просмотр строки с целью поиска элемента, равного заданному.

Каждая из этих операций выполняется только над одним элементом строки, однако одновременно происходит автоматическая настройка на следующий или предыдущий элемент строки. Предусмотрены команды повторения (REP и др.), которые заставляют следующую за ними строковую команду многократно повторяться (до 2^{16} раз), в связи с этим такая пара команд позволяет обработать всю строку, причем намного быстрее, чем запрограммированный цикл.

Кроме того, строки можно просматривать вперед (от их начала к концу) и назад. Направление просмотра зависит от флага направления DF, значение которого можно менять с помощью команд STD (DF := 1) и CLD (DF := 0). При DF = 0 все последующие строковые команды программы просматривают строки вперед, а при DF = 1 — назад.

В строковых командах операнды явно не указываются, а подразумеваются. Если команда работает с одной строкой, то адрес очередного, обрабатываемого сейчас элемента строки задается парой регистров DS и SI или парой ES и DI, а если команда работает с двумя строками, то адрес элемента одной из них определяется парой DS:SI, а адрес элемента другой — парой ES:DI. После выполнения операции значение регистра SI и/или DI увеличивается (при DF = 0) или уменьшается (при DF = 1) на 1 (для байтовых строк) или на 2 (для строк из слов).

Начальная установка всех этих регистров, а также флага DF должна быть выполнена до начала операции над строкой. Если сегментный регистр DS уже имеет нужное значение, тогда загрузить регистр SI можно с помощью команды

```
LEA SI,<начальный/конечный адрес строки>.
```

Если же надо загрузить сразу оба регистра DS и SI, тогда можно воспользоваться командой

```
LDS SI,m32,
```

которая в регистр SI заносит первое слово, а в регистр DS — второе слово из двойного слова, имеющего адрес m32 (таким образом, по адресу m32+2 должен храниться сегмент, а по адресу m32 — смещение начального или конечного элемента строки).

Начальную загрузку регистров ES и DI обычно осуществляют одной командой:

```
LES DI,m32,
```

которая действует аналогично команде LDS.

Перечислим вкратце строковые команды i80x86:

- команда загрузки элемента строки в аккумулятор (LODSB или LODSW) пересылает в регистр AL или AX очередной элемент строки, на который указывает пара DS:SI, после этого увеличивает (при DF = 0) или уменьшает (при DF = 1) регистр SI на 1 или 2;
- запись аккумулятора в строку (STOSB или STOSW); содержимое регистра AL или AX заносится в тот элемент строки, на который указывает пара ES:DI, затем изменяет регистр DI на 1 или 2;
- пересылка строк (MOVSB или MOVSW); элемент первой строки, определяемый парой DS:SI, заносится в элемент второй строки, определяемый парой ES:DI, затем одновременно меняет регистры SI и DI;
- сравнение строк (CMPSB или CMPSW); сравниваются очередные элементы строк, указываемые парами DS:SI и ES:DI, и результат сравнения (равно, меньше и т. п.) фиксирует в флагах, после этого меняется содержание регистров SI и DI;
- сканирование строки (SCASB или SCASW); сравнивается элемент строки, адрес которого задается парой ES:DI, со значением регистра AL или AX и результат сравнения фиксирует в флагах, затем изменяется содержимое регистра DI.

Перед любой строковой командой можно поставить одну из двух команд, называемых «префиксами повторения», которая заставит многократно повториться эту строковую команду. Число повторений (обычно это длина строки) должно быть указано в регистре CX.

Префикс повторения REPZ (синонимы — REPE, REP) сначала заносит 1 в флаг нуля ZF, после этого, постоянно уменьшая CX на 1, заставляет повторяться следующую за ним строковую команду до тех пор, пока в CX не окажется «0» или пока флаг ZF не изменит свое значение на «0».

Другой префикс повторения REPNZ (синоним — REPNE) действует аналогично, но только вначале устанавливает флаг ZF

в «0», а при изменении его на «1» прекращает повторение строковой команды.

Пример. Пусть надо переписать 10 000 байтов начиная с адреса A в другое место памяти, начиная с адреса B. Если оба эти имени относятся к сегменту данных, на начало которого указывает регистр DS, тогда эту пересылку можно сделать так:

```
CLD           ; DF:=0 (просмотр строки вперед)
MOV CX,1000   ; CX — число повторений
MOV AX,DS
MOV ES,AX     ; ES:=DS
LEA SI,A      ; ES:SI — «откуда»
LEA DI,B      ; DS:DI — «куда»
REP MOVSB     ; пересылка CX байтов.
```

Стек. Подпрограммы

Стек. В i80x86 имеются специальные команды работы со *стеком,* т. е. областью памяти, доступ к элементам которой осуществляется по принципу «последним записан — первым считан». Но для того чтобы можно было воспользоваться этими командами, необходимо соблюдение ряда условий.

Во многих случаях программе требуется временно запомнить информацию, а затем считывать ее в обратном порядке. Эта проблема в ПК решена посредством реализации стека LIFO («последний пришел — первый ушел»), называемого также стеком включения/извлечения (stack — кипа, например бумаг). Наиболее важное использование стека связано с процедурами. Стек обычно рассчитан на косвенную адресацию через регистр SP — указатель стека. При включении элементов в стек производится автоматический декремент указателя стека, а при извлечении — инкремент, т. е. стек всегда «растет» в сторону меньших адресов памяти. Адрес последнего включенного в стек элемента называется вершиной стека (TOS).

Под стек можно отвести область в любом месте памяти. Размер ее может быть любым, но не должен превосходить 64 Кбайт, а ее начальный адрес должен быть кратным 16. Другими словами, эта область должна быть сегментом памяти; он называется сегментом стека. Начало этого сегмента (первые 16 битов на-

чального адреса) должно обязательно храниться в сегментном регистре SS.

Хранимые в стеке элементы могут иметь любой размер, однако следует учитывать, что в i80x86 имеются команды записи в стек и чтения из него только слов. Поэтому для записи байта в стек его надо предварительно расширить до слова, а запись или чтение двойных слов осуществляются парой команд.

В i80x86 принято заполнять стек снизу вверх, от больших адресов к меньшим: первый элемент записывается в конец области, отведенной под стек, второй элемент — в предыдущую ячейку области и т. д. Считывается всегда элемент, записанный в стек последним. В связи с этим нижняя граница стека всегда фиксирована, а верхняя — меняется. Слово памяти, в котором находится элемент стека, записанный последним, называется вершиной стека. Адрес вершины, отсчитанный от начала сегмента стека, обязан находиться в указателе стека — регистре SP. Таким образом, абсолютный адрес вершины стека определяется парой SS:SP.

Значение «0» в регистре SP свидетельствует о том, что стек полностью заполнен (его вершина «дошла» до начала области стека). Поэтому для контроля за переполнением стека надо перед новой записью в стек проверять условие SP = 0 (i80x86 этого не делает). Для пустого стека значение SP должно равняться размеру стека, т. е. пара SS:SP должна указывать на байт, следующий за последним байтом области стека. Контроль за чтением из пустого стека, если надо, обязана делать сама программа.

Начальная установка регистров SS и SP может быть произведена в самой программе, однако в MASM предусмотрена возможность автоматической загрузки этих регистров. Если в директиве SEGMENT, начинающей описание сегмента стека, указать параметр STACK, тогда ассемблер (точнее, загрузчик) перед тем, как передать управление на первую команду машинной программы, загрузит в регистры SS и SP нужные значения. Например, если в программе сегмент стека описан следующим образом:

```
ST SEGMENT STACK
DB 256 DUP(?); размер стека — 256 байтов
ST ENDS,
```

и если под этот сегмент была выделена область памяти, начиная с абсолютного адреса 12340h, тогда к началу выполнения про-

граммы в регистре SS окажется величина 1234h, а в регистре SP — величина 100h (=256).

Отметим, что эти значения соответствуют пустому стеку.

Основные стековые команды. При соблюдении указанных требований в программе можно использовать команды, предназначенные для работы со стеком. Основными из них являются следующие.

Запись слова в стек:

```
PUSH op
```

Здесь op обозначает любой 16-битовый регистр (в том числе и сегментный) или адрес слова памяти. По этой команде значение регистра SP уменьшается на 2 (вычитание происходит по модулю 2^{16}), затем указанное операндом слово записывается в стек по адресу SS:SP.

Чтение слова из стека:

```
POP op
```

Слово, считанное из вершины стека, присваивается операнду op (регистру, в том числе сегментному, но не CS, или слову памяти), затем значение SP увеличивается на 2.

Переход с возвратом:

```
CALL op
```

Эта команда записывает адрес следующей за ней команды в стек и затем делает переход по адресу, определяемому операндом op. Она используется для переходов на подпрограммы с запоминанием в стеке адреса возврата. Имеются следующие разновидности этой команды (они аналогичны вариантам команды безусловного перехода JMP):

- *внутрисегментный относительный длинный* переход (op — непосредственный операнд размером в слово, а в MASM — это метка из текущего сегмента команд или имя близкой процедуры (см. ниже)); в этом случае в стек заносится только текущее значение счетчика команд IP, т. е. смещение следующей команды;
- *внутрисегментный абсолютный косвенный* переход (op — адрес слова памяти, в которой находится ад-

рес (смещение) той команды, на которую и будет сделан переход); и здесь в стек записывается только смещение адреса возврата;

- *межсегментный абсолютный прямой* переход (op — непосредственный операнд вида SEG:OFS, а в MASM — это FAR PTR <метка> или имя дальней процедуры); здесь в стек заносятся текущие значения регистров CS и IP (первым в стек записывается содержимое CS), т. е. абсолютный адрес возврата, затем изменяются регистры CS и IP;
- *межсегментный абсолютный косвенный* переход (op — адрес двойного слова, в котором находится пара seg:ofs, задающая абсолютный адрес перехода); и здесь в стеке спасается содержимое регистров CS и IP.

Переход (возврат) по адресу из стека:

```
RET op
```

Из стека считывается адрес и по нему производится переход. Если указан операнд (а это должно быть неотрицательное число), то после чтения адреса стек еще очищается на это число байтов (к SP добавляется это число). Команда используется для возврата из подпрограммы по адресу, записанному в стек по команде CALL при вызове подпрограммы, и одновременной очистки стека от параметров, которые основная программа занесла в стек перед обращением к подпрограмме.

Команда RET имеет две разновидности (хотя в MASM они записываются одинаково):

- в одном случае из стека считывается только одно слово — смещение адреса возврата;
- во втором — из стека считывается пара seg:ofs, указывающая абсолютный адрес возврата. Как ассемблер определяет, какой из этих двух случаев имеет место, объяснено ниже.

В i80x86 стек в основном используется для организации подпрограмм и прерываний. Однако даже если программе не нужен стек, она все равно должна отвести под него место. Дело в том, что стеком будет неявно пользоваться операционная система при обработке прерываний, которые возникают (например, при нажатии клавиш на клавиатуре) в то время, когда выполняется программа. Для нужд ОС рекомендуется выделять в стеке 64 байта.

Прерывания и подпрограммы

Прерывания. Иногда необходимо выполнить одну из набора специальных процедур, если в системе или в программе возникают определенные условия, например, нажата клавиша на клавиатуре. Действие, стимулирующее выполнение одной из таких процедур, называется прерыванием, поскольку основной процесс при этом приостанавливается на время выполнения этой процедуры.

Существует два общих класса прерываний — внутренние и внешние. Первые инициируются состоянием ЦП или командой, а вторые — сигналом, подаваемым от других компонентов системы. Типичные внутренние прерывания: деление на нуль, переполнение и т. п., а типичные внешние — запрос на обслуживание со стороны какого-либо устройства ввода-вывода.

Переход к процедуре прерывания осуществляется из любой программы, а после выполнения процедуры прерывания обязательно происходит возврат в прерванную программу. Перед обращением к процедуре прерывания должно быть сохранено состояние всех регистров и флагов, используемых процедурой прерывания, а после окончания прерывания эти регистры должны быть восстановлены.

Прерывание вынуждает процессор прекратить выполнение одной последовательности команд и начать выполнение другой, при этом адрес очередной команды, которая должна была бы выполняться (содержимое регистра `IP`), если бы не было прерывания, запоминается. Адрес команды, которая должна выполняться после возникновения прерывания, выбирается из таблицы, хранящейся в начальной области памяти. Эта таблица называется таблицей векторов прерываний. В таблице записано 256 адресов. Когда устройство вызывает прерывание процессора, оно сообщает ему, какой адрес из таблицы следует использовать для перехода к новой последовательности команд.

Аппаратное прерывание. Типичным примером устройства, требующего прерывания, является клавиатура. После нажатия клавиши на клавиатуре в систему прерываний передается сигнал, приводящий к прерыванию работы процессора. Очевидно, что процессор прекращает текущую работу и, используя адрес, переданный из системы прерываний, начинает выполнение специальной программы взаимодействия с клавиатурой. Программа вводит с клавиатуры код, соответствующий нажатой клавише, и

заносит его в область сохранения, расположенную в памяти. Затем программа взаимодействия с клавиатурой возвращает управление программе, которая выполнялась до прерывания.

Программные прерывания. Прерывания также могут иметь место при выполнении специальной команды — прервать, используя адрес X. (Число X указывает один из адресов в таблице векторов прерываний.) При выполнении команды «прервать» процессор определяет и запоминает адрес команды, которая должна была бы выполняться следующей, а затем переходит к выполнению программы, начинающейся с адреса X, извлеченного из таблицы векторов прерываний. Такая последовательность действий называется программным прерыванием.

Подпрограммы. Типичная схема организации подпрограмм, обычно используемая трансляторами с языков высокого уровня для реализации процедур и функций (в частности, рекурсивных), следующая.

При обращении к подпрограмме в стек заносятся параметры для нее и адрес возврата, после этого делается переход на ее начало:

```
PUSH param1   ; запись 1-го параметра в стек
...
PUSH paramk   ; запись последнего (k-го) параметра в стек
CALL subr     ; переход с возвратом на подпрограмму.
```

Если необходимо вычислить параметр или его размер отличен от слова, тогда для записи параметра в стек необходимо несколько команд.

Первыми командами подпрограммы обычно являются следующие:

```
PUSH BP     ; сохранить в стеке старое значение BP
MOV SP,BP   ; установить BP на вершину стека
SUB SP,m    ; отвести в стеке место (m байтов) под локальные
              величины подпрограммы.
```

Поясним эти «входные» команды. В подпрограмме для обращения к ячейкам стека, занятым параметрами, используется (как базовый) регистр BP: если в BP занести адрес вершины стека, то для доступа к этим ячейкам следует использовать адресные выражения вида i[BP] или, что то же самое, [BP+i]. (От-

метим, что применять здесь регистры-модификаторы BX, SI и DI нельзя, так как формируемые по ним исполнительные адреса будут сегментироваться по умолчанию по регистру DS, а в данном случае следует сегментировать по SS.) Однако данная подпрограмма может быть вызвана из другой, также использующей регистр BP, поэтому прежде чем установить BP на «вершину стека», надо сохранить в стеке старое значение этого регистра, что и делает первая из «входных» команд.

Вторая же команда устанавливает BP на «вершину стека». Если предположить, что каждый параметр и адрес возврата занимают по слову памяти, тогда доступ к первому параметру обеспечивается адресным выражением [BP+4], ко второму — выражением [BP+6] и т. д.

Подпрограмме может потребоваться место для ее локальных величин. Такое место обычно отводится в стеке (а для рекурсивных подпрограмм только в стеке) «над» ячейкой, занимаемой старым значением BP. Если под эти величины нужно m байтов, то такой «захват» места можно реализовать простым уменьшением значения регистра SP на m, что и делает 3-я «входная» команда. Доступ к локальным величинам обеспечивается адресными выражениями вида [BP-i]. Если подпрограмме не нужно место под локальные величины, тогда третью из «входных» команд следует опустить.

Выход из подпрограммы реализуется следующими командами:

```
MOV SP,BP     ; очистить стек от локальных величин
POP BP        ; восстановить старое значение BP
RET 2×k       ; возврат из подпрограммы и очистка стека
```

от параметров (предполагается, что они занимают $2 \times k$ байтов).

Первая из этих «выходных» команд заносит в регистр SP адрес той ячейки стека, которая содержит «старое» значение регистра BP, т. е. происходит очистка стека от локальных величин (если их не было, то данную команду следует опустить). Вторая команда восстанавливает в BP это старое значение, одновременно удаляя его из стека. В этот момент состояние стека будет таким же, как и перед входом в подпрограмму. Третья команда считывает из стека адрес возврата (в результате SP «опускается» на 2 байта), затем добавляет к SP число, которое должно равняться числу байтов, занимаемых всеми параметрами подпро-

граммы, и затем осуществляет переход по адресу возврата. В этот момент состояние стека будет таким же, каким оно было перед обращением к подпрограмме.

Контрольные вопросы

1. Что такое система команд? Что такое команда? Что описывает команда?
2. Какого рода информацию может содержать адресная часть команды?
3. Приведите примеры команд одно-, двух-, трехадресных.
4. Каким образом процессор при выполнении программы осуществляет выбор очередной команды?
5. Опишите основной цикл процесса обработки команд.
6. В чем различие функций СчАК и РК?
7. Что такое индексирование и базирование?
8. Что такое регистровая память?
9. Какие типы процессоров вам известны?
10. Что такое суперскаляризация?
11. В чем заключается технология конвейерной обработки?
12. Охарактеризуйте технологию динамического исполнения.
13. Что такое 3D Now?
14. Перечислите основные этапы развития процессоров Intel, AMD, Cyrix.

Глава 4
АРХИТЕКТУРЫ ОБРАМЛЕНИЯ. ИНТЕРФЕЙСЫ. ОПЕРАТИВНАЯ ПАМЯТЬ

Обращаясь к рис. 2.2, замечаем, что из основных компонентов архитектуры остались нерассмотренными важные функции — памяти и коммутационно-коммуникационная. Поэтому в настоящей главе более подробно будут освещены принципы оперативной памяти (ОП), интерфейсы устройств ЭВМ и общая конфигурация их взаимодействия между собой и с ЦП, которую обеспечивает набор микросхем системной платы ПК — чипсет.

4.1. Организация оперативной памяти

Запоминающие устройства (ЗУ), именуемые также *устройствами памяти*, предназначены для хранения данных. Основными характеристиками ЗУ являются:

- емкость памяти, измеряемая в *битах* либо *байтах*;
- методы доступа к данным;
- быстродействие (время обращения к устройству).

Основные принципы построения ОП

Необходимо отметить, что все распространенные операционные системы, если для работы нужно больше памяти, чем физически присутствует в компьютере, не прекращают работу, а сбрасывают неиспользуемое в данный момент содержимое памяти в дисковый файл (называемый свопом — swap) и затем по мере необходимости «перегоняют» данные между ОП и свопом. Это гораздо медленнее, чем доступ системы к самой ОП. Поэто-

му от количества оперативной памяти напрямую зависит скорость системы.

Адресация данных. Команды, исполняемые ЭВМ при выполнении программы, равно как и числовые и символьные операнды, хранятся в памяти компьютера. Память состоит из миллионов ячеек, в каждой из которых содержится один бит информации (значения 0 или 1). Биты редко обрабатываются поодиночке, а, как правило, группами фиксированного размера. Для этого память организуется таким образом, что группы по *n* бит могут записываться и считываться за одну операцию. Группа *n* бит называется *словом*, а значение *n* — длиной слова. Схематически память компьютера можно представить в виде массива слов (рис. 4.1).

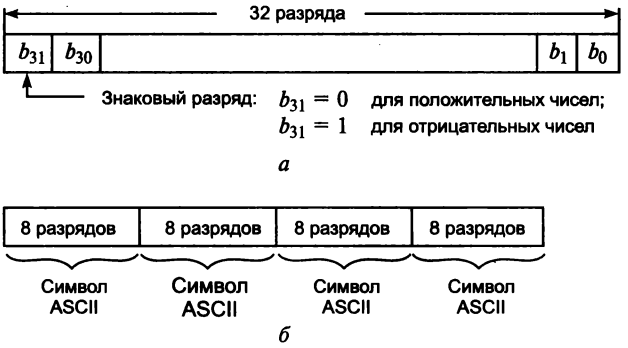


Рис. 4.1. Размещение числовой (*а*) и символьной (*б*) информации в слове

Обычно длина машинного слова компьютеров составляет от 16 до 64 бит. Если длина слова равна 32 битам, в одном слове может храниться 32-разрядное число в дополнительном коде или четыре символа ASCII, занимающих 8 бит каждый. Восемь идущих подряд битов являются байтом. Для представления машинной команды требуется одно или несколько слов.

Основная память соединяется с процессором посредством адресной шины и шины данных. Каждая шина состоит из совокупности электрических цепей (линий или бит). Ширина (разрядность) адресной шины определяет, сколько адресов может быть в ОП (адресное пространство), а шины данных — сколько данных может быть передано за один цикл. Например, в 1985 г. процессор Intel 386 имел 32-разрядную адресную шину, что давало возможность поддерживать адресное пространство в

4 Гбайт. В процессоре Pentium (1993 г.) ширина шины данных была увеличена до 64 бит, что позволяет передавать 8 байт информации одновременно.

Байтовая адресация. Отдельные биты, как правило, не адресуются и чаще всего адреса назначаются байтам памяти. Память, в которой каждый байт имеет отдельный адрес, называется памятью с байтовой адресацией. Последовательные байты имеют адреса 0, 1, 2 и т. д. Таким образом, при использовании слов длиной 32 бита последовательные слова имеют адреса 1, 4, 8, ..., и каждое слово состоит из 4 байт.

Прямой и обратный порядок байтов. Существует два способа адресации байтов в словах:

- в прямом порядке (рис. 4.2, *а*);
- обратном порядке (рис. 4.2, *б*).

Обратным порядком байтов (*big-endian*) называется система адресации, при которой байты адресуются слева направо, так что самый старший байт слова (расположенный с левого края) имеет наименьший адрес.

Прямым порядком байтов (*little-endian*) называется противоположная система адресации, при которой байты адресуются справа налево, так что наименьший адрес имеет самый младший байт слова (расположенный с правого края). Слова «старший» и «младший» определяют вес бита, т. е. степень двойки, соответствующей данному биту, когда слово представляет число. В ПЭВМ на основе 80x86 используется прямой порядок, а в ПЭВМ на основе Motorola 68000 — обратный. В обеих этих системах адреса байтов 0, 4, 8 и т. д. применяются в качестве адресов последовательных слов памяти в операциях чтения и записи слов.

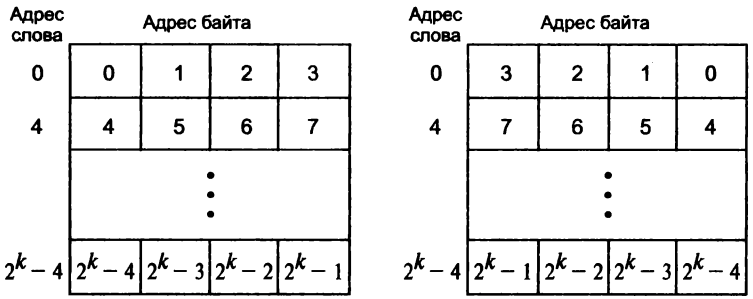


Рис. 4.2. Способы адресации байтов в ОП

Наряду с порядком байтов в слове важно также определить порядок битов в байте. Наиболее естественный порядок битов для кодирования числовых данных (непосредственно соответствующий их разрядам) — «слева направо»: b_{32}, ..., b_1, b_0. Однако существуют компьютеры, для которых характерен обратный порядок битов.

Расположение слов в памяти. В случае 32-разрядных слов их естественные границы располагаются по адресам 0, 4, 8 и т. д. При этом считается, что слова *выровнены по адресам* в памяти. Если говорить в общем, слова считаются выровненными в памяти в том случае, если адрес начала каждого слова кратен количеству байтов в нем. По практическим причинам, связанным с манипулированием двоично-кодированными адресами, количество байтов в слове обычно является степенью двойки. Поэтому, если длина слова равна 16 бит (2 байтам), выровненные слова начинаются по байтовым адресам 0, 2, 4, ..., а если она равна 64 бит (2^3, т. е. 8 байтам), то выровненные слова начинаются по байтовым адресам 0, 8, 16, ...

Не существует причины, по которой слова не могли бы начинаться с произвольных адресов. Такие слова называются невыровненными. Как правило, слова выравниваются по адресам памяти, но иногда этот принцип нарушается.

Обычно число занимает целое слово, поэтому для того чтобы обратиться к нему, нужно указать адрес слова, по которому оно хранится. Точно так же доступ к отдельно хранящемуся в памяти символу осуществляется по адресу содержащего его байта.

Адресное пространство. Для доступа к памяти необходимы имена или адреса, определяющие расположение данных в памяти. В качестве адресов традиционно используются числа из диапазона от 0 до $2^k - 1$ со значением k, достаточным для адресации всей памяти компьютера. Все 2^k адресов составляют адресное пространство компьютера. Следовательно, память состоит из 2^k адресуемых элементов. Например, использование 24-разрядных (как в процессоре 80286) адресов позволяет адресовать 2^{24} (16 777 216) элементов памяти. Обычно это количество адресуемых элементов обозначается как 16 Мбайт (1 Мбайт = 2^{20} = = 1 048 576 байт, адресное пространство 8086 и 80186). Поскольку у процессоров 80386, 80486 Pentium и их аналогов 32-разрядные адреса, им соответствует адресное пространство в 2^{32} байт, или 4 Гбайт.

Адресное пространство ЭВМ графически может быть изображено прямоугольником, одна из сторон которого представляет разрядность адресуемой ячейки (слова) процессора, а другая сторона — весь диапазон доступных адресов для этого же процессора. Диапазон доступных адресов процессора определяется разрядностью шины адреса системной шины. При этом минимальный номер ячейки памяти (адрес) будет равен 0, а максимальный определяется из формулы $M = 2^n - 1$.

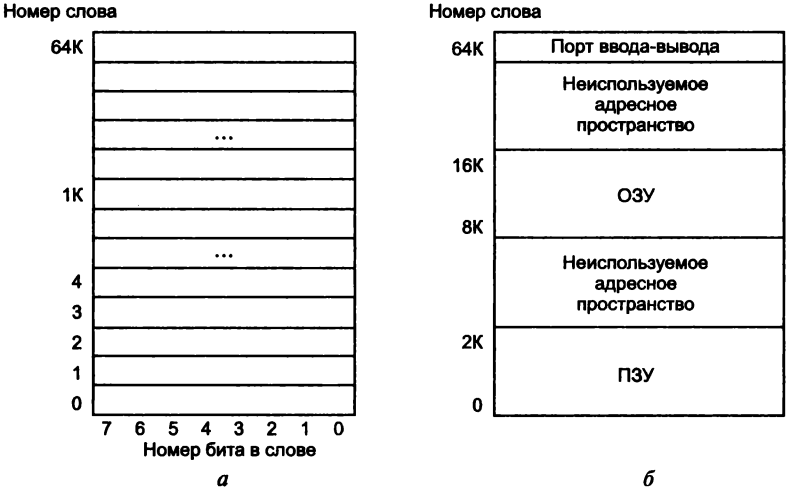


Рис. 4.3. Адресное пространство шестнадцатиразрядной шины адреса (*а*); распределение памяти процессора с 16-разрядной шиной адреса (*б*)

Для 16-разрядной шины это будет 65 535 (64 K) — рис. 4.3, *а*. Пример соответствующего распределения памяти приводится на рис. 4.3, *б*.

Иерархическая организация памяти

Компромиссом между производительностью и объемами памяти является решение использовать иерархию запоминающих устройств, т. е. применять иерархическую модель памяти.

Применение иерархических систем памяти оправдывает себя вследствие двух важных факторов — принципа локально-

сти обращений и низкого (экономически выгодного) соотношения стоимость/производительность. Принцип локальности обращений состоит в том, что большинство программ обычно не выполняют обращений ко всем своим командам и данным равновероятно, а в каждый момент времени оказывают предпочтение некоторой части своего адресного пространства.

Иерархия памяти обычно состоит из многих уровней, но в каждый момент времени взаимодействуют только два близлежащих уровня. Минимальная единица информации, которая может присутствовать либо отсутствовать в двухуровневой иерархии, называется блоком или строкой.

Успешное или неуспешное обращение к более высокому уровню называют соответственно попаданием (hit) или промахом (miss). Попадание — есть обращение к объекту в памяти, который найден на более высоком уровне, в то время как промах означает, что он не найден на этом уровне.

Доля попаданий (hit rate) — доля обращений к данным, найденным на более высоком уровне. Доля промахов (miss rate) — это доля обращений к данным, которые не найдены на более высоком уровне.

Время обращения при попадании (hit time) есть время обращения к более высокому уровню иерархии, которое включает в себя, в частности, и время, необходимое для определения того, является ли обращение попаданием или промахом. Потери на промах (miss penalty) есть время для замещения блока в более высоком уровне на блок из более низкого уровня плюс время для пересылки этого блока в требуемое устройство (обычно в процессор).

Потери на промах далее включают в себя две компоненты:

- время доступа (access time) — время обращения к первому слову блока при промахе;
- время пересылки (transfer time) — дополнительное время для пересылки оставшихся слов блока. Время доступа связано с задержкой памяти более низкого уровня, а время пересылки — с полосой пропускания канала между устройствами памяти двух смежных уровней.

Кэш-память или cache memory — компонента иерархической памяти — представляет собой буферное ЗУ, работающее со скоростью, обеспечивающей функционирование ЦП без режимов ожидания (рис. 4.4).

Рис. 4.4. Иерархия оперативной памяти

Необходимость создания кэш-памяти возникла потому, что появились процессоры с высоким быстродействием. Между тем для выполнения сложных прикладных процессов нужна большая память. Использование же большой сверхскоростной памяти экономически невыгодно. Поэтому между ОП и процессором стали устанавливать меньшую по размерам высокоскоростную буферную память, или *кэш-память*. В дальнейшем она была разделена на два уровня — *встроенная* в процессор (on-die) и *внешняя* (on-motherboard).

Кэш 1-го уровня (Level 1 cache, или L1). Кэш 1-го уровня, или первичный кэш, находится на плате центрального процессора и используется для временного хранения команд и данных, организованных в блоки по 32 байта. Первичный кэш — самая быстрая форма памяти. Будучи встроенным в чип, он обеспечивает минимальную задержку интерфейса с АЛУ, однако ограничен в размере. L1-кэш реализуется, используя принцип статической оперативной памяти (SRAM), и длительное время в среднем имел размер 16 Кбайт.

Процессор P55 Pentium MMX, выпущенный в начале 1997 г., содержал кэш 1-го уровня размером до 32 Кбайт. Процессоры AMD K6 и Cyrix M2, вышедшие в том же году, уже обеспечивали 64 Кбайт объема кэша 1-го уровня.

Кэш 2-го уровня (Level 2 cache, или L2). Кэш 2-го уровня (вторичный кэш) использует ту же самую логику управления, что и кэш 1-го уровня, и также относится к типу SRAM.

Кэш 2-го уровня обычно имеет два размера — 256 или 512 Кбайт и помещается на системной плате в гнезде типа Card Edge Low Profile (CELP) или в модуле «кэш-на-плате» («cache on a stick» — COAST). Последний напоминает SIMM, но немного короче и включает гнездо COAST, которое обычно расположено близко к процессору и напоминает слот PCI. В процессоре Pentium, однако, кэш 2-го уровня помещался на чипе процессора

непосредственно. При этом нередко используются две кэш-памяти, одна из которых хранит команды, а другая — данные.

Цель кэша 2-го уровня состоит в том, чтобы поставлять сохраненную информацию на процессор без какой-либо задержки (состояния ожидания). Для этой цели интерфейс шины процессора имеет специальный протокол передачи, названный *групповым* (или пакетным) режимом (burst mode). При этом обычно используется синхронный тип памяти, управляемой тактовым генератором ЦП. Цикл пакета состоит из четырех передач данных, где на адресную шину выводится адрес только первых 64 бит. Обычно кэш 2-го уровня — это синхронная пакетно-конвейерная память (Pipelined Burst Static RAM — PB SRAM) [16].

Эксклюзивным называется кэш, в котором данные, хранящиеся в кэш-памяти первого уровня, не обязательно должны быть продублированы в кэшах нижележащих уровней.

Инклюзивный кэш — когда любая информация, хранящаяся в кэшах высших уровней, дублируется в кэш-памяти.

Применение кэширования особенно эффективно, когда доступ к данным осуществляется преимущественно в последовательном порядке. Тогда после первого запроса на чтение данных, расположенных в медленной (кэшируемой) памяти, можно заранее (*упреждающее чтение*) выполнить чтение следующих блоков данных в кэш-память для того, чтобы при следующем запросе на чтение данных почти мгновенно выдать их из кэш-памяти.

Стратегии управления иерархической памятью

При построении систем с иерархической памятью целью является получение максимальной производительности подсистемы памяти при ее минимальной стоимости. Эффективность той или иной системы кэш-памяти зависит от стратегии управления памятью. Стратегия управления памятью включает:

- метод отображения основной памяти на кэш;
- порядок замещения информации в кэше;
- алгоритм взаимодействия между основной и кэш-памятью.

Отображение памяти на кэш. Существует три основных способа размещения блоков (строк) основной памяти в кэше:

- кэш-память с прямым отображением (direct-mapped cache);

- полностью ассоциативная кэш-память (fully associative cache);
- частично ассоциативная (или множественно ассоциативная, partial associative, set-associative cache) кэш-память.

Память с прямым отображением. В этом случае каждый блок основной памяти имеет только одно фиксированное место, на котором он может появиться в кэш-памяти. Все блоки основной памяти, имеющие одинаковые младшие разряды в своем адресе, попадают в один блок кэш-памяти. При таком подходе справедливо соотношение:

```
(Адрес блока кэш-памяти) = (Адрес блока основной
    памяти) mod (Число блоков в кэш-памяти).
```

Этот тип памяти наиболее прост, но и наименее эффективен, так как данные из разных областей памяти могут конфликтовать из-за единственной строки кэша, где они только и могут быть размещены.

Полностью ассоциативная память может отображать содержимое любой области памяти в любую область кэша, но при этом крайне сложна с точки зрения схемотехнической реализации.

Частично ассоциативный кэш является наиболее распространенным в данный момент среди процессорных архитектур. Характеризуется тем или иным количеством *n* «каналов» (степенью ассоциативности, «n-way») и может отображать содержимое данной строки памяти на каждую из *n* своих строк. Этот вариант является разумным компромиссом между полностью ассоциативным и кэшем «прямого отображения».

В современных процессорах, как правило, используется либо кэш-память с прямым отображением, либо двух- (четырех-) канальная множественно ассоциативная кэш-память. Например, в архитектурах K7 и K8 применяется 16-канальный частично ассоциативный кэш L2.

Порядок замещения информации в кэше определяет блок, подлежащий замещению при возникновении промаха. Простота при использовании кэша с прямым отображением заключается в том, что аппаратные решения здесь наиболее простые: легко реализуется сама аппаратура, легко происходит замещение данных. При замещении просто нечего выбирать — на попадание

проверяется только один блок и только этот блок может быть замещен.

При полностью или частично ассоциативной организации кэш-памяти имеются несколько блоков, из которых надо выбрать кандидата в случае промаха. Как правило, для замещения блоков применяются две основные стратегии:

- случайная (Random) — блоки-кандидаты выбираются случайно (чтобы иметь равномерное распределение). В некоторых системах используют псевдослучайный алгоритм замещения;
- замещается тот блок, который не использовался дольше всех (LRU — Least-Recently Used). В этом случае, чтобы уменьшить вероятность удаления информации, которая скоро может потребоваться, все обращения к блокам фиксируются.

Достоинство случайного способа заключается в том, что его проще реализовать в аппаратуре. Когда количество блоков увеличивается, алгоритм LRU становится все более дорогим и часто только приближенным.

Алгоритмы обмена с кэш-памятью (свопинга) включают следующие разновидности:

- алгоритм сквозной записи (Write Through) или сквозного накопления (Store Through);
- алгоритм простого свопинга (Simple Swapping) или обратной записи (Write Back);
- алгоритм свопинга с флагами (Flag Swapping) или обратной записи в конфликтных ситуациях с флагами (CUX);
- алгоритм регистрового свопинга с флагами (FRS).

Алгоритм сквозной записи — самый простой алгоритм свопинга. Каждый раз при появлении запроса на запись по некоторому адресу обновляется содержимое области по этому адресу как в быстрой, так и в основной памяти, даже если копия содержимого по этому адресу находится в быстром буфере. Такое постоянное обновление содержимого основной памяти, как и буфера, при каждом запросе на запись позволяет постоянно поддерживать информацию, находящуюся в основной памяти, в обновленном состоянии.

Поэтому, когда возникает запрос на запись по адресу, относящемуся к области, содержимое которой не находится в данный момент в быстром буфере, новая информация записывается просто на место блока, которое предполагается переслать в ос-

новную память (без необходимости пересылки этого слова в основную память), так как в основной памяти уже находится его достоверная копия.

Данный алгоритм несложен для реализации и понимания, поэтому он был широко распространен в системах с кэш-памятью. Данный алгоритм также просто позволяет реализовать когерентность данных для мультипроцессорных систем с раздельными кэшами и общей памятью.

Алгоритм простого свопинга. Обращения к основной памяти имеют место в тех случаях, когда в быстром буфере не обнаруживается нужное слово. Эта схема свопинга повышает производительность системы памяти, так как в ней обращения к основной памяти не происходят при каждом запросе на запись, что имеет место при использовании алгоритма сквозной записи. Однако в связи с тем, что содержимое основной памяти не поддерживается в постоянно обновленном состоянии, если необходимого слова в быстром буфере не обнаруживается, из буфера в основную память надо возвратить какое-либо устаревшее слово, чтобы освободить место для нового необходимого слова. Поэтому из буфера в основную память сначала пересылается какое-то слово, место которого занимает в буфере нужное слово. Таким образом, происходит две пересылки между быстрым буфером и основной памятью.

Алгоритм свопинга с флагами. Данный алгоритм является улучшением алгоритма простого свопинга. В алгоритме простого свопинга, когда в кэш-памяти не обнаруживается нужное слово, происходит два обращения к основной памяти — запись удаляемого значения из кэша и чтение нового значения в кэш. Если слово с того момента, как оно попало в буфер из основной памяти, не подвергалось изменениям, т. е. по его адресу не производилась запись (оно использовалось только для чтения), то нет необходимости пересылать его обратно в основную память, потому что в ней и так имеется достоверная его копия; это обстоятельство позволяет в ряде случаев обойтись без обращений к основной памяти. Если, однако, слово подвергалось изменениям с тех пор, когда его копия была в последний раз записана обратно в основную память, то приходится перемещать его в основную память. Отслеживать изменения слова можно, пометив слово (блок) дополнительным флаг-битом. Варьируя значение флаг-бита при изменении слова, можно сформировать

информацию о состоянии слова. Пересылать в основную память необходимо лишь те слова, флаги которых оказываются в установленном состоянии.

Алгоритм регистрового свопинга с флагами. Повышение эффективности алгоритма свопинга с флагами возможно за счет уменьшения эффективного времени цикла, что можно получить при введении регистра (регистров) временного хранения между кэш-памятью и основной памятью. Теперь, если данные должны быть переданы из быстрого буфера в основную память, они сначала пересылаются в регистр (регистры) временного хранения; новое слово сразу же пересылается в буфер из основной памяти, а уже потом слово, временно хранившееся в регистре, записывается в основную память. Действия в ЦП начинают опять выполняться, как только для этого возникает возможность. Алгоритм обеспечивает совмещение операций записи в основную память с обычными операциями над буфером, что обеспечивает еще большее повышение производительности.

Сохранение когерентности (согласованности) кэш-памяти в многопроцессорных системах. Распространенными протоколами поддержания когерентности кэш-памяти являются следующие.

MESI (Modified, Exclusive, Shared, Invalid) — протокол поддержания когерентности кэш-памяти. Процессор различает кэш-строки, которых нет в кэшах других процессоров (Exclusive) и присутствующие более чем в одном кэше (Shared). Если изменяется Shared-строка, то другим ЦП компьютера передается сигнал проверить свои кэши и сделать напротив своих записей в кэшах пометку «неправильно» (Invalid). Если изменяется Exclusive-строка, то в этом нет необходимости. В любом случае строка помечается как Modified, но попыток немедленно записать в оперативную память эту строку не предпринимается.

MOESI (Modified, Owner, Exclusive, Shared, Invalid). Здесь Modified, Exclusive, Shared, Invalid соответствуют протоколу MESI. Owner соответствует измененной строке, содержащей данные, которые еще не записаны в оперативную память и которые есть в кэшах других процессоров. В то время как в MESI процессоры почти не используют кэши других ЦП, в протоколе MOESI любая операция чтения сопровождается проверкой кэшей других ЦП — если нужные данные находятся в одном из них, то читаются прямо оттуда; причем сохранение этих данных в ОП не производится, а в кэш-памяти делается пометка «Owner».

4.2. Конкретные системы памяти

Рассмотрим вкратце основные типы систем памяти.

В большинстве случаев основная оперативная память — RAM (Random Access Memory, т. е. память с произвольным доступом) ПЭВМ строится на микросхемах динамического типа (DRAM — Dynamic Random Access Memory), где в качестве запоминающего элемента (ЗЭ) используется простейшая сборка, состоящая из транзистора и конденсатора. Основными причинами широкого применения этой памяти является высокая плотность интеграции (увеличение числа ЗЭ на чип и сокращение числа чипов, необходимых для одного модуля), малое потребление энергии (тратится минимум энергии на хранение одного бита, уменьшается потребляемая системой мощность, снижается стоимость) и т. д.

Каждый бит такой памяти представляется в виде наличия (или отсутствия) заряда на конденсаторе, образованном в структуре полупроводникового кристалла. Конденсатор управляет транзистором. Если транзистор открыт и ток идет, это соответствует «1», если закрыт — «0». С течением времени конденсатор разряжается, и его заряд нужно периодически восстанавливать.

Между периодами доступа к памяти посылается электрический ток, обновляющий заряд на конденсаторах для поддержания целостности данных (вот почему данный тип памяти называется динамическим ОП). Этот процесс называется регенерацией памяти.

Имеется другой вид памяти, который лишен этого недостатка. Эта память называется статической (Static RAM — SRAM), где в качестве ЗЭ используется так называемый статический триггер, который может хранить данные, пока питание подается на схему. Это отличает ее от динамической оперативной памяти, которая должна регенерироваться с высокой частотой.

SRAM изготовляется по технологии, подобной процессорной, — фотогравирование кремния. Каждый бит SRAM требует от четырех до шести транзисторов, чем и объясняется то обстоятельство, что SRAM занимает намного больше места по сравнению с DRAM, которая требует только один транзистор (плюс конденсатор) на разряд.

Следовательно, если бы SRAM устанавливалась в качестве оперативной памяти, это привело бы к увеличению быстродей-

ствия ПК, однако при этом существенно увеличилась бы его стоимость, поскольку стоимость микросхемы SRAM значительно выше стоимости DRAM.

Каждая передача данных между процессором и памятью называется циклом шины. Количество бит, которое процессор может передать за один цикл шины, влияет на производительность компьютера и определяет, какой тип памяти требуется.

Для описания характеристик быстродействия оперативной памяти в пакетном режиме применяются так называемые циклы чтения/записи (или временна́я схема пакета). Эти числа относятся к количеству тактов процессора для каждого доступа при чтении. Дело в том, что при обращении к памяти на считывание или запись первого машинного слова расходуется больше тактов, чем на обращение к трем последующим словам. Так, для асинхронной SRAM (обеспечивает быстродействие от 12 до 20 нс при частоте шины ЦП от 50 до 66 МГц) чтение одного слова выполняется за 3 такта, запись — за 4 такта, чтение нескольких слов определяется последовательностью 3-2-2-2 такта (что означает, что чтение 1-го элемента данных занимает 3 такта ЦП, включая 2 такта ожидания, а чтение последующих — по 2 временных такта), а запись — 4-3-3-3.

Динамическая память

Динамическая память (DRAM) в современных ПК используется обычно в качестве оперативной памяти общего назначения, а также как память для видеоадаптера.

Принципы построения. Микросхема памяти этого типа представляет собой прямоугольный массив ячеек со вспомогательными логическими схемами, которые используются для чтения или записи данных, а также включают цепи регенерации, которые поддерживают целостность данных. Массивы памяти организованы в *строки (row)* и *столбцы (column)* ячеек памяти, именуемые соответственно линиями *слов (wordlines)* и линиями *бит (bitlines)*. Каждая ячейка памяти имеет уникальный адрес, задаваемый пересечением строки и столбца. Цепи, поддерживающие работу памяти, включают:

- усилители, считывающие сигнал, обнаруженный в ячейке памяти;
- схемы адресации для выбора строк и столбцов;

- схемы выбора адреса строки (Row address select — /RAS) и столбца (Column address select — /CAS), чтобы открывать и закрывать адреса строк и столбцов, а также начинать и заканчивать операции чтения и записи;
- цепи записи и чтения информации;
- внутренние счетчики или регистры, следящие за циклами регенерации данных;
- схемы разрешения выхода (Output enable — OE).

Интервал регенерации измеряется в наносекундах (нс), и это число отражает «скорость» ОП. Большинство ПК на основе процессоров Pentium используют скорость 60 или 70 нс. Процесс регенерации снижает скорость доступа к данным, поэтому доступ к DRAM обычно осуществляется через кэш-память. Однако когда быстродействие процессоров превысило 200 МГц, кэширование перестало существенно влиять на присущую DRAM низкую скорость и возникла необходимость использования других технологий ОП.

Цикл чтения включает следующие события (рис. 4.5, для EDO DRAM):

- выбор строки. Активизация цепи /RAS используется для связывания со строкой памяти и инициации цикла памяти. Это требуется при начале каждой операции с памятью. Активное состояние /RAS задается низким уровнем напряжения на линии, т. е. сигнал /RAS соответствует переходу от высокого напряжения в цепи к низкому. Сигнал /RAS может также использоваться для запуска цикла регенерации;
- выбор столбца. Сигнал /CAS используется для связывания со столбцом памяти и инициации операции записи-чтения. Активное состояние /CAS также задается низким напряжением на линии;
- разрешение записи (Write enable /WE). Сигнал /WE задает тип операции; высокий уровень напряжения определяет операцию записи, низкий — чтения информации;
- разрешение вывода (Output enable /OE). Во время операций чтения из памяти этот сигнал предотвращает появление данных прежде времени. Если уровень напряжения в цепи низкий, то данные передаются на выходные линии, как только это будет возможным. При записи в память эта линия игнорируется;
- ввод-вывод данных. Выводы DQ (также именуемые входо-выходными или I/Os) на чипе памяти предназначены

для ввода и вывода. Во время операции записи высокое («1») или низкое («0») напряжение подается на DQ. При чтении данные считываются из выбранной ячейки и передаются на DQ, если доступ осуществлен и /OE открыт. Все остальное время DQ находятся в закрытом состоянии (высокое входное сопротивление) — они не потребляют электрический ток и не выдают сигналов.

Рассмотрим некоторые конструкции систем динамической оперативной памяти.

FPM DRAM (Fast page mode DRAM) представляет собой стандартный тип памяти, быстродействие которой составляет 60 или 70 нс. Система управления памятью в процессе считывания активирует адреса строк, столбцов, осуществляет проверку данных и передачу информации в систему. Столбцы после этого деактивируются, что приводит к нежелательному состоянию ожидания процессора в некоторых сочетаниях операций с памятью. В наилучшем случае данный режим реализует временную схему пакета вида 5-3-3-3.

EDO RAM (RAM с расширенным выходом). Обращение на чтение осуществляется таким же образом, как и в FPM, за исключением того, что высокий уровень /CAS не сбрасывает выходные данные, а использование триггера позволяет сохранять данные то тех пор, пока уровень CAS снова не станет низким. Тем самым не происходит сброса адреса столбцов перед началом следующей операции с памятью.

Упрощенная схема работы EDO показана на рис. 4.5. Выходная величина поддерживается последовательностью стробирующих импульсов до тех пор, пока она не будет считана CPU, что

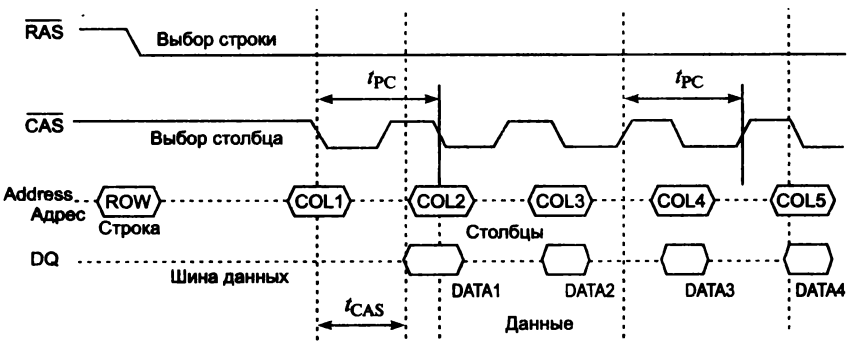


Рис. 4.5. Временна́я диаграмма EDO DRAM

особенно важно для быстрых процессоров наподобие Pentium; эта память обеспечивает лучшие параметры для серии быстрых последовательных считываний, чем FPM RAM. Теоретически быстродействие такой памяти на 27 % выше, чем для FMP DRAM.

Данный вид памяти является модификацией типовой FPM RAM с небольшими отличиями во временной последовательности /CAS и выходных данных. EDO DRAM обеспечивает более частую выдачу выходных данных, чем стандартная DRAM. Наибольшая скорость EDO RAM в циклах процессора — это 5-2-2-2 для пакета чтения из четырех величин (байт/слово/двойное слово). Память выпускается в трех вариантах — 70, 60 и 50 нс. EDO RAM не может работать при частоте шины, превышающей 66 МГц, а этот предел уже достигнут.

BEDO RAM (Burst extended data out DRAM — пакетная с расширенным выходом), читает данные в виде пакета, что означает, что после получения адреса каждая из следующих трех единиц информации читается за один цикл таймера, а процессор считывает данные в виде пакета 5-1-1-1. Быстродействие системы на 100 % превосходит FPM и на 50 % — EDO DRAM.

SDRAM (Synchronous DRAM — синхронная динамическая память). Этот тип памяти использует то обстоятельство, что большинство обращений к памяти являются последовательными и спроектирован так, чтобы передать все биты пакета данных как можно быстрее (когда начинается передача пакета, все последующие биты поступают с интервалом 10 нс). SDRAM содержит в своем составе счетчик пакетов, который автоматически увеличивает адреса и обеспечивает быструю последовательную выборку. Контроллер памяти обеспечивает локализацию требуемого блока памяти с максимальной скоростью (рис. 4.6). Данная система памяти может превосходить по быстродействию EDO RAM на 18 %.

DDR SDRAM (SDRAM II). Традиционно в устройствах с синхронизацией данные передаются по фронту импульса синхронизации (clock tick). Так как сигнал генератора импульсов изменяется между «1» и «0», данные могут передаваться как по переднему фронту импульса (изменение с «0» на «1»), так и по заднему (с «1» на «0»). В системах DDR (Double Data Rate) SDRAM или SDRAM II передача данных осуществляется по обоим фронтам тактовых импульсов, этим достигается удвоение скорости передачи при той же тактовой частоте (табл. 4.1).

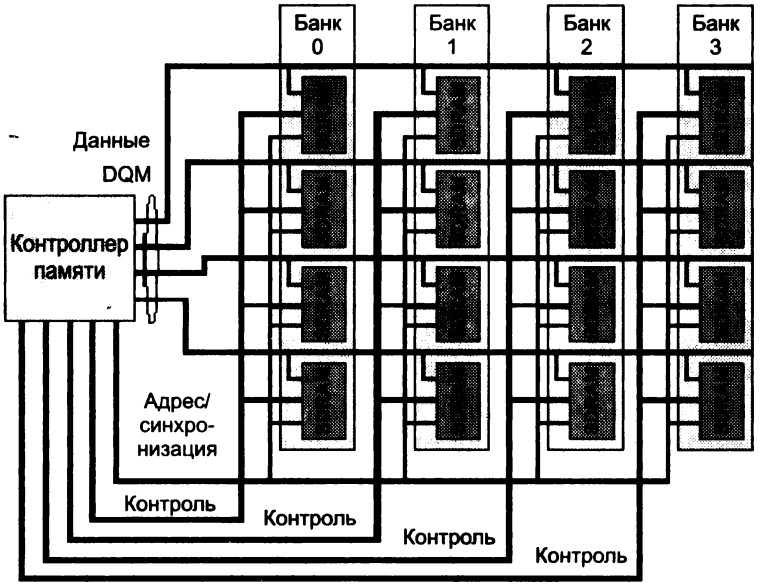


Рис. 4.6. Организация SDRAM

DDR2 SDRAM. К числу основных отличий технологии DDR2 от предыдущего варианта (DDR1) относится то, что в ней размер выборки данных увеличен вдвое — с 2 до 4 бит, а значит, во столько же раз возрастает и скорость передачи данных. Например, при 100 МГц она составит 400 Мбайт/с.

DDR3 SDRAM — синхронная память с произвольным доступом и удвоенной скоростью передачи данных (версия 3), является дальнейшим развитием DDR2 SDRAM.

Основное преимущество заключается в том, что частота шины ввода-вывода в 4 раза (табл. 4.1) выше быстродействия ячеек памяти, что повышает пиковую скорость передачи данных. Однако это достигается ценой возрастания задержек (латентности памяти). Стандарт DDR3 предусматривает емкость чипов от 512 Mib до 8 Gib, позволяя строить модули с максимальной емкостью 16 GiB.

Системы DDR3 должны на 30 % уменьшить потребление энергии за счет снижения напряжения питания до 1,5 В (1,8 В для DDR2 или 2,5 В для DDR). Это снижение сочетается с переходом на технологический процесс 90 нм, по которому изготов-

Таблица 4.1. **Чипы и модули DDR SDRAM, DDR2, DDR3**

Чипы памяти					Модули	
Тип чипа	Частота памяти, МГц	Цикл, нс	Частота шины, МГц	Число передач данных (млн в секунду — MT/s)	Название модуля	Пиковая скорость передачи данных, Мбайт/с
DDR (DDR1)						
DDR-200	100	10	100	200	PC-1600	1600
DDR-266	133	7,5	133	266	PC-2100	2133
DDR-333	166	6	166	333	PC-2700	2667
DDR-400	200	5	200	400	PC-3200	3200
DDR2						
DDR2-400	100	10	200	400	PC2-3200	3200
DDR2-533	133	7,5	266	533	PC2-4200	4264
DDR2-667	166	6	333	667	PC2-5300	5336
DDR2-800	200	5	400	800	PC2-6400	6400
DDR2-1066	266	3,7	533	1066	PC2-8500	8500
DDR3						
DDR3-800	100	10	400	800	PC3-6400	6400
DDR3-1066	133	7,5	533	1066	PC3-8500	8533
DDR3-1333	166	6	667	1333	PC3-10600	10 667
DDR3-1600	200	5	800	1600	PC3-12800	12 800

ляется большинство чипов DDR3. Некоторые производители предполагают в дальнейшем использовать транзисторы с двойным затвором, чтобы снизить токи утечки.

Более широкая полоса пропускания DDR3 обусловливается также использованием 8-разрядного буфера предвыборки (в DDR2 это 4, а в DDR — 2 бита).

Теоретически такие модули могут передавать данные со скоростью 800—1600 MT/s (миллионов передач данных в секунду), используя оба фронта импульсов тактового генератора шины ввода-вывода при частоте 400—800 МГц (DDR2 — 400—800 MT/s при 200—400 МГц шины или DDR — 200—400 MT/s при

100—200 МГц шины). Такая скорость передачи востребована в основном на современном рынке графических приложений.

Первые образцы систем были объявлены в 2005 г., а продукты появились на рынке в середине 2007 г. в форме системных плат на базе чипсета Intel P35 «Bearlake» и модулей памяти DIMM со скоростью DDR3 1600. Модули DDR3 DIMM имеют также 240 контактов, как и DDR2, и тот же размер, однако электрически несовместимы с ними, что поддерживается другим размещением ключа.

Преимущества DDR3 по сравнению с DDR2:

- более высокая полоса пропускания (до 1600 млн передач в секунду);
- повышена эффективность работы при малом энергопотреблении (более длительная работа батарей в ноутбуках);
- улучшен термический дизайн (кулер).

Недостатки по сравнению с DDR2:

- более высокая задержка сигнала CAS;
- общая стоимость выше, чем для эквивалентной памяти DDR2.

SLDRAM (Synchronous linked DRAM). Повышение производительности достигается за счет распространения пакетного протокола передачи данных на сигналы управления (отчего и пошло название этого типа памяти — Linked SDRAM). В SLDRAM адреса, команды, а также сигналы управления передаются в пакетном режиме по однонаправленной шине Command Link.

ESDRAM (Enhanced SDRAM — улучшенная SDRAM) — более быстрая версия SDRAM, сделанная в соответствии со стандартом JEDEC компанией Enhanced Memory Systems (EMS). С точки зрения времени доступа производительность ESDRAM в 2 раза выше по сравнению со стандартной SDRAM. В большинстве приложений ESDRAM, благодаря более быстрому времени доступа к массиву SDRAM и наличию кэша, обеспечивает даже большую производительность, чем DDR SDRAM.

CDRAM (Cached DRAM — DRAM с кэш-памятью) представляет собой улучшенный вариант ESDRAM. Cached DRAM имеет раздельные адресные линии для статического кэша и динамического ядра памяти. Необходимость управлять разнородными типами памяти усложняет контроллер, однако эффективность кэш-памяти, размещенной «внутри» микросхемы, выше, чем при традиционной архитектуре ПК, так как перенос в кэш

осуществляется блоками, в 8 раз большими, чем при выдаче «наружу» из микросхемы обычной DRAM.

Direct Rambus (DRDRAM). Обычная архитектура DRAM достигает своего практического потолка при частоте ЦП в 300 МГц.

DRDRAM — высокоскоростная динамическая память с произвольным доступом, разработанная Rambus Inc. Она обеспечивает более высокую пропускную способность по сравнению с большинством других DRAM. Direct Rambus DRAM представляет интегрированную на системном уровне технологию (рис. 4.7).

Подсистема памяти Rambus состоит из следующих компонентов:

- основного контроллера (RMC — Rambus Memory Controller);
- канала (RC — Rambus Channel);
- разъема для модулей (RRC — Rambus RIMM Connector);
- модуля памяти (RIMM — Rambus In-line Memory Module);
- генератора дифференциальных импульсов (DRCG — Direct Rambus Clock Generator);
- микросхем памяти (RDRAM — Rambus DRAM).

MDRAM (Multibank DRAM, мультибанковая память) применяет принцип обращения к памяти с перекрытием для реализации кэша 2-го уровня в качестве более дешевой и быстродействующей альтернативы SRAM. Память расщепляется на небольшие блоки (по 256 Кбайт), и обращение осуществляется к двум различным банкам памяти за один цикл процессора. Данный тип памяти первоначально использовался в графических картах с чипсетами Tseng Labs ET6x00 и выпускался в MoSys.

PSRAM (или PSDRAM — Pseudostatic RAM, псевдостатическая память) — система динамической памяти со встроенными цепями регенерации и управления выборкой, которые делают ее похожей на статическую память (SRAM). Здесь комбинируется высокая плотность данных, характерная для DRAM, и простота использования SRAM. Некоторые компоненты DRAM имеют режим «авторегенерации», который эквивалентен режиму ожидания, при котором контроллер памяти не потребляет электроэнергии, но без разрушения данных.

1T DRAM (однотранзисторная динамическая память) представляет собой альтернативный подход к построению ячеек памяти, которые в этом случае не содержат встроенных конденсаторов, но используют «паразитные емкости» (корпус транзистора совместно с изолирующей подложкой образуют конденсатор),

возникающие в транзисторах, создаваемых по технологии SOI (Silicon on Insulator).

Несмотря на то, что здесь сохраняется необходимость регенерации данных, считывание данных не разрушает информацию за счёт эффекта «плавающих зарядов» (floating body effect), которые запирают транзисторы.

1T DRAM выпускается Innovative Silicon Inc. под торговой маркой Z-RAM (от «Zero capacitor RAM»).

TTRAM (Twin Transistor RAM — двухтранзисторная динамическая память) — ячейка памяти аналогична обычной DRAM, состоящей из одного транзистора и одного конденсатора, однако роль конденсатора здесь играет второй транзистор (точнее, его паразитная емкость — между корпусом и кремниевой основой). Поскольку транзистор, созданный по технологии SOI значительно меньше по размеру, чем конденсатор, TTRAM может обеспечить большую плотность хранения данных, нежели DRAM. Однако необходимость выпуска таких чипов на производственных линиях КНИ (SOI), которые являются доминирующими в настоящее время, делает затруднительными прогнозы о стоимости модулей памяти данной технологии. Заметим, что описанная выше Z-RAM аналогична TTRAM, хотя использует только один транзистор и, следовательно, может обеспечить даже более высокую плотность, чем TTRAM.

VCM (Virtual Channel Memory) — технология, позволяющая оптимизировать доступ к оперативной памяти нескольких процессов (запись данных центральным процессором, перенос содержимого оперативной памяти на жесткий диск, обращения графического процессора и т. п.) таким образом, что переключение между процессами не приводит к падению производительности. В отличие от традиционной схемы, когда все процессы делят одну и ту же шину ввода-вывода, в технологии VCM каждый из них использует «виртуальную» шину.

При работе обычной памяти Memory Master (любое активное системное устройство, которому понадобился доступ к системной памяти, — контроллер PCI или AGP, кэш процессора L2, видеокарта и т. п.) выдает запрос, обладающий уникальными характеристиками, — адресом, размером блока данных и т. д.

При наличии нескольких устройств, одновременно выполняющих запросы в разные области памяти (причем, доступ в один момент времени может иметь только одно из них), о большой эффективности работы говорить не приходится.

В соответствии с технологией VCM в VC SDRAM активное устройство (memory master) может сделать запрос посредством виртуальных каналов.

По этой технологии при записи данные не сразу заносятся в память, а помещаются в буфер — виртуальный канал — и хранятся там до тех пор, пока память не будет готова их принять (она, например, может быть занята регенерацией или обменом с другим устройством) — рис. 4.7.

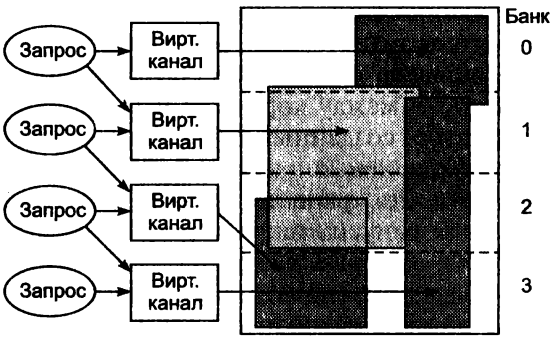


Рис. 4.7. Организация доступа к VCM

Active Link — разработка NEC, которая использует в DRAM архивацию (сжатие информации — примерно в 4 раза). Чтобы не загружать этой работой процессор, функции компрессии/декомпрессии данных возлагаются на микросхемы DRAM. В результате несколько расширилось обрамление кристалла, но получен двойной выигрыш — необходима меньшая по количеству ячеек микросхема DRAM, а также доступ к информации происходит быстрее, чем обычно.

IRAM (Intellectual Random Access Memory). Главная идея технологии IRAM заключается в размещении процессора и DRAM в одном чипе. Это дает возможность считывания и записи данных длинными словами (в пределах 128—16384 бит), обеспечивая высокую пропускную способность памяти. Раньше это было невозможно — все упиралось в неприемлемо большое число выводов микросхемы. Средняя скорость RAC/CAS равна приблизительно 10—30 нс для модулей емкостью 64—256 Мбайт IRAM.

Магнитная оперативная память. Следует отметить, что первые образцы ОП были построены на магнитных ферритовых сердечниках, которые пронизывали адресные и информацион-

ные шины (провода). Емкость таких ЗУ обычно не превосходила 64 Кбайт. В последующем в течение длительного периода времени устройства ОП выполнялись на кремниевых полупроводниковых элементах.

В 2000 г. IBM и немецкая фирма по производству полупроводников Infineon Technologies AG объявили программу разработки MRAM (Magnetic Random Access Memory). Принцип организации элементов памяти — магнитная среда, заключенная между металлическими пленками, образующими линии записи и чтения данных (рис. 4.8).

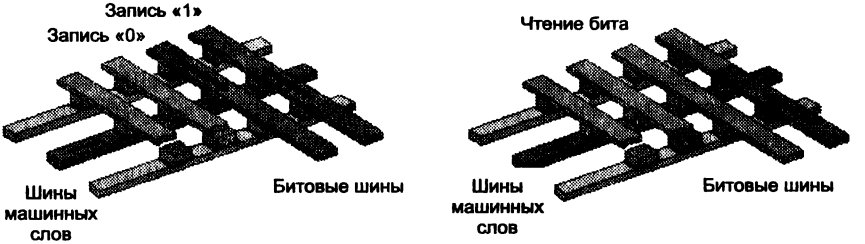


Рис. 4.8. Принцип функционирования MRAM:
а — запись данных; *б* — считывание

Преимущества технологии — высокая емкость и скорость, низкая стоимость, возможность применения как в форме статической, так и динамической памяти, более низкое энергопотребление.

Статическая память

Статическая память (SRAM) обычно применяется в качестве кэш-памяти второго уровня (L2) для кэширования основного объема ОП. Статическая память выполняется обычно на основе ТТЛ-, КМОП- или БиКМОП-микросхем и по способу доступа к данным может быть как *асинхронной*, так и *синхронной*. Асинхронным называется доступ к данным, который можно осуществлять в произвольный момент времени. Асинхронная SRAM применялась на материнских платах для третьего — пятого поколений процессоров. Время доступа к ячейкам такой памяти составляло от 15 нс (33 МГц) до 8 нс (66 МГц).

Синхронная память обеспечивает доступ к данным не в произвольные моменты времени, а одновременно (синхронно) с тактовыми импульсами. В промежутках между ними память может готовить для доступа следующую порцию данных. В большинстве материнских плат пятого поколения используется разновидность синхронной памяти — синхронно-конвейерная SRAM (Pipelined Burst SRAM), для которой типичное время одиночной операции чтения/записи составляет 3 такта, а групповая операция занимает 3-1-1-1 такта при первом обращении и 1-1-1-1 при последующих обращениях, чем и обеспечивается увеличение скорости доступа более чем на 25 %.

Принципы построения. *К*аждый бит в памяти системы SRAM хранится на четырех транзисторах (рис. 4.9), которые образуют два инвертора (Q и $\overline{Q}$) с перекрестными связями (или статический триггер, см. рис. 1.12). Ячейка поэтому имеет 2 стабильных состояния, обозначаемых как «1» и «0». Два дополнительных транзистора доступа контролируют операции записи и чтения. Таким образом, необходимо 6 транзисторов (MOSFET) для записи бита памяти.

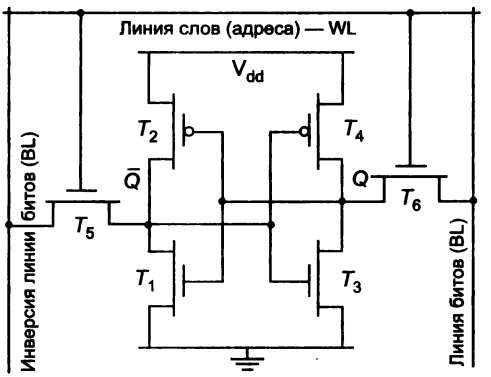


Рис. 4.9. Схема ячейки памяти SRAM

Доступ к ячейке разрешается подачей сигнала по «линии слов» (адреса, WL), которая контролирует два транзистора управления доступом (T_5 и T_6), которые в свою очередь определяют, должна ли ячейка подсоединиться к линиям битов (данных) — прямой (BL) и инвертированной ($\overline{BL}$), которые используются для передачи данных как при записи, так и при чтении. Хотя можно было бы ограничиться и одной битовой линией, по-

дача прямого и инвертированного сигналов повышает помехозащищенность.

Размер памяти SRAM с m линиями адреса и n линиями данных составляет 2^m слов или $2^m \times n$ бит.

Ячейка SRAM может находиться в трех различных состояниях — ожидания (standby), когда цепи свободны, чтения, когда данные запрошены, и записи, при обновлении содержимого:

- ожидание. Если доступа к слову не происходит, транзисторы доступа T_5 и T_6 отсоединяют ячейку от битовых линий. Триггер, состоящий из транзисторов T_1—T_4, сохраняет свое состояние;
- чтение. Предположим, что содержание памяти есть «1», хранящаяся в Q (т. е. выход Q равен «1»). Цикл чтения начинается с предустановки обеих битовых линий в логическую «1», а затем линия адреса WL получает доступ к обоим транзисторам T_5 и T_6. Следующий шаг состоит в том, что величины, хранящиеся в Q и $\overline{Q}$, передаются на битовые линии, оставляя BL в предустановленном состоянии и сбрасывая $\overline{\text{BL}}$ через M_1 и M_5 в логический «0». Со стороны BL транзисторы T_4 и T_6 устанавливают на линии битов напряжение V_{DD}, соответствующее логической «1». Если содержание ячейки было «0», осуществляются противоположные действия — $\overline{\text{BL}}$ устанавливается в «1», а BL — в «0»;
- запись. Цикл записи начинается с того, что записываемая величина подается на битовые линии. Если необходимо записать «0», то подается «0», устанавливая $\overline{\text{BL}}$ в состояние «1», а BL — в «0». Это аналогично подаче импульса на Т-триггер, что заставляет его перебрасываться в противоположное состояние. Запись «1» обеспечивается инвертированием значений на битовых линиях.

Статический тип памяти обладает более высоким быстродействием и используется, например, для организации кэш-памяти. Рассмотрим разновидности статической памяти.

Async SRAM (асинхронная статическая память). Это кэш-память, которая используется в течение многих лет с тех пор, как появился первый 386-й компьютер с кэш-памятью второго уровня. Обращение к ней осуществляется быстрее, чем к DRAM, и может, в зависимости от скорости процессора, использовать варианты с 20-, 15- или 10-нс доступом (чем меньше время обращения к данным, тем быстрее память и тем короче может быть пакетный доступ к ней). Тем не менее, как видно из названия, эта

память является недостаточно быстрой для синхронного доступа, что означает, что при обращении процессора все-таки требуется ожидание, хотя и меньшее, чем при использовании DRAM.

SyncBurst SRAM (Synchronous Burst Static RAM — синхронная пакетная статическая память). При частотах шины, не превышающих 66 МГц, синхронная пакетная SRAM является наиболее быстрой из существующих видов памяти. Причина этого в том, что если процессор работает на не слишком большой частоте, синхронная пакетная SRAM может обеспечить полностью синхронную выдачу данных, что означает отсутствие задержки при пакетном чтении процессором 2-1-1-1, т. е. синхронная пакетная SRAM выдает данные в пакетном цикле 2-1-1-1. Когда частота процессора становится больше 66 МГц, синхронная пакетная SRAM не справляется с нагрузкой и выдает данные пакетами по 3-2-2-2, что существенно медленнее, чем при использовании конвейерной пакетной SRAM. К недостаткам относится и то, что синхронная пакетная SRAM производится меньшим числом компаний и поэтому стоит дороже. Синхронная пакетная SRAM имеет время адрес/данные от 8,5 до 12 нс.

PB SRAM (Pipelined Burst Static RAM — конвейерная пакетная статическая память). Конвейер — это распараллеливание операций SRAM с использованием входных и выходных регистров.

Благодаря этому такая память является наиболее быстрой кэш-памятью для систем с производительностью шины более 75 МГц. PB SRAM может работать при частоте шины до 133 МГц. Она, кроме того, работает не намного медленнее, чем синхронная пакетная SRAM при использовании в медленных системах: она выдает данные все время пакетами по 3-1-1-1. Насколько высока производительность этой памяти, можно видеть по времени адрес/данные, которое составляет от 4,5 до 8 нс.

1-T SRAM. Как уже отмечалось ранее, традиционные конструкции SRAM используют статический триггер для запоминания одного разряда (ячейки). Для реализации одной такой схемы на плате должно быть размещено от 4 до 6 транзисторов (4-T, 6-T SRAM). Фирма Monolithic System Technology (MoSys) объявила о создании нового типа памяти, в которой каждый разряд реализован на одном транзисторе (1-T SRAM). Фактически здесь применяется технология DRAM, поскольку приходится осуществлять периодическую регенерацию памяти. Однако интерфейс с памятью выполнен в стандарте SRAM, при этом циклы регене-

рации скрыты от контроллера памяти. Схемы 1-T позволяют снизить размер кремниевого кристалла на 50—80 % по сравнению с аналогичными для традиционных SRAM, а потребление электроэнергии — на 75 %.

Реализация систем основной памяти

Модули памяти характеризуются такими параметрами, как объем (16, 32, 64, 128, 256 или 512 Мбайт), число микросхем, паспортная частота (100 или 133 МГц), время доступа к данным (нс) и число контактов (72, 168 или 184). В 2001 г. начался выпуск модулей памяти на 1—2 Гбайт.

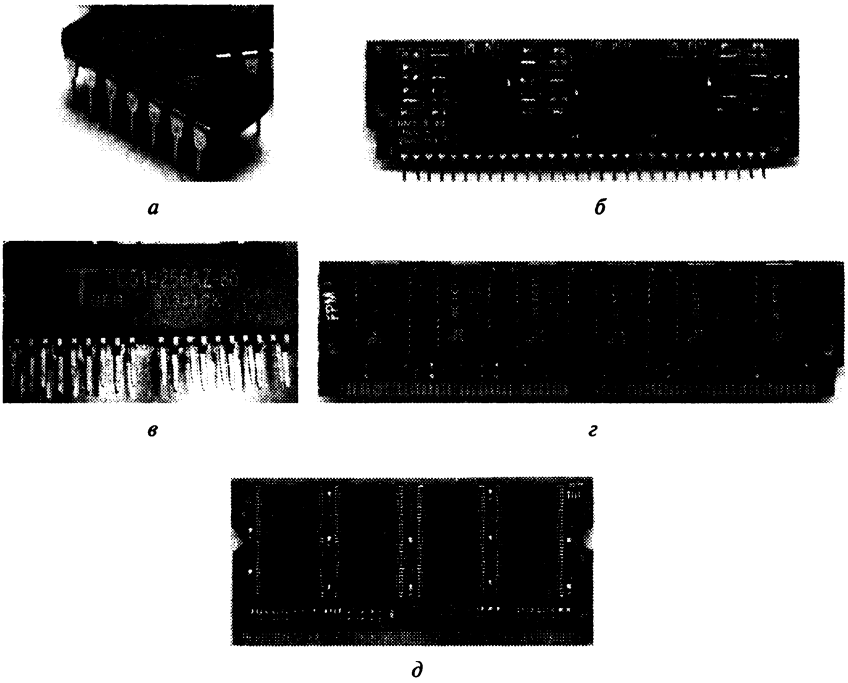
Модули DIP. Микросхемы DRAM упаковываются в так называемый DIP-корпус, при этом DIP обозначает Dual In-line Package (корпус с двухрядным расположением выводов). Этот термин относится к корпусам памяти, у которых выводы (pins) расположены по бокам (напоминают жука) — рис. 4.10, *а*. Сам кристалл, на котором размещены ячейки памяти, существенно меньше, чем корпус. Данная конструкция корпуса обусловлена такими требованиями, как удобство печатного монтажа и установки микросхемы в панельки на системной плате, а также соблюдение температурного режима работы элементов.

Большинство модулей DIP имеют интервалы между выводами в ряду 2,54 мм (0,1"), а расстояние между рядами — 7,62 мм (0,3" - «Skinny DIP», «Тощий DIP») или 15,24 мм (0,6"). Типичное число контактов равно 8 или любому другому четному числу от 14 до 24 (реже — 28) для корпусов на 0,3" и 24, 28, 32 или 40 (реже 36, 48 или 52) для корпусов на 0,6". На территории бывшего СССР используются аналогичные корпуса, но с размерами, выдержанными в метрической системе мер (например, интервал выводов 2,5 мм вместо 2,54 мм/0,1").

Известны различные варианты корпусов DIP, в основном различающиеся материалом изготовления:

- керамические (Ceramic Dual In-line Package — CERDIP);
- пластмассовые (Plastic Dual In-line Package — PDIP);
- пластмассовые уплотненные (Shrink Plastic Dual In-line Package — SPDIP) — уплотненная версия PDIP с интервалом выводов 1,778 мм (0,07).

Важнейшими параметрами микросхем DRAM являются емкость и организация памяти. Элементы DRAM в виде отдельных



а *б* *в* *г* *д*

Рис. 4.10. Внешний вид модулей памяти:
а — корпус DIP-14; *б* — модуль SIP; *в* — модуль ZIP; *г* — SIMM на 72 контакта; *д* — DDR SO-DIMM (PC2700, 200 контактов)

микросхем обычно устанавливались на старых материнских платах. В настоящее время эти микросхемы используются в качестве составных элементов модулей памяти, таких, как SIP-, ZIP- и SIMM-модули.

Информация о микросхеме в ее обозначении состоит, как правило, из нескольких полей. Первое поле содержит информацию о производителе и типе отбраковки при изготовлении микросхемы, следующее характеризует емкость, а дальнейшее — материал, из которого изготовлен корпус, и время доступа.

Например, для микросхем фирмы Mostek первые две буквы MK являются обозначением фирмы, MKB означает, что данная микросхема фирмы Mostek отбракована согласно военному стандарту (MIL STD-833), а микросхема MKI прошла отбраковку в соответствии с промышленным диапазоном температур. Цифра 4 говорит о том, что микросхема является элементом DRAM.

Следующая за ней цифра обозначает количество информационных разрядов: 1 — один разряд, 4 — четыре разряда. Группа цифр, следующая далее, обозначает количество информационных разрядов в килобитах (64 — 64 Кбит, 256 — 256 Кбит, 1000 — 1 Мбит). Далее буквой указывается тип корпуса (например, P — пластмассовый, хотя тип может быть и не указан). Через дефис указывается время доступа в наносекундах. Таким образом, по обозначению MKB44256-70 можно легко определить, что это микросхема фирмы Mostek, прошедшая отбраковку согласно военному стандарту, имеет емкость 4 разряда по 256 Кбит каждый и время доступа 70 нс.

SIP-модули. Микросхемы DRAM довольно легко и просто устанавливать в ПК, однако они занимают много места. С целью уменьшения размеров компонентов ПК, в том числе и элементов оперативной памяти, был разработан ряд конструктивных решений, приведших к тому, что каждый элемент памяти больше не устанавливался в отдельную панель, а совместимые элементы DRAM объединены в один модуль, выполненный на небольшой печатной плате.

Технология, реализующая такую конструкцию элементов памяти, называется SMT (Surface Mounting Technology), дословно переводимая как технология поверхностного монтажа.

В качестве реализации технологии SMT можно назвать так называемые SIP-модули с однорядным расположением выводов (Single In-line Package — SIP). SIP-модули представляют собой небольшую плату с установленными на ней совместимыми чипами DRAM (см. рис. 4.10, *б*). Такая плата имеет 30 выводов, размеры ее в длину около 8 см и в высоту около 1,7 см.

SIP-модули устанавливаются в соответствующие разъемы на системной плате. Однако при установке и извлечении таких модулей тонкие штырьки выводов часто обламываются, и контакт между штырьком и разъемом ненадежен. Это привело к дальнейшему развитию модулей памяти и появлению SIMM-модулей.

ZIP (zig-zag in-line package) — недолго просуществовавшая технология интегральных схем, в частности, чипов DRAM. Она была разработана для замены DIP. Интегральная схема ZIP заключается в пластиковый корпус, обычно размером 3 × 30 × 10 мм. Выводы устройства расположены в 2 ряда на одной из сторон корпуса. Эти ряды находятся на расстоянии 1,27 мм (0,05"), друг от друга в шахматном порядке, что дает возможность их более компактного

размещения, чем обычная прямоугольная решетка (рис. 4.10, *в*). Корпуса схем при этом могут располагаться на плате более плотно, нежели чем при схемотехнике DIP, при том же размере. ZIP были в дальнейшем вытеснены такими конфигурациями, как TSOP (thin small-outline packages), используемыми в SIMM (single-in-line memory modules) и DIMM (dual-in-line memory modules).

SIMM-модули. Когда речь идет о SIMM-модуле, имеют в виду плату, которая по своим размерам примерно соответствует SIP-модулю. Различие прежде всего, состоит в конструкции контактов. В отличие от SIP-модуля выводы для SIMM-модуля заменены так называемыми контактами типа PAD (вилка). Эти контакты выполнены печатным способом и находятся на одном краю платы, которым SIMM-модули устанавливаются в слоты на системной плате (рис. 4.10, *г*).

Кроме того, удобная конструкция SIMM-модулей позволяет пользователям самостоятельно менять и добавлять элементы памяти, не опасаясь повредить выводы.

В PC с CPU 80386 и ранних моделях с CPU 80486 использовались 30-контактные SIMM-модули памяти (DRAM), и число слотов на системной плате колебалось от 4 до 8. В настоящее время найти в продаже подобные модули весьма не просто. В более поздних моделях PC с CPU 80486 и Pentium стали использоваться 72-контактные SIMM-модули памяти (FPM DRAM).

DIMM-модули. В дальнейшем на многих системных платах появились слоты для 168-контактных модулей памяти DIMM (Dual In-line Memory Module). Модули DIMM обладают внутренней архитектурой, схожей с 72-контактными SIMM-модулями, но благодаря более широкой шине обеспечивают повышенную производительность подсистемы «CPU-RAM».

Для правильного позиционирования DIMM-модулей при установке в слоты на системной плате в их конструкции предусмотрены два ключа:

- первый ключ расположен между контактами *10* и *11* и служит для определения типа памяти модуля (FPM DRAM или SDRAM);
- второй ключ расположен между контактами *40* и *41* и служит для определения напряжения питания модуля (5 или 3,3 В).

RIMM. С появлением Direct RDRAM (DRDRAM) в 1999 г. появляются модули RIMM (название — не сокращение, а торго-

вая марка Rambus Inc). Разъемы RIMM имеют типоразмеры, подобные DIMM, и могут устанавливаться в пределах той же самой области системной платы, как и DIMM. Они имеют 184 штырька по сравнению с 168 для DIMM, но используют ту же спецификацию гнезда, как и стандарт DIMM на 100 МГц. BIOS ПК способен определить, какая оперативная память установлена, так что SDRAM-модули на 100 МГц должны работать в RIMM-совместимой системе.

SO-DIMM (Small Outline Dual In-Line Memory Module) представляет собой тип интегральных схем оперативной памяти компьютера. SO-DIMM являются малогабаритной альтернативой для DIMM и обычно занимают около половины пространства, требуемого для обычных модулей DIMM (рис. 4.10, *д*). В результате SO-DIMM в основном используются в таких устройствах как ноутбуки, небольшие настольные ПК (с платами типа Mini-ITX), высококачественные принтеры и сетевое оборудование (например, маршрутизаторы).

Модули SO-DIMM могут иметь 72, 100, 144 или 200 контактов, поддерживая передачу данных, соответственно по 32 бита (100) и 64 бита (144 и 200). Обычные DIMM имеют по 168, 184 или 240 и все поддерживают 64-битовую передачу данных.

Различные типы SO-DIMM распознаются по размещению «ключей» — модули на 100 контактов имеют два ключа, 144-контактный SO-DIMM имеет один ключ близко к центру корпуса, 200-контактный SO-DIMM — один ключ ближе к краю корпуса.

SO-DIMM примерно соответствуют (или меньше чем) по мощности DIMM, и обе технологии SO-DIMM и DIMM обеспечивают примерно равные скорости (тактовая частота, например, 400 МГц для PC3200 и латентность CAS величиной 2,0, 2,5 и 3,0) и емкость (512 Мбайт, 1 Гбайт и пр.). Более современные модули DDR2 SO-DIMM имеют частоту до 800 МГц PC6400 и предполагается, что достигнут частоты 1066 МГц PC8500.

FB-DIMM (Fully Buffered DIMM) — полностью буферизованный DIMM, технология, предназначенная для повышения надежности, быстродействия и емкости систем ОП. В обычных конструкциях ОП линии данных, идущие от контроллера памяти, соединяются со всеми DIMM-модулями. При возрастании электрической нагрузки (увеличение числа модулей или же разрядности памяти), а также с повышением частоты доступа проходящие сигналы начинают искажаться, что ограничивает эффективность системы в целом.

Архитектура Fully Buffered DIMM предусматривает промежуточный буфер (Advanced Memory Buffer — AMB), устанавливаемый между контроллером и модулем памяти (рис. 4.11). В отличие от параллельной шинной архитектуры для традиционных DRAM, FB-DIMM имеет последовательный интерфейс между контроллером и AMB. Это позволяет повысить разрядность памяти без увеличения количества линий контроллера памяти.

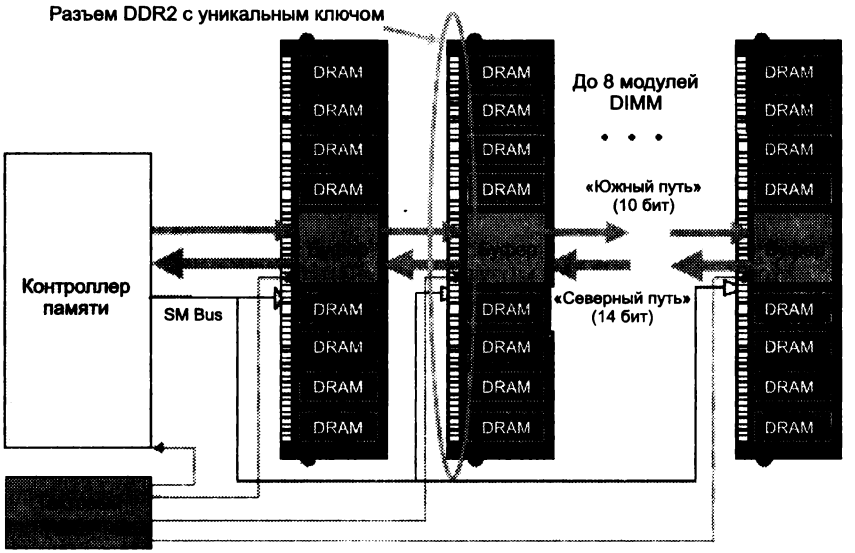


Рис. 4.11. Архитектура памяти FB-DIMM

Контроллер не передает сигнал непосредственно на модуль памяти, а действует через буфер, который восстанавливает форму сигнала и передает его дальше. Кроме того, AMB может осуществлять коррекцию ошибок, разгружая от этой функции процессор и контроллер памяти. Это сопровождается, однако, повышением латентности ОП.

Существует стандарт (протокол JESD82-20), определяющий интерфейс AMB с памятью DDR2. Канал FB-DIMM состоит из 14 битовых линий «Северного пути» («northbound»), по которым данные передаются из памяти на процессор, и 10 линий «Южного пути» («southbound»), предающих команды и данные из процессора.

Каждый бит передается на частоте, в 12 раз большей, чем базовая частота памяти (в 6 раз, если используется удвоенная скорость, DDR—DDR3). Например, для чипа DDR2-667 DRAM-канал будет работать на частоте 667 × 12/2 = 4000 МГц. Каждые 12 циклов образуют кадр: 168 бит «Северного пути» (144 бита данных, передаваемых 72-битовой DDR SDRAM плюс 24 бита для CRC-коррекции) и 120 бит «Южного» (98 полезных бит и 22 CRC-бита). Из 98 бит здесь 2 задают тип кадра, 24 — команда; в оставшихся битах могут содержаться (в зависимости от типа кадра) либо 72 бита записываемых данных, либо две или более 24-битовых команд, либо одна или более команда плюс 36 бит записываемых данных.

Поскольку записываемые данные подаются медленнее, чем это необходимо для ОП DDR, они накапливаются в AMB, а затем записываются в одном пакете (обычно по четыре кадра данных).

Команды соответствуют стандартным циклам доступа DRAM, например, выбор строки (/RAS), предвыборка, регенерация и пр. Команды чтения и записи содержат только адреса столбцов (/CAS) массива памяти. Все команды содержат трехразрядные адреса FB-DIMM, что позволяет подключать до 8 модулей FB-DIMM на 1 канал.

4.3. Внутренние интерфейсы

Связь устройств автоматизированных систем друг с другом осуществляется с помощью средств сопряжения, которые называются *интерфейсами*. Интерфейс представляет собой совокупность коммутаторов, линий, сигналов, электронных схем и алгоритмов (протоколов), предназначенную для осуществления обмена информацией между устройствами. С точки зрения обобщенной структуры ЭВМ и систем коммуникационные среды (см. рис. 2.2) состоят из процессорных устройств и коммутаторов, обеспечивающих соединения между ними.

Впервые реализованная в машинах IBM PC, IBM PC/XT и PC/AT концепция открытой архитектуры предполагает, что периферийные устройства связываются с ЦУ (процессор и ОП) посредством сменных карт расширения (или адаптеров), содержащих электронику, согласующую ЦУ и периферию —

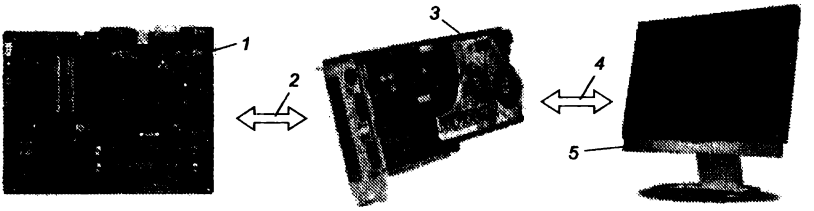


Рис. 4.12. Открытая архитектура IBM PC:
1 — системная плата (процессор, память, chipset); *2* — внутренний интерфейс (ISA, LPC, PCI, AGP, HyperTransport и пр.); *3* — плата расширения (адаптер, интерфейсная карта, контроллер внешнего устройства); *4* — интерфейс внешнего устройства (RS-232, Centronics, SCSI, USB, Firewire, инфракрасный, eSATA, Bluetooth и пр.); *5* — периферийное устройство (клавиатура, монитор, принтер, сканер и пр.)

рис. 4.12. Развитие или замена одних внешних устройств на другие в таких условиях сопровождается простой заменой карты.

Классификация интерфейсов

В соответствии с функциональным назначением интерфейсы можно разделить на следующие основные классы:

- системные интерфейсы ЭВМ;
- интерфейсы периферийного оборудования (общие и специализированные);
- программно-управляемых модульных систем и приборов;
- интерфейсы сетей передачи данных и др.

Мы предполагаем здесь рассмотреть *внутренние интерфейсы* (шины); *внешние интерфейсы* (порты) и *интерфейсы процессоров*.

Различные микросхемы и устройства, образующие компьютер, должны быть соединены друг с другом таким образом, чтобы они имели возможность обмениваться данными и были целенаправленно (системно) управляемыми. Эта проблема решается путем применения *унифицированных шин*. Используется набор проводников (на системной плате это печатные проводники), к которым подключены разъемы — *гнезда* (socket) или *слоты* (slot). В слоты расширения могут вставляться платы адаптеров (контроллеров) отдельных устройств и, что особенно важно, новых устройств. Таким образом, любой компонент, вставленный в слот, может взаимодействовать с каждым подключенным к шине компонентом ЭВМ.

Шина представляет собой набор проводников (линий), соединяющий различные компоненты компьютера для подвода к ним питания и обмена данными. В минимальной комплектации шина имеет три типа линий:

- линии (биты, разряды) управления;
- линии адреса;
- линии данных.

Обычно системы включают два типа шин:

- *системная шина*, соединяющая процессор с ОП и кэш-памятью 2-го уровня;
- множество *шин ввода-вывода*, связывающие процессор с различными периферийными устройствами. Последние соединяются с системной шиной *мостом*, который встроен в набор микросхем (chipset), обеспечивающий функционирование процессора (рис. 4.13).

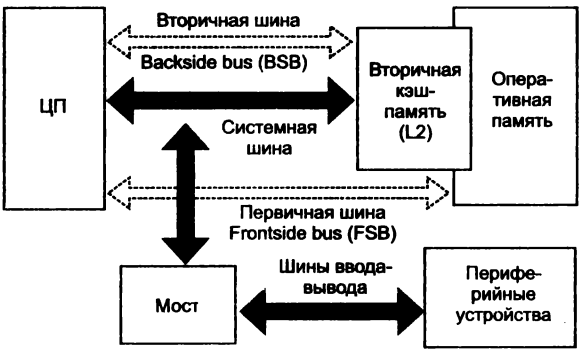


Рис. 4.13. Системные интерфейсы и интерфейсы ввода-вывода

Системная шина при архитектуре DIB (Dual independent bus) физически разделена на две:

- первичная шина (Frontside bus — FSB), связывающая процессор с ОП и ОП с периферийными устройствами;
- вторичная шина (Backside bus — BSB) для связи с кэш-памятью.

Использование двойной независимой шины повышает производительность за счет возможности для процессора параллельно обращаться к различным уровням памяти. Довольно часто термины «FSB» и «системная шина» используют как синонимы.

Следует отметить, что терминология, используемая в различных источниках для описания интерфейсов, не является вполне

однозначной и ясной. Системная шина часто упоминается как «главная шина», «шина процессора» или «локальная шина». Для шин ввода-вывода используются термины «шина расширения», «внешняя шина», «хост-шина» и опять же — «локальная шина».

Внутренние интерфейсы

Интерфейсы, характеристики которых приводятся в табл. 4.2, относятся к внутренним.

Таблица 4.2. **Основные характеристики внутренних интерфейсов**

Стандарт	Типичное применение	Пиковая пропускная способность	Примечания
ISA/ EISA	Звуковые карты, модемы Сети, адаптеры SCSI	2—8,33—33 Мбайт/с	Практически не используется, замещается PCI
PCI	Графические карты, адаптеры SCSI, звуковые карты новых поколений	133 Мбайт/с (32-битовая шина с частотой 33 МГц)	Стандарт для периферийных устройств
PCI-X	Универсальный интерфейс	1 Гбайт/с (64-битовая шина с частотой 133 МГц)	Расширение PCI, предложенное IBM, HP, Compaq. Увеличена скорость и количество устройств
PCI Express	Универсальный интерфейс	До 16 Гбайт/с	Разработка «интерфейса 3-го поколения» (Third generation Input/Output — 3GIO), может заменить AGP. Последовательная шина
AGP/AGP PRO	Графические карты, 3D-графика	528 Мбайт/с (2-x-mode) 800 Мбайт/с (4-x-mode)	Стандарт для Intel-PC начиная с Pentium II сосуществует с PCI
HT (Гипер-Транспорт)	Универсальный интерфейс	До 32 Гбайт/с	Разработка AMD для процессоров K7-K8

Шина ISA. ISA BUS (Industry Standard Architecture) — стандартные шины IBM XT (8 бит) и AT (16 бит).

Шина XT характеризуется следующими показателями (рис. 4.14, *а*):

- 8-битовая шина данных;
- 20-битовая шина адреса, что позволяет адресоваться к 2^{20} бит (1 Мбайт) памяти;

а

б

Рис. 4.14. Разъемы шин ISA (*а*), EISA (*б*)

- три канала прямого доступа к памяти (DMA);
- тактовая частота 8 МГц;
- пропускная способность 4 Мбайт/с;
- 62-контактный разъем.

Шина XT поддерживает централизованный метод арбитража, с этой целью в ней имеются общие линии запроса и ответа. Для обеспечения арбитража всем устройствам присваивается фиксированный уровень приоритета. В настоящее время XT практически не применяется.

Шина AT. В компьютерах IBM PC/AT шину расширили до 16 бит данных и 24 бит адреса. В таком виде она существует и поныне как самая распространенная шина для периферийных адаптеров. Параметры шины AT:

- 6-битовая шина данных;
- 24-битовая шина адреса, что позволяет адресовать 16 Мбайт памяти;
- 8 каналов прямого доступа (DMA);
- тактовая частота 8—16 МГц.

Максимальная скорость передачи данных составляет 8 Мбайт/с (8 МГц × 16 бит = 128 Мбит/с), 128 Мбит/с : 2 (передача данных требует от 2 до 8 тактов) = 64 Мбит/с = 8 Мбайт/с. Для шины ISA выпускаются два типа плат расширения — 16- и 8-разрядные.

Шина EISA (Extended Industry Standard Architecture). Шина EISA (рис. 4.14, *б*) явилась «асимметричным ответом» производителей клонов PC на попытку IBM поставить рынок под свой контроль путем выпуска MCA. Основные характеристики этой шины:

- 32-разрядная передача данных;
- максимальная пропускная способность — 33 Мбайт/с;
- 32-разрядная адресация памяти (позволяет адресовать до 4 Гбайт);
- поддержка многих активных устройств (bus master);
- возможность задания уровня двухуровневого (edge-triggered) прерывания (что позволяло нескольким устройствам использовать одно прерывание, как и в случае многоуровневого (level-triggered) прерывания);
- автонастройка плат расширения.

LPC. Шина Low Pin Count («малоконтактный» интерфейс), или LPC, используется на IBM-совместимых персональных компьютерах для подсоединения низкоскоростных устройств, таких, как «преемственные» (legacy) устройства ввода-вывода (последовательный и параллельный порты, клавиатура, мышь, контроллер НГМД). Физически LPC обычно подсоединяется к чипу «Южного моста». Шина LPC была предложена Intel в 1998 г. как замена для шины ISA (рис. 4.15).

Спецификация LPC определяет 7 электросигналов для двунаправленной передачи данных, 4 из которых несут мультиплексированные адрес и данные, оставшиеся 3 — управляющие сигналы (кадр, сброс, синхросигнал).

Шина LPC предусматривает только 4 линии вместо 8 или 16 для ISA, но она имеет полосу пропускания ISA (33 МГц). Другим преимуществом LPC является то, что количество контактов для присоединяемых устройств равно 30 вместо 72 для эквивалента ISA.

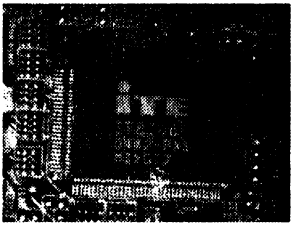


Рис. 4.15. Интерфейс Low Pin Count IT8705F

Шина PCI (Peripheral Component Interconnect bus). Разработка шины PCI закончилась в июне 1992 г. как внутренний проект корпорации Intel. Основные возможности шины следующие (рис. 4.16, 4.17):

- синхронный 32- или 64-разрядный обмен данными (64-разрядная шина в настоящее время используется только в Alpha-системах и серверах на базе процессоров Intel Xeon). При этом для уменьшения числа контактов (и стоимости) используется мультиплексирование, т. е. адрес и данные передаются по одним и тем же линиям;

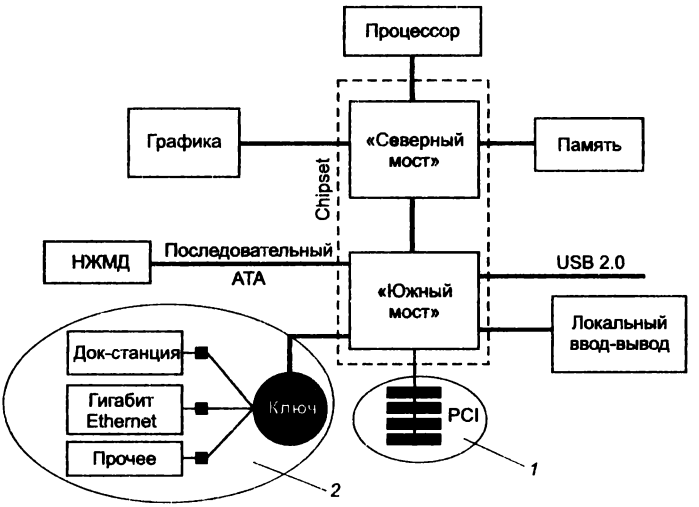
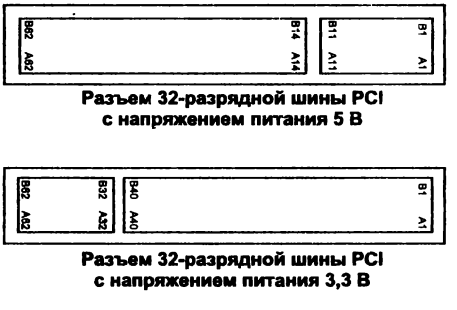
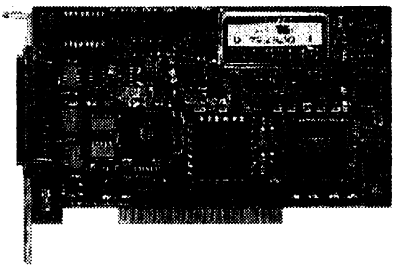


Рис. 4.16. Архитектуры шин PCI (*1*); PCI-E (*2*)





а *б*

Рис. 4.17. Разъемы шины PCI (*а*), типичное PCI-устройство (*б*)

- частота работы шины 33 или 66 МГц (в версии 2.1) позволяет обеспечить широкий диапазон пропускных способностей (с использованием пакетного режима) — 132 Мбайт/с (32-бит/33 МГц); 264 Мбайт/с (32-бит/66 МГц); 264 Мбайт/с (64-бит/33 МГц); 528 Мбайт/с (64-бит/66 МГц). При этом для работы шины на частоте 66 МГц необходимо, чтобы все периферийные устройства работали на этой частоте;
- полная поддержка многих активных устройств (например, несколько контроллеров жестких дисков могут одновременно работать на шине);
- спецификация шины позволяет комбинировать до восьми функций на одной карте (например, видео, звук и т. д.);

Известны также более поздние разновидности — PCI-X и PCI-Express, кроме того, к данному типу относится и PCMCIA — стандарт на шину для ноутбуков. Она позволяет подключать расширители памяти, модемы, контроллеры дисков и стримеров, SCSI-адаптеры, сетевые адаптеры и др.

PCI-X. PCI-X не только увеличивает скорость PCI-шины, но также и число высокоскоростных слотов. В обычной шине PCI-слоты работают на 33 МГц, а один слот может работать при 66 МГц. PCI-X удваивает производительность стандарта PCI, поддерживая один 64-битовый слот на частоте 133 МГц, а общую производительность увеличивает до 1 Гбайт/с. Новая спецификация также предлагает расширенный протокол для увеличения эффективности передачи данных и снизить требования к электропитанию.

PCI Express (PCI-E). Стандарт PCI-E определяет гибкий, масштабируемый, высокоскоростной, последовательный, «горячего подключения» интерфейс, программно-совместимый с PCI. В отличие от предшественника, PCI-E поддерживает систему связи «точка—точка», подобную ГиперТранспорту AMD, а не многоточечную схему, используемую в параллельной шинной архитектуре. Это устраняет потребность в шинном арбитраже, обеспечивает низкое время ожидания и упрощает «горячее» подключение-отключение системных устройств.

Ожидается, что одним из последствий этого будет сокращение площади платы на 50 %. Архитектура PCI Express обеспечивает полную полосу пропускания 16 Гбайт/с — Топология шины PCI-E содержит главный мост (Host Bridge) и несколько оконечных пунктов (устройств ввода-вывода). Многократные соединения «точка—точка» вводят новый элемент — переключа-

тель (ключ, switch) в топологию системы ввода-вывода (см. рис. 4.14, *б*).

Интерфейс PCI-E включает пары проводов — каналы (lane), и одна такая пара (PCI-E-lane) представляет собой интерфейс PCI-E 1x (800 Мбайт/с). Каналы могут быть соединены параллельно, и максимум 32 канала (PCI-E 32x) обеспечивают полную пропускную способность 16 Гбайт/с, достаточную, чтобы поддерживать требования систем связи в обозримом будущем.

Одним из направлений развития PCI-E является замена AGP (см. рис. 4.18). Действительно 8 Гбайт/с двунаправленной пропускной способности достаточно для поддержки телевидения высокого разрешения (HDT). Предполагается также, что PCI Express в конечном счете сможет заменить в чипсетах контроллер внешних устройств «southbridge». Это не повлияет на функции контроллера оперативной памяти «northbridge».

AGP (Accelerated graphics port). Несмотря на разрядность и скорость шины PCI, оставалась задача, которая выходила за пределы ее возможностей, — передача графической информации. Если адаптер CGA ($4 = 2^2$ цвета, экран 320×200 точек, частота 60 Гц) требует пропускную способность $2 \times 320 \times 200 \times 60 = 7\,680\,000$ бит/с = 960 Кбайт/с, то адаптер XGA (2^{16} цветов, экран 1024×768 пикселей, частота 75 Гц) требует $16 \times 1024 \times 758 \times 75 = 9\,433\,718\,400$ бит/с $\approx$ 118 Мбайт/с. В то же время пиковая пропускная способность PCI тогда составляла до 132 Мбайт/с.

Фирмой Intel было предложено решение в виде AGP — Accelerated graphics port (порт ускоренного графического вывода). Появление шины AGP в начале 1998 г. было своеобразным прорывом в области обработки графики. При частоте шины в 66 МГц она была способна передавать два блока данных за один такт. Пропускная способность шины составляет 500 Мбайт/с (V2.0) при двух режимах работы — DMA и Execute. На сегодняшний день существует стандарт AGP 4-х (поддерживаемый новыми чипсетами Intel и Via), позволяющий повысить пропускную способность до 1 Гбайт/с.

Схемы AGP взаимодействуют непосредственно с четырьмя источниками информации (Quadro port acceleration, рис. 4.18):

- процессором (кэш-память 2-го уровня);
- оперативной памятью;

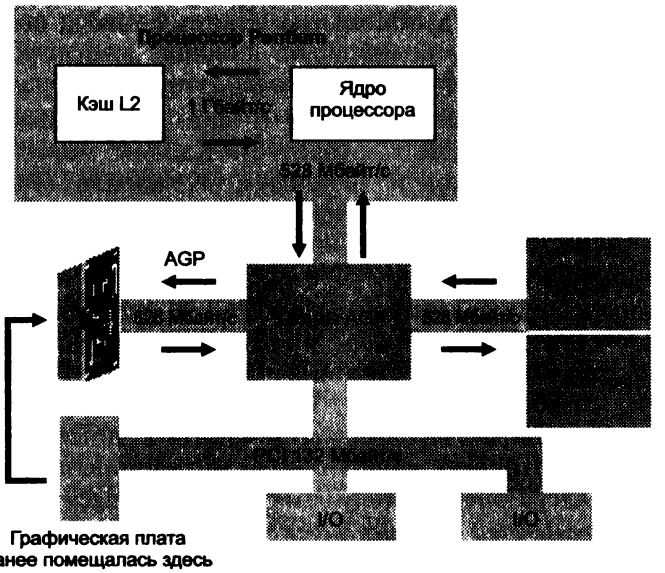


Рис. 4.18. Схема взаимодействия элементов с использованием AGP

- графической картой AGP;
- шиной PCI.

AGP функционирует на скорости процессорной шины (FSB). При тактовой частоте 66 МГц, например (в 2 раза выше, чем скорость PCI), достигается пиковая пропускная способность в 264 Мбайт/с. В графических картах, специально спроектированных для AGP, передача происходит как по переднему, так и по заднему фронту тактовых импульсов ЦП, что позволяет при частоте 133 МГц осуществлять передачу со скоростью до 528 Мбайт/с («2-х графика»). В дальнейшем была выпущена версия AGP 2.0, которая поддерживала «4-х графику» или четырехкратную передачу данных за один такт ЦП.

Контроллер HyperTransport. Фирмой AMD (процессор Hammer) была предложена архитектура ГиперТранспорт (HyperTransport), обеспечивающая внутреннее соединение процессоров и элементов чипсета для организации многопроцессорных систем.

Устройства, связываемые по шине ГиперТранспорт, соединяются по принципу «точка—точка» (peer-to-peer), что подразумевает возможность связывания в цепочку множества устройств без использования специализированных коммутаторов. Пропускная

способность шины ГиперТранспорт меняется от 200 Мбайт/с (частота 200 МГц и два 2-битовых канала) до 12,8 Гбайт/с (частота 800 МГц и два 32-битовых канала) — рис. 4.19.

Рисунок 4.20 демонстрирует, насколько разводка ГиперТранспорт экономичнее, чем у традиционных шин — достаточно сравнить площади, занимаемые на системной плате шиной AGP 8x с пропускной способностью 2 Гбайт/с и ГиперТранспорт (до 6,4 Гбайт/с).

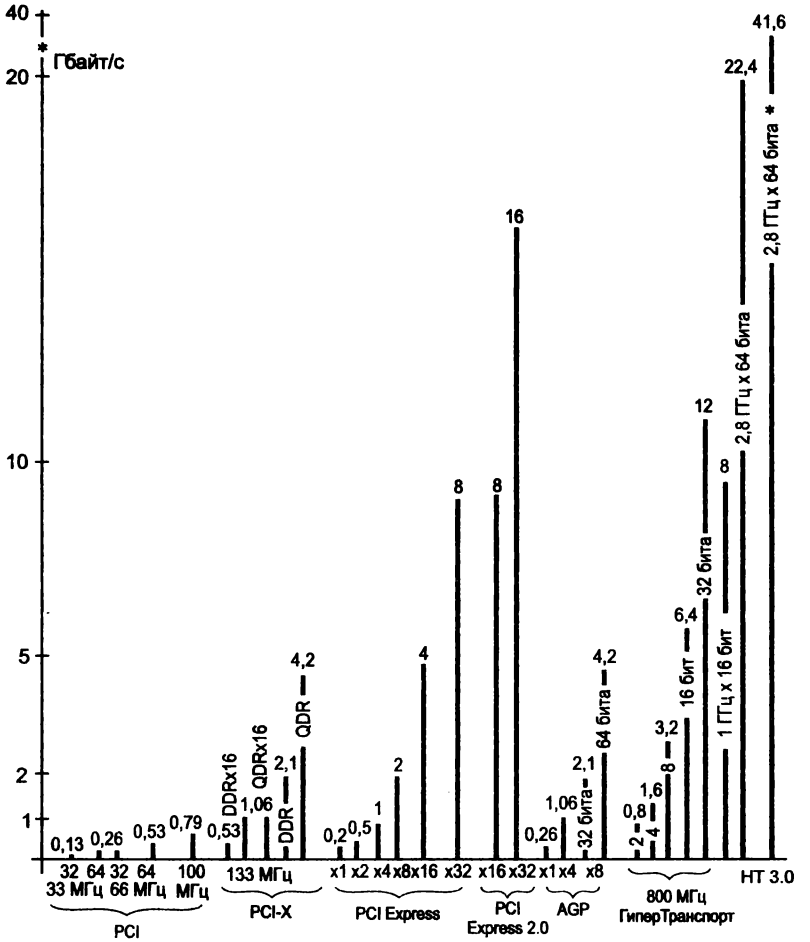


Рис. 4.19. Масштабируемость шин PCI, AGP и ГиперТранспорт

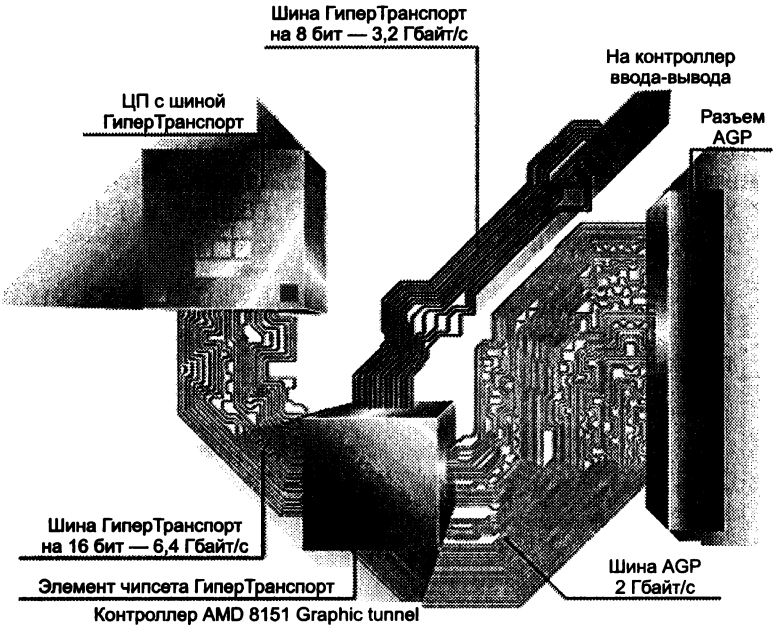


Рис. 4.20. Сравнительные физические размеры шин ГиперТранспорт и AGP

В таблице 4.3 приводятся основные характеристики различных поколений данного интерфейса.

Таблица 4.3. **Характеристики версий интерфейса HyperTransport**

Версия	Год выхода	Максимальная частота, МГц	Максимальная ширина линии, бит	Максимальная скорость передачи (два направления), Гбайт/с
1.0	2001	800	32	12,8
1.1	2002	800	32	12,8
2.0	2004	1400	32	22,4
3.0	2006	2600	32	41,6
3.1	2008	3200	32	51,2

Интерфейсы центральных процессоров

Способность ПК поддерживать множество различных интерфейсов, допускающих подключение различных классов добавочных компонентов, составляющих и периферийных устройств,

была одной из основных причин его быстрого успеха, ключом к которому была стандартизация.

Основа системы — процессор — не является исключением в этом смысле. В сущности ЦП — плоский квадратный слой кремния со схемами, выгравированными на его поверхности. Этот элемент укрепляется на основе — керамической или пластмассовой, — образуя пакет с контактами, выполненными или по плоской нижней стороне, или по одному из краев. Пакет ЦП связан с системной платой через разъем некоторой формы — гнездо (Socket) в первом и слот (Slot) во втором случае.

Socket 7, 8. Ранние процессоры — 386, 486, классический Pentium и Pentium MMX — представляли собой плоский квадратный пакет с массивом выводов-штырьков на нижней стороне, называемым матрицей выводов (Pin Grid Array, или PGA), который предполагал включение в гнездо на системной плате. Самым ранним таким интерфейсом, для которого было спроектировано много системных плат, работающих и по сей день (потому что этим поддерживались центральные процессоры различных изготовителей), является Socket 7. Первоначально разработанный Intel как преемник Socket 5, он имел тот же самый размер, но другие электрические характеристики. Socket 8 был разработан для ЦП Pentium PRO. Чтобы размещать кэш второго уровня в пакете (но не в ядре процессора), на плате пришлось устанавливать до трех отдельных схем, что оказалось чрезмерно дорогим для производства, которое и было довольно быстро прекращено.

Slot 1, 2. С введением ЦП Pentium II Intel переключилась к намного более дешевому решению упаковки чипов, которые состояли более чем из одного кристалла — SEC (Single edge contact cartridge) — см. рис. 3.16, *е*. На картридже SEC размещены шесть отдельных устройств: процессор, четыре модуля пакетной статической кэш-памяти второго уровня и один модуль дополнительной памяти. SEC-картридж имеет важные преимущества — разъем Pentuim Pro содержит 387 контактов, в то время как SEC — только 242. Это сокращение числа контактов на треть произошло вследствие того, что SEC-картридж содержит дополнительные терминаторы («заглушки»), которые обеспечивают разъединение сигналов, что приводит к уменьшению количества требуемых контактов напряжения питания.

Процессор Pentium II Xeon имел кэш-память второго уровня, работающую на полной тактовой частоте ЦП. Это требовало большего теплорассеяния, что в свою очередь привело к боль-

шей высоте картриджа. Решением был Slot 2, который также имел больше соединителей, чем Slot 1, что давало возможность поддерживать многопроцессорный протокол.

Super 7. Когда фирма Intel прекратила выпускать ЦП Pentium MMX в середине 1998 г., она фактически полностью оставила поле применения Socket 7 своим конкурентам, преимущественно AMD и Cyrix.

Разработанная AMD и ключевыми партнерами, архитектура платформы «Super 7» усилила возможности Socket 7, добавив поддержку для шинных интерфейсов с частотой 100 и 95 МГц, а также AGP-спецификацию и другие ведущие технологии, включая SDRAM на 100 МГц, USB, Ultra DMA (Direct memory access) и ACPI.

Процессор AMD K6-2 (запущенный в конце мая 1998 г.) обладал значительными архитектурными преимуществами, использовал производственную технологию 250 нм и соответствовал спецификациям Super 7.

Slot A. При выпуске ЦП Athlon в середине 1999 г. AMD также перешла от разъема «гнездо» к разъему «слот», в данном случае — Slot A (см. рис. 3.30, *в*). Физически идентичный разъему Slot 1, он использовал совершенно другой протокол (первоначально созданный DEC под наименованием EV6), который организует передачу между ОП и ЦП на частоте 200 МГц. Slot A содержал модуль регулятора напряжения (Voltage regulator module — VRM), возлагая на ЦП обязанность устанавливать правильное рабочее напряжение (в случае использования Slot A ЦП может работать в диапазоне между 1,3 и 2,05 В).

Socket 370. В 1999 г. Intel возвращается к архитектуре интерфейса «гнездо» с началом выпуска процессоров Pentium Celeron. Это квадратный пакет PGA, имеющий 370 контактов (Socket 370).

Внезапный отказ от Slot 1 в пользу Socket 370 вызвал потребность в адаптерах, чтобы можно было ЦП с интерфейсом PGA использовать в системных платах типа Slot 1. К счастью, промышленность сориентировалась и начала выпускать адаптеры (конвертеры) «Slot 1 — Socket 370», которые обеспечивали не только соответствующие соединения, но также и преобразование напряжения.

Socket A. Подобно Slot 1, Slot A фирмы AMD также недолго просуществовал. С появлением ядер процессора Athlon «Thunderbird» и «Spitfire», AMD также вернулась к пакетам стиля

PPGA для нового семейства процессоров Athlon и Duron. Они соединяются с системной платой через интерфейс AMD, названный Socket A. Он имеет 462 контакта, из которых 453 используются ЦП, и поддерживает как шину EV6 (200 МГц), так и ее модификацию на 266 МГц.

Socket 423, 478. С выпуском Pentium IV в конце 2000 г. Intel представила другой разъем, а именно Socket 423. Показательный для тенденции развития процессоров в сторону уменьшения потребляемой мощности, разъем PGA-стиля имеет эксплуатационный диапазон модуля регулировки напряжения (VRM) между 1,0 и 1,85 В.

Socket 423 использовался только несколько месяцев, когда Intel объявила о выпуске нового разъема Socket 478. Основное различие между ним и его предшественником — использование гораздо более плотного формата расположения контактов (μPGA — микроматрица выводов).

LGA775/Socket T. Интерфейсы LGA775 используют процессоры Pentium 4 (с ядрами Prescott и Cedar Mill), а также Pentium D (с ядрами Smithfield и Presler). В июле 2006 г. Intel выпустила версию для настольных ПК Core 2 Duo (Conroe), а позднее — Kentsfield Quad-Core ЦП, использующие LGA775. Интерфейс LGA775 обеспечивает лучшее охлаждение процессора, позволяя увеличить частоту FSB. Охлаждающий механизм теперь полностью закрепляется на системной плате.

Socket AM2. В мае 2006 г. AMD выпускает интерфейс процессора Socket AM2 (четвертое поколение архитектуры, начинавшейся от Hammer в 2003 г.), преемника более ранних Socket 754, 939 и 940. В то же самое время компания объявила, что это является переходом имеющихся двухъядерных AMD Athlon 64 X2 к новой платформе в дополнение к представлению нового ряда двухъядерных ЦП AMD Athlon 64 X2 на 5000+ и 4000+.

Основная инновация в ЦП, использующих AM2, заключается в размещении на чипе процессора непосредственно контроллера памяти (System Memory Controller Hub — MCH), что устраняет необходимость иметь отдельный Northbridge на системной плате. До сих пор при этом использовалась технология памяти DDR (SDRAM II, в которой передача данных осуществляется по обоим фронтам тактовых импульсов), которая к 2006 г. немного устарела, и новые ЦП на Socket AM2 используют контроллер памяти DDR-II, который работает в соответствующих скоростях DDR-II-667 и DDR-II-800.

В заключение рассмотрим некоторые последние разработки Intel и AMD.

Socket F (известен также как Socket 1207, LGA, 1207 контактов) — для мультипроцессорных систем AMD, использующих DDR2-SDRAM. Заменяет Socket 940 для процессоров Opteron и Athlon 64 FX.

Socket F+ — предназначен для AMD-серверов и двухпроцессорных изделий платформы Quad FX архитектуры K10, которые поддерживают интерфейс HyperTransport 3.0 с частотой от 2,6 ГГц. Предназначен для замены Socket F (процессоры Phenom FX Opteron 8300, серии 2300, Athlon 64 FX-70).

Socket AM2+ — будущий интерфейс AMD для однопроцессорных систем, поддерживает DDR2 и HyperTransport 3.0 с раздельными линиями питания. Заменяет Socket AM2 (PGA, 940 контактов, электрически совместим с Socket AM2).

Socket AM3 (PGA, 940 контактов) — перспективный интерфейс AMD для однопроцессорных систем, с поддержкой DDR3 и HyperTransport 3.0. Планируется к выпуску в 2008 г. для замены Socket AM2+ с поддержкой DDR3-SDRAM.

Socket S1 (PGA, 638 контактов) — разработка AMD для мобильных платформ с памятью DDR2-SDRAM. Заменяет 754 для мобильных процессоров.

Socket FM1, ***Socket FM2*** и ***Socket FM3*** — предназначены для будущих процессоров на основе архитектуры «Fusion».

Socket M — предназначен для Intel Core Solo, Intel Core Duo и Intel Core 2 Duo.

Socket P — разработка Intel, для замены Socket 479 и Socket M.

Socket B (LGA 1366) — новый интерфейс для будущих процессоров Intel, включающих контроллер памяти и Intel QuickPath Interconnect.

Socket H (LGA 715) — будущая замена для Socket T (LGA 775), без включения контроллера памяти и более новыми возможностями соединения процессоров по принципу «точка—точка».

Socket J (также известен как Socket 771 или LGA 771) — предназначен для ЦП Intel Xeon (ядро Woodcrest).

Socket N — для двухъядерного процессора Intel Dual-Core Xeon LV.

Таблица 4.4 характеризует все основные интерфейсы ЦП со времен Socket 1, первого разъема OverDrive-процессоров, используемого процессорами Intel 486 в начале 1990-х гг.

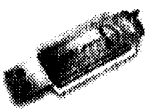
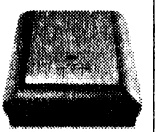
Таблица 4.4. **Характеристики некоторых интерфейсов процессоров**

Наименование	Число контактов	Описание
Socket 1/ Socket 8	169/387	Использовались в платах i486, Pentium OverDrive, Pentium MMX и Pentium Pro
Slot 1	242	Поддерживается кэш-память L1 до 512 Кбайт, состоящая из двух по 256 Кбайт. Использовался для ЦП Pentium II, Pentium III и Celeron
Slot 2	330	Аналогичен разъему Slot 1, однако может поддерживать до 2 Мбайт кэш-памяти, работающей на частоте ЦП. Использовался для Pentium II/III Xeon
Slot A	242	Разработан для AMD Athlon, механически совместим с разъемом Slot 1, но поддерживает совершенно другие электрические цепи
Socket 370	370	Заменяет Slot 1 для ЦП Celeron с 1999 г. Также используется для ЦП Pentium III Coppermine и Tualatin в вариантах, известных как FC-PGA и FC-PGA2 соответственно
Socket A/462	462	Разработан для процессора AMD Athlon (Thunderbird), который содержал на кристалле кэш-память L2
Socket 423	423	Введен для удовлетворения новых требований Pentium IV, который поддерживал новую системную шину (FSB). Включает теплорассеиватель
Socket 603	603	Предназначен для Pentium IV. Дополнительные контакты ориентированы на ЦП, которые будут содержать на кристалле кэш-память, а также для подключения других процессоров в мультипроцессорных системах
Socket 478	478	Разработан для поддержки технологий 0,13 мкм для ЦП Pentium IV Northwood в 2002 г. Используется технология µPGA (micro Pin Grid Array)
Socket 754	754	Введен AMD для 64-разрядных процессоров Athlon 64 осенью 2003 г.
Socket 940	940	Разработка AMD для ЦП Opteron и Athlon 64 FX. Для последнего был заменен позднее на Socket 939, который ориентирован на более дешевые системные платы
Socket 939	939	Выпущен AMD осенью 2004 г. для использования как в Athlon 64 (ранее — Socket 754), так и Athlon 64 FX (ранее — Socket 940)
LGA775/ Socket T	775	Land Grid Array 775 — введен Intel летом 2004 г. Используется для процессоров — Pentium 4, Celeron D, Pentium 4 Extreme Edition, Pentium D, Core 2 Duo
Socket 479	479	Также известен как mPGA479M и наиболее распространен как разъем для Pentium M (мобильный процессор)
Socket AM2	940	Выпущен AMD в 2006 г. для Athlon 64 X2 на 5000+ и 4000+
Socket F	1207	Также называемый Socket 1207 — мультипроцессорный интерфейс для процессоров Opteron и Athlon 64 FX
Socket S1	638	Разработка AMD для мобильных систем с использованием DDR2-SDRAM. Заменяет Socket 754 для мобильных ЦП

4.4. Интерфейсы периферийных устройств и внешние интерфейсы

Рассмотрим вкратце назначение и некоторые характеристики интерфейсов периферийных устройств (подробнее см., например, [10, 11, 15, 17]), перечень которых приводится в табл. 4.5.

Таблица 4.5. **Основные типы периферийных устройств и их интерфейсы**

Общий вид оборудования	Наименование, тип и назначение, интерфейс	Общий вид оборудования	Наименование, тип и назначение, интерфейс
	**Samsung SE-W164L** Внешний накопитель на CD/DVD, пишущий DVD±R 16x ***USB/SCSI***		**Nikon Coolpix S5** Цифровая фотокамера, разрешающая способность 6,0 мегапикселей, 3x zoom, вес 135 г ***USB, IEEE1394***
	**Plextor PX-760A** Встроенный накопитель на CD/DVD, пишущий DVD±R 18x ***IDE/ATAPI/SCSI***		**Linksys ADSL2MUE** Модем ADSL2, устройство цифровой связи по телефонной линии ***Последовательный порт, USB***
	**Hitachi Deskstar T7K250** Внутренний НЖМД («винчестер») для настольных ПК, форм-фактор — 3,5", емкость до 350 Гбайт ***IDE/PATA/SATA/SCSI***		**Samsung ML-1610** Лазерный принтер настольных ПК ***Параллельный порт, USB***
	**SanDisk Cruzer Micro** Флэш-накопитель, емкость 256 Мбайт — 2 Гбайта ***USB***		**HP Scanjet 4890** Настольный сканер для документов и пленок ***Параллельный интерфейс, USB***

Спецификации PC 98, PC 99, PC 2001

По инициативе корпораций Intel и Microsoft были разработаны рекомендации по конфигурированию ПК, их компонентам и характеристикам [2, 3].

В частности, в соответствии со спецификацией PC 98, подсоединяемые устройства (см. табл. 4.5) рекомендуется обозначать ярлыками, приведенными в табл. 4.6.

Таблица 4.6. **Обозначения для устройств и портов ПК, рекомендуемые спецификацией PC 98**

Символическое обозначение	Описание	Символическое обозначение	Описание	Символическое обозначение	Описание
	Шина расширения/докинг-станция		Аудиовыход		Сеть (Thicknet+ Twisted)
	Игровой порт (джойстик)		Микрофон		Телефонная линия
	Монитор		Наушники		Телефонный аппарат
	Клавиатура		Последовательный порт		Шина SCSI
	Мышь		Последовательный порт 1, 2...		Шина USB
	Аудиовход		Параллельный порт/ Принтер		Интерфейс IEEE 1394

Интерфейсы периферийных устройств

IDE (Integrated Device Electronics) — интерфейс устройств со встроенным контроллером (рис. 4.21). При создании этого интерфейса разработчики ориентировались на подключение

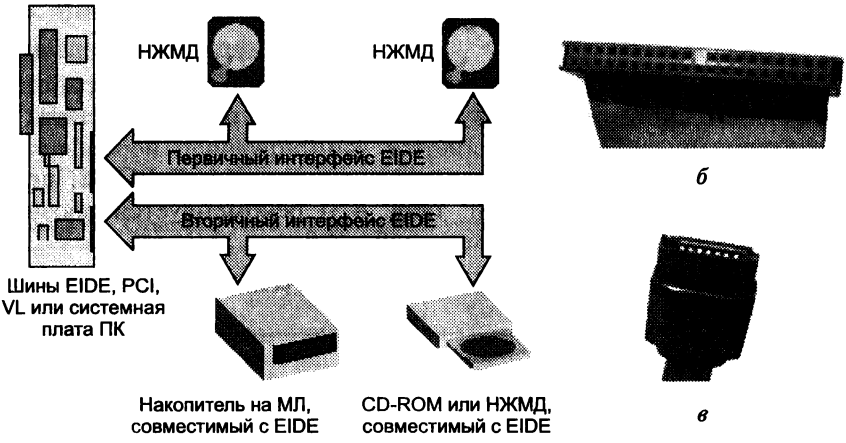


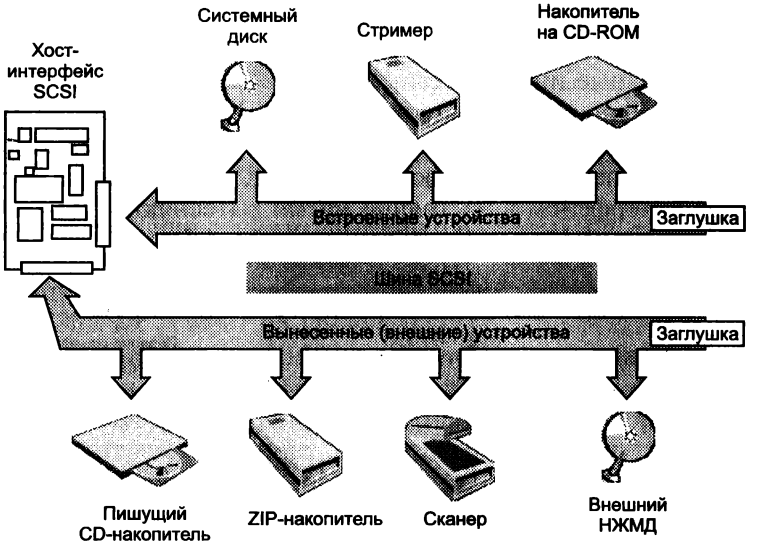
Рис. 4.21. Интерфейс IDE/EIDE:
а — общая архитектура; *б* — параллельный разъем ATA/IDE; *в* — последовательный разъем ATA

дисковых накопителей. За счет минимального удаления контроллера от диска существенно повышается быстродействие. В качестве синонима интерфейса IDE применяется термин ATA (AT Attachment).

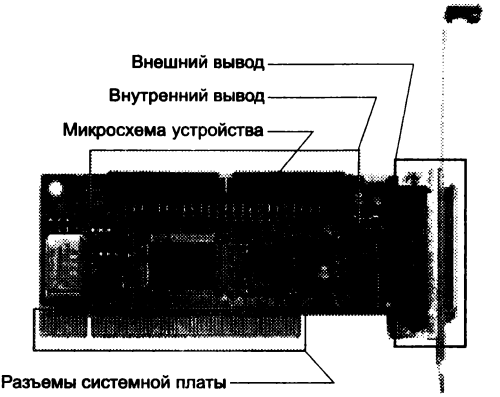
Физически интерфейс IDE реализован с помощью плоского 40-жильного кабеля, на котором размещаются разъемы для подключения одного или двух устройств. Общая длина кабеля не должна превышать 45 см, причем расстояние между разъемами должно быть не менее 15 см.

Интерфейс SCSI. SCSI (читается «скази») — один из промышленных стандартов для подключения периферийных устройств — винчестеров, стримеров, сменных жестких и магнитооптических дисков, сканеров, CD-ROM и CD-R, DVD-ROM и т. п. (рис. 4.22). К шине SCSI можно подключить до восьми устройств, включая основной контроллер SCSI (или хост-адаптер). Контроллер SCSI является по сути самостоятельным процессором и имеет свою собственную BIOS (которая иногда может размещаться в BIOS материнской платы).

Интерфейс SCSI является параллельным и физически представляет собой плоский кабель с 25-, 50-, 68-контактными разъемами для подключения периферийных устройств.



а



б

Рис. 4.22. Интерфейс SCSI:
а — подключение устройств; *б* — адаптер

Внешние интерфейсы

Принтеры, модемы и другое периферийное оборудование подключаются к компьютеру через стандартизированные интерфейсы, иногда называемые *портами*. В зависимости от способа

передачи информации (параллельного или последовательного) между сопрягаемыми устройствами различают параллельные и последовательные интерфейсы.

Распространенными являются интерфейс Centronics, обеспечивающий радиальное подключение широкого круга устройств с параллельной передачей информации, и RS-232. Данные традиционные интерфейсы в настоящее время вытесняются более быстродействующими — USB и FireWire (табл. 4.7).

Таблица 4.7. **Характеристики основных внешних интерфейсов**

Стандарт	Год выпуска	Первоначальная скорость, Мбит/с
Последовательный порт (RS-232)	1960	0,02
Параллельный порт (LPT)	1981	1,1
USB	1995	12
FireWire	1995	400
USB 2.0	2000	480
FireWire 800	2001	850

Последовательный порт стандарта RS-232-C. Обычно персональный компьютер оборудован хотя бы одним последовательным асинхронным адаптером (который расположен на системной плате либо оформлен в качестве сменной карты), по-другому называемым *последовательным портом RS-232-C.* Интерфейс RS-232-C является стандартом для соединения ЭВМ с различными последовательными внешними устройствами, в качестве которых первоначально выступали в основном терминалы и печатающие устройства. В операционных системах компьютеров IBM PC каждому порту RS-232-C присваивается логическое имя `COM1:—COM4:`.

Последовательная передача данных состоит в побитовой передаче каждого байта цифровой информации в форме *кадра данных,* содержащего сигнал начала передачи (Start), сигнал окончания передачи (Stop) и информационные биты (рис. 4.23).

Иногда используется контрольный бит P, которому присваивается такое значение, чтобы общее число единиц или нулей

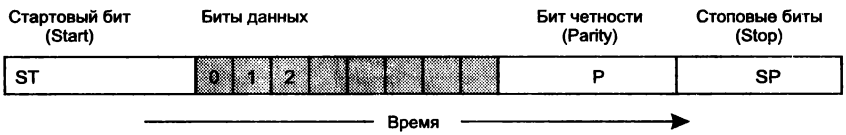


Рис. 4.23. Структура кадра данных при передаче байта информации в стандарте RS-232-C

было четным или нечетным. Бит (или биты) SP сигнализирует об окончании передачи байта.

Использование (или нет) битов P, ST, SP задает *формат передачи данных (кадра)* на уровне RS-232. Принимающее и передающее устройства должны применять одинаковые форматы.

Параллельный порт. Параллельный порт (Centronics) используется для одновременной передачи 8 битов информации. В компьютерах этот порт используется главным образом для подключения принтера, хотя это не исключает возможность подсоединения к нему других устройств, например графопостроителей или даже других ПЭВМ.

Конструктивно обычно оформлен в виде 25-контактного разъема типа D (DB25). Имеется восемь шин данных, для каждой шины данных своя линия заземления.

Параллельное соединение применяется на расстояниях не более 5 м, некоторые источники ограничивают расстояние 1—2 м; при увеличении длины параллельных проводов возрастает межпроводная емкость, что приводит к перекрестным помехам, кроме того, растут материальные затраты на реализацию линии.

Он располагается обычно на задней стенке компьютера как D-образная 25-контактная розетка. Там может также иметься D-образная 25-контактная вилка.

USB (Universal Serial Bus) — стандарт, разработанный совместно фирмами Compaq, DEC, Microsoft, IBM, Intel, NEC и Northern Telecom (версия первого утвержденного варианта появилась довольно давно — 15 января 1996 г.) и предназначенный для организации соединения многочисленных и разнотипных внешних устройств с помощью единого интерфейса (рис. 4.24).

Основная цель стандарта, поставленная перед его разработчиками, — создать реальную возможность пользователям работать в режиме Plug-n-Play с периферийными устройствами. Это означает, что должно быть предусмотрено подключение устройства к работающему компьютеру, автоматическое распознавание его немедленно после подключения и последующей установки

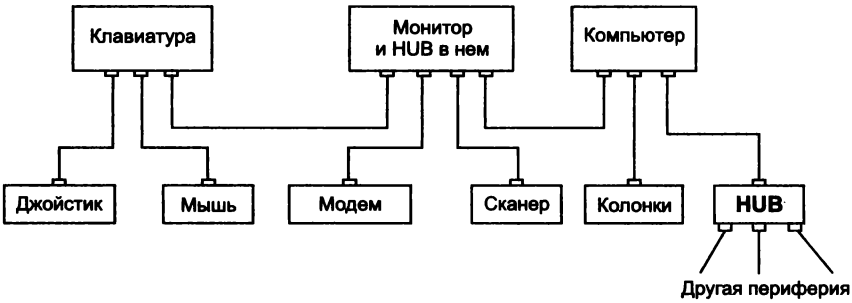


Рис. 4.24. Пример соединения периферийных устройств в USB-сеть

соответствующих драйверов. Кроме этого, желательно питание для маломощных устройств подавать через саму шину.

Стандарт USB позволяет подсоединить до 127 устройств последовательно или используя концентратор USB (hub), к которому подсоединяются семь устройств. Разъемы содержат четыре контакта, включая провода питания (5 В) для таких небольших устройств, как ручной сканер или звуковая колонка.

Интерфейс FireWire. Впервые разъемы IEEE 1394 были установлены на цифровых камерах DCR-VX1000 и DCR-VX700 (Sony). Сегодня любая DV-камера оснащается интерфейсом IEEE 1394.

Интерфейс во многом подобен USB 1.0, но является более быстродействующим (в различных спецификациях устанавливается быстродействие от 12,5 Мбит/с до 1,6 Гбит/с и более). Это создает возможность для соединения интерфейсом FireWire ЭВМ с такими устройствами, как аналоговые и цифровые видеокамеры, телевизоры, принтеры, сетевые карты и накопители информации (рис. 4.25).

Интерфейс FireWire поддерживает синхронную и асинхронную передачу данных и предоставляет возможность подключения до 63 устройств на один порт. При этом поддерживается скорость передачи 100, 200 и 400 Мбит/с (т. е. 12,5, 25, 50 Мбайт/с), прорабатываются варианты на 800 и 1600 Мбит/с. При этом различные пары устройств могут обмениваться данными на различной скорости, например на 100 и 400 Мбит/с. Для связи используется шестижильный медный кабель или оптоволокно. Из этих шести проводов два идут к источнику питания, а четыре других, организованных как две экранированные витые пары, используются для передачи данных. Кабель в целом также

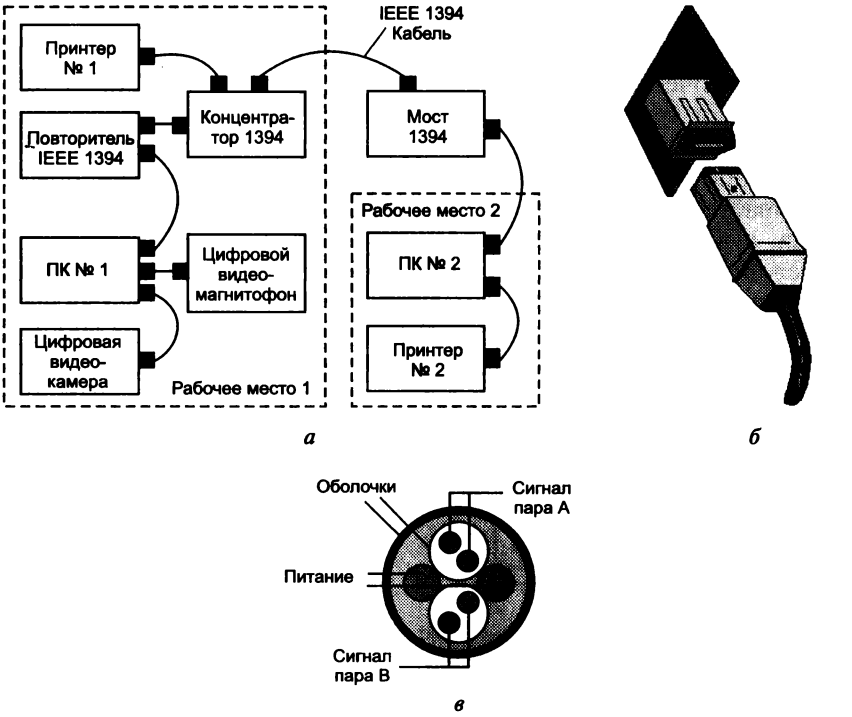


Рис. 4.25. Соединение разнотипных устройств в сеть с использованием IEEE 1394 (*а*); разъем (*б*) и кабель (*в*)

экранирован. По проводам питания может подаваться напряжение от 8 до 40 В (ток до 1,5 А), это позволяет отказаться от источников питания в периферийных устройствах.

4.5. Архитектуры набора микросхем системной платы (чипсет)

Chipset, или «PCIset», — совокупность микросхем, размещенных на системной плате, которые организуют потоки команд и данных в ПЭВМ. Сюда входят основная память, вторичная кэш-память и устройства, связанные с шинами ISA и PCI. Кроме того, чипсет контролирует потоки данных НЖМД и других устройств, соединенных с каналом IDE. Иногда в состав чипсет включают и сам процессор.

Архитектура «Северный мост — Южный мост»

Классический чипсет состоит, как правило, из двух (реже — одной или трех) микросхем, в которых одна, поддерживающая контроллер памяти, AGP, и какую-либо вспомогательную шину (PCI либо любую другую) между микросхемами чипсета, обычно называется Northbridge. Соответственно, контроллеры ввода-вывода, а в последнее время и контроллер шины PCI, интегрируются в Southbridge. Northbridge и Southbridge соединяются той или иной шиной. Примером Northbridge — Southbridge является Triton 430 TX (рис. 2.24, *г*). На рис. 4.26 приведена структура чипсета AMD 750, который имеет аналогичную архитектуру и ориентирован на процессор Athlon. Набор микросхем состоит из двух устройств: системного контроллера AMD-751 и контроллера периферийных устройств AMD 756.

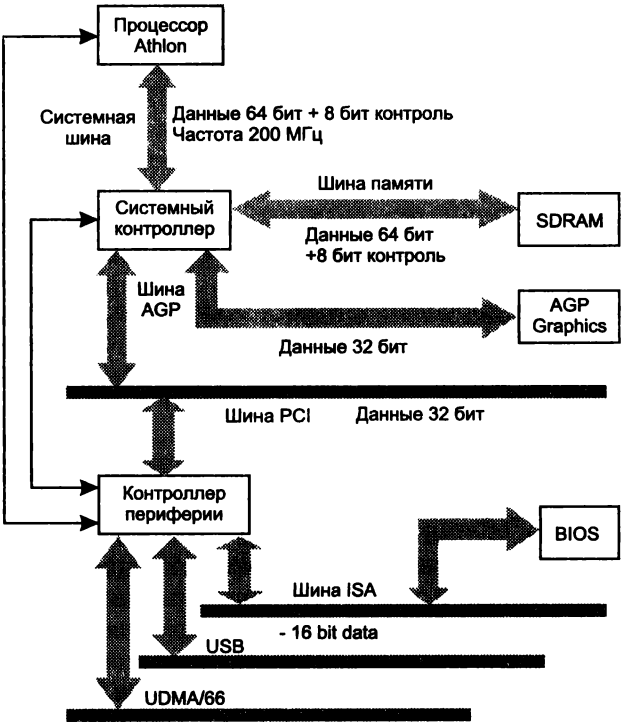


Рис. 4.26. Архитектура «Северный мост — Южный мост» (чипсет AMD 750)

Функции системного контроллера (Northbridge):
- поддержка шинного интерфейса с процессором на частоте 200 МГц;
- поддержка шины PCI 2.2 с подключением до шести устройств;
- поддержка до 768 Мбайт ОП SDRAM DIMM;
- совместимость со спецификациями AGP (1-х и 2-х графика).

Функции периферийного контроллера (Southbridge):
- поддержка устройств Plug-n-Play и стандартов управления питанием ACPI 1.0 и APM 1.2;
- поддержка контроллера клавиатуры и мыши;
- функции IDE-контроллера с поддержкой возможностей Ultra DMA-33/66;
- интегрированный контроллер шины ISA и мост ISA-PCI, удовлетворяющий спецификациям PC-97;
- контроллер USB и концентратор на четыре порта.

Архитектура AGPset

Предусматривает ускоренное взаимодействие с портом AGP (2-х графика) и предназначена для использования в серверах и рабочих станциях на процессорах Xeon. Выпущенный в одно время с процессором Pentium II Xeon в середине 1998 г., чипсет 440GX является развитием более раннего варианта 440BX.

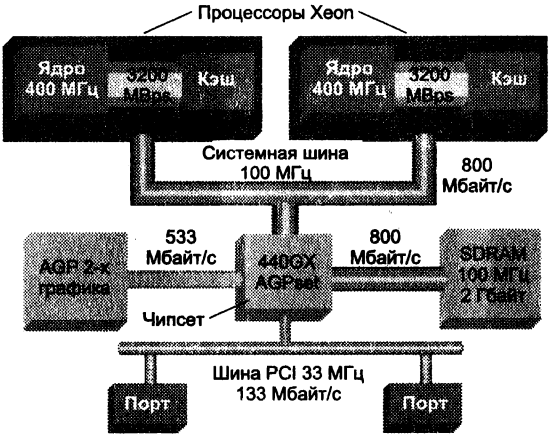


Рис. 4.27. Архитектура AGPSet (чипсет Intel Triton 440GX)

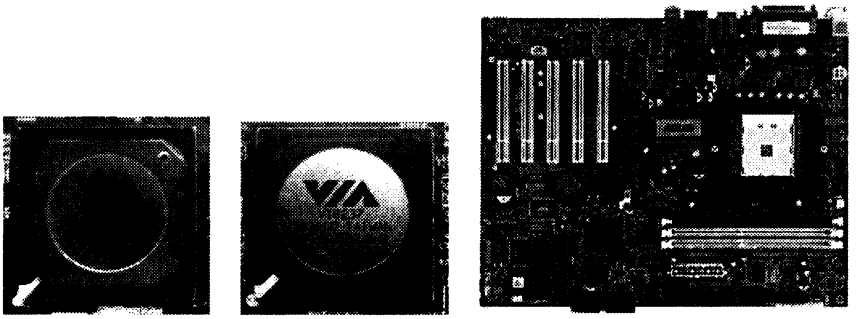
Чипсет поддерживает два типа разъемов для процессора — Slot 1 и Slot 2, слот расширения 2x AGP, подключение 2-х ЦП и максимум 2 Гбайта памяти (рис. 4.27). Поддерживается также работа вторичной шины процессора с полной частотой ЦП, что позволяет кэш-памяти 2-го уровня процессора Pentium II Xeon работать на частоте ЦП.

«Неоклассические» архитектуры для AMD K8

Как уже отмечалось ранее, в архитектуре K8 контроллер памяти интегрирован в ЦП и тем самым часть функций «Северного моста» воплощены в процессоре. Поэтому чипсету остается только реализовать контроллеры ввода-вывода, PCI и AGP.

VIA-чипсет для процессоров AMD (VIA K8T800) выполнен в почти классическом варианте (рис. 4.28, 4.29) — с «Северным мостом» (не загруженным взаимодействием с памятью) и «Южным», которые соединены шиной 8X V-Link с пропускной способностью 533 Мбайт/с. В качестве «Южного моста» используется чип VT8237, который поддерживает:

- восемь портов USB 2.0;
- два порта Parallel ATA133/100/66 с поддержкой до четырех устройств;
- звуковые решения от VIA — VIA Vinyl 5.1 & Vinyl Gold 7.1;
- два порта SATA с поддержкой RAID;



а *б* *в*

Рис. 4.28. Чипсет VIA K8T800:
а — «Северный мост» K8T800; *б* — «Южный мост» VT8237; *в* — системная плата ASUS K8V Deluxe с чипсетом VIA K8T800

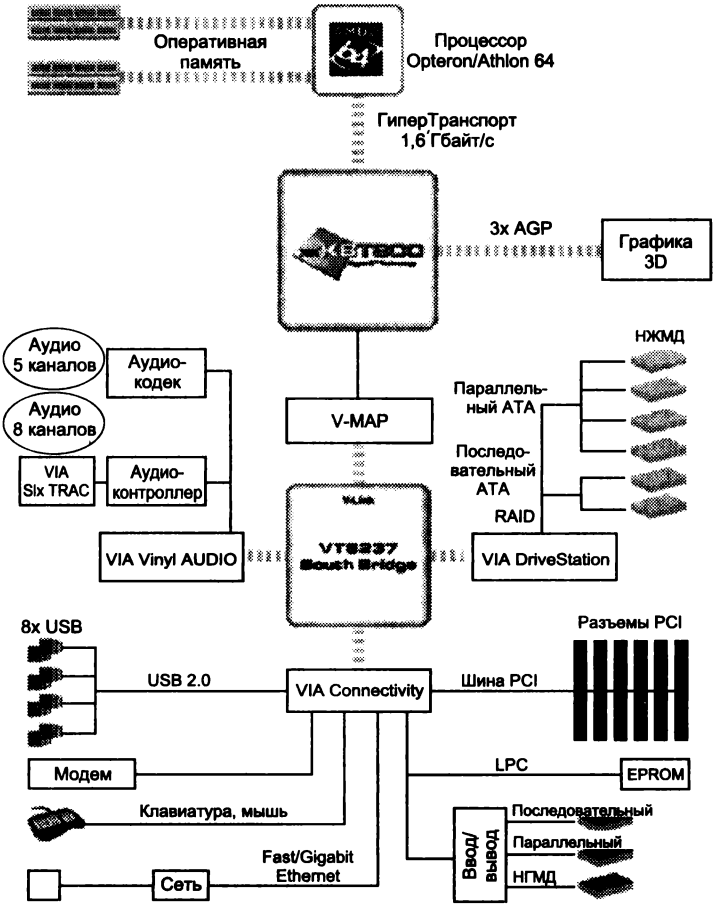
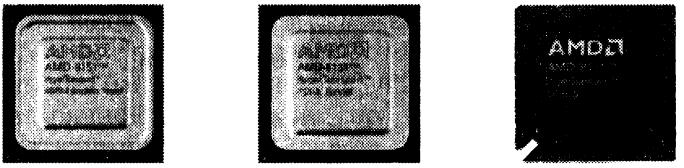


Рис. 4.29. Архитектура чипсета VIA K8T800

- интегрированный 10/100 BaseT сетевой контроллер;
- подключение контроллера Gigabit Ethernet.

Архитектура AMD-8000. AMD разработала набор из трех микросхем, называемых туннелями, под общим наименованием AMD-8000. С помощью комбинаций этих микросхем можно создавать как чипсеты для дешевых настольных моделей, так и для высокопроизводительных рабочих станций и даже для многопроцессорных систем. Набор микросхем AMD-8000 можно сравнить с набором кубиков, из которых можно создавать любые по сложности решения (рис. 4.30).



а *б* *в*

Рис. 4.30. Компоненты AMD-8000:
а — AMD-8151, HyperTransport-AGP tunnel; *б* — AMD-8131, HyperTransport-PCI-X контроллер; *в* — AMD-8111, HyperTransport (контроллер ввода-вывода, шины PCI, BIOS и все оставшиеся функции — IDE-контроллер, USB 2.0, сеть и пр.)

AMD-8151 Graphics Tunnel представляет собой контроллер AGP 8x с пропускной способностью 2,1 Гбайт/с. Кроме того, данный туннель имеет два встроенных порта HyperTransport — Link A (входной) и Link B (выходной). Первый интерфейс (Link A) является 16-битовым с полосой пропускания 6,4 Гбайт/с (по 3,2 Гбайт/с в каждом направлении). Второй (Link B) — уже 8-битовый с полосой пропускания 1,6 Гбайт/с (по 0,8 Гбайт/с в каждом направлении).

AMD-8131 I/O Bus Tunnel — контроллер шины, предназначенный для использования в серверных системах. Туннель обладает двумя интерфейсами HyperTransport (Link A и Link B), как и в AMD-8151. Сам контроллер PCI-X поддерживает две независимые PCI-X-шины с возможностью установки до пяти устройств на каждую.

AMD-8111 I/O Hub — контроллер ввода-вывода, без которого вообще невозможно построить систему — он должен обязательно присутствовать на плате. В минимально возможной конфигурации плата может содержать лишь один туннель AMD-8111 I/O Hub (при этом предполагается, что система будет без поддержки AGP). Туннель AMD-8111 I/O Hub обладает одним-единственным 8-битовым интерфейсом Hyper Transport с пропускной способностью 800 Мбайт/с и содержит все необходимые контроллеры ввода-вывода, как и классический «Южный мост» поддерживает до восьми устройств PCI (32 бит/33 МГц), шесть портов USB 2.0, два канала IDE с поддержкой ATA 33/66/100/133, сетевой адаптер 10/100 Мбит/с, модем AC'97, шестиканальное аудио и LPC-шину.

Архитектура AMD-8000 позволяет, кроме однопроцессорных систем, создавать многопроцессорные (рис. 4.31, 4.32).

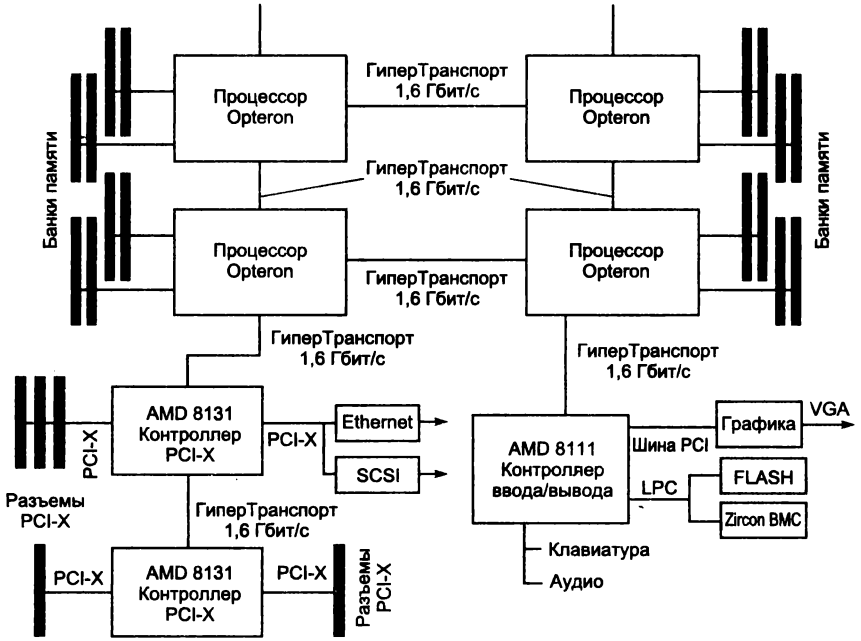


Рис. 4.31. Система на четырех процессорах Opteron с использованием AMD-8000 (архитектура «плоская решетка», см. рис. 2.41)

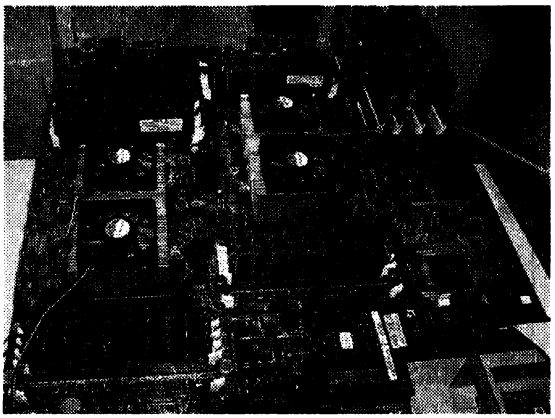


Рис. 4.32. Системная плата компьютера с четырьмя процессорами Opteron

В табл. 4.8 приводятся некоторые характеристики наборов микросхем Intel и Via.

Таблица 4.8. **Основные характеристики некоторых вариантов чипсетов**

Тип	Характеристики							
	Процессор	Частота системной шины, МГц	Модули память	Тип памяти	Графика	Порт IDE/ATA	USB	Звук
i865PE	Pentium IV	800/533/400	4 DIMMs	DDR 400/333/266	AGP 8x	2 порта ATA 150	8 портов USB 2.0	AC'97
i875P	Pentium IV	533/400	4 DIMMs	DDR 400/333/266	AGP 8x	2 порта ATA 150	8 портов USB 2.0	AC'97
i845PE	Pentium IV Celeron	533/400	2 двухсторонних DIMMs	DDR 333/266	AGP 4x	PATA	8 портов Hi-Speed USB 2.0	AC'97
AMD8000	AMD K8	HT 800 Мбайт/с			AGP 8x/PCI-X	ATA 33/133	6 портов USB 2.0	
Intel P965 Express	Core 2 Duo Pentium D Pentium 4 LGA775	1066/800	2 DIMM, 2 channels	DDR2 800/677/533	PCI Express x16 (1x16)	SATA (3 Гбит/с) External SATA	8 портов USB 2.0	Intel HDAudio, AC'97/20 бит
VIA K8T800 + VT8237	AMD Athlon 64, Socket 754	HT 1,6 Гбайт/с		3 DDR до 3 ГБ	AGP/ 5 PCI/ ASUS WiFi	2 порта PATA (ATA133), 2 порта SATA	2 разъема по 2 USB 2.0	VIA AudioVinyl 5.1
NVIDIA nForce3 Pro 150	AMD Opteron, Athlon 64 FX, Socket 940			4 DDR до 4 ГБ	AGP/ 5 PCI	2 канала PATA/IDE с поддержкой UltraATA 133 и RAID	4 USB 2.0 + 1 разъем на 2 USB 2.0	AC'97-кодек Avance Logic ALC650

Контрольные вопросы

1. Охарактеризуйте стратегии управления иерархической памятью.
2. Назовите две основные разновидности памяти компьютера.
3. Что представляет собой ОП? Каково его назначение?
4. В чем различие между статической и динамической памятью?
5. В чем суть принципа однородности памяти? Какие возможности он открывает?
6. В чем заключается принцип адресности?
7. Что такое «прямой» и «обратный» порядок байтов?
8. Опишите принципы открытой архитектуры IBM PC.
9. Что такое интерфейсы? Каковы их классы?
10. Перечислите внутренние интерфейсы и дайте их краткие характеристики.
11. Перечислите интерфейсы накопителей и дайте их краткие характеристики.
12. Перечислите интерфейсы внешних устройств и дайте их краткие характеристики.
13. Что такое локальные шины?
14. Охарактеризуйте разновидности архитектуры чипсета.
15. Что входит в состав чипсет?
16. Что такое архитектура Northbridge-Southbridge?

Заключение

Компьютеры в современном обществе взяли на себя значительную часть работ, связанных с информацией. По историческим меркам компьютерные технологии обработки информации еще очень молоды и находятся в самом начале своего развития, преобразуя или вытесняя традиционные технологии обработки информации.

ЭВМ характеризуются множеством показателей — система команд (общее количество выполняемых операций), быстродействие центрального процессора, емкости оперативной и внешней памяти, количество и номенклатура одновременно подсоединяемых периферийных устройств, потребляемая электроэнергия и др. На всех этапах истории развития ЭВМ создатели стремились все больше повысить их производительность, а пользователи мечтали (и мечтают) получить в свое распоряжение еще более мощные и функционально более богатые вычислительные средства. Появились разного типа вычислительные системы, разнообразные многопроцессорные архитектуры, в архитектуре стало применяться множество различных функциональных элементов и узлов (например, таких, как кэш-память) и т. д.

Если же рассмотреть историю развития информационных технологий, то параллельно с развитием ЭВМ успешно развивались существующие и появлялись новые ИТ. С появлением микропроцессоров в истории вычислительной техники началась эпоха микроЭВМ, а с развитием микроэлектронной технологии (по мере возрастания степени интеграции и расширения функциональных возможностей микроэлектронных схем) стали появляться новые микропроцессоры и на их базе более совершенные ЭВМ. В конечном счете этот период ознаменовался широчайшим распространением во всех сферах человеческой деятельности персональных ЭВМ (ПЭВМ) и развитием технологии автоформализации знаний.

Увеличение емкости систем памяти и повышение общей производительности ПК привели к более широкому распространению баз данных в различных автоматизированных системах управления. Эти и ряд других факторов привели к повышению ценности различных информационных ресурсов. Появилась необходимость предоставить доступ к этим ресурсам многим пользователям. Возникли локальные вычислительные сети (ЛВС), которые дали возможность в некоторой степени решить эту задачу, а также повысить эффективность использования дорогостоящих аппаратных средств (например, принтеров, дисков и др.).

Расширялись функциональные возможности новых ПК и одновременно появлялись новые задачи, требующие еще более мощных накопителей, процессоров и т. д. Объединение технологии сбора, хранения, передачи и обработки информации на компьютерах с техникой связи привели к идее создания более мощных и широких глобальных вычислительным сетей. Разрабатывались новые ИТ, ставящие свои специфические требования перед вычислительной техникой. Этот циклический процесс прогресса непрерывно продолжается.

Список литературы

1. Dictionary of Computing (Data Communication, Hardware and Software Basics, Digital Electronic) Ed. by Frank J. Galland, John Wiley & Sons, Datology Press Ltd, Windsor, England, 1983.

2. PC 2001 System Design Guide. A Technical Reference for Designing PCs and Peripherals for the Microsoft® Windows® Family of Operating Systems. 1999—2000 Intel Corporation and Microsoft Corporation.

3. PC 99 System Design Guide, Microsoft Press®, 1998.

4. *Алферова З. В.* Теория алгоритмов. М.: Статистика, 1973.

5. Информационные технологии: учеб. пособие / О. Л. Голицына, Н. В. Максимов, Т. Л. Партыка, И. И. Попов. М.: ФОРУМ: ИНФРА-М, 2006.

6. *Голицына О. Л., Попов И. И.* Основы алгоритмизации и программирования: учеб. пособие. 3-е изд. М.: ФОРУМ, 2008.

7. *Каган Б. М.* Электронные вычислительные машины и системы. М.: Энергоиздат, 1991.

8. *Каймин В. А.* Информатика: учебник. М.: ИНФРА-М, 2000.

9. Основы информатики (учеб. пособие для абитуриентов экономических вузов) / К. И. Курбаков, Т. Л. Партыка, И. И. Попов, В. П. Романов М.: Экзамен, 2004.

10. *Максимов Н. В., Партыка Т. Л., Попов И. И.* Архитектура ЭВМ и вычислительных систем. 4-е изд. М.: ФОРУМ, 2012.

11. *Максимов Н. В., Партыка Т. Л., Попов И. И.* Технические средства информатизации: учебник. 2-е изд. М. ФОРУМ: ИНФРА-М, 2008.

12. *Малиновский Б. Н.* История вычислительной техники в лицах. Киев: КИТ; ТОО «А.С.К.», 1995.

13. *Надточий А. И.* Технические средства информатизации: учеб. пособие / под общ. ред. К. И. Курбакова М.: КОС-ИНФ; Рос. экон. акад., 2003.

14. *Нортон П., Сандлер К., Батпей Т.* Персональный компьютер изнутри. М: Бином, 1995.

15. *Партыка Т. Л., Попов И. И.* Вычислительная техника: учеб. пособие. 3-е изд. М.: ФОРУМ, 2012.

16. *Партыка Т. Л., Попов И. И.* Операционные системы, среды и оболочки. 4-е изд. М.: ФОРУМ, 2012.

17. *Партыка Т. Л., Попов И. И.* Периферийные устройства вычислительной техники. М.: ФОРУМ: ИНФРА-М, 2007.

18. *Шпаковский Г. И.* Организация параллельных ЭВМ и суперскалярных процессоров: учеб. пособие. Мн.: Белгосуниверситет, 1996.

Интернет-адреса

19. http://www.cluster.bsu.by
20. http://www.INTUIT.ru
21. http://www.parallel.ru
22. http://www.pctechguide.com
23. http://www.stolica.ru
24. http://www.top500.org
25. http://www.wikipedia.org

Глоссарий терминов и сокращений (русский язык)

Адаптер (от *лат.* adaptare — прилаживать, короче говоря, — «приспособление»). Устройство сопряжения центрального процессора и периферийных устройств компьютера; кроме этого, иногда осуществляет функции управления периферийным устройством. Обычно выполнен в виде микросхемы и помещен на системную плату, может быть представлен отдельной платой. Иногда называется картой или контроллером.

Адрес — номер конкретного байта оперативной памяти компьютера.

Аккумулятор — 1) устройство, вырабатывающее электричество путем преобразования химической энергии в электрическую; 2) ячейка памяти, используемая для хранения результатов вычисления; обычно так называют один из регистров (сумматор) в арифметико-логическом устройстве процессора.

Аксессуар (от *фр.* accesoire — принадлежность). Элемент компьютера или программной среды, который может быть использован только вместе со всей системой, но приобрести и установить его можно отдельно. Например, к мультимедийным аксессуарам компьютера относятся компакт-диски, звуковые адаптеры и колонки, и пр.

Алгебра логики — раздел математики, изучающий высказывания, рассматриваемые со стороны их логических значений (истинности или ложности) и логических операций над ними.

Алгоритм — понятное и точное предписание (указание) исполнителю совершить определенную последовательность действий, направленных на достижение указанной цели или решение поставленной задачи (приводящую от исходных данных к искомому результату).

Аналоговая вычислительная машина — вычислительная машина, которая оперирует данными, представленными в аналоговом виде. Аналоговые вычислительные машины практически всегда жестко специализированы.

Аналого-цифровой преобразователь (АЦП, ADC) — устройство, преобразующее аналоговый сигнал в цифровой.

Арифметико-логическое устройство (АЛУ) — часть процессора, которая производит выполнение операций, предусмотренных данным компьютером.

Арифметические операции — четыре действия арифметики (+, –, *, /) и операция получения остатка от деления (%) образуют группу арифметических операций. Их выполнение не имеет каких-либо особенностей, кроме как преобразование типов переменных при их несовпадении.

Архитектура двойной независимой шины (DIB — Dual independent Bus) — архитектура построения процессора, при которой данные передаются по двум шинам независимо друг от друга, одновременно и параллельно.

Архитектура открытая — архитектура, разработанная фирмой IBM для персональных компьютеров. Основные признаки: наличие общей информационной шины, к которой возможно подключение различных дополнительных устройств через разъемы расширения; модульное построение компьютера.

Архитектура процессора — комплекс его аппаратных и программных средств, предоставляемых пользователю. В это общее понятие входит набор программно-доступных регистров и исполнительных (операционных) устройств, система основных команд и способов адресации, объем и структура адресуемой памяти, виды и способы обработки прерываний.

Архитектура фон Неймана — архитектура компьютера, имеющего одно арифметико-логическое устройство, через которое проходит поток данных, и одно устройство управления, через которое проходит поток команд.

Архитектура ЭВМ — общее описание структуры и функции ЭВМ на уровне, достаточном для понимания принципов работы и системы команд ЭВМ. Архитектура не включает в себя описание деталей технического и физического устройства компьютера. Основные компоненты архитектуры ЭВМ: процессор, внутренняя (основная) память, внешняя память, устройства ввода, устройства вывода.

Асинхронная передача данных — способ передачи и метод извлечения данных из непрерывного потока сообщений, при которых передающая сторона в каждое данное вводит стартовый и стоповый биты, указывающие, где данное начинается и где кончается.

Байт — машинное слово минимальной размерности, адресуемое в процессе обработки данных.

Безусловная передача управления производится командой JMP, которая загружает в программный счетчик СчАК новое содержимое, являющееся адресом следующей выполняемой команды. Этот адрес либо непосредственно указывается в команде JMP (прямая адресация), либо вычисляется как сумма текущего содержимого СчАК и заданного в команде смещения, которое является числом со знаком (относительная адресация).

Бит (от *англ.* **BI**nary digi**T** — двоичная единица) — единица измерения количества информации, равная количеству информации, содержащемуся в опыте, имеющем два равновероятных исхода. Это наименьшая единица информации в цифровом компьютере, принимающая значения «0» или «1».

Блок-схема — 1) графическое представление алгоритма, повышающее наглядность алгоритма. Составление блок-схем особенно полезно начинающим программистам; 2) графическое представление состава технических средств или структуры системы.

Бод (baud), бит/с (bps) — единица измерения скорости передачи данных.

Булева алгебра — раздел математической логики, изучающий высказывания и операции над ними. Над высказываниями возможны операции: И (конъюнкция &, $\wedge$); ИЛИ (дизъюнкция, $\vee$); «если ..., то» (импликация, $\rightarrow$); двусторонняя импликация (эквивалентность, $\sim$); НЕ (отрицание, $\neg$).

Буфер — 1) дополнительная память для временного хранения данных. Буфер предназначен для компенсации более низкой скорости работы выходного устройства по сравнению со скоростями работы процессора и оперативной памяти. Буфер многих лазерных принтеров, например, составляет от 1 до нескольких мегабайт; 2) часть оперативной памяти, используемая системой для временного хранения данных при выполнении операций копирования и переноса.

Быстродействие накопителя — скорость чтения/записи данных в накопителе. Определяется двумя параметрами: средним временем доступа и скоростью передачи данных.

Быстродействие процессора — скорость выполнения операций процессором. Так как скорость выполнения отдельных операций у процессора разная, то за скорость работы всего процессора принимают либо скорость выполнения команд «регистр—регистр», либо скорость выполнения команд над числами с плавающей запятой. Последняя имеет специальное название — флопс (от Flops — **FL**oating point **O**perations **P**er **S**econd). Типичными мерами быстродействия являются 1 Гигафлопс (10^9 оп./с) и 1 Терафлопс (10^{12} оп./с). Обобщенным показателем скорости процессора является тактовая частота и тип процессора. Например, при тактовой частоте 66 МГц у процессора 486-DX скорость 54 млн команд/с; у Pentium при той же частоте — 112 млн команд/с.

Ввод — считывание и передача информации с внешнего устройства в память компьютера.

Вещественное число — тип данных, содержащий числа, записанные с десятичной точкой и (или) с десятичным порядком.

Виртуальная память отличается от обычной ОП тем, что какие-то ее редко используемые фрагменты могут находиться на диске и подгружаться в реальную ОП по мере необходимости.

Внешние устройства (ВУ) подключаются к системе с помощью интерфейсов, реализующих определенные протоколы параллельного или последовательного обмена. К ВУ относятся — клавиатура, монитор, внешние запоминающие устройства, использующие гибкие или жесткие магнитные диски, оптические диски (CD-ROM), магнитные ленты и другие виды носителей информации, датчики и преобразователи информации (аналого-цифровые или цифроаналоговые), исполнительные устройства (индикаторы, принтеры, электродвигатели, реле и другие).

Восьмеричная система счисления — позиционная система счисления с основанием 8. Например, $123_8 = 1 \cdot 8^2 + 2 \cdot 8^1 + 3 \cdot 8^0 = 64 + 16 + 3 = 83_{10}$.

Вывод — результаты работы программы, выдаваемые компьютером пользователю, другому компьютеру или во внешнюю память.

Высказывание — понятие математической логики, определяемое как повествовательное предложение, которое может быть истинным или ложным, но не может быть истинным и ложным одновременно. Над высказываниями возможно производить логические операции. Из простых высказываний строятся сложные.

Гарвардская архитектура характеризуется физическим разделением памяти команд (программ) и памяти данных, что позволяет одновременно с чтением-записью данных производить выборку и декодирование следующей команды.

Генератор тактовой частоты — устройство для выработки через равные отрезки времени последовательности импульсов. Время между двумя последовательными импульсами называется тактом.

Гибкий магнитный диск — диск из гибкой пластмассы в защитной пластмассовой упаковке, в которой прорезаны отверстия для подхода магнитных головок ввода-вывода. Диск покрыт магнитным составом. Часто называется флоппи-диском или дискетой. Используются диски диаметром 5,25 и 3,5 дюймов.

Гигабайт (Гбайт) — единица измерения количества данных или объема памяти, равная 10^9 байт. Альтернативой является предложенный IEC двоичный гигабайт — GiB (GibiByte) = 2^{30} байт = 1024 MiB. Расхождение составляет 2,4 %.

Главная, (внутренняя, оперативная) память компьютера — упорядоченная последовательность (массив) байтов или машинных слов (ячеек памяти). Номер байта или слова памяти, через который оно доступно как из команд компьютера, так и во всех других случаях, называется адресом. Если в команде непосредственно содержится адрес памяти, то такой доступ этому слову памяти называется прямой адресацией.

Датчик — устройство, обеспечивающее регистрацию какой-либо физической величины, преобразование ее в сигналы (обычно электрические) и передачу этих сигналов для обработки в систему управления.

Двоичная система счисления — позиционная система счисления с основанием 2. Для записи чисел используются двоичные цифры 0 и 1. Например, $101101_2 = 1 \cdot 2^5 + 0 \cdot 2^4 + 1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = 45_{10}$. Известны также производные системы счисления (степени 2) — восьмеричная и шестнадцатеричная.

Двойная точность (Double) — числовое данное (с фиксированной или плавающей точкой), размещенное в двух машинных словах, требует поддержки операций специальной арифметики.

Двойное слово — машинное слово двойной длины, используется для увеличения диапазона представления целых чисел. Двойные слова обрабатываются либо отдельными командами процессора, либо программно (эмуляция).

Динамическая оперативная память (от *англ.* Dynamic Random Access Memory — DRAM — динамическая память с произвольным досту-

пом) — тип полупроводниковой оперативной памяти. Каждый двоичный разряд (бит) хранится в схеме, состоящей из транзистора и конденсатора.

Диск — носитель данных в форме круглой пластины, на которую осуществляется запись разными способами. Устройство, которое записывает (читает) данные на/с диск/а, называется накопителем данных. Диски различаются по способу записи/чтения данных, возможности их замены, плотности записи. По способу записи/чтения диски делятся на магнитные, лазерные (оптические), магнитооптические. Магнитные диски, в свою очередь, делятся на гибкие и жесткие. Часто под «диском» подразумевается несколько дисков, объединенных в пакет.

Дисковод — устройство, управляющее вращением магнитного/оптического диска, чтением и записью данных на нем.

Дисциплина обслуживания FIFO (First In First Out) — обслуживание в порядке поступления: «первым пришел — первым обслуживается».

Дисциплина обслуживания LIFO — обслуживание в порядке, обратном порядку поступления: «последним пришел — первым обслуживается» (Last In First Out). Используется при обмене со стековой памятью.

Дополнительный код — беззнаковая форма представления чисел со знаком. В двоичной системе счисления дополнение каждой цифры выглядит как инвертирование двоичного разряда, т. е. замена 0 на 1 и наоборот. Если же знак числа представляется старшим разрядом машинного слова, то получается простой способ представления отрицательного числа — взять абсолютное значение числа в двоичной системе; инвертировать все разряды, включая знаковый; добавить 1.

Драйвер (Driver) — резидентный программный модуль, осуществляющий управление внешним устройством и связь с операционной системой и прикладными программами.

Жесткий магнитный диск (ЖМД, НЖМД) — диск для долговременного хранения данных на компьютерах. В отличие от гибкого магнитного диска, который является съемным, жесткий магнитный диск составляет единое целое с дисководом.

Запоминающее устройство (ЗУ) — устройство для записи, хранения и выдачи данных. Различают устройства долговременного и оперативного хранения данных, они же энергонезависимые и энергозависимые; только для чтения данных (постоянное запоминающее устройство — ПЗУ, компакт-диски) и как для чтения, так и для записи. В зависимости от физических принципов хранения данных различают магнитные, магнитооптические, оптические и полупроводниковые (схемные) устройства.

Защита памяти осуществляется путем блокировки доступа к памяти других процессов. Один из способов — вся память делится на страницы, и у каждой есть замок — 4-битовый признак, который можно установить только привилегированной командой.

Защищенный режим (protected mode) — режим работы процессора Intel 386, при котором он выполняет множество проверок корректности

обращений к памяти, вызовов функций, доступа к портам ввода-вывода и т. д. Такая защищенность позволяет операционной системе обрабатывать ошибочные операции. Для того чтобы иметь возможность использовать все адресное пространство и преимущества виртуальной памяти процессора Intel 386, приложение должно работать в защищенном режиме.

Интегральная схема (ИС) — реализация электронной схемы, выполняющей некоторую функцию, в виде единого полупроводникового кристалла, в котором изготовлены все компоненты, необходимые для осуществления этой функции. Включает совокупность транзисторов, диодов, конденсаторов, резисторов и др. связанных между собой микропроводниками. ИС изготавливается по технологии, обеспечивающей предельно малые размеры и высокую надежность. По количеству элементов ИС условно делят на малые (МИС) — с количеством элементов в кристалле до 10^2, средние (СИС) — до 10^3 , большие (БИС) — до 10^4, сверхбольшие (СБИС) — до 10^6, ультрабольшие (УБИС) — до 10^9 и гигабольшие (ГБИС) — более 10^9 элементов в кристалле.

Интерфейс (от *англ.* inter — между и face — лицо) — 1) взаимодействие между элементами системы или системами; 2) совокупность средств, стандартов, сигналов, разъемов, обеспечивающая обмен данными между устройствами; 3) взаимодействие между человеком и компьютером.

Информация, единицы измерения. Элементарной единицей информации является бит — это один двоичный разряд, 8 бит образуют 1 байт. Более крупными единицами являются: 1) десятичные единицы, а именно — 1 Кбайт = 10^3 байт, 1 Мбайт = 10^3 Кбайт, 1 Гбайт = 10^3 Мбайт, 1 Тбайт = 10^3 Гбайт; 2) бинарные единицы, в том числе 1 KiB = 1024 байт, 1 MiB = 1024 KiB, 1 GiB = 1024 MiB, 1 TiB = 1024 GiB.

Исключения (exception) — нештатные ситуации (ошибки), возникающие при работе процессора.

Исходный код программы — код, написанный на языке программирования.

Картридж (от *англ.* cartrige — патрон, кассета) — сменяемая часть устройства. Обычно это кассета, в которой хранится красящая лента для принтеров, тонер для лазерных принтеров или множительных аппаратов, чернила для струйных принтеров и пр.

Килобайт (Кбайт) — единица измерения количества данных или объема памяти, равная 10^3 байт. Альтернативой является предложенный IEC двоичный килобайт — KiB (KibiByte) = 2^{10} байт = 1024 байт. Расхождение составляет 2,4 %.

Код ASCII (от *англ.* American Standard Code for Information Interchange — Американский стандартный код для обмена информацией) — стандарт кодирования символов латинского алфавита, цифр и вспомогательных символов или действий в виде однобайтового двоичного кода (1 байт = 8 бит). Первоначально стандарт определял только

128 символов, используя 7 бит (от 0 до 127). Использование всех 8 бит позволяет кодировать еще 128 символов. В этом случае говорят о расширенном ASCII-коде. Дополнительные символы могут быть любыми, им отводятся коды от 128 до 255. Символы кириллицы кодируются именно в этой части ASCII-кода.

Код Unicode — стандарт для представления символов с использованием 16-разрядных кодов (2 байта). Допускает 65 536 символов.

Кодирование (Coding) — установление согласованного (узаконенного) соответствия между набором символов и сигналами или битовыми комбинациями, представляющими каждый символ для целей передачи, хранения или обработки данных.

Команда представляет собой двоичное число, которое состоит из двух частей (полей) — кода операции (КОП) и адресной части (АДЧ). Код операции задает вид операции, выполняемой данной командой, а АДЧ определяет выбор операндов (способ адресации), над которыми производится заданная операция. В зависимости от типа микропроцессора команда может содержать различное число разрядов (байтов). Например, команды процессоров Pentium содержат от 1 до 15 байтов, а большинство процессоров с RISC-архитектурой использует фиксированный 4-байтовый формат для любых команд.

Команда тестирования `TST` является одноопернадным вариантом команды сравнения. При выполнении этой команды устанавливаются признаки `N`, `Z` в соответствии со знаком и значением (равно или не равно нулю) адресуемого операнда.

Команды арифметических операций. Основными в этой группе являются команды сложения, вычитания, умножения и деления, которые имеют ряд вариантов. Команды сложения `ADD` и вычитания `SUB` выполняют соответствующие операции с содержимым двух регистров, регистра и ячейки памяти или с использованием непосредственного операнда. Команды `ADC`, `SBB` производят сложение и вычитание с учетом значения признака `C`, устанавливаемого при формировании переноса или заема в процессе выполнения предыдущей операции. Команда `NEG` изменяет знак операнда, переводя его в дополнительный код. Операции умножения и деления могут выполняться над числами со знаком (команды `IMUL`, `IDIV`) или без знака (команды `MUL`, `DIV`). Один из операндов всегда размещается в регистре, второй может находиться в регистре, ячейке памяти или быть непосредственным операндом. Результат операции располагается в регистре.

Команды битовых операций. Эти команды производят установку значения признака `C` в регистре состояний в соответствии со значением тестируемого бита b_n в адресуемом операнде. В некоторых микропроцессорах по результату тестирования бита производится установка признака *Z*. Номер тестируемого бита *n* задается либо содержимым указанного в команде регистра, либо непосредственным операндом.

Команды ввода IN и вывода OUT реализуют пересылку данных из регистра процессора во внешнее устройство или прием данных из внеш-

него устройства в регистр. В этих командах задается номер интерфейсного устройства (порта ввода-вывода), через которое производится передача данных.

Команды логических операций. Практически все процессоры производят логические операции «И», «ИЛИ», «Исключающее ИЛИ», которые выполняются над одноименными разрядами операндов с помощью команд AND, OR, XOR. Операции выполняются над содержимым двух регистров, регистра и ячейки памяти или с использованием непосредственного операнда. Команда NOT инвертирует значение каждого разряда операнда.

Команды организации программных циклов осуществляют условный переход в зависимости от значения содержимого заданного регистра, который используется как счетчик циклов. Например, в процессорах Pentium для организации циклов используется команда LOOP и регистр ECX. Команда LOOP уменьшает содержимое ECX на 1 (декремент) и проверяет полученное значение. Если содержимое ECX $\neq$ 0, то выполняется переход к команде, адрес которой определяется с помощью относительной адресации (смещение задано в команде LOOP). Если ECX = 0, то выполняется следующая команда программы.

Команды пересылки. Основной командой этой группы является команда MOV, которая обеспечивает пересылку данных между двумя регистрами или между регистром и ячейкой памяти.

Команды прерываний INT обеспечивают переход к одной из программ обслуживания исключений и прерываний.

Команды расширения SIMD («Single Instruction-Multiple Data» — «Одна Команда — Множество Данных»). Такие операции широко используются для обработки изображений, звуковых сигналов и в других приложениях. Для выполнения этих операций в состав процессоров введены блоки, реализующие соответствующие наборы команд.

Команды сдвига. Сдвигаемый операнд может находиться в регистре или ячейке памяти, число разрядов сдвига задается с помощью непосредственного операнда, содержащегося в команде, или определяется содержимым заданного регистра. В реализации сдвига обычно участвует признак переноса C в регистре состояний (SR или EFLAGS), в котором располагается последний разряд операнда, выдвигаемый из регистра или ячейки памяти.

Команды сравнения и тестирования. Сравнение операндов обычно осуществляется с помощью команды CMP, которая производит вычитание операндов с установкой значений признаков N (знака), Z (ноля), V (переполнения), C (переноса) в регистре состояния в соответствии с полученным результатом. При этом результат вычитания не сохраняется, и значения операндов не изменяются. Последующий анализ полученных значений признаков позволяет определить относительное значение (>, <, =) операндов со знаком или без знака.

Команды управления признаками обеспечивают запись-чтение содержимого регистра состояния, в котором хранятся признаки, а также изме-

нение значений отдельных признаков. Например, в процессорах Pentium реализуются команды LAHF и SAHF, которые выполняют загрузку младшего байта, где содержатся признаки, из регистра состояния EFLAG в младший байт регистра EAX и заполнение младшего байта EFLAGS из регистра EAX.

Команды управления процессором — команды останова, отсутствия операции и ряд команд, определяющих режим работы процессора или его отдельных блоков. Команда HLT прекращает выполнение программы и переводит процессор в состояние останова.

Команды условных переходов (ветвлений программы) производят загрузку в СчАК нового содержимого, если выполняются определенные условия, которые обычно задаются в соответствии с текущим значением различных признаков в регистре состояния. Если условие не реализуется, то выполняется следующая команда программы.

Компакт-диск — диск для постоянного хранения данных, представляющий собой круг из пластика или алюминиевого сплава, покрытый защитной прозрачной пленкой.

Контро́ллер (от *англ.* control — управлять) — устройство, которое связывает периферийное оборудование или каналы связи с центральным процессором, освобождая процессор от непосредственного управления функционированием данного оборудования. Контроллер выполняет интерпретацию команд процессора для отдельных устройств.

Косвенная адресация — случай, когда машинное слово содержит адрес другого машинного слова. Тогда доступ к данным во втором машинном слове через первое называется косвенной адресацией.

Кэш-память — сверхоперативная память, обращение к которой намного быстрее, чем к оперативной, и в которой хранятся наиболее часто используемые участки последней.

Логические операции. Над значениями условных выражений можно выполнить логические операции «И» (&, AND), «ИЛИ» (|, OR) и «НЕ» (!, NOT), которые объединяют по правилам логики несколько условий в одно. Благодаря тому, что любая логическая операция может быть представлена с помощью трех основных логических операций, набора элементов «И», «ИЛИ» и «НЕ» в принципе достаточно для построения любого устройства процессора компьютера, а также для описания любых алгоритмов.

Логический элемент (вентиль) — часть электронной логической схемы, выполняющая элементарную логическую функцию.

Логическое высказывание — любое предложение, в отношении которого можно однозначно сказать, истинно оно или ложно.

Лэптоп (наколенник — laptop) — портативный компьютер, по своим размерам близкий к портфелю. По быстродействию и памяти примерно соответствует настольным персональным компьютерам.

Магистрально-модульная структура — распространенная структура ЭВМ, в которой отдельные устройства (модули), входящие в состав сис-

темы, обмениваются информацией по общей системной шине — магистрали.

Масштабируемость — свойство системы или ее отдельных частей, характеризующее возможность системы приспосабливаться к уменьшению или увеличению ее отдельных параметров. Например, операционные системы Windows имеют масштабируемый пользовательский интерфейс, который обеспечивает одинаковый внешний вид при использовании дисплеев разных размеров.

Математический сопроцессор — интегральная схема, дополнительная к главному (центральному) процессору, которая выполняет команды, работающие с числами, представленными в форме с плавающей точкой (запятой).

Машинное слово (МС) — упорядоченное множество двоичных разрядов, используемое для хранения команд программы и обрабатываемых данных. Каждый разряд, называемый битом, — это двоичное число, принимающее значения только «0» или «1». Разряды в МС обычно нумеруются справа налево начиная с 0. Количество разрядов в МС называется размерностью МС или его разрядностью. Типовая длина МС — 16 или 32 бита.

Машинный язык — совокупность машинных команд компьютера.

МДП-структура — структура «металл—диэлектрик—полупроводник», используемая при создании электронных приборов, в том числе микропроцессоров, памяти для компьютеров. Представляет собой упорядоченную совокупность очень тонких (менее 1 мкм) слоев металла и диэлектрика, нанесенных на полупроводниковую пластину. Если в качестве диэлектрика используются оксиды (оксид алюминия, диоксид кремния), то образуется МОП-структура («металл—оксид—полупроводник»). Метод создания приборов на основе таких структур называется МДП- или МОП-технологией.

Мегабайт (Мбайт) — единица измерения количества данных или объема памяти, равная 10^6 байт. Альтернативой является предложенный IEC двоичный мегабайт — MiB (MibiByte) = 1024 KiB. Расхождение составляет более 4,8 %.

Микрокоманда — элементарное действие, обеспечивающее выполнение процессором заданной операции, УУ процессора генерирует последовательность микрокоманд в соответствии с кодом поступившей команды. Каждая микрокоманда выполняется в течение одного машинного такта.

Микрометр (мкм) — 10^{-6} м, 1000 нанометров (нм).

Микрооперации (micro-operations, micro-ops, μops) — простые RISC-подобные команды микропроцессора, используемые в некоторых CISC-процессорах для реализации более сложных команд.

Микропроцессор — устройство, осуществляющее обработку данных и управляющее устройствами, выполненное в виде одной или нескольких больших (сверхбольших) интегральных схем. Микропроцессоры встраиваются в устройства управления и входят основной частью в компью-

тер. Например, в автомобиле марки «БМВ» установлены 54 интегральные схемы, которые управляют тормозами с антиблокировкой и воздушными подушками безопасности. В состав микропроцессора входят арифметико-логическое устройство, выполняющее арифметические и логические операции; блок управления и синхронизации; блок ввода-вывода; регистры и пр.

Микросекунда (мс) — 10^{-6} с, 1000 наносекунд (нс).

Многозадачность — режим одновременного решения нескольких задач на компьютере. Под задачей в данном случае понимается часть работы, выполняемой процессором.

МОП-структура — структура материала, из которого изготовляются транзисторы, конденсаторы и др. электронные приборы. Сокращение от «металл—оксид—полупроводник».

Мультимедиа — собирательное понятие для различных компьютерных технологий, при которых используется несколько информационных сред, таких, как графика, текст, видео, фотография, движущиеся образы (анимация), звуковые эффекты, высококачественное звуковое сопровождение.

Мэйнфрейм (от *англ.* main — главный, frame — сооружение, буксир, тягач) — большой, очень мощный компьютер общего назначения, используемый для работы в качестве суперсерверов в мощных сетях и объемных научных расчетах. Мэйнфреймы занимают промежуточное место между персональными и суперкомпьютерами.

Накопитель на гибком магнитном диске (НГМД) — устройство для записи/чтения данных на гибкий пластиковый диск, покрытый магнитным слоем. Устройство состоит из двух двигателей, один из которых вращает диск (360 об/мин), а другой (шаговый) передвигает головки чтения/записи по концентрическим окружностям (трекам). Диски свободно вставляются в устройство. НГМД часто называют дисководом.

Накопитель на жестком магнитном диске (НЖМД) — устройство для записи/чтения данных на жестком диске (иногда называют «винчестером»). НЖМД впервые использован в персональном компьютере в 1983 г. фирмой IBM. Наиболее массовое запоминающее устройство большой емкости, в котором носителями информации являются круглые алюминиевые пластины — платтеры, обе поверхности которых покрыты слоем магнитного материала. Используется для постоянного хранения больших объемов информации.

Накопитель на компакт-диске (CD-ROM) — оптический накопитель, допускающий только считывание данных с носителя (**C**ompact **D**isk **R**ead **O**nly **M**emory — память только для чтения на компактном диске). Содержит двигатель для вращения диска, генератор лазерного луча и преобразователь отраженного лазерного луча в электрические сигналы, соответствующие «0» и «1».

Нанометр (нм) — 10^{-9} м = 0,001 микрометра (мкм).

Наносекунда (нс) — 10^{-9} с = 0,001 микросекунды (мс).

Нанотехнология — технология изготовления интегральных схем, которая базируется на величинах, соответствующих нанометрам и наносекундам. Например, выражение «технология (процесс) 130 нм (или 0,13 мкм)» означает, что размеры структурных элементов микросхемы не превосходят 130 нм.

Ноутбук (блокнот) — портативный компьютер, по своим размерам близкий к книге крупного формата. Обычно комплектуется модемом и снабжается приводом CD-ROM.

Оперативная память (ОП) (то же, что оперативное запоминающее устройство — ОЗУ) предназначена для хранения программ и данных, которыми они манипулируют. Физически выполнена в виде некоторого числа микросхем. Логически ОП можно представить как линейную совокупность ячеек, каждая из которых имеет свой номер, называемый адресом (или матрицу, пересечение строк и столбцов которой определяет конкретный бит памяти).

Операции управления программой — команды безусловной передачи управления; условных переходов; организации программных циклов; команды прерывания; команды изменения признаков.

Основание системы счисления — количество различных цифр, используемых для изображения чисел в данной системе счисления.

Палмтоп (наладонник, palmtop) — самый маленький современный персональный компьютер, магнитные диски в нем заменяет энергонезависимая память.

Поколения компьютеров — условная, нестрогая классификация вычислительных систем по степени развития аппаратных и программных средств, а также способов общения с ними.

Порты устройств — электронные схемы, содержащие один или несколько регистров ввода-вывода и позволяющие подключать периферийные устройства компьютера к внешним шинам микропроцессора. Последовательный порт обменивается данными с процессором побайтно, а с внешними устройствами — побитно. Параллельный порт получает и посылает данные побайтно.

Постоянное запоминающее устройство (ПЗУ) служит для хранения констант и стандартных (неизменяемых) программ. В ПЗУ обычно записываются программы начальной инициализации (загрузки) систем, тестовые и диагностические программы и другое служебное программное обеспечение, которое не меняется в процессе эксплуатации систем. В микропроцессорных системах, управляющих определенными объектами с использованием фиксированных или редко изменяемых программ, для их хранения также обычно используется ПЗУ (память ROM — Read-Only Memory) или репрограммируемое ПЗУ (память EEPROM — Electrically Erased Programmable Read-Only Memory или флэш-память).

Прерывания (interruption) — ситуации, возникающие при поступлении соответствующих команд (программные прерывания) или сигналов от внешних устройств (аппаратные прерывания). При этом процессор останавливает всякую другую деятельность и вызывает программу ***обра-***

ботчик прерывания. По окончании ее работы он продолжает прерванную работу с того места, где она остановилась.

Принстонская архитектура (часто называется архитектурой фон Неймана) характеризуется использованием общей оперативной памяти для хранения программ, данных, а также для организации стека.

Принтер — печатающее устройство, преобразует закодированную информацию, выходящую из процессора, в форму, удобную для чтения на бумаге.

Принципы фон Неймана включают в себя: *принцип программного управления.* Программа состоит из набора команд, которые выполняются процессором автоматически друг за другом в определенной последовательности; *принцип адресности.* Основная память состоит из перенумерованных ячеек; процессору времени доступна любая ячейка; *принцип однородности памяти.* Программы и данные хранятся в одной и той же памяти и над командами можно выполнять такие же действия, как и над данными.

Процессор — центральное устройство компьютера. Возможности компьютера как универсального исполнителя по работе с информацией определяются системой команд процессора. Отдельная команда определяет отдельную операцию (действие) компьютера. Эта система команд представляет собой язык машинных команд, который содержит команды арифметических и логических операций, операций управления последовательностью выполнения команд, передачи данных из одних устройств памяти в другие и пр. В состав процессора входят устройство управления (УУ), арифметико-логические устройство (АЛУ), регистры процессорной памяти.

Прямой доступ к памяти (DMA — Direct Memory Access) используется для пересылки значительного массива информации между ОП и каким-либо внешним устройством, которое подает в систему соответствующий запрос. Реализация такой пересылки с помощью соответствующей программы обмена требует выполнения отдельной команды пересылки для передачи каждого байта или слова. При этом необходим определенный объем памяти для хранения программы и требуется значительное время для ее выполнения.

Реальный режим (real mode) — режим работы процессора Intel 386, совместимый с процессором Intel 8086. В реальном режиме невозможны доступ к *виртуальному адресному пространству* процессора 386 или такие возможности, как *замещение страниц по требованию.*

Регистр — запоминающий элемент процессора, выполняющий функции кратковременного хранения чисел или команд и выполнения над ними некоторых операций.

Регистр команд — регистр УУ для хранения кода команды на период времени, необходимый для ее выполнения.

Регистр состояния (State Register, `SR` — в микропроцессорах Pentium называется `EFLAGS`) определяет текущее состояние процессора при выполнении программы. Регистр содержит биты управления, задающие ре-

жим работы процессора, и биты признаков (флаги), указывающие характеристики результата выполненной операции: N — признак знака (старший бит результата), N = 0 — при положительном результате, N = 1 — при отрицательном результате; C — признак переноса, C = 1, если при выполнении операции образовался перенос из старшего разряда результата; V — признак переполнения, V = 1, если при выполнении операций над числами со знаком произошло переполнение разрядной сетки процессора; Z — признак нуля, Z = 1, если результат операции равен нулю. Некоторые микропроцессоры фиксируют также другие виды признаков.

Символьное данное (Character) — тип данных, предназначенный для ввода и отображения алфавитной, цифровой и спецсимвольной информации; типу соответствуют определенные операции над переменными и функции (строчные).

Система команд. Процессоры выполняют набор команд, которые реализуют следующие основные группы операций — пересылки; арифметические; логические; сдвига; сравнения и тестирования; битовые операции; управления программой; управления процессором.

Система счисления — совокупность правил наименования и изображения чисел с помощью набора символов, называемых цифрами. Системы счисления делятся на позиционные и непозиционные. Пример непозиционной системы счисления — римская, к позиционным системам счисления относится двоичная, десятичная, восьмеричная, шестнадцатеричная.

Системная шина содержит несколько десятков (в сложных системах более 100) проводников (линий), которые в соответствии с их функциональным назначением подразделяются на отдельные шины — адреса A, данных D и управления C. Шина A служит для передачи адреса, который формируется микропроцессором и позволяет выбрать необходимую ячейку ОП (ПЗУ) или адрес при обращении к внешнему устройству. Шина D служит для выборки команд, поступающих из ОП или ПЗУ в УУ микропроцессора, и для пересылки обрабатываемых данных (операндов) между микропроцессором и ОП или внешним устройством. По шине C передаются управляющие сигналы, определяющие режимы работы памяти (запись или считывание), интерфейсных устройств (ввод или вывод информации) и микропроцессора (запуск, запросы внешних устройств на обслуживание, информация о текущем режиме работы и другие сигналы).

Сканер — устройство для ввода в компьютер документов — текстов, чертежей, графиков, рисунков, фотографий. Создает оцифрованное изображение документа и помещает его в память компьютера.

Стандартное машинное слово — машинное слово, размерность которого совпадает с разрядностью процессора. Большинство команд процессора использует для обработки данных стандартное машинное слово.

Стек — совокупность ячеек памяти, которые доступны не в произвольном порядке, а только в стековом («магазинном»): LIFO — «последним вошел — первым вышел», подобно патронам в обойме (магази-

не) винтовки (автомата), например среда для размещения данных для возврата из подпрограмм, а также их аргументов.

Стример — устройство для резервного копирования больших объемов информации. В качестве носителя применяются кассеты с магнитной лентой емкостью 1—2 Гбайта и больше.

Сумматор — электронная логическая схема, выполняющая суммирование двоичных чисел.

Суперкомпьютер — очень мощный компьютер с производительностью свыше 100 мегафлопов (1 мегафлоп, Мфлоп, Mflop — миллион операций с плавающей точкой в секунду). Представляет собой многопроцессорный и (или) многомашинный комплекс, работающий на общую память и общее поле внешних устройств. Архитектура основана на идеях параллелизма и конвейеризации вычислений.

Суперскалярная структура процессора обеспечивает повышение производительности процессора путем введения в структуру процессора нескольких параллельно включенных операционных устройств, обеспечивающих одновременное выполнение нескольких операций. В таких процессорах реализуется параллельная работа нескольких исполнительных конвейеров, в каждый из которых поступает для выполнения одна из выбранных и декодированных команд.

Счетчик адреса команд (СчАК, или регистр адреса команд — РАК; или программный счетчик — Program Counter, PC; в x86 — Instruction Pointer, IP) — регистр процессора, который служит для хранения адреса очередной команды и содержимое которого автоматически увеличивается на 1 после выборки следующей команды. Таким образом обеспечивается последовательная выборка команд в процессе выполнения программы. При выборке очередной команды содержимое PC поступает на шину адреса, обеспечивая считывание из ОП следующей команды выполняемой программы. При реализации безусловных или условных переходов (ветвлений) или других изменений последовательности выполнения команд производится загрузка в PC нового содержимого, в результате осуществляется переход к другой ветви программы или подпрограмме.

Счетчик является устройством, которое на своих выходах выдает (в двоичной форме) сумму числа импульсов, подаваемых на его единственный вход. Максимальное число импульсов, которое счетчик может подсчитать, называется его *емкостью.*

Таблица истинности — табличное представление логической схемы (операции), в котором перечислены все возможные сочетания значений истинности входных сигналов (операндов) вместе со значением истинности выходного сигнала (результата операции) для каждого из этих сочетаний.

Терабайт (Тбайт) — единица измерения количества данных, равная 10^3 Гбайт. В качестве двоичной альтернативы IEC предложила в 1998 г. TiB (TibiByte) = 1024 GiB (GibiByte).

Тип данных — форма представления данных, которая характеризуется способом организации данных в памяти, множеством допустимых

значений, набором операций. Иначе говоря, тип данных — это схема определенного вида переменных, заложенная в транслятор. Тип данных `INTEGER` (`int`, `FIXED` и пр.) задает свойства целой переменной — она может принимать только целые значения в определенном диапазоне, зависящем от разрядности процессора, например, `-32767...+32767`. Тип данных `REAL` (`double`, `FLOAT` и пр.) задает свойства переменной с плавающей точкой. Число с плавающей точкой (или вещественное, действительное число) может принимать значения десятичной дроби (например, 287,3) и имеет ограничения на количество значащих цифр (точность представления). Если же в операции присутствуют переменные обоих типов, то первый тип преобразуется во второй (целое число в вещественное).

Триггер — электронная схема, применяемая в регистрах компьютера для запоминания одного бита информации. Имеет два устойчивых состояния, которые соответствуют двоичным «1» и «0».

Устройство управления (УУ) — часть процессора, выполняющая функции управления устройствами компьютера.

Файл (file) — именованный организованный набор данных определенного типа и назначения, находящийся под управлением операционной системы. Это однородная по своему составу и назначению совокупность информации, хранящаяся на носителе информации и имеющая имя.

Фиксированная точка (fixed) — простейший тип числовых данных, когда число размещено в машинном слове, а диапазон значений зависит только от разрядности слова.

Флоппи-диск (дискета) — съемный гибкий магнитный диск.

Число с плавающей точкой (float) — числовое данное, размещенное в машинном слове в форме мантиссы и порядка, что позволяет представлять широкий диапазон значений; предполагает наличие встроенной или эмулируемой арифметики (операции с плавающей точкой). Для использования чисел с дробной частью, а также для расширения диапазона используемых числовых данных вводится форма представления вещественных чисел или чисел с плавающей точкой: $X = m \times 10^p$, например $25{,}4 = 0{,}254 \times 10^2$, где $0{,}1 < m < 1$ — значащая часть числа, приведенная к интервалу 0,1 ... 1, называемая мантиссой, а p — целое число, называемое порядком. Аналогично, если взять основание степени 2, то получим: $X = m \times 2^p$, где $0{,}5 < m < 1$ — мантисса, а p — двоичный порядок.

Язык ассемблера — система обозначений, используемая для представления в удобочитаемой форме программ, записанных в машинном коде. Перевод программы с языка ассемблера на машинный язык осуществляется специальной программой, которая называется *ассемблером* и является, по сути, простейшим транслятором.

Глоссарий терминов (английский язык)

3DNow! — наименование мультимедийного расширения традиционной системы команд (x86), используемого AMD в своих процессорах, начиная с K6-2 в 1998 г. Это SIMD-инструкции, повышающие эффективность ЦП при операциях над векторами, типичными для обработки графической информации.

ADC (Analogue-to-Digital Converter) — аналого-цифровой преобразователь (АЦП) — устройство, преобразующее непрерывный аналоговый сигнал, который поступает от физического датчика и соответствует скорости, температуре, интенсивности звука, света и пр., в бинарный код для ввода в компьютер (каждому значению напряжения входного аналогового сигнала соответствует определенное значение выходного цифрового кода).

APIC (Advanced Programmable Interrupt Controller) — усовершенствованный программируемый контроллер прерываний. Встроенный контроллер прерываний в процессорах Pentium/Atom для симметричной многопроцессорной обработки.

ASCII (American Standard Code for Information Interchange) — разработанный American National Standards Institute (ANSI) стандарт представления символьной информации в ЭВМ. Символы ASCII содержат 128 символов с кодами от 0 до 127 и включают цифры, знаки пунктуации, буквы и управляющие коды, такие, как конец строки или перевод страницы.

Bandwidth (полоса пропускания) — количество информации, которое может быть передано через конкретный интерфейс за данный период времени, например шина памяти SDRAM на 64 бита и 100 МГц имеет полосу пропускания 800 Мбайт/с.

Banks — банки (памяти).

BEDO DRAM (Burst EDO DRAM) — EDO DRAM с пакетным доступом.

Bit — двоичная единица, базовая мера для измерения количества данных. Имеет значение «1» или « 0». Для размещения 1 байта требуется 8 бит.

BPI (Bits Per Inch — бит на дюйм) — мера измерения плотности информации на запоминающем устройстве.

Branch prediction — предсказание переходов, один из методов повышения производительности ЦП. Обычно компилятор транслирует оператор ветвления (например, `if-then-else`) в блоки машинного кода, расположенные последовательно в потоке. Современные про-

цессоры стараются предсказать результат вычисления условий ветвления и предварительно выполняют предсказанный блок.

Buffered Write Through — буферизированная сквозная запись (в кэш-памяти).

Bus (шина) — линия передачи информации посредством электрических сигналов от одной микросхемы (устройства) к другой. Наиболее часто упоминается в контексте коммуникации между процессором и другими компонентами системы. Известны различные типы шин, включающие ISA, EISA, LPC и стандарты локальных шин PCI и VL.

Byte — восемь бит, рассматриваемые как единое целое и представляющие, например, символ кода ASCII.

Cache Level 1 (L1) — кэш первого уровня (внутренняя память процессора, обычно находится или на кристалле — on-die, или на плате процессора — on-chip).

Cache Level 2 (L2) — кэш второго уровня (внешняя память, обычно находится на плате процессора или на системной плате).

Cache Memory (кэш-память) — память, необходимая для того, чтобы центральный процессор меньше простаивал из-за низкого быстродействия основной памяти, расположена между процессором и основной памятью. Объем и быстродействие кэш-памяти являются определяющими параметрами быстродействия системной платы и/или центрального процессора для подавляющего большинства задач, решаемых на компьютере.

Call Far — дальний вызов (в ассемблере).

CAS (Column Address Strobe) — сигнал выборки столбца адреса в DRAM.

CCIA (Computer and Communications Industry Association) — ассоциация фирм-производителей компьютеров и средств коммуникации, представляющая их интересы в зарубежной и национальной торговле, а также разрабатывающая соответствующие стандарты.

CD-DA (Digital Audio) — классический аудиодиск. Поддерживается практически всеми приводами.

CD-ROM (Compact Disk Read Only Memory) — устройство для считывания компакт-дисков (CD). Диск диаметром 5 дюймов емкостью 640—700 Мбайт имеет одну спиральную дорожку. Время доступа относительно велико (у лучших моделей — 80 нс), чувствителен к вибрациям при работе. Интерфейсы: SCSI, IDE (E-IDE, IDE ATAPI). Исполнение — внутреннее и внешнее (SCSI, LPT-порт).

CD-RW (CD ReWritable) — перезаписываемые диски, оптические диски, допускающие многократную запись информации. Как правило, возможно выполнить до 1000 циклов записи на один диск.

Chipset — набор микросхем (обычно для системной платы).

CISC (Complex Instruction Set Computer) — архитектура, для которой характерен большой набор разноформатных команд с использованием многочисленных способов адресации. Эта классическая архитектура процессоров, которая начала свое развитие в 1940-х годах с появлением

первых компьютеров. Типичным примером CISC-процессоров являются микропроцессоры семейства Pentium. Они выполняют более 200 команд разной степени сложности, которые имеют размер от 1 до 15 байт и обеспечивают более 10 различных способов адресации.

Clone — компьютерные системы, совместимые со стандартом IBM PC.

CMOS RAM (Complimentary Metal Oxide Semiconductor Memory) — КМОП-память.

COM Port — разъем и соответствующие электрические цепи, которые позволяют последовательным устройствам (таким, как модем), присоединяться к ЭВМ. Коммуникационный порт также называется последовательным портом и в операционных системах обозначается именами, начинающимися с COM (например, COM1 или COM2).

Controller — контроллер, специализированный процессор управления обменом с внешними устройствами.

CP-866 — стандарт IBM для интерпретации 2-й половины (128—256) кода ASCII, таблица предназначена для русской кириллицы.

CPU (Central Processing Unit) — центральный процессор.

CRC (Cyclical Redundancy Check) — циклическая контрольная сумма — математический метод, позволяющий обнаружить ошибку в длинном блоке данных с высокой точностью.

CS (Code Segment) — код сегмента (содержит начальный адрес сегмента памяти).

CYL (CYLinders) — цилиндры.

DAC (Digital-to-Analogue Converter) — цифроаналоговый преобразователь (ЦАП), устройство (как правило, на одной микросхеме), которое преобразует цифровые данные в аналоговый сигнал. Обычно содержатся в графических картах, видеокартах, модемах и др.

DD (Double Density) — двойная плотность записи (на дискетах).

DDR SDRAM (Double Data Rate SDRAM) — SDRAM с удвоенной скоростью обмена данными, вид памяти. Как видно из названия, пропускная способность DDR SDRAM в 2 раза выше обычной. Этот вид памяти также иногда называется SDRAM II.

Device — устройство.

Device Driver — драйвер устройства. Программное обеспечение, соединяющее внешнее устройство с операционной системой. Драйвер выступает в качестве «переводчика» между командами устройства и программных приложений.

DIMM (Dual In-line Memory Module) — двухсторонний модуль памяти, конструктив модуля памяти, имеет по 84 вывода с каждой стороны. Собственно память, размещенная на модуле, может быть как FPM или EDO, так и SDRAM. Память в DIMM имеет разрядность 64 бита (с четностью 72) и может использоваться поодиночке, а не парами, как обычные SIMM.

DIN connector — европейский (точнее, германский) стандарт разъема, используемый в основном для подключения устройств записи-вос-

произведения звука. Разъемы DIN используются для подсоединения к компьютеру клавиатуры, мыши PS/2 и пр.

Direct-mappad cache — кэш прямого отображения.

DMA (Direct Memory Access) — прямой доступ к памяти, режим обмена данными между памятью и устройством ввода-вывода, управляемый специальным устройством (контроллером DMA) и выполняемый без участия центрального процессора. Использование этого режима значительно ускоряет пересылку данных, так как исключает пересылки данных в процессор и обратно.

DPI (Dots Per Inch) — разрешающая способность устройств отображения и печати, обозначается также как «тнд» — «точек на дюйм».

DRAM (Dinamic Random Access Memory) — динамическая память прямого доступа — память, схемотехнически выполненная в виде двумерной матрицы (строки × столбцы) конденсаторов. Очень дешева, но требует постоянной регенерации (*refresh*) заряда на конденсаторах. Регенерация выполняется как «пустое» чтение памяти. Этот процесс отнимает значительное время, так как в этот период никакое устройство не может получить доступ к памяти, кроме контроллера регенерации.

DTR (Data Transfer Rate) — скорость передачи данных, обычно указываемая в KBps (Кбайс/с) или MBps (Мбайт/с), иногда — в MB/min (Мбайт/мин). Чаще всего характеризует максимальную (пиковую) пропускную способность.

ECC (Error Control Correction) — режим контроля функционирования памяти с восстановлением ошибок. Применяется в некоторых системных платах с соответствующими наборами микросхем для особо критичных к надежности функционирования компьютеров. В основном это серверы и мощные графические станции с большими объемами памяти. Для этого обязательно должны быть установлены планки памяти с истинной четностью. В этом режиме возможно исправление только одиночных ошибок.

EDO DRAM (Extended Data Output DRAM) — DRAM с уменьшенным временем доступа к данным. Extended Data Output DRAM при работе на запись не дает никаких преимуществ по сравнению с FPM DRAM, но требует существенно меньше времени при чтении за счет того, что данные удерживаются на выходе микросхемы EDO DRAM дольше, чем в FPM DRAM. EDO (как и FPM) не дает никаких преимуществ при произвольной выборке данных из памяти, но такая ситуация возникает крайне редко. На микросхемах EDO DRAM указывается время доступа в наносекундах к данным в случайном порядке.

EDRAM (Enhanced DRAM) — усовершенствованная DRAM.

EEPROM (Electrically Erasable Programmable ROM) — электрически стираемое программируемое ПЗУ.

EIA (Electronic Industries Association) — ассоциация производителей, представляющих высокотехнологичные производства США, основанная в 1924 г. как Radio Manufacturers Association. EIA, в частности, разрабо-

таны такие стандарты последовательных устройств, как RS-232, RS-422 и RS-423.

EIDE-контроллер (**E**nhanced **IDE**) — улучшенный EIDE-контроллер.

EPIC (**E**xplicitly **P**arallel **I**nstruction **C**omputing) — параллельное выполнение команд — парадигма вычислений, предложенная Intel и HP и реализованная в архитектуре Intel IA-64 (процессоры Itanium и Itanium 2). Целью EPIC является повышение способности процессоров выполнять команды параллельно, используя компилятор вместо электронных цепей процессора, для того, чтобы выявить в программе и реализовать возможности параллельных вычислений. Технологию Intel выполнения команд в формате IA-64 (команды упакованы по три в 128-битовый пакет-«связку» для параллельной обработки) обычно называют «LIW encoding» («кодирование в длинные слова команд»).

EPROM (**E**rasable **P**rogrammable **ROM**) — перезаписываемое программируемое ПЗУ. Неразрушаемая память, предназначенная для хранения данных после выключения питания Это массив транзисторов с плавающим затвором, программируемых путем приложения более высокого напряжения, чем это необходимо для работы устройства. Информация может быть удалена путем УФО (длина волны обычно 235 нм). Чип EPROM легко распознать по наличию прозрачного кварцевого окна, предназначенного для пропуска УФО.

Expansion Card — карта расширения. Плата с электроцепями и микросхемами, которая вставляется в разъем расширения ПК и реализует определенную добавочную функцию (например, модем, звуковая карта, интерфейс SCSI и пр.)

Execute Disable Bit (XDB), или **N**o-e**X**ecute bit (NXB) — функция, которая при поддержке операционной системы позволяет помечать области памяти как выполнимые или невыполнимые. Если производится попытка выполнения кода в невыполнимой области, процессор передает ОС сообщение об ошибке. Эта функция может предотвратить атаки некоторых классов вирусов или червей, направленные на переполнение буфера, и позволяет повысить общую безопасность системы.

FAT (**F**ile **A**llocation **T**able) — таблица размещения файлов.

FDD (**F**loppy **D**isc **D**rive) — накопитель на гибких магнитных дисках.

Fetch — выборка команды из ОП в процессор. В современных процессорах чаще трактуется как выборка команды из кэш-памяти ЦП в декодер инструкций. Выборка же из ОП в кэш ЦП называется `prefetch` (предвыборка). Причем в ЦП AMD K8 различаются: `prefetchnta` — перемещение данных в L1-кэш, минуя L2-кэш; `prefetcht0` — в L1-кэш и L2-кэш; `prefetcht1` — только в L2.

FIFO (**F**irst **I**nput **F**irst **O**ut, «первым вошел — первым вышел») — способ организации памяти, при котором записанные данные сдвигаются вперед при поступлении новых данных. Как правило, память с та-

кой организацией используется в качестве буферной при приеме-передаче данных.

Firmware — ПЗУ (постоянное запоминающее устройство).

Flash Memory — флэш-память, хранящая информацию при выключенном питании.

Flops (FLoating point operation per second) — быстродействие в количестве операций с плавающей точкой, выполняемых в секунду. Производные единицы — Mflops (10^6 оп./с), Gflops (10^9 оп./с), Tflops (10^{12} оп./с). Употребляются также транслитерации: флопс, Мегафлопс, Гигафлопс, Терафлопс.

FPM (Fast Page Mode) — быстрый страничный метод (вид памяти DRAM).

FPU (Floating Point Unit) — устройство/блок для выполнения арифметических операций с плавающей точкой.

Fully associative cache — полностью ассоциативная кэш-память.

GB (Gigabyte) — гигабайт, единица измерения, содержащая 10^3 Мбайт. В качестве двоичной альтернативы IEC предложила в 1998 г. GiB (GibiByte), 1 GiB = 1024 MiB (MibiByte).

GBps (Gigabytes per second) — гигабайт в секунду, скорость передачи информации.

HDD (Hard Disc Drive) — накопитель на жестких магнитных дисках (винчестер).

Gigatransfer (GT) и ***Megatransfer (MT)*** — термины, относящиеся к количеству операций по передаче данных. Наиболее часто используются для измерения числа передач данных в секунду (GT/s или MT/s). 1 GT/s означает 10^9 или один миллиард передач за секунду, 1 MT/s — 10^6 или один миллион передач. Эти единицы характеризуют реальное (эффективное) количество передач данных, осуществляемых по шине или другому интерфейсу. В частности, если шина памяти работает с двойной частотой передачи данных (информация передается как по переднему, так и по заднему фронту импульсов задающего генератора — системы памяти DDR), то при частоте генератора в 100 МГц эффективная скорость составит 200 MT/s. Единица MT/s чаще ассоциируется с таким интерфейсом, как SCSI (Small Computer Systems Interface), в то время как GT/s — с системами PCI Express или HyperTransport.

Huffman Coding — код Хаффмана, основанный на использовании коротких кодовых последовательностей для более часто появляющихся символов и более длинных для редких. Код Хаффмана минимизирует среднее количество битов для представления заданного потока символов текста.

Hz — Гц (герц), мера частоты в циклах в секунду (1/с).

IBM PC — фирма IBM явилась одним из основателей индустрии ПК, выпустив IBM PC в 1981 г. Затем были разработаны PC XT, AT и пр.

IBM POWER (Performance Optimization With Enhanced RISC) — оптимизация эффективности путем развития RISC. Это набор RISC-по-

добных команд, предложенный фирмой IBM, а также наименование серии микропроцессоров, которые поддерживают эти команды. Процессоры серии POWER используются как ЦП в различных машинах фирмы IBM — серверах, рабочих станциях, мини- и суперкомпьютерах (см. табл. 2.9). Это были ЦП — POWER1 (RS/6000, или RIOS, позднее «RIOS I», 1990 г., ЭВМ PowerPC); POWER2 (1993 г.); POWER3 (1997 г. 64-разрядная архитектура); POWER4 (2001 г., ЭВМ PowerPC 2.00), POWER5 (2005 г.) — двухъядерный процессор, поддерживавший 4 логических подпроцесса.

iCOMP (Intel Comparative Microprocessor Performance) — тест для вычисления производительности микропроцессоров.

IDE-контроллер (Integrated Drive Electronics) — интегрированная электроника накопителя.

IEC (International Electrotechnical Commission) — международная организация по стандартизации в области электротехники. В частности, в 1998 г. IEC предприняла попытку устранить разночтения в трактовке 1 Мбайта как 1000 байт (десятичный мегабайт) и 1024 байт (бинарный мегабайт), предложив обозначения — 1 Megabyte и 1 MibiByte, соответственно. Аналогично были определены 1 KibiByte = 1024 байт, 1 GibiByte = 1024 MibiByte и 1 TibiByte = 1024 GibiByte.

IEEE (Institute of Electrical and Electronics Engineers) — общественная организация, которая объединяет специалистов, ученых, студентов и других лиц, заинтересованных в электронике и смежных областях. Известна более как разработчик и популяризатор стандартов в вычислительной технике и связи, например IEEE 802 — стандарт для локальных сетей.

IEEE floating-point standard использует форму арифметики насыщения для работы с действительными числами, где при переполнении результат устанавливается в «+∞» или «–∞» и любые дальнейшие операции не изменяют полученного «значения». Например, 32 бита — одинарный формат IEEE-754 — включает 1 знаковый разряд, 8 разрядов порядка и 23-разрядную мантиссу числа с одним спрятанным разрядом (в итоге 24 разряда точности).

Integer — тип чисел, представляемый в памяти ЭВМ в формате с фиксированной запятой/точкой (целый, `FIXED`). Число занимает машинное слово. Варианты — числа двойной длины (два машинных слова) — `DOUBLE FIXED`, `LONG FIXED` и половинной (0,5 слова) — `SHORT FIXED`.

IP (Instruction Pointer) — указатель команды (регистр в МП, хранящий адрес выполняемой команды). Он же счетчик адреса команд — СчАК.

ISA (Industry Standard Architecture) — стандарт системной шины.

ISO (International Standards Organisation) — Международная организация по стандартизации (МОС) — международный орган, ответственный за создание и контроль деятельности различных комитетов по

стандартизации и рабочих групп, работающих над стандартами обработки данных (например, сжатия изображений и пр.).

KB (**K**ilo**b**yte, Кбайт) — килобайт, мера количества информации в 10^3 байт. Альтернативой является предложенная IEC двоичная единица KiB (KibiByte); 1 KiB = 1024 байт.

Kbit (Kilobit, Кбит) — килобит, единица измерения объема данных, равная 1024 бит, часто употребляемая для измерения скорости передачи данных (Кбит/с).

KBps (**K**ilobytes **p**er second) — килобайт в секунду, мера скорости передачи данных.

LIFO (Last In First Out) — «последним вошел — первым вышел» (дисциплина обслуживания стековой памяти).

Linpack Benchmark (тест Linpack) — измерение производительности вычислительной системы (R_{max}). Использует программную библиотеку Linpack, написанную на языке Фортран (J. Dongarra и др.). Тот факт, что решение системы n уравнений с n неизвестными ($\mathbf{A} \times \mathbf{x} = \mathbf{b}$) методом Гаусса требует $\frac{2}{3}n^3 + n^2$ операций с ПЗ, позволяет оценить производительность системы, измеряемую в мегафлопс (миллионы операций с ПЗ в секунду).

Macintosh (Mac). Выпущенный Apple Computer в 1984 г., ПК Macintosh явился торговой маркой для ПК, использующего графический интерфейс пользователя (GUI), который включает окна, ярлыки и манипулятор «мышь». Успех Macintosh GUI открыл новый период развития интерфейсов и ОС, базирующихся на графике. В дальнейшем Microsoft в своей ОС Windows скопировала многие возможности Mac.

MB (**M**ega**b**yte, Мбайт) — мегабайт, единица измерения количества информации, равная 1000 Кбайт. Альтернативой является предложенная IEC двоичная единица **MiB** (MibiByte), 1 MiB = 1024 KibiByte.

Mbit (Megabit, Мбит) — мегабит, единица измерения, равная 1024 Кбит.

MBps (**M**ega**B**ytes **p**er second) — мегабайт в секунду (Мбайт/с) — мера скорости передачи данных.

MDRAM (Multibank **DRAM**) — мультибанковая DRAM.

Memory refresh — регенерация памяти.

Mflops (**M**ega**flops**) — мера производительности ЭВМ, 1 миллион операций с плавающей точкой (запятой) в секунду.

MHz (**M**ega**h**er**z**, МГц) — мегагерц, мера частоты в 1 миллион циклов в секунду (1/с).

Microcode (микрокод, микрокоманда) — система команд нижнего (аппаратурного) уровня, которая напрямую управляет микропроцессором. Отдельная машинная команда (инструкция) обычно транслируется в несколько микрокоманд. Обычно микрокоманды встроены в ПЗУ и не могут быть перепрограммированы.

Micron (μm, микрон, микрометр, мкм) — единица измерения, равная одной миллионной доле метра.

Microsecond (μs, микросекунда, мкс) — одна миллионная доля секунды (0,000001 с).

Millisecond (ms, миллисекунда, мс) — одна тысячная секунды (0,001 с).

MIPS (**M**illions of **I**nstructions **P**er **S**econd) — миллион операций в секунду, мера производительности, используемая для сопоставления различных процессоров.

Moore's Law (закон Мура). Гордон Мур в 1965 г. за 3 года до основания Intel Corporation (где он был соучредителем) высказал знаменитое предсказание о том, что количество транзисторов в микросхемах будет удваиваться каждые 18 месяцев.

NaN (или NAN — Not-a-Number, «не-число») — значение числа с ПЗ в математических сопроцессорах, означающее, что предыдущая математическая операция завершилась неопределенным результатом. К операциям, приводящим к появлению NaN в результате, относятся: деление на ноль; вычисление квадратного корня из отрицательного числа. Различают четыре вида NaN: ±SNaN — «сигнализирующие не-числа» (signaling NaN), вызывающие прерывание, если с ними пытаются выполнять арифметические действия (порядок 111...111, мантисса 1,0xxx...xxx, ненулевая); ±QNaN — «тихие не-числа» (quiet NaN) не вызывают исключений при арифметических операциях (порядок 111...111, мантисса 1,1xxx...xxx, ненулевая).

Nanometre (от *греч.* nanos — карлик, nm, νm, нанометр, нм) — одна миллиардная доля метра (10^{-9} м).

Nanosecond (ns, νs, наносекунда, нс) — одна миллиардная доля секунды (0,000000001 с). Свет проходит около 8" за 1 нс.

NIC (Network Interface Card) — сетевая интерфейсная карта.

NorthBridge («Северный мост») — термин, принятый среди изготовителей chipset (наборов микросхем), обозначающий системный контроллер, в который входит контроллер системной шины, шин AGP и PCI, памяти и кэш-памяти (для наборов под обычный Pentium). Как правило, это одна микросхема и именно по ней называется весь набор.

NX bit (**N**o e**X**ecute) — технология, используемая в ЦП AMD для того, чтобы изолировать область памяти, в которой не могут находиться команды исполняемой программы. Если секция памяти помечена атрибутом NX, это значит, что она используется только для размещения данных. Фирма Intel обозначает данную возможность как **XD bit** (eXecute Disable), однако как XD (Intel), так и NX (AMD) выполняют одинаковые задачи и различаются только в названиях.

OTROM (**O**ne **T**ime **P**rogrammable **ROM**) — однократно программируемая ROM.

Out-of-order execution — выполнение команд с изменением естественного порядка («вне очереди», «внеочередное выполнение») — один

из методов динамического выполнения, обеспечивающий распараллеливание вычислений.

PCB (**P**rinted **C**ircuit **B**oard) — печатная схема (плата), на которую нанесены слои металлических проводников, соединяющих установленные на ней микросхемы.

PCI-bus — разработанная Intel шина, которая предназначена для поддержки высокоскоростного 32-разрядного обмена данными между устройствами, памятью и процессором.

Peripheral — внешнее устройство, такое, как НГМД, НЖМД, стример, принтер или модем.

Pipeline Burst — конвейерная пакетная (тип кэш-памяти).

Pixel (**Pi**cture **E**lement) — пиксель (точка экрана монитора в графическом режиме).

POP — ассемблерная команда считывания из стека.

Predication (предикация) — технология «отмеченных команд» для устранения потерь производительности из-за неправильно предсказанных переходов и необходимости пропуска участков кода после ветвлений. Когда процессор встречает «отмеченное» ветвление в процессе выполнения программы, он начинает одновременно выполнять все ветви. После того как будет определена «истинная» ветвь, процессор сохраняет необходимые результаты и сбрасывает остальные.

Processor Performance Rating (P-рейтинг) используется для оценки производительности процессоров (обозначение P1000+, например, символизирует производительность, эквивалентную Pentium с частотой 1000 МГц).

Programmable ROM — программируемая ROM.

Protected Mode — защищенный режим (в МП). Система адресации памяти, поддерживающая 32-битовые команды процессора. Память разделена между несколькими программами, которые работают одновременно и размещены внутри выделенных разделов.

PS/2 — серия персональных компьютеров IBM, выпущенная в 1987 г. для замещения PC (XT/AT). Впервые введены НГМД на 3,5", графический адаптер VGA и шина Micro Channel (MCA), которая в дальнейшем была заменена на PCI.

PUSH — ассемблерная команда записи в стек.

RAM (**R**andom **A**ccess **M**emory) — оперативное запоминающее устройство (с произвольным доступом).

RAS (**R**ow **A**ddress **S**trobe) — строб адреса строки в микросхеме DRAM или же сигнал в памяти.

RDRAM (**R**ambus **DRAM**) — одна из разновидностей DRAM.

Real — тип чисел, представляемый в памяти ЭВМ в формате с плавающей запятой/точкой, в виде «мантисса—порядок» (действительный, вещественный, `FLOAT`).

Real Mode — реальный режим (в МП).

RegRen (**reg**ister **ren**ame) — переименование, или ротация регистров, метод повышения производительности процессора, впервые введенный

в архитектуре P6. Вместо пересылки данных между регистрами осуществляется их переназначение, в результате этого команда (или микро/макрооперация) сразу получает доступ к необходимым данным.

Ret Far — дальний возврат.

Retirement («отставка») — завершение макро- или микроопераций в процессоре и освобождение ресурсов после исполнения, сопровождающееся окончательной записью результатов в регистровый файл процессора.

RISC (Reduced Instruction Set Computer) — архитектура, характеризующаяся ограниченным набором команд фиксированного формата. RISC-процессоры обычно реализуют около 100 команд, имеющих фиксированный формат длиной 4 байта. Обычно в RISC-процессорах все команды обработки данных выполняются только с регистровой или непосредственной адресацией.

Rmax (R_{max}) — производительность суперЭВМ, оцененная на основе теста Linpack Benchmark.

ROM (Read Only Memory) — постоянное запоминающее устройство (только для чтения).

Rpeak (R_{peak}) — теоретическая мера производительности суперкомпьютеров, оцениваемая путем суммирования производительности процессоров системы. Для примера, теоретическая пиковая производительность (R_{peak}) однопроцессорной системы на базе процессора Intel Pentium Pro с частотой 200 МГц не превышает величины 62 Мфлопс.

RS-232 порт (RS-232-C serial port) — стандарт (протокол, формат) соединения ЭВМ с последовательными внешними устройствами (модемы, некоторые принтеры, текстовые терминалы).

RTC (Real Time Clock) — часы реального времени (один из блоков информации, хранимый в CMOS RAM).

Saturation arithmetic — SARITH («арифметика насыщения») — версия арифметики, в которой результат операций типа перемножения или суммирования ограничен интервалом между максимальным и минимальным значениями.

SCSI (Small Computer System Interface) — интерфейс малых компьютерных систем.

SDRAM (Synchronous DRAM) — синхронная DRAM, вид памяти, который имеет преимущества только при последовательной выборке данных из памяти. Но при последовательной выборке (или потоке, конвейере — burst) чтение/запись выполняются в 2 раза быстрее, чем для EDO DRAM. На микросхемах SDRAM указывается время доступа в наносекундах к данным при последовательной выборке.

Set-associative cache — частично ассоциативная кэш-память.

SIMD (Single Instruction Multiple Data — Одна команда с множеством данных) — основный принцип построения MMX-команд для микропроцессоров. Эти команды в качестве операндов используют регистры сопроцессора, предназначенные для хранения операндов в 80-разрядной сетке.

SIMM (Single In-line Memory Module) — односторонний модуль памяти. Имеет 72 вывода с каждой стороны, но пары выводов с одной и другой стороны замкнуты между собой, поэтому они считаются односторонними. Собственно память, размещенная на модуле, может быть как FPM, так и EDO. Память в SIMM имеет разрядность 32 (с четностью — 36) бита и может использоваться в компьютерах с процессорами Pentium только парами.

SMP (Symmetric Multi-Processing) — симметричная многопроцессорная обработка; метод, позволяющий более чем одному процессору распределять между собой вычислительную нагрузку.

SMT (Surface Mounting Technology) — технология поверхностного монтажа.

SouthBridge («Южный мост») — обозначение периферийного контроллера в chipset (наборе микросхем), включающего обычно контроллер EIDE, клавиатуры, моста PCI-to-PCI, последовательных/параллельного портов, шины USB и других подобных устройств.

SP (Stack pointer) — указатель стека (один из регистров-указателей в МП).

Speculative loading — опережающее считывание данных из ОП в регистры процессора. Компиляторы архитектуры IA-64 должны просматривать исходный код с целью поиска команд, использующих данные из памяти. Найдя такую команду, они добавляют пару команд — предварительное чтение (speculative loading) на достаточно большом расстоянии перед командой, использующей данные, и проверку считывания (speculative check) непосредственно перед командой, использующей данные. Во время выполнения программы первая из команд загружает данные в регистры до того, как они понадобятся процессору. Вторая команда проверяет, успешно ли произошло чтение, перед тем, как разрешить ЦП использовать эти данные. Опережающее чтение позволяет уменьшить потери производительности из-за задержек при доступе к памяти, а также повысить параллелизм.

SPP (Standard Parallel Port) — стандартный параллельный порт, классический принтерный интерфейс, называемый, как правило, Centronics, по имени давно ликвидированной фирмы, разработавшей этот интерфейс. Интерфейс позволяет передавать данные по байту со скоростью до 80 Кбайт/с. При необходимости приема данных можно использовать четыре линии сигнала от принтера (обрыв бумаги, буфер принтера полон и т. д.).

SRAM (Static RAM) — статическая RAM.

SSE (система команд Streaming SIMD Extensions) — расширение набора команд x86. Предложена Intel в 1999 г. для процессора Pentium III как ответ на выпущенную годом ранее архитектуру AMD 3DNow!. SSE вводит 80 дополнительных команд, которые имели кодовое наименование KNI (Katmai New Instructions). AMD включила поддержку SSE в свои процессоры, начиная с Athlon XP.

SSE2 — развитие команд SSE (иногда называемых теперь «SSE1»), введенное в ЦП Pentium IV. SSE2 содержит новые математические операции над числами ПЗ двойной точности (64-разряда) и над 8/16/32-разрядными числами ФЗ, которые все выполняются в массиве 128-разрядных регистров XMM (ранее появившихся в SSE).

SSE3 — кодовое наименование Prescott New Instructions (PNI), развитие SSE2 путем включения команд цифровой обработки сигналов (DSP) и команд управления подпроцессами («нитями», thread).

SSE4 — прогнозируемое обновление SSE3 путем добавления ряда команд над целыми числами и пр.

SSSE3 — обновление системы команд SSE3 путем добавления 16 новых команд (opcodes), в том числе — перестановка битов в слове, перемножение 16-разрядных целых чисел с округлением результата и операции внутри слова.

Stack — стековая память.

Star processors — семейство процессоров AMD архитектуры K10, наименования ядер которых выбраны по именам звезд — Agena (β Центавра), Deneb (α Лебедя) Heka (λ Ориона), Kuma (ν Дракона), Propus (η Близнецов), Rana (δ Эридана), Regor, (γ Паруса), Spica (α Девы) и т. д.

Synchronous Burst — синхронная пакетная (тип кэш-памяти).

Synchronous DRAM (SDRAM) — вид памяти, который имеет преимущества только при последовательной выборке данных из памяти. Но при последовательной выборке (или потоке, конвейере — burst) чтение/запись выполняются в 2 раза быстрее, чем для EDO DRAM. На микросхемах SDRAM указывается время доступа в наносекундах к данным при последовательной выборке. Реально же цифры на корпусах микросхем синхронной памяти фактически сообщают максимальную тактовую частоту системной шины, на которой данная память может работать. SDRAM выпускается сейчас только в 168-выводных 64-разрядных модулях DIMM. В отличие от обычных модулей SIMM эти могут устанавливаться на системной плате поодиночке. В соответствии с JEDEC-стандартом на модуле DIMM должна быть установлена специальная микросхема SPD устройства. Некоторые современные системные платы, например фирмы Intel на наборе 440LX, не запускаются, если установлена планка памяти без SPD. Микросхемы SDRAM также широко используются в качестве локальной памяти видеокарт.

Target Buffer — буфер кэширования блоков.

TB (Terabyte, терабайт) — единица измерения емкости памяти, содержащая 1000 Гбайт. В качестве альтернативной бинарной единицы IEC предложила в 1998 г. TiB (TibiByte), 1 TiB = 1024 GiB (GibiByte).

TDP (от *англ.* thermal design power) — величина, показывающая, на отвод какой тепловоймощности должна быть рассчитана система охлаждения процессора или другого полупроводникового прибора. К примеру, если система охлаждения процессора рассчитана на TDP 30 Вт,

она должна быть в состоянии отвести 30 Вт тепла при некоторых заданных условиях.

TLB (сокращение от Translation Lookaside Buffer) — буфер быстрого преобразования адреса (БПА), представляющий собой специальную кэш-память; используется для ускорения страничного преобразования.

TOP 500 (500 наиболее производительных ЭВМ мира). Для получения достоверной информации в исследовании и определении лучших суперкомпьютеров мира был начат проект Top 500. Первая публикация официальной версии Top 500 состоялась 5 ноября 1998 г., и обновление осуществляется дважды в год. Рейтинги устанавливаются в зависимости от результата (R_{max}), показанного системой при проведении теста Linpack и теоретической пиковой производительности R_{peak}.

Torrenza — объявленная AMD инициатива, направленная на поддержку интеграции специализированных сопроцессоров в вычислительные машины и системы, базирующиеся на процессорах AMD Opteron. Torrenza не связана с какими-либо конкретными продуктами или технологиями, выдвигая только концепцию интеграции сопроцессорных устройств в системы путем сопряжения через порты HyperTransport с ЦП Opteron и через интерфейс PCI Express с другими устройствами. Инициатива имеет целью усилить техническую и технологическую поддержку внешних разработчиков сопроцессорных устройств, снизить стоимость использования интерфейсов HyperTransport для этих устройств и повысить общую эффективность интегрированных систем.

VLIW (Very Large Instruction Word) — архитектура, использующая очень длинные команды (до 128 бит и более), отдельные поля которых содержат коды, обеспечивающие выполнение различных операций параллельно в нескольких исполнительных устройствах, входящих в структуру микропроцессора. При трансляции программ, написанных на языке высокого уровня, соответствующий компилятор производит формирование «длинных» VLIW-команд, каждая из которых вызывает реализацию процессором целой процедуры или группы операций.

Volume — том, или логический носитель данных, содержащий множество файлов. В случае НЖМД обычно том размещается в разделе диска, форматируется для поддержки файловой системы (FAT или NTFS) и ему назначается идентификатор (буква — `C:`, `D:` и пр.).

VTOC (Volume Table of Contents) — таблицы содержания тома (системная часть CD ROM).

WORM (Write Once Read Many) — память с одноразовой записью и многоразовым чтением (на компакт-дисках).

Write Back (обратная запись) — термин применяется при описании устройств кэш-памяти. Если установлен режим обратной записи, то в случае изменения данных, находящихся в кэш-памяти, процессор меняет их только в кэше, но не в основной памяти. Только при замене одной области данных в кэше на другую процессор сохраняет данные из кэш-памяти в основную память. Реально это означает, что данные кэ-

шируются на запись и на чтение, и именно такой режим применяется сейчас в подавляющем большинстве компьютеров.

Write Combining — объединенная запись, термин применяется при описании устройств кэш-памяти и означает накопление записываемой информации в кэш-памяти с последующим «выстреливанием» готового пакета данных на шину.

Write Through (сквозная запись) — термин применяется при описании устройств кэш-памяти. Если установлен режим сквозной записи, то в случае изменения данных, находящихся в кэш-памяти, процессор одновременно меняет их как в кэше, так и в основной памяти.

x86-64 — 64-битовая архитектура микропроцессоров и соответствующий набор команд (надмножество команд Intel x86, которые также здесь поддерживаются). Введена AMD в процессорах AMD K8 и затем переименована в AMD64. Это первый случай, когда не Intel, а другая компания предложила существенные дополнения к IA-32. Intel была вынуждена «догонять», срочно вводя новые процессоры семейства NetBurst (первоначально названные «IA-32e»). Система x86-64 в дальнейшем принята Intel под именами EM64T, IA-32e или Intel 64 (почти точная копия AMD64). Оба расширения (AMD и Intel) обратно совместимы с кодом на 32 бита без потери эффективности. Например, «Руководства разработчиков», выпускаемые Intel для архитектуры IA-32, относятся также и к IA-32e. Наименования «x86-64» или «x64» иногда используются как нейтральные для общей ссылки на эти две почти идентичные (AMD и Intel) реализации.

x87 — расширение набора команд (x86) процессоров Intel путем ввода «математических» (например, с плавающей запятой) операций. Первоначально появляются как команды сопроцессоров, название которых заканчивается на «x87». Как и другие расширения базового набора x86, не обязательны для нормального функционирования процессора, однако являются аппаратным обеспечением математических расчетов и существенно повышают производительность, поскольку задачи решаются меньшим числом команд. В частности, x87 содержит команды вычисления синуса (sin) и косинуса (cos).

XMS (eXtended Memory Specification) — спецификация расширенной памяти.

ZIF (zero insertion force, нулевое усилие включения) — принцип конструирования разъемов (в основном, интерфейсов микропроцессоров), при котором не требуется усилий (обычно достаточно веса микросхемы), для помещения микросборки в контактное гнездо. Перед включением микросхемы вспомогательные рычаги отводят контакты гнезда от их обычного положения, в связи с чем те не оказывают давления на вводимые штырьки разъема, а затем контакты освобождаются и приходят в соприкосновение со штырьками. ZIF-разъемы более дороги, чем обычные конструкции, а также требуют больше свободного места на плате.

Набор команд x86

В табл. П3.1 приводится практически полный набор команд x86 (процессоров 8086—8088), большинство из которых функционируют в 32-разрядных процессорах, используя 32-битовые регистры (EAX, EBX и пр.) и величины, вместо их 16-битовых (AX, BX и пр.) аналогов.

Таблица П3.1. **Некоторые из исходных команд набора 8086—8088**

Команда	Расшифровка	Действие
AAA	ASCII adjust AL after addition	Выравнивание символов после сложения
AAD	ASCII adjust AX before division	Выравнивание символов перед делением
AAM	ASCII adjust AX after multiplication	Выравнивание символов после перемножения
AAS	ASCII adjust AL after subtraction	Выравнивание символов после вычитания
ADC	Add with carry	Сложение с переносом единицы
ADD	Add	Сложение
AND	Logical AND	Логическая операция И
CALL	Call procedure	Вызов процедуры
CBW	Convert byte to word	Преобразование байта в слово
CLC	Clear carry flag	Очистка флага переноса
CLD	Clear direction flag	Очистка флага направления?
CLI	Clear interrupt flag	Очистка флага прерывания
CMC	Complement carry flag	Дополнение флага переноса?
CMP	Compare operands	Сравнение операндов
CMPSB	Compare bytes in memory	Сравнение операндов в памяти
CMPSW	Compare words	Сравнение слов
CWD	Convert word to doubleword	Преобразовать слово в двойное слово

Продолжение табл. П3.1

Команда	Расшифровка	Действие
DAA	Decimal adjust AL after addition	Выравнивание двоично-десятичного после сложения
DAS	Decimal adjust AL after subtraction	Выравнивание двоично-десятичного после вычитания
DEC	Decrement by 1	Уменьшение на 1
DIV	Unsigned divide	Деление чисел без знака
HLT	Enter halt state	Остановить процессор
IDIV	Signed divide	Деление чисел со знаком
IMUL	Signed multiply	Перемножение чисел со знаком
IN	Input from port	Ввод данных из порта
INC	Increment by 1	Увеличение на 1
INT	Call to interrupt	Вызов по прерыванию
INTO	Call to interrupt if overflow	Вызов по переполнению
IRET	Return from interrupt	Возврат из прерывания
Jxx (JE, JG, JZ и т. д.)	Jump if condition	Условный переход (JE — переход, если «равно», JG — переход, если больше, JZ — переход, если «0», и пр.)
JMP	Jump	Безусловный переход
LAHF	Load flags into AH register	Загрузить флаги в регистр AH
LDS	Load pointer using DS	Загрузить указатель, используя сегмент данных (DS)
LODSB	Load byte	Загрузить (в регистр) байт
LODSW	Load word	Загрузить слово
LOOP/LOOPx	Loop control	Управление циклом «ДО/FOR» (LOOPE — повторить, если «равно», LOOPNE — повторить, если «не равно», LOOPZ — повторить, если «0», и пр.)
MOV	Move	Переслать
MOVSB	Move byte from string to string	Переслать байт из строки в строку
MOVSW	Move word from string to string	Переслать слово из строки в строку

Продолжение табл. П3.1

Команда	Расшифровка	Действие
MUL	Unsigned multiply	Перемножение чисел без знака
NEG	Two's complement negation	Дополнение к отрицанию
NOP	No operation	Нет операции
NOT	Negate the operand, logical NOT	Инвертировать операнд (логическое отрицание)
OR	Logical OR	Логическое ИЛИ
OUT	Output to port	Вывести в порт
POP	Pop data from stack	Принять («поднять») данные из стека
POPF	Pop data from flags register	Принять данные из регистра флагов
PUSH	Push data onto stack	Поместить («опустить») данные в стек
PUSHF	Push flags onto stack	Поместить флаги в регистр флагов
RCL	Rotate left (with carry)	Циклический сдвиг влево (с переносом)
RCR	Rotate right (with carry)	Циклический сдвиг вправо (с переносом)
REPxx	Repeat	Управление циклом «ПОКА/WHILE» (REPE — повторить, если «равно», REPNE — повторить, если «не равно», и т. д.)
RET/RETN. RETF	Return from procedure	Возврат из процедуры
ROL	Rotate left	Циклический сдвиг влево
ROR	Rotate right	Циклический сдвиг вправо
SAHF	Store AH into flags	Поместить регистр AH во флаги
SAL	Shift Arithmetically left (multiply)	Арифметический сдвиг влево (перемножение)
SAR	Shift Arithmetically right (signed divide)	Арифметический сдвиг вправо (деление со знаком)
SBB	Subtraction with borrow	Вычитание с займом единицы
SCASB	Compare byte string	Сравнить битовые строки
SCASW	Compare word string	Сравнить строки слов
SHL	Shift left (multiply)	Сдвиг влево (перемножение)

Окончание табл. П3.1

Команда	Расшифровка	Действие
SHR	Shift right (unsigned divide)	Сдвиг вправо (беззнаковое деление)
STC	Set carry flag	Установить флаг переноса
STD	Set direction flag	Установить флаг направления
STI	Set interrupt flag	Установить флаг прерывания
STOSB	Store byte in string	Поместить байт в строку
STOSW	Store word in string	Поместить слово в строку
SUB	Subtraction	Вычитание
TEST	Logical compare (AND)	Логическое сравнение (И)
WAIT	Wait until not busy	Ожидать, пока линия BUSY# не занята (используется с блоком ПЗ)
XCHG	Exchange data	Обмен данными
XLAT	Table look-up translation	Использование таблицы транслитерации
XOR	Exclusive OR	Исключающее ИЛИ

MMX-команды исполняются в том же режиме процессора, что и команды с плавающей запятой. Поэтому при исполнении всех MMX-команд (кроме EMMS) «портится» слово состояния регистров с плавающей запятой. В табл. П3.2 приведены примеры некоторых команд (здесь MP — MMX-регистр; П32, П64 — 32- и 64-разрядные слова в ОП; НО — непосредственный операнд; P32 — целочисленный регистр).

Полный список MMX-инструкций:

EMMS, MOVD, MODQ, PACKSSDW, PACKSSWB, PACKUSWB, PADDB, PADDD, PADDSB, PADDSW, PADDUSB, PADDUSW, PADDW, PAND, PANDN, PCMPEQB, PCMPEQD, PCMPEQW, PCMPGTB, PCMPGTD, PCMPGTW, PMADDWD, PMULHW, PMULLW, POR, PSLLD, PSLLQ, PSLLW, PSRAD, PSRAW, PSRLD, PSRLQ, PSRLW, PSUBB, PSUBD, PSUBSB, PSUBSW, PSUBUSB, PSUBUSW, PSUBW, PUNPCKHBW, PUNPCKHDQ, PUNPCKHWD, PUNPCKLBW, PUNPCKLDQ, PUNPCKLWD, PXOR.

Все команды MMX оперируют с двумя операндами — источник и приемник. В обозначениях команды правый операнд является источником, а левый — приемником.

Операнд-приемник может также быть и вторым операндом-источником. Команды записывают в приемник результат.

Операнд-источник для всех команд MMX (кроме команд пересылок) может располагаться либо в памяти, либо в регистре MMX. Операнд-приемник должен располагаться в регистре MMX.

Для команд пересылок операндами могут также выступать целочисленные РОН или ячейки памяти. Все 57 MMX-команд разделены на следующие классы — арифметические, сравнения, преобразования, логические, сдвига, пересылки и смены состояний. Основные форматы представлены в табл. П3.2 (не представлены команды сравнения, логические, сдвига и смены состояний).

Таблица П3.2. **Некоторые команды технологии MMX**

Мнемоника	Действие
Арифметические	
PADD[B,W,D]	Сложение с округлением [byte, word, doubleword]
PADDS[B,W]	Сложить знаковые с насыщением [byte, word]
ADDUS[B,W]	Сложить беззнаковые с насыщением [byte, word]
PSUB[B,W,D]	Вычесть с округлением [byte, word, doubleword]
PSUBS[B,W]	Вычесть знаковые с насыщением [byte, word]
PSUBUS[B,W	Вычесть беззнаковые с насыщением [byte, word]
PMULHW	Упакованное умножение, старшее для слов
PMULLW	Упакованное умножение, младшее для слов
PMADDWD	Упакованное умножение слов и сложение результирующих пар
Преобразования	
PACKUSWB	Упаковка слов в байты (беззнаковая с насыщением)
PACKSS[WB,DW]	Упаковка слов в байт, двойных слов в слова (знаковая с насыщением)
PUNPCKH [BW,WD,DQ]	Распаковка с расслоением, старшая часть [bytes, words, doublewords]
PUNPCKL [BW,WD,DQ]	Распаковка с расслоением, младшая часть [bytes, words, doublewords]
Пересылка	
MOV[D,Q]	Пересылка [doubleword, quadword] в или из MMX-регистра

Технология 3DNow! предлагает для выполнения векторных MMX и 3DNow! операций несколько устройств (пара умножителей, сумматоров и т. д.). Все команды 3DNow! имеют длительность исполнения 2 такта и полностью конвейеризированы.

Полный список 3DNow!-инструкций:

- ***введены с процессорами K6-2*** — `FEMMS`, `PAVGUSB`, `PF2ID`, `PFACC`, `PFADD`, `PFCMPEQ`, `PFCMPGE`, `PFCMPGT`, `PFMAX`, `PFMIN`, `PFMUL`, `PFRCP`, `PFRCPIT1`, `PFRCPIT2`, `PFRSQIT1`, `PFRSQRT`, `PFSUB`, `PFSUBR`, `PI2FD`, `PMULHRW`, `PREFETCH`, `PREFETCHW`;
- ***введены с процессорами Athlon*** — `PF2IW`, `PFNACC`, `PFPNACC`, `PI2FW`, `PSWAPD`.

Формат команд 3DNow!:

mnemonic	mmreg1	mmreg2/mem64

Операнд-приемник должен быть MMX-регистром (`MM0`—`MM7`). Операнд-источник может быть как MMX-регистром, так и операндом в памяти. Используемые сокращения в мнемонике команд: `P` — `Packed` (упакованный), `F` — `Float` (вещественный), `I` — `Integer` (целочисленный). Список команд представлен в табл. П3.3.

Таблица П3.3. **Система команд 3DNow!**

Команда	Действие
`PAVGUSB`	Упакованное побайтное целочисленное усреднение
`PFADD`	Упакованное вещественное сложение
`PFSUB`	Упакованное вещественное вычитание
`PFSUBR`	Упакованное вещественное обратное вычитание
`PFACC`	Упакованное вещественное сложение с накоплением
`PFCMPGE`	Упакованное вещественное сравнение, >=
`PFCMPGT`	Упакованное вещественное сравнение, >
`PFCMPEQ`	Упакованное вещественное сравнение, =
`PFMIN`	Упакованный вещественный минимум
`PFMAX`	Упакованный вещественный максимум
`PI2FD`	Преобразование 32-битовых целых в вещественные числа
`PF2ID`	Преобразование вещественных чисел в 32-битовые целые числа

Окончание табл. П3.3

Команда	Действие
PFRCP	Приближенное обратное значение
PFRSQRT	Приближенное обратное значение квадратного корня
PFMUL	Упакованное вещественное умножение
PFRCPIT1	Первый шаг нахождения обратного значения
PFRSQIT1	Первый шаг нахождения обратного значения квадратного корня
PFRCPIT2	Второй шаг нахождения обратного значения или обратного значения квадратного корня
PMULHRW	Упакованное умножение целочисленных слов с округлением
FEMMS	Быстрая смена MMX состояния и FPU x87
PREFETCH	Предвыборка как минимум 32 байт в кэш данных L1

SSE. Система команд Streaming SIMD Extensions — набор команд, которые обрабатывают только значения с ПЗ, подобно 3DNow!.

SSE вводит новые 128-битовые регистры, известные как XMM0 — XMM7. Каждый регистр позволяет обработать четыре упакованных 32-битовых числа ПЗ одинарной точности.

Полный список SSE-инструкций (см. также табл. П3.4):

- ***операции с ПЗ*** — ADDPS, ADDSS, ANDNPS, ANDPS, CMPPS, CMPSS, COMISS, CVTPI2PS, CVTPS2PI, CVTSI2SS, CVTSS2SI, CVTTPS2PI, CVTTSS2SI, DIVPS, DIVSS, LDMXCSR, MAXPS, MAXSS, MINPS, MINSS, MOVAPS, MOVHLPS, MOVHPS, MOVLHPS, MOVLPS, MOVMSKPS, MOVNTPS, MOVSS, MOVUPS, MULPS, MULSS, ORPS, RCPPS, RCPSS, RSQRTPS, RSQRTSS, SHUFPS, SQRTPS, SQRTSS, STMXCSR, SUBPS, SUBSS, UCOMISS, UNPCKHPS, UNPCKLPS, XORPS;
- ***операции с ФЗ*** — PAVGB, PAVGW, PEXTRW, PINSRW, PMAXSW, PMAXUB, PMINSW, PMINUB, PMOVMSKB, PSADBW, PSHUFW.

SSE2 добавляет команды обработки чисел ПЗ двойной точности (64 бит) и 8/16/32-битовых целых чисел с использованием файла тех же 128-битовых XMM-регистров.

Полный список SSE2-инструкций:

- ***операции с ПЗ*** — ADDPD, ADDSD, ANDNPD, ANDPD, CMPPD, CMPSD, COMISD, CVTDQ2PD, CVTDQ2PS, CVTPD2DQ, CVTPD2PI, CVTPD2PS, CVTPI2PD, CVTPS2DQ, CVTPS2PD, CVTSD2SI,

Таблица П3.4. **Некоторые команды набора SSE**

Команда	Расшифровка	Действие
ADDPS	Packed Single-FP ADD	Суммирует два векторных операнда (по четыре 32-разрядных числа ПЗ)
ADDSS	Scalar Single-FP ADD	Суммирует числа (32 разряда с ПЗ), расположенные в младших четвертях регистров
ANDNPS	Bitwise Logical AND NOT	Побитовая операция AND над операндами, один их которых предварительно обращен (NOT)
ANDPS	Bitwise Logical AND	Побитовая операция AND
CMPccPS	Packed Single-FP Compare	Сравнивает векторы (по четыре 32-разрядных числа ПЗ)
CMPccSS	Scalar Single-FP Compare	Сравнивает числа (32 разряда с ПЗ), расположенные в младших четвертях регистров
COMISS	Scalar Compare and Set EFLAGS	Сравнивает числа (32 разряда с ПЗ), расположенные в младших четвертях регистров, и соответственно устанавливает флаги процессора. Вырабатывает сигнал исключения, если один из операндов равен SNaN или QNaN
CVTPI2PS	Packed Integer to Floating-Point Conversion	Конвертирует пару 32-битовых целых чисел в 32-битовые с ПЗ, результат записывается в младшие 64 разряда регистра результата
CVTPS2PI	Packed Floating-Point to Integer Conversion	Конвертирует пару 32-битовых чисел с ПЗ в 32-битовые с ФЗ
CVTSI2SS	Scalar Integer to Floating-Point Conversion	Конвертирует число (32 разряда с ФЗ) в 32-битовое ПЗ, записывает его в младшую четверть регистра результата
CVTSS2SI	Scalar Floating-Point to Integer Conversion	Конвертирует число (32 разряда с ПЗ) в 32-битовое ФЗ
FXRSTOR	Restore FP, MMX and SSE States	Восстанавливает состояние процессора (FP, MMX или SSE) по данным в памяти, предварительно сохраненных командой FXSAVE
LDMXCSR	Load MXCSR	Считывает из памяти 32 бита и заносит их в MXCSR (регистр статуса SSE)
MAXPS	Packed Single-FP Maximum	Сравниваются два вектора по 32 бита (ПЗ) и максимумы заносятся в соответствующие четверти регистра результата
MAXSS	Scalar Single-FP Maximum	Определяется максимум из чисел (32 разряда с ПЗ), расположенных в младших четвертях регистров

Окончание табл. П3.4

Команда	Расшифровка	Действие
MINPS	Packed Single-FP Minimum	Сравниваются два вектора по 32 бита (ПЗ) и минимумы заносятся в соответствующие четверти регистра результата
MINSS	Scalar Single-FP Maximum	Определяется минимум из чисел (32 разряда с ПЗ), расположенных в младших четвертях регистров
MOVHLPS	Move High to Low	Помещает 8 байт из верхней половины исходного регистра в младшую половину регистра результата
MOVHPS	Move High	Перемещает 8 байт из памяти в верхнюю половину регистра
MOVLHPS	Move Low to High	Помещает 8 байт из нижней половины исходного регистра в старшую половину регистра результата
MOVLPS	Move Low	Перемещает 8 байт из памяти в нижнюю половину регистра
ORPS	Bitwise Logical OR	Побитовая операция OR
RCPPS	Packed Single-FP Reciprocal Approximation	Для каждого из четырех 32-битовых значений с ПЗ вычисляется приближенное значение обратной величины (с точностью 3/8192) и записывается в соответствующие четверти регистра результата
RCPSS	Scalar Single-FP Reciprocal Approximation	Вычисляется приближенное значение обратной величины для числа (ПЗ) в младшей четверти регистров
STMXCSR	Store MXCSR	Копирует регистр (регистр статуса SSE) в память
XORPS	Bitwise Logical XOR	Побитовая операция XOR

CVTSD2SS, CVTSI2SD, CVTSS2SD, CVTTPD2DQ, CVTTPD2PI, CVTPS2DQ, CVTTSD2SI, DIVPD, DIVSD, MAXPD, MAXSD, MINPD, MINSD, MOVAPD, MOVHPD, MOVLPD, MOVMSKPD, MOVSD, MOVUPD, MULPD, MULSD, ORPD, SHUFPD, SQRTPD, SQRTSD, SUBPD, SUBSD, UCOMISD, UNPCKHPD, UNPCKLPD, XORPD;

- *операции с ФЗ* — MOVDQ2Q, MOVDQA, MOVDQU, MOVQ2DQ, PADDQ, PMULUDQ, PSHUFHW, PSHUFLW, PSHUFD, PSLLDQ, PSRLDQ, PUNPCKHQDQ, PUNPCKLQDQ.

Однако дополнительной трудностью явилось то, что операционные системы должны были «уметь» обрабатывать команды этого нового набора, чтобы сохранять состояния их регистров. Intel предложила немного измененную версию защищенного режима,

названную улучшенным режимом (Enhanced mode), который допускает использование команд SSE (в то время как они блокируются в обычном защищенном режиме). ОС, которая «знает» о SSE, автоматически переключается в улучшенный режим, тогда как «ОС-незнайка» войдет в традиционный защищенный режим.

SSE3 — набор команд, также известный как Prescott New Instructions (PNI), является третьей итерацией набора команд SSE для IA-32 (табл. П3.5). Intel использует SSE3 с начала 2004 г. в ЦП Pentium IV Prescott. В апреле 2005 г. AMD также включает SSE3 в ЦП Athlon 64 (версия E — Venice и San Diego). SSE3 не требует дополнительных регистров.

Таблица П3.5. **Некоторые команды SSE3**

Команда	Расшифровка	Исходные данные	Результат
«Вертикальная» арифметика			
ADDSUBPD	Add-Subtract-Packed-Double	$P_1 = \{A_0 A_1\}$, $P_2 = \{B_0 B_1\}$	$P_3 = \{A_0 - B_0, A_1 + B_1\}$
ADDSUBPS	Add-Subtract-Packed-Single	$\{A_0 A_1 A_2 A_3\}$, $\{B_0 B_1 B_2 B_3\}$	$\{A_0 - B_0, A_1 + B_1, A_2 - B_2, A_3 + B_3\}$
«Горизонтальная» арифметика			
HADDPD	Horizontal-Add-Packed-Double	$\{A_0 A_1\}$, $\{B_0 B_1\}$	$\{B_0 + B_1, A_0 + A_1\}$
HADDPS	Horizontal-Add-Packed-Single	$\{A_0 A_1 A_2 A_3\}$, $\{B_0 B_1 B_2 B_3\}$	$\{B_0 + B_1, B_2 + B_3, A_0 + A_1, A_2 + A_3\}$
HSUBPD	Horizontal-Subtract-Packed-Double	$\{A_0 A_1\}$, $\{B_0 B_1\}$	$\{A_0 - A_1, B_0 - B_1\}$
HSUBPS	Horizontal-Subtract-Packed-Single	$\{A_0 A_1 A_2 A_3\}$, $\{B_0 B_1 B_2 B_3\}$	$\{A_0 - A_1, A_2 - A_3, B_0 - B_1, B_2 - B_3\}$
Прочие команды			
LDDQU		Загрузка 128 бит невыровненных данных из памяти в регистр XMM, с предотвращением пересечения границы строки кэша	

Окончание табл. П3.5

Команда	Расшифровка	Исходные данные	Результат
MOVDDUP, MOVSHDUP, MOVSLDUP		Используются при операциях над комплексными числами или при обработке звуковых сигналов	
FISTTP		FISTTP — преобразование вещественного числа в целое с сохранением целочисленного значения и округлением в сторону нуля	
MONITOR, MWAIT		Оптимизируют многопроцессные приложения (multi-threaded), увеличивают производительность процессоров в режиме Hyper-Threading	

SSSE3 — Supplemental Streaming SIMD Extension 3 (дополнительное расширение потоковых команд SIMD 3), название четвертой итерация для набора команды SSE (Intel считает это просто пересмотром SSE3). Эти команды также известны как TNI (Tejas New Instructions) или MNI (Merom New Instructions) по первым процессорам, которые должны были их поддерживать.

Представленный в микроархитектуре Intel Core, набор SSSE3 доступен в процессорах Xeon 5100 (серверы и рабочие станции) и Intel Core 2 (ноутбуки и настольные ПК).

SSSE3 содержит 16 новых команд сравнительно с SSE3 (Intel считает это как 32, так как каждая команда имеет две формы — для 64-битовых регистров MMX и 128-битовых XMM).

В табл. П3.6 функция `satsw (X)` (`saturate to signed word` — насыщение к слову со знаком) означает, что целое со знаком `X` преобразуется в `-32 768` (если оно меньше, чем -32 768) либо в `+32 767` (если оно больше, чем 32 767), и остается без изменения в противном случае. «Регистр» (P_i) относится к векторным регистрам MMX или XMM.

SSE4 — набор команд, используемых в процессорах с микроархитектурами Intel Core и AMD K10 (K8L). Система команд была объявлена 27 сентября 2007 г. на Форуме разработчиков Intel.

Intel SSE4 включает 54 команды. Подмножество, содержащее 47 команд, известное как SSE4.1, впервые появляется в ЦП Penryn. Дополнение — SSE4.2, включает еще 7 команд, впервые использованные в ядре Nehalem.

AMD также вводит ряд новых SSE-инструкций (с выпуском процессора Phenom), называя их SSE4a. Эти команды не под-

держиваются ЦП Intel-SSE4.1 и наоборот, процессоры AMD не поддерживают Intel-SSE4.1 (табл. П3.7).

Таблица П3.6. **Команды SSSE3**

Команда	Расшифровка	Действие
PSIGNB, PSIGNW, PSIGND	Упаковка со знаками (Packed Sign)	Добавляет знак «–» к элементам (W, B, D и пр.) в регистре, если соответствующие элементы другого регистра отрицательны
PABSB, PABSW, PABSD	Упаковка абсолютных величин (Packed Absolute Value)	Заполняет элементы регистра (W, B, D и пр.) соответствующими элементами другого регистра
PALIGNR	Упаковка с выравниваем вправо (Packed Align Right)	Соединяет содержание двух регистров, затем подстрока с длиной регистра сдвигается на величину, заданную непосредственным операндом
PSHUFB	Упаковка с перестановкой байтов (Packed Shuffle Bytes)	Входные регистры: $P_1 = \{A_0, A_1, A_2, \ldots\}$, $P_2 = \{B_0, B_1, B_2, \ldots\}$. Результат: $P_1 = \{A_{B_0}, A_{B_1}, A_{B_2}, \ldots\}$
PMULHRSW	Перемножение упакованных с округлением (Packed Multiply High with Round and Scale)	Рассматриваются 16-битовые слова регистров P_1 и P_2 как числа ФЗ 15 бит со знаком (между –1 и +1) и перемножаются с округлением
PMADDUBSW	Суммирование и перемножение (Multiply and Add Packed Signed and Unsigned Bytes)	Входные регистры: $P_1 = \{A_0, A_1, A_2, \ldots\}$, $P_2 = \{B_0, B_1, B_2, \ldots\}$. Результат: $\{\text{satsw}(A_0B_{0_i} + A_1B_{1_i}),$ $\{\text{satsw}(A_2B_{2_i} + A_3B_{3_i}), \ldots\}$
PHADDW/PHSUBW, PHADDD/PHSUBD	Горизонтальное сложение/вычитание (Packed Horizontal Add/Subtract — Words or Doublewords)	Входные регистры: $P_1 = \{A_0, A_1, A_2, \ldots\}$, $P_2 = \{B_0, B_1, B_2, \ldots\}$. Результат: $\{A_0 \pm A_1, A_2 \pm A_3, \ldots$ $\ldots B_0 \pm B_1, B_2 \pm B_3, \ldots\}$
PHADDSW/PHSUBSW	Горизонтальное сложение/вычитание с насыщением (Packed Horizontal Subtract and Saturate Words)	Подобна PHADDW/PHSUBW, но результат: $\{\text{satsw}(A_0 \pm A_1), \text{satsw}(A_2 \pm A_3), \ldots$ $\ldots \text{satsw}(B_0 - B_1), \ldots\}$

Таблица П3.7. **Некоторые команды SSE4**

Команда	Расшифровка	Действие
	Набор SSE4.1	
BLENDPD, BLENDPS, BLENDVPD, BLENDVPS, PBLENDVB, PBLENDW	Blend Packed Double and Single Precision Floating-Point Values	Условное копирование чисел ПЗ простой или двойной точности из одного места размещения в другое
DPPD, DPPS	Dot Product of Packed Double and Single Precision Floating-Point Value	Скалярное произведение упакованных чисел ПЗ обычной и двойной точности
EXTRACTPS, INSERTPS	Extract and Insert Packed Single Precision Floating-Point Value	Извлечение и вставка упакованных чисел ПЗ обычной точности
MPSADBW	Compute Multiple Packed Sums of Absolute Difference	Вычислить абсолютную разность нескольких упакованных сумм
PACKUSDW	Pack with Unsigned Saturation	Упаковать с насыщением без знака
PEXTRB, PEXTRD/PEXTRQ, PEXTRW	Extract Byte, Dword/Qword, and Word	Извлечь байт, сдвоенное/счетверенное слово и слово
PHMINPOSUW	Packed Horizontal Word Minimum	Горизонтальный минимум упакованных слов
PINSRB, PINSRD/PINSRQ	Insert Byte and Dword/Qword	Вставить байт или сдвоенное/счетверенное слово
PMAXSB, PMAXSD, PMAXUD, PMAXUW, PMINSB, PMINSD, PMINUD, PMINUW	Find Minimum and Maximum of Packed Signed, Unsigned, Dword and Word-length Integers	Найти минимум и максимум упакованных, со знаком или без целых чисел обычной и двойной длины
PMOVSX, PMOVZX	Packed Move with Sign and Zero Extend	Перемещение упакованных величин различных типов (слова в двойные слова и пр.)
PMULDQ, PMULLD	Multiply Packed Signed Dword Integers and Store Low Result	Перемножение целых двойной длины со знаком и запись младших разрядов
PTEST	Logical Compare	Логическое сравнение
ROUNDPD, ROUNDPS, ROUNDSD, ROUNDSS	Round Packed and ScalarDouble and Single Precision Floating-Point Values	Округление упакованных чисел ПЗ до целых значений
	Набор SSE4.2	
CRC32	Accumulate CRC32 Value	Накопление контрольной суммы
PCMPESTRI, PCMPISTRI	Packed Compare Explicit and Implicit Length Strings, Return Index	Сравнить упакованные строки равной или различной длины, выдать указатель
PCMPESTRM, PCMPISTRM	Packed Compare Explicit and Implicit Length Strings, Return Mask	Сравнить упакованные строки равной или различной длины, выдать маску

Окончание табл. П3.7

Команда	Расшифровка	Действие
PCMPEQQ, PCMPGTQ	Compare Packed Data For Equal or Greater Than	Сравнить упакованные данные на «равно» или «больше»
POPCNT	«Population Count» (Return the Count of Number of Bits Set to 1)	Подсчет количества битов (например, установленных в «1»)
Набор SSE4a		
LZCNT	Leading Zero Count	Бит-манипуляции. Подсчет ведущих нулей слова
POPCNT	Population count (count number of bits set to 1)	Бит-манипуляции. Подсчет нулей или единиц в слове
EXTRQ/INSERTQ	Combined mask-shift instructions	Команды комбинированного маскирования/сдвига
MOVNTSD/MOVNTSS	Scalar streaming store instructions	Команды потоковой записи данных

SSE5 (сокращение от Streaming SIMD Extensions, версия 5) — объявлен AMD 30 августа 2007 г., представляет собой набор дополнительных 128-разрядных команд для AMD64 для процессорного ядра Bulldozer, намечающегося к выходу в 2009 г.

SSE5 включает 170 команд (в том числе 46 основных команд), многие из которых предназначены для ускорения параллельной обработки. Некоторые из SSE5-команд являются трехадресными, но приспособленными к условиям RISC-процессоров, и их использование снижает среднее число команд на цикл для обработки кодов x86. Основные новые инструкции включают:

- команды смешанного перемножения с накоплением (FMACxx);
- команды целочисленного умножения с накоплением (IMAC, IMADC)
- команды перестановок (PPERM, PERMPx) и условных пересылок;
- команды управления точностью, округлением и преобразования данных.

Специалисты AMD полагают, что SSE5 существенно повысит эффективность, особенно в высокоскоростных вычислениях (High Performance Computing — HPC), мультимедийных приложениях и защите данных, включая 5-кратное повышение эффективности кодирования (на примере алгоритма DES) и не менее чем на 30 % повышение эффективности цифровой обработки сигналов видеопотока.

Оглавление

Максимов Николай Вениаминович
Партыка Татьяна Леонидовна
Попов Игорь Иванович

Архитектура ЭВМ и вычислительных систем

Учебное издание

Редактор *А. В. Волковицкая*
Корректор *А. В. Алешина*
Компьютерная верстка *И. В. Кондратьевой*
Оформление серии *П. Родькина*

Подписано в печать 20.01.2013. Формат 60×90/16.
Гарнитура «Таймс». Усл. печ. л. 32,0. Уч.-изд. л. 32,6.
Печать офсетная. Бумага офсетная. Тираж 1000 экз.
Заказ № 4207.

Издательство «ФОРУМ»
101990, Москва — Центр, Колпачный пер., д. 9а
Тел./факс: (495) 625-32-07, 625-52-43
E-mail: forum-knigi@mail.ru

Отдел продаж издательства «ФОРУМ»:

101990, Москва — Центр, Колпачный пер., д. 9а
Тел./факс: (495) 625-52-43
E-mail: forum-ir@mail.ru
www.forum-books.ru

Книги издательства «ФОРУМ»
вы также можете приобрести:

Отдел продаж «ИНФРА-М»
127282, Москва, ул. Полярная, д. 31в
Тел.: (495) 380-05-40 (доб. 252)
Факс: (495) 363-92-12

Отдел «Книга-почтой»
E-mail: podpiska@infra-m.ru;
books@infra-m.ru

Отпечатано с электронных носителей издательства.
ОАО "Тверской полиграфический комбинат", 170024, г. Тверь, пр-т Ленина, 5.
Телефон: (4822) 44-52-03, 44-50-34, Телефон/факс: (4822) 44-42-15
Home page - www.tverpk.ru Электронная почта (E-mail) - sales@tverpk.ru