МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ АГЕНТСТВО ПО ОБРАЗОВАНИЮ
ЕЛЕЦКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ И.А. БУНИНА

О.Н. Масина

Курс лекций по дисциплине "Архитектура компьютера"

Учебное пособие

Елец – 2007

УДК 002
ББК 22.18
М 31

Печатается по решению редакционно-издательского совета Елецкого государственного университета имени И.А. Бунина от 14.03.2007, протокол № 1

Рецензенты:
доктор физико-математических наук, профессор, ведущий научный сотрудник отдела надежности и устойчивости, академик Международной академии нелинейных наук О.В. Дружинина
(*Вычислительный центр им. А.А. Дородницына Российской академии наук*);
доктор физико-математических наук, профессор Ю.Н. Меренков
(*Елецкий государственный университет им. И.А. Бунина*)

Масина О.Н.

М 31 Курс лекций по дисциплине «Архитектура компьютера»: Учебное пособие. – 2-е изд., доп. и перераб. – Елец: ЕГУ им. И.А.Бунина, 2007. – 268 с.

В учебном пособии описана структурная организация персонального компьютера и работа его основных внутренних компонентов. Изложены информационно-логические основы построения ПК, принципы организации внутренней и внешней памятей компьютера, а также принципы работы внешних устройств электронно-вычислительных машин. Проанализированы современное состояние и тенденции развития ЭВМ. Описаны вопросы организации программы на языке ассемблер.

Издание предназначено для студентов физико-математических факультетов вузов, а также учителям школ и преподавателям технических училищ и колледжей.

УДК 002
ББК 22.18

© Елецкий государственный университет им. И.А. Бунина, 2007
© О.Н. Масина, 2007

Содержание

Введение 6
1. Структура и функции персонального компьютера 7
2. Состояние и тенденции развития ЭВМ 18
2.1. Этапы развития вычислительной техники 18
2.2. Классификация ЭВМ 24
2.2.1. Классификация ЭВМ по принципу действия 24
2.2.2. Классификация ЭВМ по назначению 25
2.2.3. Классификация ЭВМ по размерам и функциональным возможностям 26
2.2.4. Классификация ЭВМ по этапам создания (поколения ЭВМ) 29
2.3. Тенденции развития вычислительных систем 34
3. Информационно-логические основы построения персонального компьютера 37
3.1. Представление информации в ЭВМ 37
3.1.1. Системы счисления 37
3.1.2. Формы представления чисел 45
3.1.3. Представление целых чисел в ЭВМ в прямом и дополнительном кодах 49
3.1.4. Кодирование символьной информации 56
3.1.5. Типы данных, поддерживаемые микропроцессором 60
3.1.6. Кодирование десятичных чисел (BCD) 62
3.2. Логические основы построения ПК 64
3.2.1. Булевы алгебры 64
3.2.2. Комбинационные схемы 69
3.2.3. Сумматоры 73
3.2.4. Триггеры 75
3.3. Программное управление ЭВМ 78
3.3.1. Общая организация памяти 78
3.3.2. Структура ЭВМ по Нейману 80
3.3.3. Характеристики набора машинных команд 82
3.3.4. Формат машинной команды 85
3.3.5. Типы команд 87
3.3.6. Адресная часть команды 88
3.3.7. Типы операций 92
3.3.8. Режимы адресации 95
4. Начальные сведения о машинно-ориентированном языке ассемблер 99
4.1. Организация программы на ассемблере: алфавит, лексемы, предложения и выражения языка 100

4.2. Команды пересылки 111
4.3. Команды сложения и вычитания 113
4.4. Команды умножения и деления 119
4.5. Понятие о макропрограммировании 123
5. Функционально-структурная организация ПК 126
5.1. Взаимодействие компонентов компьютера через магистраль (шину). Типы шин, реализованных в ПК 126
5.2. Структура процессора 136
5.3. Сопроцессоры 141
5.4. Режимы процессора 142
5.5. Параметры процессоров 146
5.6. Цикл обработки команды 149
5.7. Обработка прерываний 152
5.8. Организация набора регистров 159
5.8.1. Программно доступные регистры 162
5.8.2. Регистры управления и состояния 166
5.8.3. Примеры организации регистров в микропроцессоре 169
5.9. Устройство управления 177
5.10. Функциональные характеристики персонального компьютера 183
6. Запоминающие устройства ПК 187
6.1. Характеристики запоминающих устройств 187
6.1.1. Размещение 187
6.1.2. Емкость 187
6.1.3. Передаваемая порция 187
6.1.4. Метод доступа 188
6.1.5. Производительность 190
6.1.6. Физические типы запоминающих устройств 191
6.2. Базовая система ввода-вывода 192
6.3. Оперативная память: структурная организация 194
6.4. Типы запоминающих устройств с произвольным доступом 196
6.4.1. Память типа DRAM 196
6.4.2. Назначение и принцип работы кэш-памяти (SRAM) 197
6.4.3. Память типа ROM 202
6.5. Физическая память 205
6.6. Логическая организация памяти 206
7. Внешняя память компьютера 211
7.1. Устройства магнитного хранения данных 211
7.2. Накопители на жестких дисках 215
7.3. Хранение данных на гибких носителях 222
7.4. Накопители со сменными носителями 226
7.4.1. Магнитооптические накопители 226
7.4.2. Хранение информации на магнитных лентах 227

7.4.3. Флэш-память 228
7.5. Устройства оптического хранения данных 229
7.5.1. CD-ROM 229
7.5.2. Стираемые оптические диски 231
7.5.3. Цифровые видеодиски 232
8. Основные внешние устройства компьютера 233
8.1. Видеоадаптеры и мониторы 234
8.2. Принтеры и сканеры 243
8.3. Сканеры 250
8.4. Клавиатура 252
8.5. Мышь 254
8.6. Модули ввода-вывода 257
8.7. Внешние интерфейсы SCSI и FireWare 261
Заключение 263
Библиографический список 266

Введение

Персональный компьютер (ПК) стал обязательным атрибутом в любом современном офисе. Это основная техническая база информационной технологии. Даже профессионалы, работающие вне компьютерной сферы, считают непременной составляющей своей компетентности знание аппаратной части персонального компьютера, хотя бы его основных технических характеристик. А особенно велик интерес к компьютерам среди молодежи, широко использующей их для своих целей.

Возможности ПК определяются характеристиками его функциональных блоков. Замена одних блоков на другие в настоящее время не представляет особой проблемы, и при необходимости можно достаточно быстро произвести модернизацию ПК. Однако, современный рынок компьютерной техники столь разнообразен, что довольно не просто выбрать нужный блок, определить конфигурацию ПК с требуемыми характеристиками. Без специальных знаний здесь практически не обойтись.

Поэтому курс «Архитектура компьютера» призван познакомить студентов с информационно-логическими основами построения ПК, со структурной организацией ПК и назначением основных функциональных блоков, а также с многоаспектной классификацией компьютерной техники.

Вот почему цель данного пособия — изложить доступно и в тоже время исчерпывающе основные принципы построения и функционирования компьютерных систем без каких-либо претензий на авторские права авторов книг, используемых при подготовке этой работы. Это не простая задача, поскольку на сегодняшний день существует огромное разнообразие электронных устройств, претендующих на название «компьютер», — от однокристальных микропроцессоров до суперкомпьютеров. Отличаются они функциональными возможностями, производительностью, размерами, областью применения. На современном этапе скорость развития компьютерной технологии не уменьшилась: эта отрасль прогрессирует такими темпами, с которыми вряд ли сравниться любая другая. Прогресс затрагивает все аспекты компьютерной технологии: от производства электронных компонентов до принципов параллельной организации таких компонентов в единую систему. Целью пособия является знакомство с архитектурой и принципами структурной организации компьютерных систем. Некоторые вопросы в пособии рассматриваются как с точки зрения архитектуры (тех атрибутов системы, которые видимы), так и с точки зрения организации (операционных блоков и их взаимосвязей, которые и реализуют архитектуру).

1. Структура и функции персонального компьютера

При описании компьютеров принято различать их структурную организацию и архитектуру.

Термин *архитектура* компьютера относится к тем характеристикам компьютера, которые доступны извне, т.е. со стороны программы, или, с другой точки зрения, оказывает непосредственное влияние на логику выполнения программы. Архитектура – это наиболее общие принципы построения ЭВМ, реализующие программное управление работой и взаимодействием основных ее функциональных узлов.

В число характеристик архитектуры входит набор машинных команд, формат разрядной сетки для представления данных разных типов, механизм обращения к средствам ввода–вывода, скрытые от программиста детали аппаратной реализации системы — управляющие сигналы, аппаратный интерфейс между компьютером и периферийным оборудованием, технологию функционирования памяти.

Под термином структурная организация компьютерной системы подразумевается совокупность операционных блоков (устройств) и их взаимосвязей, обеспечивающая реализацию спецификаций, заданных архитектурной компьютера.

Структура — это способ объединения компонентов компьютера в единое целое, а функции — операции, выполняемые каждым компонентом в процессе реализации функции, назначенной всей подсистемой.

В общем смысле функций всего четыре: обработка данных; хранение данных; перемещение данных; управление.

Персональный компьютер (Personal Computer) — это настольная или переносная ЭВМ, удовлетворяющая требованиям общедоступности и универсальности применения.

Организация ПК рассматривается применительно к самым распространенным в настоящее время IBM PC-подобным компьютерам.

На рис. 1.1. изображена типичная структурная схема ПК.

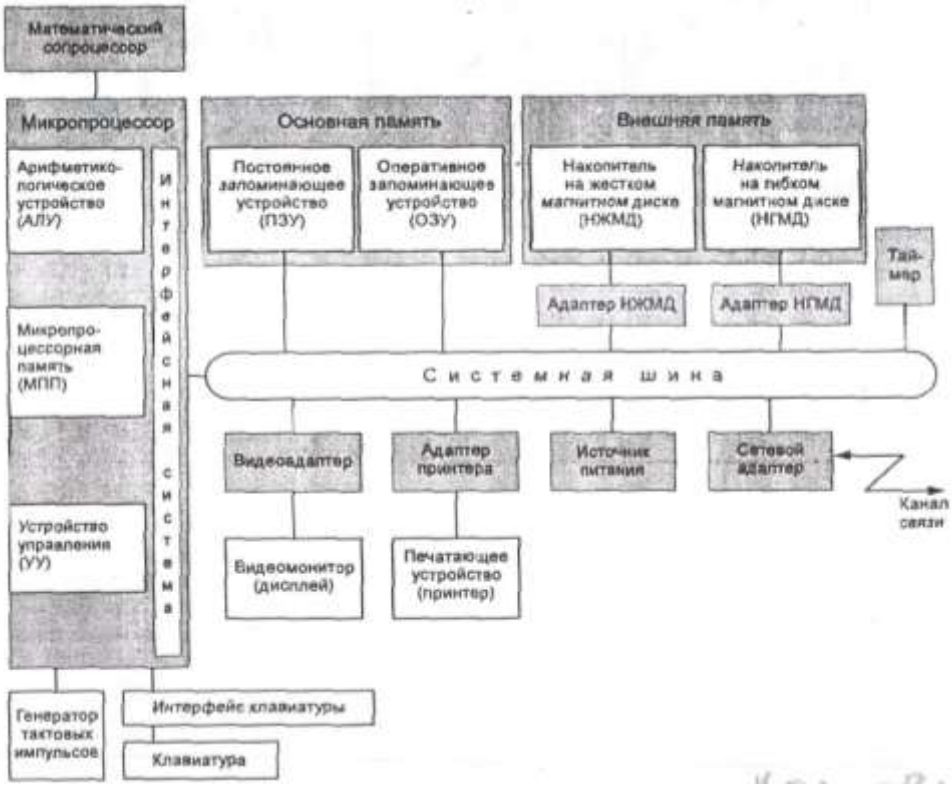


Рис. 1.1. *Структурная схема персонального компьютера*

Рассмотрим кратко основные блоки ПК.

«Мозгом» персонального компьютера является *микропроцессор*, или *центральный процессор CPU* (*Central Processing Unit*). Микропроцессор выполняет вычисления и обработку данных (за исключением некоторых математических операций, осуществляемых в компьютерах, имеющих сопроцессор) и, как правило, является самой дорогостоящей микросхемой компьютера. Это функционально законченное программно-управляемое устройство обработки информации,

выполненное в виде одной или нескольких больших (БИС) или сверхбольших (СБИС) интегральных схем.

Во всех PC-совместимых компьютерах используются процессоры, совместимые с семейством микросхем Intel.

В 1981 году IBM (International Business Machine) выпустила первый персональный компьютер IBM PC с процессором Intel 8088 (4,77 МГц) и операционной системой Microsoft Disk Operating System (DOS) версии 1.0. С этого момента практически во все персональные компьютеры устанавливаются процессоры фирмы Intel и операционные системы Microsoft.

В состав центрального процессора (ЦП) входят:

1) *устройство управления* (УУ), на которое возлагают функции управления прочими компонентами ЦП и, следовательно, всем компьютером; УУ формирует и подает во все блоки машины в нужные моменты времени определенные сигналы управления (управляющие импульсы), обусловленные спецификой выполняемой операции и результатами предыдущих операций; формирует адреса ячеек памяти, используемых выполняемой операцией, и передает эти адреса в соответствующие блоки ЭВМ; опорную последовательность импульсов устройство управления получает от генератора тактовых импульсов;
2) *арифметическое и логическое устройство* (АЛУ), которое выполняет все операции, связанные с содержательной обработкой числовой и символьной информации.
3) *микропроцессорная память* (МПП) — служит для кратковременного хранения, записи и выдачи информации, непосредственно используемой в вычислениях в ближайшие такты работы машины. МПП строится на регистрах и используется для обеспечения высокого быстродействия машины, ибо основная память (ОП) не всегда

обеспечивает скорость записи, поиска и считывания информации, необходимую для эффективной работы быстродействующего микропроцессора. *Регистры* — быстродействующие ячейки памяти различной длины (в отличие от ячеек ОП, имеющих стандартную длину 1 байт и более низкое быстродействие);

4) *интерфейсная система микропроцессора* — реализует сопряжение и связь с другими устройствами ПК; включает в себя внутренний интерфейс МП, буферные запоминающие регистры и схемы управления портами ввода-вывода (ПВВ) и системной шиной. *Интерфейс* (interface) — совокупность средств сопряжения и связи устройств компьютера, обеспечивающая их эффективное взаимодействие. *Порт ввода-вывода* (I/O — Input/Output port) — аппаратура сопряжения, позволяющая подключить к микропроцессору другое устройство ПК.

Генератор тактовых импульсов генерирует последовательность электрических импульсов; частота генерируемых импульсов определяет тактовую частоту машины. Промежуток времени между соседними импульсами определяет время одного такта работы машины или просто *такт работы машины.* Измеряется в мегагерцах: 1 МГц=1000000 тактов в секунду (например, тактовая частота 133 МГц).

Системная шина. Это основная интерфейсная система компьютера, обеспечивающая сопряжение и связь всех его устройств между собой. Системная шина включает в себя:

• *кодовую шину данных* (КШД), содержащую провода и схемы сопряжения для параллельной передачи всех разрядов числового кода (машинного слова) операнда;

• *кодовую шину адреса* (КША), включающую провода и схемы сопряжения для параллельной передачи всех разрядов кода адреса ячейки основной памяти или порта ввода-вывода внешнего устройства;

• *кодовую шину инструкций* (КШИ), содержащую провода и схемы сопряжения для передачи инструкций (управляющих сигналов, импульсов) во все блоки машины;

• *шину питания,* имеющую провода и схемы сопряжения для подключения блоков ПК к системе энергопитания.

Системная шина обеспечивает три направления передачи информации:

1) между микропроцессором и основной памятью;

2) между микропроцессором и портами ввода-вывода внешних устройств;

3) между основной памятью и портами ввода-вывода внешних устройств (в режиме прямого доступа к памяти).

Все блоки, а точнее их порты ввода-вывода, через соответствующие унифицированные разъемы (стыки) подключаются к шине единообразно: непосредственно или через *контроллеры (адаптеры).* Управление системной шиной осуществляется микропроцессором либо непосредственно, либо, что чаще, через дополнительную микросхему — *контроллер шины,* формирующий основные сигналы управления. Обмен информацией между внешними устройствами и системной шиной выполняется с использованием ASCII-кодов.

Основная память (*ОП*) предназначена для хранения и оперативного обмена информацией с прочими блоками машины. ОП содержит два вида запоминающих устройств: постоянное запоминающее устройство (ПЗУ) и оперативное запоминающее устройство (ОЗУ).

ПЗУ служит для хранения неизменяемой (постоянной) программной и справочной информации, позволяет оперативно только считывать хранящуюся в нем информацию (изменить информацию в ПЗУ нельзя).

ОЗУ предназначено для оперативной записи, хранения и считывания информации (программ и данных), непосредственно участвующей в информационно-вычислительном процессе, выполняемом ПК в текущий период времени. Главными достоинствами оперативной памяти являются ее высокое быстродействие и возможность обращения к каждой ячейке памяти отдельно (прямой адресный доступ к ячейке). В качестве недостатка ОЗУ следует отметить невозможность сохранения информации в ней после выключения питания машины (энергозависимость).

Внешняя память относится к внешним устройствам ПК и используется для долговременного хранения любой информации, которая может когда-либо потребоваться для решения задач. В частности, во внешней памяти хранится все программное обеспечение компьютера. Внешняя память содержит разнообразные виды запоминающих устройств, но наиболее распространенными, имеющимися практически на любом компьютере, являются накопители на жестких (НЖМД) и гибких (НГМД) магнитных дисках.

Назначение этих накопителей — хранение больших объемов информации, запись и выдача хранимой информации по запросу в оперативное запоминающее устройство. Различаются НЖМД и НГМД лишь конструктивно, объемами хранимой информации и временем поиска, записи и считывания информации.

В качестве устройств внешней памяти используются также запоминающие устройства на кассетной магнитной ленте (стриммеры),

накопители на оптических дисках (CD-ROM — Compact Disk Read Only Memory — компакт-диск с памятью, только читаемой) и др.

Источник питания. Это блок, содержащий системы автономного и сетевого энергопитания ПК.

Таймер. Это внутримашинные электронные часы, обеспечивающие при необходимости автоматический съем текущего момента времени (год, месяц, часы, минуты, секунды и доли секунд). Таймер подключается к автономному источнику питания — аккумулятору и при отключении машины от сети продолжает работать.

Внешние устройства (*ВУ*). От состава и характеристик ВУ во многом зависят возможность и эффективность применения ПК в системах управления и в народном хозяйстве в целом. ВУ ПК обеспечивают взаимодействие машины с окружающей средой: пользователями, объектами управления и другими ЭВМ. ВУ весьма разнообразны и могут быть классифицированы по ряду признаков. Так, по назначению можно выделить следующие виды ВУ:

- внешние запоминающие устройства (ВЗУ) или внешняя память ПК;
- диалоговые средства пользователя;
- устройства ввода информации;
- устройства вывода информации;
- средства связи и телекоммуникации:

Диалоговые средства пользователя включают в свой состав видеомониторы (дисплеи), реже пультовые пишущие машинки (принтеры с клавиатурой) и устройства речевого ввода-вывода информации.

Видеомонитор (*дисплеи*) — *у*стройство для отображения вводимой и выводимой из ПК информации.

Устройства речевого ввода-вывода относятся к быстроразвивающимся средствам мультимедиа. Устройства речевого ввода — это

различные микрофонные акустические системы, "звуковые мыши", например, со сложным программным обеспечением, позволяющим распознавать произносимые человеком буквы и слова, идентифицировать их и закодировать.

Устройства речевого вывода — это различные синтезаторы звука, выполняющие преобразование цифровых кодов в буквы и слова, воспроизводимые через громкоговорители (динамики) или звуковые колонки, подсоединенные к компьютеру.

К устройствам ввода информации относятся:

• *клавиатура* — устройство для ручного ввода числовой, текстовой и управляющей информации в ПК;

• *сканеры* (читающие автоматы) — для автоматического считывания с бумажных носителей и ввода в ПК машинописных текстов, графиков, рисунков, чертежей; в устройстве кодирования сканера в текстовом режиме считанные символы после сравнения с эталонными контурами специальными программами преобразуются в коды ASCII, а в графическом режиме считанные графики и чертежи преобразуются в последовательности двухмерных координат;

• *манипуляторы* (устройства указания): *джойстик* — рычаг,

• *мышь* — для ввода информации на экран дисплея.

К *устройствам вывода информации* относятся:

• *принтеры* — печатающие устройства для регистрации информации на бумажный носитель;

• *графопостроители* (*плоттеры*) — для вывода графической информации (графиков, чертежей, рисунков) из ПК на бумажный носитель.

Устройства связи и телекоммуникации используются для связи с приборами и другими средствами автоматизации (согласователи интерфейсов, адаптеры, цифро-аналоговые и аналого-цифровые преобразователи и т.п.) и для подключения ПК к каналам связи, к дру-

гим ЭВМ и вычислительным сетям (сетевые интерфейсные платы, "стыки", мультиплексоры передачи данных, модемы).

Сетевой адаптер является внешним интерфейсом ПК и служит для подключения его к каналу связи для обмена информацией с другими ЭВМ, для работы в составе вычислительной сети. В глобальных сетях функции сетевого адаптера выполняет модулятор-демодулятор (модем).

Многие из названных выше устройств относятся к условно выделенной группе — средствам мультимедиа.

Средства мультимедиа (multimedia — многосредовость) — это комплекс аппаратных и программных средств, позволяющих человеку общаться с компьютером, используя самые разные, естественные для себя среды: звук, видео, графику, тексты, анимацию и др.

Дополнительные схемы. К системной шине и к МП ПК наряду с *типовыми* внешними устройствами могут быть подключены и некоторые *дополнительные* платы с интегральными микросхемами, расширяющие и улучшающие функциональные возможности микропроцессора: математический сопроцессор, контроллер прямого доступа к памяти, сопроцессор ввода-вывода, контроллер прерываний и др.

Математический сопроцессор широко используется для ускоренного выполнения операций над двоичными числами с плавающей запятой, над двоично-кодированными десятичными числами, для вычисления некоторых трансцендентных, в том числе тригонометрических, функций. Математический сопроцессор имеет свою систему команд и работает параллельно (совмещение во времени) с основным МП, но под управлением последнего. Ускорение операций происходит в десятки раз. Последние модели МП, начиная с МП 80486 DX, включают сопроцессор в свою структуру.

Контроллер прямого доступа к памяти освобождает МП от прямого управления накопителями на магнитных дисках, что существенно повышает эффективное быстродействие ПК. Без этого контроллера обмен данными между ВЗУ и ОЗУ осуществляется через регистр МП, а при его наличии данные непосредственно передаются между ВЗУ и ОЗУ, минуя МП.

Сопроцессор ввода-вывода за счет параллельной работы с МП значительно ускоряет выполнение процедур ввода-вывода при обслуживании нескольких внешних устройств (дисплей, принтер, НЖМД, НГМД и др.); освобождает МП от обработки процедур ввода-вывода, в том числе реализует и режим прямого доступа к памяти.

Важнейшую роль играет в ПК контроллер прерываний.

Прерывание — временный останов выполнения одной программы в целях оперативного выполнения другой, в данный момент более важной (приоритетной) программы.

Прерывания возникают при работе компьютера постоянно. Все процедуры ввода-вывода информации выполняются по прерываниям, например, прерывания от таймера возникают и обслуживаются контроллером прерываний 18 раз в секунду (естественно, пользователь их не замечает).

Контроллер прерываний обслуживает процедуры прерывания, принимает запрос на прерывание от внешних устройств, определяет уровень приоритета этого запроса и выдает сигнал прерывания в МП. МП, получив этот сигнал, приостанавливает выполнение текущей программы и переходит к выполнению специальной программы обслуживания того прерывания, которое запросило внешнее устройство. После завершения программы обслуживания восстанавливается выполнение прерванной программы. Контроллер прерываний является программируемым.

Элементы конструкции ПК

Конструктивно ПК выполнены в виде центрального системного блока, к которому через разъемы подключаются внешние устройства: дополнительные устройства памяти, клавиатура, дисплей, принтер и др.

Системный блок обычно включает в себя системную плату, блок питания, накопители на дисках, разъемы для дополнительных устройств и *платы расширения с* контроллерами — адаптерами внешних устройств.

На *системной плате* (часто ее называют *материнской платой*— Mother Board), как правило, размещаются:

- микропроцессор;
- математический сопроцессор;
- генератор тактовых импульсов;
- блоки (микросхемы) ОЗУ и ПЗУ;
- адаптеры клавиатуры, НЖМД и НГМД;
- контроллер прерываний; таймер и др.

2. Состояние и тенденции развития ЭВМ

2.1. Этапы развития вычислительной техники

Выделяют следующие периоды в истории развития ВТ:

1) домеханический (до 17 в);
2) механический (17-19 века);
3) электромеханический (19-середина 20 веков);
4) электронный (середина 20 века по наши дни).

Домеханический период

Первые счетные приспособления стали использоваться в эпоху позднего палеолита. Найденные предметы (например, "вестоницкая кость" с зарубками, принадлежавшая к периоду около 30 тыс. лет до н.э.) показали, что широкое распространение получила тогда фиксация счета на предметах: нанесение зарубок (нарезок, насечек), завязывание узелков, проведение полос на песке или на снегу, получившие впоследствии название бирок.

На смену биркам и узелкам приходит абак. Это счетный прибор, на котором отмечены места (колонки или строчки) для отдельных разрядов чисел. Камешки, косточки и т. д., помещенные в различных колонках, имеют различное числовое значение. В основе абака лежит десятичная система счисления, в то время, как перекладывание предметов производится отдельно в разрядах единиц, десятков, сотен. Разновидность абака – русские счеты были изобретены в конце 16 в. и совершенствовались до начала 18 в.

Однако, некоторые ученые (С.В. Симонович) считают, что изобретение абак не обладало свойством автоматического выполнения вычислений, поскольку первоначально абаком называли глиняную пластину с желобами, в которых раскладывались камни.

Механический период

Открытие логарифмов шотландским математиком Джоном Непером (1550-1617). С помощью логарифмических таблиц можно было легко выполнять операции умножения и деления больших чисел. В основу изобретения Непера был положен способ умножения решеткой. Этот способ умножения Непер использовал, создавая свои счетные палочки, основным назначением которых было умножение больших чисел, а принцип работы - замена умножения сложением. Созданная им машина могла выполнять операцию умножения двух чисел.

Дальнейшее развитие эти устройства получили у Блеза Паскаля (1642 год), который изобрел механическое устройство-вычислитель, позволяющий складывать числа, и Готфрида Вильгельма фон Лейбница (описана в 1671, а создана в 1694 году), который изобрел арифметическую машину, выполняющую все четыре арифметических действия.

Первый коммерческий механический калькулятор (арифмометр) был создан Чарльзом Ксавьером Томасом в 1820 году. Это была «совершенная» машина — она выполняла операции сложения, вычитания, умножения и деления (перемножала два восьмизначных числа за 18 секунд).

Отцом первого механического компьютера можно по праву назвать Чарльза Бэббиджа, профессора математики Кембриджского университета. Эта машина, созданная в 1812 году, могла решать полиномиальные уравнения различными методами. Создав в 1822 году небольшую рабочую модель своего компьютера и продемонстрировав ее Британскому правительству, Бэббидж получил средства на дальнейшее развитие своей системы. Новая машина была создана в 1823 году. Она была паровой, полностью автоматической и даже распечатывала результаты в виде таблицы.

Работа над этим проектом продолжалась еще 10 лет, и в 1833 году был создан первый "многоцелевой" компьютер, названный *аналитической машиной.* Она могла оперировать числами с 50 десятичными знаками и сохраняла до 1 000 чисел. Впервые в этой машине было реализовано условное выполнение операций — прообраз современного оператора IF. Программы для этой машины записывались с помощью перфокарт.

Электромеханический период

Электромеханический этап развития ВТ продолжался около 60 лет.

В середине 30-х годов в универмаге г. Питтсбург (США) была установлена система, в состав которой входило 250 терминалов, соединенных телефонными линиями с 20 табуляторами Пауэрса с 15 пишущими машинками. С терминалов передавались данные, которые наносились на перфокарты и использовались для выписки счетов.

В 40-е годы 20 в. был построен ряд мощных релейных систем, характеризующихся алгоритмической универсальностью и способностью выполнять сложные научно-технические расчеты в автоматическом режиме и со скоростями, на порядок превышающими скорость работы арифмометра с электроприводом.

В 1936 г. Цузе разработал механическое запоминающее устройство (получившее впоследствии название Ц-1). Машина была выполнена на механических элементах, имела память емкостью 16 чисел (по 24 бита) и управлялась перфолентой, на которую записывалась программа ее работы. Машина работала в двоичной системе. Работа над машиной была закончена в 1938 г., но она оказалась ненадежной, и Цузе решает отказаться от механических элементов, заменив их электромеханическими реле, но сохранив механическое запоминающее

устройство. Работа над новой моделью (впоследствии названной Ц-2) проводилась Цузе совместно с Гельмутом Шрейером.

В 1939-1940 гг. Цузе разрабатывает машину Ц-3, которой суждено было стать первой программно-управляемой универсальной вычислительной машиной. Машина Ц-3 выполняла девять арифметических команд; была чисто релейной, включая память, емкость которой составляла 64 числа по 22 бита. В данной машине применялась двоичная система счисления и плавающая запятая.

Говард Айкен (1900-1973, США) в 1940 г. предложил проект автоматической ЦВМ МАРК-1. Ее основной элемент — механические счетные колеса, приводимые в движение электромотором. Эта машина была предназначена для работы в десятичной системе счисления, и числа фиксировались на 10-позиционных цифровых колесах.

Последний крупный проект в области программно-управляемых релейных машин был выполнен в Советском Союзе. Его автором был Н.И.Бессонов (1906-1963), специалист в области счетно-аналитических машин. Машина, созданная по проекту Бессонова, получила название РВМ-1 (релейная вычислительная машина). Отличительным признаком этой машины служит большой объем обрабатываемой информации и относительно малое число операций, выполняемых над введенными данными. Машина была закончена в 1957 г. и эксплуатировалась в течение восьми лет. Она содержала 5500 электромеханических реле и обладала очень высоким для релейных машин быстродействием.

Место электромеханических устройств в развитии ВТ определяется более высокой производительностью по сравнению с механическими.

Электронный период

Первым электронным компьютером можно назвать систему, созданную в 1942 году Джоном В. Атанасовым (John V. Atanasoff) в колледже штата Айова. В этом устройстве в качестве переключателей использовались вакуумные лампы.

Для военных целей требовались более совершенные вычислительные системы. И в 1946 году в Пенсильванском университете профессором электротехники Джоном Мошли и его аспирантом Дж. Преспером Экертом была создана электронно-вычислительная машина ENIAC (Electrical Numerical Integrator and Calculator — электронный цифровой интегратор и вычислитель). Она работала с 10-разрядными числами и выполняла 300 операций в секунду. Производительность компьютера ENIAC была в 1 000 раз выше, чем у механических и электромеханических устройств. В этом компьютере использовалось 18 тыс. вакуумных ламп, он занимал площадь 167 квадратных метров и потреблял 180 тыс. ватт.

Данные были представлены в десятичной системе счисления, и в этой же системе выполнялись арифметические операции. Память машины состояла из 20 "аккумуляторов", каждый из которых хранил 10-разрядное десятичное число. Для хранения одного десятичного разряда использовалась кольцевая схема на 10 электронных лампах, из которых в каждый момент времени только одна была "включена". Номер включенной лампы и соответствовал значению разряда. Основной недостаток ENIAC состоял в том, что программирование процесса вычислений в ней выполнялось вручную с помощью электрических переключателей и перекоммутации электрических кабелей.

ENIAC эксплуатировалась до 1955 года, после чего машина была разобрана.

В 1949 году появилась электронная машина в Англии под руководством Уилкса (Кембриджский уноверситет) и называлась EDSAC.

А в 1950 году была создана в Киеве МЭСМ (малая электронно-счетная машина), в 1952 году в Москве —– БЭСМ (быстродействующая электронная машина).

В конце 1973 года фирма Intel выпустила микропроцессор 8080, который мог адресовать память объемом до 64 Кбайт. Это стало толчком к промышленному производству ПК.

В 1975 году был разработан комплект Altair фирмы MITS, который можно считать первым ПК. Он состоял из процессора 8080, блока питания, лицевой панели с множеством индикаторов и запоминающего устройства емкостью 256 байт (не килобайт!). Этот ПК был построен по схеме с открытой шиной (разъемами), что позволяло другим фирмам разрабатывать дополнительные платы и периферийное оборудование.

До появления известного сейчас IBM PC (модель 5150) были разработаны модели 5110 и 5120. Хотя эти компьютеры и предшествовали IBM PC, они не имели с ним ничего общего. IBM PC был больше похож на выпущенную в 1980 году для применения в офисах модель System/23 DataMaster.

В 1976 году новая фирма, Apple Computer, вышла на рынок с компьютером Apple 1. Его системная плата была привинчена к куску фанеры, а корпуса и блока питания не было вообще. Но появившийся в 1977 году компьютер Apple II стал прообразом большинства последующих моделей, включая и IBM PC.

В конце 1980 года фирма IBM разработала первый настоящий IBM PC (Personal Computer — персональный компьютер).

В IBM PC использовался процессор 8088, который имел адресное пространство 1 Мбайт, 16-разрядную внутреннюю шину данных, но внешняя шина данных была 8-разрядной.

12 августа 1981 года с рождением IBM PC в мире микрокомпьютерной индустрии появился новый стандарт.

2.2. Классификация ЭВМ

2.2.1. Классификация ЭВМ по принципу действия

По принципу действия вычислительные машины делятся на три больших класса: аналоговые (АВМ), цифровые (ЦВМ) и гибридные (ГВМ). Критерием деления вычислительных машин на эти три класса является форма представления информации, с которой они работают.

Цифровые **вычислительные машины** (ЦВМ) — вычислительные машины дискретного действия, работают с информацией, представленной в дискретной, а точнее, в цифровой форме.

Аналоговые **вычислительные** машины (АВМ) — вычислительные машины непрерывного действия, работают с информацией, представленной в непрерывной (аналоговой) форме, т.е. в виде непрерывного ряда значений какой-либо физической величины (чаще всего электрического напряжения).

Аналоговые вычислительные машины весьма просты и удобны в эксплуатации; программирование задач для решения на них, как правило, нетрудоемкое; скорость решения задач изменяется по желанию оператора и может быть сделана сколь угодно большой (больше, чем у ЦВМ), но точность решения задач очень низкая (относительная погрешность 2-5 %). На АВМ наиболее эффективно решать математические задачи, содержащие дифференциальные уравнения, не требующие сложной логики.

Гибридные вычислительные машины (ГВМ) — вычислительные машины комбинированного действия, работают с информацией, представленной и в цифровой, и в аналоговой форме; они сов-

мещают в себе достоинства АВМ и ЦВМ. ГВМ целесообразно использовать для решения задач управления сложными быстродействующими техническими комплексами.

Наиболее широкое применение получили ЦВМ с электрическим представлением дискретной информации — электронные цифровые вычислительные машины, обычно называемые просто *электронными вычислительными машинами* (ЭВМ), без упоминания об их цифровом характере.

2.2.2. Классификация ЭВМ по назначению

По назначению ЭВМ можно разделить на три группы: универсальные (общего назначения), проблемно-ориентированные и специализированные.

Универсальные ЭВМ предназначены для решения самых различных инженерно-технических задач: экономических, математических, информационных и других задач, отличающихся сложностью алгоритмов и большим объемом обрабатываемых данных. Они широко используются в вычислительных центрах коллективного пользования и в других мощных вычислительных комплексах.

Характерными чертами универсальных ЭВМ являются:

- высокая производительность;
- разнообразие форм обрабатываемых данных: двоичных, десятичных, символьных, при большом диапазоне их изменения и высокой точности их представления;
- обширная номенклатура выполняемых операций, как арифметических, логических, так и специальных;
- большая емкость оперативной памяти;
- развитая организация системы ввода-вывода информации, обеспечивающая подключение разнообразных видов внешних устройств.

Проблемно-ориентированные ЭВМ служат для решения более узкого круга задач, связанных, как правило, с управлением технологическими объектами; регистрацией, накоплением и обработкой относительно небольших объемов данных; выполнением расчетов по относительно несложным алгоритмам; они обладают ограниченными по сравнению с универсальными ЭВМ аппаратными и программными ресурсами.

К проблемно-ориентированным ЭВМ можно отнести, в частности, всевозможные управляющие вычислительные комплексы.

Специализированные ЭВМ используются для решения узкого круга задач или реализации строго определенной группы функций. Такая узкая ориентация ЭВМ позволяет четко специализировать их структуру, существенно снизить их сложность и стоимость при сохранении высокой производительности и надежности их работы.

К специализированным ЭВМ можно отнести, например, программируемые микропроцессоры специального назначения; адаптеры и контроллеры, выполняющие логические функции управления отдельными несложными техническими устройствами, агрегатами и процессами; устройства согласования и сопряжения работы узлов вычислительных систем.

2.2.3. Классификация ЭВМ по размерам и функциональным возможностям

По размерам и функциональным возможностям ЭВМ можно разделить на сверхбольшие (суперЭВМ), большие, малые, сверхмалые (микроЭВМ).

Функциональные возможности ЭВМ обуславливают важнейшие технико-эксплуатационные характеристики:

• быстродействие, измеряемое усредненным количеством операций, выполняемых машиной за единицу времени;

• разрядность и формы представления чисел, с которыми оперирует ЭВМ;

• номенклатура, емкость и быстродействие всех запоминающих устройств;

• номенклатура и технико-экономические характеристики внешних устройств хранения, обмена и ввода-вывода информации;

• типы и пропускная способность устройств связи и сопряжения узлов ЭВМ между собой (внутримашинного интерфейса);

• способность ЭВМ одновременно работать с несколькими пользователями и выполнять одновременно несколько программ (многопрограммность);

• типы и технико-эксплуатационные характеристики операционных систем, используемых в машине;

• наличие и функциональные возможности программного обеспечения;

• способность выполнять программы, написанные для других типов ЭВМ (программная совместимость с другими типами ЭВМ);

• система и структура машинных команд;

• возможность подключения к каналам связи и к вычислительной сети;

• эксплуатационная надежность ЭВМ;

• коэффициент полезного использования ЭВМ во времени, определяемый соотношением времени полезной работы и времени профилактики. Некоторые сравнительные параметры названных классов современных ЭВМ показаны в таблице 2.1.

Исторически первыми появились *большие ЭВМ* (*ENIAC*, *IBM* 360, *IBM* 370, *ES*/9221), элементная база которых прошла путь от

электронных ламп до интегральных схем со сверхвысокой степенью интеграции. Они используются для решения задач управления производством.

Однако производительность больших ЭВМ оказалась недостаточной для ряда задач: прогнозирования метеообстановки, управления сложными оборонными комплексами. Это явилось предпосылкой для разработки и создания *суперЭВМ (типа Cray и Эльбрус),* самых мощных вычислительных систем, интенсивно развивающихся и в настоящее время. Они применяются для решения сложнейших народнохозяйственных задач и проведения экспериментов (косимческие проекты, геофизические исследования и т. д.).

Таблица 2.1. Сравнительные параметры классов современных ЭВМ

Параметр	СуперЭВМ	Большие ЭВМ	Малые ЭВМ	МикроЭВМ
Производительность, MIPS	1000-100000	10-1000	1-100	1-100
Емкость ОП, Мбайт	2000-10000	64-10000	4-512	4-256
Емкость ВЗУ, Гбайт	500-5000	50-1000	2-100	0,5-10
Разрядность, бит	64-128	32-64	16-64	16-64

Появление в 70-х гг. *малых ЭВМ* обусловлено, с одной стороны, прогрессом в области электронной элементной базы, а с другой — избыточностью ресурсов больших ЭВМ для ряда приложений. Малые ЭВМ используются чаще всего для управления технологическими процессами. Они более компактны и значительно дешевле больших ЭВМ.

Дальнейшие успехи в области элементной базы и архитектурных решений привели к возникновению *супермини-ЭВМ* (*Eclips, DEC PDP*-8) — вычислительной машины, относящейся по архитектуре,

размерам и стоимости к классу малых ЭВМ, но по производительности сравнимой с большой ЭВМ. Они используются для автоматизации различных технологических процессов, научных исследований, проектно-конструкторских работ и т. д.

Изобретение в 1969 г. микропроцессора (МП) привело к появлению в 70-х гг. еще одного класса ЭВМ— *микроЭВМ* (*Pentium III, Celeron и др.*). Именно наличие МП служило первоначально определяющим признаком микроЭВМ. Сейчас микропроцессоры используются во всех без исключения классах ЭВМ.

Многопользовательские микроЭВМ — это мощные микроЭВМ, оборудованные несколькими видеотерминалами и функционирующие в режиме разделения времени, что позволяет эффективно работать на них сразу нескольким пользователям.

Персональные компьютеры (ПК) — однопользовательские микроЭВМ, удовлетворяющие требованиям общедоступности и универсальности применения.

Рабочие станции (work station) представляют собой однопользовательские мощные микроЭВМ, специализированные для выполнения определенного вида работ (графических, инженерных, издательских и др.).

Серверы (server) — многопользовательские мощные микроЭВМ в вычислительных сетях, выделенные для обработки запросов от всех станций сети.

2.2.4. Классификация ЭВМ по этапам создания (поколения ЭВМ)

По этапам создания и используемой элементной базе (т. е. тому материалу, из которого изготавливались компьютеры) ЭВМ условно делятся на следующие поколения (табл. 2.2). Каждое последующее поколение характеризуется усилением быстродействия элементов,

увеличением объема памяти, уменьшением физических размеров и повышения надежности.

Таблица 2.2. Поколения компьютеров

Поколение	*Период*	*Технология (элементная база)*	*Типичное быстродействие (операций в секунду)*
1	1946-1957	Вакуумные лампы	40 000
2	1958-1964	Транзисторы	200 000
3	1965-1971	Интегральные схемы малой и средней степени интеграции	1 000 000
4	1972-1977	Большие интегральные схемы	10 000 000
5	начиная с 1978	Сверхбольшие интегральные схемы	100 000 000

В первых компьютерах или *компьютерах первого поколения*, таких как ENIAC, EDSAC, МЭСМ, БЭСМ, УРАЛ (1954 г.), М-20 (1957 г.), «Стрела», «Минск-1» в качестве переключателей применялись вакуумные лампы, но, хотя они в принципе работали, это порождало массу сложностей. Лампа в качестве электронного переключателя малоэффективна. Она потребляет много электроэнергии и выделяет огромное количество тепла, отвод которого был одной из основных технических проблем. Кроме того, при выделении тепла надежность ламп снижалась, и поэтому из-за выхода их из строя среднее время безотказной работы больших вычислительных систем не превышало двух часов.

Именно с изобретения в 1948 году полупроводниковых устройств — транзисторов — началась революция в компьютерной индустрии. А в конце 50-х годов появились компьютеры на транзисторах, которые принято считать *компьютерами второго поколения* (Минск-32, М-220, Мир, Наири и др.).

Построенные на их основе компьютеры имели гораздо меньшие размеры и отличались более высоким быстродействием и эффек-

тивностью. Появление транзистора послужило началом миниатюризации компьютеров, эта тенденция сохраняется и в настоящее время.

Увеличение объема памяти привело к повсеместному отказу от программирования в кодах машинных команд и широкому внедрению в практику программирования языков высокого уровня. В результате появился отдельный класс программ, объединенных термином *системное программное обеспечение.*

Транзистор, выполненный в виде отдельного электронного прибора, принято называть *дискретным компонентом.* Такие компоненты сами по себе производились на разных предприятиях и собирались в единую электрическую схему на заводе — изготовителе электронной техники с помощью проводного (в единичном и мелкосерийном производстве) или печатного (в крупносерийном производстве) монтажа. Затем из таких узлов монтировалось определенное устройство — блок компьютера.

По мере совершенствования компьютеров второго поколения количество транзисторов в них постоянно повышалось.

В 1958 году сотрудники фирмы Texas Instruments изобрели интегральную схему— полупроводниковое устройство, в котором без проводов соединяется несколько расположенных на одном кристалле транзисторов. Началась эра микроэлектроники. Повсеместное применение интегральных схем является отличительной чертой *компьютеров третьего поколения* (Урал-12, ЕС ЭВМ (машины единой серии), СМ ЭВМ (средние машины), IBM System/360 (США, 1964 г.), миникомпьютер PDP-8).

Технология производства интегральных микросхем основана на том, что все базовые электронные компоненты — транзисторы, резисторы и конденсаторы — можно изготовить из одного и того же полупроводникового материала, в качестве которого, как правило, ис-

пользуется кремний. На тонкой пластине (*wafer* — вафля) особо чистого монокристалла кремния по специальной технологии формируется матрица небольших одинаковых ячеек, каждая размером в несколько квадратных миллиметров. Во всех ячейках в ходе технологического процесса формируются идентичные электронные схемы, а затем пластина разрезается по границам ячеек на *чипы* (*chips*). Электронная схема одного чипа может насчитывать огромное количество элементарных вентилей и/или ячеек памяти плюс множество выводов для подключения внешних связей. Затем чип помещается в специальный защитный корпус, а к выводам подключаются контактные пластины. В результате получается конструктивный элемент — *интегральная микросхема* (ИС). Множество таких ИС монтируется на печатной плате, и образуется узел компьютера или даже законченное устройство.

В конце 1950-х — начале 1960-х годов по такой технологии изготавливались сравнительно простые ИС, получившие наименование микросхем малого и среднего уровня интеграции (МИС и СИС). По мере развития технологии появилась возможность повысить плотность размещения элементов в чипе.

Выделение первых трех поколений считаются устоявшимся, в дальнейшем специалисты по-разному делят поколения. В табл. 2.1. представлены еще два поколения, которые характеризуются уровнем интегральной технологии, используемой при производстве компонентов. Микросхемы высокой степени интеграции (большие интегральные схемы — БИС) содержат более 1 000 электронных компонентов в чипе (*четвертого поколения*), а к сверхбольшим интегральным схемам (СБИС) относят такие, которые содержат свыше 10 000 компонентов в чипе (*пятое поколение*). В настоящее время производятся СБИС, включающие порядка 100 000 компонентов.

В процессе совершенствования технологии производства микросхем одновременно с ростом плотности размещения элементов памяти шло и повышение плотности размещения логических переключающих схем — вентилей. В результате все меньше и меньше микросхем требовалось для того, чтобы собрать электронную схему процессора.

Качественный скачек произошел в 1971 году, когда фирма Intel, созданная 18 июля 1968 года Робертом Нойсом (Robert Noyce), Гордоном Муром (Gordon Moore) и Эндрю Гроувом (Andrew Grove), объявила о выпуске микросхемы 4004, которая содержала все компоненты процессора в одном корпусе. Этот год считается годом появления на свет нового класса устройств — *микропроцессоров.*

В 1972 году был выпущен микропроцессор 8008 фирмы Intel. Это был первый 8-разрядный микропроцессор и практически вдвое более сложный, чем его предшественник модели 4004. В 1974 году — фирма Intel начала производство микропроцессора общего назначения модели 8080. Прежние модели годились только для построения процессоров специализированных компьютеров, а процессор 8080 предназначался для компьютеров самого широкого применения. Процессор 8080, как и 8008, был 8-разрядным, но отличался более развитой системой команд, повышенным быстродействием и большим диапазоном адресуемых ячеек памяти.

Примерно в то же время началась работа над созданием 16-разрядных микропроцессоров, однако первый промышленный образец вполне работоспособного устройства такого типа появился только в конце 1970-х годов. И опять первенство принадлежало фирме Intel — этим процессором была модель 8086. Следующий барьер был преодолен всего через пару лет — в 1981 году фирмы Bell Labs и Hewlett-Packard объявили о начале производства 32-разрядных микро-

процессоров. Intel начала выпускать свой 32-разрядный микропроцессор 80386 в 1985 году.

В 1993 году Intel анонсировала свой первый процессор семейства Pentium, производительность которого была в пять раз выше по сравнению с процессором 486. Он содержал 3,1 млн. транзисторов и мог выполнять 90 млн. инструкций в секунду.

Первый процессор семейства P6 — Pentium Pro— появился на свет в 1995 году. Он содержал 5,5 млн. транзисторов и мог выполнять 300 млн. инструкций в секунду.

Процессор Pentium II был представлен фирмой Intel в мае 1997 года. Он содержал 7,5 транзисторов. В апреле 1998 года в семействе процессоров Pentium II появились Celeron (недорогая версия оригинального процессора Pentium II) и Pentium II Xeon (высокопроизводительный процессор для рабочих станций и серверов). В феврале 1999 года увидел свет процессор Pentium III.

В 2000 году Intel объявила, что создан первый процессор семейства P7 (кодовое название Merced)— Itanium, Это первый 64-разрядный процессор, открывающий новый этап в жизни операционных систем и приложений, но в то же время полностью совместимый с разрядным программным обеспечением.

Таким образом, каждое следующее поколение имеет по сравнению с предыдущим существенно лучшие характеристики: усиление быстродействия элементов, увеличение объема памяти, уменьшение физических размеров и повышения надежности.

2.3. Тенденции развития вычислительных систем

Представление о совершенствовании технологии использования компьютеров дает таблица 2.3.

Таблица 2.3. Совершенствование технологии использования компьютеров

Параметр	Этапы развития компьютеров				
	50-е гг.	60-е гг.	70-е гг.	80-е гг.	90-е гг.
Цель использования ЭВМ (преимущественно)	Научно-технические расчеты	Технические и экономические расчеты	Управление и экономические расчеты	Управление; предоставление информации	Телекоммуникации, информационное обслуживание и управление
Режим работы ЭВМ	Одно-программный	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	Пользователи с общей компьютерной подготовкой	Слабообученные пользователи
Тип диалога	Работа за пультом ЭВМ	Обмен перфоносителями и машинограммами	Интерактивный (через клавиатуру и экран)	Интерактивный по жесткому меню	Интерактивный экранный типа «вопрос-ответ»

Тенденции: 1) расширение сфер применения ЭВМ и, как следствие, переход от отдельных машин к их системам — вычислительным системам и комплексам разнообразных конфигураций с широким диапазоном функциональных возможностей и характеристик; 2) ведутся поисковые работы по созданию ЭВМ 6-го поколения, базирующихся на распределенной нейронной архитектуре, — нейрокомпьютеров. В частности, в нейрокомпьютерах могут использоваться уже имеющиеся специализированные сетевые МП — транспьютеры. Транспьютер — микропроцессор сети со встроенными средствами связи; 3) широкое внедрение средств мультимедиа, в первую очередь аудио- и видео средств ввода и вывода информации, позволит об-

щаться с компьютером на естественном языке; 4) расширение и совершенствование набора внешних устройств, усложнение системы связей между узлами ЭВМ; 5) появление многопроцессорных вычислительных машин, расширение методов параллельных операций; 6) возрастание роли межкомпьютерных коммуникаций; 7) появление быстродействующих машин для логического анализа информации и, как следствие, пересмотр фон-неймановской архитектуры.

3. Информационно-логические основы построения персонального компьютера

3.1. Представление информации в ЭВМ

3.1.1. Системы счисления

Информация в ЭВМ кодируется, как правило, в двоичной или в двоично-десятичной системе счисления.

Система счисления — это способ наименования и изображения чисел с помощью символов, имеющих определенные количественные значения.

В зависимости от способа изображения чисел системы счисления делятся на позиционные и непозиционные.

В *позиционной* системе счисления количественное значение каждой цифры зависит от ее места (позиции) в числе. В *непозиционной* системе счисления цифры не меняют своего количественного значения при изменении их расположения в числе. Количество (Р) различных цифр, используемых для изображения числа в позиционной системе счисления, называется основанием системы счисления. Значения цифр лежат в пределах от 0 до *P*-1. В общем случае запись любого смешанного числа в системе счисления с основанием Р будет представлять собой ряд вида:

$$a_{m-1}P^{m-1}+a_{m-2}P^{m-2} + ... + a_1P^1+a_0P^0 + a_{-1}P^{-1} + a_{-2}P^{-2} + ... + a_{-s}p^{-s}, \qquad (1)$$

где нижние индексы определяют местоположение цифры в числе (разряд):

• положительные значения индексов — для целой части числа (*m* разрядов);

• отрицательные значения — для дробной (*s* разрядов).

Пример 3.1. Позиционная система счисления — арабская десятичная система, в которой: основание *P*=10, для изображения чисел

используются 10 цифр (от 0 до 9). Непозиционная система счисления — римская, в которой для каждого числа используется специфическое сочетание символов (XIV, CXXVII и т.п.).

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,* а в дробной *s* разрядов, можно записать всего P^{m+s} разных чисел.

Десятичная система счисления

В повседневной жизни мы пользуемся системой счисления, основанной на десяти цифрах (0, 1, 2, 3, 4, 5, 6, 7, 8, 9), и представляем в ней любые числовые величины. Рассмотрим, какую величину представляет последовательность цифр 83, если считать ее представлением этой величины в десятичной системе счисления. Она означает сумму восьми десятков и трех единиц:

$$83 = (8 \times 10) + 3.$$

Последовательность цифр 4728 означает сумму четырех тысяч, семи сотен, двух десятков и восьми единиц:

$$4728= (4 \times 1000) + (7 \times 100) + (2 \times 10) + 8.$$

Десятичная система счисления *имеет основание* 10. Это означает, что значение, представленное последовательностью цифр, вычисляется суммированием отдельных членов, каждый из которых есть результат умножения некоторой цифры из этой последовательности на число 10, возведенное в степень, соответствующую положению цифры в последовательности (точнее, ее положением относительно разделительной точки).

$83 = (8x10^1) + (3x10^\circ)$; $4728 = (4 \times 10^3) + (7 \times 10^2) + (2 \times 10^1) + (8 \times 10^0)$

По этому же принципу представляются и величины, имеющие дробную часть:

$$472.83 = (4 \times 10^2) + (7 \times 10^1) + (2 \times 10^\circ) + (8 \times 10^{-1}) + (3 \times 10^{-2}) .$$

В общем случае число *X*, представленное в десятичной системе последовательностью цифр $\{\ldots x_2 x_1 x_0 x_{-1} x_{-2} x_{-3} \ldots\}$, имеет величину

$$X = \sum_i x_i 10^i.$$

Двоичная система счисления

В десятичной системе счисления основанием является число 10, а для представления любого числа используется 10 цифр. В двоичной системе счисления основанием является число 2, а для представления любого числа используется всего две цифры — 0 и 1.

Рядом с последовательностью цифр представления числа используют индекс, который соответствует значению основания системы счисления. Например, записи 83_{10} и 4728_{10} означают, что эти последовательности цифр следует интерпретировать в десятичной системе счисления (для десятичной с.с. индекс можно не указывать). Цифры 1 и 0 в двоичной системе счисления обозначают те же величины, что и в десятичной:

$$0_2 = 0_{10};\ 1_2 = 1_{10.}$$

В двоичной системе счисления для представления чисел в общем случае, как и в десятичной системе, используется взвешенная сумма цифр (значений разрядов), причем вес является степенями основания системы счисления, в данном случае, степенями числа 2:

$$10_2 = (1 \times 2^1) + (0 \times 2^\circ) = 2_{10}; 11_2 = (1 \times 2^1) + (1 \times 2^\circ) = 3_{10};$$

$$100_2 = (1 \times 2^2) + (0 \times 2^1) + (0 \times 2^\circ) = 4_{10}.$$

При представлении дробной части числа вес разрядов является отрицательными степенями основания системы счисления:

$$1001.101_2 = 2^3 + 2^\circ + 2^{-1} + 2^{-3} = 9.625_{10}.$$

Восьмеричная система счисления

Цифры - 0,1,2,3,4,5,6,7. Число восемь (основание системы) записывается двумя цифрами как 10, т.е. $8=10_8$. Число девять запишется как восемь плюс один, т.е. $10_8+1=11_8$, число десять запишется как 12_8 (восемь плюс два) и т.д.

Запишем в восьмеричной системе число восемьдесят пять (85). В соответствии с выражением (1) разложим число 85 по степеням основания:

$$88 = \underbrace{1\bullet 8^2}_{64} + \underbrace{2\bullet 8^1}_{16} + \underbrace{8\bullet 8^0}_{8}$$

Коэффициенты перед степенями восьмерок дадут сокращенную запись числа: $88=128_8$.

Двоично-десятичная система счисления

Хотя внутри компьютера можно выполнять операции только над двоичными кодами, пользователю в конечном счете нужны числовые данные в десятичной системе счисления. Следовательно, при выполнении большинства операций ввода-вывода, связанных с приемом данных от пользователя или представлением ему результатов, необходимо выполнять преобразование из одной системы счисления в другую. Приложения, в которых поток данных между пользователем и компьютером очень интенсивный, а вычисления не слишком сложны, целесообразно строить таким образом, чтобы данные хранились в десятичной форме, т.е. чтобы преобразование в десятичную систему счисления не требовало никаких вычислений и, следовательно, не отнимало много времени. Наиболее распространенным способом представления десятичных чисел в компьютерах, который используется буквально с первых месяцев компьютерной эры, является двоично-десятичный.

Каждая десятичная цифра заменяется 4-разрядным двоичным кодом, который формируется по обычному способу представления чисел в двоичной системе счисления: 0 = 0000, 1 = 0001, ..., 8 = 1000, 9 = 1001. С точки зрения теории информации, этот способ представления довольно избыточен, поскольку из шестнадцати возможных комбинаций 4-разрядного двоичного кода используется только десять. Для представления многоразрядного десятичного числа 4-разрядные коды цифр объединяются, причем длина последовательности обычно выбирается кратной 8 бит. Так, число 246 представляется двоичным кодом 0000001001000110. Этот код менее компактный, чем представление этого же числа в двоичной системе счисления (11110110), но зато для его получения не требуется выполнять последовательные деления — достаточно просто каждую цифру заменить ее двоичным эквивалентом. Для представления отрицательных чисел в двоично-десятичный код слева или справа включается дополнительный 4-разрядный двоичный код с комбинацией, не соответствующей ни одной из десятичных цифр. Обычно для представления знака «минус» выбирается код 1111.

Шестнадцатеричная система счисления

Поскольку современная электронная вычислительная техника принципиально построена на двоичных элементах, любые данные в ней представляются только двоичными кодами. При представлении внутренних двоичных кодов пользователю применять начертание кода в виде длинной последовательности нулей и единиц очень неудобно. Поэтому специалисты пользуются для работы с двоичными кодами "на бумаге" более компактным их представлением.

Им является шестнадцатеричная система счисления. Для преобразования из двоичного представления в шестнадцатеричное дво-

ичные цифры группируются по четыре, и каждая четырехразрядная комбинация заменяется шестнадцатеричной цифрой:

0000 = 0 1000 = 8

0001 = 1 1001 = 9

0010 = 2 1010 = A

0011 = 3 1011 = B

0100 = 4 1100 = C

0101 = 5 1101 = D

0110 = 6 1110 = E

0111 = 7 1111 = F

Таких цифр всего 16, а потому и система счисления называется шестнадцатеричной. Основанием этой системы счисления является число 16. Например:

$1A_{16} = (1_{16} \times 16^1) + (A_{16} \times 16°) = (1_{10} \times 16^1) + (10_{10} \times 16°) = 26.$

Шестнадцатеричное представление используется не только при работе с числовыми величинами, но и для компактной записи любых двоичных кодов.

1. Запись оказывается более компактной.

2. Преобразование из одного представления в другое выполняется довольно просто и наглядно.

3. Во многих случаях форматы двоичных кодов величин разбиваются на поля, длина которых кратна четырем. Тогда значение каждого поля также можно представить шестнадцатеричным кодом.

Продемонстрируем, как просто выполняется преобразование длинного двоичного кода 110111100001 в компактный шестнадцатеричный:

1101 1110 0001 $= DE1_{16}$

D E 1

Преобразование из одной системы счисления в другую

Как видно из приведенных выше примеров, преобразовать двоичное представление числа в десятичное довольно просто — нужно значение каждого разряда двоичного представления умножить на его вес — соответствующую степень двойки — и сложить эти произведения.

Сложнее выполняется обратное преобразование — из десятичной системы в двоичную. Целая и дробная части представления преобразуются раздельно. Пусть необходимо преобразовать десятичное представление целого числа N в двоичное. Разделив число N на 2 по правилам десятичной системы, получим целочисленное частное N_1 и остаток R_1, для которых выполняется соотношение:

$N = 2 \times N_1 + R_1$; $R_1 = 0$ или 1.

Далее разделим первое частное опять на 2. Получим новое целочисленное частное N_2 и остаток R_2, для которых выполняется соотношение:

$N_1 = 2 \times N_2 + R_2$; $R_2 = 0$ или 1. Из обоих соотношений следует

$N = 2(2N_2 + R_2) + R_1 = 2^2N_2 + R_2 \times 2^1 + R_1 \times 2^\circ$.

Повторив деление в третий раз, получим

$N_2 = 2 \times N_2 + R_3$; $R_2 = 0$ или 1. Отсюда следует:

$N = 2^3N_3 + R_3 \times 2^2 + R_2 \times 2^1 + R_1 \times 2^\circ$.

Поскольку $N > N_1 > N_2$ ..., то продолжая эту процедуру, получим в конце концов частное $N_k = 1$ (если только исходное число не равно 0 или 1; в этих случаях сразу получается двоичный эквивалент 0_2 или 1_2) и остаток R_k., который равен 0 или 1. Следовательно,

$N = (1 \times 2^k) + (R_k \times 2^{k-1}) + ... + (R_3 \times 2^2) + (R_2 \times 2^1) + (R_1 \times 2^\circ)$.

Таким образом, преобразование из десятичного представления в двоичное выполняется последовательным делением на 2. Последнее целочисленное частное, равное 1, и остатки, выписанные в обратном

порядке, представляют собой цифры двоичного представления числа N.

Процедура преобразования дробей представляет собой последовательное умножение на 2. На каждом шаге дробная часть предыдущего произведения вновь умножается на 2, а цифры, представляющие целую часть произведений (они всегда равны либо 0, либо 1), выписываются как последовательные (слева направо) цифры двоичного представления.

Рассмотрим, как это происходит, на примере преобразования числа $F < 1$. Число F можно представить в виде

$$F = (a_{-1} \times 2^{-1}) + (a_{-2} \times 2^{-2}) + (a_{-3} \times 2^{-3}) + \dots ,$$

где a_{-i} равны либо 0, либо 1. После умножения числа F на 2 получим

$$2F = a_{-1} + (a_{-2} \times 2^{-1}) + (a_{-3} \times 2^{-2}) + (a_{-4} \times 2^{-3}) + \dots .$$

Поскольку $0 < F < 1$, то a_{-1} — целая часть $2F$ — может быть равна только либо 0, либо 1. Следовательно,

$2F = a_{-1} + F_1$, где $0 < F_1 < 1$ и

$$F_1 = (a_{-2} \times 2^{-1}) + (a_{-3} \times 2^{-2}) + (a_{-4} \times 2^{-3}) + \dots .$$

Для определения a_{-2} процедуру нужно повторить. Эта процедура не дает точного результата — ограниченной последовательностью двоичных цифр нельзя точно представить десятичную дробь. Поэтому алгоритм преобразования дробей обычно заранее настраивается на получение результата определенной разрядности.

Выполнение арифметических действий в двоичной, восьмеричной и шестнадцатеричной системах счисления

Для выполнения четырех арифметических действий в любой системе счисления необходимо знать таблицы сложения и умножения. В двоичной системе счисления эти таблицы выглядят так:

Таблица сложения

a \ b	0	1
0	0	1
1	1	10

Таблица умножения

a \ b	0	1
0	0	0
1	0	1

Пользуясь данными таблицами, можно выполнять арифметические действия над двоичными числами по тем же правилам, что и для десятичных чисел.

Арифметические действия над восьмеричными и шестнадцатеричными числами выполняются по тем же правилам, какие применяются в десятичной системе счисления: каждую элементарную операцию над двумя 8-ричными (или 16-ричными) разрядами выполняют в уме, также как и над десятичными, а затем записывают десятичный результат 8-ричным (или 16-ричным) числом.

Пример 3.2. Сложить два 8-ричных числа 23_8 и 48_8.

23_8 Поступают так:

\+

$\underline{48_8}$ 1) $3+8=11_{10}\rightarrow 13_8$

73_8 3 пишем, 1 в уме

2) $2+4+1=7_{10}\rightarrow 7_8$

3.1.2. Формы представления чисел

В вычислительных машинах применяются две формы представления двоичных чисел:

• естественная форма или форма с фиксированной запятой (точкой);

• нормальная форма или форма с плавающей запятой (точкой).

С фиксированной запятой все числа изображаются в виде последовательности цифр с постоянным для всех чисел положением запятой**,** отделяющей целую часть от дробной.

Пример 3.3. В десятичной системе счисления имеются 5 разрядов в целой части числа (до запятой) и 5 разрядов в дробной части

числа (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 и поэтому не всегда приемлема при вычислениях.

Пример 3.4. 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:

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

При P=2, m=10 и s= 6: $0{,}015 \le N \le 1024$.

Если в результате операции получится число, выходящее за допустимый диапазон, происходит переполнение разрядной сетки, и дальнейшие вычисления теряют смысл. В современных ЭВМ естественная форма представления используется как вспомогательная и только для целых чисел.

Числа с плавающей запятой можно представить в виде

$$\pm S \times B^{\pm E}.$$

Компоненты такого представления можно сохранить в двоичном слове, состоящем из трех полей:

- поле знака числа (плюс или минус);
- поле мантиссы S;
- поле порядка E.

Основание характеристики B во всех существующих стандартах такой формы представления подразумевается неявно и не сохраняется, поскольку оно одинаково для всех чисел

Принципы представления двоичных чисел в формате с плавающей точкой поясним на примере. На рис. 3.1 схематически показан 32-разрядный формат с плавающей точкой. В крайнем левом бите

слова хранится знак числа (как обычно, положительному числу соответствует значение 0, а отрицательному — значение 1). В следующих восьми битах хранится значение порядка. Для представления порядка используется так называемый *смещенный формат.* Для получения действительного двоичного кода порядка из значения, сохраняемого в этом поле, нужно вычесть фиксированное *смещение.* Как правило, смещение равно $(2^{k-1}-1)$, где k — разрядность поля порядка. В данном случае $k = 8$, и в поле порядка можно представить коды в диапазоне от 0 до 255. Если принять значение смещения 127, то действительное значение порядка чисел, представленных в таком формате, может находиться в интервале от -127 до +128. В данном примере считается, что основание характеристики совпадает с основанием системы счисления и равно 2. Последнее поле в слове (23 бит) отводится для хранения значения мантиссы *S.*

Любое число можно представить в форме с плавающей точкой множеством способов. Так, приведенные ниже формы представления эквивалентны, если считать, что мантисса выражена в двоичной системе счисления:

$$0.110 \times 2^5$$

$$110 \times 2^2$$

$$0.0110 \times 2^6$$

Для упрощения алгоритмов выполнения арифметических операций обычно принято *нормализовать* мантиссу. Нормализованная мантисса числа, отличного от нуля, имеет вид

$$0.1\ bbb...b \times 2^{\pm E},$$

где b представляет произвольную двоичную цифру (0 или 1). Это означает, что старший (левый) значащий разряд кода мантиссы всегда равен 1. Так как он всегда равен 1, то нет смысла хранить его в составе числа, а можно просто учитывать этот факт при выполнении опе-

раций. Таким образом, в 23-битовом поле фактически хранится 24-разрядный код мантиссы, значение которой может быть в диапазоне от 0.5_{10} до 1.0.

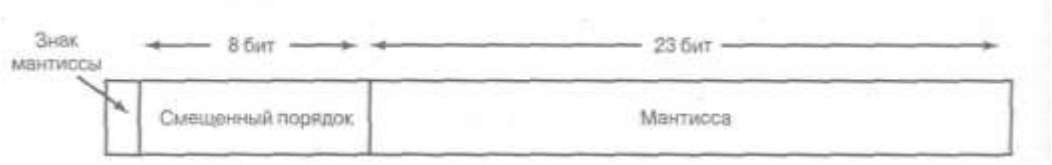


Рис. 3.1. *Формат с плавающей точкой для представления вещественных чисел*

В формате с плавающей точкой с распределением полей можно хранить:

- отрицательные числа от $-(1-2^{-24}) \times 2^{128}$ до -0.5×2^{-127},
- положительные числа от 0.5×2^{-127} до $(1-2^{-24}) \times 2^{128}$.

Формат с плавающей точкой не позволяет представить больше *отличающихся друг от друга* числовых величин — их количество по-прежнему (как и в формате с фиксированной точкой) равно 2^{32} для слова длиной 32 бит. Речь идет только о том, что эти числовые величины по-иному распределены на числовой оси. Интервал между соседними числами (кодовыми комбинациями, отличающимися значением в младшем разряде мантиссы) — переменный и зависит от абсолютной величины числа. "Соседние" числа, ближе расположенные к центру числовой оси, отличаются на меньшую величину, чем «соседние» числа, удаленные от центра числовой оси. В этом отличие такой формы представления от формы представления с фиксированной точкой. Т. о., при выполнении вычислений в формате с плавающей точкой необходимо искать компромисс между диапазоном представления и точностью вычислений. Фактически результаты округляются с точностью, определяемой имеющейся разрядной сеткой.

3.1.3. Представление целых чисел в ЭВМ в прямом и дополнительном кодах

В общем случае *n*-битовая последовательность двоичных цифр $a_{n-1}a_{n-2}\ldots a_1a_0$ может быть представлена как целое число *A*, значение которого равно:

$$A = \sum_{i=0}^{n-1} 2^i a_i \ .$$

Старший бит слова является битом хранения знака или *знаковым разрядом.* Все последующие биты слова представляют *значащие разряды* числа, которые в каждом формате интерпретируются как представление всем словом отрицательного числа.

Прямой код целого числа может быть получен следующим образом:

число переводится в двоичную систему счисления, а затем его двоичную запись слева дополняют таким количеством незначащих нулей, сколько требует тип данных, к которому принадлежит число. Например, если число $37_{(10)}= 100101_{(2)}$ объявлено величиной типа Integer (диапазон от –32768 (-2^{-15}) до 32627 (2^{15}-1)), то его прямым кодом будет 0000000000100101, а если величиной типа LongInt (диапазон от -2^{-31} до 2^{31}-1), то его прямой код будет 00000000000000000000000000100101. Для более компактной записи чаще используют шестнадцатеричный код. Полученные коды можно переписать соответственно как $0025_{(16)}$ и $00000025_{(16)}$.

Формат представления чисел в прямом коде неудобен для использования в вычислениях. Во-первых, сложение и вычитание положительных и отрицательных чисел выполняется по-разному, а потому требуется анализировать знаковые разряды операндов. Во-вторых, в прямом коде числу 0 соответствуют две кодовых комбинации:

$00000000 = +0_{10}$

$10000000 = -0_{10}$,

что усложняет анализ результата на равенство нулю, а такая операция в программах встречается очень часто.

Из-за этих недостатков прямой код практически не применяется при реализации в АЛУ арифметических операций над целыми числами. Вместо этого наиболее широкое применение находит другой формат, получивший наименование дополнительного кода.

Дополнительный код целого положительного числа совпадает с его *прямым кодом*.

Дополнительный код целого отрицательного числа может быть получен по следующему алгоритму:

1) записать прямой код модуля числа;

2) инвертировать его (заменить единицы нулями, нули — единицами);

3) прибавить к инверсному коду единицу.

Пример 3.5. Запишем дополнительный код числа (-37), интерпретируя его как величину типа LongInt:

1) прямой код числа 37 есть 00000000000000000000000000100101;

2) инверсный код 11111111111111111111111111011010;

3) дополнительный код 11111111111111111111111111011011 или $FFFFFFDB_{(16)}$.

При получении числа по его дополнительному коду прежде всего необходимо определить его знак. Если число окажется положительным, то просто перевести его код в десятичную систему счисления. В случае отрицательного числа необходимо выполнить следующий алгоритм:

1) вычесть из кода числа 1;

2) инвертировать код;

3) перевести в десятичную систему счисления. Полученное число записать со знаком минус.

Пример 3.6. Запишем числа, соответствующие дополнительным кодам:

а) 0000000000010111. Поскольку в старшем разряде записан нуль, то результат будет положительным. Это код числа 23;

б) 1111111111000000. Здесь записан код отрицательного числа. Исполняем алгоритм:

1) $1111111111000000_{(2)} - 1_{(2)} = 1111111110111111_{(2)}$;

2) 0000000001000000;

3) $1000000_{(2)} = 64_{(10)}$. Ответ: - 64.

Несколько иной способ применяется для представления в памяти персонального компьютера действительных чисел.

Персональный компьютер IBM PC позволяет работать со следующими действительными типами (диапазон значений указан по абсолютной величине):

Тип	Диапазон	Мантисса	Байты
Real	$2{,}9\cdot10^{-39}...1{,}7\cdot10^{38}$	11—12	6
Single	$1{,}5\cdot10^{-45}...3{,}4\cdot10^{38}$	7—8	4
Double	$5{,}0\cdot10^{-324}...1{,}7\cdot10^{308}$	15—16	8
Extended	$3{,}4\cdot10^{-4932}...1{,}1\cdot10^{4932}$	19—20	10

Рассмотрим преобразование действительного числа для представления его в памяти ЭВМ на примере величины типа Double.

Как видно из таблицы, величина этого типа занимает в памяти 8 байт. На рисунке ниже показано, как здесь представлены поля мантиссы и порядка:

S	Смещенный порядок	Мантисса
63	62 52	51 0

Старший бит, отведенный под мантиссу, имеет номер 51, т.е. мантисса занимает младшие 52 бита. Черта указывает здесь на положение двоичной запятой. Перед запятой должен стоять бит целой части мантиссы, но поскольку она всегда равна 1, здесь данный бит не требуется и соответствующий разряд отсутствует в памяти (но он подразумевается). Значение порядка для упрощения вычислений и сравнения действительных чисел хранится в виде *смещенного числа***,** т.е. к настоящему значению порядка перед записью его в память прибавляется смещение. Смещение выбирается так, чтобы минимальному значению порядка соответствовал нуль, т. Е. смещение вычисляется по формуле 2^{k-1}-1, где k – число бит. Например, для типа Double порядок занимает 11 бит и имеет диапазон от 2^{-1023} до 2^{1023}, поэтому смещение равно 2^{11-1}-1=2^{10}-1=1024-1=$1023_{(10)}$ = $11111111111_{(2)}$. Наконец, бит с номером 63 указывает на знак числа.

Таким образом, из вышесказанного вытекает следующий алгоритм для получения представления действительного числа в памяти ЭВМ:

1) перевести модуль данного числа в двоичную систему счисления;

2) нормализовать двоичное число, т.е. записать в виде $M*2^p$, где M — мантисса (ее целая часть равна $1_{(2)}$) и p — порядок, записанный в десятичной системе счисления;

3) прибавить к порядку смещение и перевести смещенный порядок в двоичную систему счисления;

4) учитывая знак заданного числа (0 — положительное; 1 — отрицательное), выписать его представление в памяти ЭВМ.

Пример 3.7. Запишем код числа -312,3125.

1) Двоичная запись модуля этого числа имеет вид 100l1l000,0101.

2) Имеем 100111000,0101 = 1,001110000101*2^8.

3) Получаем смещенный порядок 8 + 1023 = 1031. Далее имеем $1031_{(10)}$ = = $10000000111_{(2)}$.

4) Окончательно

1	10000000111	0011100001010000000000000000000000000000000000000000000000
63	52	0

Более компактно полученный код стоит записать следующим образом: $C073850000000000_{(16)}$.

Другой пример иллюстрирует обратный переход от кода действительного числа к самому числу.

Пример 3.8. Пусть дан код $3FEC600000000000_{(16)}$ или

0	01111111110	1100011000000000000000000000000000000000000000000000000000
63	52	0

1) Прежде всего замечаем, что это код положительного числа, поскольку в разряде с номером 63 записан нуль. Получим порядок этого числа: $01111111110_{(2)} = 1022_{(10)}$; 1022 - 1023 = -1.

2) Число имеет вид: 1,1100011*2^{-1} или 0,11100011.

3) Переводом в десятичную систему счисления получаем 0,88671875.

Отрицание

Операция отрицания числа, представленного в прямом коде, выполняется очень просто — нужно инвертировать значение знакового разряда. Если же число представлено в дополнительном коде, отрицание выполняется несколько сложнее. Правило выполнения этой операции формулируется следующим образом.

1. Следует инвертировать значение в каждом разряде представления исходного числа (положительного или отрицательного), включая и знаковый, т.е. установить значение 1 в тех разрядах, где ранее было значение 0, и значение 0 — в тех разрядах, где ранее было

значение 1 (эту операцию иногда называют *поразрядным дополнением* — bitwise complement, а ее результат — *инверсным кодом*).

2. Нужно сложить образовавшееся число с числом 0. . .001 по правилам сложения чисел без знака.

Иногда эту операцию называют вычислением *дополнения числа в дополнительном коде* (*twos complement operation*).

Например:

+18 = 00010010 (дополнительный код)

поразрядное дополнение = 11101101

\+ 1

11101110 = -18

Сложение и вычитание в дополнительном коде

Рассмотрим алгоритмы выполнения основных арифметических операций над целыми числами, представленными в дополнительном коде.

Операции сложения и вычитания рассмотрим на приведенных ниже примерах.

```
(а)  -7 =   1001          (б)  -4 =   1100
     +5 = + 0101               +4 = + 0100
          1110  = -2                 10000  = 0

(в)  +3 =   0011          (г)  -4 =   1100
     +4 = + 0100               -1 = + 1111
          0111  = 7                  11011  = -5

(д)  +5 =   0101          (е)  -7 =   1001
     +4 = + 0100               -6 = + 1010
          1001  переполнение         10011  переполнение
```

Первые четыре примера демонстрируют успешное выполнение операции. Если результат операции должен быть положительным, получается код положительного числа в дополнительном коде, а если отрицательным — код отрицательного числа в дополнительном коде. В примере (г) формируется перенос из старшего (знакового) разряда, который игнорируется.

При выполнении сложения чисел с одинаковыми знаками результат может оказаться таким, что не вмещается в используемую разрядную сетку, т.е. получается число, которое выходит за диапазон представления. Появление такого результата расценивается как *переполнение* (*overflow*), и на схему АЛУ возлагается функция выявить переполнение и выработать сигнал, который должен воспрепятствовать использованию в дальнейшем полученного ошибочного результата. Существует следующее правило обнаружения переполнения:

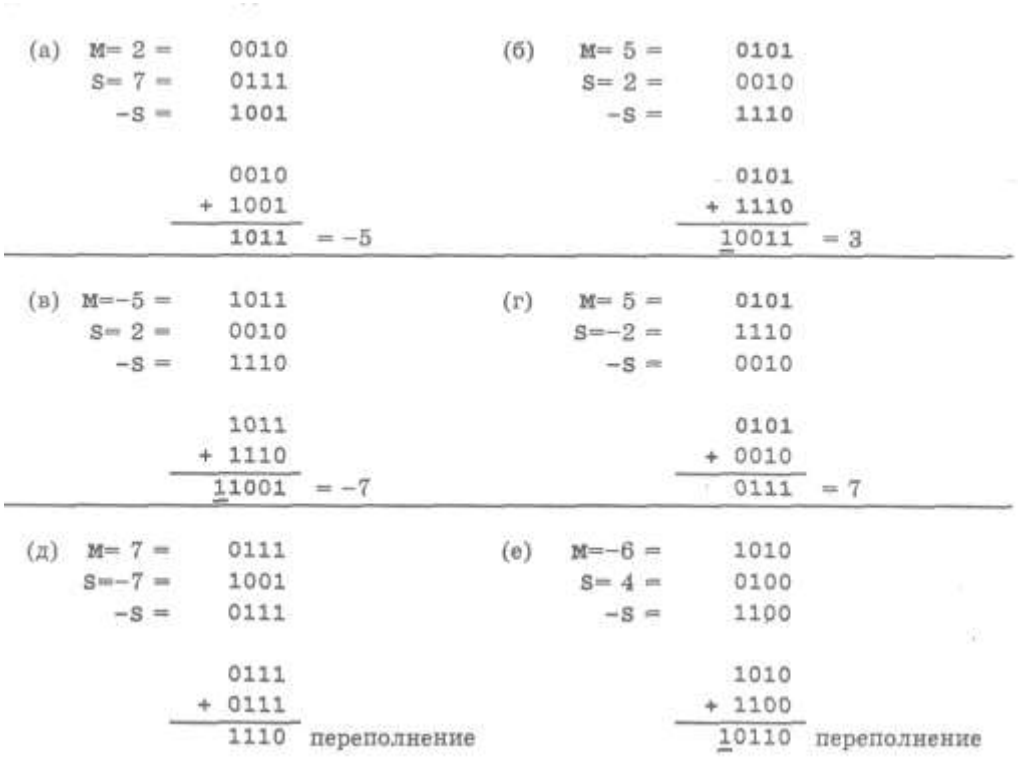
Если знаки слагаемых совпадают, то переполнение возникает в том и только в том случае, когда знак суммы, полученной по правилам сложения в дополнительном коде, отличается от знака слагаемых.

Примеры (д) и (е) иллюстрируют появление переполнения при сложении положительных и отрицательных чисел. Переполнение может появиться и в том случае, когда возникает перенос из знакового разряда и когда перенос не возникает.

Операция вычитания выполняется по следующему правилу:

Для вычитания одного числа (*вычитаемого*) из другого (*уменьшаемого*) необходимо предварительно выполнить операцию *отрицания* над вычитаемым, а затем сложить результат с уменьшаемым по правилам сложения в дополнительном коде.

Примеры выполнения операций вычитания при разных комбинациях знаков операндов приведены ниже. В этих примерах символом M обозначено уменьшаемое, а символом S — вычитаемое. Последние два примера иллюстрируют возможность появления переполнения при вычитании.



3.1.4. Кодирование символьной информации

Как и любая другая информация, символьные данные должны храниться в памяти ЭВМ в двоичном виде. Для этого каждому символу ставится в соответствие некоторое неотрицательное число, называемое кодом символа, и это число записывается в память ЭВМ в двоичном виде. Конкретное соответствие между символами и их кодами называется системой кодировки.

В ЭВМ, как правило, используются 8-разрядные коды символов. Это позволяет закодировать 256 различных символов, чего вполне достаточно для представления многих символов, используемых на практике. Поэтому для кода символа достаточно выделить в памяти один байт. Так и делают: каждый символ представляется своим кодом, который записывают в один бант памяти.

В ПК обычно используется система кодировки ASCII (American Standard Code for Information Interchange - американский стандартный код для обмена информации). Конечно, в ней не предусмотрены

коды для букв русского алфавита, поэтому в нашей стране используются варианты этой системы кодировки, в которые включают буквы русского алфавита.

Код EBCDIC (Extended Binary Coded Decimal Interchange Code - *расширенный бинарно-кодированный десятичный код*), дополненный большими и малыми русскими буквами, отличными по написанию от латинских, известен в нашей стране под названием ДКОИ-8 (двоичный код обмена информацией - 8 битовый).

Ниже показаны печатаемые (32..126) и управляющие (0..31, 127) символы — Char (в десятичной и в шестнадцатеричной системах счисления — соответственно Dec и Hex).

Первые 32 кода от - до $1F_{16}$ в табл.3.1. являются *управляющими*; они служат для представления сигналов, которые имеют специальное назначение при выводе информации на печатающее устройство, а также при передаче ее по каналам связи.

Другие коды основной таблицы ASCII (от 20 до 7F, в десятичной системе от 32 до 127) используются для кодирования цифр, букв латинского алфавита, знаков пунктуации, знаков арифметических операций и операций отношения и специальных символов. Общее количество кодов в основной таблице ASCII равно 128 (от 0 до 127).

Таблица 3.1. Таблица стандартных кодов ASCII (0-127)

Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
0	0		32	20		64	40	@	96	60	`
1	1		33	21	!	65	41	A	97	61	a
2	2		34	22	"	66	42	B	98	62	b
3	3		35	23	#	67	43	C	99	63	c
4	4		36	24	$	68	44	D	100	64	d
5	5		37	25	%	69	45	E	101	65	e
6	6		38	26	&	70	46	F	102	66	f
7	7		39	27	'	71	47	G	103	67	g
8	8		40	28	(	72	48	H	104	68	h
9	9		41	29	)	73	49	I	105	69	i
10	A		42	2A	*	74	4A	J	106	6A	j
11	B		43	2B	+	75	4B	K	107	6B	k
12	C		44	2C	,	76	4C	L	108	6C	l
13	D		45	2D	-	77	4D	M	109	6D	m
14	E		46	2E	.	78	4E	N	110	6E	n
15	F		47	2F	/	79	4F	O	111	6F	o
16	10		48	30	0	80	50	P	112	70	p
17	11		49	31	1	81	51	Q	113	71	q
18	12		50	32	2	82	52	R	114	72	r
19	13		51	33	3	83	53	S	115	73	s
20	14		52	34	4	84	54	T	116	74	t
21	15		53	35	5	85	55	U	117	75	u
22	16		54	36	6	86	56	V	118	76	v
23	17		55	37	7	87	57	W	119	77	w
24	18		56	38	8	88	58	X	120	78	x
25	19		57	39	9	89	59	Y	121	79	y
26	1A		58	3A	:	90	5A	Z	122	7A	z
27	1B		59	3B	;	91	5B	[	123	7B	{
28	1C		60	3C	<	92	5C	\	124	7C	|
29	1D		61	3D	=	93	5D	]	125	7D	}
30	1E		62	3E	>	94	5E	^	126	7E	~
31	1F		63	3F	?	95	5F	_	127	7F	

Вторая половина таблицы с 16-ричными кодами от 80 до FF (десятичные коды от 128 до 255) является *расширением* стандарта ASCII и может изменяться в разных моделях ЭВМ. В ней обычно размещаются символы национальных алфавитов (в частности, русского), символы псевдографики, а также математические и другие символы.

Отметим следующие особенности, которые важно знать при работе с символьными данными:

- код пробела меньше кода любой буквы и цифры и вообще меньше кода любого графически представимого символа;

- коды цифр упорядочены по возрастанию и идут без пропусков. Поэтому из неравенств код('0') <= код(с) <= код('9') следует, что с - цифра, и поэтому справедливо равенство код(1)=код('0')+i, где i - число от 0 до 9. Код('0') ≠ 0;
- коды больших латинских букв упорядочены согласно алфавиту и также идут без пропусков. Поэтому из неравенств код('A') <= код(с) <= код('7') следует, что с - большая латинская цифра, и поэтому код i-й по порядку (при нумерации с 0) буквы латинского алфавита равен сумме код('A')+I;
- все то же самое верно и для малых латинских букв;
- в альтернативной кодировке коды русских букв (больших и малых) упорядочены согласно алфавиту.

Ниже представлена кодировка кириллицы и символов псевдографики IBM ср-866 (128-255).

Таблица 3.2. Таблица стандартных кодов ASCII (0-127)

Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
128	80	А	160	A0	а	192	C0	└	224	E0	р
129	81	Б	161	A1	б	193	C1	┴	225	E1	с
130	82	В	162	A2	в	194	C2	┬	226	E2	т
131	83	Г	163	A3	г	195	C3	├	227	E3	у
132	84	Д	164	A4	д	196	C4	─	228	E4	ф
133	85	Е	165	A5	е	197	C5	┼	229	E5	х
134	86	Ж	166	A6	ж	198	C6	╞	230	E6	ц
135	87	З	167	A7	з	199	C7	╟	231	E7	ч
136	88	И	168	A8	и	200	C8	╚	232	E8	ш
137	89	Й	169	A9	й	201	C9	╔	233	E9	щ
138	8A	К	170	AA	к	202	CA	╩	234	EA	ъ
139	8B	Л	171	AB	л	203	CB	╦	235	EB	ы
140	8C	М	172	AC	м	204	CC	╠	236	EC	ь
141	8D	Н	173	AD	н	205	CD	═	237	ED	э
142	8E	О	174	AE	о	206	CE	╬	238	EE	ю
143	8F	П	175	AF	п	207	CF	╧	239	EF	я
144	90	Р	176	B0	▒	208	D0	╨	240	F0	Ё
145	91	С	177	B1	▒	209	D1	╤	241	F1	ё
146	92	Т	178	B2	▓	210	D2	╥	242	F2	Є
147	93	У	179	B3	│	211	D3	╙	243	F3	є
148	94	Ф	180	B4	┤	212	D4	╘	244	F4	Ї
149	95	Х	181	B5	╡	213	D5	╒	245	F5	ї
150	96	Ц	182	B6	╢	214	D6	╓	246	F6	Ў
151	97	Ч	183	B7	╖	215	D7	╫	247	F7	ў
152	98	Ш	184	B8	╕	216	D8	╪	248	F8	°
153	99	Щ	185	B9	╣	217	D9	┘	249	F9	·
154	9A	Ъ	186	BA	║	218	DA	┌	250	FA	·
155	9B	Ы	187	BB	╗	219	DB	▒	251	FB	√
156	9C	Ь	188	BC	╝	220	DC	▄	252	FC	№
157	9D	Э	189	BD	╜	221	DD	▌	253	FD	¤
158	9E	Ю	190	BE	╛	222	DE	▐	254	FE	■
159	9F	Я	191	BF	┐	223	DF	▀	255	FF	

3.1.5. Типы данных, поддерживаемые микропроцессором

Микропроцессор на уровне команд поддерживает *логическую* интерпретацию этих типов.

Целый тип со знаком — двоичное значение со знаком размером 8, 16 или 32 бита. Знак в этом двоичном числе содержится в 7, 15 или 31 бите соответственно. Нуль в этих битах в операндах соответствует положительному числу, а единица — отрицательному. Отрицательные числа представляются в дополнительном коде. Общая формула для вычисления диапазона для этого типа данных имеет вид: -2^{k-1} до $2^{k-1}-1$, где k – число бит. Числовые диапазоны следующие:

8-разрядное целое — от-128 до+127;

16-разрядное целое — от-32 768 до+32 767;

32-разрядное целое — от -2^{31} до $+2^{31} - 1$.

Целый тип без знака — двоичное значение *без знака,* размером 8, 16 или 32 бита. Общая формула для вычисления диапазона для этого типа данных имеет вид: 0 до 2^k-1, где k – число бит. Числовой диапазон следующий:

байт — от 0 до 255; слово — от 0 до 65 535;

двойное слово — от 0 до $2^{32} - 1$.

Указатель на память бывает двух типов.

• *Ближний тип* — 32-разрядный логический адрес, представляющий собой относительное смещение в байтах от начала сегмента. Эти указатели могут также использоваться в сплошной (плоской) модели памяти, где сегментные составляющие одинаковы.

• *Дальний тип* — 48-разрядный логический адрес, состоящий из двух частей: 16-разрядной сегментной части — *селектора* и 32-разрядного смещения.

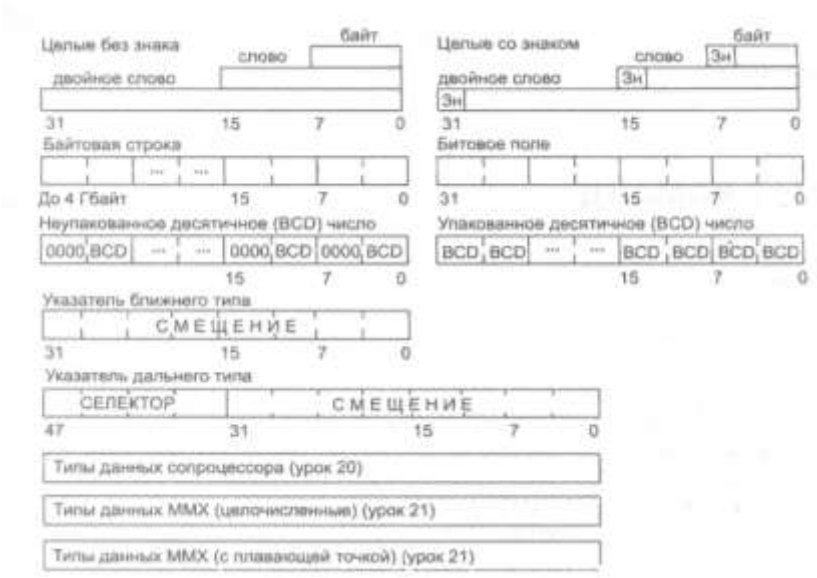


Рис. 3.2. *Основные логические типы данных микропроцессора*

Цепочка представляет собой некоторый непрерывный набор байтов, слов или двойных слов максимальной длиной до 4 Гбайт.

Битовое поле представляет собой непрерывную последовательность битов, в которой каждый бит является независимым и может рассматриваться как отдельная переменная. Битовое поле может начинаться с любого бита любого байта и содержать до 32 битов.

3.1.6. Кодирование десятичных чисел (BCD)

Десятичные числа — специальный вид представления числовой информации, в основу которого положен принцип кодирования каждой десятичной цифры числа группой из четырех бит. При этом каждый байт числа содержит одну или две десятичные цифры в так называемом *двоично-десятичном коде* (BCD — Binary-Coded Decimal). Микропроцессор хранит BCD-числа в двух форматах (рис. 3.3):

неупакованный формат — каждый байт содержит одну десятичную цифру в четырех младших битах. Старшие четыре бита имеют нулевое значение. Это так называемая *зона.* Следовательно, диапазон представления десятичного неупакованного числа в одном байте составляет от 0 до 9. Значение цифры определяется младшим полубайтом;

упакованный формат — каждый байт содержит две десятичные цифры. Десятичная цифра представляет собой двоичное значение в диапазоне от 0 до 9 размером 4 бита. При этом код старшей цифры числа занимает старшие 4 бита. Следовательно, диапазон представления десятичного упакованного числа в одном байте составляет от 00 до 99. Цифра в старшем полубайте (биты 4-7) является старшей.

Типы данных с плавающей точкой. Сопроцессор имеет несколько собственных типов данных, несовместимых с типами данных целочисленного устройства.

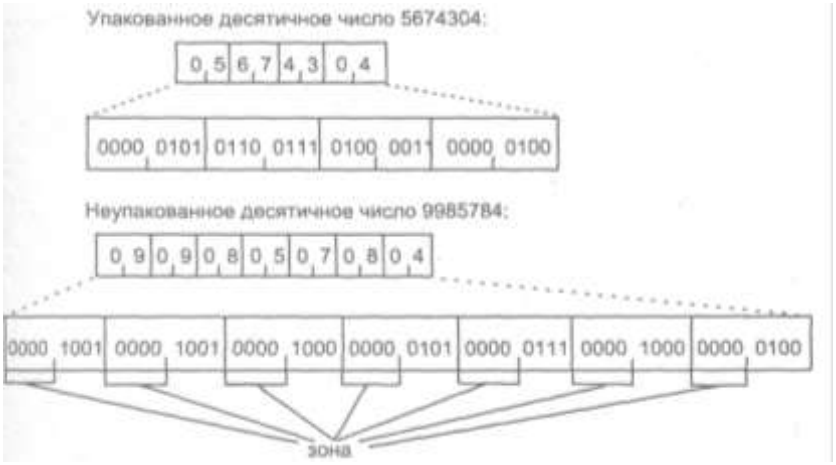


Рис. 3.3. *Представление BCD-чисел*

3.2. Логические основы построения ПК

3.2.1. Булевы алгебры

Булева алгебра является основным математическим аппаратом, который широко используется при проектировании и анализе схем обработки двоичной информации в цифровых компьютерах и других цифровых устройствах. Данный раздел математики назван в честь английского математика *Джорджа Буля* (*George Boole*), изложившего основные положения этой алгебры в своем трактате "Исследование законов мышления с помощью математических теорий логики и вероятностей", опубликованном в 1854 году. В 1937 году *Клод Шеннон* (*Claude Shannon*), в то время ассистент электротехнического факультета МИТ, предположил, что алгебру Буля можно использовать для решения проблем проектирования релейных переключающих схем. В компьютерной науке математический аппарат булевой алгебры применяется в двух областях:

- *анализ* — исследование поведения цифровых логических цепей;
- *синтез* — при проектировании цифровых устройств с помощью методов булевой алгебры отыскивается наиболее рациональный способ реализации заданной логической функции.

Булева алгебра использует переменные и операции над ними. В данном случае переменные и операции являются логическими переменными и логическими операциями. Переменные могут принимать только два значения — ИСТИНА (1) и ЛОЖЬ (0). Базовые логические операции — *конъюнкция* (AND), *дизъюнкция* (OR) и *инверсия* (NOT), — которые символически представляются знаками точки, плюса и надчеркиванием:

A AND *B* = *A* • *B;*

A OR *B*=*A*+*B*;

NOT $A = \overline{A}$.

Результатом операции AND будет 1 (ИСТИНА) в том и только в том случае, если оба операнда имеют значение 1. Операция OR даст результат 1 в том случае, если *любой* из операндов (или оба вместе) имеет значение 1. Унарная операция NOT инвертирует значение операнда. При отсутствии скобок существует приоритет операции AND над операцией OR. Кроме того, при обозначении операции AND символ точки часто опускается, если это не порождает двусмысленного толкования.

В табл. 3.3 перечислены основные тождества булевой алгебры.

Таблица 3.3. Булевы логические операции

P	*Q*	NOT *P*	*P* AND *Q*	*P* OR *Q*	*P* XOR *Q*	*P* NAND *Q*	*P* NOR *Q*
0	0	1	0	0	0	1	1
0	1	1	0	1	1	1	0
1	0	0	0	1	1	1	0
1	1	0	1	1	0	0	0

Существует два класса тождеств: базовые, или *постулаты,* которые принимаются без доказательств, и производные от постулатов (табл. 3.4).

Один из двух дистрибутивных законов отличается от привычного дистрибутивного закона обычной алгебры:

$$A + (B \bullet C) = (A + B) \bullet (A + C).$$

Два последних тождества в табл. 3.4 называются *правилами де Моргана* (De Morgan). Их можно записать в следующем виде:

$$A \text{ NOR } B = \overline{A} \text{ AND } \overline{B};$$

$$A \text{ NAND } B = \overline{A} \text{ OR } \overline{B}.$$

Таблица 3.4. Основные тождества булевой алгебры

Постулаты		
$A \cdot B = B \cdot A$	$A + B = B + A$	Коммутативные законы
$A \cdot (B + C) = (A \cdot B) + (A \cdot C)$	$A + (B \cdot C) = (A + B) \cdot (A + C)$	Дистрибутивные законы
$1 \cdot A = A$	$0 + A = A$	
$A \cdot \overline{A} = 0$	$A + \overline{A} = 1$	
Производные тождества		
$0 \cdot A = 0$	$1 + A = 1$	
$A \cdot A = A$	$A + A = A$	
$A \cdot (B \cdot C) = (A \cdot B) \cdot C$	$A + (B + C) = (A + B) + C$	Ассоциативные законы
$\overline{A \cdot B} = \overline{A} + \overline{B}$	$\overline{A + B} = \overline{A} \cdot \overline{B}$	Формулы де Моргана

Базовыми "строительными блоками" любых схем цифровых устройств являются логические переключающие элементы или *вентили* (*gates*).

Вентиль представляет собой электронную схему, которая формирует выходной сигнал в соответствии с простой булевой операцией преобразования сигналов, поданных на его входы. Как правило, в качестве функций преобразования простейших вентилей используются элементарные функции AND, OR, NOT, NAND и NOR. На рис. 3.4 показаны условные обозначения вентилей разных типов, логические выражения, реализуемые ими, и таблицы истинности. Обратите внимание на то, что в условных обозначениях инвертирование обозначается кружочком на выходе элемента.

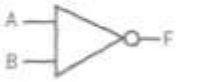
Наименование	Условное графическое обозначение	Реализуемая функция	Таблица истинности
AND		F=A•B или F=AB	A B | F 0 0 | 0 0 1 | 0 1 0 | 0 1 1 | 1
OR		F=A + B	A B | F 0 0 | 0 0 1 | 1 1 0 | 1 1 1 | 1
NOT		$F=\overline{A}$ или F=A'	A | F 0 | 1 1 | 0
NAND		$F=\overline{(AB)}$	A B | F 0 0 | 1 0 1 | 1 1 0 | 1 1 1 | 0
NOR		$F=\overline{(A + B)}$	A B | F 0 0 | 1 0 1 | 0 1 0 | 0 1 1 | 0

Рис. 3.4. *Базовые логические элементы*

Каждый элемент имеет один или два входа и один выход. Когда значение на входе изменяется, выходной сигнал реагирует на это с определенной, хотя и небольшой, задержкой, которая называется *задержкой переключения*.

Помимо логических вентилей с двумя входами, существуют и элементы с большим количеством входов — тремя, четырьмя и т.д. Так, логическое выражение $X+Y+Z$ может быть реализовано одним вентилем OR с тремя входами.

При проектировании сложных логических схем используются не все типы представленных вентилей, а только один или два типа. Ниже перечислены некоторые функционально полные множества логических функций (и, соответственно, множества логических элементов):

• AND, NOT;

• OR, NOT;

• NAND;

• NOR.

Набор трех функций — AND, OR и NOT — является функционально полным. Тогда множество AND, NOT также является функционально полным, поскольку функцию OR можно сформировать из этих двух функций, пользуясь одним из правил де Моргана:

$$A + B = \overline{\overline{A} \bullet \overline{B}}.$$

Аналогично, но с помощью другой формулы де Моргана, можно показать, что набор OR, NOT является функционально полным, поскольку с помощью этих двух функций можно реализовать функцию AND.

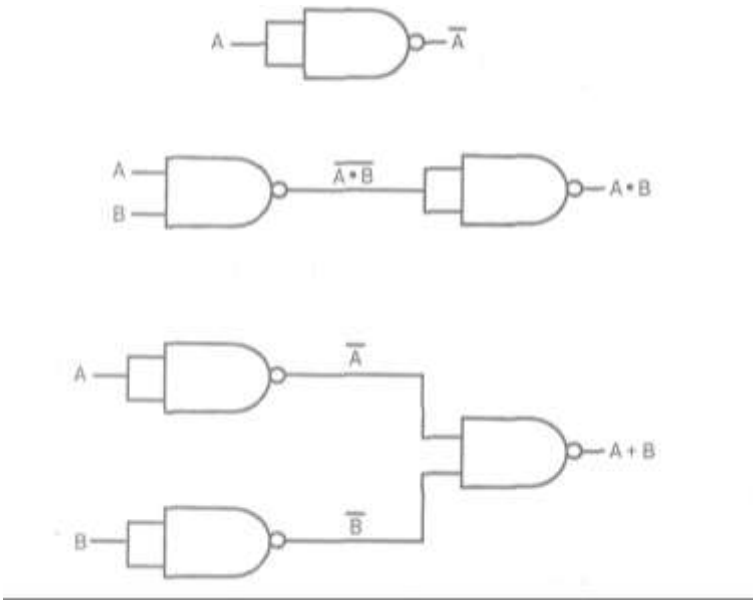


Рис. 3.5. *Использование вентилей NAND*

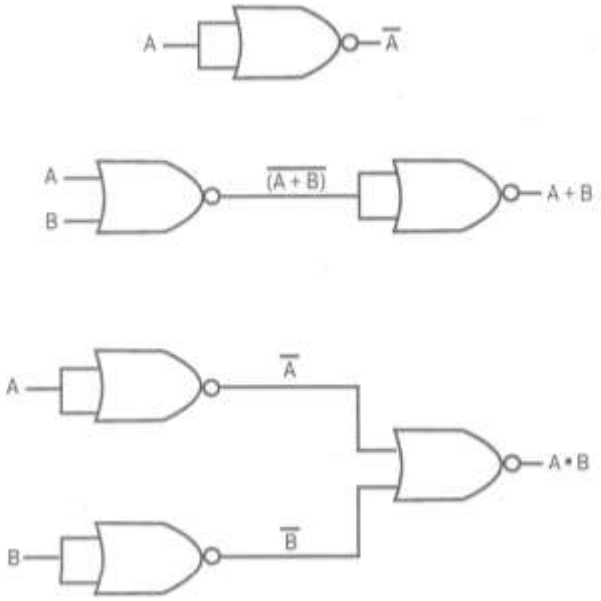


Рис. 3.6. *Использование вентилей NOR*

На рис. 3.5 показано, как с помощью вентилей NAND можно реализовать логические функции AND, OR и NOT; а на рис. 3.6 — как это можно сделать с помощью вентилей типа NOR. Из всего сказанного следует вывод: логическую схему, реализующую любую логическую функцию, можно построить из вентилей только одного типа, что часто и используется на практике.

3.2.2. Комбинационные схемы

Комбинационная логическая схема представляет собой объединение множества вентилей, и значения выходных сигналов такой схемы в каждый момент времени зависят только от значений сигналов на входах и никак не связаны с предысторией. Как и в схеме, состоящей из вентиля, сигнал на выходе комбинационной схемы устанавливается через очень малый промежуток времени, определяемый динамическими характеристиками электронных схем отдельных вентилей.

В общем случае комбинационная схема имеет *n* двоичных входов и *m* двоичных выходов. Функциональные возможности конкретной комбинационной схемы можно описать тремя способами:

- *посредством таблицы истинности,* в которой будут представлены значения набора из *m* выходных сигналов для каждой из 2^n возможных комбинаций входных сигналов;
- *посредством электрической функциональной схемы,* на которой будут изображены вентили и их соединения;
- *с помощью булева выражения,* точнее набора булевых выражений для каждого из *m* выходов, операндами которых будут *n* входов.

Реализация булевых функций

Любую булеву функцию можно реализовать в виде соединения вентилей. Одну и ту же таблицу истинности могут иметь разные булевы выражения, или, другими словами, одна и та же таблица истинности может быть по-разному представлена в виде комбинации элементарных операций. Рассмотрим, например, таблицу истинности трех переменных, представленную в табл. 3.5.

Таблица 3.5. Таблица истинности функции трех переменных

A	*B*	*C*	*F*
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

Самый простой способ представить булеву функцию в виде комбинации элементарных функций — просуммировать конъюнктивные члены, обращающие функцию в 1:

$$F = \overline{A}B\overline{C} + \overline{A}BC + AB\overline{C}.$$

Такая форма представления называется *дизъюнктивной нормальной формой* (ДНФ). На рис. 3.7 показано, как будет реализована полученная ДНФ с помощью вентилей AND, OR и NOT.

Другая форма представления также может быть получена непосредственно из таблицы истинности. ДНФ выражает тот факт, что значение функции будет равно 1, если любой из членов дизъюнкции равен 1. Но можно также сказать, что значение функции будет равно 1, если ни один из членов конъюнкции не равен 0. Следовательно,

$$F = \overline{(\overline{ABC})} \bullet \overline{(\overline{ABC})} \bullet \overline{(\overline{ABC})} \bullet \overline{(\overline{ABC})} \bullet \overline{(\overline{ABC})}.$$

Это выражение можно переписать, воспользовавшись правилами де Моргана:

$$\overline{X \bullet Y \bullet Z} = \overline{X} + \overline{Y} + \overline{Z}.$$

Тогда

$$F = \left(\overline{\overline{A}} + \overline{\overline{B}} + \overline{\overline{C}}\right) \bullet \left(\overline{\overline{A}} + \overline{\overline{B}} + \overline{C}\right) \bullet \left(\overline{\overline{A}} + \overline{\overline{B}} + \overline{C}\right) \bullet \left(\overline{\overline{A}} + \overline{\overline{B}} + \overline{C}\right) \bullet \left(\overline{\overline{A}} + \overline{\overline{B}} + \overline{C}\right)$$
$$= (A + B + C) \bullet \left(A + B + \overline{C}\right) \bullet \left(\overline{A} + B + C\right) \bullet \left(\overline{A} + B + \overline{C}\right) \bullet \left(\overline{A} + \overline{B} + \overline{C}\right).$$

Такая форма представления логической функции называется *конъюнктивной нормальной формой* (КНФ). Ее реализация с помощью элементарных вентилей показана на рис. 3.8. На схеме не показаны вентили NOT, и предполагается, что доступны как истинные значения входных сигналов, так и их инверсии.

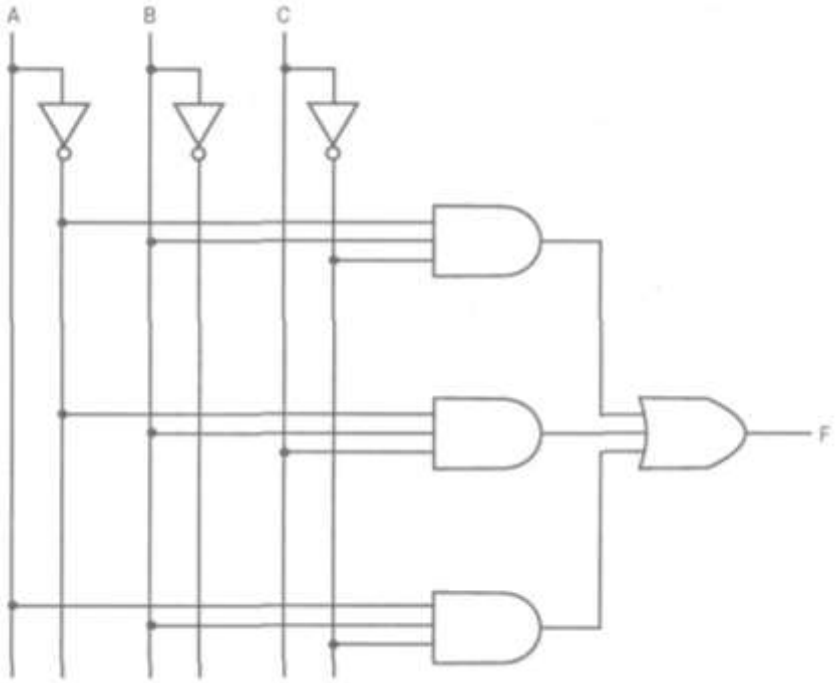


Рис. 3.7. *Реализация дизъюнктивной нормальной формы логической функции*

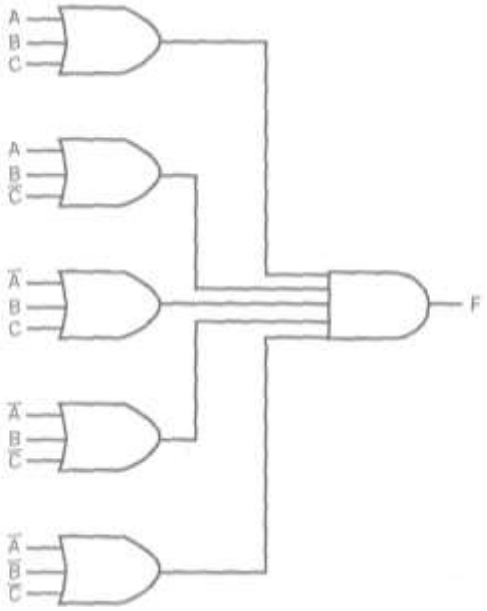


Рис. 3.8. *Реализация конъюнктивной нормальной формы логической функции*

Любая логическая функция может быть представлена либо в виде ДНФ, либо в виде КНФ. При проектировании логических схем обращают внимание на следующие факты:

• в общем случае возможно получение более простого выражения, чем то, которое дает и ДНФ, и КНФ;

• иногда предпочтительнее искать выражение, которое можно реализовать с помощью вентилей только одного типа.

В этом случае производят упрощение логического выражения алгебраическим методом, основанным на тождествах булевой алгебры.

3.2.3. Сумматоры

По основному принципу работы АЛУ может выполнять только сложение двоичных чисел, а все остальные операции реализует как совокупность операций сложения. Следовательно, основой АЛУ является схема для сложения двоичных чисел – сумматор. Основные элементы, из которых строятся сумматоры, называются одноразрядными двоичными сумматорами. Особенностью поразрядного сложения в двоичной системе является формирование сигнала переноса:

0	0	1	1
+0	+1	+0	+1
0	1	1	10

В табл. 3.6 представлена таблица истинности функции поразрядного сложения, которая формирует значение в данном разряде и сигнал переноса в следующий разряд.

Таблица 3.6. Таблица истинности поразрядного сложения без учета переноса из предыдущего разряда

A	B	Сумма	Перенос
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Поскольку нужно складывать *n*-разрядные двоичные коды, следует предусмотреть не только перенос в следующий разряд, но и учет переноса из предыдущего разряда. Тогда схемы поразрядного

сложения, реализующие модифицированную таблицу истинности, можно будет связать друг с другом и реализовать в такой составной схеме сложение кодов заданной разрядности. Модифицированная таблица истинности одноразрядного сумматора приведена в табл. 3.7, а схема 4-разрядного сумматора — на рис. 3.9. Две логические функции одноразрядного сумматора выражаются следующим образом:

$$\text{Сумма} = \overline{A}\overline{B}C + \overline{A}B\overline{C} + ABC + A\overline{B};$$
$$\text{Перенос} = AB + AC + BC.$$

Схема одноразрядного сумматора, построенного по этой формуле, показана на рис. 3.10.

Таблица 3.7. Таблица истинности поразрядного сложения с учетом переноса из предыдущего разряда

Перенос из предыдущего разряда C	A	B	Сумма	Перенос в следующий разряд
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

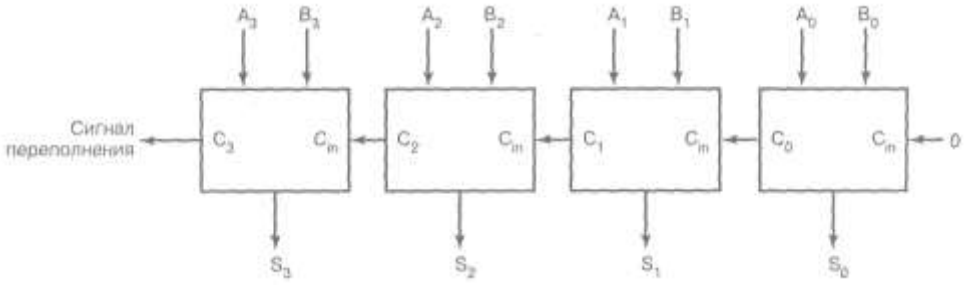


Рис. 3.9. *Четырехразрядный сумматор*

С помощью комбинационных логических схем можно реализовать многие важные функции цифровых устройств, но они не позволяют хранить информацию, а без хранения информации не может быть построено практически ни одно устройство.

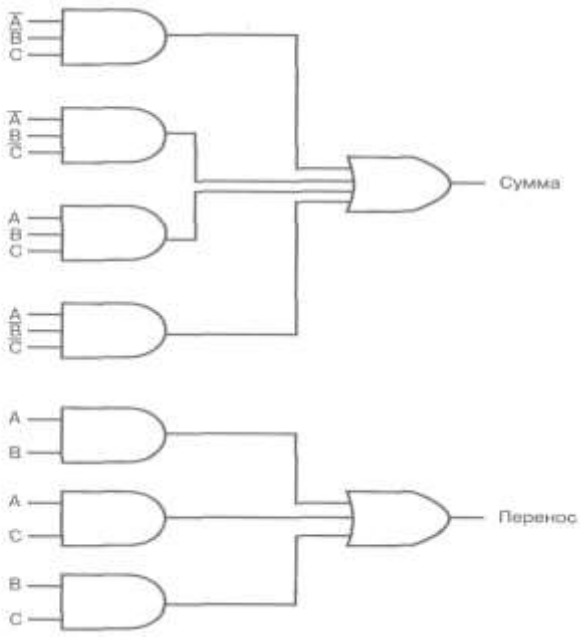


Рис. 3.10. *Схема одноразрядного сумматора*

Для того чтобы обеспечить возможность сохранения информации, в цифровых устройствах используется более сложный класс логических элементов, получивший в технической кибернетике наименование *элементов с памятью* или *конечных автоматов.* Сигнал на выходе элемента с памятью зависит не только от комбинации сигналов на его входах, но и от предыстории — текущего состояния элемента.

3.2.4. Триггеры

Простейшим логическим элементом с памятью является *триггер* (trigger – защелка). Существует множество вариантов схем триггеров, но все их объединяет наличие двух важных свойств.

- Триггер является двухстабильным элементом. При отсутствии сигналов на входах любой триггер пребывает в одном из двух состояний и остается в нем сколь угодно долго. Таким образом, триггер является элементарной ячейкой памяти емкостью 1 бит.
- Триггер имеет два выхода, сигналы на которых всегда взаимно инверсны. Принято обозначать эти выходы как Q и $\overline{Q}$.

RS-триггер

На рис. 3.11 показана схема триггера с установочными входами, который для краткости принято называть RS-триггером. Схема имеет два входа: S (*set — установить*) и R (*reset — сбросить*) и два выхода: Q и $\overline{Q}$. Построена схема на двух вентилях NOR, причем выход одного вентиля присоединен к одному из входов второго.

Покажем, что эта схема является двухстабильной. Предположим, что сигналы на входах S и R равны 0, а сигнал на выходе Q также равен 0. Тогда на обоих входах нижнего вентиля NOR будут сигналы 0 и, следовательно, сигнал на выходе $\overline{Q}$ будет равен 1. На входах верхнего вентиля NOR будут сигналы 0 и 1, а значит на его выходе Q будет сигнал 0. Таким образом, состояние схемы будет устойчивым до тех пор, пока сигналы на входах S и R будут оставаться равными 0. Точно так же можно показать, что если исходное состояние схемы $\overline{Q}$ = l, Q = 0 , то оно будет оставаться таким до тех пор, пока S = R = 0.

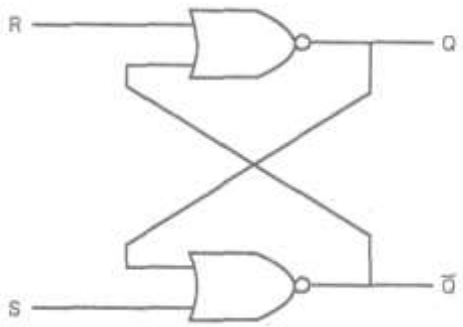


Рис. 3.11. *RS-триггер, построенный из двух вентилей NOR*

Следовательно, эта схема может работать как ячейка памяти емкостью 1 бит. Значение сигнала на выходе Q можно рассматривать как значение хранимого бита информации. Входы S и R служат для записи значений 1 или 0 в эту ячейку. Чтобы убедиться в этом рассмотрим, что произойдет, когда значение сигнала на входе станет рав-

ным 1, а триггер перед этим находился в состоянии 0 (т.е. Q = 0, $\overline{Q}$ = 1). После этого на входах нижнего вентиля NOR будут сигналы S = 1 и Q = 0. Следовательно, спустя некоторое время, соответствующее задержке переключения вентиля NOR, на его выходе появится сигнал $\overline{Q} = 0$. Тотчас же на входах верхнего вентиля NOR появятся сигналы R = 0 и $\overline{Q} = 0$. Через время Δt сигнал на выходе Q верхнего вентиля NOR станет равным 1 и установится новое стабильное состояние схемы. Если после этого снять сигнал со входа S, состояние триггера не изменится.

При подаче сигнала 1 на вход R события будут развиваться по аналогичному сценарию, но схема перейдет в устойчивое состояние Q=1, $\overline{Q} = 0$. И на этот раз время полного переключения триггера будет равно удвоенному времени переключения вентиля NOR.

Для описания работы RS-триггера часто используется табличная форма, аналогичная таблице истинности для комбинационных схем, но в данном случае она называется *таблицей переходов автомата.* В этой таблице, помимо входных сигналов, отображается и текущее состояние схемы, а в последней колонке таблицы фиксируется новое состояние схемы (табл. 3.8).

Таблица 3.8. Таблица переходов RS-триггера

Входы		Текущее состояние	Новое состояние
S	R	Q_n	Q_{n+1}
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	–
1	1	1	–

3.3. Программное управление ЭВМ

3.3.1. Общая организация памяти

Минимальной единицей представления информации в памяти ЭВМ является 1 бит, выражаемый двоичной цифрой 0 или 1. Вся память ЭВМ разбивается (подразделяется) на некоторое количество элементарных частей, называемых ячейками. Каждая ячейка представляет собой последовательность определенного числа битов памяти. Количество битов *n*, образующих одну ячейку, определяет ее длину. Длина ячейки памяти для различных моделей ЭВМ может быть различной. Память ПК делится на ячейки по 8 разрядов. Ячейки такого размера называются байтами (byte). Все байты пронумерованы, начиная с нуля. Порядковый номер байта называется его адресом. Ниже изображена структурная организация памяти компьютера.

Номера байтов	Биты							
0	0	1	0	1	0	1	0	1
1	1	1	0	0	1	0	0	1
2	1	0	1	1	0	1	1	0
3	0	0	1	0	1	1	0	0
…								

Таким образом, память компьютера обладает следующими свойствами:

1) дискретность — разделение на ячейки;
2) адресуемость, т. е. занесение информации в память и извлечение ее из памяти производится по арресам.

Байт – это наименьшая адресуемая ячейка памяти, представляющая собой 8 последовательно расположенных битов,

пронумерованных от 0 до 7, при этом бит 0 является самым младшим значащим битом.

Но в ПК есть и более крупные адресуемые ячейки – слова, двойные слова и учетверенные слова (рис. 3.12).

Рис. 3.12. *Основные типы данных микропроцессора*

Слово — последовательность из двух байтов, имеющих последовательные адреса. Размер слова — 16 битов; биты в слове нумеруются от 0 до 15. Байт, содержащий нулевой бит, называется *младшим байтом,* а байт, содержащий 15-й бит — *старшим байтом.* Микропроцессоры Intel имеют важную особенность — младший байт всегда хранится по меньшему адресу. *Адресом слова* считается адрес его младшего байта. Адрес старшего байта может быть использован для доступа к старшей половине слова.

Двойное слово — последовательность из четырех байтов (32 бита), расположенных по последовательным адресам. Нумерация этих битов производится от 0 до 31. Слово, содержащее нулевой бит, называется *младшим словом,* а слово, содержащее 31-й бит, — *старшим словом.* Младшее слово хранится по меньшему адресу. *Адресом двойного слова* считается адрес его младшего слова. Адрес старшего слова может быть использован для доступа к старшей половине двойного слова.

Учетверенное слово — последовательность из восьми байт (64 бита), расположенных по последовательным адресам. Нумерация битов производится от 0 до 63. Двойное слово, содержащее нулевой бит,

называется *младшим двойным словом,* а двойное слово, содержащее 63-й бит, — *старшим двойным словом.* Младшее двойное слово хранится по меньшему адресу. *Адресом учетверенного слова* считается адрес его младшего двойного слова. Адрес старшего двойного слова может быть использован для доступа к старшей половине учетверенного слова.

128-*битный упакованный тип данных.* Этот тип данных появился в микропроцессоре Pentium III. Для работы с ним в микропроцессор введены специальные команды.

ЭВМ является автоматическим вычислительным устройством с хранимой в памяти программой. Это означает, что команды, совокупность которых и образует программу, хранятся в памяти таким же образом, как и обычные числа.

3.3.2. Структура ЭВМ по Нейману

В 1945 году математик Джон фон Нейман (John von Neumann) опубликовал работу, посвященную обсуждению логической структуры нового компьютера EDVAC (Electronic Discrete Variable Automatic Calculator — электронный автоматический калькулятор дискретных величин) и показал, что компьютер представляет собой целостную физическую структуру и может эффективно выполнять любые вычисления без изменения аппаратной части. Другими словами, программы можно изменять, не меняя аппаратного обеспечения. Эта идея получила дальнейшее распространение в виде *хранимых в компьютере программ*.

Примерно в то же время аналогичную идею высказал и англичанин Тьюринг (Turing). В 1946 году фон Нейман и его коллеги по Институту современных исследований в Принстоне (Prinston Institute for Advanced Studies — IAS) начали работу над проектом нового ком-

пьютера с хранимой программой, известного в научном мире под именем компьютера IAS. Этот компьютер, проектирование и изготовление которого завершилось только в 1952 году, стал прототипом всех последующих компьютеров общего назначения с хранимой программой. На рис. 3.13 показана блок-схема компьютера IAS. В состав компьютера входят:

• *устройство оперативной памяти,* в котором хранятся данные и команды программы;

• *арифметическое и логическое устройство* (АЛУ), которое обрабатывает данные, представленные в двоичной системе счисления;

• у*стройство управления выполнением программы* (УУ), которое анализирует команды программы,

• *оборудование ввода-вывода***,** работающее в соответствии с сигналами, поступающими от УУ.

Рис. 3.13*. Блок-схема компьютера IAS*

За малым исключением все современные компьютеры имеют такое же распределение функций и структуру, и ее общепринято называть структурой машины фон Неймана.

Практически все современные компьютеры в общих чертах следуют современной концепции, выработанной фон НЕЙМАНОМ. Эта концепция включает три основных момента:

1) данные и команды хранятся совместно в единой подсистеме памяти, способные выполнять операции чтения и записи;
2) к отдельным элементам информации, хранящейся в памяти, можно обращаться по адресу, характеризующему ее положение в общем массиве, независимо от смысла затребованной информации, т.е. независимо от того, является информация командой или данными;
3) заданный алгоритм решения задачи реализуется последовательным выполнением элементарных команд в порядке их расположения в памяти, если только иное не будет указано явно.

3.3.3. Характеристики набора машинных команд

С точки зрения конструктора, набор машинных команд — один из основных компонентов технического задания на разработку компьютера, определяющий функциональные возможности проектируемого ЦП. С точки зрения программиста, набор машинных команд определяет, с какими регистрами и элементами оперативной памяти ему предстоит работать, какие типы данных поддерживаются на уровне операций, непосредственно выполняемых процессором, и какие операции способно выполнить АЛУ.

С описания набора машинных команд всегда начинается описание ЦП компьютера, так как операции, выполняемые центральным процессором, определяются *набором машинных команд* (machine instruction set).

Компонентами машинной команды являются:

• код операции — определяет, какие именно преобразования должен выполнить процессор с переданными ему элементами данных;

• ссылка (или ссылки) на операнд-источник — в процессе выполнения машинной команды одни данные преобразуются в другие; те данные, которые преобразуются, — операнды (может быть один операнд или два) — должны быть каким-то образом специфицированы;

• ссылка на результат — определяет, что сделать с результатом преобразования операндов (чаще всего, куда его поместить);

• ссылка на следующую выполняемую команду — определяет, откуда ЦП должен извлечь следующую команду после завершения текущей.

Следующая выполняемая команда, как правило, размещена в какой-либо ячейке оперативной памяти. В большинстве случаев очередная выполняемая команда находится в памяти сразу же за текущей, и в таких случаях информация о ней в явном виде в текущей команде не содержится. Если же эта информация включена в явном виде в команду, она должна содержать адрес в оперативной памяти.

Типовая структура (формат) команды показан на рис. 3.14.

Рис. 3.14. *Типовой формат машинной команды*

Команда включает в себя две составные части:

- код операции (кодовая часть);
- поле адресов (адресная часть).

На каждую из этих частей команды при конструировании машины отводится определенное число двоичных разрядов (битов).

Длина *кодовой части* определяется количеством (набором) элементарных операции для данной ЭВМ. Каждой операции присваивается (ставится в соответствие) определенный номер, и именно эти номера - коды операций - указываются в кодовой части команды. Формат команды должен предусматривать такую длину кодовой ча-

сти, которая позволяла бы разместить в ней код произвольной операции из системы команд ЭВМ.

В *адресной части* команды указываются (размещаются) адреса ячеек памяти, в которых хранятся обрабатываемые данные (операнды). Длина адресной части команды зависит, в общем случае, от адресного пространства памяти, способа адресации операндов, а также от количества адресов, размещаемых на адресном поле.

Изложенный выше принцип построения команды называют *адресным принципом.*

Информация об операндах и результате операции в машинной команде может относиться к одному из трех компонентов компьютера:

• оперативная или виртуальная память — как и в случае с адресом следующей команды, информация о размещении операндов или результата может иметь вид адреса в оперативной или виртуальной памяти;

• регистр ЦП — за редким исключением в составе процессора имеются один или несколько регистров (элементов сверхоперативной памяти), на которые можно ссылаться в машинной команде; если в составе процессора только один регистр, ссылка может быть неявной, в противном случае каждый регистр имеет уникальный номер, и этот номер специфицируется в машинной команде;

• устройство ввода-вывода — некоторые команды в качестве операнда или результата специфицируют модуль или устройство ввода-вывода; если в компьютере используется *совмещенный* режим адресации внешних устройств, адрес внешнего устройства в составе команды ничем не отличается от адреса в оперативной памяти или виртуального адреса.

3.3.4. Формат машинной команды

Под форматом машинной команды понимается компоновка отдельных битов в коде команды. Формат команды предусматривает наличие в ней поля кода операции и полей явной или неявной спецификации. Каждая явная спецификация операнда представляет собой ссылку на него с указанием одного из режимов адресации.

В большинстве современных компьютеров используется несколько вариантов формата. Перед началом выполнения команды ее код помещается в регистр команд (РК) центрального процессора (instruction register — IR). ЦП должен извлекать информацию из разных полей команды и выполнять необходимые операции.

Рассмотрим лишь основные форматы команд с двумя операндами.

1) Формат "регистр - регистр" (2 байта):

Команды этого формата описывают обычно действие regl:=regl*reg2 или reg2:=reg2*regl, где regl и reg2 - регистры общего назначения. Поле КОП первого байта указывает на операцию (*), которую надо выполнить. Бит w определяет размер операндов, а бит d указывает, в какой из двух регистров записывается результат:

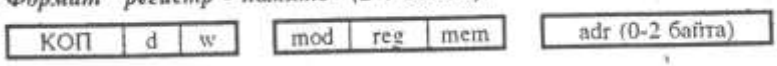
$$w=\begin{cases}1 - \text{слова}\\0 - \text{байты}\end{cases} \qquad d=\begin{cases}1 - \text{reg1:=reg1*reg2}\\0 - \text{reg2:=reg2*reg1}\end{cases}$$

Во втором байте два левых бита фиксированы (для данного формата), а трехбитовые поля regl и reg2 указывают на регистры, участвующие в операции, согласно следующей таблице:

reg	*w=1*	*w=0*
000	AX	AL
001	CX	CL
010	DX	DL
011	BX	BL

reg	*w=1*	*w=0*
100	SP	AH
101	BP	CH
110	SI	DH
111	DI	BH

2) *Формат "регистр - память" (2-4 байта):*



Эти команды описывают операции reg:=reg*adr или adr:=adr*reg, где reg - регистр, а adr - адрес ячейки памяти. Бит w первого байта определяет размер операндов (см. выше), а бит d указывает, куда записывается результат: в регистр (d=l) или в ячейку памяти (d=0). Трехбитовое поле reg второго байта указывает операнд-регистр, двухбитовое поле mod определяет, сколько байтов в команде занимает операнд-адрес (00 — 0 байтов, 01 — 1 байт, 10 — 2 байта), а трехбитовое поле mem указывает способ модификации этого адреса.

3) *Формат "регистр - непосредственный операнд" (3-4 байта):*

Команды этого формата описывают операции reg:=reg*im (im - непосредственный операнд). Бит w указывает на размер операндов, а поле reg - на регистр-операнд (см. выше). Поле КОП в первом байте определяет лишь группу операций, в которую входит операция данной команды, 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 размером в слово записывается в команде в "перевернутом" виде. Ради экономии памяти в ПК предусмотрен случаи, когда в операции над словами непосредственный операнд может быть задан байтом (на это указывает 1 в бите s при w=l), и тогда перед выполнением операции байт автоматически расширяется до слова.

4) Формат "память - непосредственный операнд" (3-6 байтов):

Команды этого формата описывают операции типа adr:=adr*im. Смысл всех полей – тот же, что и в предыдущих форматах.

3.3.5. Типы команд

Любая программа, разработанная на языке высокого уровня, какой бы сложной она не была, в конце концов преобразуется в последовательность машинных команд. Машинные команды можно разделить на следующие категории:

• команды преобразования данных — выполнение арифметических и логических операций над операндами;

• команды сохранения данных — управление памятью;

• команды перемещения — выполнение операций ввода-вывода;

• команды управления — проверка результатов и управление очередностью

выполнения команд программы.

Арифметические команды предназначены для преобразования числовых данных. Команды логических (булевых) операций работают с отдельными битами данных, представленных в двоичном коде, независимо от веса соответствующих разрядов (часто такие операции называют *поразрядными).* Арифметические и логические операции, как правило, производятся над данными, хранящимися в регистрах процессора (по крайней мере, один из операндов хранится в каком-либо регистре). Поэтому в составе набора должны быть команды, пересылающие данные из оперативной памяти в регистры и из регистров в память. Команды управления вводом-выводом необходимы для организации общения с внешним по отношению к компьютеру миром. Команды проверки и анализа используются для анализа элементов » данных и состояния вычислительного процесса. Команды ветвления (условного и безусловного переходов) используются для изменения хода вычислительного процесса в зависимости от текущих результатов арифметических и логических операций.

3.3.6. Адресная часть команды

На раннем этапе развития вычислительной техники появилась классификация вычислительных машин по количеству адресных компонентов в формате машинной команды — машины делились на одноадресные, двухадресные и трехадресные. По мере увеличения сложности процессоров вычислительных систем эта классификация становилась все менее однозначной и постепенно теряла свое значение как одна из важнейших характеристик архитектуры системы в целом.

Арифметические и логические операции делятся на унарные (операции над одним операндом) и бинарные (над двумя операндами). Таким образом, для спецификации операндов требуется не более двух адресов. Результат операции нужно где-то сохранить — это будет третий адрес. И наконец, после завершения выполнения текущей команды потребуется откуда-то извлечь следующую — это будет четвертый адрес.

На практике, однако, четырехадресных машин не существует (по крайней мере, в металле). В большинстве существующих компьютеров используются один, два или три адреса в одной команде набора, причем почти всегда "лишним" является адрес следующей команды, который формируется процессором неявно с помощью счетчика команд.

В трехадресной команде специфицируется размещение в памяти обоих операндов и результата выполнения команды. Поскольку нежелательно в процессе вычислений изменять значения исходных данных, для хранения промежуточных результатов используется ячейка по адресу T.

Поскольку трехадресный формат требует, чтобы в каждой команде хватало места для трех полей адреса, длина такой команды ока-

зывается довольно большой. Поэтому такой формат используется редко. В двухадресных командах бинарных арифметических и логических операций одно поле адреса имеет двойное назначение — оно специфицирует адрес одного из операндов и адрес результата.

Длина двухадресной команды меньше трехадресной, но у этого формата есть другой недостаток — иногда, чтобы не "запортить" значения операндов, приходится переписывать их с помощью команды MOVE в свободные ячейки памяти (программисты их называют *временными)* и в этих ячейках сохранять промежуточные результаты. В итоге количество команд, необходимых для реализации тех же вычислений, увеличивается.

Еще более короткий формат имеют одноадресные команды. Адрес второго операнда для бинарных операций задается неявно. Такой формат часто применялся в вычислительных машинах первого поколения, в которых в качестве неявно заданного хранилища второго операнда и результата использовался специальный регистр процессора — *аккумулятор*.

В *нулъадресных* командах поле адреса вообще отсутствует. Здесь используется специальная форма организации памяти — стековая организация. *Стек* представляет собой список, работающий по принципу "последним вошел — первым вышел" (LIFO — last-in-first-out). Положение вершины стека всегда известно, и, как правило, в компьютерах со стековой организацией памяти верхние две ячейки стека представляют собой регистры процессора.

Стек — это упорядоченное множество элементов данных, только один из которых может быть доступен при каждом обращении. Доступный элемент называется *вершиной стека* (или размещается в вершине стека). Количество элементов в стеке (или *длина стека)* переменное. Элементы могут добавляться только в вершину стека и

удаляться только из вершины стека. Поэтому стек иногда еще называют *магазинным списком* (pushdown list) и *списком с дисциплиной обслуживания в обратном порядке* (LIFO-list, last-in-first-out list — последним вошел, первым вышел).

Операция PUSH (заталкивание) добавляет новый элемент в вершину стека, а операция POP (выталкивание) удаляет элемент, расположенный в вершине стека. В обоих случаях вершина стека соответственно смещается. Бинарные операции, требующие двух операндов (т.е. умножение, деление, сложение, вычитание и т.п.) используют в качестве операндов два верхних элемента, которые "выталкиваются" из стека, а результат операции "заталкивается" обратно в вершину стека. Унарные операции, требующие только одного операнда (например, логическое отрицание), используют только один верхний элемент стека.

Для работы со стеком необходимо знать значения трех адресов, которые чаще всего хранятся в регистрах процессора:

- указатель стека — содержит адрес вершины стека (текущего верхнего элемента); в каждый момент времени он указывает на *текущий* верхний элемент;
- адрес основания стека — содержит адрес "дна" стека — начальный адрес области памяти, выделенной под стек;
- граничный адрес стека — содержит адрес другой границы области памяти, выделенной под стек.

Таким образом, нульадресные команды неявно обращаются за операндами к двум верхним ячейкам стека.

В табл. 3.9 суммированы сведения об интерпретации нуль-, одно-, двух- и трехадресных команд. В каждом из этих типов команд адрес следующей команды задается неявно, и выполняется бинарная операция над двумя операндами с получением одного результата.

Таблица 3.9. Интерпретация команд преобразования с разным количеством адресов в адресной части

Количество адресов	Символическое представление	Интерпретация
3	OP A, B, C	A←B OP C
2	OP A, B	A←A OP C
1	OP A	A←AC OP A
0	OP	T← T OP (T-1)

Условные обозначения: AC — аккумулятор; T — верхняя ячейка стека; A, B, C — ячейки оперативной памяти или регистры процессора

Чем меньше количество адресных полей в коде команды, тем проще набор команд и тем проще процессор, который должен реализовать выполнение этих команд. Кроме того, коды команд будут короче. Но, с другой стороны, для программирования алгоритма придется использовать больше коротких команд, что в конечном итоге приведет к увеличению времени вычислений и длины программы. Кроме того, существует определенное качественное отличие между одноадресными и многоадресными (т.е. двух- или трехадресными) командами. В первом случае программисту доступен единственный регистр процессора — аккумулятор. При работе с многоадресными командами у программиста появляется возможность обращаться в программе к нескольким регистрам. Поскольку операции с операндами, хранящимися в регистрах процессора, выполняются быстрее, чем с операндами, извлекаемыми из ячеек памяти, производительность системы при этом возрастает. Поэтому большинство современных компьютеров проектируется с ориентацией на набор многоадресных команд — как правило, используется смесь из трех- и двухадресных команд.

Проектирование набора машинных команд

Выбранный набор команд в основном определяет состав функций процессора, а, следовательно — и его структурную и схемную реализацию.

Наиболее важными вопросами проектирования набора команд являются следующие:

• набор операций, реализуемых на аппаратном уровне процессором в соответствии с машинными командами, — количество операций, степень их сложности;

• типы данных, над которыми выполняются операции на уровне машинных команд;

• формат команд — длина кода команды, количество адресных полей, размеры отдельных полей и т.д.;

• регистры — количество адресуемых регистров процессора и операции, которые можно выполнять с их содержимым;

• режимы адресации — способы (или способ) интерпретации содержимого полей адресной части команд набора.

3.3.7. Типы операций

Существует несколько типов операций, которые встречаются практически в каждой модели процессора. К ним относятся:

• операции передачи данных;

• арифметические операции;

• логические операции;

• операции преобразования;

• операции ввода-вывода;

• операции управления состоянием системы;

• операции управления ходом выполнения программы.

Ниже перечислены наиболее распространенные операции каждого типа.

Тип операции	Наименование операции	Описание
Передача данных	Перемещение	Передача слова или блока данных от источника к приемнику
	Сохранение	Передача слова из регистра процессора в память
	Загрузка (извлечение)	Передача слова из памяти в регистр процессора
	Обмен	Взаимный обмен данными между источником и приемником
	Очистка (начальная установка, сброс)	Передача слова, состоящего из всех нулей, в приемник
	Установка	Передача слова, состоящего из всех единиц, в приемник
	Запись в стек	Передача слова из источника в верхнюю ячейку стека
	Извлечение из стека	Передача слова из верхней ячейки стека в приемник
Арифметические операции	Сложение	Вычисление суммы двух операндов
	Вычитание	Вычисление разности двух операндов
	Умножение	Вычисление произведения двух операндов
	Деление	Вычисление частного от деления двух операндов
	Вычисление модуля	Замена операнда его абсолютным значением (модулем)
	Отрицание	Изменение знака операнда на противоположный
	Приращение	Увеличение значения операнда на 1
	Отрицательное приращение (операция декремента)	Уменьшение значения операнда на 1
Логические операции	И	Поразрядная булева операция конъюнкции двух операндов
	ИЛИ	Поразрядная булева операция дизъюнкции двух операндов
	НЕ	Поразрядная булева операция инверсии источника (поразрядное инвертирование)
	Исключительное ИЛИ	Поразрядная булева операция неравнозначности (суммирования по модулю 2) двух операндов
	Проверка	Проверка выполнения определенного условия и установка в соответствии с результатом значений флагов

	Сравнение	Логическое или арифметическое сравнение значений двух операндов; установка одного или нескольких флагов в соответствии с результатом
	Установка управляющих переменных	Класс машинных команд, устанавливающих определенный режим функционирования процессора с целью защиты, обработки прерывания, управления таймером и т.п.
	Сдвиг	Поразрядный сдвиг операнда вправо или влево с заполнением освобождающихся разрядов определенной константой
	Циклический сдвиг	Поразрядный сдвиг операнда вправо или влево с заполнением освобождающихся позиций значениями разрядов, "выталкиваемых" с противоположного конца
Управление ходом выполнения программы	Безусловный переход	Переход на команду, размещенную в памяти по заданному адресу; загрузка в счетчик команд заданного адреса
	Условный переход	Анализ заданного условия и при определенном результате анализа переход на команду, размещенную в памяти по заданному адресу, т.е. загрузка в счетчик команд заданного адреса
	Переход на подпрограмму	Сохранение информации о текущем состоянии программы в определенном месте и переход по заданному адресу
	Возврат из подпрограммы	Извлечение сохраненной ранее в определенном месте информации о текущем состоянии прерванной программы и восстановление этого состояния в регистрах процессора
	Выполнение	Извлечение операнда из ячейки по заданному адресу и интерпретация его значения как машинной команды; счетчик команд при интерпретации операнда не изменяется
	Пропуск	Приращение счетчика команд, приводящее к тому, что команда, следующая за текущей, не выполняется
	Условный пропуск	Анализ заданного условия и при определенном результате анализа приращение счетчика команд, приводящее к тому, что команда, следующая за текущей, не выполняется
	Останов	Прекращение выполнения программы
	Ожидание	Прекращение выполнения программы и периодический анализ заданного условия; возобновление выполнения программы, когда обнаружится, что условие удовлетворяется
	Нуль-операция	Никакие действия не выполняются, но выполнение программы продолжается
Ввод-вывод	Ввод (считывание)	Передача данных из заданного порта или устройства ввода-вывода в приемник
	Вывод (запись)	Передача данных из операнда-источника в заданной порт или устройство ввода-вывода
	Запуск ввода-вывода	Передача процессору ввода-вывода команды начать сеанс ввода-вывода
	Проверка ввода-вывода	Передача информации о текущем состоянии подсистемы ввода-вывода в операнд-приемник
Преобразование	Подстановка	Преобразование содержимого заданной области памяти в соответствии с таблицей подстановки
	Конвертация	Преобразование содержимого операнда из одного формата в другой (например, из упакованного двоично-десятичного формата в целый двоичный)

3.3.8. Режимы адресации

Распространенны следующие методы адресации:

• непосредственный;

• прямой;

• косвенный;

• регистровый;

• косвенный через регистр;

• со смещением;

• стековый.

На рис. 3.15 схематически показана суть каждого из режимов:

A — содержимое адресного поля в коде команды;

R — содержимое адресного поля в коде команды, которое ссылается на определенный регистр процессора ;

EA— исполнительный адрес ячейки, в которой находится операнд;

O — значение операнда;

(X) — содержимое ячейки X.

В табл. 3.10 показано, какие вычисления следует выполнять для определения исполнительного адреса в каждом из перечисленных режимов адресации.

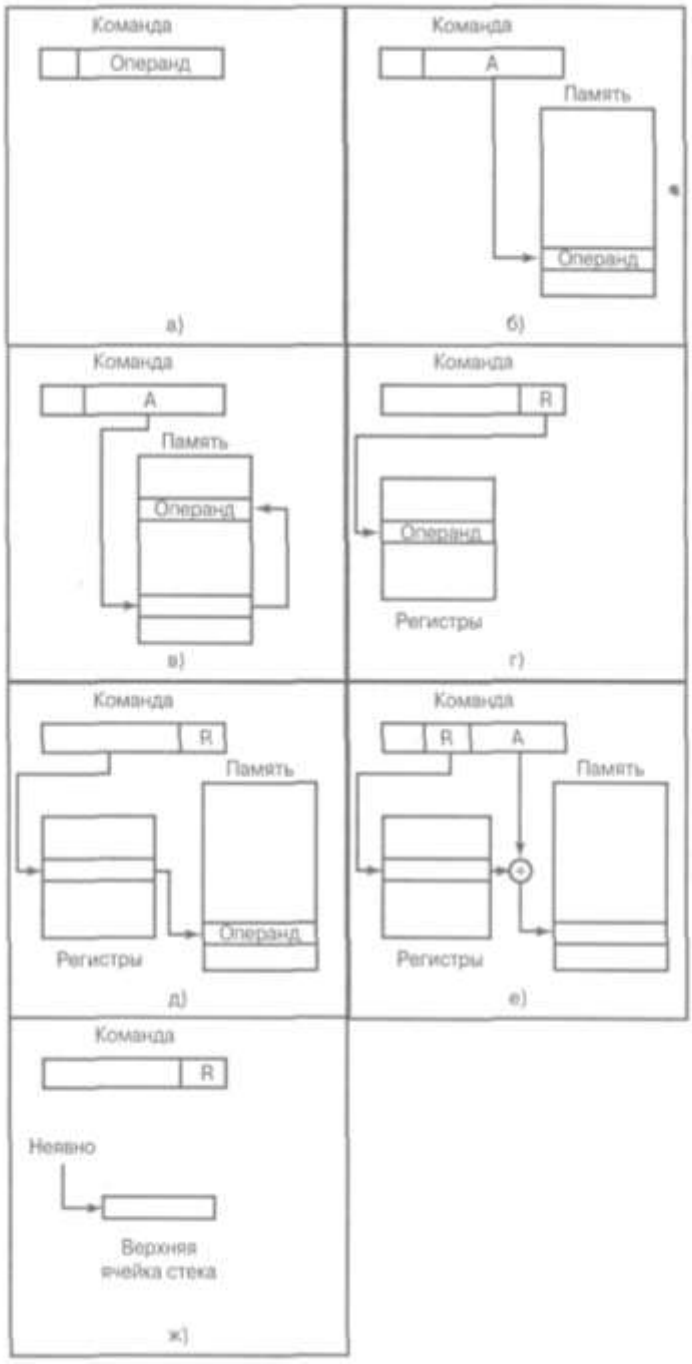


Рис. 3.15. *Режимы адресации: а*) *непосредственный; б*) *прямой; в*) *косвенный; г*) *регистровый; д*) *косвенный через регистр; е*) *смещенный; ж*) *стековый*

Таблица 3.10. Основные режимы адресации

Режим	Алгоритм	Достоинства	Недостатки
Непосредственный	O=A	Отсутствует дополнительное обращение к памяти для выборки операнда	Ограниченный диапазон значений операнда
Прямой	EA=A	Простота	Ограниченное адресуемое пространство
Косвенный	EA=(A)	Большой объем адресуемого пространства	Несколько обращений к памяти для выборки операнда
Регистровый	EA=R	Отсутствует дополнительное обращение к памяти для выборки операнда	Ограниченное адресуемое пространство
Косвенный через регистр	EA=(R)	Большой объем адресуемого пространства	Дополнительное обращение к памяти для выборки операнда
Со смещением	EA=A+(R)	Гибкость	Сложность
Стековый	EA=вершина стека	Отсутствует дополнительное обращение к памяти для выборки операнда	Ограниченная возможность применения

Практически во всех существующих моделях компьютеров предусмотрено несколько режимов адресации. Поэтому возникает вопрос, каким образом узел управления процессора узнает, какой именно режим адресации задан в конкретной машинной команде. Существует несколько методов решения этой задачи. Часто за определенным кодом операции "закрепляется" и определенный режим адресации. Альтернативный вариант предусматривает, что в формате команды выделяется несколько битов под *поле режима адресации.* Код в этом поле и определяет режим адресации при выполнении конкретной команды.

Адресация со смещением

Существуют следующие варианты:

- относительная адресация;
- адресация через регистр базы;
- индексная адресация.

Относительная адресация

В режиме относительной адресации неявно указываемым регистром является счетчик программы (общепринятое его обозначение — PC, program counter), т.е. для вычисления исполнительного адреса суммируются содержимое адресного поля команды и адрес текущей команды. При выполнении этой операции содержимое адресного поля обычно интерпретируется как число в дополнительном коде.

Адресация через регистр базы

В этом режиме содержимое полей кода команды интерпретируется следующим образом: указанный регистр содержит адрес в памяти, а адресное поле — смещение относительно этого адреса (обычно рассматривается как целое число без знака). Ссылка на регистр может быть явной или неявной.

Индексная адресация

При индексной адресации информация в коде команды интерпретируется следующим образом: содержимое адресного поля указывает адрес в оперативной памяти, а содержимое указанного в команде регистра содержит положительное смещение относительно этого адреса.

Стековая адресация

Стековая адресация — одна из форм неявной адресации. При этом в машинную команду не нужно включать ссылку на операнд, поскольку кодом операции неявно предусмотрено выполнение действий с данными, находящимися в вершине стека. В вычислительных машинах первых поколений стековая адресация использовалась довольно редко, но ее стали широко применять в микропроцессорах.

4. Начальные сведения о машинно-ориентированном языке ассемблер

Программировать на машинном языке сложно. Одна из причин этого – цифровая форма записи команд и данных, для людей более привычны символьные обозначения. И уже давно придумали средство, упрощающее составление машинных программ. Это язык ассемблера (другое название – автокод), представляющий собой фактически символьную форму записи машинного языка: в нем вместо цифровых кодов операции выписывают привычные знаки операций или их словесные названия, вместо адресов – имена, а константы записывают в десятичном системе.

Выполнение программы, написанной на языке ассемблера, осуществляется в четыре этапа: сначала создание текстового файла с расширением .asm, затем трансляция (перевод исходного текста в машинный код – файл с расширением .obj), компоновка программы (преобразование объектного кода в выполняемый вид – в файл с расширением .exe), отладка и выполнение программы.

Для любой ЭВМ можно придумать разные языки ассемблера, хотя бы потому, что можно по-разному обозначать машинные операции. В частности, и для ПК разработано несколько таких языков. Большую популярность на рынке ассемблеров для микропроцессоров фирмы Intel имеют два пакета:

- «Макроассемблер» MASM фирмы Microsoft;
- Turbo Assembler TASM фирмы Borland.

Пакет макроассемблера фирмы Microsoft (MASM) получил свое название потому, что он позволяет программисту задавать макроопределения (макросы), представляющие собой именованные группы команд. Пакет Turbo Assembler (TASM) имеет два режима работы:

режим MASM (поддерживает основные возможности макроассемблера MASM), и режим IDEAL, представляющий более удобный синтаксист написания программ, а также эффективное использование памяти при трансляции программы и другие новшества, которые приближают компилятор ассемблера к компиляторам языков высокого уровня. Для работы будет использован далее пакет ассемблера TASM 5.0 фирмы Borland.

4.1. Организация программы на ассемблере: алфавит, лексемы, предложения и выражения языка

В общем случае программы состоят из одного или более сегментов, где каждый сегмент представляет собой физически различные части кода или данных (или код и данные), к которым можно обращаться через сегментный регистр. При этом программу разбивают на инструкции программы и ее данные.

Лексемы

Лексемы — это простейшие конструкции (идентификаторы, числа и строки).

Идентификаторы нужны для обозначения различных объектов программы - переменных, меток, названий операций и т. п.

В языке ассемблера идентификатор — это последовательность из латинских букв (больших и малых), цифр и следующих знаков: ? . @ _ $

На эту последовательность накладываются следующие ограничения:

- длина идентификатора может быть любой, но значащими являются только первые 31 символ (идентификаторы, отличающиеся только в 32-й и последующих позициях, считаются одинаковыми);
- идентификатор не должен начинаться с цифры (6A — ошибка);

• точка может быть только первым символом идентификатора (.A — можно, A . — нельзя);

• в идентификаторах одноименные большие и малые буквы считаются эквивалентными (AX, Ax, aX и ax — одно и то же);

В идентификаторах нельзя использовать буквы русского алфавита.

Идентификаторы делятся на служебные слова и имена. Служебные слова имеют заранее определенный смысл, они используются для обозначения таких объектов, как регистры (AX, SI и т. п.), названия команд (ADD, NEG и т. п.) и т. п. Все остальные идентификаторы называются именами, программист может пользоваться ими по своему усмотрению, обозначая ими переменные, метки и другие объекты программы.

Целые числа могут быть записаны в десятичной, двоичной, восьмеричной и шестнадцатеричной системах счисления (другие системы не допускаются). Десятичные числа записываются как обычно, а вот при записи чисел в других системах в конце числа ставится спецификатор — буква, которая указывает, в какой системе записано это число: в конце двоичного числа ставится буква b (binary), в конце восьмеричного числа - буква o (octal) или буква q (буква "o" очень похожа на ноль, поэтому для меньшей путаницы рекомендуется использовать букву "q"), а в конце шестнадцатеричного числа - буква h (hexadecimal). Ради общности букву d (decimal) разрешается указывать и в конце десятичного числа, но обычно этого не делают.

Пример 4.1. Десятичные числа: 25, -386, +4, 25d, -386d; двоичные числа: l0lb, -11000b; восьмеричные числа: 74q, -74q; шестнадцатеричные числа: lAFh, -lAFh.

Если число начинается с "буквенной" цифры (A-F), например A5h, тогда становится непонятным, что означает эта запись — число

или идентификатор. Чтобы не было путаницы, вводится следующее требование: если шестнадцатеричное число начинается с цифры A-F, то в начале числа обязательно должен быть записан хотя бы один незначащий нуль: 0A5h – число, A5h – идентификатор.

В числах малые и большие буквы отождествляются, поэтому буквы-спецификаторы (h, b и т. д.) и буквенные шестнадцатеричные цифры (A-F) можно записывать как малыми, так и большими буквами. Например, lAh, lah, 1aH и 1AH — это одно и то же число.

Символьные данные

Символы заключаются либо в одинарные, либо в двойные кавычки: 'A' или "A". Естественно, левая и правая кавычки должны быть одинаковыми: 'B" или "B' — ошибка.

Строки (последовательности символов) также заключаются либо в одинарные, либо в двойные кавычки: 'A+B' или "A+B".

Замечание:

- в качестве символов можно использовать русские буквы;
- в строках одноименные большие и малые буквы не отождествляются ('A+B' и 'a+b' — разные строки);
- если в качестве символа или внутри строки надо указать кавычку, то делается это так: если символ или строка заключена в одинарные кавычки, то одинарную кавычку надо удваивать, а вот двойную кавычку не надо удваивать, и наоборот, если внешние кавычки двойные, то двойная кавычка должна удваиваться, а одинарная не удваивается:

'don ' 't' 'don"t' "don't" "don""t"

Чтобы не удваивать внутренние кавычки, в язык введены два вида кавычек, ограничивающих символы и строки.

Предложения

Каждый сегмент содержит совокупность предложений языка, каждое из которых занимает отдельную строку кода программы. Пе-

реносить предложение на следующую строку или записывать в одной строке два предложения нельзя. Если в предложении более 131 символа, то 132-й и все последующие символы игнорируются.

При записи предложений действуют следующие правила расстановки пробелов:

• пробел обязателен между рядом стоящими идентификаторами и/или числами (чтобы отделить их друг от друга);

• внутри идентификаторов и чисел пробелы недопустимы; в остальных местах пробелы можно ставить или не ставить;

• там, где допустим один пробел, можно ставить любое число пробелов.

Замечание: эти правила не относятся к пробелам внутри строк, где пробел - обычный значащий символ.

Предложения ассемблера бывают четырех типов:

- *команды или инструкции*, представляющие собой символические аналоги машинных команд.
- *макрокоманды* — оформляемые определенным образом предложения текста программы, замещаемые во время трансляции другими предложениями;
- *директивы*, являющиеся указанием транслятору ассемблера на выполнение некоторых действий. У директив нет аналогов в машинном представлении;
- *строки комментариев*, содержащие любые символы, в том числе и буквы русского алфавита. Комментарии игнорируются транслятором.

Допустимыми символами при написании текста программ являются:

- все латинские буквы: A—Z, a—z. При этом заглавные и строчные буквы считаются эквивалентными;
- цифры от 0 до 9 ;

- знаки ? , @ , $, _ , & ;
- разделители: , . [] () <> { } + / * % ! “ ” ? \ = # ^.

Комментарии

Комментарии не влияют на смысл программы, при трансляции ассемблер игнорирует их. Они поясняют смысл программы.

Комментарием считается любая строка, начинающаяся со знака "точка с запятой" (перед ним может быть любое число пробелов) либо пустая строка (строка, в котором нет иных символов, кроме пробелов). В комментариях можно использовать любые символы, в том числе и русские буквы.

Пример 4.2. Комментариями являются 1-я и 3-я строки в следующем тексте:

```
; это  комментарий
ADD AX, 0
; MOV  BX, 2
```

В языке ассемблера допустим и многострочный комментарий. Он должен начинаться со строчки

COMMENT <маркер> <текст>

(COMMENT - это одна из директив языка ассемблера). В качестве маркера берется первый за словом COMMENT символ, отличный от пробела; этот символ начинает комментарий. Концом такого комментария считается конец первой из последующих строк программы, в которой (в любой позиции) снова встретился этот же маркер.

Пример 4.3.

```
COMMENT   *   все
это является
комментарием  *  и это тоже
```

Такой вид комментария обычно используется, когда надо (например, при отладке) временно исключить из программы некоторый ее фрагмент.

Команды

Предложения-команды — это символьная форма записи машинных команд. Общий синтаксис этого типа предложений таков:

[<метка>:] <мнемокод> [<операнды>] [**;**<комментарий>]

Пример 4.4.

```
LAB: ADD SI, 2 ; изменение индекса
NEG A
CBW
```

Мнемокод (мнемонический код) является обязательной частью команды. Это служебное слово, указывающее в символьной форме операцию, которую должна выполнить команда.

Операнды команды, если они есть, отделяются друг от друга запятыми. Операнды обычно записываются в виде выражений.

Директивы

Синтаксис директив следующий:

[<имя>] <название директивы> [<операнды>] [;<комментарий>]

Пример 4.5.

```
X  DB   10, -5, 0FFh ;   массив  X
```

Формат директив в целом совпадает с форматом команд. Единственное отличие — в директиве после имени, если оно есть, не ставится двоеточие.

Имя, указываемое в начале директивы, — это имя константы или переменной, описываемой данной директивой. В примере 4.5 X – это имя переменной-массива из трех элементов.

Названия директив, как и мнемокоды, — служебные слова. Заранее известно, какие служебные слова обозначают директивы, а какие – мнемокоды.

Остальные части директивы (операнды и комментарий) записываются так же, как и в командах.

Директивы определения данных

По директиве DB (define byte, определить байт) определяются данные размером в байт. Ее синтаксис (без учета возможного комментария в конце) таков:

[<имя>] DB <операнд> {,<операнд>}

Встречая такую директиву, ассемблер вычисляет операнды и записывает их значения в последовательные байты памяти. Первому из этих байтов дается указанное имя, по которому на этот байт можно ссылаться из других мест программы.

Существует два основных способа задания операндов директивы DB:

• ? (знак неопределенного значения), под переменную выделяется байт, в который ассемблер ничего не записывает, т. е. переменная не получает начального значения.

• константное выражение со значением от -128 до 255.

Остальные способы задания операндов производны от этих двух.

Пример 4.6. Если надо описать массив из 4 байтов с некоторыми начальными значениями, то это можно сделать так:

```
M  DB   2
DB   -2
DB  ?
DB   '*'
```

или

```
M DB  2,-2,?,'*'
```

Возможно еще одно сокращение в директиве DB: если в ней несколько соседних операндов — символы, то их можно объединить в одну строку.

Пример 4.7. Следующие две директивы эквивалентны:

```
S   DB   'a', 'b', 'c'                    S  DB   'abc'
```

Пример 4.8. Если нужно описать байтовый массив R из 8 элементов с начальным значением 0 для каждого из них, то это можно сделать так:

```
R DB 0,0,0,0,0,0,0,0
```

Эту директиву можно записать и короче:

```
R DB   8   DUP(0)
```

Директивой DW (define word, определить слово) описываются переменные размером в слово. Она аналогична директиве DB.

Существует два основных способа задания операндов директивы DB: операнд ? и константное выражение со значением от -32768 до 65535.

По директиве DD (define double word, определить двойное слово) описываются переменные, под которые отводятся двойные слова. Поэтому имена этих переменных имеют тип 4 или DWORD (значением этой стандартной константы как раз является число 4). Существует два основных способа задания операндов директивы DB: операнд ? и константное выражение со значением от -2^{15} до $2^{16}-1$

Директивы эквивалентности и присваивания

Константы описываются с помощью директивы эквивалентности — директивы EQU (equal, равно), имеющей следующий синтаксис:

<имя> EQU <операнд>

Здесь обязательно должно быть указано имя, должен быть и операнд, причем только один.

Директивой EQU автор программы заявляет, что указанному операнду он дает указанное имя, и требует, чтобы все вхождения этого имени в текст программы ассемблер заменял на этот операнд.

Пример 4.9. Если есть директива

```
STAR  EQU   '*'
```

то ассемблер будет рассматривать предложение

```
T  DB  STAR
```

как предложение

```
T  DB  '*'
```

т. е., указать имя STAR и указать '*' — это одно и то же.

Директива EQU носит чисто информационный характер, по ней ассемблер ничего не записывает в машинную программу. Поэтому директиву EQU можно ставить в любое место программы — и между командами, и между описаниями переменных, и в других местах.

Если в правой части директивы указано имя регистра, переменной, константы и т. п., тогда имя слева объявляется синонимом данного имени и все последующие вхождения в текст программы этого имени-синонима ассемблер будет заменять на имя, указанное справа.

Пример 4.10.

```
A  DW   ?
B  EQU A
C DW   B ;   эквивалентно:   C  DW  A
```

Если в правой части директивы EQU стоит константное выражение, тогда указанное слева имя принято называть именем константы. Также операндом может быть любой другой текст.

Пример 4.11.

```
S EQU 'Вы ошиблись'
```

Директива присваивания:

<имя> = <константное выражение>

Эта директива определяет константу с именем, указанным в левой части, и с числовым значением, равным значению выражения справа. Но в отличие от констант, определенных по директиве EQU, данная константа может менять свое значение, обозначая в разных частях текста программы разные числа.

Пример 4.12.

```
К=10
A  DW  К    ;  эквивалентно:  A  DW  10
К=К+4
B  DB  К    ;  эквивалентно:  B  DB  14
```

Выражения

Выражения используются для описания операндов команд. В языке ассемблера выражения делятся на два класса - на константные и адресные, в зависимости от типа их значений. Если значением выражения является целое число, то оно называется константным выражением, а если значением является адрес, то это адресное выражение.

Укажем старшинство всех операторов языка ассемблера (в порядке убывания; в каждой строке указаны операторы одного старшинства):

1. (), [], LENGTH, SIZE, WIDTH, MASK
2. .
3. :
4. PTR, OFFSET, SEG, TYPE, THIS
5. HIGH, LOW
6. одноместные + и -
7. *, /, MOD, SHL, SHR
8. двухместные + и -

9. EQ, NE, LT, LE, GT, GE

10. NOT

11. AND

12. OR, XOR

13. SHORT, .TYPE

Значениями константных выражений всегда являются 16-битовые целые числа (единственным исключением является директива DD, в которой можно явно указывать 32-битовые числа). При этом, если значением (в математическом смысле) константного выражения является отрицательное число, то в формируемую машинную программу оно записывается в дополнительном коде.

К простейшим константным выражениям относятся:

• число от -2^{15} до $2^{16}-1$ (числа вне этого диапазона рассматриваются как ошибка);

• символ (значением такого выражения является код символа; например, значением выражения '0' будет число 30h);

• строка из двух символов (значением является число-слово, составленное из кодов этих символов);

• имя константы (значением такого выражения является значение константы; например, если была директива K EQU 10, то значение K равно 10).

Среди константных операторов, т. е. операторов с числовым значением, отметим оператор типа TYPE (значение оператора является размер в байтах ячейки, выделенную под переменную с указанным именем, например, TYPE X=BYTE=1) и следующие арифметические операторы (k, kl и k2 означают любые константные выражения):

- одноместные плюс и минус: +k, -k
- операторы сложения и вычитания: kl+k2, kl-k2
- операторы умножения и деления: kl *k2, kl/k2, kl MOD k2

(* — умножение, / — деление нацело, MOD — взятие остатка от деления)

Операндами арифметических операторов должны быть константные выражения.

Значениями адресных выражений являются 16-битовые адреса. Все операции над адресами выполняются по модулю 2^{16} (l0000h).

К простейшим адресным выражениям относятся:

• метка (имя команды) и имя переменной, описанное в директиве DB, DW или DD (значениями таких выражений являются адреса меток и имен);

• счетчик размещения; он записывается как $ и обозначает адрес того предложения, в котором он встретился. При трансляции программы ассемблер следит за тем, в ячейку с каким адресом должен попасть машинный эквивалент очередного предложения программы на языке ассемблера; так вот, если мы хотим сослаться на этот адрес, то и надо указать $. Отсюда следует, что в разных предложениях $ обозначает разные адреса.

Пример 4.13. Если адрес переменной A равен l00h, то имеем:

```
A DW $  ; эквивалентно A DW l00h
 B DW $  ; эквивалентно B DW 102h
```

4.2. Команды пересылки

Команда MOV

Пересылка (move): MOV opl, op2

По команде MOV на место первого операнда пересылается значение второго операнда: opl:=op2. Флаги эта команда не меняет.

Пример 4.14.

```
MOV AX,500      ; A:=500
MOV BL,DH       ; BL:=DH
```

В команде MOV допустимы следующие комбинации операндов:

op1	*op2*	
r8	i8, r8, m8	пересылка байтов
m8	i8, r8	
r16	i16, r16, sr, m16	пересылка слов
sr (кроме CS)	r16, m16	
m16	i16, r16, sr	

Команда работы с адресами: lea назначение, источник — загрузка эффективного адреса. Команда lea похожа на команду mov тем, что она также производит пересылку. Но команда lea производит пересылку не данных, а эффективного адреса данных (то есть смещения данных относительно начала сегмента данных) в регистр, указанный операндом назначение.

Оператор указания типа (PTR)

Оператор указания типа PTR (от pointer, указатель) записывается следующим образом:

<тип> PTR <выражение>

где <тип> – это BYTE, WORD или DWORD, а выражение может быть константным или адресным.

Если указано константное выражение, то оператор "говорит", что значение этого выражения (число) должно рассматриваться ассемблером как величина указанного типа (размера); например, BYTE PTR 0 – это нуль как байт, а WORD PTR 0 - это нуль как слово (запись BYTE PTR 300 ошибочна, т. к. число 300 не может быть байтом). В этом случае оператор PTR относится к константным выражениям.

Если же в PTR указано адресное выражение, то оператор "говорит", что адрес, являющийся значением выражения, должен восприниматься ассемблером как адрес ячейки указанного типа (размера); например: WORD PTR A — адрес A обозначает слово (байты с

адресами A и A+1). В данном случае оператор PTR относится к адресным выражениям.

Команда XCHG

Перестановка (exchange): XCHG opl, op2

Эта команда меняет местами значения своих операндов (они должны быть либо байтами, либо словами): op1 <=> op2. Флаги при этом не меняются.

Пример 4.15.

```
MOV AX,62    ; AX=62
MOV SI,135   ;  SI=135
XCHG AX, SI  ; AX=135, SI=62
```

Допустимые типы операндов команды XCHG:

op1	*op2*	
r8	r8, m8	перестановка байтов
m8	r8	
r16	r16, m16	перестановка слов
m16	r16	

Не допускается перестановка содержимого двух ячеек памяти. Если все-таки надо сделать такую перестановку, то это реализуется через какой-нибудь регистр.

Пример 4.16. Поменять местами значения байтовых переменных X и Y можно так:

```
MOV AL,X     ;AL=X
XCHG AL,Y    ;AL=Y, Y=X
MOV  X,AL    ;X=Y (исходное значение)
```

4.3. Команды сложения и вычитания

Особенности сложения и вычитания целых чисел в ПК

Беззнаковые числа складываются как обычно, только в двоичной системе счисления. Однако здесь есть одна проблема: что делать,

если сумма получится очень большой - такой, что она не умещается в ячейке? Например, если при ячейке размером в 8 битов мы складываем 250 и 10, то получим число 260 (l0000l00b), которое не "влезает" в ячейку.

В ПК реакция иная: ошибка не фиксируется, левая единица (единица переноса) отбрасывается и в качестве ответа выдается искаженная сумма (в нашем примере ответом будет байт 00000100b, т. е. число 4), Но зато в флаг переноса CF записывают 1. Это сигнал о том, что получилась неправильная сумма (если переноса не было, то в CF записывают 0). Затем можно проанализировать этот флаг и "поймать" такую ошибку.

Такое суммирование с отбрасыванием единицы переноса в математике называется суммированием по модулю 2^k (k - размер ячейки), при этом в флаге CF фиксируется, был ли перенос:

$$\text{сумма}(x,y)=(x+y) \bmod 2^k=\begin{cases} x+y, & \text{если } x+y< 2^k,\ CF=0 \\ x+y-2^k, & \text{если } x+y>=2^k,\ CF=1 \end{cases}$$

При вычитании беззнаковых целых чисел также возникает проблема: что делать, если при вычитании x-y число x меньше числа y? Ведь в этом случае получится отрицательная разность, а это уже вне области беззнаковых чисел.

В ПК поступают следующим образом: при x>=y выполняется обычное вычитание, но если x<y, тогда числу x дается "заем" единицы (к числу x прибавляется величина 2^k) и только после этого производится вычитание. Полученное таким образом число и объявляется разностью.

Пример 4.17. При k=8 вычитание 1-2 происходит таким образом:

1-2 => (2^8+1)-2 = (256+1)-2 = 257-2 = 255 (в двоичной системе замена 1 на 256+1 - это замена 00000001 на 100000001, т. е. приписывание 1 слева) и именно число 255 объявляется результатом вычитания 1-2. При этом ошибка не фиксируется, зато в флаг переноса CF заносится 1, что сигнализирует о заеме единицы, о неправильном результате (при x>=y в CF заносится 0).

Выполняемое так вычитание в математике называют вычитанием по модулю 2^k, при этом фиксируется, был ли заем:

$$\text{разность}(x,y)=(x-y) \bmod 2^k=\begin{cases} x-y, & \text{если } x>=y, \ CF=0 \\ (2^k+x)-y, & \text{если } x<y, \ CF=1 \end{cases}$$

В ПК сложение и вычитание беззнаковых целых чисел — это на самом деле сложение и вычитание по модулю 2^k, где k - размер ячеек. Причем появление после операции значения 1 в флаге переноса CF свидетельствует о том, что выданный ответ неправильным.

Рассмотрим сложение и вычитание знаковых чисел. Если целые со знаком представлены в дополнительном коде, то складывать и вычитать их можно по алгоритмам для беззнаковых чисел. Делается это так: дополнительные коды знаковых операндов рассматривают как числа без знака и в таком виде их складывают или вычитают, а полученный результат затем рассматривают как дополнительный код знакового ответа.

Пример 4.18 (при ячейке в 8 битов). Пусть надо сложить +3 и -1. Их дополнительные коды - это 3 и (256-1)=255. Складываем их как числа без знака: 3+255 (mod 256) = 258 (mod 256) = 2. Теперь величина 2 рассматривается как дополнительный код ответа, поэтому получается ответ +2.

Пример 4.19. Пусть надо сложить -3 и +1. Дополнительные коды этих знаковых чисел: (256-3)=253 и 1. Складываем их

как беззнаковые числа: 253+1 (mod 256) = 254. Теперь, рассматривая эту величину как дополнительный код ответа, получаем результат -2 (254=256-2).

Из сказанного следует, что в ЭВМ, где знаковые числа представляются в дополнительном коде, не нужны разные машинные команды для сложения и вычитания беззнаковых и знаковых чисел, достаточно и одного набора этих команд.

Не все так просто при сложении и вычитании знаковых чисел, здесь есть свои неприятности. Напомним, что при размере ячеек в 8 битов в дополнительном коде представляются только числа от -128 до +127.

Пример 4.20. Рассмотрим сложение знаковых чисел +127 и +2. Складывая их как беззнаковые числа 127 и 2, получаем величину 129, которую теперь надо рассмотреть как дополнительный код ответа: поскольку 129=256-127, то суммой должно быть признано число -127. Таким образом, складывая два положительных числа, мы получили отрицательное число.

При представлении чисел (размером в байт) в дополнительном коде левый разряд является знаковым, а на модуль числа отводится 7 правых разрядов. У нас же получился ответ 129=10000001b, модуль которого не вмещается в эти 7 разрядов, поэтому модуль и "залез" в знаковый разряд, изменив его на противоположный.

Такое налезание модуля (мантиссы, цифровой части) числа на знаковый разряд называют "переполнением мантиссы". В общем случае оно происходит, если складываются числа одного знака и настоящая сумма оказывается вне диапазона представимых знаковых чисел ([-128, +127] при k=8). Переполнение мантиссы фиксируется в флаге переполнения OF: он получает значение 1, если было переполнение, и значение 0 иначе. Таким образом, при OF=0 результат правильный, а

при OF=1 - неправильный, однако эта ошибка не фиксируется и "поймать" ее можно только последующим анализом флага OF.

Аналогичное переполнение мантиссы возможно и при вычитаний.

Пример 4.21. При вычитании (+127)-(-2)=127+2 получаем 129, а это дополнительный код числа -127, которое и выдается как результат вычитания, хотя истинной разностью является число 129.

В общем случае переполнение мантиссы происходит, если вычитаются числа разных знаков и настоящая разность оказалась вне диапазона представимых знаковых чисел. И здесь факт переполнения фиксируется в флаге OF: он получает значение 1, если было переполнение мантиссы и результат операции неправильный, и значение 0, если не было переполнения и ответ правильный.

При сложении и вычитании как беззнаковых, так и знаковых чисел возможны особые *случаи,* когда настоящий (в математическом смысле) результат выходит за диапазон представимых чисел, и тогда результат искажается. Такое искажение результата фиксируется в флагах CF и OF. Распознать такую ошибку можно лишь последующим анализом этих флагов.

Команды сложения и вычитания

В ПК имеется несколько команд сложения и вычитания. Основными из них являются следующие:

Сложение: ADD opl, op2

Вычитание (subtract): SUB opl, op2

В этих командах допустимы следующие комбинации операндов:

op1	*op2*	
r8	i8, r8, m8	сложение/вычитание байтов
m8	i8, r8	
r16	i16, r16, m16	сложение/вычитание слов
m16	i16, r16	

Команда ADD складывает операнды и записывает их сумму на место первого операнда: opl:=opl+op2. По команде SUB из первого операнда вычитается второй операнд и полученная разность записывается вместо первого операнда: opl:=opl-op2.

Пример 4.22.

```
ADD AH,12      ;AH:=AH+12
SUB  SI, Z     ;SI:=SI-Z
ADD   Z,-300   ;Z:=Z+(-300)
```

Следующая пара команд сложения и вычитания:

Увеличение на 1 (increment): INC op

Уменьшение на 1 (decrement): DEC op

В этих командах допустимы следующие типы операнда: r8, m8, r16, m16.

Команда INC аналогична команде ADD op,l, т. е. увеличивает свой операнд на 1: opl:=opl+l, а команда DEC аналогична команде SUB op,l, т. е. уменьшает операнд на 1: opl:=opl-l (единственное отличие: команды INC и DEC не меняют флаг переноса CF).

Еще одна команда из группы сложения и вычитания:

Изменение знака (negative): NEG op

Допустимые типы операнда этой команды: r8, m8, r16, m16.

Команда NEG рассматривает свои операнд как число со знаком и меняет его знак на противоположный: op:=-op.

Пример 4.23.

```
MOV  AH,1
NEG AH     ;AB:=-1   (OFFh)
```

Еще одна пара команд сложения и вычитания:

Сложение с учетом переноса (add with carry): ADC opl, op2

Вычитание с учетом заема (subtract with borrow): SBB opl, op2

Допустимые типы операндов — как в командах ADD и SUB. Эти команды аналогичны командам обычного сложения и вычитания (ADD и SUB) за одним исключением - в команде ADC к сумме операндов еще прибавляется значение флага переноса CF: opl:=opl+op2+CF, а в команде SBB из разности операндов еще вычитается значение этого флага: opl:=opl-op2-CF.

4.4. Команды умножения и деления

Команды умножения

Умножение целых без знака (multiply): MUL op

Умножение целых со знаком (integer multiply): IMUL op

В остальном эти команды действуют одинаково:

Умножение байтов: AX:=AL*op (op: r8, m8)

Умножение слов: (DX,AX) :=AX*op (op: r16, m16)

Операнд op, указываемый в команде, — это лишь один из сомножителей; он может находиться в регистре или в памяти, но не может быть непосредственным операндом. Местонахождение другого сомножителя фиксировано и потому в команде не указывается явно: при умножении байтов он берется из регистра AL, а при умножении слов - из регистра AX.

Местонахождение результата умножения также заранее известно и потому в команде явно не указывается. При этом под результат отводится в два раза больше места, чем под сомножители. Это связано с тем, что умножение *n*-значных чисел в общем случае дает произведение из 2n цифр, и с желанием сохранить все цифры произведения. При умножении байтов результат имеет размер слова и записывается в весь регистр AX (в AH - старшие цифры произведения, в AL - младшие), а при умножении слов результат имеет размер двойного

слова и записывается в два регистра - в регистр DX заносятся старшие цифры произведения, а в регистр AX - младшие цифры.

Пример 4.24.

```
N  DB 10
…
MOV AL,2
MUL  N            ; AX=2*10=20=0014h:  AH=00h,  AL=14h    .
MOV AL,26
MUL N             ;  AX=26*10=260=0104h:  AH=01h,  AL=04h
MOV AX,8
MOV  BX,-1
IMUL  BX          ;(DX,AX)=-8=OFFFFFFF8h:DX=OFFFFh,
X=OFFF8h
```

Если CF=OF=1 — произведение занимает двойной формат; если CF=OF=0 — произведению достаточен формат сомножителей; при CF=0 (одновременно и OF=0) можно считать, что произведение байтов занимает только регистр AL, а произведение слов — только регистр AX, и дальше можно работать только с этими регистрами. Но если CF=1, то далее приходится работать с произведением как с числом удвоенного формата.

Команды деления

Деление чисел без знака и со знаком также реализуется двумя командами:

Деление целых без знака (divide): DIV op

Деление целых со знаком (integer divide): IDIV op

Первая из этих команд предназначена для деления беззнаковых целых чисел, а вторая — для деления знаковых чисел, в остальном же эти команды действуют одинаково:

Деление слова на байт:

AH:=AX mod op, AL:=AX div op (op: r8, m8)

деление двойного слова на слово:

DX: = (DX,AX) mod op, AX: = (DX,AX) div op (op: r16, m16)

При выполнении команды деления возможно появление ошибки с названием "деление на 0 или переполнение". Она возникает в двух случаях:

- делитель равен 0 (op=0),
- неполное частное не вмещается в отведенное ему место (регистр AL или AX).

Пример 4.25.

```
MOV AX, 600
MOV BH, 2
DIV BH    ;  600 div 2 = 300, но 300 не вмещается в AL
```

При такой ошибке ПК прекращает выполнение программы.

Замечание: для обработки вещественных данных (чисел с плавающей запятой) служит специальное устройство — математический сопроцессор (FPU — Floating Point Unit).

Команды преобразования типов

Расширение байта до слова (convert byte to word): CBW

Расширение слова до двойного (convert word to double): CWD

Существуют также следующие виды команд преобразования типа:

- cwde (Convert Word to Double) — команда преобразования слова (в регистре ax) в двойное слово (в регистре eax) путем распространения значения старшего бита ax на все биты старшей половины регистра eax;
- cdq (Convert Double Word to Quarter Word) — команда преобразования двойного слова (в регистре eax) в учетверенное слово (в ре-

гистрах edx:eax) путем распространения значения старшего бита eax на все биты регистра edx.

Команды работы со стеком: **push** источник — запись значения источник в вершину стека; pop назначение — запись значения из вершины стека по месту, указанному операндом назначение. Значение при этом “снимается” с вершины стека.

Пример 4.26. Написать программу на языке ассемблера для вычисления целочисленного выражения (2*c+d-52)/(a/4+1), для исходных данных a, c, d длиной в байт.

```
model small
.data
a db 4
d db 6
c db 25
m dw ?
.stack
db 256 dup ('?')
.code
begin:
mov ax,@data
mov ds,ax
mov al,c
mov bl,2
imul bl
mov dl,d
add al,dl
sub al,52
jnc n1
neg al
```

```
    n1:
    cbw
    push ax
    mov al,a
    cbw
    mov bl,4
    idiv bl
    inc al
    mov bl,al
    pop ax
    idiv bl
    mov m,ax
    exit:
    mov ax,4c00h
    int 21h
end begin
```

4.5. Понятие о макропрограммировании

Макропрограммирование (macro programming) — разработка программ с использованием макросредств.

Макросредства включают *макроязык,* основа которого — средства определения и использования макрокоманд и средства *макрогенерации;* с их помощью макрокоманды превращаются в макрорасширения — последовательность предложений, вставляемых в программу вместо макрокоманды.

Макрокоманда *(macros, macro, instruction)* — специфическое понятие, зависимое от контекста использования (языков программирования, инструментальной среды работы конечного пользователя). Макрокоманда (*macros, macro, instruction*) — предложение языка,

идентифицирующее набор простейших команд (инструкций). Макрокоманда используется со списком параметров. В определенном смысле макропрограмма — минимально возможный программный модуль, или *макромодуль.*

Расширение языка ассемблера за счет макросредств называют макроязыком.

Программа, написанная на макроязыке, транслируется в два этапа. Сначала она переводится на, так сказать, чистый язык ассемблера, т. е. преобразуется к виду, где нет никаких макросредств. Этот этап называется этапом макрогенерации, его осуществляет специальный транслятор — макрогенератор. На втором этапе полученная программа переводится на машинный язык. Это этап ассемблирования, его осуществляет ассемблер.

Трансляция программы, написанной на макроязыке, занимает больше времени, чем трансляция программы на "чистом" языке ассемблера. Но зато макроязык предоставляет больше возможностей, чем просто язык ассемблера, и потому писать программы, особенно большие, на макроязыке проще.

Макрогенератор и ассемблер могут взаимодействовать двояко. Во-первых, они могут действовать независимо друг от друга: сначала текст программы обрабатывает макрогенератор и только затем начинает работать ассемблер. Такой способ взаимодействия достаточно прост для понимания и для реализации и используется довольно часто, однако у него есть серьезный недостаток: макрогенератор, работая до ассемблера, не может воспользоваться информацией, извлекаемой из текста программы ассемблером.

Во-вторых, макрогенератор и ассемблер могут действовать совместно, чередовать свою работу: первым каждое предложение программы просматривает макрогенератор; если это конструкция соб-

ственно макроязыка, то макрогенератор соответствующим образом преобразует ее в группу предложений "чистого" языка ассемблера и тут же передает их на обработку ассемблеру, а если это обычное предложение "чистого" языка ассемблера, то макрогенератор сразу передает ее ассемблеру. В таких условиях в конструкциях макроязыка уже можно ссылаться на объекты (например, константы) "чистого" языка ассемблера.

При "чередующемся" способе взаимодействия макрогенератора и ассемблера они фактически являются не двумя независимыми трансляторами, а частями одного транслятора, который принято называть макроассемблером.

Именно такой макроассемблер используется в рассматриваемой ранее системе MASM. Этим и объясняется полное название ее входного языка — язык макроассемблера (macroassembler language, MASM).

5. Функционально-структурная организация ПК

5.1. Взаимодействие компонентов компьютера через магистраль (шину). Типы шин, реализованных в ПК

Под *магистралью* (или *шиной*) — применительно к информационным системам *информационной магистралью* — понимается совокупность электрических связей и обслуживающих электронных приборов (обрамления), обеспечивающих обмен информацией между двумя или более устройствами.

В состав магистрали входит множество электрических связей – линий магистрали, — по каждой из которых может передаваться двоичный сигнал, т.е. сигнал, имеющий два четко различимых уровня – "лог. 0" и "лог. 1". Часто сигналы передаются по нескольким линиям одновременно и воспринимаются в совокупности. В таком случае говорят о *параллельной* передаче двоичных сигналов. Например, можно по восьми линиям параллельно передать один 8-разрядный байт информации.

Магистраль, связывающая основные компоненты (компоненты высшего уровня иерархии — процессор, память, модули ввода-вывода), называется *системной магистралью* (*system bus*). Большинство распространенных на сегодняшний день вариантов структуры подсистемы взаимодействия компонентов компьютерной системы основано на использовании одной или даже нескольких системных магистралей.

Системная магистраль включает от 50 до 100 отдельных линий, по каждой из которых передаются сигналы определенного функционального назначения. Хотя существует множество вариантов конструкции системной магистрали, линии любой магистрали можно

разделить на три больших группы: линии данных, адреса и управляющих сигналов. Кроме того, в состав магистрали могут входить и силовые линии питания, обеспечивающие энергоснабжение подключенных модулей.

По *линиям данных* передаются сигналы, соответствующие информации, передаваемой одним из подключенных модулей. В совокупности линии данных называются *магистралью данных.* Как правило, количество линий магистра данных кратно 8. Этот параметр принято называть *разрядностью* или *шириной* (*width*) магистрали.

По *линиям адреса* передаются сигналы, которые в совокупности определяют источник или приемник данных, выставленных на линии данных. Если, например, процессор намеревается прочесть слово (8-, 16- или 32-разрядное) из определенной ячейки памяти, он выставляет двоичный код адреса этой ячейки на линии адреса. *Разрядность магистрали адреса* определяет максимальный объем памяти, адресуемой по такой магистрали. Порты модулей ввода-вывода адресуются по этим же линиям адреса. Как правило, формат кода адреса порта выбирается таким образом, что старшие разряды задают определенный модуль ввода-вывода из подключенных к магистрали, а младшие — адрес в памяти или порт в модуле ввода-вывода. Например, если магистраль адреса 8-разрядная, то коды 0111111 и меньшие задают адреса ячеек в модуле памяти (модуле с адресом 0), а коды, начиная с 10000000, задают порты устройств ввода-вывода, находящихся под управлением модуля ввода-вывода (модуля с адресом 1). Линии управляющих сигналов используются для управления доступом к линиям данных и адреса. Управляющие сигналы несут определенную информационную нагрузку, т.е. передают команды, предписывающие, как поступать с прочими сигналами в данной ситуации, и синхронизируют процессы в магистрали. Сигналы синхронизации определяют

те моменты (или интервалы) времени, когда код, выставленный на линии данных и адреса, можно считать достоверным. Управляющие сигналы задают выполняемые операции. Типовой набор управляющих сигналов следующий:

- Memory write (запись в память) — инициирует запись кода, выставленного на линии данных в память по адресу, код которого выставлен на линии адреса;
- Memory read (чтение из памяти) — инициирует чтение из ячейки памяти по адресу, код которого выставлен на линии адреса, прочитанный код выставляется на линии данных;
- I/O write (запись в устройство ввода-вывода) — инициирует запись кода, выставленного на линии данных в устройство ввода-вывода, код адреса которого (адрес модуля ввода-вывода и номер порта устройства) выставлен на линии адреса;
- I/O read (чтение из устройства ввода-вывода) — инициирует считывание из порта устройства ввода-вывода, код адреса которого (адрес модуля ввода-вывода и номер порта устройства) выставлен на линии адреса, прочитанный код выставляется на линии данных;
- Transfer ACK (подтверждение передачи) — сигнализирует, что данные считаны с шины модулем, которому они предназначались командой записи, или выставлены на шину модулем, от которого они затребованы командой чтения;
- Bus request (запрос управления магистралью) — сигнализирует, что какой-то модуль просит предоставить в его распоряжение ресурсы магистрали;
- Bus grant (предоставление управления магистралью) — сигнализирует, что модулю, передавшему сигнал Bus request, предоставлено право распоряжаться ресурсами магистрали;

• Interrupt request (запрос прерывания) — сигнализирует, что произошло событие, требующее прерывания текущей программы;

• Interrupt ACK (подтверждение прерывания) — сигнализирует, что запрос прерывания воспринят;

• Clock (синхронизация) — используется для синхронизации выполняемых операций;

• Reset (общий сброс) — используется для установки всех модулей, подключенных к магистрали, в исходное состояние.

Функционирует магистраль следующим образом. Когда один из модулей "желает" передать данные другому, он должен выполнить две операции:

1) получить право пользоваться магистралью;

2) передать данные по магистрали.

Когда какому-либо модулю необходимо получить данные от другого модуля, он также должен выполнить две операции:

1) получить право пользоваться магистралью;

2) передать запрос другому модулю, выставив соответствующий код на адресных линиях и сформировав сигналы на определенных линиях управляющих сигналов.

Физически (конструктивно) системная магистраль представляет собой множество проводников, подключенных к одноименным контактам всех разъемов, в которые вставляются платы модулей. Как правило, проводники выполняются на многослойной печатной плате (ее иногда называют *генпанелъю*), в которую впаиваются разъемы для модулей. Вся конструкция (генпанель и вставленные в нее платы модулей) закрывается общим кожухом.

Такая конструкция получила наибольшее распространение. При комплектации небольших компьютеров обычно остаются свободные разъемы, что позволяет в дальнейшем наращивать их функци-

ональные возможности, вставляя в эти разъемы дополнительные модули памяти или модули управления дополнительными устройствами ввода-вывода. Если какой-либо модуль выходит из строя, его довольно просто извлечь и заменить новым, исправным.

В ПК используют несколько магистралей, организуя их по иерархическому принципу. Используется структура компьютера с несколькими магистралями, показанная на рис. 5.1, а). Она включает *локальную магистраль,* объединяющую процессор, кэш-память и одно или несколько локальных периферийных устройств. Контроллер кэш-памяти обеспечивает связь кэша не только с локальной магистралью, но и с системной магистралью, к которой подключены все модули оперативной памяти. Такой вариант использования кэш-памяти позволяет переключить оперативную память с локальной магистрали на системную. При такой структуре все процессы передачи данных из устройств ввода-вывода в память (и наоборот) идут через системную магистраль в обход процессора, что позволяет ему заниматься своим делом с максимальной отдачей.

Хотя модули ввода-вывода можно и непосредственно подключить к системной магистрали, более эффективно использовать для этого дополнительную *магистраль расширения (Expansion Bus).* Такая структура позволяет разделить информационные потоки "процессор—оперативная память" и "оперативная память—контроллеры ввода-вывода".

На схеме рис. 5.1, а) представлены типичные модули ввода-вывода современных компьютерных систем, которые подключаются к магистрали расширения: адаптеры ЛВС (локальной вычислительной сети) Ethernet с пропускной способностью до 10 Мбит/с и адаптеры глобальных сетей, в частности сетей, работающих в режиме переключения пакетов. Интерфейс SCSI (small computer system interface – ин-

терфейс малых вычислительных систем) сам по себе также представляет магистраль, которая используется для поддержки обмена данными с локальными дисками и другими периферийными устройствами. Порт последовательного обмена используется для работы с принтером или сканером.

Чаще используют еще одну высокоскоростную магистраль, которая объединяет периферийные устройства с высоким быстродействием и связывается с системной магистралью через специальный *модуль согласования* (*bridge*). Такую структуру в литературе иногда называют *мезонинной архитектурой* (*mezzanine architecture*).

Типичный пример реализации такого подхода представлен на рис. 5.1, б). Как и в традиционном варианте, здесь имеется локальная магистраль, объединяющая процессор и контроллер кэш-памяти, который, в свою очередь, связан с модулями оперативной памяти через системную магистраль. Контроллер кэш-памяти составляет единый модуль с согласователем высокоскоростной магистрали, оснащенным буферной памятью. Высокоскоростная магистраль обеспечивает связь с контроллерами периферийных высокопроизводительных устройств – адаптерами ЛВС, например Fast Ethernet с пропускной способностью до 100 Мбит/с, видеоконтроллерами и профессиональными контроллерами устройств вывода графической информации рабочих станций, контроллерами высокоскоростных локальных интерфейсов типа SCSI и FireWire. Последний представляет собой специализированную магистраль для поддержки функционирования внешних устройств большой информационной емкости. Для связи с внешними устройствами, обладающими низкой производительностью, по-прежнему используется магистраль расширения, подключаемая к высокоскоростной магистрали через специальный модуль расширения, оснащенный буферной памятью.

Преимущество такой структуры состоит в том, что периферийные устройства с высокой производительностью довольно тесно интегрируются с процессором и в то же время могут функционировать самостоятельно, не загружая процессор.

Все события, происходящие с сигналами на линиях магистрали, должны быть каким-то образом скоординированы во времени — *синхронизированы.* Существует два подхода к координации процессов во временной области — синхронный и асинхронный.

Синхронный метод координации предполагает, что любое событие совершается синхронно со специальными импульсами отметок времени — *clock* (чаще всего их называют *тактовыми импульсами*). При *асинхронном методе* координации каждое очередное событие происходит после завершения предыдущего.

Синхронную магистраль легче реализовать и протестировать, но она хуже работает с модулями разной производительности, поскольку все они должны быть "привязаны" к тактовым импульсам. При использовании асинхронной магистрали в систему можно без особых опасений включать модули разного быстродействия.

В компьютере реализовано несколько типов шин.

• *Шина процессора.* Эта высокоскоростная шина является ядром набора микросхем и системной платы. Используется для передачи данных между процессором и основной системной шиной или между процессором и внешней кэш-памятью. В системах на базе процессоров Pentium II эта шина работает на частоте 66, 100, 133 или 200 МГц и имеет ширину 64 разряда.

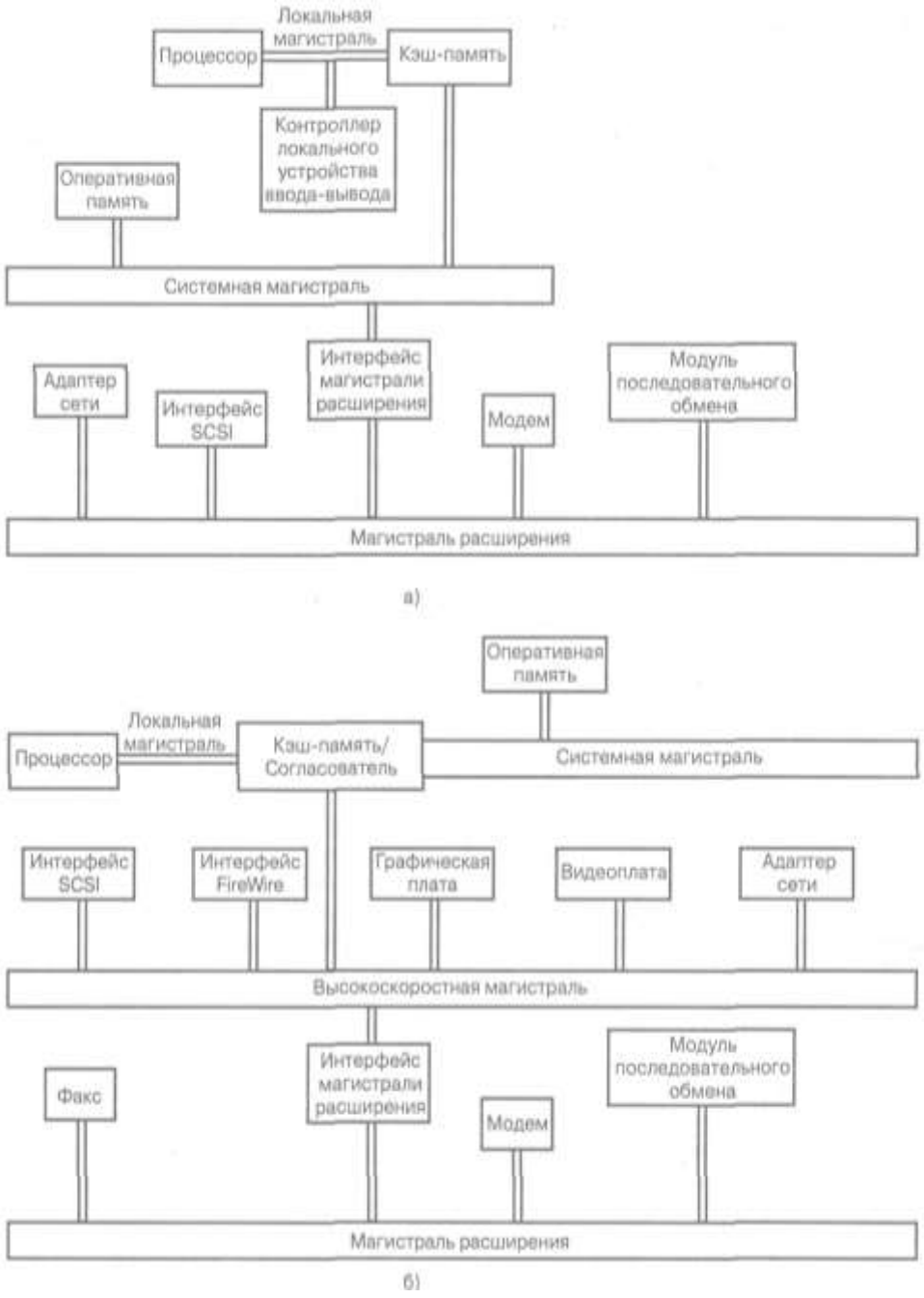


Рис. 5.1. *Структура компьютера с несколькими магистралями*:
а) *традиционная*; *б*) *обладающая повышенной производительностью*

Когда говорят о шине процессора, чаше всего имеют в виду *шину данных*, представленную как набор соединений (или выводов) для передачи или приема данных. Современные процессоры типа Pentium имеют 64-разрядные внешние шины данных. Это означает, что процессоры Pentium могут передавать в системную память (или получать из нее) одновременно 64 бита данных.

Поскольку шина процессора должна обмениваться информацией с процессором с максимально высокой скоростью, в компьютере она функционирует намного быстрее любой другой шины. Сигнальные линии (линии электрической связи), представляющие шину, предназначены для передачи данных, адресов и сигналов управления между отдельными компонентами компьютера. Например, в компьютере с процессором Pentium шина состоит из 64 линий данных, 32 линий адреса и соответствующих линий управления. Компьютеры с процессорами Pentium Pro и Pentium II имеют по 36 линий адреса.

Тактовая частота, используемая для передачи данных по шине процессора, соответствует его внешней частоте.

В процессорах Pentium 133, 166, 200 и 233 шина работает на тактовой частоте 66,6 МГц. В большинстве современных компьютеров соотношение частоты процессора и частоты шины соответствует одному из коэффициентов: 1,5x, 2x, 2,5x, 3x и т.д.

Для определения скорости передачи данных по шине процессора необходимо умножить разрядность шины данных (64 для Pentium, Pentium Pro или Pentium II) на тактовую частоту шины (она равна базовой (внешней) тактовой частоте процессора).

Эта величина характеризует *скорость передачи данных,* называемую также *полосой пропускания шины,* и является максимальной.

• *Шины ввода-вывода*. Шина ввода-вывода позволяет процессору взаимодействовать с периферийными устройствами. Эта шина и подключенные к ней разъемы расширения предназначены для того, чтобы компьютер мог выполнить все предъявляемые запросы. Шина ввода-вывода позволяет подключать к компьютеру дополнительные устройства для расширения его возможностей. В разъемы расширения устанавливают такие жизненно важные узлы, как контроллеры накопителей на жестких дисках и платы видеоадаптеров; к ним можно

подключить и более специализированные устройства, например звуковые платы, сетевые интерфейсные платы, адаптеры SCSI и др.

Шины ввода-вывода различаются архитектурой. Основными являются:

• ISA (Industry Standard Architecture);

• MCA (Micro Channel Architecture);

• EISA (Extended Industry Standard Architecture);

• VESA (также называемая VL-Bus или VLB (VESA — Video Electronic Standards Association));

• локальная шина PCI;

• AGP;

• PC Card (или PCMCIA);

• FireWire (IEEE-1394);

• USB (Universal Serial Bus).

Шины ISA, MCA и EISA имели один общий недостаток — сравнительно низкое быстродействие. Поэтому наилучшим подход к решению этой проблемы явилось размещение дополнительных разъемов ввода-вывода на самой быстродействующей шине, т.е. на шине процессора. Такая конструкция получила название *локальной шины* (Local Bus), поскольку внешние устройства (платы адаптеров) теперь имеют доступ к шине.

Однако согласно спецификации PC 99 в компьютерах должны использоваться только две шины — PCI и AGP.

Шина PCI (Peripheral Component Interconnect bus - шина взаимосвязи периферийных компонентов). Эта 32-разрядная шина работает на частоте 33 МГц; используется начиная с систем на базе процессоров 486. В настоящее время есть реализация этой шины с частотой 66 МГц. Находится под управлением контроллера PCI — части компонента North Bridge или Memory Controller Hub (MCH) набора мик-

росхем. На системной плате устанавливаются разъемы, обычно четыре или более, в которые можно подключать SCSI-, сетевые и видеоадаптеры, а также другое оборудование, поддерживающее этот интерфейс. К шине PCI подключается компонент South Bridge набора микросхем, который содержит реализации интерфейса IDE и USB (Universal Serial Bus).

Она обладает высокой пропускной способностью (528 Мбайт/с или 4,224 Гбит/с) и используется как в качестве высокоскоростной магистрали, так и в качестве дополнительной шины, объединяющей периферийные устройства. Достоинства магистрали PCI особенно проявляются в компьютерах, включающих множество высокоскоростных периферийных устройств (графических адаптеров, контроллеров локальной сети, контроллеров дисков и т. д.).

Шина AGP (Accelerated Graphics Port – ускоренный графический порт). Эта 32-разрядная шина работает на частоте 66 (AGP 1x), 133 (AGP2x) или 266 МГц (AGP 4x) и предназначена для подключения видеоадаптера. Она подключается к компоненту North Bridge или Memory Controller Hub (MCH) набора микросхем системной логики.

Шина ISA (*Industry Standard Architecture*). Это 16-разрядная шина, работающая на частоте 8 МГц; впервые стала использоваться в системах AT в 1984 году (была 8-разрядной и работала на частоте 5 МГц). Имела широкое распространение до настоящего времени, но из спецификации PC99 исключена. Реализуется с помощью компонента South Bridge. Чаще всего к этой шине подключается микросхема Super I/O.

5.2. Структура процессора

«Мозгом» персонального компьютера является *микропроцессор*, или *центральный процессор— CPU* (*Central Processing Unit*).

Микропроцессор выполняет вычисления и обработку данных (за исключением некоторых математических операций, осуществляемых в компьютерах, имеющих сопроцессор) и, как правило, является самой дорогостоящей микросхемой компьютера. Это функционально законченное программно-управляемое устройство обработки информации, выполненное в виде одной или нескольких больших (БИС) или сверхбольших (СБИС) интегральных схем.

Во всех PC-совместимых компьютерах используются процессоры, совместимые с семейством микросхем Intel, но выпускаются и проектируются они как самой фирмой Intel.

Звездный час фирм Intel и Microsoft наступил в 1981 году, когда IBM выпустила первый персональный компьютер IBM PC с процессором Intel 8088 (4,77 МГц) и операционной системой Microsoft Disk Operating System (DOS) версии 1.0. С этого момента практически во все персональные компьютеры устанавливаются процессоры фирмы Intel и операционные системы Microsoft.

При обработке машинной команды процессор выполняет следующие операции:

- *извлечение команды* — процессор считывает команду из памяти;
- *интерпретация команды* — процессор расшифровывает команду и определяет, какие операции ему предстоит выполнить;
- *извлечение данных* — для выполнения команды может потребоваться прочесть данные из памяти или из модуля ввода-вывода;
- *обработка данных* — выполнение команды может потребовать преобразования данных, т.е. выполнения над ними определенных арифметических или логических операций;
- *запись данных* — если в процессе выполнения команды данные были изменены, результат необходимо где-то зафиксировать. В частно-

сти, результат может быть записан в память или передан в модуль ввода-вывода.

Схема процессора представлена на рис. 5.2. На ней обозначены пути передачи данных и сигналов управления, включая и структурный компонент, обозначенный как *внутренняя магистраль ЦП*. По этой магистрали данные передаются между регистрами и АЛУ, поскольку АЛУ может оперировать только с данными, хранящимися в регистрах. На этой же схеме показана и типовая структура АЛУ.

Таким образом, базовыми функциональными элементами процессора являются:

• АЛУ;

• регистры;

• внутренние магистрали передачи данных;

• внешние магистрали передачи данных;

• устройство управления.

АЛУ (арифметико-логическое устройство) представляет собой компонент процессора, в котором выполняется преобразование информации — арифметические и логические операции над данными (операндами). Все остальные узлы компьютера фактически существуют для того, чтобы обеспечить работой АЛУ.

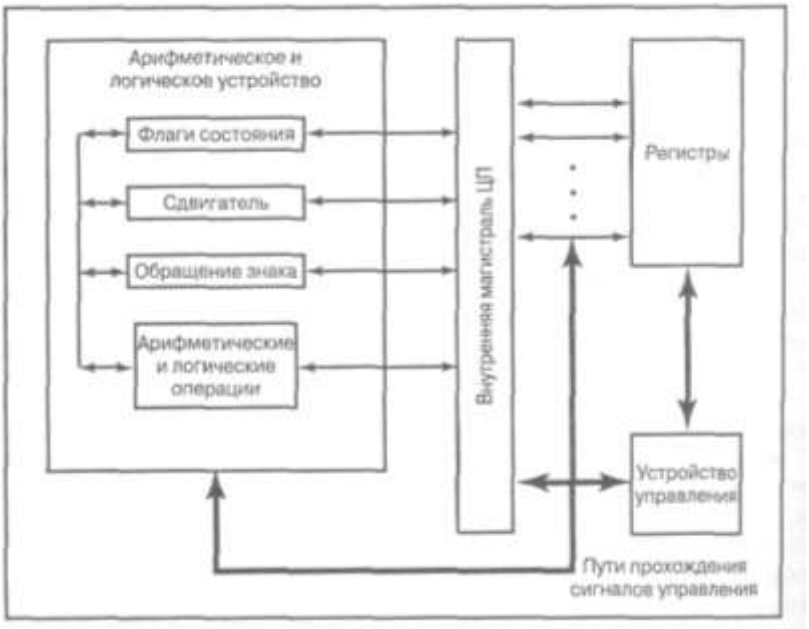


Рис. 5.2. *Внутренняя структура процессора*

АЛУ представляет собой совокупность простейших электронных логических схем, способных хранить двоичные числа и выполнять над ними элементарные операции булевой алгебры.

Функционально АЛУ (рис. 5.3) состоит обычно из двух регистров, сумматора и схем управления (местного устройства управления).

На рис. 5.4 схематично показано, как АЛУ взаимодействует с остальными компонентами процессора.

Данные, над которыми выполняются операции, поступают из регистров процессора, а результат также направляется в регистры. Эти регистры представляют собой элементы внутренней памяти процессора и связаны с АЛУ линиями передачи сигналов. В качестве вторичного результата АЛУ формирует значения набора двоичных флагов (признаков).

Значения флагов также фиксируются в регистрах процессора. Блок управления передает в АЛУ сигналы, управляющие процессом выполнения операции, передачей операндов в АЛУ и считыванием результатов.

При выполнении операций в Pr1 помещается первое число, участвующее в операции, а по завершении операции — результат; в Pr2 — второе число, участвующее в операции (по завершении операции информация в нем не изменяется). Регистр 1 может и принимать информацию с кодовых шин данных, и выдавать информацию на них, регистр 2 только получает информацию с этих шин.

Схемы управления принимают по кодовым шинам инструкций управляющие сигналы от устройства управления и преобразуют их в сигналы для управления работой регистров и сумматора АЛУ.

АЛУ выполняет арифметические операции (+, -, *, :) только над двоичной информацией с запятой, фиксированной после последнего разряда, т.е. только над целыми двоичными числами. Выполнение операций над двоичными числами с плавающей запятой и над двоично-кодированными десятичными числами осуществляется или с привлечением математического сопроцессора, или по специально составленным программам.

Внутренние магистрали данных обеспечивают пересылку информации между регистрами или между регистрами и АЛУ. Внешние магистрали связывают регистры с подсистемой памяти и модулями ввода-вывода. Устройство управления координирует работу всех узлов процессора. В процесс выполнения команд программы вовлекаются все перечисленные функциональные узлы процессора.

Рис. 5.3. *Функциональная схема АЛУ*

Рис. 5.4. *Входная и выходная информация АЛУ*

5.3. Сопроцессоры

С любыми процессорами фирмы Intel (и их аналогами) могут использоваться сопроцессоры. В процессорах Pentium и 486 они расположены на том же кристалле, что и сам процессор. *Сопроцессоры* выполняют операции с плавающей запятой, которые потребовали бы от основного процессора больших затрат машинного времени. Выигрыш можно получить только при выполнении программ, написанных с расчетом на использование сопроцессора.

Центральный процессор взаимодействует сопроцессором по следующему сценарию: когда надо выполнить вещественную операцию, центральный процессор посылает сигнал сопроцессору и передает ему соответствующие операнды (в ПК есть специальная команда для этого); сопроцессор выполняет указанную операцию, записывает результат в определенное место и возвращает управление центральному процессору, который после этого продолжает свою работу.

Сопроцессоры выполняют такие сложные операции, как деление длинных операндов, вычисление тригонометрических функций, извлечение квадратного корня и нахождение логарифма, в 10-100 раз быстрее основного процессора. Точность результатов при этом значительно выше обеспечиваемой вычислителями, входящими в состав самих процессоров. Операции сложения, вычитания и умножения выполняются основным процессором и не передаются сопроцессору.

Система команд сопроцессора отличается от системы команд процессора. Выполняемая программа должна сама определять наличие сопроцессора и после этого использовать написанные для него инструкции; в противном случае сопроцессор только потребляет ток и ничего не делает. Большинство современных программ, рассчитанных на использование сопроцессоров, обнаруживают его присутствие и используют предоставляемые возможности. Наиболее эффективно сопроцессоры используются в программах со сложными математическими расчетами: в электронных таблицах, базах данных, статистических программах и системах автоматизированного проектирования. В то же время при работе с текстовыми редакторами сопроцессор совершенно не используется.

5.4. Режимы процессора

Все 32-разрядные и более поздние процессоры Intel, начиная с 386-го, могут выполнять программы в нескольких режимах. Режимы процессора предназначены для выполнения программ в различных средах; в разных режимах возможности чипа неодинаковы, потому что команды выполняются по-разному. В зависимости от режима процессора изменяется схема управления памятью системы и задачами.

Процессоры могут работать в трех режимах: реальном, защищенном и виртуальном реальном режиме (реальном внутри защищенного).

Реальный режим

В первоначальном IBM PC использовался процессор 8088, который мог выполнять 16-разрядные команды, применяя 16-разрядные внутренние регистры, а адресовать только 1 Мбайт памяти, используя 20 разрядов для адреса. Все программное обеспечение PC перво-

начально было предназначено для этого процессора; оно было разработано на основе 16-разрядной системы команд и модели памяти объемом 1 Мбайт.

Шестнадцатиразрядный режим, в котором выполнялись команды процессоров 8088 и 286, был назван *реальным режимом.* Все программы, выполняющиеся в реальном режиме, должны использовать только 16-разрядные команды, 20-разрядные адреса и поддерживаться архитектурой памяти, рассчитанной на емкость до 1 Мбайта. Для программного обеспечения этого типа обычно используется однозадачный режим, т.е. одновременно может выполняться только одна программа. Нет никакой встроенной защиты для предотвращения перезаписи ячеек памяти одной программы или даже операционной системы другой программой; это означает, что при выполнении нескольких программ вполне могут быть испорчены данные или код одной из них, а это может привести всю систему к краху (или останову).

Защищенный режим

Первым 32-разрядным процессором, предназначенным для PC, был 386-й. Этот чип мог выполнять абсолютно новую 32-разрядную систему команд. Чтобы полностью использовать преимущество 32-разрядной системы команд, были необходимы 32-разрядная операционная система и 32-разрядные приложения. Этот новый режим назывался *защищенным,* так как выполняющиеся в нем программы защищены от перезаписи своих областей памяти другими программами. Такая защита делает систему более надежной, поскольку ни одна программа с ошибками уже не сможет так легко повредить другие программы или операционную систему. Кроме того, программу, "потерпевшую крах", можно довольно просто завершить без ущерба для всей системы.

Поэтому были необходимы новые операционные системы и приложения, которые бы выполнились на современных процессорах в 32-разрядном защищенном режиме. Однако некоторые пользователи поначалу сопротивлялись всяческим попыткам перехода к 32-разрядной среде.

Из-за этого сопротивления Microsoft разработали Windows 95 как промежуточную систему для перехода в 32-разрядный мир. Windows 95 — в основном 32-разрядная операционная система, но в ней можно выполнять старые 16-разрядные приложения. Windows 95 появилась в августе 1995 года, т.е. через 10 лет после появления первого 32-разрядного процессора для PC. Потребовалось "всего лишь" 10 лет, чтобы внедрить программное обеспечение, которое может полностью использовать процессоры.

Виртуальный реальный режим

Виртуальный реальный режим является режимом выполнения 16-разрядной среды (реальный режим), которое реализовано внутри 32-разрядного защищенного режима (т.е. виртуально, а не реально). Выполняя команды в окне подсказки DOS внутри Windows 95/98, вы создаете виртуальный сеанс реального режима. Поскольку защищенный режим является подлинно многозадачным, фактически можно выполнять несколько сеансов реального режима, причем в каждом сеансе собственное программное обеспечение выполняется на виртуальном компьютере. И все эти приложения могут выполняться одновременно, даже во время выполнения других 32-разрядных программ.

Любая программа, выполняющаяся в виртуальном окне реального режима, может обращаться только к памяти объемом до 1 Мбайт, причем для каждой такой программы это будет первый и единственный мегабайт памяти в системе. Другими словами, если вы выполняете приложение DOS в виртуальном реальном окне, ему будет доступ-

на память только объемом до 640 Кбайт. Так происходит потому, что имеется только 1 Мбайт общей оперативной памяти в 16-разрядной среде, а верхние 384 Кбайт зарезервированы для системы. Виртуальное реальное окно полностью имитирует среду процессора 8088, и, если не учитывать быстродействие, программное обеспечение будет выполняться так, как оно выполнялось первым PC в реальном режиме.

SMM

Система управления питанием процессоров называется *SMM* (System Management Mode — режим управления системой).

SMM физически интегрирована в процессор, но функционирует независимо. Благодаря этому она может управлять потреблением мощности, в зависимости от уровня активности процессора. Это позволяет пользователю определять интервалы времени, по истечении которых процессор будет частично или полностью выключен. Данная схема также поддерживает возможность приостановки/возобновления, которая позволяет мгновенно включать и отключать мощность, что обычно используется в портативных компьютерах. Соответствующие параметры устанавливаются в BIOS.

Суперскалярное выполнение

В процессорах Pentium пятого и последующих поколений встроен ряд внутренних конвейеров, которые могут выполнять несколько команд одновременно. Процессор 486 и все предшествующие в течение определенного отрезка времени могли выполнять только одну команду. Технология одновременного выполнения нескольких команд называется *суперскалярной.*

Суперскалярная архитектура обычно ассоциируется с микросхемами RISC (Reduced Instruction Set Computer — компьютер с упрощенной системой команд). Процессор Pentium — одна из первых микросхем CISC (Complex Instruction Set Computer — компьютер со

сложной системой команд), в которой применяется суперскалярная технология, реализованная во всех процессорах пятого и последующих поколений.

5.5. Параметры процессоров

Процессоры можно классифицировать по двум основным параметрам: разрядности и быстродействию. *Быстродействие* процессора измеряется в мегагерцах (МГц); 1 МГц равен миллиону тактов в секунду. Чем выше быстродействие, тем лучше (тем быстрее процессор). *Разрядность* процессора — параметр более сложный. В процессор входит три важных устройства, основной характеристикой которых является разрядность:

• шина ввода и вывода данных;

• внутренние регистры;

• шина адреса памяти.

Процессоры с тактовой частотой менее 16 МГц не имеют встроенной кэш-памяти. В системах до 486-го процессора быстрая кэш-память устанавливалась на системную плату. Начиная с процессоров 486, кэш-память первого уровня устанавливалась непосредственно в корпусе и работала на частоте процессора. А кэш-память на системной плате стали называть кэш-памятью второго уровня. Она работала уже на частотах, поддерживаемых системной платой.

В процессорах Pentium Pro и Pentium II кэш-память второго уровня устанавливается в корпусе процессора и физически представляет отдельную микросхему. Чаше всего такая память работает на половинной (процессоры Pentium II/III и AMD Athlon) или даже меньшей (две пятых или треть) частоте ядра процессора. В табл. 5.1 приведены данные о рабочих частотах кэш-памяти второго уровня процессоров Pentium II/III/Celeron и Athlon (модели 1 и 2).

Таблица 5.1. Параметры кэш-памяти второго уровня Pentium II/III/Celeron и Athlon

Процессор частота	Частота, МГц	Обьем кэш-памяти, Кбайт	Тип кэш-памяти	Рабочая частота
Pentium III	450-600	512	Внешний	Половина частоты ядра (225-300 МГц)
Athlon	550-700	512	Внешний	Половина частоты ядра (275-300 МГц)
Athlon	750-850	512	Внешний	Две пятых частоты ядра (300-340 МГц)
Athlon	900-1000	512	Внешний	Треть частоты ядра (300-333 МГц)

А в процессорах Pentium Pro, Pentium II/III Xeon, современных моделях Pentium III, Celeron, K6-3, Athlon (модель 4), Duron кэш-память работает на частоте ядра (табл. 5.2).

Таблица 5.2. Параметры кэш-памяти второго уровня современных процессоров

Процессор	Частота, МГц	Объем кэш-памяти	Тип-кэш памяти	Рабочая частота
Pentium Pro	150-200	256 Кбайт-1Мбайт	Внешний	Частота ядра
K6-3	350-450	256 Кбайт	На пластине	Частота ядра
Duron	550-700+	64 Кбайт	На пластине	Частота ядра
Celeron	300-600+	128Кбайт	На пластине	Частота ядра

Процессор	Частота, МГц	Объем кэш-памяти	Тип-кэш памяти	Рабочая частота
Pentium II Xeon	400-450	512Кбайт-2 Мбайт	Внешний	Частота ядра
Athlon	650-1000+	256 Кбайт	На пластине	Частота ядра
Pentium III	500-1000+	256 байт	На пластине	Частота ядра
Pentium III Xeon	500-1000+	256 Кбайт-2Мбайт	На пластине	Частота ядра

Быстродействие компьютера во многом зависит от тактовой частоты, обычно измеряемой в мегагерцах (МГц, Герц — одно колебание в секунду). Она определяется параметрами кварцевого резонатора, представляющего собой кристалл кварца, заключенный в небольшой оловянный контейнер. Под воздействием электрического напряжения в кристалле кварца возникают колебания электрического тока с частотой, определяемой формой и размером кристалла. Частота этого переменного тока и называется *тактовой частотой.* Единица измерения частоты названа герцем в честь немецкого физика Генриха Герца. В 1885 году Герц экспериментальным путем подтвердил правильность электромагнитной теории, согласно которой свет является видом электромагнитного излучения и распространяется в виде волн.

Наименьшей единицей измерения времени (квантом) для процессора как логического устройства является *период тактовой частоты,* или просто *такт.* На каждую операцию затрачивается минимум один такт. Например, обмен данными с памятью процессор Pentium II выполняет за три такта плюс несколько циклов ожидания. (*Цикл ожидания* — это такт, в котором ничего не происходит; он необходим только для того, чтобы процессор не "убегал" вперед от менее быстродействующих узлов компьютера).

Почти все современные компьютеры, начиная с 486DX2, работают на тактовой частоте, которая равна произведению некоторого множителя на тактовую частоту системной платы.

5.6. Цикл обработки команды

Процесс обработки отдельной команды называется циклом обработки команды. Он включает три основных фазы:

1) фазу извлечения: считывание очередной команды из памяти в процессор;

2) фазу выполнения: расшифровка кода операции и выполнение соответствующих действий;

3) фазу прерывания: если прерывания разрешены и имеется запрос прерывания, сохранение текущего состояния процесса и обслуживание прерывания.

В свою очередь, фаза выполнения может включать в себя несколько операций, назначение которых индивидуально для каждой команды. Завершив выполнение одной команды, компьютер начинает выполнять следующую, и так продолжается до тех пор, пока кто-либо не выключит питание, не будет обнаружена неисправимая (фатальная) ошибка или в программе не встретится команда останова.

Извлечение и выполнение команды

В начале каждого цикла из памяти извлекается очередная команда, адрес которой заранее готовится в *счетчике команд* PC (*program counter*). При обращении к PC адрес в нем автоматически увеличивается и, таким образом, сразу подготавливается к следующему циклу, поскольку команды в памяти хранятся в естественном порядке, т.е. в последовательных ячейках.

Считанная команда помещается в *регистр команд* IR (*instruction register*). Часть битов кода команды представляет собой *код опе-*

рации, определяющий, какая именно операция должна быть выполнена процессором. Выполняя заданную операцию, процессор будет производить элементарные действия, которые можно разделить на четыре категории:

- *передача "процессор — память":* данные передаются из процессора в память или из памяти в процессор;
- *передача "процессор — модуль ввода-вывода":* данные передаются из процессора в один из модулей ввода-вывода или наоборот;
- *обработка данных:* процессор выполняет заданную арифметическую или логическую операцию над данными;
- *управление:* команда изменяет естественный порядок выполнения команд программы.

На рис. 5.5 представлена подробная диаграмма состояний при выполнении команды. Смысл представленных на диаграмме состояний следующий.

- *Вычисления адреса команды* (IAC — Instruction Address Calculation). При естественном порядке выполнения команд программы адрес следующей команды образуется в результате прибавления константы к адресу текущей команды. Если, например, длина команды 16 разрядов, и такую же разрядность имеют слова в памяти, то адрес каждой очередной команды отличается от адреса предыдущей на 1. Если же оперативная память организована таким образом, что адресуемой единицей данных является 8-разрядный байт, то следует адрес в PC увеличивать на 2.
- *Извлечение команды* (IF — Instruction Fetch). Считывание команды из ячейки памяти по адресу, заданному в PC, в регистр процессора.

Рис. 5.5. *Диаграмма состояний цикла обработки команды*

• *Расшифровка кода операции* (IOD —Instruction Operation Decoding). Анализ кода операции и выяснение типа затребованной операции, количества и типов участвующих в ней операндов (или операнда).

• *Вычисления адреса операнда* (OAC — Operand Address Calculation). Если в операции используются операнды, хранящиеся в памяти или передаваемые через подсистему ввода-вывода, то на этой фазе определяются их "физические" адреса.

• *Извлечение операнда* (OF — Operand Fetch). Считывание операнда из заданной ячейки памяти или из модуля ввода-вывода.

• *Обработка данных* (DO — Data Operation). Выполнение операции, предусмотренной кодом операции.

• *Сохранение результата* (OS — Operand Store). Запись результата операции в заданную ячейку памяти или передача в модуль ввода-вывода.

В состояние OAC система может попадать дважды, поскольку в процессе выполнения команды может потребоваться чтение операнда (или операндов) и/или запись результата, но суть выполняемых в этом состоянии действий не зависит от того, как дальше будет ис-

пользоваться вычисленный адрес — для записи или для чтения.

В процессе выполнения команды процессору могут понадобиться операнды (один или два), находящиеся в памяти. Извлечение каждого операнда требует обращения к памяти. Если в спецификации операнда указан режим косвенной адресации, то помимо обращения к памяти за значением операнда, потребуется еще одно обращение к памяти за адресом операнда.

5.7. Обработка прерываний

Практически во всех современных компьютерах имеются средства, позволяющие определенным компонентам (памяти или модулям ввода-вывода) прерывать нормальную работу процессора. Наиболее распространенными классами прерываний являются следующие.

- *Программные прерывания.* Генерируются в некоторых исключительных ситуациях, которые возникают при выполнении команд — при арифметическом переполнении, делении на нуль, попытке выполнения команды с несуществующим кодом операции, обращении по адресу, выходящему за пределы физического адресного пространства компьютера.
- *Прерывания таймера.* Генерируются таймером, который организационно входит в состав процессора. Эти прерывания используются операционной системой для выполнения некоторых регулярных операций.
- *Прерывания ввода-вывода.* Генерируются модулями ввода-вывода, чтобы сигнализировать ЦП о том, что текущая операция ввода-вывода завершена или возникла какая-либо ошибка в процессе ее выполнения.

• *Аварийные прерывания.* Генерируются специальными средствами контроля работоспособности аппаратуры при обнаружении сбоев, в частности пропадания питания или ошибки контроля памяти по четности.

Если обнаружен сигнал запроса прерывания (рис. 5.6), процессор выполняет такие операции.

1. Запоминает информацию о текущем состоянии выполняемой программы — адрес следующей команды (содержимое PC) и все данные, которые касаются текущего состояния узлов процессора и могут оказать влияние на выполнение следующей команды.

2. В счетчик команд записывается начальный адрес *обработчика прерывания* (подпрограммы обработки прерывания).

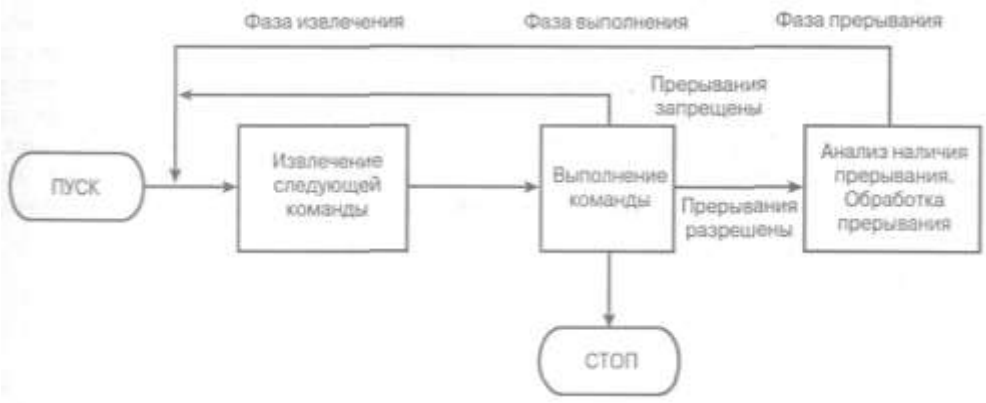


Рис. 5.6. *Цикл обработки команды, включающий фазу прерывания.*

На этом текущий цикл обработки команды завершается, и начинается фаза извлечения следующего цикла — из памяти считывается первая команда обработчика прерывания. Как правило, обработчик прерывания входит в состав операционной системы. Эта подпрограмма определяет причину возникшего прерывания и выполняет соответствующие этому событию операции. Например, обработчик определяет, какой канал ввода-вывода сгенерировал сигнал запроса прерывания, и переходит к подпрограмме, которая передает дальнейшую порцию данных соответствующему внешнему устройству.

Когда выполнение подпрограммы пересылки очередной порции завершится, процессор может восстановить состояние прерванной программы и продолжить ее выполнение.

Рассмотрим более детально роль процессора в обработке прерывания ввода-вывода. Появление сигнала запроса прерывания запускает несколько процессов, часть которых реализуется аппаратными средствами, а часть — программными. На рис. 5.7 представлен типичный алгоритм обработки прерывания. После завершения модулем ввода-вывода заданной операции на аппаратном уровне выполняются следующие действия.

1. Модуль ввода-вывода формирует сигнал запроса прерывания, который предназначается процессору.
2. Процессор завершает выполнение текущей команды программы прежде, чем анализировать наличие сигнала запроса прерывания.
3. Процессор анализирует, не поступил ли сигнал запроса прерывания, обнаруживает его и посылает сигнал подтверждения тому модулю, который сформировал сигнал запроса. Получив сигнал подтверждения, модуль, инициировавший прерывание, снимает соответствующий сигнал с линии управления магистрали.
4. Процессор выполняет действия, предшествующие передаче управления подпрограмме обработки прерывания. Прежде всего нужно позаботиться о сохранении текущего состояния процессора, что позволит в дальнейшем возобновить выполнение прерванной программы с той же точки. В этот момент в счетчике команд процессора находится адрес команды, которая должна была бы выполняться процессором, если бы не произошло прерывание. Эту команду (а точнее, ее адрес) принято называть *точкой прерывания,* и она же является *точкой возврата из прерывания.* Поэтому процессор должен каким-то образом сохранить состояние счетчика команд, регистра состояния, который

содержит *слово состояния процессора* PSW (*processor status word*). Как правило, эта информация помещается в системный стек.

5. Затем процессор загружает в счетчик команд начальный адрес программы (подпрограммы) обработки прерывания. В разных компьютерах и разных операционных системах эта операция выполняется поразному. В одних — каждый источник прерывания (модуль ввода-вывода, вызвавший прерывание) имеет свою подпрограмму обработки, в других — прерывания разбиты на группы, и каждая группа имеет свою программу обработки, а в третьих — вообще существует одна программа обработки всех прерываний, внутри которой происходит ветвление в соответствии с конкретным источником прерывания.

Если архитектура системы предусматривает наличие множества подпрограмм обработки прерываний, то информацию о том, какую именно подпрограмму нужно вызывать в каждом конкретном случае, процессор должен получать вместе с самим сигналом запроса прерывания. Другой вариант — процессор получает ее от устройства, инициировавшего прерывание, в ответ на сигнал подтверждения.

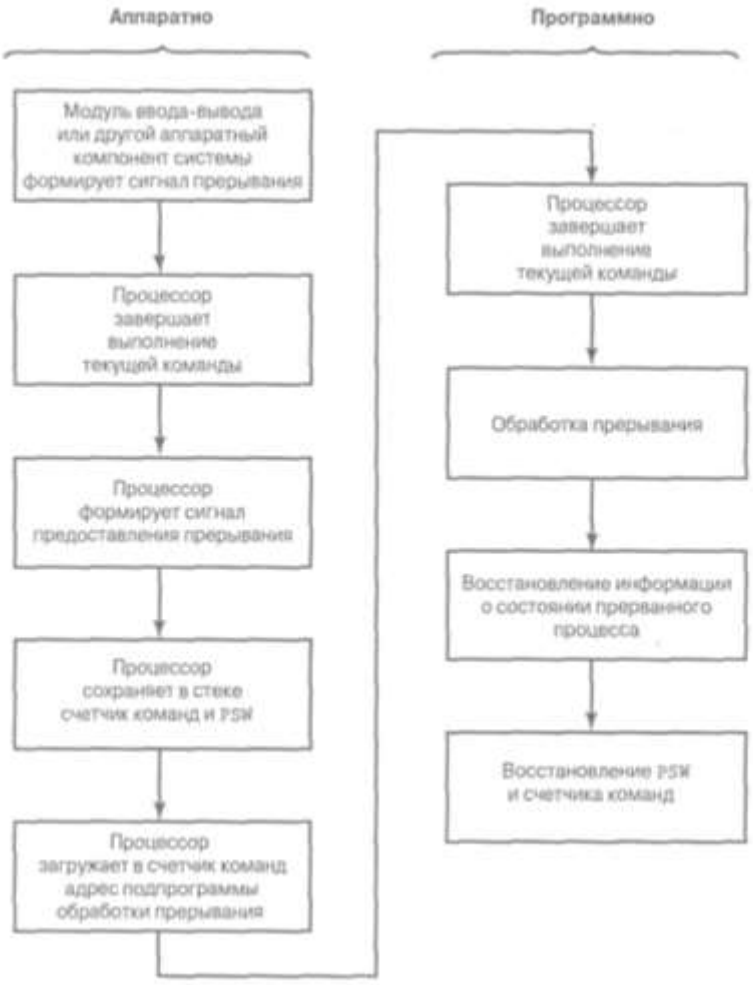


Рис. 5.7. *Алгоритм обработки прерывания*

Как только в счетчик команд будет помещен адрес подпрограммы, процессор начнет новый цикл выполнения команды. Первой фазой цикла будет извлечение команды, адрес которой находится в счетчике. В результате процессор приступит к выполнению заданной подпрограммы обработки прерывания. Что именно будет при этом сделано, определяет программист, который разработал подпрограмму.

6. Перед передачей управления подпрограмме обработки прерывания процессор аппаратными средствами выполнил сохранение счетчика команд и PSW в системном стеке. Но помимо этой информации для возобновления прерванной программы может потребоваться и восстановление состояния других регистров процессора. Поэтому, как правило, первые команды подпрограммы обработки сохраняют в стеке

состояния всех остальных регистров процессора, поскольку они могут понадобиться самой подпрограмме и, следовательно, их содержимое к моменту завершения подпрограммы будет изменено. На рис. 5.8, а) показано, что при этом происходит. В данном случае пользовательская программа была прервана в тот момент, когда выполняла команду, хранящуюся по адресу N. Аппаратно в стек помещается содержимое счетчика команд, каким оно было в момент завершения выполнения этой команды — N+1, а в счетчик команд заносится начальный адрес подпрограммы обработки Y. Сама подпрограмма затем помещает в стек содержимое регистров процессора, в результате чего указатель стека смещается и после завершения сохранения регистров будет указывать на ячейку T.

7. После этого подпрограмма приступает к собственно обработке прерывания. Этот процесс, как правило, включает анализ состояния модуля ввода-вывода, вызвавшего прерывание, и прием или передачу порции данных. Иногда в процессе обработки модулю посылаются другие команды или сообщения, в частности команды запуска следующего сеанса обмена, если передается большой блок данных.

8. После завершения обработки подпрограмма восстанавливает содержимое в регистрах, которое было сохранено в стеке в самом начале выполнения подпрограммы (рис. 5.8, б)), и только после этого выполняет последнюю команду — команду возврата из прерывания.

9. Выполнение команды возврата из прерывания сводится к восстановлению из стека слова состояния программы PSW и счетчика команд. В результате состояние регистров процессора полностью восстанавливается в том виде, каким оно было в момент обнаружения прерывания.

Важно перед началом обработки прерывания сохранить всю информацию, необходимую для продолжения выполнения прерван-

ной программы. Дело в том, что прерывание может возникнуть в любой момент, и подпрограмме обработки прерывания абсолютно неизвестно, какую именно программу она прервала, какие регистры процессора эта программа использовала, а какие нет. Поэтому надежное функционирование вычислительной системы, имеющей дело с прерываниями, может быть гарантировано только в том случае, если аппаратными и программными средствами предусмотрено полное сохранение состояния системы в момент прерывания и последующее его восстановление.

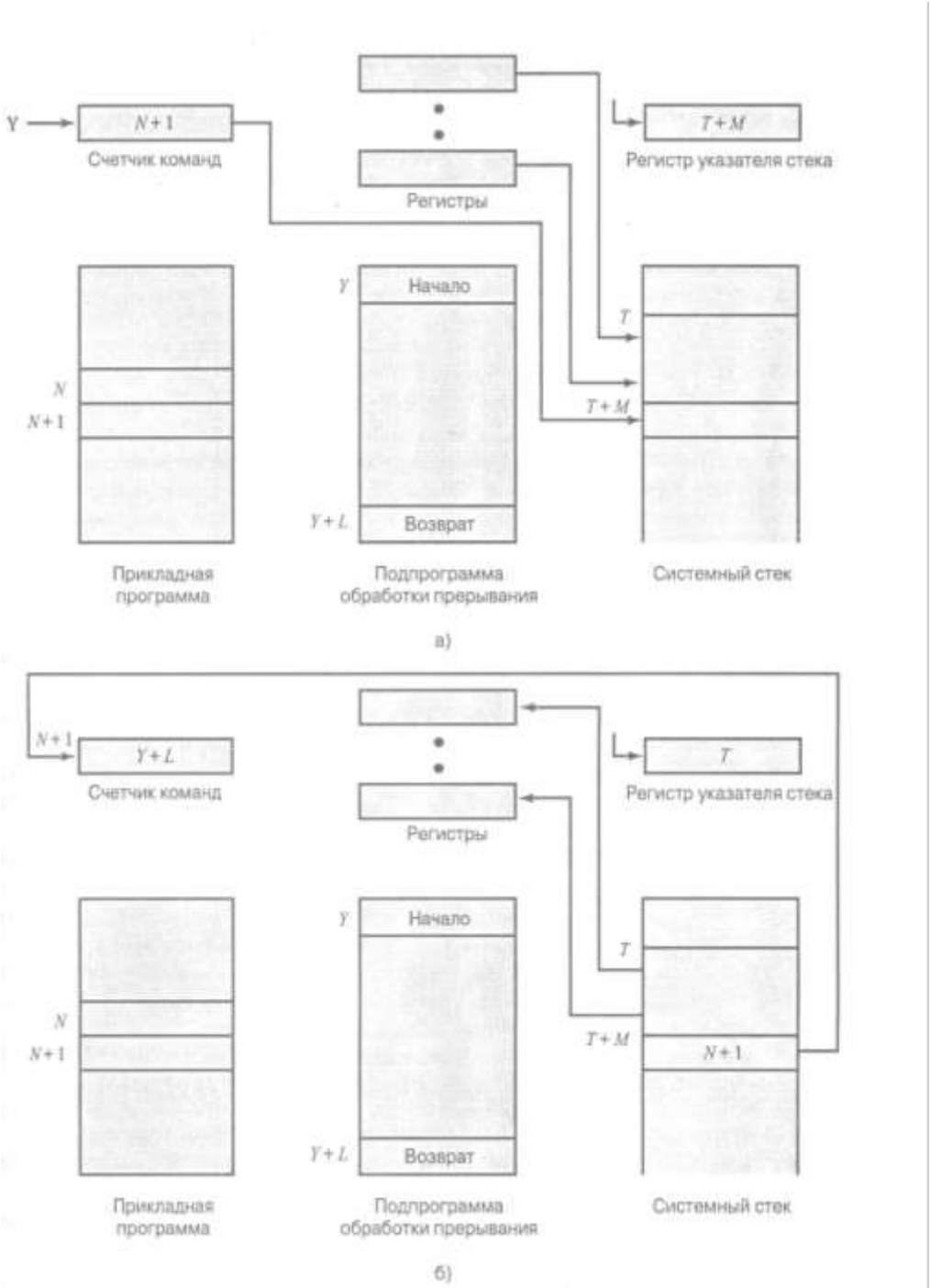


Рис. 5.8. *Изменение состояния памяти и регистров процессора в процессе обработки прерывания: а) вход в подпрограмму обработки прерывания; б) завершение подпрограммы обработки прерывания*

5.8. Организация набора регистров

Количество битов данных, которые может обработать процессор за один прием, характеризуется *разрядностью* внутренних реги-

стров. *Регистр* — это, по существу, ячейка памяти внутри процессора; например, процессор может складывать числа, записанные в двух различных регистрах, а результат сохранять в третьем регистре. Разрядность регистра определяет количество разрядов обрабатываемых процессором данных. Разрядность регистра также определяет характеристики программного обеспечения и команд, выполняемых чипом. Например, процессоры с 32-разрядными внутренними регистрами могут выполнять 32- -разрядные команды, которые обрабатывают данные 32-разрядными порциями, а процессоры с 16-разрядными регистрами этого делать не могут. Во всех современных процессорах внутренние регистры являются 32-разрядными.

В некоторых процессорах разрядность внутренней шины данных (а шина состоит из линий передачи данных и регистров) больше, чем разрядность внешней. Так, например, в процессорах 8088 и 386SX разрядность внутренней шины только вдвое больше разрядности внешней шины. Такие процессоры (их часто называют *половинчатыми* или *гибридными)* обычно являются более дешевыми вариантами исходных. Например, в процессоре 386SX внутренние операции 32-разрядные, а связь с внешним миром осуществляется через 16-разрядную внешнюю шину. Это позволяет разработчикам проектировать относительно дешевые системные платы с 16-разрядной шиной данных, сохраняя при этом совместимость с 32-разрядным процессором 386.

Если разрядность внутренних регистров больше разрядности внешней шины данных, то для их полной загрузки необходимо несколько циклов считывания. Например, в процессорах 386DX и 386SX внутренние регистры 32-разрядные, но процессору 386SX для их загрузки необходимо выполнить два цикла считывания, а процессору

386DX достаточно одного. Аналогично передаются данные от регистров к системной шине.

В процессорах Pentium шина данных 64-разрядная, а регистры 32-разрядные. Такое построение на первый взгляд кажется странным, если не учитывать, что в этом процессоре для обработки информации служат два 32-разрядных параллельных конвейера. Pentium во многом подобен двум 32-разрядным процессорам, объединенным в одном корпусе, а 64-разрядная шина данных позволяет быстрее заполнить рабочие регистры. Архитектура процессора с несколькими конвейерами называется *суперскалярной.*

Современные процессоры шестого поколения, например Pentium Pro и Pentium II/III, имеют целых шесть внутренних конвейеров для выполняющихся команд. Хотя некоторые из указанных внутренних конвейеров специализированы (т.е. предназначены для выполнения специальных функций), эти процессоры могут все же выполнять три команды за один цикл.

Набор регистров процессора представляет память самого верхнего уровня, находящегося в иерархии над уровнями оперативной памяти и кэш-памяти. Регистры процессора разделяются по функциям.

- *Программно доступные регистры* позволяют программисту, разрабатывающему программы на машинном языке или на языке ассемблера, минимизировать в программе обращение к оперативной памяти и, где это возможно, использовать для хранения промежуточных результатов быстродействующую внутреннюю память процессора.
- *Регистры управления и состояния* используются для управления функционированием процессора. К этим регистрам иногда могут иметь доступ специальные привилегированные команды,

которые используются только в программах операционной системы.

Не существует раз и навсегда узаконенного разделения регистров на эти категории. Например, в большинстве компьютеров счетчик команд, PC, программно недоступен, но существуют и такие машины, как VAX, в которых он доступен со стороны программы, как и любой другой.

5.8.1. Программно доступные регистры

Программно доступные регистры можно разделить на четыре категории:

• регистры общего назначения (или универсальные регистры);

• регистры данных;

• регистры адреса;

• регистры кодов условий.

Регистры общего назначения программист может использовать по своему усмотрению для выполнения самых разных функций. Иногда система команд никак не связывает их функции с выполняемой операцией (это бывает в системах команд, в которых поля кода операций и операндов *ортогональны).* В таком случае мы действительно имеем дело с *универсальными* регистрами, т.е. любой регистр может использоваться для спецификации любого операнда в любой машинной команде. Но чаще всего возможность применения того или иного регистра все-таки в определенной мере ограничивается. Например, для работы с системным стеком и числами в формате с плавающей точкой используются не любые, а только определенные регистры.

В некоторых конструкциях процессоров регистры общего назначения можно применять для адресации операндов (т.е. косвенной адресации через регистр, адресации со смещением и т.п.). В дру-

гих конструкциях существует жесткое разделение между регистрами адреса и регистрами данных. *Регистры данных* разрешается использовать только для хранения операндов и результатов, но они не принимают участия в вычислении исполнительного адреса. *Регистры адреса* могут быть в определенной степени универсальными, а могут и предназначаться только для определенного режима адресации. Ниже приведены некоторые регистры адреса.

- *Указатель сегмента.* В компьютерах с сегментной организацией оперативной памяти выделяется специальный регистр сегмента, в котором хранится базовый адрес сегмента. Таких регистров может быть несколько, один хранит базовый адрес сегмента операционной системы, а другой (другие) — базовый адрес сегмента текущего процесса (или нескольких параллельно выполняемых процессов).
- *Индексные регистры.* Эти регистры используются в режимах адресации с индексацией, и в некоторых процессорах при обращении к ним автоматически выполняется приращение или уменьшение значения на 1 (происходит *автоиндексация*).
- *Указатель стека.* Если в компьютере механизм управления системным стеком программно доступен, то стек размещается в оперативной памяти и выделяется специальный регистр, который содержит текущее значение указателя вершины стека. Наличие системного стека позволяет неявно специфицировать операнды в нульадресных командах.

Перед конструктором процессора встает несколько вопросов, связанных с организацией регистров. Первый из них — следует ли отдать предпочтение универсальным программно доступным регистрам или специализировать их на выполнение определенного круга функций. Если в процессоре существуют специализированные регистры, то

их применение специфицируется неявно через код операции, который требует участия в определенной операции именно такого-то регистра и никакого другого. Явная спецификация операнда при такой организации может только уточнить, какой именно из группы регистров одинакового назначения будет использоваться при выполнении данной команды (если специализированные регистры объединены в группы по назначению). В результате сохраняется пространство в коде команды для другой информации. С другой стороны, любая специализация ограничивает программиста в использовании ресурсов процессора. Поэтому не существует однозначного решения этой проблемы.

Другой вопрос, встающий перед конструктором процессора, — *сколько* регистров включить в его состав. Опять же с ответом на него связан и формат команд — чем больше регистров, тем больший размер поля спецификации регистров в команде. Принято считать, что оптимальное количество программно доступных регистров находится в диапазоне от 8 до 32. Чем меньше регистров в процессоре, тем чаще придется обращаться к оперативной памяти, а значит медленнее будет выполняться программа. Тем не менее увеличение количества регистров свыше определенного порога не приводит к дальнейшему сокращению частоты обращения к памяти, а значит затраты на их включение в процессор не окупаются.

Последний вопрос проектирования набора регистров — выбор разрядности. Регистры, предназначенные для хранения адресов, должны иметь разрядность, соответствующую длине адреса при включении в состав компьютера оперативной памяти максимального объема. Регистры данных должны иметь разрядность, позволяющую хранить элементы большинства типов данных, используемых в проектируемой модели компьютера. В некоторых конструкциях процессо-

ров пары регистров организованы таким образом, что могут совместно хранить единый элемент данных удвоенной длины.

Регистры условий также в определенной мере программно доступны, хотя набор операций с ними со стороны программы очень ограничен. Эти регистры (или регистр) хранят *однобитовые коды условий,* которые принято называть флагами. Большинство флагов устанавливается процессором в зависимости от значения результата последней выполненной арифметической или логической операции. Например, арифметическая операция может породить положительный или отрицательный результат, равный нулю или отличный от нуля, может произойти переполнение и т.д. Все эти свойства результата фиксируются в соответствующих флагах дополнительно к собственно результату, который фиксируется в каком-либо регистре данных или в ячейке памяти. Эти флаги могут быть затем проанализированы при выполнении команд условного перехода.

Биты кодов условий объединяются в один или несколько регистров условий. Обычно они включаются в регистр управления. В общем случае эти флаги могут анализироваться машинными командами, но машинная команда не может их явно изменить. (Это возможно только при выполнении арифметической команды со специально подобранными значениями операндов.)

В некоторых процессорах при выполнении команд вызова подпрограмм автоматически сохраняется вся информация, размещенная в программно доступных регистрах. При выходе из подпрограммы эта информация восстанавливается. Такой механизм вызова и возврата из подпрограммы позволяет каждой подпрограмме независимо пользоваться всеми регистрами. В других процессорах такого автоматического сохранения/восстановления не происходит, и заботы о сохранении информации в регистрах перекладываются на программиста.

5.8.2. Регистры управления и состояния

В управлении работой процессора принимает участие несколько регистров. В большинстве моделей процессоров эти регистры недоступны программе, но могут неявно опрашиваться при выполнении некоторых привилегированных машинных команд в супервизорном режиме (такие команды используются в программах операционной системы).

Нужно отметить, что разные модели процессоров имеют отличную друг от друга организацию регистров и, следовательно, отличную терминологию.

В процессе обработки машинной команды существенную роль играют четыре регистра управления.

- *Счетчик команд* (PC — *program counter*) содержит адрес извлекаемой машинной команды.
- *Регистр текущей команды* (IR — *instruction register*) содержит последнюю извлеченную машинную команду.
- *Регистр адреса в памяти* (MAR — *memory address register*) содержит адрес ячейки памяти.
- *Регистр данных памяти* (MBR — *memory buffer register*) содержит слово данных, которое должно быть записано в память или слово, только что извлеченное из памяти.

Счетчик команд PC обновляется процессором сразу после того, как извлекается очередная машинная команда, и, таким образом, практически в течение всего цикла обработки команды в нем содержится адрес команды, которая должна выполняться в следующем цикле. Содержимое PC также изменяется командами переходов и пропуска. Извлеченная из памяти машинная команда помещается а регистр IR, а затем производится анализ отдельных ее полей — кода операции и полей операндов. Данные извлекаются из памяти с помо-

щью регистров MAR и MBR. В компьютерах с системной магистралью регистр MAR соединен напрямую с линиями адреса, а регистр MBR — с линиями данных. В свою очередь, программно доступные регистры обмениваются данными с MBR.

Четыре перечисленных регистра используются при перемещении данных между процессором и памятью. Но внутри процессора данные должны передаваться в АЛУ для обработки. АЛУ имеет доступ как к регистру MBR, так и к программно доступным регистрам. Но возможен и такой вариант структуры процессора, в котором существуют дополнительные буферные регистры на стыке АЛУ с другими компонентами. Эти буферные регистры играют роль входных и выходных для АЛУ и обмениваются данными с MBR и с программно доступными регистрами.

Во всех моделях процессоров существуют один или несколько регистров, хранящих *слово состояния программы* (PSW —*program status word*). В состав PSW входят биты кодов условий и некоторая другая информация о текущем состоянии программы (фактически, процессора). В большинстве моделей в PSW присутствуют если не все, то уж обязательно некоторые из перечисленных ниже флагов или полей.

- *S* (*sign — знак*). Содержит знаковый разряд результата последней арифметической операции.
- *Z* (*zero — нуль*). Устанавливает в 1, если результат последней операции равен нулю.
- C (*carry — перенос*). Устанавливает в 1, если в результате последней арифметической операции возник перенос (при сложении) или заем (при вычитании) из старшего значащего разряда. Используется при выполнении арифметических операций над многословными операндами. 1

- E (*equal — равенство*). Устанавливает в 1, если при логическом сравнении обнаружено равенство операндов.
- O (*overflow — переполнение*). Индикатор переполнения при выполнении последней арифметической операции.
- I (*interrupt enable/disable — разрешение/блокировка прерывания*). Используется для разрешения или блокировки прерывания.
- SV (*supervisor — режим супервизора*). Индикатор привилегированного режима выполнения программы (режима супервизора). В режиме супервизора разрешается выполнение некоторых привилегированных машинных команд и доступ к защищенным областям памяти.

В конкретных моделях процессоров можно встретить и другие регистры, хранящие информацию о текущем состоянии процессора и управляющие режимом его функционирования. Помимо PSW, это может быть регистр указателя блока памяти, содержащего дополнительную информацию о состоянии (т.е. *блока управления процессом*). В состав процессоров, работающих с векторными прерываниями, включается регистр вектора прерывания. Если для выполнения некоторых функций используется стек, в составе процессора должен быть регистр указателя системного стека. Регистр указателя таблицы страниц необходим в системах, работающих с виртуальной памятью. Определенные регистры нужны для управления операциями ввода-вывода.

При проектировании этой части процессора нужно учитывать множество факторов. Ключевым моментом является необходимость поддержки операционной системы. Определенная управляющая информация представляет особый интерес для операционной системы. Если разработчик хорошо представляет себе принципы работы опера-

ционной системы, которая будет использоваться с проектируемым процессором, то он соответствующим образом выберет организационную структуру регистров. Другой, не менее важный вопрос — распределение управляющей информации между памятью и регистрами. Общепринятым является использование первых нескольких сотен или тысяч слов в памяти для целей управления. Проектировщик должен решить, какая часть управляющей информации будет храниться в регистрах, а какая — в памяти. Как всегда, приходится искать компромисс между затратами (стоимостью компонентов) и быстродействием.

5.8.3. Примеры организации регистров в микропроцессоре

Проанализируем и сравним организацию регистров в разных моделях процессоров примерно одного класса: 16-разрядных микропроцессоров, Motorola MC68000 и Intel 8086. Организация регистров микропроцессора MC68000 показана схематически на рис. 1, *а),* а микропроцессора Intel 8086 — на рис. 1, б). *Регистры,* имеющие отношение к абсолютно внутренним функциям процессора, как, например регистр адреса в памяти, на схемах не показаны.

В микропроцессоре MC68000 имеется восемь 32-разрядных регистров данных и девять регистров адреса такой же длины. Хотя основное назначение регистров данных — манипуляция со значениями операндов, они используются и в качестве индексных регистров при формировании исполнительного адреса. Разрядность регистров позволяет работать с 8-, 16- или 32-разрядными элементами данных в зависимости от кода операции. Регистры адреса содержат 32-разрядные адреса (без сегментации). Два из них используются в качестве указателей стека: один — стека прикладной программы, а другой — стека операционной системы в соответствии с текущим режимом выполнения программы. Оба регистра имеют номер 7, поскольку в каждый

момент времени может быть активным только один из них. В состав MC68000 входит также 32-разрядный счетчик команд и 16-битовый регистр состояния.

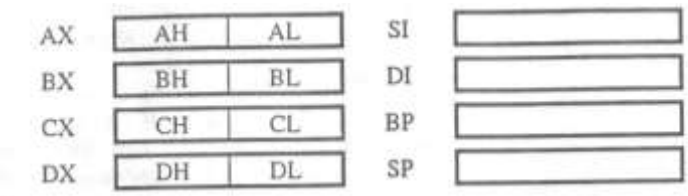
Разработчики фирмы Motorola создали систему команд с регулярной структурой, в которой отсутствуют регистры специального назначения. Стремление обеспечить высокую эффективность программы в машинном коде привела их к мысли разделить регистры на две функциональные группы и таким образом сэкономить один бит в поле спецификации регистра.

Разработчики микропроцессора Intel 8086 использовали при организации регистров другой подход. Каждый регистр имеет свое назначение, хотя некоторые из них могут использоваться и в качестве универсальных.

Все регистры имеют размер слова (16 разрядов), за каждым из них закреплено определенное имя (AX, SP и т, п.). По назначению и способу использования регистры можно разбить на следующие группы:

- регистры общего назначения (AX, BX, CX, DX, SI, DI, BP, SP);
- сегментные регистры CCS, DS, SS, ES);
- указатель команд (IP);
- указатель флагов (Flags).

К *регистрам общего назначения* относятся следующие 8 регистров:



AX accumulator, аккумулятор;

BX base, база;

CX counter, счетчик;

DX data, данные;

(буква X - от слова eXtended, расширенный: в процессоре 8080 были байтовые регистры A, B, C и D, но затем их расширили до размера слова)

SI source index, индекс источника;

DI destination index, индекс приемника;

BP base pointer, указатель базы;

SP stack pointer, указатель стека.

Особенностью всех этих регистров является то, что их можно использовать в любых арифметических, логических и т. п. машинных операциях. Например, можно сложить число из регистра DI с числом из регистра SP или вычесть из содержимого регистра BP содержимое регистра CX.

В то же время каждый из этих регистров имеет определенную специализацию: некоторые команды требуют, чтобы их операнд или операнды обязательно находились в определенных регистрах. Например, команда деления требует, чтобы первый операнд (делимое) находился в регистре AX или в регистрах AX и DX (в зависимости от размера операнда), а команды управления циклом используют регистр CX в качестве счетчика цикла.

Регистр SP используется при работе со стеком. Стек - это хранилище информации, функционирующее по правилу: первым из стека всегда считывается элемент, записанный в стек последним. Стек полезен во многих, случаях, например, для реализации процедур. В ПК имеются команды, поддерживающие работу со стеком. Так вот, в этих

командах предполагается, что регистр SP указывает на ячейку стека, в которой находится элемент, записанный в стек последним.

Регистры AX, BX, CX и DX устроены так, что возможен независимый доступ к их старшей и младшей половинам; каждый из этих регистров состоит из двух байтовых регистров. Обозначают эти половины буквами H (high - выше, старший) и L (low - ниже, младший) и первой буквой из названия регистра: AH и AL - в AX, BH и BL - в BX и т. д. Например, с регистром AX можно работать так: можно записать в него слово (16 битов), затем можно считать только левую половину этого слова (байт из AH), не считывая правую половину, далее можно сделать запись только в AL, не меняя AH. Таким образом, возможен доступ как ко всему регистру AX, так и к любой его половине. Целиком регистр используется при работе с числами, а его половины - при работе с частями чисел или символами.

На части делятся только регистры AX, BX, CX и DX, Запись же в другие регистры и чтение из них осуществляются только целиком.

К *сегментным регистрам* относятся следующие

названия которых расшифровываются так:

CS code segment, сегмент команд;

DS data segment, сегмент данных;

SS stack segment, сегмент стека;

ES extra segment, дополнительный сегмент.

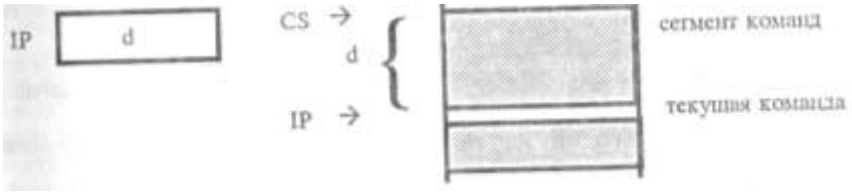
Ни в каких арифметических, логических и т. п. операциях эти регистры не могут участвовать. Можно только записывать в них и считывать из них, да и то здесь есть определенные ограничения.

Эти регистры используются для сегментирования адресов, которое является разновидностью модификации адресов и которое используется для сокращены размера команд. Суть дела здесь в следующем.

Если в ЭВМ используется память большого объема, тогда для ссылок на ее ячейки приходится использовать "длинные" адреса, а поскольку эти адреса указываются в командах, то и команды оказываются "длинными". Это плохо, т. к увеличиваются размеры машинных программ. Сократить размеры команд при "длинных" адресах" можно, например, так. Любой адрес A можно представит в виде суммы B+D, где B - начальный адрес (база) того участка (сегмента памяти, в котором находится ячейка A, а D - это смещение, адрес ячейки A отсчитанный от начала этого сегмента (от B). Если сегменты памяти небольшие тогда и величина D будет небольшой, поэтому большая часть "длинного" адрес; A будет сосредоточена в базе B. Этим и можно воспользоваться: если в команде надо указать адрес A, тогда "упрятываем" базу B в какой-нибудь регистр S, а в команде вместо A указываем этот регистр и смещение D. Поскольку для записи D надо меньше места, чем для адреса A, то тем самым уменьшается размер команды. С другой стороны, благодаря модификации адресов данная команда будет работать с адресом, равным сумме D и содержимого регистра S, т. е с нужным нам адресом A.

Рассмотренный способ задания адресов в командах называется сегментированием адресов (другое название – базирование адресов), а регистры, используемые для хранения начальных адресов сегментов памяти, – сегментными. В ПК в качестве сегментных регистров ложно использовать не любой регистр, а только один из следующих четырех: CS, DS, SS и ES.

В регистре CS должен находиться начальный адрес сегмента команд – той области памяти, где расположены команды программы; регистр DS должен указывать на начало сегмента данных, в котором размещаются данные программы; регистр SS должен указывать на начало области памяти, отведенной под стек. Если так и сделать, тогда при ссылках на эти сегменты (команд, данных и стека) можно явно не указывать в командах соответствующие сегментные регистры (CS, DS и SS), они будут подразумеваться по умолчанию.



В регистре IP (instruction pointer) всегда находится адрес команды, которая должна быть выполнена следующей, т. е. в IP находится адрес этой команды, отсчитанный от начала сегмента команд, на начало которого указывает регистр CS. Поэтому абсолютный адрес этой команды определяется парой регистров CS и IP. Изменение любого из этих регистров есть ничто иное, как переход. Поэтому содержимое регистра IP (как и CS) можно менять только командами перехода.

Флаг – это бит, принимающий значение 1 ("флаг установлен"), если выполнено некоторое условие, и значение 0 ("флаг сброшен") в противном случае. В ПК используется 9 флагов, причем конструктивно они собраны в один 16-разрядный регистр, называемый регистром флагов и обозначаемый как Flags. Каждый флаг — это один из разрядов данного регистра (некоторые разряды регистра не заняты):

Flags					OF	DF	IF	TF	SF	ZF		AF		PF		CF
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Некоторые флаги называются флагами условий: они автоматически меняются при выполнении команд и фиксируют те или иные свойства их результата. Другие флаги называются флагами состояний; сами по себе они не меняются и менять их должна программа; состояние этих флагов оказывает влияние на дальнейшее поведение процессора.

Флаги условий:

CF (*carry* flag) – флаг переноса. Наиболее полезен в арифметических операция над числами без знака; например, если при сложении беззнаковых чисел получилась слишком большая сумма - с единицей переноса, которой нет места в ячейке, тогда флаг CF принимает значение 1, а если сумма "укладывается" в размер ячейки, то значением CF будет 0.

OF (*overflow flag*) – флаг переполнения. Полезен в арифметических операция над числами со знаком; например, если при сложении или вычитании знаковых чисел получился результат, по модулю превосходящий допустимую величину (произошло переполнение мантиссы), тогда флаг 0 получает значение 1, а если переполнения мантиссы не было - значение 0.

ZF (*zero flag*) - флаг нуля. Устанавливается в 1, если результат команды оказался нулевым.

SF (*sign flag*) – флаг знака. Устанавливается в 1, если в операции над знаковым числами получился отрицательный результат.

PF (*parity flag*) – флаг четности. Равен 1, если в 8 младших битах результат очередном команды содержится четное количество двоичных единиц. Учитывается обычно только в 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 начата к концу), при DF=1 – в обратном направлении.

IF (*interrupt flag*) – флаг прерываний. При IF=0 процессор перестает реагировать на поступающие к нему прерывания, а при IF=1 блокировка прерывали снимается.

TF (*trap flag*) – флаг трассировки. При TF=1 после выполнения каждой команды процессор делает прерывание, чем можно воспользоваться при отладке программы – для ее трассировки.

Из описания, данного выше, следует, что не существует единого общепринятого подхода к организации регистров в процессоре. Эта проблема до сих пор является темой оживленных дискуссий и экспериментов.

Второй вывод, касающийся организации регистров в процессоре, иллюстрируется схемой регистров процессора Intel 80386, представленной на рис. 5.9, в). Этот микропроцессор разрабатывался как 32-разрядная модификация процессора 8086. В состав процессора 80386 входят 32-разрядные регистры, но чтобы сохранить обратную совместимость программного обеспечения, в нем сохранена прежняя схема распределения функций между регистрами.

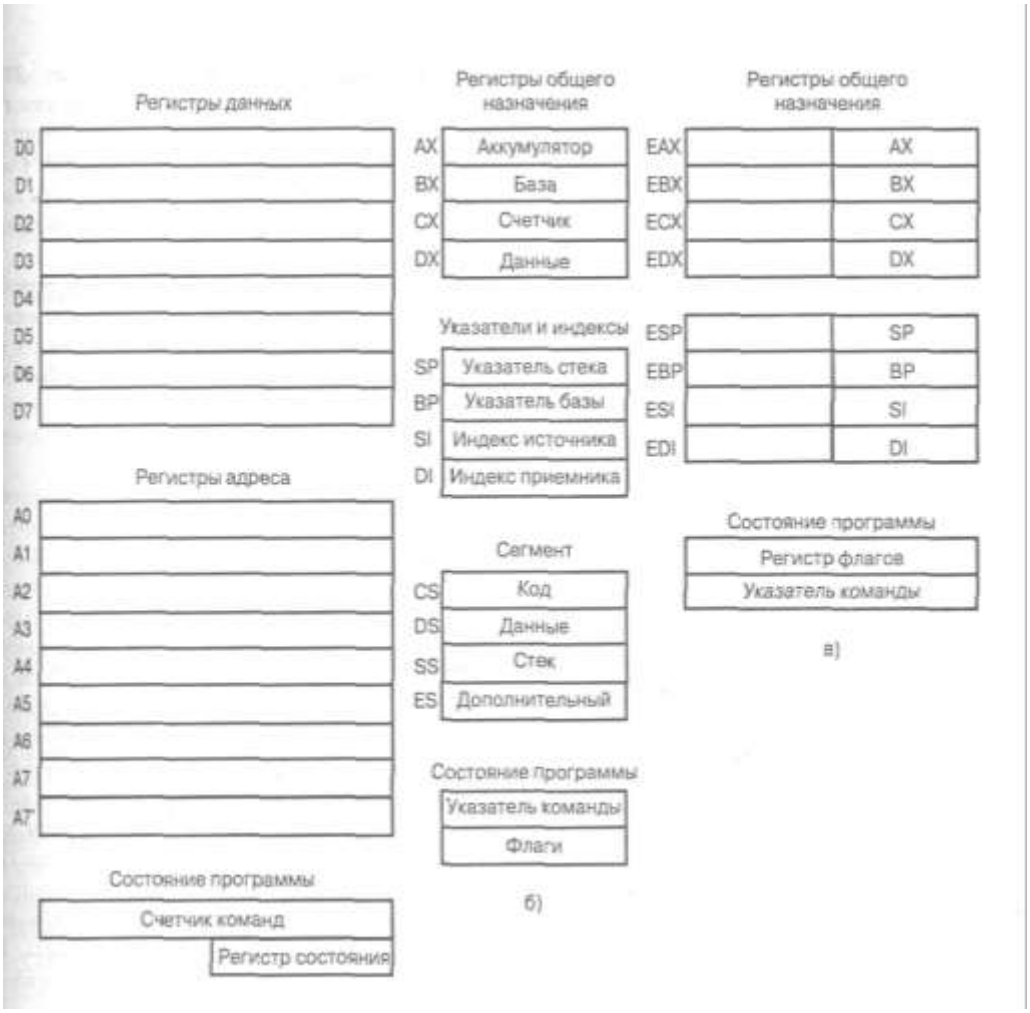


Рис. 5.9. *Организация регистров в микропроцессорах:*

а) Motorola MC68000; б) Intel 8086; в) Intel 80386, 80486, Pentium, Pentium II

5.9. Устройство управления

Для спецификации функций процессора необходима следующая информация:

• перечень операций (кодов операций в машинных командах);

• перечень режимов адресации;

• программно доступные регистры;

• интерфейс модулей ввода-вывода;

• интерфейс модулей памяти;

• сведения об обработке прерываний.

Задача выяснить, как эти функции реализуются, или точнее, как организуется совместная работа разных компонентов процессора при реализации этих функций, возлагается на устройство управления, входящее в состав процессора.

Процесс выполнения программы является последовательностью циклов выполнения отдельных команд, причем сам цикл разбивается на более мелкие компоненты: фазы извлечения, косвенной адресации, выполнения и прерывания.

Каждая более мелкая фаза включает в том или ином виде операции с регистрами процессора. Такие операции называются *микрооперациями. Микро-* означает, что каждая операция очень проста и совершается очень быстро.

Набор микроопераций можно разделить на следующие категории:

- пересылка данных из одного регистра в другой;
- пересылка данных из регистра в узел внешнего интерфейса (на внешнюю магистраль);
- пересылка данных из узла внешнего интерфейса в регистр;
- выполнение арифметических и логических операций, в которых в качестве исходных данных используется содержимое регистров, а результаты также передаются в регистры.

Эти категории охватывают все виды микроопераций, необходимые для прохождения цикла обработки команд, в том числе и для их выполнения.

Каждый цикл выполнения команды начинается с извлечения команды из подсистемы памяти. В этих операциях используются четыре регистра:

• MBR — *регистр данных памяти.* Этот регистр подсоединен к адресным линиям системной магистрали, и код в нем определяет адрес ячейки памяти для операций записи или чтения.

• MAR — *регистр адреса памяти.* Регистр подсоединен к линиям данных системной магистрали и содержит код, который должен быть сохранен в памяти (операция записи) или извлечен из памяти после выполнения последней по времени операции чтения.

• PC — *счетчик команд.* Регистр содержит адрес следующей команды, которая должна быть извлечена из памяти.

• IR — *регистр текущей команды.* Регистр содержит последнюю извлеченную из памяти команду.

Для того чтобы устройство управления могло решать возложенные на него задачи, оно должно располагать определенной входной информацией о текущем состоянии системы и формировать выходные сигналы, которые будут определять дальнейшее поведение системы. Это спецификация устройства управления извне. Если же рассматривать внутреннюю структуру узла, то в ней должны присутствовать логические схемы, необходимые для решения перечисленных выше задач.

В качестве входной информации устройство управления использует следующие сигналы и данные.

- *Тактовые импульсы.* Тактовые импульсы синхронизируют выполнение всех операций во времени. Длительность выполнения каждой микрооперации (или нескольких одновременно выполняемых микроопераций) равна одному такту.
- *Регистр команд.* Поле кода операции в регистре команды определяет, какую последовательность микроопераций должно сформировать устройство управления.

- *Флаги.* Они необходимы для определения текущего состояния процессора и результата выполнения последней операции в АЛУ.
- *Сигналы на управляющих линиях магистрали.* По этим линям в устройство управления поступают сигналы запроса прерывания и подтверждения от внешних устройств.

Устройство управления формирует следующие сигналы.

- *Управляющие сигналы, распространяемые внутри процессора.* Эти сигналы делятся на две категории: сигналы, задающие передачу данных из одного регистра в другой, и сигналы запуска определенных операций в АЛУ.
- *Сигналы, передаваемые по управляющим линиям системной магистрали.* Эти сигналы также делятся на две категории: сигналы, управляющие работой памяти, и сигналы, управляющие работой модулей ввода-вывода.

С помощью управляющих сигналов устройство управления организует работу процессора при излечении команды. Устройство постоянно следит за тем, какая фаза цикла обработки команды является текущей в данный момент. По состоянию регистра ICC (instruction cycle code) оно определяет, что должна выполняться фаза извлечения. Первым делом устройство организует пересылку содержимого PC в MAR. Для этого формируются управляющие сигналы, открывающие вентили между разрядами PC и разрядами MAR. Следующая операция — чтение слова из памяти в MBR и приращение кода адреса в PC. Для этого устройство управления одновременно формирует такие сигналы:

- сигнал, который открывает вентили, связывающие выходы разрядов регистра MAR с линиями адреса системной магистрали;

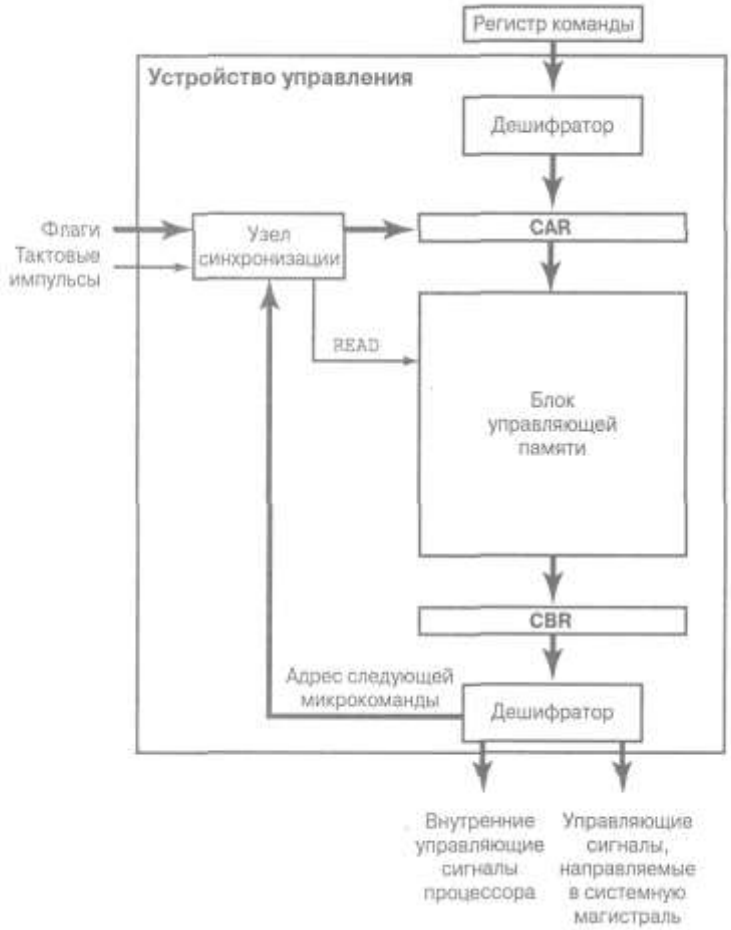


Рис. 5.10. *Схема микропрограммного устройства управления.*

- сигнал READ на линии управления системной магистрали;
- сигнал, который открывает вентили, связывающие линии данных системной магистрали со входами разрядов регистра MBR;
- внутренний управляющий сигнал процессора, инициирующий работу схемы приращения содержимого регистра PC и сохранение результата в PC.

Этим практически завершается выполнение операций, необходимых для извлечения команды. Остается еще одна операция — устройство управления должно решить, следует ли переходить к фазе косвенной адресации или, минуя ее, сразу к фазе выполнения команды. Для этого оно анализирует содержимое поля адреса в регистре команд IR.

Таким же образом организуется работа процессора и модулей ввода-вывода в фазах косвенной адресации и прерывания. В фазе выполнения устройство управления сначала анализирует код текущей команды в регистре IR. В зависимости от этого кода выбирается та или иная последовательность микроопераций.

Процесс проектирования устройства управления можно разделить на три стадии.

1. Определение базовых функциональных элементов процессора.
2. Составление перечня микроопераций, которые процессор будет выполнять.
3. Определение требующихся от устройства управления действий для организации выполнения последовательности микроопераций.

На рис. 5.10 показаны основные узлы устройства, реализующего микропрограммное управление. Последовательность микрокоманд, образующая микропрограмму, хранится в *блоке управляющей памяти* Регистр *адреса управляющей памяти* (*control address register* — CAR) содержит адрес следующей микрокоманды. Микрокоманда, считанная из блока управляющей памяти, помещается в *буферный регистр управляющей памяти* (*control buffer register* — CBR). Левая часть этого регистра подсоединена к выходным линиям управляющих сигналов. Следовательно, чтение микрокоманды является одновременно и ее выполнением. Третий узел — узел синхронизации (синхронизация — координация сигналов во времени) и управления последовательностью выполнения микропрограммы — организует загрузку в CAR и формирует сигнал чтения из блока управляющей памяти. Эта схема принимает те же входные сигналы — содержимое IR, флаги АЛУ и тактовые импульсы, формирует такие же выходные сигналы. Работает микропрограммное устройство управления следующим образом.

1. Узел синхронизации формирует сигнал READ (чтение), который поступает в блок управляющей памяти.
2. Микрокоманда, адрес которой установлен в регистре CAR, считывается в регистр CBR.
3. На основании содержимого CBR формируются управляющие сигналы и адрес следующей микрокоманды.
4. Узел синхронизации анализирует флаги АЛУ и загружает адрес следующей микрокоманды в CAR.

Все эти действия производятся в течение одного такта. В зависимости от значений флагов АЛУ и кода в CBR узел синхронизации выбирает один из трех возможных вариантов.

- *Извлечь микрокоманду, размещенную по следующему адресу в блоке управляющей памяти.* Для этого необходимо добавить 1 к содержимому CAR.
- *Перейти на новую микропрограмму.* Для этого код из поля адреса регистра CBR загружается в CAR.
- *Перейти на микропрограмму выполнения определенной машинной команды.* Для этого в CAR загружается адрес, соответствующий коду операции в регистре IR процессора.

На рис. 5.9 показаны два узла дешифраторов. Верхний дешифратор преобразует код операции из регистра IR в адрес в блоке управляющей памяти. Нижний дешифратор используется только в тех устройствах управления, в которых применяется *вертикальное микропрограммирование*.

5.10. Функциональные характеристики персонального компьютера

Основными характеристиками персонального компьютера (ПК) являются:

1. *Быстродействие, производительность, тактовая частота.*

2. *Разрядность машины и кодовых шин интерфейса.*

Разрядность — это максимальное количество разрядов двоичного числа, над которым одновременно может выполняться машинная операция, в том числе и операция передачи информации; чем больше разрядность, тем будет больше и производительность ПК.

3. *Типы системного и локальных интерфейсов.*

Разные типы интерфейсов обеспечивают разные скорости передачи информации между узлами машины, позволяют подключать разное количество, внешних устройств и различные их виды.

4. *Емкость оперативной памяти.*

Емкость оперативной памяти измеряется чаще всего в мегабайтах (Мбайт), реже в килобайтах (Кбайт). Многие современные прикладные программы при оперативной памяти емкостью меньше 8 Мбайт просто не работают либо работают, но очень медленно.

Увеличение емкости основной памяти в 2 раза, дает повышение эффективной производительности ЭВМ при решении сложных задач примерно в 1,7 раза.

5. *Емкость накопителя на жестких магнитных дисках (винчестера).*

Емкость винчестера измеряется обычно в мегабайтах или гигабайтах.

6. *Тип и емкость накопителей на гибких магнитных дисках.*

Сейчас применяются в основном накопители на гибких магнитных дисках, использующие дискеты диаметром 3,5 (1 дюйм = 25,4 мм). Они имеют стандартную емкость 1,44 Мбайта.

7. *Виды и емкость КЭШ-памяти.*

КЭШ-память — это буферная, не доступная для пользователя быстродействующая память, автоматически используемая компьютером для ускорения операций с информацией, хранящейся в более медленно действующих запоминающих устройствах. Например, для ус-

корения операций с основной памятью организуется регистровая КЭШ-память внутри микропроцессора (КЭШ-память первого уровня) или вне микропроцессора на материнской плате (КЭШ-память второго уровня); для ускорения операций с дисковой памятью организуется КЭШ-память на ячейках электронной памяти. Наличие КЭШ-памяти емкостью 256 Кбайт увеличивает производительность ПК примерно на 20%.

8. *Тип видеомонитора (дисплея) и видеоадаптера.*

9. *Тип принтера.*

10. *Наличие математического сопроцессора.*

Математический сопроцессор позволяет в десятки раз ускорить выполнение операций над двоичными числами с плавающей запятой и над двоично-кодированными десятичными числами.

11. *Имеющееся программное обеспечение и вид операционной системы*

12. *Аппаратная и программная совместимость с другими типами ЭВМ.*

Аппаратная и программная совместимость с другими типами ЭВМ означает возможность использования на компьютере соответственно тех же технических элементов и программного обеспечения, что и на других типах машин.

13. *Возможность работы в вычислительной сети*.

14. *Возможность работы в многозадачном режиме.*

Многозадачный режим позволяет выполнять вычисления одновременно по нескольким программам (многопрограммный режим) или для нескольких пользователей (многопользовательский режим). Совмещение во времени работы нескольких устройств машины, возможное в таком режиме, позволяет значительно увеличить эффективное быстродействие ЭВМ.

15. *Надежность.*

Надежность — это способность системы выполнять полностью и правильно все заданные ей функции. Надежность ПК измеряется обычно средним временем наработки на отказ.

16. *Стоимость.*

17. *Габариты и масса.*

6. Запоминающие устройства ПК

6.1. Характеристики запоминающих устройств

6.1.1. Размещение

Запоминающие устройства (ЗУ) по месту размещения в компьютере делятся на *внутренние* и *внешние.* Внутренние ЗУ: оперативная память; небольшая *локальная* память в виде *регистров* в составе процессора; собственная внутренняя память для хранения микропрограмм в устройстве управления; *кэш-память* (компонент современных процессоров). К *внешним ЗУ* относят периферийные устройства хранения информации, такие, как накопители на магнитных дисках и лентах, для доступа к которым процессору нужно обратиться к соответствующему модулю ввода-вывода.

6.1.2. Емкость

Емкость — одна из основных естественных характеристик любого ЗУ. Общепринятой единицей измерения информационной емкости является 8-битовый байт, но емкость внутренних ЗУ измеряется и в словах. Длина слова в современных компьютерах кратна 1 байт — 1, 2 или 4 байт (8, 16 или 32 бит). Емкость внешней памяти всегда оценивается в байтах.

6.1.3. Передаваемая порция

При каждом обращении к внутреннему ЗУ передается (считывается из ЗУ или записывается в ЗУ) определенная порция данных. Как правило, размер порции соответствует длине слова ЗУ, но это не обязательно. В большинстве компьютеров размер слова оперативной памяти равен количеству разрядов, используемых для представления целого числа и машинной команды.

• *Адресуемая единица.* Во многих системах адресуемой единицей также является слово, но иногда можно адресовать и отдельный байт, хотя слово имеет длину 2 байт или более, В любом случае между длиной (разрядностью) кода адреса A и количеством адресуемых единиц N соблюдается соотношение $2^A=N$.

• *Передаваемая порция.* Для оперативного ЗУ этот параметр характеризует количество битов информации, считываемых или записываемых за один цикл обращения, причем он не обязательно должен равняться длине слова или адресуемой единицы. При обращении к внешним ЗУ передаваемая порция имеет значительную длину, намного превышающую длину слова, и принято говорить, что обмен выполняется *блоками.*

6.1.4. Метод доступа

Различают следующие методы доступа к хранящейся в ЗУ информации.

- *Последовательный доступ* (*sequential access*). Хранимая информация разделена на элементы, называемые *записями* (*records*). Помимо собственно данных хранится и дополнительная адресная информация, которая служит для разделения записей и используется в процессе поиска нужной записи. Для доступа используется совмещенный переключаемый механизм чтения/записи, который перемещается из текущего положения к нужной записи, проходя все промежуточные. В результате время доступа к нужной записи может изменяться в довольно широких пределах и зависит от исходного положения механизма чтения/записи и положения искомой записи. Последовательный доступ характерен для некоторых конструкций внешних ЗУ, в частности — накопителей на магнитной ленте.

- *Прямой доступ* (*direct access*). При прямом доступе используется совмещенный механизм чтения/записи. Но каждая запись имеет свой уникальный адрес, соответствующий ее физическому положению на носителе. Доступ осуществляется прямым обращением к зоне носителя, в которой находится адресуемая запись, и далее последовательно просматриваются записи внутри этой зоны, пока механизм чтения/записи не будет совмещен с искомой записью. Время доступа к произвольной записи переменно, хотя разброс значений и не так значителен, как при последовательном доступе. Прямой доступ реализован в конструкциях тех внешних ЗУ, которые используют в качестве носителя информации диски с концентрическими дорожками.
- *Произвольный доступ* (*random access*). Каждая адресуемая единица имеет свой встроенный механизм адресации. Время доступа к любой единице (ячейке) не зависит от предыстории и от адреса этой ячейки. Таким образом, возможен непосредственный доступ к ячейкам в произвольном порядке с минимальными потерями времени. Именно такой способ доступа используется в оперативных ЗУ.
- *Ассоциативный доступ* (*associative access*). Существуют такие ЗУ с произвольным доступом, которые оснащены встроенным механизмом сравнения определенных битов в каждой ячейке с заданным образцом, причем сравнение выполняется по всем ячейкам одновременно. Нужная ячейка отыскивается не столько по своему физическому адресу, сколько по содержимому. Как и в обычном ЗУ с произвольным доступом, время доступа не зависит от адреса ячейки или предыстории. Такой метод доступа получает все большее распространение в конструкциях блоков кэш-памяти.

6.1.5. Производительность

Для оценки производительности используется три параметра.

- *Время доступа* (*access time*). Для ЗУ с *произвольным* доступом этот параметр характеризует время выполнения операций чтения или записи, и измеряется как длительность временного интервала между моментом, когда в устройство передан адрес, и моментом, когда данные зафиксированы в памяти (при выполнении записи) или когда ими может воспользоваться другое устройство компьютера (при выполнении чтения). Для ЗУ с *последовательным* или *прямым* доступом этот параметр характеризует время, необходимое для перевода механизма чтения/записи в нужную позицию по отношению к носителю информации.
- *Длительность цикла обращения к памяти* (*memory cycle time*). Этот параметр характеризует ЗУ с произвольным доступом. Он измеряется минимальной длительностью временного интервала между последовательными сеансами доступа к памяти. Время цикла включает время доступа плюс время выполнения дополнительных операций, связанных с подготовкой устройства к следующему обращению.
- *Скорость передачи* (*transfer rate*). Этот параметр характеризует интенсивность информационного потока между ЗУ и теми устройствами, которые нуждаются в хранящейся в нем информации. Для ЗУ с произвольным доступом скорость передачи есть величина, обратная *длительности цикла обращения.* Для ЗУ с последовательным или прямым доступом справедливо соотношение:

$$T_N = T_A + \frac{N}{R},$$

где

T_N — среднее время чтения или записи блока данных объемом в N бит;

T_A — среднее время доступа для данного ЗУ;

N — объем блока данных (бит);

R — скорость передачи (бит/с).

6.1.6. Физические типы запоминающих устройств

- *Полупроводниковые*, в которых хранение информации производится либо в электронных элементах с двумя устойчивыми состояниями, либо в запоминающих полупроводниковых емкостях;
- *магнитные,* где для хранения используется эффект перемагничивания магнитного материала, нанесенного на какую-либо поверхность (ленту, диск, барабан и т.п.);
- *оптические*;
- *магнитооптические*.

Все магнитные, оптические и магнитооптические ЗУ (точнее, носители информации) являются энергонезависимыми. Что касается полупроводниковых ЗУ, то большинство их конструкций требует наличия электропитания для сохранения информации, но существуют и специальные конструкции, сохраняющие информацию при отключении питания. Среди последних особое место отводится постоянным ЗУ, или в современной терминологии — *ROM-памяти* (*read only memory*). Информация в ROM-память записывается на стадии изготовления компонента или включения его в компьютерную систему, а затем может только считываться.

6.2. Базовая система ввода-вывода

BIOS (Basic Input/Output System) — это термин, который используется для описания *базовой системы ввода-вывода.* По существу, BIOS представляет собой "промежуточный слой" между программной и аппаратной частями системы. Большинство пользователей под BIOS подразумевают *драйверы устройств.* Кроме системной, существует еще BIOS адаптеров, которые загружаются при запуске системы.

Базовая система ввода-вывода — это комбинация всех типов BIOS, а также загружаемые драйверы устройств.

Во всех системных платах есть микросхема, в которой записано программное обеспечение, называемое BIOS или ROM BIOS. Эта микросхема содержит стартовые программы и драйверы, необходимые для запуска системы и функционирования основного аппаратного обеспечения. В ней также содержится процедура POST (самотестирование при включении питания) и данные системной конфигурации. Все эти параметры записаны в CMOS-память, которая питается от батарейки, установленной на системной плате. Эту CMOS-память часто называют NVRAM (Non-Volatile RAM).

BIOS представляет собой комплект программ, хранящихся в одной или нескольких микросхемах. Эти программы выполняются при запуске компьютера до загрузки операционной системы. BIOS в большинстве PC-совместимых компьютеров выполняет четыре основные функции:

- *POST*— самотестирование при включении питания процессора, памяти, набора микросхем системной логики, видеоадаптера, контроллеров диска, дисковода, клавиатуры и других жизненно важных компонентов системы;

- *программа установки параметров BIOS* (*Setup BIOS*) — конфигурирование параметров системы. Эта программа запускается при нажатии определенной клавиши (или комбинации клавиш) во время выполнения процедуры POST;
- *начальный загрузчик системы*— выполнение поиска главного загрузочного сектора на дисковых устройствах;
- *BIOS* — набор драйверов, предназначенных для взаимодействия операционной системы и аппаратного обеспечения при загрузке системы. При запуске DOS или Windows в режиме защиты от сбоев используются драйверы устройств только из BIOS.

Операционная система, в свою очередь, через BIOS обращается непосредственно к аппаратному обеспечению. Эта связь реализована в виде драйверов устройств. Причем в каждой операционной системе — DOS, Windows *9x*, Windows NT, Windows 2000, OS/2, Linux или другой — для одного и того же устройства необходимы свои драйверы.

Связь между приложениями и операционной системой осуществляется с помощью соответствующего API (Application Programming Interface). Этот интерфейс определяет, например, как выполняется запись и считывание данных на диск, печать и другие функции. Поскольку приложение не зависит от установленного аппаратного обеспечения, то все его вызовы обрабатывает операционная система, которая уже содержит информацию об установленном оборудовании.

Практически все современные производители BIOS предоставляют ее код производителям системных плат и готовых компьютеров.

На разработке PC-совместимых программ ROM BIOS специализируются такие компании, как American Megatrends, Inc. (AMI), Award Software и Phoenix Software. Изготовители системных плат получают от них лицензии на установку ROM BIOS, после чего могут

работать над аппаратной частью, не занимаясь программным обеспечением.

6.3. Оперативная память: структурная организация

Оперативная память — это рабочая область для процессора компьютера. В ней во время работы хранятся программы и данные. Оперативная память часто рассматривается как временное хранилище, потому что данные и программы в ней сохраняются только при включенном компьютере или до нажатия кнопки сброса (reset). Перед выключением или нажатием кнопки сброса все данные, подвергнутые изменениям во время работы, необходимо сохранить на запоминающем устройстве, которое может хранить информацию постоянно (обычно это жесткий диск). При новом включении питания сохраненная информация вновь может быть загружена в память.

Устройства оперативной памяти иногда называют *запоминающими устройствами с произвольным доступом* (RAM-ЗУ – *random access memory*). От прочих типов их отличает возможность выполнять операции двух видов — запись информации в ячейку по произвольному адресу и считывание содержимого ячейки по произвольному адресу. В выполнении обеих операций применяются только электронные схемы (т.е. не используются никакие электромеханические узлы). Это означает, что обращение к данным, хранящимся в оперативной памяти, не зависит от порядка их расположения в ней.

Во время выполнения программы в оперативной памяти хранятся ее данные. Микросхемы оперативной памяти иногда называют энергозависимой памятью: после выключения компьютера данные, хранимые в них, будут потеряны, если они предварительно не были хранены на диске или другом устройстве внешней памяти. Чтобы из-

бежать этого, некоторые приложения автоматически делают резервные копии данных.

Базовым элементом ОП является *запоминающий элемент* (*ЗЭ*) – ячейка памяти. Свойства:

- элемент может находится в одном из двух устойчивых (или квазиустойчивых) состояний, одно из которых интерпретируется как хранение двоичного кода 1, а другое — кода 0;
- в элемент хотя бы один раз можно записать нужный двоичный код и таким образом целенаправленно установить его состояние;
- текущее состояние элемента можно считывать.

Чаще всего элемент имеет три вывода, по которым могут передаваться электрические сигналы. Сигнал на выводе *Выборка*, как и следует из его названия, выбирает элемент для выполнения операции. Сигнал на выводе *Управление* задает режим обращения (выполняемую операцию) — запись или чтение. В режиме записи на третий вывод, *Ввод/считывание*, подается сигнал, соответствующий записываемому двоичному коду: 0 или 1. При чтении третий вывод является выходом, и на нем формируется сигнал, соответствующий текущему состоянию элемента: 0 или 1.

Запоминающие элементы поступают к конструктору в виде микросхемы, в которую "упаковано" множество ЗЭ и необходимые для поддержки их функционирования электронные схемы (обрамление).

Физически оперативная память в системе представляет собой набор микросхем или модулей, содержащих микросхемы, которые обычно подключаются к системной плате. Эти микросхемы или модули могут иметь различные характеристики и, чтобы функционировать

правильно, должны быть совместимы с системой, в которую устанавливаются.

6.4. Типы запоминающих устройств с произвольным доступом

В современных компьютерах используются запоминающие устройства трех основных типов:

• *ROM* (*Read Only Memory*). Постоянное запоминающее устройство — ПЗУ, не способное выполнять операцию записи данных;

• *DRAM* (*Dynamic Random Access Memory*). Динамическое запоминающее устройство с произвольным порядком выборки;

• *SRAM* (*Static RAM*). Статическая оперативная память.

6.4.1. Память типа DRAM

Динамическая оперативная память (Dynamic RAM — DRAM) используется в большинстве систем оперативной памяти современных персональных компьютеров. Основное преимущество памяти этого типа состоит в том, что ее ячейки упакованы очень плотно.

Элементом памяти динамического RAM-ЗУ является *полупроводниковый конденсатор.* Наличие или отсутствие заряда на конденсаторе интерпретируется как хранение двоичных величин 1 и 0. Запоминающий конденсатор может сколь угодно долго сохранять состояние 0 (заряд отсутствует), но из-за неизбежных утечек состояние 1 сохраняется только в течение ограниченного времени. Поэтому информацию в динамическом RAM-ЗУ нужно периодически *восстанавливать* (освежать — *refresh*). Этот процесс получил название регенерации информации в динамической памяти.

В устройствах DRAM для хранения одного бита используется только один транзистор и пара конденсаторов, поэтому они более вместительны, чем микросхемы других типов памяти.

Другой отличительной особенностью *DRAM* является необходимость постоянного энергопитания, т.е. это устройство является энергозависимым. При отключении питания информация, хранящаяся в таком ЗУ, теряется. Следовательно, динамическое ЗУ может использоваться только для временного хранения информации в процессе выполнения программы.

6.4.2. Назначение и принцип работы кэш-памяти (SRAM)

Повышение производительности отдельных компонентов компьютера привело к включению между оперативной полупроводниковой памятью (как правило, на основе DRAM-модулей) и процессором промежуточной быстродействующей памяти небольшой (по сравнению с общим объемом оперативной памяти) емкости — статической оперативной памяти (Static RAM — SRAM). Она названа так потому, что, в отличие от динамической оперативной памяти (DRAM), для сохранения ее содержимого не требуется периодической регенерации. SRAM имеет более высокое быстродействие, чем динамическая оперативная память, и может работать на той же частоте, что и современные процессоры. В *статических* RAM-ЗУ (SRAM-ЗУ) информация хранится в электронных схемах с двумя устойчивыми состояниями (триггерах), построенных на биполярных транзисторах.

Эта промежуточная память недоступна для программы в том смысле, что никак не может быть адресована машинными командами, от чего и произошло ее название *кэш-память* (*cache memory*) — *тайник*. В составе компьютера имеется оперативная память довольно большого объема и относительно невысокого быстродействия и блок

кэш-памяти, обладающий высоким быстродействием, но значительно меньшей емкостью, чем оперативная память.

В кэше временно хранится копия некоторого фрагмента информации из оперативной памяти. Когда процессору требуется получить от подсистемы памяти очередное слово (команду или операнд), первым делом проверяется, не содержится ли уже оно в кэше. Если да (такое событие принято называть кэш-попаданием — cache hit), то это слово считывается из кэша и передается процессору. Если нет (такое событие принято называть кэш-промахом — cache miss), то из оперативной памяти в кэш считывается блок фиксированной длины, в составе которого имеется и требуемое слово, а затем это слово передается процессору.

Чтобы минимизировать время ожидания при считывании процессором данных из медленной оперативной памяти, в современных персональных компьютерах обычно предусмотрены два типа кэш-памяти: *кэш-память первого уровня* и *кэш-память второго уровня*. Кэш-память первого уровня также называется *встроенным,* или *внутренним кэшем.* Кэш-память второго уровня называется *вторичным,* или *внешним кэшем.*

Кэш-память первого уровня

Во всех процессорах, начиная с 486-го, имеется встроенный (первого уровня) *кэш-контроллер* с *кэш-памятью* объемом 8 Кбайт в процессорах 486DX, а также 32, 64 Кбайт и более в современных моделях. Кэш-память первого уровня (или встроенный кэш) работает на частоте процессора.

Благодаря этому обмен данными с относительно медленной системной памятью значительно ускоряется. Процессору не нужно ждать, пока очередная порция программного кода или данных поступит из основной области памяти, а это приводит к ощутимому повы-

шению производительности компьютера. При отсутствии кэш-памяти такие паузы возникали бы довольно часто.

В современных процессорах встроенный кэш играет еще более важную роль, потому что он часто является единственным типом памяти во всей системе, который может работать синхронно с процессором. В большинстве современных процессоров используется множитель тактовой частоты, следовательно, они работают на частоте, в несколько раз превышающей тактовую частоту системной платы, к которой они подключены. Например, тактовая частота (1 ГГц), на которой работает процессор Pentium III, в семь с половиной раз превышает тактовую частоту системной платы, равную 133 МГц. Поскольку оперативная память подключена к системной плате, она также может работать только на тактовой частоте, не превышающей 133 МГц. В такой системе из всех видов памяти только встроенный кэш может работать на тактовой частоте 1 ГГц.

Если необходимые данные во встроенном кэше отсутствуют, процессор обращается за ними в кэш-память второго уровня или непосредственно к системной шине.

Кэш-память второго уровня

Наличие двух типов кэш-памяти приводит к тому, что 90% времени данные извлекаются из кэш-памяти первого уровня (система работает на максимальной частоте; в рассматриваемом примере — 233 МГц), а 10% — из кэш-памяти второго уровня (система работает на частоте системной платы; в рассматриваемом примере — 66 МГц), т.е. для увеличения производительности системы в первую очередь необходимо увеличить объем кэш-памяти второго уровня.

В процессорах Pentium (P5) кэш-память второго уровня располагается на системной плате и работает на ее частоте. Существенное повышение производительности процессоров произошло после пере-

носа этой кэш-памяти с системной платы в процессор, т.е. ее рабочая частота сравнялась с частотой процессора. Вначале кэш-память и ядро процессора располагались на одном чипе, что влекло за собой существенное удорожание производства. Начиная с процессоров Pentium II корпорация Intel стала приобретать микросхемы кэш-памяти у сторонних производителей (Sony, Toshiba, NEC, Samsung и т.д.) и монтировать микросхему процессора и микросхемы кэш-памяти на плате, что повлекло изменение корпуса процессоров.

Рабочая частота самых производительных микросхем кэш-памяти составляет 333 МГц. Именно поэтому в старших моделях процессоров Pentium II и первых Pentium III кэш-память работает на половинной частоте процессора.

Эта память интегрирована с процессором и работает на той же частоте, что повышает производительность системы.

Организация работы кэш-памяти

Вся кэш-память разбивается на четыре блока, в каждом из которых хранятся копии различных фрагментов основной памяти. Хорошим примером работы процессора сразу с несколькими областями памяти является использование многозадачной операционной системы Windows. Здесь четырехстраничный кэш значительно повышает производительность процессора.

Содержимое кэша всегда должно соответствовать содержимому основной памяти, чтобы процессор работал с самыми свежими данными. Поэтому в семействе процессоров 486 используется кэш со *сквозной записью (write-through),* при которой данные, записанные в кэш, автоматически записываются и в основную память.

В процессорах Pentium используется *двунаправленный кэш* (*write-back*), который работает при выполнении как операций считы-

вания, так и операций записи. Это позволяет еще больше повысить производительность процессора.

Одной из функций встроенного кэш-контроллера является отслеживание состояния системной шины при передаче управления шиной другому устройству. Если устройство, управляющее шиной, записывает что-либо в область памяти, копия которой хранится во встроенном кэше, содержимое кэша перестает соответствовать содержимому основной памяти. В этом случае кэш-контроллер отмечает эти данные как ошибочные и при следующем обращении процессора к памяти обновляет содержимое кэша, поддерживая целостность системы.

Внешний кэш (второго уровня) представляет собой быстродействующую статическую память, которая также позволяет сократить время простоя процессора при обращениях к системной памяти. Внешний кэш работает так же, как и встроенный: он хранит информацию, передаваемую в процессор, сокращая потери на ожидание. Время выборки данных из микросхем внешнего кэша обычно не превышает 15 нс, что значительно меньше, чем у основной памяти.

Вторичный кэш для процессоров Pentium находится на системной плате, а для процессоров Pentium Pro и Pentium II — внутри корпуса процессора. Переместив вторичный кэш в процессор, можно заставить его работать с тактовой частотой, более высокой, чем у системной платы, — такой же, как и у самого процессора.

Два уровня кэша между быстрым центральным процессором и значительно более медленной оперативной памятью помогают сократить время ожидания, которое потребовалось бы процессору для считывания и записи данных в оперативную память. Именно благодаря кэш-памяти процессор может работать с быстродействием, приближенным к истинному.

6.4.3. Память типа ROM

Противоположностью запоминающим устройствам с произвольным доступом по функциональному назначению являются устройства для хранения постоянной информации, которая после изготовления ЗУ не изменяется, — *ROM-ЗУ (read-only memory*). Такие ЗУ играют очень важную роль в блоках микропрограммного управления. ROM-ЗУ применяются для хранения библиотек часто используемых функций, системных программ, таблиц функций и т.д.

Модули ROM-ЗУ изготавливаются с помощью интегральной технологии, причем запись информации в устройство является одним из этапов технологического производственного процесса. Это влечет за собой определенные проблемы.

- Стоимость записи данных в ROM-ЗУ зависит от серийности — изготавливается несколько десятков или тысячи экземпляров идентичных микросхем.
- Малейшая ошибка при записи недопустима, поскольку исправить ее уже невозможно. Искажение одного бита в ЗУ емкостью в несколько мегабайтов приводит к тому, что вся микросхема бракуется.

Такая память используется только для чтения данных. ROM также часто называется *энергонезависимой памятью,* потому что любые данные, записанные в нее, сохраняются при выключении питания. Поэтому в ROM помещаются команды запуска персонального компьютера, т.е. программное обеспечение, которое загружает систему. ROM представляет собой часть адресного пространства оперативной памяти.

Основной код BIOS содержится в микросхеме ROM на системной плате, но на платах адаптеров также имеются аналогичные микросхемы. Они содержат вспомогательные подпрограммы базовой си-

стемы ввода-вывода и драйверы, необходимые для конкретной платы, особенно для тех плат, которые должны быть активизированы на раннем этапе начальной загрузки, например видеоадаптер. Платы, не нуждающиеся в драйверах на раннем этапе начальной загрузки, обычно не имеют ROM, потому что их драйверы могут быть загружены с жесткого диска позже — в процессе начальной загрузки.

Существует четыре различных типа микросхем памяти ROM:

- *ROM* (*Read Only Memory*);
- *PROM* (*Programmable ROM*) – программируемая ROM;
- *EPROM* (*Erasable PROM*) – стираемая программируемая ROM;
- *EEPROM* (*Electrically Erasable PROM*) – электронно-стираемая программируемая ROM, также называемая *Flash ROM*.

Если необходимо всего несколько экземпляров постоянных ЗУ с идентичной информацией, экономически более целесообразно применять *программируемые* (*programmable*) постоянные ЗУ — PROM-ЗУ. Устройства этого типа также являются энергонезависимыми, но процесс записи отделен от процесса изготовления микросхемы и выполняется посредством электрических сигналов с помощью специального оборудования вне комплектуемой вычислительной системы. Поэтому запись можно выполнять не на заводе — изготовителе микросхем, а при изготовлении компьютера, в котором устройство с конкретной информацией должно использоваться.

ЗУ, допускающие ее стирание и замену — *перепрограммируемые постоянные* ЗУ (иногда можно встретить термин *полупостоянные ЗУ*). В отличие от RAM-ЗУ такие устройства являются энергонезависимыми, и в то же время позволяют изменять хранящуюся в модуле информацию. Существует три типа таких перепрограм-

мируемых ПЗУ: стираемые ПЗУ, ПЗУ с электрическим стиранием и флэш-ЗУ (*flash memory*).

В стираемых постоянных ЗУ — EPROM-ЗУ (*erasable programmable ROM*) — и чтение, и запись информации выполняется электрическими сигналами, но перед процедурой записи все ячейки модуля должны быть переведены в исходное состояние, соответствующее хранению в них кодов 0. Эта процедура называется *стиранием* ЗУ. Стирание EPROM-ЗУ выполняется оптически — засветкой модуля потоком ультрафиолетового излучения. Стирать один и тот же модуль EPROM-ЗУ можно многократно. Записанная в модуль после этого информация может храниться довольно долго — до нескольких лет (при соблюдении определенных условий эксплуатации и хранения оборудования).

Существуют перепрограммируемые ЗУ с электрическим стиранием — EEPROM-ЗУ (electrically erasable programmable ROM). В таких ЗУ нет необходимости стирать перед записью все содержимое модуля — операции стирания с записью выполняются на уровне адресуемой ячейки (как правило, байта). Операция записи занимает значительно больше времени, чем операция считывания — порядка нескольких сот микросекунд на байт. В EEPROM-ЗУ сочетаются достоинства энергонезависимого хранения информации с возможностью ее произвольно изменять, не извлекая устройство из системы, т.е. посылая сигналы по тем же линиям магистрали (адресным и линиям данных).

Тип перепрограммируемых ПЗУ — так называемые устройства флэш-памяти (*flash memory*) по своим функциям сходен с EEPROM-ЗУ — запись новой информации возможна с помощью чисто электрических сигналов, а по конструктивным характеристикам, в частности плотности упаковки запоминающих элементов в модуле, — с

EPROM-ЗУ. Время стирания информации — несколько секунд, причем стираться может не все содержимое памяти, а выбранные блоки (но все-таки не отдельные байты).

6.5. Физическая память

Объем физической памяти компьютера зависит от типа используемого процессора и архитектуры системной платы. В процессорах 8086 и 8088 с двадцатью линиями адреса объем памяти не превышает 1 Мбайт (1024 Кбайт). Процессоры 286 и 386SX имеют 24 линии адреса и могут адресовать до 16 Мбайт памяти. Процессоры 386DX, 486, Pentium, Pentium MMX и Pentium Pro имеют 32 линии адреса и могут взаимодействовать с памятью объемом до 4 Гбайт. Процессор Pentium II/II1 имеет 36 линий адреса и в состоянии обрабатывать 64 Гбайт.

В большинстве современных компьютеров вместо отдельных микросхем памяти используются модули SIMM (Single In-line Memory Module) или DIMM (Dual In-line Memory Module), представляющие собой небольшие платы, которые устанавливаются в специальные разъемы на системной плате или плате памяти. Расположенные на системной плате и платах памяти микросхемы (SIMM, DIMM, DIP, SIPP) организуются в банки данных. Отдельные микросхемы так припаяны к плате модуля SIMM или DIMM, что выпаять и заменить их практически невозможно. При появлении неисправности приходится заменять весь модуль. По существу, модуль SIMM или DIMM можно считать одной большой микросхемой.

В PC-совместимых компьютерах применяются в основном два типа модулей SIMM: 30-контактные (9 разрядов) и 72-контактные (36 разрядов). Первые из них меньше по размерам. Микросхемы в модулях SIMM могут устанавливаться как на одной, так и на обеих сторо-

нах платы. Использование 30-контактных модулей неэффективно, поскольку для заполнения одного банка памяти новых 64-разрядных систем требуется *восемь* таких модулей. Поэтому в новых системах с процессорами Pentium MMX, Pentium Pro и Pentium И используются 168-контактные модули DIMM (64-разрядные без битов четности или 72-разрядные с битами четности).

Па рис. 6.1 показан 168-контактный модуль DIMM. Контакты нумеруются слева направо и располагаются с обеих сторон модуля.

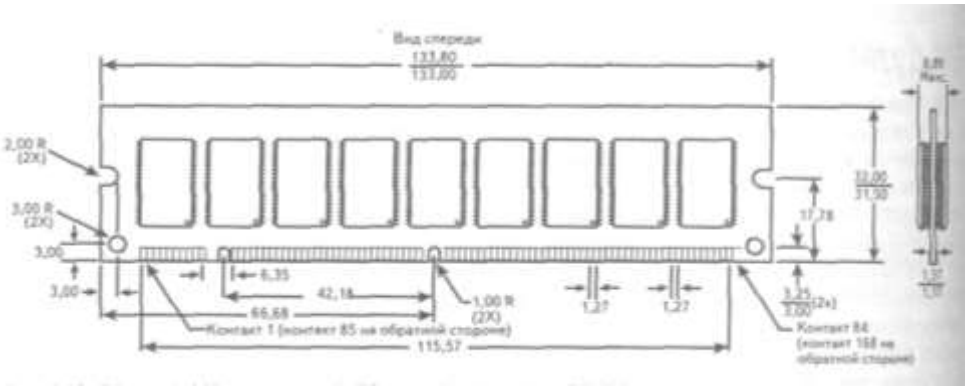


Рис. 6.1. *Обычный* 168-*контактный* (72-*разрядный*) *модуль DIMM*

Модули с разной емкостью могут иметь различное быстродействие. Для микросхем динамической памяти (DRAM), устанавливаемых в модулях SIMM, оно варьируется от 50 до 120 нс. В первых компьютерах использовались модули с временем доступа 120 нс. Довольно быстро они были вытеснены более совершенными модулями (время доступа 100 нс и меньше. Сейчас чаше всего можно встретить модули SIMM с временем доступа 70 и 60 нс. Практически во всех современных системах используются модули DIMM, а в некоторых еще применяются модули SIMM.

6.6. Логическая организация памяти

Типичное распределение адресного пространства представлено в таблице 6.1.

Таблица 6.1. Логическая организация памяти

Объем адресного пространства		Физические адреса	Сегментные адреса	
		00000 h	0000 h	
1 Кбайт	Векторы прерываний	00400h	0040h	Обычная память (640 Кбайт)
256 Кбайт	Область данных BIOS	00500 h	0050 h	
	Операционная система MS-DOS			
	Свободная память для загружаемых прикладных и системных программ	A0000 h	A000 h	
64 Кбайт	Графический видеобуфер	B0000 h	B000 h	Старшая часть (384 Кбайт)
32 Кбайт	Свободные адреса	B8000 h	B800 h	
32 Кбайт	Текстовый видеобуфер	C0000 h	C000 h	
64 Кбайт	ПЗУ-расширения BIOS	D0000 h	D000 h	
128 Кбайт	Свободные адреса	F0000 h	F000 h	
64 Кбайт	ПЗУ BIOS	100000 h		
64 Кбайт	HMA	10FFF0 h		Расширенная память
до 4 Гбайт	XMS			

Выделяют следующие участки (блоки) памяти, используемые в современных компьютерах.

1. Основная память (Conventional Memory).

В первых компьютерах PC/XT рабочее пространство памяти составляло 1 Мбайт и называлось *памятью с произвольным доступом* (*Random-Access Memory— RAM*) или *оперативной памятью.* Это пространство было разделено на несколько областей, часть из которых предназначалась для специальных целей. DOS может обращаться ко всему пространству размером 1 Мбайт, но программы можно загружать только в область памяти, называемую *основной памятью* (*con-*

ventional memory), емкость ее в первом PC была равна 512 Кбайт. Оставшиеся 512 Кбайт были зарезервированы для использования некоторыми компонентами компьютера, такими как системная плата и платы адаптеров, установленных в разъемах расширения.

После выпуска первого PC фирма IBM пришла к выводу, что для обслуживания системы вполне достаточно области размером 384 Кбайт, поэтому в следующих компьютерах объем доступной для использования памяти был увеличен до 640 Кбайт. Эти 640 Кбайт стали стандартным объемом памяти, который DOS может использовать для выполнения программ (*барьер в* 640 *Кбайт*). Память свыше 640 Кбайт зарезервирована для графических плат и других адаптеров, а также для системной ROM BIOS.

Барьер в 640 Кбайт имеет значение только для 16-разрядных программ, таких как DOS и Windows 3.1. Что касается 32-разрядных программ, таких как Windows9x и NT/2000, то на них он не оказывает существенного влияния.

2. Верхняя память (Upper Memory Area — UMA).

Верхняя память (*Upper Memory Area — UMB*) представляет собой 384 Кбайт, зарезервированных у верхней границы системной памяти для компьютеров класса PC/XT и у верхней границы первого мегабайта памяти для компьютеров AT. Адреса этой области находятся в пределах от A000 до FFFFF.

Верхняя память разделена на несколько частей:

- первые 128 Кбайт, расположенные сразу после основной памяти, являются областью *видеопамяти* и предназначены для использования видеоадаптерами. Когда на экран выводится текст или графика, в этой области хранятся образы изображений. Видеопамять занимает адреса A000-BFFFF;

- следующие 128 Кбайт отведены для программ BIOS адаптеров, которые записаны в микросхемах ROM на соответствующих платах, установленных в разъемы расширения. Большинство видеоадаптеров VGA и совместимых с ними адаптеров используют для своих программ BIOS первые 32 Кбайт из этой области, а оставшаяся ее часть доступна для других устройств. Некоторые сетевые адаптеры используют эту область в качестве памяти специального назначения. Для ROM адаптеров и специальной памяти отведены адреса C000-DFFFF;
- оставшиеся 128 Кбайт зарезервированы для системной BIOS, которая записана в микросхемах ROM или RAM. В этой же области хранятся программа POST и начальный системный загрузчик, который управляет компьютером до запуска операционной системы, В большинстве компьютеров используются только последние 64 Кбайт этого пространства (или меньше), а первые 64 Кбайт с помощью программ — диспетчеров памяти могут быть перераспределены для нужд операционной системы. В некоторых системах в этой области также размещена программа Setup CMOS. Для системной BIOS отведены адреса E000-FFFFF.

3. Область верхних адресов (High Memory Area — HMA).

4. Дополнительная память (extended Memory Specification — XMS).

Под дополнительной памятью обычно подразумевают всю память за пределами первого мегабайта, которая становится доступной только при работе процессора в защищенном режиме.

5. Расширенная память (Expanded Memory Specification — EMS) является устаревшей разновидностью.

6. Видеопамять (Video RAM Memory) расположена в области верхней памяти.

Видеоадаптер, установленный в компьютер, использует часть нижней памяти для вывода графики или текстовой информации на дисплей.

7. Область ROM адаптеров и RAM специального назначения расположена в области верхней памяти;

8. ROM BIOS также расположена в области верхней памяти.

7. Внешняя память компьютера

7.1. Устройства магнитного хранения данных

Долгое время основным устройством хранения данных в компьютерном мире были перфокарты. 21 мая 1952 года IBM анонсировала модуль ленточного накопителя IBM 726 для вычислительной машины IBM 701. 13 сентября 1956 года небольшая команда разработчиков все той же фирмы IBM объявила о создании первой дисковой системы хранения данных— 305 RAMAC (Random Access Method of Accounting and Control). Эта система могла хранить 5 млн. символов (5 Мбайт) на 50 дисках диаметром 24 дюйма (около 61 см). В отличие от ленточных устройств хранения данных, в системе RAMAC запись осуществлялась с помощью головки в произвольное место поверхности диска. Такой способ заметно повысил производительность компьютера, поскольку данные записывались и извлекались намного быстрее, чем при использовании ленточных устройств.

Магнитный диск как носитель информации представляет собой пластину круглой формы из немагнитного металла (чаще всего алюминия) или пластика, покрытую слоем магнитного материала. Данные записываются на носитель и считываются с него с помощью магнитной головки, представляющей собой миниатюрный электромагнит. Запись и считывание информации происходит в результате взаимодействия движущегося носителя с неподвижной головкой.

Процесс записи основан на явлении возбуждения магнитного поля при протекании электрического тока через обмотку головки. Импульсы тока подаются в обмотку и перемагничивают материал поверхности носителя в том месте, которое в текущий момент находится в непосредственной близости от головки. Направление магнитного поля, зависящего от полярности электрического импульса в обмотке,

определяет знак локального перемагничивания материала. Считывание информации производится при прохождении под головкой перемагниченного носителя. Магнитный поток, образуемый проходящим под головкой намагниченным участком, частично замыкается через сердечник головки, пронизывая ее обмотку. При прохождении под головкой участков с разной полярностью намагничивания потокосцепление обмотки меняется, и в ней возникают электрические импульсы той или иной полярности, которые в соответствии с принятым методом записи воспринимаются как сигналы «лог. 1» или «лог. 0».

Головка записи/считывания имеет очень маленькие размеры и формируемое ею магнитное поле влияет только на маленький участок поверхности носителя. Поскольку носитель под головкой вращается, это дает возможность организовать на нем данные в виде концентрических окружностей — *дорожек* (или треков — *track*). Ширина дорожки зависит от конструкции головки, как правило на поверхности магнитного диска формируется от 500 до 2000 дорожек.

На рис. 7.1 показана схема распределения отдельных зон на поверхности диска. Соседние дорожки разделены *зазором* (*gap*), который препятствует возникновению ошибок (или, по крайней мере, снижает вероятность их возникновения) вследствие неточного позиционирования головки относительно дорожки или взаимовлияния магнитных полей соседних дорожек. Для упрощения электронных схем одинаковое количество битов информации записывается на всех дорожках, хотя они и отличаются по диаметру, а следовательно, имеют разную длину. Таким образом, *плотность записи* (*density*) информации — количество битов на единицу длины дорожки — уменьшается по мере перехода от внутренних дорожек диска к расположенным на его периферии дорожкам.

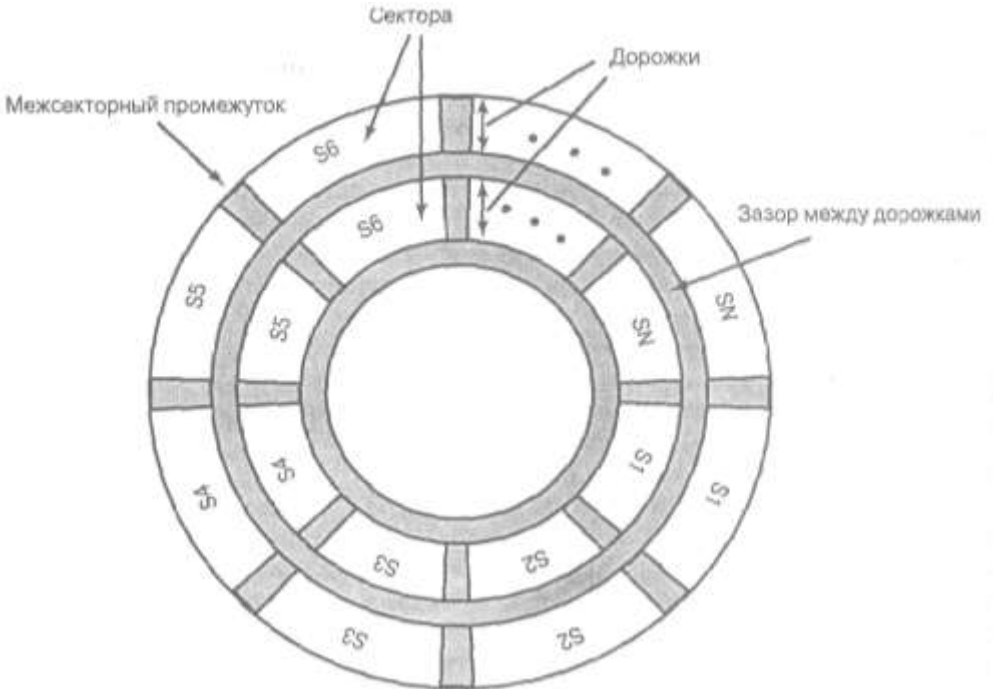


Рис. 7.1. *Разметка магнитного диска*

Основной критерий оценки накопителей на жестких дисках – поверхностная плотность записи. Она определяется как произведение линейной плотности записи вдоль дорожки, выражаемой в битах на дюйм (Bits Per Inch — BPI), и количества дорожек на дюйм (Tracks Per Inch — TPI). В результате поверхностная плотность записи выражается в Мбит/дюйм^2 или Гбит/дюйм^2. В современных накопителях размером 3,5 дюйма величина этого параметра составляет 10-20 Гбит/дюйм^2, а в экспериментальных моделях достигает 40 Гбит/дюйм^2. Это позволяет выпускать накопители емкостью более 400 Гбайт.

Данные записываются на диск и считываются с него *блоками.* Обычно размер блока таков, что на одной дорожке записывается множество блоков. Следовательно, дорожка разбивается на участки, соразмерные длине блока, которые называются *секторами*. Как правило, вдоль одной дорожки размещается от 10 до 100 секторов, имею-

щих либо фиксированную, либо переменную длину. Между секторами на дорожке существует межсекторный промежуток.

Количество секторов может быть разным в зависимости от плотности дорожек и типа накопителя. Например, дорожка гибких дисков может содержать от 8 до 36 секторов, а дорожка жесткого диска — от 380 до 700. Секторы, создаваемые с помощью стандартных программ форматирования, имеют емкость 512 байт. Нумерация секторов на дорожке начинается с единицы, в отличие от головок и цилиндров, отсчет которых ведется с нуля. Например, дискета HD (High Density) формата 3,5 дюйма (емкостью 1,44 Мбайт) содержит 80 цилиндров, пронумерованных от 0 до 79, в дисководе установлены две головки (с номерами 0 и 1), и каждая дорожка цилиндра разбита на 18 секторов (1-18).

Необходимо каким-то образом идентифицировать положение каждого сектора. Для этого, во-первых, на дорожке должна иметься некая начальная отметка, а во-вторых, каким-либо образом должны быть помечены начало и конец каждого сектора. Эти требования реализуются записью на диск служебной информации. Таким образом, на диске помимо основной записана еще и специальная служебная информация, которая нужна только схемам управления устройством и недоступна пользователю.

Физические характеристики устройств внешней памяти на магнитных дисках

В табл. 7.1 перечислены основные характеристики, по которым классифицируются типы *устройств внешней памяти на магнитных дисках* (УВПМД).

Таблица 7.1. Характеристики УВПМД

Характеристика	Набор параметров/Возможное значение
Перемещение головки	с фиксированными головками (одна головка на дорожку); с подвижными головками (од-

Характеристика	Набор параметров/Возможное значение
	на головка на поверхность)
Возможность смены носителя	Несменные (жесткие) диски; сменные диски, в том числе гибкие (флоппи-диски)
Количество рабочих поверхностей	Односторонний диск; двухсторонний диск
Количество дисков в пакете	Однодисковый пакет; многодисковый пакет
Конструкция головки	Контактная головка (для флоппи-дисков); бесконтактная головка с фиксированным зазором; плавающая головка с аэродинамической регулировкой зазора (диски типа "Винчестер")

В накопителях на жестких дисках используются головки четырех типов: ферритовые; тонкопленочные; с металлом в зазоре; магниторезистивные.

7.2. Накопители на жестких дисках

Накопители на жестких дисках обычно называют винчестерами. Этот термин появился в 60-е годы, когда IBM выпустила высокоскоростной накопитель с одним несъемным и одним сменным дисками емкостью по 30 Мбайт. Этот накопитель состоял из пластин, которые вращались с высокой скоростью, и "парящих" над ними головок, а номер его разработки — 30-30. Такое цифровое обозначение (30-30) совпало с обозначением популярного нарезного оружия Winchester, поэтому термин винчестер вскоре стал применяться в отношении любого стационарно закрепленного жесткого диска.

Основными элементами накопителя являются несколько круглых алюминиевых или кристаллических стекловидных пластин. В отличие от гибких дисков (дискет), их нельзя согнуть; отсюда и появилось название жесткий диск.

Основными узлами накопителей на жестких дисках являются:

1. Диски.

Наиболее распространенными являются устройства с дисками следующих диаметров:

- 5,25 дюйма (на самом деле — 130 мм, или 5,12 дюйма);
- 3,5 дюйма (на самом деле — 95 мм, или 3,74 дюйма);
- 2,5 дюйма (на самом деле — 65 мм, или 2,56 дюйма);
- 1 дюйм (на самом деле — 34 мм, или 1,33 дюйма).

В большинстве накопителей устанавливается минимум два диска, хотя в некоторых малых моделях бывает и по одному. Количество дисков ограничивается физическими размерами накопителя, а именно высотой его корпуса.

2. Головки чтения/записи.
3. Механизм привода головок.
4. Двигатель привода дисков.
5. Печатная плата со схемами управления.
6. Кабели и разъемы.
7. Элементы конфигурации (перемычки и переключатели).

Диски, двигатель привода дисков, головки и механизм привода головок обычно размещаются в герметичном корпусе, который называется *HDA* (*Head Disk Assembly — блок головок и дисков*). Обычно этот блок рассматривается как единый узел; его почти никогда не вскрывают. Прочие узлы, не входящие в блок HDA (печатная плата, лицевая панель, элементы конфигурации и монтажные детали) являются съемными.

Независимо от того, какой материал используется в качестве основы диска, он покрывается тонким слоем вещества, способного сохранять остаточную намагниченность после воздействия внешнего магнитного поля. Этот слой называется *рабочим* или *.магнитным,* и именно в нем сохраняется записанная информация. Самыми распространенными являются два типа рабочего слоя:

• оксидный;

• тонкопленочный.

Оксидный слой представляет собой полимерное покрытие с наполнителем из окиси железа. *Тонкопленочный рабочий слой* имеет меньшую толщину, он прочнее, и качество его покрытия гораздо выше.

В накопителях на жестких дисках данные записываются и считываются универсальными головками чтения/записи с поверхности вращающихся магнитных дисков, разбитых на *дорожки* и *секторы* (512 байт каждый).

В накопителях обычно устанавливается несколько дисков, и данные записываются с обеих сторонах каждого из них. В большинстве накопителей есть по меньшей мере два ни три диска (что позволяет выполнять запись на четырех или шести сторонах), но существуют также устройства, содержащие до 11 и более дисков. Однотипные (одинаково расположенные) дорожки на всех сторонах дисков объединяются в *цилиндр*. Для каждой стороны диска предусмотрена своя дорожка чтения/записи, но при этом все головки смонтированы на общем стержне, или *стопке.* Поэтому головки не могут перемещаться независимо друг от друга и двигаются только синхронно.

Жесткие диски вращаются намного быстрее, чем гибкие. Частота их вращения 10000 об/мин. Скорость работы такого или иного жесткого диска зависит от частоты его вращения, скорости перемещения системы головок и количества секторов на дорожке.

При нормальной работе жесткого диска головки чтения/записи не касаются (и не должны касаться) дисков. Но при выключении питания и остановке дисков они опускаются на поверхность. Во время работы устройства между головкой и поверхностью вращающегося диска образуется очень малый воздушный зазор (воздушная подушка). Если в этот зазор попадет пылинка или произойдет сотрясение, головка "столкнется" с диском, вращающимся "на полком ходу". Если

удар будет достаточно сильным, произойдет *поломка головки.* Поэтому в большинстве накопителей поверхности магнитных дисков легируют и покрывают специальными смазками, что позволяет устройствам выдерживать ежедневные «взлеты» и «приземления» головок, а также более серьезные потрясения.

Поскольку пакеты магнитных дисков содержатся в плотно закрытых корпусах и их ремонт не предусмотрен, плотность дорожек на них очень высока — до 30000 и более на дюйм. Блоки HDA *(Head Disk Assembly* — блок головок и дисков) собирают в специальных цехах.

Форматирование дисков

При форматировании диска в начале каждого сектора записывается его *заголовок* (или *префикс — prefix portion*), по которому определяется начало и номер сектора, а в конце — *заключение* (или *суффикс* — (*suffix portion*), в котором находится *контрольная сумма* (*checksum*), необходимая для проверки целостности данных. Помимо указанных областей служебной информации, каждый сектор содержит область данных емкостью 512 байт.

Идентификатор (ID) сектора состоит из полей записи номеров цилиндра, головки и сектора, а также контрольного поля CRC для проверки точности считывания информации ID. В большинстве контроллеров седьмой бит поля номера головки используется для маркировки дефектных секторов в процессе низкоуровневого форматирования или анализа поверхности. Однако такой метод не является стандартным, и в некоторых устройствах дефектные секторы . помечаются иначе. Но как правило, отметка делается в одном из полей ID.

Интервал включения записи следует сразу за байтами CRC; он гарантирует, что информация в следующей области данных будет записана правильно.

В поле данных можно записать 512 байт информации. За ним располагается еще одно поле CRC для проверки правильности записи данных. Записанные в этом поле байты кодов коррекции ошибок позволяют при считывании обнаруживать и исправлять некоторые ошибки.

Интервал между записями необходим для того, чтобы застраховать данные из следующего сектора от случайного стирания при записи в предыдущий сектор. Это может произойти, если при форматировании диск вращался с частотой, несколько меньшей, чем при последующих операциях записи. При этом сектор всякий раз будет немного длиннее, и для того, чтобы он не выходил за установленные при форматировании границы, их слегка "растягивают", вводя упомянутый интервал. Его реальный размер зависит от разности частот вращения диска при форматировании дорожки и при каждом обновлении данных.

Прединдексный интервал необходим для компенсации неравномерности вращения диска вдоль всей дорожки. Размер этого интервала зависит от возможных значений частоты вращения диска и сигнала синхронизации при форматировании и записи.

Послеиндексный интервал нужен для того, чтобы при перемещении головки на новую дорожку переходные процессы (установка) закончились до того, как она окажется перед ее первым сектором. В этом случае его можно начать считывать сразу, не дожидаясь, пока диск совершит дополнительный оборот.

Различают два вида форматирования диска: физическое, или форматирование *низкого уровня* и логическое, или форматирование *высокого уровня.* При форматировании гибких дисков выполняются обе операции, но для жестких дисков эти операции следует выполнять отдельно. Для жесткого диска существует и третий этап, выполняе-

мый между двумя указанными операциями форматирования, — *разбиение диска на разделы.* Создание разделов абсолютно необходимо в том случае, если вы предполагаете использовал на одном компьютере несколько операционных систем. Физическое форматирование всегда выполняется одинаково, независимо от свойств операционной системы и параметров форматирования высокого уровня (которые могут быть различными для разных операционных систем). Это позволяет совмещать несколько операционных систем на одном жестком диске. При организации нескольких разделов на одном накопителе каждый из них может использоваться для работы под управлением своей операционной системы либо представлять отдельный *том* (*volume*), или *логический диск* (*logical drive*). Том, или логический диск, — это то, чему система присваивает буквенное обозначение.

Таким образом, форматирование жесткого диска выполняется в три этапа:

1) форматирование низкого уровня;

2) разбиение диска на разделы;

3) форматирование высокого уровня.

Форматирование низкого уровня

В процессе форматирования низкого уровня дорожки диска разбиваются на секторы. При этом записываются заголовки и заключения секторов (префиксы и суффиксы), а также формируются интервалы между секторами и дорожками. Область данных каждого сектора заполняется фиктивными значениями или специальными тестовыми наборами данных.

При зонной записи цилиндры разбиваются на группы, которые называются *зонами,* причем по мере продвижения к внешнему краю диска дорожки разбиваются на все большее число секторов. Во всех цилиндрах, относящихся к одной зоне, количество секторов на до-

рожках одинаковое. Возможное количество зон зависит от типа накопителя; в большинстве устройств их бывает 10 и более.

При разбиении диска на области, называемые *разделами*, в каждой из них может быть создана файловая система, соответствующая определенной операционной системе. Сегодня в работе операционных систем чаще других используется три файловые системы:

- *FAT* (*File Allocation Table — таблица размещения файлов*). Это стандартная файловая система для DOS, Windows 9*x* и Windows NT. В разделах FAT под DOS допустимая длина имен файлов— 11 символов (8 символов собственно имени и 3 символа расширения), а объем тома (логического диска) — до 2 Гбайт. Под Windows 9*x* и Windows NT 4.0 и выше допустимая длина имен файлов — 255 символов;
- *FAT*32 (*File Allocation Table*, 32-*bit* — 32-*разрядная таблица размещения файлов*). Используется с Windows 95 OSR2 (OEM Service Release 2), Windows 98 и Windows 2000. В таблицах FAT 32 ячейкам размещения соответствуют 32-разрядные числа. При такой файловой структуре объем тома (логического диска) может достигать 2Тбайт (2048 Гбайт);
- *NTFS* (*Windows NT File System — файловая система Windows NT*). Доступна только операционной системе Windows NT/2000. Длина имен файлов может достигать 256 символов, а размер раздела (теоретически) — 16 Эбайт ($16x10^8$ байт). NTFS обеспечивает дополнительные возможности, не предоставляемые другими файловыми системами, например средства безопасности.

При форматировании высокого уровня операционная система (Windows 9x, Windows NT или DOS) создает структуры для работы с файлами и данными. В каждый раздел (логический диск) заносится загрузочный сектор тома (Volume Boot Sector — VBS), две копии таб-

лицы размещения файлов (FAT) и корневой каталог (Root Directory). С помощью этих структур данных операционная система распределяет дисковое пространство, отслеживает расположение файлов и даже "обходит", во избежание проблем, дефектные участки на диске.

7.3. Хранение данных на гибких носителях

Алан Шугарт (Alan Shugart) в конце 60-х годов изобрел накопитель на гибких дисках. В 1983 году фирма Sony впервые представила компьютерному сообществу накопитель и дискету диаметром 3,5 дюйма.

Компоненты дисковода

Дисковод имеет две головки для чтения и записи данных, т.е. является двусторонним. Для каждой стороны диска предназначено по одной головке; обе головки используются для чтения и записи на соответствующих поверхностях диска.

Головки приводятся в движение устройством, которое называется *приводом головок.* Они могут перемещаться по прямой линии и устанавливаться над различными дорожками. Головки двигаются по касательной к дорожкам, которые они записывают на диск. Поскольку верхняя и нижняя головки монтируются на одном держателе (или механизме), они двигаются одновременно и не могут перемещаться независимо друг от друга. Головки представляют собой электромагнитные катушки с сердечниками из мягкого сплава железа. Каждая головка является сложным устройством, в котором головка чтения/записи расположена между двумя стирающими головками в одном физическом устройстве.

Метод записи называется *туннельной подчисткой.* При нанесении дорожек дополнительные головки стирают внешние границы, аккуратно подравнивая их на диске. Эти головки слеш, чтобы данные

находились только в пределах определенного узкого "туннеля" на каждой дорожке. Это препятствует искажению сигнала одной дорожки сигналами с соседних дорожек. *Позиционирование* — это расположение головок относительно дорожек, которые используются ими для чтения и записи.

Головки снабжены пружинами и прижимаются к диску под небольшим давлением. Это означает, что они находятся в непосредственном *контакте* с поверхностью диска во время чтения и записи. Поскольку дисководы для гибких дисков в персональных компьютерах имеют скорость вращения всего 300 или 360 об/мин, это давление не вызывает особых проблем, связанных с трением. Новейшие диски покрываются специальными составами для уменьшения трения и повышения скольжения. В результате контакта между головками и диска головках постепенно образуется налет оксидного материала диска. Этот слой должен периодически счищаться с головок во время профилактического ремонта или обычного обслуживания.

Для того чтобы информация была считана и записана правильно, головки должны находиться в непосредственном контакте с записывающей средой.

Дисководы формата 3,5 дюйма на 1,44 Мбайт высокой плотности (High Density— HD) впервые появились в компьютерах IBM типа PS/2 в 1987 году. Эти дисководы записывают 80 цилиндров из двух дорожек с 18 секторами на дорожке, создавая в результате емкость 1,44 Мбайт. Многие производители дискет указывают на них емкость 2,0 Мбайт (разница между емкостями появляется после форматирования). Общая емкость отформатированного диска не учитывает площади, которая отводится операционной системой для управления файлами, оставляя для хранения файлов только 1423.5 Кбайт.

Эти дисководы имеют скорость вращения 300 об/мин.

Конструкции дискет

В дискетах диаметром 3,5 дюйма гибкий диск находится внутри жесткого футляра, отверстие для доступа головок закрыто металлической заслонкой. Заслонка открывается дисководом. Это защищает поверхность диска от воздействия окружающей среды и прикосновения пальцев. Заслонка также устраняет необходимость в дополнительном чехле для диска (рис. 7.2).

Рис. 7.2. *Конструкция дискеты диаметром 3,5 дюйма*

В дискетах диаметром 3,5 дюйма используется металлическая втулка с установочным отверстием, которая находится в центре дискеты. Дисковод захватывает металлическую втулку, а отверстие в ней позволяет правильно установить дискету.

В нижней левой части дискеты расположено отверстие с пластиковой заслонкой, предназначенное для защиты от записи. Если заслонка расположена так, что отверстие открыто, значит, диск защищен от записи. Когда заслонка закрывает отверстие, запись разрешена. Для более надежной защиты от записи некоторые коммерческие программы поставляются на дискетах без заслонки, поэтому осуществить запись на диск далеко не просто.

На противоположной относительно отверстия защиты от записи стороне дискеты (справа) в футляре может быть еще одно отвер-

стие, которое называется *отверстием для датчика типа дискеты.* Его наличие означает, что диск имеет особое покрытие и является диском высокой или сверхвысокой плотности. Если отверстие для датчика типа дискеты находится точно напротив отверстия зашиты, значит, емкость дискеты 1,44 Мбайт. Если оно смещено к верхней части дискеты (металлическая заслонка в этом случае находится в верхней части дискеты), значит, это дискета сверхвысокой плотности. Отсутствие отверстий на правой половине означает, что дискета имеет низкую плотность. В большинстве дисководов формата 3,5 дюйма имеется датчик типа дискеты, который управляет записью в зависимости от расположения и наличия этих отверстий.

Типы дискет, существующие в настоящее время, описаны в таблице 7.2.

Таблица 7.2. Параметры магнитных покрытий дискет

5,25 дюйма				3,5 дюйма		
Параметр магнитного покрытия	Двойная плотность (DD)	Четверная плотность (QD)	Высокая плотность (HD)	Двойная плотность (DD)	Высокая плотность (HD)	Сверхвысокая плотность (ED)
Плотность дорожек (TPI)	48	96	96	135	135	135
Линейная плотность (BPI)	5 876	5876	9646	8717	17 434	34 868
Основа магнитного слоя	Fe	Fe	Co	Co	Co	Ba
Коэрцитивная сила	300	300	600	600	720	750
Толщина, микродюйм	100	100	50	70	40	100
Полярность записи	Горизонтальная	Горизонтальная	Горизонтальная	Горизонтальная	Горизонтальная	Вертикальная

Параметры, используемые для описания обычной дискеты:

– *плотность записи. Плотность записи* (*density*) — это количество информации, которое может быть надежно размещено на определенной площади записывающей поверхности. Диски имеют два типа плотности — радиальную и линейную. *Радиальная плотность* указывает, сколько дорожек может быть записано на диске, и выражается в количестве дорожек на дюйм (Track Per Inch — TP1). *Линейная плотность* — это способность отдельной дорожки накапливать данные; часто выражается в количестве битов на дюйм (Bits Per Inch— BPI);

– *коэрцитивная сила и толщина магнитного слоя.* Коэрцитивная сила обозначает напряженность магнитного поля, необходимую для правильной записи данных на диск. *Коэрцитивная сила,* как и напряженность магнитного поля, измеряется в эрстедах (Э). Чем тоньше магнитный слой, тем меньше влияет одна область диска на другую — соседнюю.

7.4. Накопители со сменными носителями

Существует три основных категории сменных носителей: дисковые, ленточные и флэш-память.

7.4.1. Магнитооптические накопители

Открытые для коммерческого использования в 1985 году, современные магнитооптические накопители имеют емкость более 5 Гбайт, что практически в 2,5 раза больше, чем самый емкий съемный магнитный накопитель.

Существует два типа магнитооптических накопителей и носителей: 3,5- и 5,25-дюймовые емкостью до 640 Мбайт и 5 Гбайт соответственно. В первых моделях магнитооптических накопителей при-

менялась технология однократной записи, которая позволяла дописывать информацию на носитель, но не позволяла ее стирать.

При нормальных температурах магнитная поверхность магнитооптического диска очень стабильна и может хранить данные около 30 лет. Для изменения данных на магнитооптическом диске используется и лазерный луч, и магнитное поле.

Оптической частью магнитооптического накопителя является лазерный луч, который во время стирания работает в режиме высокой мощности, разогревая необходимый участок магнитооптического диска до температуры порядка 200°C ("точка Кюри", в которой немагнитное в нормальных условиях вещество становится восприимчивым к магнитному полю). Это позволяет стереть любую существующую на разогретом участке информацию с помощью однородного магнитного поля, не задевая другие части диска, которые имеют нормальную температуру.

После этого лазерный луч и магнитное поле используются для записи информации в определенное место за счет увеличения излучаемой мощности лазера и приложения контролируемого магнитного поля к носителю.

При считывании лазер используется в режиме низкой мощности для создания нейтрально поляризованного освещения на поверхности магнитооптического диска. Места на диске, которые содержат логический 0, отражают свет с углом поляризации, отличным от угла поляризации областей, содержащих логическую 1. Эта разница в один градус называется *эффектом Керра.*

7.4.2. Хранение информации на магнитных лентах

Физика процесса записи/считывания информации в устройствах внешней памяти на магнитной ленте (УВПМЛ) ничем не отличается от аналогичных процессов в УВПМД. Просто в качестве носителя используется не вращающийся диск, покрытый магнитным материалом, а скользящая лента. Механизм перемещения ленты в современных УВПМЛ примерно такой же, как в бытовом аудио магнитофоне.

На ленте информация записывается на несколько параллельных дорожек. В современных УВПМД используются 18 или 36 дорожек, что позволяет параллельно записывать двухбайтовое слово или четырехбайтовое двойное слово и побайтовые контрольные коды четности. Информация записывается на ленту блоками фиксированной длины (их принято называть *физической записью).* Блоки на ленте разделены *промежутками.*

УВПМЛ относится к устройствам памяти с *последовательным* доступом. Особенность механики УВПМЛ в том, что носитель перемещается только во время выполнения операций доступа, тогда как в УВПМД носитель вращается постоянно.

УВПМД является устройством памяти с *прямым* доступом к данным, поскольку для доступа к определенному блоку в нем нет необходимости считывать все промежуточные.

Исторически УВПМЛ были первыми устройствами массовой памяти.

7.4.3. Флэш-память

Новейшим типом сменного накопителя является не основанная на принципе магнетизма так называемая флэш-память. Это твердотельный чип памяти, который не требует постоянного питания для хранения информации. Карты флэш-памяти можно легко перемещать

из цифровых камер в портативные либо стационарные компьютеры или даже напрямую подключать к фотопринтерам. Флэш-память можно использовать для хранения любых цифровых данных, однако основная область ее применения — цифровая фотография.

7.5. Устройства оптического хранения данных

В устройствах оптического хранения запись и считывание осуществляются на вращающийся диск с помощью лазерного луча, а не магнитного поля.

7.5.1. CD-ROM

В 1978 году фирмы Sony и Philips объединили свои усилия в области разработки современных звуковых компакт-дисков. *CD-ROM* (*Compact Disc Read-Only Memory — память только для чтения на компакт-диске*) — это оптический носитель информации, предназначенный только для чтения данных. Другие форматы CD-R и CD-RW позволяют записывать данные на компакт-диск, а новая технология DVD позволяет существенно увеличить емкость обычного оптического диска.

CD-ROM — это оптический носитель информации, предназначенный только для чтения, на котором может храниться до 700 Мбайт данных, что соответствует примерно 333 тыс. страниц текста, 74 минутам высококачественного звучания или их комбинации.

Технология записи компакт-дисков

Компакт-диск диаметром 120 мм (около 4,75 дюйма) изготовлен из полимера и покрыт металлической пленкой (обычно каким-нибудь сплавом алюминия). Информация считывается именно с этой металлической пленки, которая покрывается полимером, защищающим данные от повреждения. Этикетка обычно помещается на верх-

ней стороне диска, а считывание выполняется с нижней стороны. Таким образом, компакт-диск является односторонним носителем информации. Поскольку компакт-диск является оптическим устройством, его поверхность должна быть чистой и гладкой.

Считывание информации с диска происходит за счет регистрации изменений интенсивности отраженного от алюминиевого слоя излучения маломощного лазера. Приемник или фотодатчик определяет, отразился ли луч от гладкой поверхности, был он рассеян или поглощен. Рассеивание или поглощение луча происходит в местах, где в процессе записи были нанесены углубления (штрихи). Сильное отражение луча происходит там, где этих углублений нет. Затем эта информация в виде электрических сигналов поступает на микропроцессор, который преобразует эти сигналы в двоичные данные или в звук.

Данные на компакт-дисках записываются с использованием технологии *CLV* (*Constant Linear Velocity* — запись с постоянной линейной скоростью). Это означает, что запись и воспроизведение данных с компакт-диска происходят с постоянной линейной скоростью перемещения дорожки относительно считывающего устройства. Другими словами, при считывании информации с внутренних дорожек диск должен вращаться быстрее, а при считывании с внешних — медленнее. Этот способ применяется потому, что первоначально компакт-диски были предназначены для воспроизведения звука, при котором требовалась постоянная скорость считывания данных.

Алгоритм работы накопителя CD-ROM:

1) *полупроводниковый лазер* генерирует маломощный инфракрасный луч, который попадает на отражающее зеркало;

2) *серводвигатель* по командам, поступающим от встроенного микропроцессора, смешает подвижную каретку с отражающим зеркалом к нужной дорожке на компакт-диске;
3) отраженный от диска луч фокусируется линзой, расположенной под диском, отражается от зеркала и попадает на разделительную призму;
4) *разделительная призма* направляет отраженный луч на другую фокусирующую линзу;
5) эта *линза* направляет отраженный луч на фотодатчик, который преобразует световую энергию в электрические импульсы;
6) сигналы с *фотодатчика* декодируются встроенным микропроцессором и передаются в компьютер в виде данных.

Недостатки компакт-дисков в сравнении с УВПМД:

- компакт-диск пригоден только для чтения информации; изменить что-либо в его содержимом после завершения производственного процесса невозможно;
- время доступа к информации на компакт-диске значительно больше, чем в УВПМД, и составляет порядка половины секунды.

7.5.2. Стираемые оптические диски

Стираемый оптический диск (CD-RW, или иногда называют CD-E (erase CD)) может многократно перезаписываться, как любой магнитный диск. Разработано несколько методов реализации таких функциональных возможностей, но только один из них является чисто оптическим. Этот метод основан на изменении фазы. В таких устройствах используются диски, покрытые материалом, изменяющим отражательные свойства в зависимости от фазового состояния. Когда вещество находится в аморфном состоянии (говорят еще "в аморфной фазе"), его молекулы ориентированы совершенно произвольно, и ве-

щество плохо отражает падающий на него свет. В кристаллическом состоянии вещество имеет практически гладкую поверхность, и его отражательная способность значительно выше. Локальное изменение состояния вещества производится лучом лазера.

Недостатком носителей такого типа является то, что вещество после многократного изменения фаз теряет способность стабильно удерживать фразу, т.е. самопроизвольно может изменить состояние того или иного участка.

7.5.3. Цифровые видеодиски

В сентябре 1995 года создать единый стандарт CD-ROM большой емкости. Новый стандарт был назван DVD и совмещал элементы своих предшественников, т.е. в его лице был получен унифицированный стандарт как для компьютерных технологий, так и для индустрии развлечений. Вначале DVD расшифровывали как *цифровой видеодиск* (*Digital Video Disc*), но позднее переименовали в *цифровой универсальный диск* (*Digital Versatile Disc*).

На один цифровой видеодиск можно записать информации примерно в 7 раз больше, чем на современный компакт-диск. Для хранения изображений в видеодисках используется метод уплотнения информации.

8. Основные внешние устройства компьютера

Обмен информацией между вычислительной системой и внешним миром производится через множество специализированных устройств, которые принято называть *внешними* или *периферийными.* Такое устройство подсоединяется к вычислительной системе (а точнее к модулю ввода-вывода системы) с помощью специализированных линий, по которым циркулируют управляющие сигналы, информация о текущем состоянии устройства и данные.

Внешние устройства можно разделить на три большие группы:

• предназначенные для непосредственной работы с человеком;

• предназначенные для работы с каким-либо оборудованием;

• коммуникационные, предназначенные для связи с удаленными абонентами.

Взаимодействие устройства с модулем ввода-вывода осуществляется сигналами трех типов:

• *управляющие сигналы,* которые передаются от модуля ввода-вывода на внутренние схемы управления устройством;

• *информация о текущем состоянии,* которая передается внутренними схемами управления устройством на модуль ввода-вывода;

• *данные,* которые в. зависимости от специфики устройства могут поступать от модуля ввода-вывода или от устройства, или циркулировать в обоих направлениях, как это происходит в устройствах внешней памяти.

Внутренняя схема управления устройством (локальный контроллер) управляет отработкой управляющих сигналов, поступающих от модуля ввода-вывода. Преобразователь служит для преобразования двоичных кодов данных в форму, специфичную для данного типа устройства (или наоборот, из формы, специфичной для устройства, в

двоичный код). Это может быть преобразование в аналоговый электрический сигнал, в другой вид энергии, например в механическое перемещение, в дискретный сигнал управления заданным исполнительным механизмом из множества подключенных и т.д. Как правило в устройстве между линиями данных и преобразователем имеется буфер небольшого объема (8-16 бит) для временного хранения поступившей/передаваемой информации.

8.1. Видеоадаптеры и мониторы

Система отображения компьютера состоит из двух главных компонентов:

• монитора (дисплея);

• видеоадаптера (называемого также *видеоплатой* или *графической платой)*.

Работа электронно-лучевого монитора

Информация на мониторе может отображаться несколькими способами. Самый распространенный — отображение на экране *электронно-лучевой трубки* (ЭЛТ), такой же, как в телевизоре. ЭЛТ представляет собой электронный вакуумный прибор в стеклянной колбе, в горловине которого находится электронная пушка, а на дне — экран, покрытый люминофором.

Нагреваясь, электронная пушка испускает поток электронов, которые с большой скоростью движутся к экрану. Поток электронов (электронный луч) проходит через фокусирующую и отклоняющую катушки, которые направляют его в определенную точку покрытого люминофором экрана. Под воздействием ударов электронов люминофор излучает свет, который видит пользователь, сидящий перед экраном компьютера.

Химическое вещество, используемое в качестве люминофора, характеризуется *временем послесвечения,* которое отображает длительность свечения люминофора после воздействия электронного пучка. *Время послесвечения* и *частота обновления* изображения должны *соответствовать* друг другу, чтобы не было заметно мерцание изображения (если время послесвечения очень мало) и отсутствовала размытость и удвоение контуров в результате наложения последовательных кадров (если время послесвечения слишком велико).

Электронный луч движется очень быстро, прочерчивая экран строками слева направо и сверху вниз по траектории, которая получила наименование *растр.* Период сканирования по горизонтали определяется скоростью перемещения луча поперек экрана.

В *процессе развертки* (перемещения по экрану) луч воздействует на те элементарные участки люминофорного покрытия экрана, в которых должно появиться изображение. Интенсивность луча постоянно меняется, в результате чего изменяется яркость свечения соответствующих участков экрана. Поскольку свечение исчезает очень быстро, электронный луч должен вновь и вновь пробегать по экрану, возобновляя его. Этот процесс называется *возобновлением* (или *регенерацией)* изображения.

В большинстве мониторов *частота регенерации,* которую также называют *частотой вертикальной развертки,* во многих режимах приблизительно равна 85 Гц, т.е. изображение на экране обновляется 85 раз в секунду. Снижение частоты регенерации приводит к мерцанию изображения, которое очень утомляет глаза. Следовательно, чем выше частота регенерации, тем комфортнее себя чувствует пользователь.

Очень важно, чтобы частота регенерации, которую может обеспечить монитор, соответствовала частоте, на которую настроен

видеоадаптер. Если такого соответствия нет, изображение на экране вообще не появится, а монитор может выйти из строя.

Многочастотные мониторы

В одних мониторах установлена фиксированная частота развертки. В других поддерживаются разные частоты в некотором диапазоне (такие мониторы называются *многочастотными — multiple-frequency monitor).* Большинство современных мониторов многочастотные, т.е. могут работать с разными стандартами видеосигнала

Тип экрана монитора

Экраны мониторов могут быть двух типов: выпуклые и плоские. Экран типичного дисплея выпуклый. Такая конструкция характерна для большинства ЭЛТ. Обычно экран искривлен как по вертикали, так и по горизонтали.

Цифровые сигналы для электронно-лучевых мониторов

Это использование цифрового входа в соответствии со стандартом DVI (Digital Video Interface), применяемым в плоскопанельных дисплеях. Преимущества: более точная передача цветового спектра, общее улучшение качества изображения, точная автонастройка и др.

Мониторы с электронно-лучевыми трубками

Монитор необходимо подключать к источнику входной информации. Сигналы, которые управляют работой монитора, поступают от электронных схем, размешенных внутри компьютера. В некоторых компьютерах (чаще всего с платами LPX или NLX) эти схемы располагаются на системной плате. Однако в большинстве систем (с платами Baby-AT и ATX) используются отдельные платы, которые вставляются в разъемы системной платы. Такие платы расширения, вырабатывающие сигналы управления отображением, называются *видеоадаптерами* или *видеоплатами.*

Жидкокристаллические дисплеи

Жидкокристаллические дисплеи называются также *LCD-дисплеями* (*Liquid-Crystal Display*). Для них характерен безбликовый плоский экран и низкая потребляемая мощность (некоторые модели таких дисплеев потребляют 5 Вт, в то время как мониторы с электронно-лучевой трубкой — порядка 100 Вт). По качеству цветопередачи жидкокристаллические панели с *активной матрицей* в настоящее время превосходят большинство моделей мониторов с электронно-лучевой трубкой.

В жидкокристаллическом экране поляризационный светофильтр создает две раздельные световые волны и пропускает только ту, у которой плоскость поляризации параллельна его оси. Располагая в жидкокристаллическом мониторе второй светофильтр так, чтобы его ось была перпендикулярна оси первого, можно полностью предотвратить прохождение света (экран будет темным). Вращая ось поляризации второго фильтра, т.е. изменяя угол между осями светофильтров, можно изменить количество пропускаемой световой энергии, а значит, и яркость экрана. В цветном жидкокристаллическом экране есть еще один дополнительный светофильтр, который имеет три ячейки на каждый пиксель изображения — по одной для отображения красной, зеленой и синей точек.

Световая волна проходит через жидкокристаллическую ячейку, причем каждый цвет имеет свою ячейку. Жидкие кристаллы представляют собой стержнеобразные молекулы, свойства которых подобны жидкости. Это вещество свободно пропускает свет, плоскость поляризации которого параллельна оптической оси, но под воздействием электрического заряда молекулы изменяют свою ориентацию. Одно-

временно меняется ориентация плоскости поляризации проходящей через нее световой волны.

В жидкокристаллических мониторах с пассивной матрицей яркостью каждой ячейки управляет электрический заряд (точнее, напряжение), протекающий через транзисторы, номера которых равны номерам строки и столбца данной ячейки в матрице экрана. Количество транзисторов (по строкам и столбцам) и определяет разрешение экрана. Например, экран с разрешением 800x600 содержит 800 транзисторов по горизонтали и 600 по вертикали.

На ячейки жидкокристаллического монитора с пассивной матрицей подается пульсирующее напряжение, поэтому они уступают по яркости изображения жидкокристаллическим мониторам с активной матрицей, в каждую ячейку которых подается постоянное напряжение. Для повышения яркости изображения в некоторых конструкциях используется метод управления, получивший название *двойное сканирование,* и соответствующие ему устройства — *жидкокристаллические мониторы с двойным сканированием* (*double-scan LCD*). Экран развивается на две половины (верхнюю и нижнюю), которые работают независимо, что приводит к сокращению интервала между импульсами, поступающими на ячейку.

В жидкокристаллических мониторах с активной матрицей каждой ячейкой управляет отдельный транзисторный ключ. Например, дисплей с активной матрицей 1024x768 содержит 786432 транзисторов. Это обеспечивает более высокую яркость изображения, чем в жидкокристаллических мониторах с пассивной матрицей, поскольку каждая ячейка оказывается под воздействием постоянного, а не импульсного электрического поля.

В цветных жидкокристаллических мониторах на один пиксель приходится три ячейки, и, управляя их яркостью, можно добиться различного цвета изображения на экране.

Размер экрана

Размеры экранов мониторов могут колебаться от 9 до 42 дюймов (или от 23 до 106 см) по диагонали. Чем больше размер экрана, тем дороже монитор. Самыми распространенными являются мониторы с экранами, у которых длина диагонали равна 15, 17, 19 или 21 дюйм. Это не действительный размер активной области экрана, а диагональ электроннолучевой трубки.

Разрешающая способность

Это размер минимальной детали изображения, которую можно различить на экране. Данный параметр характеризуется количеством элементов разложения — *пикселей* (*pixel*) — по горизонтали и вертикали экрана. Чем больше количество пикселей, тем более детальное изображение формируется на экране.

Видеоадаптеры компьютеров поддерживают несколько стандартных разрешений, приведенных ниже вместе с общепринятыми наименованиями режимов.

Разрешение, пиксели	Наименование режима
640x480	VGA (Video Graphics Array)
800x600	SVGA (Super VGA)
1 024x768	XGA (eXtended Graphics Array)
1 280x1024	UVGA (Ultra VGA)

Ниже приведены минимальные размеры экрана, рекомендуемые для самых распространенных форматов изображения.

Разрешение, пиксели	Размер монитора, дюймы
640x480	13

Разрешение, пиксели	Размер монитора, дюймы
800x600	15
1024x768	17
1280x1024	21

Большинство 15-дюймовых мониторов может выводить изображение с разрешением 1024x768 пикселей, но символы, пиктограммы и вся картинка будут меньше. Если работать с мелкими изображениями с разрешением 1024x768 пикселей, то нужен 17-дюймовый монитор.

Размер экрана, указанный в документации жидкокристаллических мониторов, соответствует действительному размеру изображения. Так что жидкокристаллический монитор размером 12,1 дюйма на самом деле формирует изображение размером 12,1 дюйма по диагонали, что сравнимо с обычным 14-дюймовым монитором с электронно-лучевой трубкой. Жидкокристаллический монитор обеспечивает такую четкость, которая позволяет работать с более высокими разрешениями, чем на мониторах с электронно-лучевой трубкой сравнимых размеров.

Видеоадаптеры

Видеоадаптер формирует сигналы управления монитором. Большинство видеоадаптеров поддерживают по крайней мере один из следующих стандартов:

• MDA (Monochrome Display Adapter);

• HGC (Hercules Graphics Card);

• CGA (Color Graphics Adapter);

• EGA (Enhanced Graphics Adapter);

• VGA (Video Graphics Array);

• SVGA (Super VGA);

• XGA (extended Graphics Array).

В настоящее время на рынке можно найти адаптеры VGA, SVGA и XGA. Остальные типы видеоадаптеров уже можно считать реликтами компьютерной эры.

Адаптеры и мониторы стандарта VGA

В VGA используется передача аналоговых сигналов. Аналоговый монитор работает по тому же принципу, что и цифровой, т.е. передаются RGB-сигналы (red — красный, green — зеленый и blue — синий) управления тремя основными цветами, но каждый сигнал имеет несколько уровней яркости (в стандарте VGA— 64). В результате число возможных комбинаций (цветов) возрастает до 262144 (64^3).

Вся аппаратура VGA обеспечивает отображение до 256 оттенков на экране из палитры в 262144 цвета (256 Кбайт).

В настоящее время основным адаптером VGA считается адаптер с 16-ю цветами и разрешением 640x480.

Адаптеры SVGA

С появлением видеоадаптеров XGA конкуренты IBM решили не копировать эти расширения VGA, а начать выпуск более дешевых адаптеров с разрешением, которое выше разрешения продуктов IBM. Эти видеоадаптеры образовали категорию *Super VGA*, или *SVGA* Поначалу SVGA не являлся стандартом. Под этим термином подразумевались многочисленные и отличающиеся одна от другой разработки различных фирм, требования к параметрам которых были жестче, чем требования к VGA.

Внешне платы SVGA мало чем отличаются от своих собратьев VGA. На них установлены такие же разъемы.

В октябре 1989 года ассоциация VESA, учитывая сложность программирования множества выпускаемых модификаций плат SVGA, предложила стандарт для единого программного интерфейса с этими платами. В эту ассоциацию вошли представители большинства

компаний, выпускающих аппаратуру для ПК, в том числе и аппаратуру отображения.

Новый стандарт был назван *VESA BIOS Extension.*

Компоненты видеосистемы

Для работы видеоадаптера необходимы следующие основные компоненты.

1. BIOS (Basic Input/Output System — базовая система ввода-вывода).

Видеоадаптеры имеют свою BIOS, которая подобна системной BIOS, но полностью независима от нее.

2. Графический процессор, иногда называемый набором микросхем системной логики.

Существует промежуточный вариант архитектуры — *видеоакселератор* (*accelerator chip*) с ограниченным набором функций. Такая архитектура, применяемая во многих видеоадаптерах, представленных на современном компьютерном рынке, предполагает, что электронные схемы видеоадаптера решают алгоритмически простые, но отнимающие много времени задачи.

3. Видеопамять.

4. Цифроаналоговый преобразователь, он же DAC — Digital to Analog Converter.

5. Разъем

6. В идеодрайвер.

Программный драйвер — существенный элемент видеосистемы, с помощью которого осуществляется связь программного обеспечения с видеоадаптером.

Видеодрайверы используются для поддержки процессора видеоадаптера. Несмотря на то, что видеоадаптеры поставляются изготовителем вместе с драйверами, иногда используются драйверы, поставляемые вместе с набором микросхем системной логики. Ви-

деодрайвер обеспечивает интерфейс, который используется для настройки методов управления дисплеем, применяемых адаптером.

Многие популярные адаптеры сегодня имеют дополнительные модули, предназначенные для специальных целей, например для ускорения отображения трехмерных объектов.

8.2. Принтеры и сканеры

На сегодняшний день существует следующие три основные технологии печати.

• *Лазерная.* Лазерный принтер работает следующим образом: на фоточувствительноном барабане с помощью луча лазера создается электростатическое изображение страницы. Помещенный на барабан специально окрашенный порошок, называемый *тонером*, "прилипает" только к той области, которая представляет собой буквы или изображение на странице. Барабан поворачивается и прижимается к листу бумаги, перенося на нее тонер. После закрепления тонера на бумаге получается готовое изображение. Подобная технология используется в копировальных аппаратах.

• *Струйно-чернильная.* В струйных принтерах ионизированные капельки чернил через сопла распыляются на бумагу. Распыление происходит в тех местах, где необходимо сформировать буквы или изображения. Процессы интерпретации данных при струйной и лазерной печати в основном подобны. Различие состоит лишь в том, что струйные принтеры имеют меньший объем памяти и менее мощную вычислительную систему. Именно это различие помещает их в разряд устройств более низкого класса и стоимости. Уменьшение установленной в принтере памяти приводит к тому, что в таких принтерах используются буферы полосы вместо буферов целой страницы.

Жидкие чернила распыляются непосредственно на бумагу — в те места, где в лазерном принтере формируется массив из точек.

В настоящее время существует два основных типа струйной печати: термическая и пьезоэлектрическая. При термической струйной печати чернила в картридже нагреваются до температуры 400°С. При этом они закипают и образуется чернильный пар. Давление в резервуаре возрастает, и через сопла чернила небольшими каплями распыляются на бумагу.

В пьезоэлектрической струйной печати вместо нагревания в этих принтерах используется электрический заряд пьезоэлектрических кристаллов внутри отверстий в картридже. Эти кристаллы изменяют свою форму в результате электрического воздействия, проталкивая чернила сквозь отверстия.

• *Матрица точек.* В матричных принтерах используется группа круглых игл, которые ударяют по листу бумаги через красящую ленту. Эти иглы собраны в прямоугольную сетку, называемую *матрицей.* При нажатии определенных игл в матрице формируются различные символы или изображения. Матричные принтеры в работе слишком "шумные", печатают с низким качеством и часто заминают бумагу. Матричные принтеры, в отличие от лазерных и струйных, не формируют страницу документа. Они работают в основном с потоком ASCII-символов и, следовательно, не требуют большого объема памяти. Скорость работы матричных принтеров измеряется в символах в секунду, а не в страницах в МИНУТУ.

Поток данных, исходящих из компьютера, содержит последовательности escape-символов и используется для установки основных параметров принтера, таких как размер страницы и качество печати. Все сложные процессы формирования управляющих кодов принтера выполняются на компьютере.

В матричном принтере бумага помещается в вертикальный лоток и перемещается построчно с помощью валиков. Печатающая головка перемещается горизонтально по специальной направляющей и содержит матрицу из металлических игл (чаще всего состоящую из 9 или 24 игл), которые выдавливают изображение на бумаге. Между иглами и бумагой расположена красящая лента, как на печатной машинке. Иглы (через ленту) создают на бумаге ряд небольших точек, формируя таким образом изображение. При печати графических изображений на матричных принтерах невозможно достичь высокого качества.

Практически на всех матричных принтерах при печати можно использовать как отдельные листы, так и рулоны бумаги. В настоящее время матричный принтер в "классическом" офисе встречается довольно редко — его заменил струйный или лазерный принтер. Единственное место, где матричные принтеры еще не сдали позиций, — это банки и сфера торговли.

Цветная печать

В цветных струйных принтерах используется два картриджа: один содержит черные чернила, другой — три остальных цвета. Преимуществом такого конструктивного исполнения является упрощенная процедура замены картриджа, а недостатком — увеличение стоимости цветного картриджа (в случае, если один цвет закончился раньше остальных). Возможности струйных принтеров соответствуют их цене.

Цветные лазерные принтеры – относительно "молодые" устройства; они работают по принципу, описанному выше. Для данного типа печати применяется та же технология, что и в монохромных моделях, только вместо одноцветного тонера используется четырехцветный. Механизм подачи бумаги в цветном лазерном принтере

весьма сложен. Цветная лазерная печать — самая качественная технология печати.

Разрешение

Термин *разрешение* (*resolution*) используется для описания контрастности и качества напечатанного образца. Во всех рассматриваемых технологиях печати изображение создается путем воспроизведения точек на бумаге. Разрешение принтера, а, следовательно, и качество печати зависит от размера и количества этих точек.

Разрешение принтера обычно измеряют в *точках на дюйм* (*dots per inch — dpi*), это количество отдельных точек, которые может напечатать принтер на линии длиной в один дюйм. В большинстве принтеров разрешение определяется двумя направлениями — вертикальным и горизонтальным. Таким образом, под разрешением 300 dpi подразумевается 300x300 точек на одном квадратном дюйме. Принтер с разрешением 300 dpi может напечатать 90 тыс. точек на квадратном дюйме бумаги. Существуют принтеры, у которых разрешения в двух направлениях различаются (например, 600x1200dpi). Такой принтер может напечатать 720 тыс. точек на одном квадратном дюйме.

Современные лазерные принтеры имеют минимальное разрешение 600 dpi, а в моделях высшего класса оно может достигать 1200 dpi. Качество бумаги влияет на напечатанное изображение.

Качество печати матричных принтеров

Разрешение матричного принтера определяется количеством игл (чаще всего 9 или 24) и является постоянной величиной.

Язык описания страниц

Лазерные и струйные принтеры называют страничными, поскольку они формируют образ целой страницы в памяти перед перемещением его на бумагу. Это основное отличие лазерных и струйных принтеров от матричных, которые являются сим вольными. Для "об-

щения" компьютера со страничным принтером применяется специализированный язык *описания страницы (page description language — PDL).* Это средство кодирования каждой части печатаемого документа в поток данных, который может быть передан на принтер. После получения принтером кодов языка описания страницы встроенное программное обеспечение принтера преобразует код в шаблон точек, которые переносятся на бумагу. В настоящее время существует два языка описания страниц, ставших фактическим стандартом в компьютерной индустрии, — PCL и PostScript.

Принтеры, не поддерживающие язык описания страниц, используют *последовательность escape-кодов* (*escape code sequence*) для управления свойствами принтера в комбинации со стандартным текстом ASCII для передачи содержимого документа. За это "отвечает" драйвер принтера, который распознает передаваемые символы — escape-коды или язык описания страниц. При печати документа неважно, в какой программе он был создан и в каком формате файла был сохранен; данные для печати должны быть преобразованы в поток данных языка описания страниц или поток ASCII-текста с escape-кодами.

PCL (*Printer Control Language*)

Язык описания страниц PCL разработан фирмой Hewlett-Packard в начале 80-х годов для использования в принтерах собственного производства. Поток данных языка PCL содержит множество команд, разработанных для управления принтером.

PostScript

Этот язык описания страниц разработан фирмой Adobe и впервые использован в принтере Apple LaserWriter в 1985 году.

Язык PostScript не поддерживает последовательность escape-кодов, как PCL. PostScript называют объектно-ориентированным язы-

ком программирования, поскольку на принтер отправляется не изображение, а геометрические объекты. Для того чтобы напечатать текст определенным шрифтом, драйвер принтера должен указать последнему контур шрифта и его размер. Контур шрифта служит шаблоном для создания символов любого размера. Принтер генерирует изображение символа из его контура, а не загружает из памяти. Этот тип изображения, который генерируется индивидуально для каждой страницы, называется *векторной графикой,* в отличие от *растровой графики,* которая отправляется на принтер в виде готового набора точек.

Память принтера

В каждом принтере есть микросхемы памяти, а лазерные и струйные принтеры, помимо этого, имеют еще и встроенный процессор, поэтому можно сказать, что принтер — это специализированный компьютер. Память в принтере служит буфером для помещения данных задания печати; она предназначена для хранения данных в процессе создания изображения, шрифтов и команд, а также для временного хранения контуров шрифтов и других данных.

Принтер должен создать растровое изображение всей страницы (графические изображения и шрифты) в памяти, а затем напечатать ее. Каждое векторное изображение и контуры шрифтов перед печатью должны быть преобразованы в растр. Большое количество графики и шрифтов на странице требует много памяти. Кроме того, память необходима для хранения интерпретатора языка описания страниц и шрифтов принтера.

Шрифты

При печати документов используются шрифты. С помощью различных шрифтов можно придать документу законченный вид. Под термином *шрифт* понимают гарнитуру определенного стиля и размера. *Гарнитура* — это разработанный набор буквенно-цифровых сим-

волов, в котором буквы, цифры и символы имеют одинаковое начертание. В настоящее время созданы тысячи гарнитур и постоянно появляются новые. Некоторые из основных гарнитур включены в поставку операционной системы Windows: Times New Roman, Arial, Courier и др. *Стиль шрифта* — это вариации гарнитуры, например полужирный или курсив. Гарнитура может иметь несколько стилей.

Для печати гарнитуры разного размера необходимо отдельное графическое изображение символа для каждого размера. Такие шрифты называют *растровыми* (*bitmap*).

В настоящее время практически во всех принтерах используются *масштабируемые* (*scalable*) шрифты. В этой технологии для печати символа любого размера используется только его контур. Принтер помещает этот контур в память, а затем генерирует растровое изображение символа необходимого размера. Поскольку масштабируемые шрифты представляют собой векторную графику, они могут использовать максимально возможное разрешение принтера, в то время как растровые шрифты всегда имеют одно разрешение.

Масштабируемые шрифты не нужно постоянно хранить в памяти принтера. Драйвер принтера может при необходимости автоматически загрузить контуры того или иного шрифта и сгенерировать любой его размер. Появление технологии TrueType в операционных системах Windows и Macintosh послужило толчком к созданию тысячи гарнитур различных стилей и практически любого размера.

Существует еще одна распространенная технология масштабируемых шрифтов — TrueType. Появившаяся через шесть лет после создания PostScript, эта технология представляет собой результат совместной работы фирм Apple и Microsoft.

Основное преимущество технологии TrueType состоит в том, что она уже встроена в операционные системы Windows и Macintosh;

кроме того, для работы со шрифтами не нужно приобретать дополнительное программное обеспечение.

Драйверы принтеров

Для нормальной работы принтера необходимо установить драйвер. Драйвер принтера обеспечивает интерфейс между принтером и приложением или операционной системой. Основная функция драйвера— информировать компьютер о возможностях принтера (например, об используемом языке описания страниц, поддерживаемых типах бумаги и установленных шрифтах).

8.3. Сканеры

Сканеры преобразуют документ или изображение в цифровую форму. Сканеры подобны устройствам копирования, только вместо печати копии сканер передает оцифрованные данные в компьютер. Сканеры можно разделить на несколько групп: по типу интерфейса, способу формирования сигнала и типу сканируемых документов. После сканирования документа с помощью специальных программ данные передаются в компьютер для обработки, т.е. сканированное изображение можно сохранить в виде файла.

Настольные сканеры

В этих моделях луч проходит более длинный путь после и даже до сканирования, поскольку для сканирования цветных изображений он проходит через светофильтры для разложения на красную, зеленую и голубую составляющие.

Луч света падает на оригинал, отражается от него и через систему зеркал попадает на светочувствительные диоды, где преобразуется в электрический сигнал. Этот сигнал поступает на аналого-цифровой преобразователь, где конвертируется в сигнал, представляющий собой пиксели оригинала (черные, белые, оттенки серого или

цветные). Эта цифровая информация передается в компьютер для дальнейшей обработки.

Преимущества настольных сканеров:

• *возможность сканировать практически любой оригинал.* Настольные сканеры, как и копировальные аппараты, могут сканировать оригиналы различного размера— от миниатюр до документов широко используемых форматов, а также книг. При установке дополнительного модуля появляется возможность сканирования прозрачных пленок, негативов и слайдов. Большинство этих модулей предназначено для сканирования слайдов шириной 35 мм;

• *высокое разрешение.* В настольных сканерах всегда используется два типа разрешения — оптическое и интерполированное. Оптическое разрешение описывает возможности аппаратной (оптической) части сканера. Для увеличения четкости деталей оригинала применяются специальные программные алгоритмы, которые обеспечивает драйвер сканера. Это второе разрешение называется *интерполированным.* Обычно оно увеличивает максимальное разрешение сканера до 4х. Например, оптическое разрешение сканера 600 dpi, а максимальное интерполированное — 2 400 dpi. Поскольку это интерполированное разрешение обеспечивается программными методами, при его использовании качество сканированного оригинала может быть неудовлетворительным. Но практически все модели сканеров обеспечивают приемлемое качество при интерполированном разрешении. Нельзя применять интерполяцию при сканировании слайдов 35 мм.

Недостатки настольных сканеров:

• *большие размеры.* Настольный сканер формата А4 имеет размеры как минимум 210х297 мм и занимает значительную часть рабочего пространства;

• *ограничения на прозрачные оригиналы.* Практически все настольные сканеры среднего и высшего уровня комплектуются модулем для сканирования прозрачных пленок или слайдов. Однако приемлемое качество достигается только при сканировании оригиналов больших размеров. Не всегда качество сканирования фотонегативов 35 мм сможет вас удовлетворить.

8.4. Клавиатура

Клавиатура — одно из важнейших устройств компьютера, используемое для ввода в систему команд и данных. С появлением Windows 95 была создана модифицированная версия 101-клавишной клавиатуры, получившая название 104-клавишной расширенной клавиатуры Windows.

Конструкции клавиш

В современных клавиатурах используется несколько типов клавиш. В большинстве клавиатур установлены механические переключатели, в которых происходит замыкание электрических контактов при нажатии клавиш. В некоторых клавиатурах высокого класса используются бесконтактные емкостные датчики.

Наиболее широко распространены контактные клавиатуры. Существуют следующие их разновидности:

• с механическими переключателями;

• с замыкающими накладками;

• с резиновыми колпачками;

• мембранные.

Интерфейс клавиатуры

Клавиатура состоит из набора переключателей, объединенных в *матрицу.* При нажатии клавиши процессор, установленный в самой клавиатуре, определяет координаты нажатой клавиши в матрице.

Процессор клавиатуры определяет продолжительность нажатия и может обработать даже одновременное нажатие нескольких клавиш. В клавиатуре установлен буфер емкостью 16 байт, в который заносятся данные при слишком быстрых или одновременных нажатиях. Затем эти данные в соответствующей последовательности передаются в систему.

Обычно при нажатии клавиш возникает дребезжание, т.е. контакт устанавливается не сразу, а после нескольких кратковременных замыканий и размыканий. Процессор, установленный в клавиатуре, должен подавлять это дребезжание и отличать его от двух последовательных нажатий на одну и ту же клавишу. Сделать это довольно просто, поскольку переключение контактов при дребезжании происходит гораздо быстрее, чем при нажатии клавиш пользователем.

Клавиатура в PC-совместимой системе представляет собой небольшой компьютер. Связь с системным блоком осуществляется через последовательный канал, данные по которому передаются по 11 бит, причем восемь из них собственно данные, а остальные — синхронизирующие и управляющие.

Номера клавиш и скан-коды

При нажатии клавиши встроенный в клавиатуру процессор определяет координаты замкнутого переключателя в матрице. После этого он передает на системную плату последовательный пакет данных, содержащий скан-код нажатой клавиши. В компьютере контроллер клавиатуры преобразует текущий скан-код в один из предусмотренных в системе скан-кодов и направляет его в главный процессор компьютера. Когда клавиша «залипает» или выходит из строя, диагностическая программа, например, процедура самоконтроля POST, сообщает ее код. IBM назначает каждой клавише уникальный номер, чтобы отличить ее от остальных.

После того как контроллер клавиатуры в системе получит скан-коды, сгенерированные клавиатурой, и передаст их на главный процессор, операционная система преобразует коды в соответствующие алфавитно-цифровые символы. Ими, например, являются буквы, цифры и другие символы, находящиеся на стандартной американской клавиатуре.

Разъемы для подключения клавиатуры и мыши

В новых клавиатурах фирмы IBM используется кабель с разъемными соединениями на обоих концах, что упрощает его замену. На втором конце кабеля, предназначенного для подсоединения к клавиатуре, есть специальный разъем SDL (Shielded Data Link — экранированная линия связи), а на другом конце — разъем DIN (Douche Industrial Norm — промышленный стандарт Германии). Первый из них напоминает телефонный разъем, а второй может быть двух видов: 5-контактный и 6-контактный mini-DIN.

Для подключения мыши к системной плате устанавливается 6-контактный разъем mini-DIN, расположение и назначение выводов которого такое же, как и у разъема клавиатуры, но структура передаваемых данных другая. Это означает, что можно нечаянно подключить системную мышь (например, PS/2) к разъему mini-DIN, предназначенному для клавиатуры и наоборот. В этом случае ни одно из устройств работать не будет.

В последнее время более популярны клавиатуры, а также устройства указания, подключаемые к компьютеру с помощью универсальной последовательной шины USB, вместо стандартных портов клавиатуры и мыши.

8.5. Мышь

В 1964 году Дуглас Энгельбарт (Douglas Englebart), работав-

ший в Stanford Research Institute (SRI), изобрел мышь. Официально она была названа *указателем XY-координат для дисплея.*

Поначалу на рынке PC-совместимых компьютеров мышь не пользовалась особым спросом, но с появлением Windows стала почти обязательной принадлежностью всех систем.

Основными компонентами мыши являются:

- корпус;
- шарик — датчик перемещения мыши;
- несколько кнопок (обычно две) для подачи (или выбора) команд;
- кабель для соединения мыши с компьютером;
- разъем для подключения к компьютеру.

Количество кнопок может быть разным, но обычно их только две. Вращение шарика преобразуется в электрические сигналы, которые по кабелю передаются в компьютер. В некоторых конструкциях мыши устанавливается оптический датчик, с помощью которого регистрируются перемещения устройства относительно нарисованной координатной сетки. Такая оптическая мышь может работать только на специальном коврике.

Взаимодействие мыши и компьютера осуществляется с помощью специальной программы-драйвера, которая либо загружается отдельно, либо является частью системного программного обеспечения. В любом случае драйвер (встроенный или отдельный) преобразует получаемые от мыши электрические сигналы в информацию о положении указателя и состоянии кнопок.

Устроена мышь довольно просто: шарик касается двух валиков, один из которых вращается при движении вокруг оси X, а второй — вокруг оси Y.

На оси с валиками насажены небольшие диски с прорезями ("прерыватели"), через которые проходят (или не проходят) инфра-

красные лучи от соответствующих источников. При вращении дисков лучи периодически прерываются, что регистрируется соответствующими фотодатчиками. Каждый импульс прошедшего излучения расценивается как один шаг по одной из координат. Такие *оптико-механические* датчики получили наибольшее распространение.

Интерфейсы мыши

Мышь можно подключить к компьютеру тремя способами:

• через последовательный интерфейс;

• через специальный порт мыши на системной плате;

• через интерфейс шинной платы.

Драйвер мыши

Для того чтобы мышь работала, нужно загрузить соответствующую программу-драйвер. Лучше пользоваться драйверами, встроенными в Windows, при этом дополнительный драйвер не нужен. Загружать внешний драйвер (через файлы Autoexec.bat и Config.sys) следует только в том случае, если мышь предполагается использовать в DOS-приложениях.

Если мышь необходима для работы в DOS, т.е. вне Windows, драйвер должен быть загружен через Config.sys или Autoexec.bat. Драйвер, загружаемый через файл Config.sys, обычно называется mouse.sys, а через Autoexec.bat — mouse.com. У различных фирм-производителей эти драйверы называются по-разному. При работе с Windows данные драйверы загружать не нужно.

IntelliMouse фирмы Microsoft

В конце 1996 года Microsoft представила новую модель мыши — IntelliMouse. Новое устройство выглядит практически так же, как и стандартная мышь Microsoft, но между правой и левой кнопкой у нее есть маленькое колесико.

Функций у колесика две. Во-первых, оно работает как устройство для прокрутки изображений на экране — очень удобно просматривать документы или Web-страницы, слегка прокручивая колесико вверх и вниз указательным пальцем. Во-вторых, если на колесико нажать, оно сработает как третья кнопка мыши.

Колесико работает *только* с теми программами, которые поддерживают новые возможности мыши.

8.6. Модули ввода-вывода

Обязательным компонентом любого компьютера, помимо процессора и модулей памяти, является подсистема ввода-вывода, состоящая из одного или нескольких *модулей ввода-вывода*. Каждый модуль ввода-вывода взаимодействует через системную магистраль или какой-либо центральный распределитель информации с остальными базовыми компонентами системы и управляет работой одного или нескольких внешних устройств. Модуль ввода-вывода представляет собой не просто набор разъемов, через которые сигналы от внешних устройств подключаются к системной магистрали, а включает сложную схему управления функциями обмена информацией между устройством и магистралью.

На специализированные устройства ввода-вывода, независимо от того, какими внешними устройствами они будут управлять, возлагается выполнение двух основных функций:

1) взаимодействие с процессором и оперативной памятью через системную магистраль или какой-либо центральный распределитель информации;
2) взаимодействие с одним или несколькими внешними устройствами через специализированные линии передачи данных.

Функции, возлагаемые на модули ввода-вывода, можно разделить на следующие категории:

• управление и синхронизация;

• связь с процессором;

• связь с внешним устройством;

• временная буферизация данных;

• обнаружение ошибок и сбоев.

Взаимодействие с процессором предполагает выполнение следующих операций:

• *расшифровка команд.* Модуль ввода-вывода получает от процессора команды, как правило, в виде сигналов на линиях управления, которые входят в комплект линий системной магистрали;

• *данные.* Данные передаются между процессором и модулем ввода-вывода по линиям данных системной магистрали;

• *информация о текущем состоянии.* Поскольку периферийные устройства работают значительно медленнее, чем внутренние компоненты вычислительной системы, очень важно всегда знать, каково текущее состояние устройства;

• *распознавание адреса.*

С другой стороны, модуль ввода-вывода должен *взаимодействовать с внешним устройством* — одним или даже несколькими. Такое взаимодействие включает передачу команд, обмен данными и прием информации о текущем состоянии.

Важной для модулей ввода-вывода является задача *буферизации данных*. Данные, передаваемые из оперативной памяти на внешнее устройство, поступают с очень высокой скоростью в виде пакета. Эти данные сохраняются во внутреннем буфере модуля ввода-вывода и тоненьким ручейком (малыми порциями) сливаются во внешнее устройство. При передаче в обратном направлении данные от внешне-

го устройства накапливаются в буфере, чтобы не дергать оперативную память "по мелочам". Когда накоплен значительный объем данных, они все "одним махом" (единым пакетом) быстро передаются в память. Таким образом, модуль ввода-вывода должен быть способен работать как с высокой скоростью, присущей внутренним компонентам системы, так и с низкой скоростью, характерной для периферийных устройств.

Последняя из задач, которые возлагаются на модуль ввода-вывода — обнаружение ошибок и формирование соответствующего сообщения процессору. Один класс ошибок — это сбои в работе механических (электромеханических) и электронных узлов внешнего устройства (прекращение в подачи бумаги, обнаружение дефектной дорожки на носителе и т.д.). Другой класс — случайные искажения кода, передаваемого внешним устройством модулю ввода-вывода. Такие искажения обнаруживаются с помощью специального кодирования пересылаемых данных. Наиболее распространенным способом является использование дополнительного бита четности на каждый пересылаемый байт данных или на каждый пересылаемый символ. Этот дополнительный бит формируется на передающем конце таким образом, чтобы общее количество единиц в информационном байте и контрольном коде было четным (контроль по четности) или нечетным (контроль по нечетности). Получив информационный байт и сопровождающий его контрольный бит, модуль ввода-вывода проверяет выполнение оговоренного условия и, таким образом, способен обнаружить искажение данных, если оно произошло на пути от передатчика данных к приемнику.

Существует четыре типа команд, которые процессор может передавать модулю ввода-вывода:

- *команды управления.* Служат для активизации внешнего устройства и задают тип операции, которую устройство должно выполнить;
- *команды опроса состояния.* Используются для анализа отдельных параметров текущего состояния модуля и (или) подключенного к нему внешнего устройства;
- *команда чтения.* Инициирует прием модулем ввода-вывода очередной порции данных от внешнего устройства. Эти данные размещаются во внутреннем буфере модуля;
- *команда записи.* Инициирует передачу модулем ввода-вывода порции данных внешнему устройству. Передаваемые данные (как правило байт или слово), выставляются процессором на линии данных системной магистрали.

Функции прямого доступа к памяти

Применение технологии прямого доступа к памяти требует включения в состав аппаратуры, обслуживающей системную магистраль, дополнительного модуля — *контроллера DMA.* Такой контроллер способен в определенной степени "подменить" процессор при передаче данных по системной магистрали между оперативной памятью и внешним устройством. Магистраль передается под управление контроллера DMA только тогда, когда она не нужна процессору, либо же контроллер должен каким-то образом "заставить" процессор временно приостановить работу. Последний метод нашел наиболее широкое применение и получил наименование *занятие цикла магистрали (cycle stealing),* поскольку контроллер при этом как бы "похищает" у процессора цикл работы системной магистрали.

Если в программе встречается инструкция обмена блоком данных с внешним устройством, процессор передает соответствующую

команду контроллеру DMA, сопровождая ее следующей информацией:

- сигналом на линии чтения или записи, соединяющей процессор и контроллер DMA, в зависимости от того, какая именно операция "заказана" в программе;
- адресом внешнего устройства, с которым должен быть выполнен обмен данными;
- начальным адресом участка в оперативной памяти, который используется в операции обмена (при выводе на внешнее устройство это адрес массива выводимых данных, а при вводе с внешнего устройства это адрес буфера-приемника);
- количеством слов в блоке.

8.7. Внешние интерфейсы SCSI и FireWare

Интерфейс *SCSI* (*small computer system interface* — интерфейс малых вычислительных систем) впервые был специфицирован применительно к компьютерам Macintosh в 1984 году. Интерфейс *SCSI* является стандартным интерфейсом для подключения приводов компакт-дисков, звуковых плат и внешних устройств массовой памяти.

Структура *SCSI,* по существу, является магистральной, хотя устройства включаются в нее по принципу последовательной цепочки. Каждое SCSI-устройство имеет два разъема — один входной, а другой выходной. Все устройства объединяются в последовательную цепочку, один конец которой подключается к компьютеру (фактически к контроллеру интерфейса). Все устройства работают независимо и могут обмениваться данными как с компьютером, так и друг с другом. Данные передаются пакетами сообщений.

Преимущество интерфейса *FireWare* по сравнению с SCSI состоит в низкой стоимости и простоте внедрения при сохранении до-

статочно высокой пропускной способности. Одно из основных достоинств интерфейса *FireWare* состоит в использовании последовательного способа передачи, который позволяет заменить дорогостоящие многожильные кабели передачи сигналов и многоконтактные разъемы значительно более дешевым одножильным кабелем и более простыми коаксиальными разъемами.

Применение последовательного интерфейса позволяет значительно уменьшить физические размеры соответствующей коммутационной аппаратуры.

Подключиться к такому разъему можно на ощупь, даже не поворачивая системный блок.

Интерфейс *FireWare* обеспечивает возможность так называемого "горячего" подключения, когда устройство можно подключить к интерфейсу или отключить от него, не выключая компьютер и не переконфигурируя систему. При использовании *SCSI* на обоих концах магистрали нужно установить заглушки, а каждому устройству присвоить уникальный идентификатор — адрес на магистрали. При использовании интерфейса *FireWare* никакие заглушки не нужны, а адреса присваиваются автоматически.

Заключение

Любой компьютер, даже, на первый взгляд, самый простой, представляет собой сложную систему, состоящую из сотен тысяч или даже миллионов простейших электронных компонентов. Единственный способ описать функционирование и структуру подобного рода сложных систем — выполнить ее иерархическую декомпозицию.

В настоящем учебном пособии доступно и исчерпывающе изложены основные принципы построения и функционирования такой системы — персонального компьютера (ПК).

Даны в самом общем смысле основные функции и структура ПК; проанализированы история развития вычислительной техники и продемонстрирована в исторической перспективе эволюция взглядов на фундаментальные концепции структурной организации и архитектуры компьютерных систем.

Анализ этих тенденций дает богатую пищу для размышлений о технологиях и стратегиях достижения сбалансированности и эффективности всех компонентов системы.

В пособии также показано представление информации в электронно-вычислительных машинах на конкретных примерах. Описано, как запоминающие элементы и функциональные блоки могут быть реализованы на цифровых логических элементах.

Проиллюстрированы методы синтеза комбинированных схем и конечных автоматов из этих элементов; проанализированы основные характеристики систем команд, рассмотрено соответствие между набором машинных команд и операторами языка ассемблер.

Особое внимание акцентируется на возможных вариантах выбора и их влиянии на эксплуатационные характеристики ПК.

В работе дан полный обзор функций арифметико-логического устройства, устройства управления, регистров; описано функционирование процессора в ходе выполнения машинных команд; детально проанализированы операции, выполняемые в отдельные такты цикла выполнения команды — извлечение команды, извлечение операндов, выполнение операции и прерывание.

Также в учебном пособии исчерпывающе изложены принципы организации внутренней и внешней памятей компьютера, а также принципы работы внешних устройств электронно-вычислительных машин.

Детально обсуждается базовая система ввода-вывода (Basic Output-Input System (BIOS)). Подробно описана оперативная память компьютера; рассмотрена многоуровневая иерархия средств хранения информации.

В пособии проанализированы варианты конструкций внешней памяти: основные принципы магнитного хранения данных, жесткие диски, дисководы, высокоемкие съемные устройства хранения информации, устройства оптического хранения данных.

Рассмотрены и устройства для записи компакт-дисков и перезаписываемых компакт-дисков, а также устройства, в которых используются другие оптические технологии.

Проанализированы разные варианты механизма взаимодействия модулей ввода-вывода с остальными компонентами компьютерных систем: программируемый ввод-вывод, ввод-вывод с прямым доступом.

В работе также описаны аппаратные средства отображения информации, включая видеоадаптеры и мониторы; принципы работы клавиатуры, мыши. Рассмотрен интерфейс между модулями ввода-вывода и внешними устройствами.

Таким образом, в разработанном учебном пособии в полной мере изложена система знаний по дисциплине «Архитектура компьютера» с постепенным нарастанием сложности излагаемых сведений, с демонстрацией внутридисциплинарных связей.

Знания, полученные при изучении данной дисциплины, могут быть применены в различных областях использования компьютерных технологий.

Библиографический список

1. Абель П. Язык ассемблера для IBM PC и программирования. Пер. с англ./ П. Абель. – М.: Высшая школа, 1992.
2. Богословский А.В. Системное программирование на Ассемблере для ПК / А.В. Богословский. – М.: Наука, 1992.
3. Голубь Н.Г. Искусство программирования на Ассемблере. Лекции и упражнения / Н.Г. Голубь. – СПб.: ООО «ДиаСофтЮП», 2002. – 656 с.
4. Максимов Н.В. Архитектура ЭВМ и вычислительных систем: Учебник для вузов / Н.В. Максимов, Т.Л. Партыка, И.И. Попов. – М: Инфра-М, 2005. – 512 с.
5. Масина О.Н. Язык программирования Assembler. Лекции и лабораторный практикум: Учебное пособие / О.Н. Масина, И.И.Васильева, В.А. Дякина. – М.: МГОУ, Елец: ЕГУ им. И.А.Бунина, 2004. – 301 с.
6. Масина О.Н. Практикум на ЭВМ: Учебное пособие / О.Н. Масина, В.А. Дякина. – М.: МГОУ, Елец: ЕГУ им. И.А.Бунина, 2006. – 291 с.
7. Скотт М. Модернизация и ремонт ПК: Учебное пособие / М.Скотт — М.: Издательский дом «Вильямс», 2001. – 1184 с.
8. Пильщиков В.Н. программирование на языке ассемблера IBM PC / В.Н. Пильщиков – М.: ДИАЛОГ-МИФИ, 1998. – 288 с.
9. Пирогов В.Ю. Assembler. Учебный курс / В.Ю.Пирогов. – М.: Нолидж, 2001. – 848 с.
10. Столлинг В. Структурная организация и архитектура компьютерных систем / В.Столлинг. – М.: Издательский дом «Вильямс», 2002. — 896 с.
11. Таненбаум Э. Архитектура компьютера / Э. Таненбаум. – СПб.: Питер, 2005. – 699 с.
12. Терехов Ю.П. Электронно-вычислительные машины: основы функционирования, архитектура, программное обеспечение. Учебное пособие / Ю.П. Терехов, Е.А.Быкова. – Елец: ЕГПИ, 1999. – 185 с.
13. Финогенов К. Г. Основы языка Ассемблера / К. Г. Финогенов. – М.: Радио и связь, 2001. – 288 с.
14. Щукина О.Н. Архитектура компьютера. Часть I. Информационно-логическая и функционально-структурная организация персонального компьютера / О.Н. Щукина. – Москва-Елец: МГОУ, ЕГУ им. И.А. Бунина, 2003. – 234 с.
15. Щукина О.Н. Архитектура компьютера. Часть II. Память и внешние устройства электронно-вычислительных машин / О.Н.

Щукина. – Москва-Елец: МГОУ, ЕГУ им. И.А. Бунина, 2003. – 187 с.
16. Юров В.И. Assembler: практикум / В. И.Юров. – СПб.: Питер, 2002. – 400 с.
17. Юров В.И. Assembler: Учебник для вузов / В. И. Юров. – СПб.: Питер, 2002. – 624 с.
18. Яснева Г.Г. Логические основы ЭВМ / Г.Г. Яснева // Информатика и образование, 1998. №2.

СВЕДЕНИЯ ОБ АВТОРЕ

Масина Ольга Николаевна – кандидат физико-математических наук, доцент кафедры ВМиИ ЕГУ им. И.А.Бунина

Учебное издание

Ольга Николаевна Масина

КУРС ЛЕКЦИЙ ПО ДИСЦИПЛИНЕ

«АРХИТЕКТУРА КОМПЬЮТЕРА»

Учебное пособие

Технический редактор – Н. П. Безногих
Техническое исполнение – В. Н. Бутов
Компьютерный набор – О.Н. Масина
Компьютерная верстка – О.Н. Масина
Переплет и обложка выполнены
в МУП "Типография" г. Ельца

Лицензия на издательскую деятельность
ИД № 06146. Дата выдачи 26.10.01.

Формат 60 x 84 1/16. Гарнитура Times. Печать трафаретная.
Усл.-печ.л. 16,7 Уч.-изд.л. 16,9
Тираж 500 экз. (1-й завод 1-100 экз.). Заказ

Отпечатано с готового оригинал-макета на участке оперативной полиграфии
Елецкого государственного университета им. И.А.Бунина.

Елецкий государственный университет им. И.А. Бунина.
399770, г. Елец, ул. Коммунаров, 28.