ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ КАЗЕННОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«МОСКОВСКИЙ УНИВЕРСИТЕТ МИНИСТЕРСТВА ВНУТРЕННИХ ДЕЛ
РОССИЙСКОЙ ФЕДЕРАЦИИ ИМЕНИ В.Я. КИКОТЯ»

СРЕДСТВА ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ

Учебное пособие

Москва
2020

ББК 32.97
С75

Рецензенты:
профессор кафедры информационной безопасности
Краснодарского университета МВД России
доктор технических наук, профессор **А. В. Еськов**;
старший преподаватель кафедры информационной безопасности
Воронежского института МВД России
кандидат технических наук **С. В. Зарубин**

Коллектив авторов:
К. К. Борзунов, О. В. Пименова, В. И. Лустин,
С. А. Сребродольская

С75 **Средства вычислительной техники** : учебное пособие / [К. К. Борзунов и др.]. – М. : Московский университет МВД России имени В.Я. Кикотя, 2020. – 380 с.
ISBN 978-5-9694-0875-3

Целью учебного пособия является передача курсантам знаний по теоретическим и практическим вопросам современных тенденций в развитии аппаратного и программного обеспечения средств вычислительной техники, вычислительных систем; наличия и развития периферийных устройств; архитектур компьютера и микропроцессора; первичной инициализации компьютера и режимов загрузки операционных систем Windows и Linux; физических основ хранения данных и организации структуры хранения данных на носителях информации; архитектуры и структуры вычислительных систем, организации обработки данных; коммутации компонентов системных блоков и периферийных устройств, внутренних и внешних интерфейсов, формирования сетей и организации передачи данных.

Учебное пособие предназначено для курсантов Московского университета МВД России имени В.Я. Кикотя. Может также использоваться при подготовке специалистов в области информационной безопасности.

ББК 32.97

ISBN 978-5-9694-0875-3

© Московский университет
МВД России имени В.Я. Кикотя, 2020
© Борзунов К. К., Пименова О. В.,
Лустин В. И., Сребродольская С. А., 2020

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ
АППАРАТНОЕ И ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ КОМПЬЮТЕРА

Создание и дальнейшее развитие компьютеров привели их разработчиков к увязыванию воедино трех важнейших составляющих, таких как общий интеллектуальный замысел, аппаратное обеспечение и программное обеспечение.

На наиболее ранних этапах появления средств вычислительной техники создавались механизмы, способные выполнять отдельные вычислительные операции, в дальнейшем появились электромеханические приборы, выполнявшие более сложные вычисления. Однако они представляли собой специализированные средства для решения узкого круга задач. И лишь к сороковым годам прошлого века возникла идея разработки универсальной вычислительной машины, решающей широчайший круг задач. Именно в этот период произошел качественный скачок, связанный с появлением интеллектуальной составляющей. Это был дерзкий замысел, представление о том, что вычислительная техника должна решать практически неограниченный круг задач, в основу решений которых должны быть положены алгоритмы, реализуемые в виде программ. Вычислительные средства должны решать задачи посредством выполнения ограниченного набора команд, содержащихся в самых разнообразных вводимых программах. В качестве основы вычислительных средств были выбраны электромеханические приборы, содержащие электрические цепи с электронными элементами – радиоэлектронными лампами. В настоящее же время используется современная микроэлектроника.

Таким образом, общий интеллектуальный замысел дает в целом представление о том, каким должен быть компьютер.

Круг решаемых задач определяется на уровне замысла. Методы решения задач реализуются путем разработки алгоритмов. В последующем на основе алгоритмов создаются программы, которые будет выполнять компьютер. И это определяет появление программного обеспечения.

На этапе разработки компьютера выбирается его технологическая основа. Прорабатывается архитектура компьютера. Создаются отдельные блоки и узлы (элементы) компьютера, которые определяют появление аппаратного обеспечения.

Аппаратное обеспечение (от англ. *hardware* – аппаратные средства, технические средства) включает в себя все физические части компьютера. Аппаратное обеспечение компьютера – это набор устройств блоков, узлов, элементов, из которых состоит компьютер, и тех, которые могут быть к нему подключены. Устройства, из которых состоит компьютер (например, процессор), обычно называют *комплектующими* или *внутренними*, а устройства, которые к нему могут быть подключены (например, принтер), называют *периферийными устройствами* или *внешними*.

Согласование между устройствами (отдельными блоками, узлами, элементами) выполняется с помощью аппаратно-логических устройств, называемых аппаратными интерфейсами. Стандарты на аппаратные интерфейсы называются протоколами, представляющими собой совокупность технических условий, которые должны быть обеспечены разработчиками устройств.

Программное обеспечение (от англ. *software*) – это вся совокупность программ, хранящихся на устройствах долговременной памяти компьютера. Программное обеспечение включает в себя комплекс необходимых программ – инструкций для компьютера, записанных в понятной компьютеру форме: как ему следует

выполнять ту или иную задачу, как вводить исходные данные, как их надо обрабатывать и как выводить результаты. Программное обеспечение является неотъемлемой частью компьютерной системы, логическим продолжением технических средств. Сфера применения конкретного компьютера определяется созданным для него программным обеспечением. Программное обеспечение образует на компьютере определенную среду для работы и включает в себя инструментарий, с помощью которого имеется возможность создавать любые компьютерные объекты.

Некоторые программы необходимо просто «запустить» пользователю, и они работают без какого-либо его участия. Такие программы называют *автоматизированными*. Другие же программы, наоборот, поддерживают диалог с пользователем, задавая ему вопросы и выполняя его команды. Такие программы называют *интерактивными* (от англ. *interactive* – взаимодействующий, воздействующий друг на друга). Подавляющее множество программ, с которыми работают пользователи, являются интерактивными.

Таким образом, компьютер имеет две важные составляющие: аппаратуру и программы. Для полноценного функционирования необходимо обеспечить компьютер и тем, и другим. Аппаратное и программное обеспечение неразрывно связаны друг с другом. Без программ аппаратура является просто «железом», а без аппаратуры программы будут никому не нужными инструкциями для выполнения каких-то действий.

ГЛАВА 1
ИСТОРИЧЕСКИЕ АСПЕКТЫ РАЗВИТИЯ ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ

1.1. Поколения компьютеров

В короткой истории компьютерной техники выделяют несколько периодов на основе того, какие базисные элементы использовались для изготовления компьютера. Временное деление на периоды в определенной степени условно, так как когда еще выпускались компьютеры старого поколения, новое поколение уже начинало набирать обороты.

Можно выделить общие тенденции развития компьютеров:

1. Увеличение количества элементов на единицу площади.
2. Уменьшение размеров.
3. Увеличение скорости работы.
4. Снижение стоимости.
5. Развитие программных средств, с одной стороны, и упрощение, стандартизация аппаратных – с другой.

Нулевое поколение. Механические вычислители. Предпосылки к появлению компьютера формировались, наверное, с древних времен, однако, нередко обзор начинают со счетной машины Блеза Паскаля, которую он сконструировал в 1642 г.

Машина Блеза Паскаля могла выполнять лишь операции сложения и вычитания. В 1670-х гг. Готфрид Вильгельм Лейбниц построил машину, умеющую выполнять операции не только сложения и вычитания, но и умножения и деления.

В XIX в. большой вклад в будущее развитие вычислительной техники внес Чарльз Бэббидж. Его *разностная машина*, хотя и умела только складывать и вычитать, зато результаты вычислений выдавливались на медной пластине (аналог средств ввода/вывода информации). В дальнейшем описанная Бэббиджем *аналитическая машина* должна была выполнять все четыре основные математические операции. Аналитическая машина состояла из памяти, вычислительного механизма и устройств ввода/вывода (прямо-таки компьютер, только механический), а главное могла выполнять различные алгоритмы (в зависимости от того, какая перфокарта находилась в устройстве ввода). Программы для аналитической машины писала Ада Лавлейс (первый известный программист). На самом деле машина не была реализована в то время из-за технических и финансовых сложностей. Мир отставал от хода мыслей Бэббиджа.

В XX в. автоматические счетные машины конструировали Конрад Зус, Джорж Стибиц, Джон Атанасов. Машина последнего включала, можно сказать, прототип ОЗУ, а также использовала бинарную арифметику. Релейные компьютеры Говарда Айкена: «Марк I» и «Марк II» были схожи по архитектуре с аналитической машиной Бэббиджа.

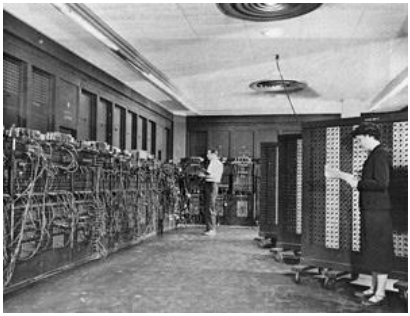
Первое поколение. Компьютеры на электронных лампах (1940-е – 1955 гг.). Быстродействие: несколько десятков тысяч операций в секунду.

Особенности компьютеров первого поколения:

– поскольку лампы имеют существенные размеры и их тысячи, то машины имели огромные размеры;

– поскольку ламп много и они имеют свойство перегорать, то часто компьютер простаивал из-за поиска и замены вышедшей из строя лампы;

– лампы выделяют большое количество тепла, следовательно, вычислительные машины требуют специальные мощные охладительные системы.



Примеры компьютеров:

Колоссус – секретная разработка британского правительства (в разработке принимал участие Алан Тьюринг). Это первый в мире электронный компьютер, хотя и не оказавший влияние на развитие компьютерной техники (из-за своей секретности), но помогший победить во Второй мировой войне.

Эниак. Создатели: Джон Моучли и Дж. Преспер Эккерт. Вес машины: 30 т. Минусы: использование десятичной системы счисления; множество переключателей и кабелей.

Эдсак. Достижение: первая машина с программой в памяти.

Whirlwind I. Слова малой длины, работа в реальном времени.

Компьютер 701 (и последующие модели) фирмы IBM. Первый компьютер, лидирующий на рынке в течение 10 лет.

Второе поколение. Компьютеры на транзисторах (1955–1965 гг.). Быстродействие: сотни тысяч операций в секунду.

По сравнению с электронными лампами использование транзисторов позволило уменьшить размеры вычислительной

техники, повысить надежность, увеличить скорость работы (до 1 млн операций в секунду) и почти свести на нет теплоотдачу. Развиваются способы хранения информации: широко используется магнитная лента, позже появляются диски. В этот период была создана первая компьютерная игра.

Первый компьютер на транзисторах *TX* стал прототипом для компьютеров ветки *PDP* фирмы *DEC*, которые можно считать родоначальниками компьютерной промышленности, так как началась массовая продажа машин. DEC выпускает первый миникомпьютер (размером со шкаф). Зафиксировано появление дисплея.

Фирма IBM также активно трудится, производя уже транзисторные версии своих компьютеров.

Компьютер 6600 фирмы CDC, который разработал Сеймур Крей, имел преимущество над другими компьютерами того времени – его быстродействие достигалось за счет параллельного выполнения команд.

Третье поколение. Компьютеры на интегральных схемах (1965–1980 гг.). Быстродействие: миллионы операций в секунду.

Интегральная схема представляет собой электронную схему, вытравленную на кремниевом кристалле. На такой схеме умещаются тысячи транзисторов. Следовательно, компьютеры этого поколения стали меньше, быстрее и дешевле.

Последнее свойство позволяло компьютерам проникать в различные сферы деятельности человека. Из-за этого они становились более специализированными (т. е. имелись различные вычислительные машины под различные задачи).

Появилась проблема совместимости выпускаемых моделей (программного обеспечения под них). Впервые большое внимание совместимости уделила компания IBM.

Было реализовано мультипрограммирование (когда в памяти находится несколько выполняемых программ, это дает эффект экономии ресурсов процессора).

Дальнейшее развитие миникомпьютеров (в частности, *PDP-11*) ознаменовало появление очередного поколения компьютеров.

Четвертое поколение. Компьютеры на больших (и сверхбольших) интегральных схемах (с 1980-х гг.). Быстродействие: сотни миллионов операций в секунду.

Появилась возможность размещать на одном кристалле не одну интегральную схему, а тысячи. Быстродействие компью-

теров значительно увеличилось. Компьютеры продолжали дешеветь и теперь их покупали даже отдельные личности, что ознаменовало так называемую эру персональных компьютеров. Но отдельная личность чаще всего не была профессиональным программистом. Следовательно, потребовалось развитие программного обеспечения, чтобы личность могла использовать компьютер в соответствие со своей фантазией.

В конце 1970-х – начале 1980-х гг. популярностью пользовался компьютер *Apple*, разработанный Стивом Джобсом и Стивом Возняком. Позднее в массовое производство был запущен персональный компьютер *IBM PC* на процессоре Intel.

Затем появились суперскалярные процессоры, способные выполнять множество команд одновременно, а также 64-разрядные компьютеры.

Пятое поколение? Сюда относят неудавшийся проект Японии (хорошо описан в Википедии). Другие источники относят к пятому поколению вычислительных машин так называемые невидимые компьютеры (микроконтроллеры, встраиваемые в бытовую технику, машины и др.) или карманные компьютеры.

Также существует мнение, что к пятому поколению следует относить компьютеры с двухъядерными процессорами. С этой точки зрения, пятое поколение началось примерно с 2005 г.

1.2. Аналоговые, цифровые и гибридные вычислительные машины

В зависимости от вида перерабатываемой информации компьютеры подразделяют на два основных класса: аналоговые и цифровые.

Аналоговые вычислительные машины представляют собой, условно говоря, первое направление развития вычислительной техники.

Аналоговый компьютер – это вычислительная машина, оперирующая информацией, представленной в виде непрерывных изменений некоторых физических величин. При этом в качестве физических переменных выступают сила тока электрической цепи, угол поворота вала, скорость и ускорение движения тела и т. п. Используя тот факт, что многие явления в природе математически описываются одними и теми же уравнениями, аналоговые вычислительные машины позволяют с помощью одного физического процесса моделировать различные другие процессы.

Аналоговые компьютеры (ранее их называли «аналоговые вычислительные машины» – АВМ) достаточно просты и удобны в эксплуатации, хотя программирование задач для решения на них все-таки трудоемкое; скорость решения задач может изменяться оператором в большую сторону (выше по сравнению с цифровыми компьютерами, но точность решения задач остается на уровне инженерных расчетов (относительная погрешность от 2 до 5 %). На АВМ наиболее эффективно решали ранее математические задачи, связанные с дифференциальными или интегральными уравнениями и их системами.

Цифровой компьютер (раньше называли «цифровые вычислительные машины» – ЦВМ) – это вычислительная машина, которая оперирует информацией, представленной в дискретном

виде или иначе – в цифровой форме. В настоящее время разработаны методы численного решения многих видов уравнений, что дало возможность решать на цифровых вычислительных машинах различные уравнения и задачи с помощью набора простых арифметических и логических операций. Поэтому если аналоговые вычислительные машины обычно предназначены для решения определенного класса задач, т. е. являются специализированными, то цифровой компьютер, как правило, – универсальное вычислительное средство.

Наиболее широкое применение получили ЦВМ с электрическим представлением дискретной информации – электронные цифровые вычислительные машины, обычно называемые просто электронными вычислительными машинами (ЭВМ) или в настоящее время – компьютеры, без упоминания об их цифровом характере. В настоящее время они производятся с использованием новейших достижений электроники в области нанотехнологий.

И третье направление развития вычислительной техники реализуется в виде устройств, способных обрабатывать информацию как в аналоговой, так и в цифровой форме.

Гибридный компьютер (называли ранее «гибридные вычислительные машины» – ГВМ) – это вычислительные машины комбинированного действия, которые способны работать с информацией, представленной как в цифровой, так и в аналоговой форме. ГВМ совмещают в себе достоинства и АВМ, и ЦВМ. ГВМ чаще всего используют для решения задач управления сложными быстродействующими техническими комплексами или быстропротекающими процессами.

ГЛАВА 2
ПРЕДСТАВЛЕНИЕ ИНФОРМАЦИИ В КОМПЬЮТЕРЕ

Компьютер, помогающий человеку хранить и обрабатывать информацию, приспособлен, в первую очередь, для обработки текстовой, числовой, графической информации. Рассмотрим способы представления информации в компьютере, в первую очередь – представление целых и вещественных чисел, текстовой информации.

2.1. Кодирование и декодирование информации. Системы счисления

Правило, описывающее однозначное соответствие букв алфавита и дополнительных знаков другому набору алфавита и дополнительных знаков, называют *кодом*.

Саму процедуру преобразования сообщения называют перекодировкой. Подобное преобразование может осуществляться в момент поступления сообщения от источника в канал связи (собственно *кодирование*) и в момент приема сообщения получателем (собственно *декодирование*).

Система счисления – принятый способ записи чисел и сопоставления этим записям реальных значений.

Система счисления – совокупность приемов и правил наименования и обозначения чисел, позволяющих установить взаимно однозначное соответствие между любым числом и его представлением в виде конечного числа символов.

Системы счисления могут быть *позиционные* и *непозиционные*.

Пример непозиционной системы счисления – это всем хорошо известные так называемые римские числа, а для позиционной системы счисления – так называемые арабские числа.

Констатируем факт: *в системах счисления используется некий набор – некоторое количество – отличных друг от друга знаков*.

Непозиционная система счисления – система, в которой символы, обозначающие то или иное количество, не меняют своего значения в зависимости от местоположения (позиции) в изображении числа.

Самой простой и, возможно, первой в истории человечества была система с одним символом (палочкой). Для изображения какого-либо числа в этой системе надо было записать количество палочек, равное данному числу.

В римской – непозиционной – системе счисления для записи чисел используется семь знаков: I, V, X, L, C, D, M.

Запись чисел в этой системе осуществляется по следующим правилам:

– если цифра слева меньше, чем цифра справа, то левая цифра вычитается из правой;

– если цифра справа меньше или равна цифре слева, то эти цифры складываются.

В настоящее время римская система используется, в основном, для нумерации.

Соответствие некоторых чисел римской – непозиционной – системы счисления числам десятичной – позиционной – системе счисления указывается в следующих таблицах:

римская – непозиционная система счисления	I	V	X	L	C	D	M
десятичная – позиционная система счисления	1	5	10	50	100	500	1000

десятичная – позиционная система счисления	2	3	4	6	9	11	30
римская – непозиционная система счисления	II	III	IV	VI	IX	XI	XXX

К основным недостаткам непозиционных систем счисления можно отнести:

– отсутствие нуля;

– необходимость содержания бесконечного количества символов;

– сложность арифметических действий над числами.

Позиционная система счисления – система, в которой значение цифры определяется ее местоположением (позицией) в изображении числа.

Число знаков в позиционных системах счисления называется *основанием системы счисления*.

Основание системы счисления	**Название системы счисления**	**Используемые знаки**
2	Двоичная	0; 1
8	Восьмеричная	0; 1; 2; 3; 4; 5; 6; 7
10	Десятичная	0; 1; 2; 3; 4; 5; 6; 7; 8; 9
16	Шестнадцатеричная	0; 1; 2; 3; 4; 5; 6; 7; 8; 9; A; B; C; D; E; F

Примеры позиционных систем счисления:

Десятичная	Восьмеричная	Шестнадцатеричная	Двоичная
0	0	0	0
1	1	1	1
2	2	2	10
3	3	3	11
4	4	4	100
5	5	5	101
6	6	6	110
7	7	7	111
8	10	8	1000
9	11	9	1001
10	12	A	1010
11	13	B	1011
12	14	C	1100
13	15	D	1101
14	16	E	1110
15	17	F	1111

Соответствие чисел в различных системах счисления:

В общем случае для задания *p* системы счисления необходимо определить основание *p* и алфавит, состоящий из *p* различных символов (чаще всего – цифр) a_i, где $i = 1, \ldots, p$.

Значение цифры на любой *i*-й позиции равно произведению:

$$A = a_i \cdot p^i,$$

где: *i* – номер позиции в числе, *p* – основание системы счисления, p^i – вес цифры *i*-го разряда.

Основание системы счисления, с одной стороны, определяют количество различных цифр, допустимое для данной системы счисления, а с другой – число, показывающее, во сколько раз вес цифры данного разряда меньше веса цифры соседнего старшего разряда.

Для позиционных систем счисления *значение числа* можно представить полиномом вида:

$$A = a_{n-1} \cdot p^{n-1} + a_{n-2} \cdot p^{n-2} + \ldots + a_i \cdot p + a_0 + a_{-1} \cdot p^{-1} + \ldots + a_{-m} \cdot p^{-m}$$

или $$A = \sum_{i=-m}^{n-1} a_i p^i .$$

Представление числа в виде суммы значений его цифр называется *развернутой* записью.

Примеры формального представления чисел в различных системах счисления:

$23{,}43_{(10)} = 2 \cdot 10^1 + 3 \cdot 10^0 + 4 \cdot 10^{-1} + 3 \cdot 10^{-2}$;

$1101_{(2)} = 1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0$;

$A1F{,}4_{(16)} = A \cdot 16^2 + 1 \cdot 16^1 + F \cdot 16^0 + 4 \cdot 16^{-1}$.

2.2. Представление информации в аналоговой и дискретной формах

Графическая и звуковая информация может быть представлена как в аналоговой, так и в дискретной форме. В случае представления данных аналоговой величиной ее физические значения могут принимать бесконечное множество. В случае дискретного представления данных физическая величина может принимать конечное множество значений. При этом она меняется скачкообразно. В качестве примера такого представления информации приведем наклонную плоскость и лестницу. Положение тела задается значениями координат **X** и **Y**. При движении тела по наклонной плоскости его координаты могут прини-

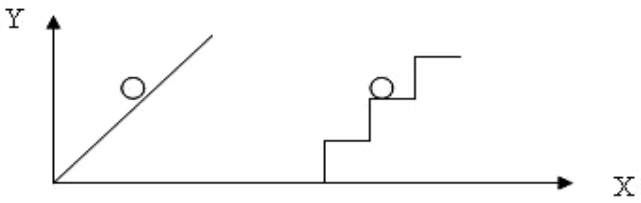


Рис. 1. Визуальный образ при аналоговом и дискретном представлении данных

мать бесконечное множество непрерывно меняющихся значений из определенного диапазона, а при движении по лестнице – только конечный набор значений, меняющихся скачкообразно.

Примером аналогового хранения звуковой информации является виниловая пластинка (звуковая дорожка свою форму меняет непрерывно), а дискретного – звуковой компакт-диск (звуковая дорожка диска содержит участки с различной отражающей способностью).

Преобразование информации из аналоговой формы в дискретную осуществляется путем дискретизации, т. е. разбиением непрерывного графического изображения или непрерывного звукового сигнала на отдельные дискретные элементы. При этом производится кодирование, т. е. присвоение каждому элементу конкретного значения в форме кода.

Под *дискретизацией* понимают преобразование непрерывных данных изображений или звуков в набор дискретных значений, каждому из которых присваивается определенный код.

В аналоговой форме аудиоданные представляют собой волну с непрерывно меняющимися амплитудой и частотой. При преобразовании аудиоданных в цифровую дискретную форму применяется дискретизация по времени, при которой в определенные моменты амплитуда звуковой волны измеряется и квантуется. С помощью аудиоредакторов открываются широкие возможности по созданию, редактированию и прослушиванию аудиофайлов. Уже созданы программы распознавания речи и, в результате, появилась возможность управления компьютером с помощью голоса.

Числовая информация – первый вид информации, который начали обрабатывать компьютеры, и долгое время работали именно с ней. Поэтому не удивительно, что создано большое разнообразие типов и представлений чисел. Прежде всего, это такие типы, как целые и вещественные числа, которые, по сути,

и по представлению в компьютере фактически не различаются. Целые числа, в свою очередь, могут быть со знаком или без знака, которые также мало различаются. Вещественные числа представляются двумя способами – с фиксированной и с плавающей запятой.

Со временем появились различные способы кодирования и декодирования информации, представляемой в компьютере. В первую очередь, это зависит от вида информации: текстовая, числовая, фото или видео, звуковая. Для числа также важно, где оно будет использовано: в тексте, в вычислениях или в процессе ввода/вывода.

Вся информация кодируется на основе двоичного кода: с помощью цифр **0** и **1**, относящихся к двоичной системе счисления. Эти два символа называют двоичными цифрами или битами. Такой способ кодирования технически реализуется просто. Например: **1** – есть электрический сигнал, **0** – нет сигнала. Недостаток двоичного кодирования – это объемные коды. Но в технике легче распознавать два разных физических состояния однотипных элементов, чем с большим числом состояний этих элементов.

Каждый регистр арифметического устройства компьютера, каждая ячейка памяти представляет собой физическую систему (рис. 2), состоящую из некоторого числа однородных элементов, обладающих двумя устойчивыми состояниями, одно из которых соответствует, например, нулю, а другое – единице. Каждый элемент служит для записи одного из разрядов двоичного числа. Именно поэтому элемент ячейки называют *разрядом*.

Рис. 2. Схема регистра и ячейки памяти

Вычислительная техника создавалась как средства автоматизации вычислений. Современные компьютеры обрабатывают различные виды информации: числовую, текстовую, звуковую, графическую и видеоданные. Тем не менее компьютер хранит и обрабатывает только дискретную информацию. Следовательно, любой вид информации, подлежащий компьютерной обработке, должен быть закодирован именно конечной последовательностью целых чисел, которая затем будет переведена в двоичные разряды для хранения на компьютере.

Задача перевода информации естественного происхождения в цифровую называется задачей *дискретизации* или *квантования*. Эту задачу постоянно решают для всех видов информации. Способы дискретизации разнообразной по видам информации различны, но подходы к решению этой задачи построены на одних и тех же принципах.

Кодирование информации проводят обычно в несколько этапов: определение объема информации, подлежащей кодированию; выбор системы кодирования и разработка кодовых обозначений; непосредственно кодирование.

2.3. Представление чисел в двоичной системе счисления

Как ранее отмечалось, основной системой счисления компьютера является двоичная система счисления (с использованием для представления чисел двух цифр 0 и 1).

Сравните:

в десятичной системе счисления –

$$821{,}46_{(10)} = 8 \cdot 10^2 + 2 \cdot 10^1 + 1 \cdot 10^0 + 4 \cdot 10^{-1} + 6 \cdot 10^{-2};$$

в двоичной системе счисления –

$$10110{,}101_{(2)} = 1 \cdot 2^4 + 0 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0 + 1 \cdot 2^{-1} + 0 \cdot 2^{-2} + 1 \cdot 2^{-3}.$$

Действия с числами в двоичной системе счисления соответствуют правилам именно двоичной арифметики. Однако можно отметить, что все основные законы арифметических действий для таких чисел выполняются.

Для сравнения рассмотрим два варианта кодирования для числа 45. При использовании числа в тексте каждая цифра кодируется 8 битами в соответствии с ASCII-кодом (т. е. требуется 2 байта): 4 – 01000011, 5 – 01010011. А при использовании этого же числа в вычислениях его код определяется по специальным правилам перевода из десятичной системы счисления в двоичную в виде 8-разрядного двоичного числа: $00101101_{(2)}$, что требует 1 байт.

Целые числа. Известно, что любое целое число можно рассматривать как вещественное, но с нулевой дробной частью, т. е. можно было бы ограничиться представлением в компьютере вещественных чисел и реализацией арифметических действий над ними. Однако для эффективного использования памяти, повышения скорости выполнения вычислений и введения операции деления нацело с остатком целые числа представляются специально для них предназначенными способами.

Введение специальных способов представления целых чисел оправдано тем, что достаточно часто в задачах, решаемых с помощью компьютера, многие действия сводятся к операциям над целыми числами[1].

Для компьютерного представления целых чисел обычно используется несколько различных способов представления,

[1] Например, в задачах экономического характера данными служат количества акций, сотрудников, деталей, транспортных средств и т. д., по своему смыслу являющиеся целыми числами. Целые числа используются и для обозначения даты и времени, и для нумерации различных объектов: элементов массивов, записей в базах данных, машинных адресов и т. п.

отличающихся друг от друга количеством разрядов и наличием или отсутствием знакового разряда. Беззнаковое представление можно использовать только для неотрицательных целых чисел, отрицательные числа представляются только в знаковом виде.

При беззнаковом представлении все разряды ячейки отводятся под само число. При представлении со знаком самый старший (левый) разряд отводится под знак числа, остальные разряды – под собственно число. Если число положительное, то в знаковый разряд помещается **0**, если отрицательное – **1**. Очевидно, в ячейках одного и того же размера можно представить больший диапазон целых неотрицательных чисел в беззнаковом представлении, чем чисел со знаком.

Например, в одном байте (8 разрядов) можно записать положительные числа от 0 до 255, а со знаком – только до 127. Поэтому, если известно заранее, что некоторая числовая величина всегда является неотрицательной, то выгоднее рассматривать ее как беззнаковую.

Целые числа в компьютере хранятся в формате с *фиксированной запятой*.

Для получения компьютерного представления беззнакового целого числа в k-разрядной ячейке памяти достаточно перевести его в двоичную систему счисления и дополнить полученный результат слева нулями до **k** разрядов. Понятно, что существует ограничение на числа, которые мы можем записать в k-разрядную ячейку.

Максимально представимому числу соответствуют единицы во всех разрядах ячейки (двоичное число, состоящее из k-единиц). Для k-разрядного представления оно будет равно $\mathbf{2k - 1}$. Минимальное число представляется нулями во всех разрядах ячейки, оно всегда равно нулю.

Таблица 1

Максимальные числа для беззнакового представления при различных значениях разрядов

Количество разрядов, k	Максимальное число
8	255 (28 − 1)
16	65535 (216 − 1)
32	4294967295 (232 − 1)
64	18446744073709551615 (264 − 1)

При знаковом представлении целых чисел в компьютере применяют прямой, обратный и дополнительный коды.

Представление числа в привычной для человека форме знак-величина, при которой старший разряд ячейки отводится под знак, остальные **k − 1** разрядов – под цифры числа, называется *прямым кодом*.

Например, прямые коды двоичных чисел 11001_2 и -11001_2 для восьмиразрядной ячейки равны 00011001 и 10011001 соответственно.

Положительные целые числа представляются в компьютере с помощью прямого кода. Прямой код отрицательного целого числа отличается от прямого кода соответствующего положительного числа содержимым знакового разряда.

Но вместо прямого кода для представления отрицательных целых чисел в компьютере используется *дополнительный код*.

Отметим, что максимальное положительное число, которое можно записать в знаковом представлении в **k** разрядах, равно $\mathbf{2^{k-1} - 1}$, что практически в два раза меньше максимального числа в беззнаковом представлении в тех же **k** разрядах.

Пример. Число $53 = 110101_2$ в восьмиразрядном представлении имеет вид:

0	0	1	1	0	1	0	1

Это же число 53 в 16 разрядах будет записано следующим образом:

0	0	0	0	0	0	0	0	0	0	1	1	0	1	0	1

В обоих случаях неважно, знаковое или беззнаковое представление при этом используется.

Пример. Для числа $200_{10} = 11001000_2$ представление в 8 разрядах со знаком невозможно, так как максимальное допустимое число в таком представлении равно 127, а в беззнаковом восьмиразрядном представлении оно имеет вид:

1	1	0	0	1	0	0	0

Для представления в компьютере целых отрицательных чисел используют дополнительный код, который позволяет заменить арифметическую операцию вычитания операцией сложения, что существенно увеличивает скорость вычислений. Прежде, чем вводить определение дополнительного кода, сделаем одно замечание.

В k-разрядной целочисленной компьютерной арифметике $2^k \equiv 0$.

Объяснить это можно тем, что двоичная запись числа $\mathbf{2^k}$ состоит из одной единицы и **k** нулей, а в ячейку из **k** разрядов может уместиться только **k** цифр, в данном случае только **k** нулей. В таком случае говорят, что значащая единица вышла за пределы разрядной сетки.

При этом **k**-разрядный дополнительный код отрицательного числа **m** – это запись в **k** разрядах положительного числа $\mathbf{2^k - |m|}$, где **|m|** – модуль отрицательного числа **m**, $\mathbf{|m| \le 2^{k-1}}$.

Обратный код является дополнением исходного числа до числа $\mathbf{2^{k-1}}$, состоящего из **k** двоичных единиц. Поэтому прибав-

ление единицы к инвертированному коду позволяет получить его искомый дополнительный код.

Итак, 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. Получим дополнительный код числа −52 для 8- и 16-разрядной ячеек.

Для восьмиразрядной ячейки:

0011 0100 – прямой код числа |−52| = 52;

1100 1011 – обратный код числа −52;

1100 1100 – дополнительный код числа −52.

Для шестнадцатиразрядной ячейки:

0000 0000 0011 0100 – прямой код числа |−52|;

1111 1111 1100 1011 – обратный код числа −52;

1111 1111 1100 1100 – дополнительный код числа −52.

Приведем значения границ диапазонов для знаковых представлений в ячейках с различной разрядностью:

Разрядность	Минимальное число	Максимальное число
8	−128	127
16	−32768	32767
32	−2147483648	2147483647
64	−9223372036854775808	9223372036854775807

Вещественные числа. Если при представлении целых чисел в компьютере ограничением может служить лишь величина записываемого числа, то при записи вещественного числа речь идет о тонности его представления, т. е. о количестве значащих цифр, которые удается сохранить в ограниченном числе разрядов.

Любое число a в экспоненциальной форме представляется в виде:

$$a = \pm m \cdot P^q,$$

где: P – основание системы счисления, m называется мантиссой числа, q – порядком числа.

Пример. Одна из сторон треугольника, гипотенуза, равна 21,6 см. В экспоненциальной форме запись можно представить таким образом:

1) $216 \cdot 10^{-1}$ см = 216 мм;
2) $2{,}16 \cdot 10^{1}$ см = 2,16 дм;
3) $21{,}6 \cdot 10^{0}$ см = 21,6 см;
4) $0{,}2168 \cdot 10^{2}$ см = 0,216 м.

Из этого примера видно, что длину одной и той же стороны треугольника можно записать с использованием различных экспоненциальных форм. Десятичная запятая «плавает» в числе и больше не помечает абсолютное место между целой и дробной частями. Эта неоднозначность записи может приводить в определенных случаях к неудобству. Из курса алгебры известно, что если **P** фиксировано и $\mathbf{1/P \le m < 1}$, то представление числа в экспоненциальной форме единственно. Такая форма экспоненциального представления называется *нормализованной формой* и используется в компьютере для однозначности представления вещественных чисел.

Нормализованная запись отличного от нуля вещественного числа – это запись вида $\mathbf{a = \pm m \cdot P^q}$, где **q** – целое число (положительное, отрицательное или ноль), а **m** – правильная **P**-ичная дробь, у которой первая цифра после запятой не равна нулю, т. е. $\mathbf{1/P \le m < 1}$.

Число ноль не может быть записано в нормализованной форме так, как она была определена. Поэтому относительно нормализованной записи нуля приходится прибегать к особым согла-

шениям. Условимся, что *запись нуля является нормализованной, если и мантисса, и порядок равны нулю*.

В нормализованной форме все числа записываются так, что запятая у них ставится в одном и том же месте – перед первой (самой левой) значащей цифрой (отличной от нуля) мантиссы. Заметим, что в двоичной системе счисления первая цифра мантиссы нормализованного числа всегда равна **1** (за исключением числа ноль). Величина же порядка числа указывается отдельно, с помощью соответствующей степени основания системы счисления, в которой это число было записано изначально. Количество цифр в мантиссе может оказаться меньше, чем число значащих цифр в исходном числе. Часто в нормализованной записи мантисса **P**-ичного числа записывается в **P**-ичной системе счисления, а порядок и само число **P** – в десятичной.

Приведем примеры нормализации чисел:

1) $0 = 0{,}0 \cdot 10^0$ *(возможная нормализация нуля);*

2) $2{,}83 = 0{,}283 \cdot 10^1$ *(количество значащих цифр не изменилось);*

3) $4000 = 0{,}4 \cdot 10^4$ *(количество значащих цифр уменьшилось с четырех до одной);*

4) $0{,}912873465 = 0{,}912873465 \cdot 10^0$ *(запятую передвигать не нужно);*

5) $9000{,}00071 = 0{,}900000071 \cdot 10^4$ *(количество значащих цифр уменьшить невозможно).*

При записи нормализованного числа в компьютере для записи мантиссы и порядка отводится заранее фиксированное количество разрядов.

В компьютерном представлении вещественных чисел максимально допустимое количество цифр в мантиссе определяет точность, с которой может быть представлено число.

Модуль разности между значением числа x и компьютерным его представлением x^* называется абсолютной погрешностью представления x.

Несмотря на то, что в абсолютном исчислении погрешность может быть значительно больше **1**, относительно величины самого числа ее порядок остается неизменным.

Относительная погрешность представления x – величина, равная:

$$\left| (x - x^*) / x \right| .$$

Абсолютная погрешность говорит о том, на сколько полученный результат (например, результат представления числа в компьютере) отличается от истинного результата (самого числа). При решении реальных задач знание этой величины позволяет оценивать, насколько достоверный результат был получен. Если его точность нас не удовлетворяет, то следует выбрать другой (более точный) способ представления чисел.

Относительная же погрешность показывает, сколько верных старших значащих цифр содержит результат. Значение относительной погрешности непосредственно связано с количеством разрядов, отводимых для представления мантиссы нормализованного числа. На практике обычно известна относительная погрешность представления чисел, так как разрядность мантиссы фиксирована.

Следовательно, можно предположить, сколько верных цифр содержит результат. Если из каких-либо априорных соображений известно значение вычисляемой величины, то можно оценить абсолютную погрешность результата.

В компьютерной записи вещественных чисел с плавающей запятой количество цифр, отводимых под запись порядка, определяет, насколько большие и насколько маленькие положительные числа могут быть представлены. Широкий диапазон пред-

ставления чисел с плавающей запятой необходим при решении научных и инженерных задач. Такое представление чисел не только позволяет сохранять в разрядной сетке большое количество значащих цифр и тем самым повышать точность вычислений, но также упрощает действия над порядками и мантиссами.

Как и для целых чисел, при представлении вещественных чисел в компьютере используется чаще всего двоичная система счисления, следовательно, предварительно десятичное число должно быть переведено в двоичную систему, а уж затем представлено в нормализованной форме с $\boldsymbol{P = 2}$. При представлении нормализованных чисел часть разрядов ячейки отводится для записи порядка числа, остальные разряды – для записи мантиссы. По одному разряду в каждой группе отводится для изображения знака порядка и знака мантиссы. О таком представлении говорят, что число записано в формате с *плавающей запятой*.

Например, можно представить себе такое распределение разрядов ячейки памяти:

s_q	s_m	b_k	b_{k-1}	...	b_2	b_1	a_1	a_2	...	a_{n-1}	a_n

Первые два разряда служат для представления знаков порядка (s_q) и мантиссы (s_m) соответственно. Следующие k разрядов используются для представления абсолютной величины порядка числа (q), остальные n разрядов – для представления абсолютной величины мантиссы. В каждом разряде ячейки может храниться одно из двух значений: **0** или **1**.

Тогда изображенному на схеме состоянию ячейки соответствует число:

$$(-1)^{S_m} \cdot 2^q \cdot \left(\frac{a_1}{2} + \frac{a_2}{2^2} + \cdots + \frac{a_n}{2^n} \right),$$

где: $q = (-1)^{Sq} (b_k \cdot 2^{k-1} + b_{k-1} \cdot 2^{k-2} + \ldots + b_2 \cdot 2 + b_1)$.

При такой системе записи наибольшее по абсолютной величине число, которое может быть представлено на компьютере, равно:

$$2^{2^{k-1}} \cdot (1 - 2^{-n}),$$

а наименьшее по абсолютной величине число, отличное от нуля, равно:

$$2^{2^{k-1}} \cdot 2^{-n} = 2^{-2^k - n + 1}.$$

Вещественных чисел, точно представимых в компьютере, конечное число. Остальные числа либо приближаются к представимым, либо оказываются непредставимыми. Последнее относится к слишком большим и слишком маленьким вещественным числам.

Использование в компьютере представления чисел в формате с плавающей запятой усложняет выполнение арифметических операций.

При сложении и вычитании чисел сначала производится подготовительная операция, называемая выравниванием порядков. Она состоит в том, что мантисса числа с меньшим порядком сдвигается в своей ячейке вправо на количество разрядов, равное разности порядков данных чисел. После этой операции одноименные разряды мантисс оказываются расположенными в одноименных разрядах обеих ячеек, и теперь уже сложение или вычитание мантисс выполняется достаточно просто, так же как над числами с фиксированной запятой.

После операций над порядками и мантиссами мы получаем порядок и мантиссу результата, но она может не удовлетворять ограничениям, накладываемым на мантиссы нормализированных чисел. Так как от результата арифметических операций в компьютере требуется, чтобы он также был нормализованным числом, необходимо дополнительное преобразование – нормализация. В зависимости от величины получившейся мантиссы

результата она сдвигается вправо или влево так, чтобы ее первая значащая цифра попала в первый разряд после запятой. Одновременно порядок результата увеличивается или уменьшается на число, равное величине сдвига.

Над мантиссами в арифметическом устройстве могут выполняться все четыре арифметических действия, а также операции сдвига, тогда как над порядками производятся только действия сложения и вычитания. Отрицательные порядки можно записывать в дополнительном коде для того, чтобы операцию вычитания свести к операции сложения.

В ряде случаев, даже если некоторые два числа были представлены в формате с плавающей запятой абсолютно точно, результат выполнения над ними арифметической операции может быть заведомо неверным.

У вещественной арифметики есть несколько потенциально опасных особенностей. Все они имеют общее происхождение, а именно, тот факт, что мантисса и порядок в представлении с плавающей запятой занимают фиксированное число разрядов.

При этом уже на стадии записи чисел в память компьютера возникают ошибки округления, которые при выполнении арифметических действий нарастают; наличие погрешностей округления приводит к следующим правилам программирования: неразумно сравнивать в программе два вещественных числа на точное равенство; в результате вычитания возникают недостоверные значащие цифры, которые могут привести к серьезной потере точности или получению неправильного результата; прибавление или вычитание малого числа может никак не сказаться на результате; получение очень больших чисел может вызвать переполнение или исчезновение числа (превращение в нуль), это может привести к аварийному завершению программы.

2.4. Представление (кодирование) чисел в ячейках памяти компьютера (разрядах)

Итак, как мы уже знаем: *вся информация (данные) представляется в компьютере в виде двоичных кодов*.

Для удобства работы вспомним следующие термины, обозначающие совокупности двоичных разрядов. Эти термины обычно используются в качестве единиц измерения объемов информации, хранимой и обрабатываемой в компьютере, и при передаче информации в телекоммуникационных сетях – сетях, объединяющих компьютеры в самом простом случае.

Примеры двоичных совокупностей:

Количество двоичных разрядов в группе	1	8	16	$8 \cdot 1024$	$8 \cdot 1024^2$	$8 \cdot 1024^3$	$8 \cdot 1024^4$
Наименование единиц измерения	бит	байт	параграф	Кбайт (кило-байт)	Мбайт (мега-байт)	Гбайт (гига-байт)	Тбайт (тера-байт)

Бит – это некоторая позиция (разряд), в которой может быть либо **0**, либо **1**.

Напомним некоторые единицы измерения: 8 бит (б) = 1 байт (Б) и далее:

1 килобайт = 1 Кбайт (КБ) = 1024 байт = 2^{10} = 1 024;

1 мегабайт = 1 Мбайт (МБ) = 1024 Кбайт = 2^{20} = 1 048 576;

1 гигабайт = 1 Гбайт (ГБ) = 1024 Мбайт = 2^{30} = 1 073 741 824;

1 терабайт = 1 Тбайт (ТБ) = 1024 Гбайт = 2^{40} = 1 099 511 627 776;

1 петабайт = 1 Пбайт (ПБ) = 1024 Тбайт = 2^{50} = 1 125 899 906 842 624;

1 эксабайт = 1 Эбайт (ЭБ) = 1024 Пбайт = 2^{60} = 1 152 921 504 606 846 976;

1 зеттабайт = 1 Збайт (ЗБ) = 1024 Эбайт = 2^{70} = 1 180 591 620 717 411 303 424;

1 йоттабайт = 1 Йбайт (ЙБ) = 1024 Збайт = 2^{80} = 1 208 925 819 614 629 174706 176.

Последовательность нескольких бит или байт часто называют *полем данных*. В особых случаях используется термин «*слово*» и его производные (но не в математическом смысле).

Биты или в числе, или в слове, или в поле, или и т. п. объектах нумеруются справа налево, начиная с нулевого разряда.

Слово состоит из двух байт.

<table>
<tr><td>↓</td><td colspan="8">старший бит</td><td colspan="8">младший бит</td><td>↓</td></tr>
<tr><td>бит</td><td>бит</td><td>бит</td><td>бит</td><td>бит</td><td>бит</td><td>бит</td><td>бит</td><td>бит</td><td>бит</td><td>бит</td><td>бит</td><td>бит</td><td>бит</td><td>бит</td><td>бит</td></tr>
<tr><td colspan="8">байт</td><td colspan="8">байт</td></tr>
<tr><td colspan="16">слово</td></tr>
</table>

Таким образом, производные термины от термина «слово»: *полуслово* – состоит из одного байта. *Двойное слово* – состоит из 4 байт. *Расширенное слово* – состоит из 8 байт.

Слово переменной длины – это некоторое слово, состоящее из определенного количества байт. Например, слово длиной 10 байт состоит из 10 байт.

Целые числа обозначаются термином «числа с фиксированной запятой». Чаще всего *числа с фиксированной запятой* имеют формат слова или полуслова. Таким образом, формат предопределяет максимальное значение числа.

Нецелые числа обозначаются термином «числа с плавающей запятой». Чаще всего *числа с плавающей запятой* имеют формат двойного слова или расширенного слова.

Поля переменной длины могут иметь любой размер от 0 до 256 байт, но обязательно равный целому числу байт.

Для представления чисел используются следующие форматы: *упакованный* и *распакованный*.

Рассмотрим эти форматы.

Упакованный формат. В нем для каждой десятичной цифры отводится по 4 двоичных разряда (бита) – *полубайта* (более понятным языком – половина байта или 4 бита). При этом знак числа кодируется в крайнем правом полубайте числа.

Приведем графическое изображение *структуры поля упакованного формата*:

бит	бит	бит	бит	бит	бит	бит	бит	бит	бит	бит	бит	бит	бит	бит	бит	…	бит	бит	бит	бит	бит	бит	бит	бит
цифра				цифра				цифра				цифра					цифра				знак числа			
полбайта				полбайта				полбайта				полбайта					полбайта				полбайта			
байт								байт									байт							

Распакованный формат. В нем для каждой десятичной цифры отводится по целому байту. При этом старшие полубайты (говорят зоны) каждого байта (кроме самого младшего) в компьютере заполняются кодом (ранее – «0011», а теперь «1111»), а в младших (левых) полубайтах обычным образом кодируются десятичные цифры. Старший полубайт (зона) самого младшего (правого) байта используется для кодирования знака числа.

Приведем графическое изображение *структуры поля распакованного формата*:

бит	бит	бит	бит	бит	бит	бит	бит	бит	бит	бит	бит	бит	бит	бит	бит	…	бит	бит	бит	бит	бит	бит	бит	бит	бит	бит	бит	бит	бит	бит	бит	бит
зона				цифра				зона				цифра					зона				цифра				знак				цифра			
полбайта				полбайта				полбайта				полбайта					полбайта				полбайта				полбайта				полбайта			
байт								байт									байт								байт							

Представим пример для следующего числа:

$-283_{(10)} = -0010\ 1000\ 0011_{(2)}$,

где: «0010» кодирует цифру 2, «1000» – цифру 8, «0011» – цифру 3.

Упакованный формат числа:

0	0	1	0	1	0	0	0	0	0	1	1	1	1	0	1
цифра				цифра				цифра				знак			
полубайт				полубайт				полубайт				полубайт			
байт								байт							

Распакованный формат числа:

1	1	1	1	0	0	1	0	1	1	1	1	1	0	0	0	1	1	0	1	0	0	1	1
зона				цифра				зона				цифра				знак				цифра			
полубайт				полубайт				полубайт				полубайт				полубайт				полубайт			
байт								байт								байт							

Следует отметить, что в литературе для *распакованного формата* используется еще и термин «*зонный формат*».

2.5. Представление текстовой информации

Текстовые данные состоят из символов – букв, цифр, знаков препинания и т. д., которые человек различает по начертанию. Однако для представления текстовой информации в компьютере такой метод неудобен, а для компьютерной обработки текстов и совсем неприемлем.

Поскольку текст изначально дискретен, т. е. состоит из отдельных символов, то для представления текстовой информации в компьютере используется иной способ: все символы *кодируются* числами. Таким образом, текст представляется в виде набора чисел – *кодов символов*, составляющих его. При выводе текста на монитор или принтер необходимо восстановить изображения всех символов, составляющих данный текст. Для этого используются кодовые таблицы символов, в которых каждому коду символа ставится в соответствие изображение символа.

Все кодовые таблицы, используемые на любых компьютерах и для любых операционных систем, соответствуют международным стандартам кодирования символов. На заре компьютерной эры США были абсолютным лидером в этой области и в силу этой причины стандарты разрабатывались Американским национальным институтом стандартизации (ANSI). Впоследствии для разработки и принятия стандартов была создана Международная организация стандартизации (ISO).

В программировании наиболее часто используются однобайтовые кодировки (код каждого символа занимает ровно 1 байт или 8 бит). При этом общее количество различаемых символов составляет $2^8 = 256$, а коды символов имеют значения от 0 до 255.

Информационный объем блока информации называется количеством бит, байт или производных единиц (килобайт, мегабайт), необходимых для записи этого блока путем заранее оговоренного способа двоичного кодирования.

Пример. Оценить в байтах объем текстовой информации в словаре из 740 страниц, если на одной странице размещается в среднем 60 строк по 80 символов (включая пробелы). Будем считать, что при записи используется кодировка «один символ – один байт». Количество символов во всем словаре равно $80 \cdot 60 \cdot 740 = 3552000$. *Следовательно, объем в байтах равен:*

$$3552000 \text{ байт} = 3468{,}75 \text{ Кбайт} \approx 3{,}39 \text{ Мбайт}.$$

Основой для компьютерных стандартов кодирования послужил ASCII (American Standard Code for Information Interchange) – американский стандартный код для обмена информацией, разработанный в 1960-х гг. и применяемый в США для любых видов передачи информации, в том числе и некомпьютерных (телеграф, факсимильная связь и т. д.). В нем используется 7-битовое кодирование: общее количество символов составляет $2^7 = 128$, из них первые 32 символа – управляющие, а остальные – «изобража-

емые», т. е. имеющие графическое изображение. Управляющие символы должны восприниматься устройством вывода текста как команды, например:

Код	Действие	Английское название
7	Подача стандартного звукового сигнала	Beep
8	Удаление предыдущего символа	Back Space (BS)
13	Перевод строки	Line Feed (LF)
26	Признак «Конец текстового файла»	End Of File (EOF)
27	Отмена предыдущего ввода	Escape (Esc)

К изображаемым символам в ASCII относятся буквы английского алфавита (прописные и строчные), цифры, знаки препинания и арифметических операций, скобки и некоторые специальные символы. Хотя в ASCII символы кодируются 7 битами, в памяти компьютера под каждый символ отводится ровно 1 байт, при этом код символа помещается в младшие биты, а старший бит не используется. Главный недостаток стандарта ASCII заключается в том, что он рассчитан на передачу только английского текста. Со временем возникла необходимость кодирования и неанглийских букв. Во многих странах для этого стали разрабатывать расширения ASCII-кодировки, в которых применялись однобайтовые коды символов; при этом первые 128 символов кодовой таблицы совпадали с кодировкой ASCII, а остальные (со 128-го по 255-й) использовались для кодирования букв национального алфавита, символов национальной валюты и т. п. Из-за несогласованности этих разработок для многих языков было создано по нескольку вариантов кодовых таблиц (например, для русского языка их около десятка!).

Таблица 2

Кодировка текстовой информации в стандарте ASCII

[	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0		☺ 1	☻ 2	♥ 3	♦ 4	♣ 5	♠ 6	• 7	8	○ 9	10	11	12	13	14	15
1	► 16	◄ 17	18	! 19	20	21	_ 22	23	↑ 24	↓ 25	→ 26	← 27	28	↔ 29	▲ 30	▼ 31
2	32	! 33	" 34	# 35	$ 36	% 37	& 38	' 39	(40	) 41	* 42	+ 43	, 44	- 45	. 46	/ 47
3	0 48	1 49	2 50	3 51	4 52	5 53	6 54	7 55	8 56	9 57	: 58	; 59	< 60	= 61	> 62	? 63
4	@ 64	A 65	B 66	C 67	D 68	E 69	F 70	G 71	H 72	I 73	J 74	K 75	L 76	M 77	N 78	O 79
5	P 80	Q 81	R 82	S 83	T 84	U 85	V 86	W 87	X 88	Y 89	Z 90	[91	\ 92	] 93	^ 94	_ 95
6	` 96	a 97	b 98	c 99	d 100	e 101	f 102	g 103	h 104	i 105	j 106	k 107	l 108	m 109	n 110	o 111
7	p 112	q 113	r 114	s 115	t 116	u 117	v 118	w 119	x 120	y 121	z 122	{ 123	| 124	} 125	~ 126	□ 127
8	Ç 128	ü 129	é 130	â 131	ä 132	à 133	å 134	ç 135	ê 136	ë 137	è 138	ï 139	î 140	ì 141	Ä 142	Å 143
9	É 144	æ 145	Æ 146	ô 147	ö 148	ò 149	û 150	ù 151	ÿ 152	Ö 153	Ü 154	¢ 155	£ 156	¥ 157	₧ 158	ƒ 159
A	160	í 161	ó 162	ú 163	ñ 164	Ñ 165	ª 166	º 167	¿ 168	⌐ 169	¬ 170	½ 171	¼ 172	¡ 173	« 174	» 175
B	░ 176	▒ 177	▓ 178	│ 179	┤ 180	╡ 181	╢ 182	╖ 183	╕ 184	╣ 185	║ 186	╗ 187	╝ 188	╜ 189	╛ 190	┐ 191
C	└ 192	┴ 193	┬ 194	├ 195	─ 196	┼ 197	╞ 198	╟ 199	╚ 200	╔ 201	╩ 202	╦ 203	╠ 204	═ 205	╬ 206	╧ 207
D	╨ 208	╤ 209	╥ 210	╙ 211	╘ 212	╒ 213	╓ 214	╫ 215	216	┘ 217	┌ 218	▉ 219	▄ 220	▌ 221	▐ 222	▀ 223
E	α 224	ß 225	Γ 226	π 227	Σ 228	σ 229	µ 230	τ 231	Φ 232	Θ 233	Ω 234	δ 235	∞ 236	φ 237	ε 238	∩ 239
F	≡ 240	± 241	≥ 242	≤ 243	⌠ 244	⌡ 245	÷ 246	≈ 247	° 248	∙ 249	· 250	√ 251	ⁿ 252	² 253	■ 254	□ 255

Кодировка символов, предложенная IBM (соответствует ASCII - кодировке)

В частности, все службы электронной почты и новостей Рунета работают в этой кодировке. Что касается Веба, то здесь ситуация сложнее. Дело в том, что более 90 % клиентских компьютеров в Интернете работают под управлением операционной системы Windows разных версий. Windows использует собственную кодировку русских букв, которую принято называть по номеру кодовой страницы Windows-1251 или CP-1251. Поскольку текстовые редакторы и средства разработки HTML-страниц

в Windows работают в этой кодировке, то абсолютное большинство веб-документов Рунета хранится в кодировке Windows-1251.

Кодировка символов, предложенная IBM, соответствует ASCII-кодировке.

Позднее применение кодовых таблиц было упорядочено: каждой кодовой таблице присвоили особое название и номер. Указав кодовую таблицу, автоматически выбирают и язык, которым можно пользоваться в дополнение к английскому, а точнее, выбирается интерпретация символов с кодами более 127.

Для русского языка наиболее распространенными являются однобайтовые кодовые таблицы CP-866 (Code Page), Windows-1251 и КОИ-8 (*КОИ – Код Обмена Информацией – 8-разрядный код*). В них первые 128 символов совпадают с ASCII-кодировкой, а русские буквы размещены во второй части таблицы, однако, коды русских букв в этих кодировках различны.

Сравним, например, кодировки КОИ-8 и Windows-1251, вторые половины которых приведены в Таблицах 3 и 4 соответственно.

Таблица 3

Кодировка КОИ-8,
ориентированная на обмен сообщениями в Интернете

─ 128	│ 129	┌ 130	┐ 131	└ 132	┘ 133	├ 134	┤ 135	┬ 136	┴ 137	┼ 138	▀ 139	▄ 140	█ 141	▌ 142	▐ 143
░ 144	▒ 145	▓ 146	⌠ 147	■ 148	∙ 149	√ 150	≈ 151	≤ 152	≥ 153	nbsp 154	⌡ 155	° 156	² 157	· 158	÷ 159
═ 160	║ 161	╒ 162	ё 163	╓ 164	╔ 165	╕ 166	╖ 167	╗ 168	╘ 169	╙ 170	╚ 171	╛ 172	╜ 173	╝ 174	╞ 175
╟ 176	╠ 177	╡ 178	Ё 179	╢ 180	╣ 181	╤ 182	╥ 183	╦ 184	╧ 185	╨ 186	╩ 187	╪ 188	╫ 189	╬ 190	© 191
ю 192	а 193	б 194	ц 195	д 196	е 197	ф 198	г 199	х 200	и 201	й 202	к 203	л 204	м 205	н 206	о 207
п 208	я 209	р 210	с 211	т 212	у 213	ж 214	в 215	ь 216	ы 217	з 218	ш 219	э 220	щ 221	ч 222	ъ 223
Ю 224	А 225	Б 226	Ц 227	Д 228	Е 229	Ф 230	Г 231	Х 232	И 233	Й 234	К 235	Л 236	М 237	Н 238	О 239
П 240	Я 241	Р 242	С 243	Т 244	У 245	Ж 246	В 247	Ь 248	Ы 249	З 250	Ш 251	Э 252	Щ 253	Ч 254	Ъ 255

Несовпадение кодовых таблиц приводит к ряду неприятных эффектов, например, так как один и тот же текст имеет различное представление в разных кодировках, то текст, набранный в одной кодировке, будет нечитабельным в другой.

Таблица 4

Кодировка Windows-1251

Character Map

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□
1	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□
2		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	□
8	Ђ	Ѓ	‚	ѓ	„	…	†	‡	€	‰	Љ	‹	Њ	Ќ	Ћ	Џ
9	ђ	‘	’	“	”	•	–	—	□	™	љ	›	њ	ќ	ћ	џ
A		Ў	ў	Ј	¤	Ґ	¦	§	Ё	©	Є	«	¬	-	®	Ї
B	°	±	І	і	ґ	µ	¶	·	ё	№	є	»	ј	Ѕ	ѕ	ї
C	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П
D	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я
E	а	б	в	г	д	е	ж	з	и	й	к	л	м	н	о	п
F	р	с	т	у	ф	х	ц	ч	ш	щ	ъ	ы	ь	э	ю	я

Пример. Вот так будет выглядеть десятичный код слова «Диск» в разных кодировках:

КОИ-8	**228 201 211 203**
Windows-1251	**196 232 241 234**
CP-866	**132 168 225 170**

Однобайтовые кодировки обладают одним серьезным ограничением: качество различных кодов символов в этих кодировках минимальное, что не позволяет одновременно использовать несколько языков. Для устранения этого ограничения в 1993 г. был разработан новый стандарт кодирования символов Unicode, который позволил использовать в текстах самые разнообразные символы языков мира.

В Unicode на кодирование символов отводится 31 бит. Первые 128 символов (коды 0–127) совпадают с таблицей ASCII. Далее размещены основные алфавиты современных языков: они полностью умещаются в первой части таблицы, их коды не превосходят 65536 (2^{16}). В целом, стандарт Unicode описывает алфавиты всех известных, в том числе и «мертвых», языков. Для языков, имеющих несколько алфавитов (например, японский и индийский), закодированы все варианты. В кодировку Unicode внесены дополнительно все математические и иные научные символьные обозначения, в том числе даже некоторые «придуманные» языки (например, письменности эльфов по Р. Р. Толкиену). Потенциальная информационная емкость 31-битового Unicode столь велика, что используется менее одной тысячной части возможных кодов символов.

Однако в современных компьютерах и операционных системах используется укороченная, 16-битовая версия Unicode, в которую входят все современные алфавиты; эта часть Unicode называется базовой многоязыковой страницей (Basic Multilingual Plane, BMP). Unicode – это 16-ти разрядная система кодирования (65536 символов). Она охватывает символы всех языков (включая языки, использующие иероглифы). Стандарт Unicode 3.2 поддерживает языки народов России с дополнительными кириллическими буквами: алтайский, башкирский, бурятский, долганский, калмыцкий, коми, корякский, марийский, нанайский, ненецкий,

осетинский, саамский (без указания долготы гласных), татарский, тувинский, удмуртский, хакасский, хантыйский, чувашский, эвенкийский, эвенский, якутский, кавказские языки с буквой «палочка».

Таблица 5

Дополнительные буквы языков народов России, поддерживаемые стандартом Unicode 3.2

0	040	041	042	043	044	045	046	047	048	049	04A	04B	04C	04D	04E	04F
0	Ѐ	А	Р	а	р	ѐ	Ҁ	Ѱ	ҁ	Ґ	Қ	Ҵ	Ӏ	Ӑ	Ӡ	Ӱ
1	Ё	Б	С	б	с	ё	ҁ	ѱ	ҁ	ґ	қ	ҵ	Ӂ	ӑ	ӡ	ӱ
2	Ђ	В	Т	в	т	ђ	Ъ	Ѳ	҂	Ғ	Ҥ	Ҷ	ӂ	Ӓ	Ӣ	Ӳ
3	Ѓ	Г	У	г	у	ѓ	ъ	ѳ	◌҃	ғ	ҥ	ҷ	Ҍ	ӓ	ӣ	ӳ
4	Є	Д	Ф	д	ф	є	Ѥ	Ѵ	◌҄	Ҕ	Ҥ	Ҹ	ҍ	Ӕ	Ӥ	Ӵ
5	Ѕ	Е	Х	е	х	ѕ	ѥ	ѵ	◌҅	ҕ	ҥ	ҹ	Ҏ	ӕ	ӥ	ӵ
6	І	Ж	Ц	ж	ц	і	Ѧ	Ѷ	◌҆	Ж	Ҧ	Ҍ	ҏ	Ӗ	Ӧ	
7	Ї	З	Ч	з	ч	ї	ѧ	ѷ		ж	ҧ	ҍ	Ӈ	ӗ	ӧ	
8	Ј	И	Ш	и	ш	ј	Ѩ	Ꙋ	◌҈	Ҙ	Ҩ	Ӌ	ӈ	Ә	Ө	Ы
9	Љ	Й	Щ	й	щ	љ	ѩ	ꙋ	◌҉	ҙ	ҩ	ӌ	Ӊ	ә	ө	ы
A	Њ	К	Ъ	к	ъ	њ	Ѫ	Ҋ	Й	Ҝ	Ҫ	Ҥ	ӊ	Ӛ	Ө	
B	Ћ	Л	Ы	л	ы	ћ	ѫ	ҋ	й	ҝ	ҫ	ҥ	Ҋ	ӛ	ө	
C	Ќ	М	Ь	м	ь	ќ	Ѭ	Ҍ	Ъ	Ҟ	Ҭ	Ҿ	ҋ	Ж	Э	
D	Ѝ	Н	Э	н	э	ѝ	ѭ	ҍ	ъ	ҟ	ҭ	ҿ	Ӎ	ж	э	
E	Ў	О	Ю	о	ю	ў	Ѯ	Ҏ	Ҁ	Ҡ	Ү	Ӏ	ӎ	Ӟ	Ӯ	
F	Џ	П	Я	п	я	џ	ѯ	ҏ	ҁ	ҡ	ү	ӏ		ӟ	ӯ	

В UNIX-подобных операционных системах, где работа с Unicode-текстами невозможна из-за особенностей архитектуры, используются особые формы этого стандарта, которые называются UTF (Unicode Transformation Format). В них символы кодируются переменным количеством бит.

Например, в UTF-8 коды символов занимают от 1 до 6 байт.

С точки зрения компьютера текст состоит из отдельных символов. К числу символов относятся не только буквы (заглавные или строчные, латинские или русские), но и цифры, знаки препинания, спецсимволы типа «=», «(», «&» и т. п. и даже пробелы между словами (пустое место в тексте тоже должно иметь свое обозначение). При нажатии клавиши клавиатуры формируется сигнал в виде двоичного числа, которое хранится в кодовой таблице. Кодовая таблица – это внутреннее представление символов для компьютера.

Каждый символ хранится в виде двоичного кода, который является номером символа. Можно сказать, что компьютер имеет собственный алфавит, где весь набор символов строго упорядочен.

В компьютере хранится не начертание буквы, а ее номер. Именно по этому номеру воспроизводится вид символа на мониторе или на бумаге.

Интересно, что каждый символ текста имеет свой числовой код, но не каждому коду соответствует отображаемый на экране символ. Речь идет об управляющих символах, величина которых меньше шестнадцатеричного числа 20 (т. е. 32 в десятичной системе счисления). При получении этих кодов внешние устройства не изображают какой-либо символ, а выполняют те или иные управляющие действия. Так, код 07 вызывает подачу стандартного звукового сигнала, а код 0C – очистку экрана. Особую роль играют коды 0A (перевод строки, обозначаемый часто LF) и 0D (возврат каретки – CR). Первый вызывает перемещение в следующую строку без изменения позиции, а второй – на начало текущей строки. Таким образом, для перехода на начало новой строки требуются оба кода, и в любом тексте эта «неразлучная пара» кодов хранится после каждой строки.

ГЛАВА 3
АРХИТЕКТУРА КОМПЬЮТЕРА

3.1. Классическая архитектура электронно-вычислительной машины (архитектура машины фон Неймана)

Первые попытки создания ЭВМ предпринимались в Англии в конце 1930-х – начале 1940-х гг. в связи с необходимостью автоматизировать процессы взлома шифров, используемых в военных целях в Германии. Идеи построения ЭВМ высказывались еще в 1941 г. Дж. Нейманом, Г. Голдстайном и А. Берксом. Работа над первой электронной вычислительной машиной в США началась в середине 1943 г. по заказу артиллерийского управления. Руководителями были Д. Моучли и Д. Эккерт, начиная с 1944 г. к работе был привлечен уже упоминавшийся крупнейший американский математик Дж. Нейман. Постройка машины была завершена в конце 1945 г. Машина получила название ЭНИАК (*ENIAC – Electronic Numerical Integrator and Computer* – электронный цифровой интегратор и вычислитель). В устройстве этой первой ЭВМ использовалось 17 000 электронных ламп, крайне ненадежных в работе, а общая занимаемая площадь составляла 300 м^2.

Итак, американский математик Джон фон Нейман выдвинул основополагающие принципы логического устройства электронно-вычислительной машины (ЭВМ) и предложил структуру, а точнее в 1946 г. трое ученых – Артур Беркс (*Arthur Burks*), Герман Голдстайн (*Herman Goldstine*) и Джон фон Нейман (*John von Neumann*) – опубликовали статью «Предварительное рассмотрение логического конструирования электронного вычислительного устройства». В статье обосновывалось использование двоичной системы для представления

данных в ЭВМ (преимущественно для технической реализации, простота выполнения арифметических и логических операций – до этого машины хранили данные в десятичном виде), выдвигалась идея использования общей памяти для программы и данных. Имя фон Неймана было достаточно широко известно в науке того времени, что отодвинуло на второй план его соавторов, и данные идеи получили название «принципы фон Неймана».

Но первый компьютер – Electronic Delay Storage Automatic Calculator (EDSAC), построенный по принципам фон Неймана, был сделан английским исследователем М. Уилксом в 1949 г. Арифметические операции этот компьютер выполнял со скоростью: для сложения – $70 \cdot 10^{-6}$ с (время, измеряемое в микросекундах) и умножения – $8{,}5 \cdot 10^{-3}$ с (время, измеряемое в миллисекундах).

Итак, подробнее остановимся на принципах фон Неймана:

1. *Принцип двоичного кодирования.* Для представления данных и команд используется двоичная система кодирования.

Данные, программы разделяются на единицы (элементы), называемые словами. Слово обрабатывается как единое целое – машинный элемент информации.

2. *Принцип однородности памяти.* Программы (команды) и данные хранятся в одной и той же памяти и различаются по способу использования, но не по способу кодирования.

Благодаря такому «однообразию» слов оказывается возможным выполнять одни и те же действия (операции) над словами различной природы – данными и командами.

3. *Принцип адресуемости памяти.* Структурно основная память состоит из пронумерованных ячеек. В них размещаются слова. Ячейки идентифицируются номерами, называемыми адресами слов.

Процессору в произвольный момент времени доступна любая ячейка. Таким образом, единственным средством для обозначения данных и команд в компьютере являются адреса.

4. *Принцип последовательного программного управления.* Все команды располагаются в памяти и выполняются последовательно, одна после завершения другой, в последовательности, определяемой алгоритмом. Алгоритм, представленный в терминах машинных команд, называется программой.

Следовательно, выполнение вычислений, предписанных алгоритмом, сводится к последовательному выполнению команд в порядке, однозначно определяемом программой.

5. *Возможность условного перехода в процессе выполнения программы.* Несмотря на то, что команды выполняются последовательно, в программах можно реализовать возможность перехода к любому участку кода.

6. *Принцип жесткости архитектуры.* Неизменяемость в процессе работы архитектуры и списка команд.

Компьютеры, построенные на этих принципах, относят к типу «фон-неймановских». Таким образом, *архитектура фон Неймана* (англ. *von Neumann architecture*) – широко известный принцип совместного хранения программ и данных в памяти компьютера. Вычислительные системы такого рода часто обозначают термином «машина фон Неймана». Однако соответствие этих понятий не всегда однозначно. В общем случае, когда говорят об архитектуре фон Неймана, подразумевают физическое отделение процессорного модуля от устройств хранения программ и данных.

Идея хранения компьютерных программ в общей памяти оказалась прогрессивной. В это же время использовались архитектуры, основанные на наборах исполняемых инструкций, и представление вычислительного процесса как процесса выпол-

нения инструкций, записанных в программе, давало чрезвычайную гибкость вычислительных систем в плане обработки данных. Один и тот же подход к рассмотрению данных и инструкций, а также их хранение в общей памяти обеспечили легкость задачи изменения самих программ.

Основными элементами структуры «машины фон Неймана» являются:

– устройство управления (УУ);
– арифметико-логическое устройство (АЛУ);
– оперативно-запоминающее устройство (ОЗУ);
– внешнее запоминающее устройство (ВЗУ);
– устройства ввода и вывода.

УУ и АЛУ в компьютерах объединены в один блок – процессор.

Запоминающее устройство (ЗУ) состоит из двух разных частей: ОЗУ и ВЗУ. ОЗУ хранит данные и программы, необходимые в текущий момент времени для обработки информации. ВЗУ хранит все данные и программы, необходимые пользователю компьютера.

Устройство ввода позволяет вводить новые данные и программы в компьютер, а устройство вывода позволяет пользователю сохранить результаты работы с компьютером в удобном для него виде.

Разработанные фон Нейманом основы архитектуры оказались настолько фундаментальными, что получили название «фон-неймановской архитектуры», которая используется и сегодня.

На рисунке 3 приведена архитектура компьютера на основе принципов фон Неймана.

Как работает машина фон Неймана? Программы и данные вводятся в память из устройства ввода через арифметико-ло-

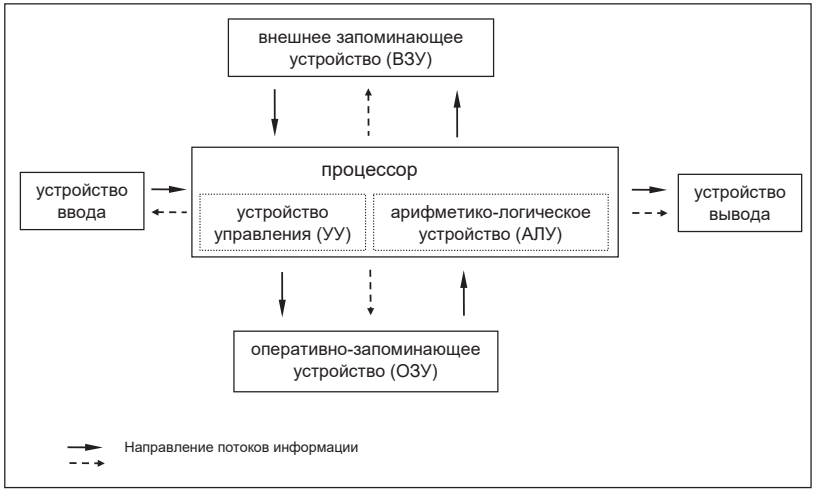


Рис. 3. Архитектура компьютера на основе принципов фон Неймана

гическое устройство. Все команды программы записываются в соседние ячейки памяти ОЗУ, а данные для обработки могут содержаться в произвольных ячейках ОЗУ. У любой программы последняя команда должна быть командой завершения работы.

Команда состоит из указания, какую операцию следует выполнить, и адресов ячеек памяти, где хранятся данные, над которыми следует выполнить указанную операцию, а также адреса ячейки, куда следует записать результат (если его требуется сохранить в ЗУ).

Арифметико-логическое устройство выполняет указанные командами операции над указанными данными.

Из арифметико-логического устройства результаты выводятся в память или устройство вывода. Принципиальное различие между ЗУ и устройством вывода заключается в том, что в ЗУ данные хранятся в виде, удобном для обработки компьютером, а на устройства вывода (принтер, монитор и др.) поступают так, как удобно человеку.

Устройство управления управляет всеми частями компьютера. От управляющего устройства на другие устройства поступают сигналы «что делать», а от других устройств УУ получает информацию об их состоянии.

3.2. Совершенствование и развитие архитектуры компьютера

Быстродействие процессоров оказалось более существенным по сравнению с другими элементами приведенной архитектуры. К тому же хотелось реализовать функции взаимодействия элементов, минуя процессор и разгрузив его тем самым от рутинных операций.

На помощь пришла шинная архитектура с интеллектуальными контроллерами, которые рассматривались как специализированные процессоры.

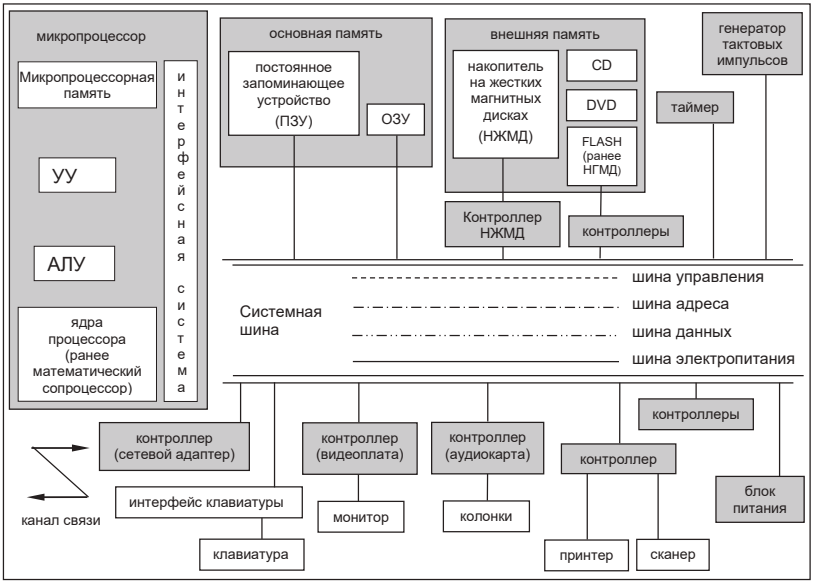


Рис. 4. Шинная архитектура компьютера

Шинная архитектура компьютера представлена на рисунке 4.

Существующую схему легко дополнять новыми устройствами. Это свойство схемы называется открытостью архитектуры компьютера. Для пользователя открытая архитектура дает возможность свободно выбирать состав устройств компьютера – конфигурировать его в зависимости от круга решаемых задач.

Основные элементы системного блока компьютера: системная шина (состоящая из шины управления, шины адреса, шины данных, шины электропитания); многочисленные контроллеры (среди которых контроллер прямого доступа к памяти, сопроцессор ввода/вывода, контроллер прерываний и др.); микропроцессор; основная память (состоящая из ПЗУ и ОЗУ); внешняя память; блок питания; генератор тактовых импульсов; таймер.

ГЛАВА 4
ОБЩЕЕ УСТРОЙСТВО (ОСНОВНЫЕ КОМПОНЕНТЫ) И ОСНОВЫ РАБОТЫ КОМПЬЮТЕРА

Устройство компьютера называют его конфигурацией. Современные компьютеры имеют блочную конструкцию. Аппаратную конфигурацию, необходимую для выполнения конкретных видов работ, можно собрать из готовых блоков и гибко изменять по мере необходимости.

4.1. Основные принципы работы компьютера

Компьютер – это, прежде всего, техническое средство преобразования информации, в основу работы которого заложены принципы обработки электрических сигналов. Входная информация, представленная различными физическими процессами как электрической, так и неэлектрической природы (буквами, цифрами, звуковыми сигналами и т. д.), преобразуется в электрические сигналы. Электрические сигналы обрабатываются в блоке обработки. А с помощью преобразователя выходных сигналов обработанные сигналы преобразуются в неэлектрические сигналы (изображения на экране, удобные для непосредственного восприятия человеком).

С позиции функционального назначения компьютер – это система, в основу которой положена «машина фон Неймана», состоящая из четырех основных устройств, выполняющих определенные функции: запоминающего устройства (ЗУ) или памяти, арифметико-логического устройства (АЛУ), устройства управления (УУ) и устройств ввода/вывода (УВВ).

Рассмотрим их роль и назначение.

Запоминающее устройство или память предназначается для хранения информации и команд программы в ЭВМ. Информация, которая хранится в памяти, представляет собой закодированные с помощью 0 и 1 числа, символы, слова, команды, адреса и т. д.

Под записью числа в память понимают размещение этого числа в ячейке по указанному адресу и хранение его там до выборки по команде программы. Предыдущая информация, находившаяся в данной ячейке, перезаписывается. При программировании, например, на языке Си, адрес ячейки связан с именем переменной, которое представляется комбинацией букв и цифр, выбираемых программистом.

Под считыванием числа из памяти понимают выборку числа из ячейки с указанным адресом. При этом копия числа передается из памяти в требуемое устройство, а само число остается в ячейке.

Пересылка информации означает, что информация читается из одной ячейки и записывается в другую.

Адрес ячейки формируется в устройстве управления (УУ), затем поступает в устройство выборки адреса, которое открывает информационный канал и подключает нужную ячейку.

Числа, символы, команды хранятся в памяти на равноправных началах и имеют один и тот же формат. Ни для памяти, ни для самого компьютера не имеет значения тип данных. Типы различаются только при обработке данных программой. Длину, или разрядность, ячейки определяет количество двоичных разрядов (бит). В современных компьютерах длина ячейки кратна 8 битам и измеряется в байтах. Минимальная длина ячейки, для которой можно сформировать адрес, равна 1 байту, состоящему из 8 бит.

Для характеристики памяти используются следующие параметры: *емкость памяти* – максимальное количество хранимой

информации в байтах; *быстродействие памяти* – время обращения к памяти, определяемое временем считывания или временем записи информации.

Основная память (ОП) предназначена для хранения и оперативного обмена информацией с различными блоками компьютера. ОП содержит два вида запоминающих устройств: постоянное запоминающее устройство (ПЗУ) и оперативное запоминающее устройство (ОЗУ) с произвольным доступом к ячейкам памяти.

ПЗУ служит для хранения неизменяемой (постоянной) программной и справочной информации, позволяет оперативно только считывать хранящуюся в нем информацию (хотя теперь уже возможно изменять информацию в ПЗУ).

ОЗУ предназначено для оперативной записи, хранения и считывания информации (программ и данных), непосредственно участвующей в информационно-вычислительном процессе, выполняемом компьютером 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 компьютера (энергозависимость).

Внешняя память относится к внешним устройствам компьютера и используется для долговременного хранения любой информации, которая, возможно, когда-либо потребуется для решения задач. В частности, во внешней памяти (иногда ее называют встроенной) хранится все программное обеспечение компьютера. В качестве внешней памяти могут использоваться разнообразные виды запоминающих устройств: накопители на жестких магнитных дисках (НЖМД) (в том числе и НЖМД, установленные во внешние корпуса и подключаемые к порту USB, –

так называемые внешние (съемные) USB-диски); CD- и DVD-приводы компакт-дисков (оптических); USB-flash-память; твердотельные накопители SSD (*solid-state drive*) и многие другие.

Назначение этих накопителей – хранение больших объемов информации, запись и выдача хранимой информации по запросу в ОЗУ. Виды внешней памяти различаются конструктивно и по физическим принципам хранения данных, объемами хранимой информации и временем поиска, записи и считывания информации.

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 компьютера;

– *шину питания*, имеющую провода и схемы сопряжения для подключения блоков компьютера 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 кодировочных таблиц (типа ASCII-кодов или Unicode и др.).

Генератор тактовых импульсов генерирует последовательность электрических импульсов; частота генерируемых импульсов определяет, прежде всего, тактовую частоту процессора.

Промежуток времени между соседними импульсами определяет время одного такта работы машины или просто *такт работы машины*.

Частота генератора тактовых импульсов является одной из основных характеристик компьютера и во многом определяет скорость его работы, ибо каждая операция в компьютере выполняется микропроцессором за определенное количество тактов.

Центральный процессор компьютера предназначен для управления работой всех его блоков и для выполнения арифметических и логических операций над информацией. Но на самом деле то, что часто называют процессором, правильно называть ***микропроцессором***. Более точное назначение процессора – это *автоматическое выполнение программы*. Другими словами, он является основным компонентом любого компьютера, системного блока.

Центральный процессор работает под управлением программы, находящейся в оперативной памяти. Данные и команды попадают в кэш не сразу из оперативной памяти, а через блок

предварительной выборки и декодирующий блок, осуществляющий преобразование данных и команд в двоичную форму, только после чего с ними может работать процессор.

Ключевыми компонентами микропроцессора являются устройство управления (УУ), арифметико-логическое устройство (АЛУ), математический сопроцессор, микропроцессорная память (МПП), интерфейсная система микропроцессора.

УУ формирует и подает во все блоки компьютера 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. От УУ зависят согласованность работы частей самого процессора и его связь с другими (внешними для него) устройствами.

Таким образом, устройство управления управляет всем ходом вычислительного и логического процесса в компьютере, т. е. выполняет функции «регулировщика движения» информации. УУ читает команду, расшифровывает ее и подключает необходимые цепи для ее выполнения. Считывание следующей команды происходит автоматически. Фактически УУ выполняет следующий цикл действий: формирование адреса очередной команды; чтение команды из памяти и ее расшифровка; выполнение команды.

АЛУ предназначено для выполнения всех арифметических и логических операций над числовой и символьной информацией. Все вычисления производятся в двоичной системе счисления.

Следует отметить, что любую арифметическую операцию можно реализовать с использованием операции сложения, а слож-

ная логическая задача раскладывается на более простые задачи, где достаточно анализировать только два уровня: *да* и *нет*.

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 с основным МП, но под управлением последнего.

Микропроцессорная память (МПП) служит для кратковременного хранения, записи и выдачи информации, непосредственно используемой в вычислениях в ближайшие такты работы процессора. МПП строится на регистрах и используется для обеспечения высокого быстродействия компьютера в целом, ибо основная память 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). В регистрах временно хранятся текущая команда, исходные, промежуточные и конечные данные (результат вычислений АЛУ). Разрядность всех регистров одинакова. *Кэш данных и команд* хранит часто используемые данные и команды. Обращение в кэш происходит намного быстрее, чем в оперативную память, поэтому чем он больше, тем лучше.

Интерфейсная система микропроцессора реализует сопряжение и связь с другими устройствами компьютера. 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-port – input/output port*) – аппаратура сопряжения, позволяющая подключить к микропроцессору другое устройство компьютера.

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

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 компьютера от сети продолжает работать.

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

Контроллер прямого доступа к памяти освобождает МП от прямого управления накопителями на магнитных дисках, что существенно повышает эффективное быстродействие компьютера. 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 компьютере контроллер прерываний.

Прерывание *– временная остановка выполнения одной программы в целях оперативного выполнения другой, в данный момент более важной (приоритетной) программы.*

Прерывания возникают при работе компьютера постоянно. Достаточно сказать, что 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.

Рассмотрим, как происходит взаимодействие карт расширения с остальными компонентами компьютера. Основную роль здесь играют прерывания (*IRQ – interrupt request*). Если какой-либо карте необходимо передать или получить данные, она выставляет выделенное ей прерывание в активное состояние, «сообщая», таким образом, процессору о своих «потребностях». Процессор, обнаружив активное прерывание, прерывает обработку текущих данных и выполняет обработку запроса карты расширения, после чего «возвращается» к прерванной задаче.

Всего имеется 32 прерывания (ранее было 16). Часть из них жестко предназначена определенным устройствам, остальные же могут использовать карты расширения. Современные карты расширения, как, впрочем, и многие аппаратные компоненты компьютера, способны разделять одну линию прерывания с другими устройствами, поэтому часто на одном прерывании находится несколько устройств (табл. 6).

Таблица 6

Описание прерываний

Номер прерывания	Использование картами	Назначение
0	–	Системный таймер
1	–	Используется контроллером клавиатуры
2		Дублирует прерывание 9
3	+	Обычно используется для порта COM2
4	+	Обычно используется для порта COM1
5	+	Свободно, часто используется звуковыми картами для совместимости с Sound Blaster Pro, может использоваться контроллером порта USB
6	–	Используется контроллером флоппи-дисковода
7	+	Обычно используется для порта LPT
8	–	Часы реального времени
9	+	Обычно используется системой расширенного управления питанием, также может использоваться контроллером порта USB
10	+	Свободно
11	+	Обычно используется видеокартой
12	+	Используется мышью, подключаемой к порту PS/2; при ее отсутствии может использоваться другими устройствами

Окончание табл. 6

13	–	Используется сопроцессором
14	+	Обычно используется для первичного канала IDE-контроллера
15	+	Обычно используется для вторичного канала IDE-контроллера

Еще один механизм, который может применяться картой расширения, называется прямым доступом к памяти (*DMA – direct memory access*). Он позволяет карте обмениваться данными с оперативной памятью напрямую, минуя процессор. Всего имеется 8 каналов, так же, как и в случае с прерываниями, один канал может использоваться несколькими устройствами (табл. 7).

Таблица 7

Каналы прямого доступа

Номер канала	Использование картами	Назначение
0	+	Свободно
1	+	Свободно, часто используется звуковыми картами для совместимости с Sound Blaster Pro
2	–	Используется контроллером флоппи-дисковода
3	+	Обычно используется для порта LPT, функционирующего в режиме ECP или ECP/EPP
4	–	Используется самим контроллером прямого доступа к памяти
5	+	Свободно
6	+	Свободно
7	+	Свободно

BIOS Setup (настройки базовой системы ввода/вывода) позволяют выбрать режимы функционирования шин, например, указать возможность одновременного обращения к PCI и ISA, вручную назначить прерывание и канал прямого доступа к памяти для карты, вставленной в определенный слот и т. д.

Чипсет материнской платы. Важную роль в работе компьютера играет набор системных микросхем материнской платы (иначе – чипсет). Это своего рода интеллектуальная «прокладка» между всеми компонентами.

В большинстве случаев чипсет состоит из двух основных микросхем: *системного контроллера (часто называемого северным мостом)* и *функционального контроллера (называемого южным мостом)*. Естественно, этими двумя микросхемами дело не ограничивается, и в чипсет входит еще множество более мелких составляющих. Просто эти две микросхемы – основа чипсета. Функциональный контроллер соединяется с системным посредством внутренней высокоскоростной шины.

К системному контроллеру через системную шину подключается процессор, а через шину памяти – оперативная память. Видеокарта подключается через свою собственную шину – AGP (или PCI, PCI-Express и др.).

Сам функциональный контроллер обеспечивает управление шинами типа PCI, работу жестких дисков, CD-приводов, портов ввода/вывода (в частности, клавиатуры и USB-устройств). Помимо этого, функциональный контроллер с помощью BIOS обеспечивает инициализацию и начальную загрузку компьютера после включения питания или перезагрузки.

В заключение можно отметить, что *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.

4.2. Программное управление компьютером

Структура и виды команд. Решение задач на компьютере реализуется программным способом, т. е. путем выполнения последовательно во времени отдельных операций над информацией, предусмотренных алгоритмом решения задачи.

Алгоритм *– это точно определенная последовательность действий, которые необходимо выполнить над исходной информацией, чтобы получить решение задачи.*

Алгоритм решения задачи, заданный в виде последовательности команд на языке вычислительной машины (в кодах машины), называется *машинной программой*.

Команда машинной программы (иначе, машинная команда) – это элементарная инструкция процессору, выполняемая им автоматически без каких-либо дополнительных указаний и пояснений.

Машинная команда состоит из двух частей: *операционной* и *адресной*.

Операционная часть команды *– это группа разрядов в команде, предназначенная для представления кода операции процессора.*

Адресная часть команды *– это группа разрядов в команде, в которых записываются коды адреса (адресов) ячеек памяти компьютера, предназначенных для оперативного хранения информации, или иных объектов, задействованных при выполнении команды.*

Часто эти адреса называются *адресами операндов*, т. е. чисел, участвующих в операции.

По количеству адресов, записываемых в команде, команды делятся на *безадресные*, *одно-*, *двух-* и *трехадресные, переменноадресные*.

Типовая структура трехадресной команды:

КОП	*ач1*	*ач2*	*ач3*

где: *КОП* – код операции; *ач1* и *ач2* – адреса ячеек (регистров), где расположены соответственно первое и второе числа, участвующие в операции; *ач3* – адрес ячейки (регистра), куда следует поместить число, полученное в результате выполнения операции.

Типовая структура двухадресной команды:

КОП	*ач1*	*ач2*

где: *КОП* – код операции; *ач1* – это обычно адрес ячейки (регистра), где хранится первое из чисел, участвующих в операции, и куда после завершения операции должен быть записан результат операции; *ач2* – обычно адрес ячейки (регистра), где хранится второе участвующее в операции число.

Типовая структура одноадресной команды:

КОП	*ач1*

где: *КОП* – код операции; *ач1* – в зависимости от модификации команды может обозначать либо адрес ячейки (регистра), где хранится одно из чисел, участвующих в операции, либо адрес ячейки (регистра), куда следует поместить число – результат операции.

Безадресная команда содержит только код операции, а информация для нее должна быть заранее помещена в определенные регистры процессора (безадресные команды могут использоваться только совместно с командами другой адресности).

Состав машинных команд. Современные компьютеры автоматически выполняют несколько сотен различных команд. Например, стандартный набор современных процессоров содержит около 240 машинных команд. Все машинные команды можно разделить на группы по видам выполняемых операций:

– операции пересылки информации внутри ЭВМ;
– арифметические операции над информацией;
– логические операции над информацией;
– операции обращения к внешним устройствам ЭВМ;
– операции передачи управления;
– обслуживающие и вспомогательные операции.

Пояснения требуют *операции передачи управления* (иначе – ветвления программы), которые служат для изменения естественного порядка выполнения команд. Бывают *операции безусловной передачи управления* и *операции условной передачи управления*.

Операции безусловной передачи управления требуют выполнения после данной команды, не следующей по порядку, а той, адрес которой в явном или неявном виде указан в адресной части.

Операции условной передачи управления требуют тоже передачи управления по адресу, указанному в адресной части команды, но только в том случае, если выполняется некоторое заранее оговоренное для этой команды условие. Это условие в явном или неявном виде указано в коде операции.

4.3. Основы работы процессора

Центральный процессор (CPU, от англ. central processing unit) – это основной рабочий блок компьютера, который выполняет арифметические и логические операции, заданные программой, управляет вычислительным процессом и координирует работу всех устройств компьютера.

В общем случае современный центральный процессор содержит в себе:

– устройство управления (УУ);
– микропроцессор (МП);

– кэш-память – очень быструю память, объемом около 8 Мбайт, возможно, состоящую из трех уровней;

– интерфейсную систему.

Интерфейсная система микропроцессора, прежде всего, включает в себя внутренний интерфейс МП, буферные запоминающие регистры и схемы управления портами ввода/вывода (ПВВ) и системной шиной. А также реализует сопряжение и связь с другими устройствами компьютера.

Микропроцессор. МП – это функционально законченное программно-управляемое устройство обработки информации.

Современные процессоры выполняются в виде *микропроцессоров, имеющих 2, 4 или уже 8 ядер*.

Одно из ядер – это *арифметико-логическое устройство*. АЛУ выполняет арифметические операции только над двоичной информацией с запятой, фиксированной после последнего разряда, т. е. только над целыми двоичными числами. Другое ядро – это *математический сопроцессор.* Иное ядро – это, возможно, *специализированное «графическое» ядро*, предназначенное для формирования изображения на экране (мониторе, дисплее).

В целом, микропроцессор выполняет следующие функции:

– чтение и дешифрацию команд из основной памяти;

– чтение данных из оперативной памяти и регистров адаптеров внешних устройств;

– прием и обработку запросов и команд от адаптеров на обслуживание внешних устройств (ВУ);

– обработку данных и их запись в оперативную память и регистры адаптеров внешних устройств (ВУ);

– выработку управляющих сигналов для всех прочих узлов и блоков компьютера.

Разрядность шины данных микропроцессора определяет разрядность компьютера в целом; разрядность шины адреса МП – его адресное пространство.

Адресное пространство – это максимальное количество ячеек основной памяти, которое может быть непосредственно адресовано микропроцессором.

Микропроцессорная память (МПП). МПП служит для кратковременного хранения, записи и выдачи информации, непосредственно используемой в вычислениях в ближайшие такты работы процессора.

МПП состоит из быстродействующих регистров с разрядностью не менее машинного слова и используется для обеспечения высокого быстродействия компьютера, поскольку основная память не всегда обеспечивает скорость записи, поиска и считывания информации, необходимую для эффективной работы быстродействующего микропроцессора. Таким образом, она характеризуется небольшой емкостью, но чрезвычайно высоким быстродействием (время обращения к МПП, т. е. время, необходимое на поиск, запись или считывание информации из этой памяти, измеряется наносекундами).

Регистры микропроцессора делятся на регистры *общего назначения* и *специальные*.

Специальные регистры применяются для хранения различных адресов (адреса команды, например), признаков результатов выполнения операций и режимов работы компьютера (регистр флагов, например) и др.

Регистры общего назначения являются универсальными и могут использоваться для хранения любой информации, но некоторые из них тоже должны быть обязательно задействованы при выполнении ряда процедур.

Устройство управления является частью микропроцессора. УУ формирует и подает во все блоки компьютера 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. От УУ зависит согласованность работы частей самого процессора и его связь с другими (внешними для него) устройствами.

Следовательно, устройство управления управляет всем ходом вычислительного и логического процесса в компьютере. УУ читает команду, расшифровывает ее и подключает необходимые цепи для ее выполнения. Считывание следующей команды происходит автоматически. Фактически УУ выполняет следующий цикл действий: формирование адреса очередной команды; чтение команды из памяти и ее расшифровка; выполнение команды.

Таким образом, УУ является функционально наиболее сложным устройством системного блока. Оно вырабатывает управляющие сигналы, поступающие по кодовым шинам инструкций во все блоки компьютера.

Упрощенная функциональная схема УУ показана на рис. 5.

Рассмотрим основные элементы данной функциональной схемы.

Регистр команд – запоминающий регистр, в котором хранится код команды: код выполняемой операции и адреса операндов, участвующих в операции. Регистр команд расположен в интерфейсной части МП, в блоке регистров команд.

Дешифратор операций – логический блок, выбирающий в соответствии с поступающим из регистра команд кодом операции (КОП) один из множества имеющихся у него выходов.

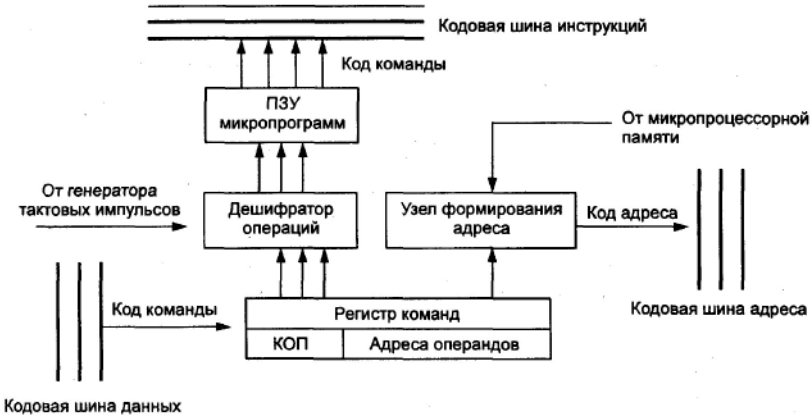


Рис. 5. Упрощенная функциональная схема УУ

Постоянное запоминающее устройство микропрограмм – хранит в своих ячейках управляющие сигналы (импульсы), необходимые для выполнения в блоках компьютера операций обработки информации. Импульс по выбранной дешифратором операции в соответствии с кодом операции считывает из ПЗУ микропрограмм необходимую последовательность управляющих сигналов.

Узел формирования адреса (находится в интерфейсной части МП) – устройство, вычисляющее полный адрес ячейки памяти (регистра) по реквизитам, поступающим из регистра команд и регистров микропроцессорной памяти (МПП).

Кодовые шины данных, адреса и инструкций – часть внутренней интерфейсной шины микропроцессора.

В общем случае УУ формирует управляющие сигналы для выполнения следующих основных процедур:

– выборки из регистра-счетчика адреса команды МПП адреса ячейки ОЗУ, где хранится очередная команда программы;

– выборки из ячеек ОЗУ кода очередной команды и приема считанной команды в регистр команд;

– расшифровки кода операции и признаков выбранной команды;

– считывания из соответствующих расшифрованному коду операции ячеек ПЗУ микропрограмм управляющих сигналов (импульсов), определяющих во всех блоках компьютера процедуры выполнения заданной операции, и пересылки управляющих сигналов в эти блоки;

– считывания из регистра команд и регистров МПП отдельных составляющих адресов операндов (чисел), участвующих в вычислениях, и формирования полных адресов операндов;

– выборки операндов (по сформированным адресам) и выполнения заданной операции обработки этих операндов;

– записи результатов операции в память;

– формирования адреса следующей команды программы.

Арифметико-логическое устройство является частью микропроцессора. АЛУ предназначено для выполнения всех арифметических и логических операций преобразования информации – операций над числовой и символьной информацией. АЛУ вычисления производит в двоичной системе счисления над целыми двоичными числами.

Следует отметить, что любую арифметическую операцию можно реализовать с использованием операции сложения, а сложная логическая задача раскладывается на более простые задачи, где достаточно анализировать только два уровня: *да* и *нет*.

Функционально АЛУ состоит обычно из двух регистров, сумматора и схем управления (местного устройства управления). Упрощенная функциональная схема АЛУ представлена на рис. 6.

Сумматор – вычислительная схема, выполняющая процедуру сложения поступающих на ее вход двоичных кодов; сумматор имеет разрядность двойного машинного слова.

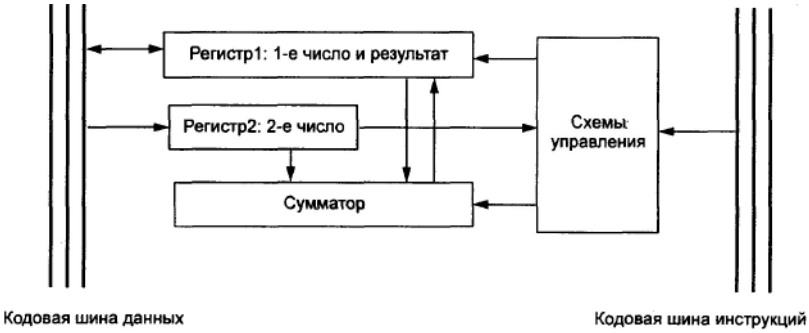


Рис. 6. Упрощенная функциональная схема АЛУ

Регистры – быстродействующие ячейки памяти различной длины: регистр 1 имеет разрядность двойного слова, а регистр 2 – разрядность слова.

При выполнении операций в регистр 1 помещается первое число, участвующее в операции, а по завершении операции – результат; в регистр 2 – 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 этой шины.

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

Итак, *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 с привлечением математического сопроцессора.

Внутренний интерфейс микропроцессора предназначен для связи и согласования МП с системной шиной компьютера, а также для приема, предварительного анализа команд выполняемой программы и формирования полных адресов операндов и команд.

Внутренний интерфейс микропроцессора включает адресные регистры МПП, узел формирования адреса, блок регистров команд, являющийся буфером команд в МП, внутреннюю интерфейсную шину МП и схемы управления шиной и портами ввода/вывода.

Порты ввода/вывода – это пункты системного интерфейса компьютера, через которые МП обменивается информацией с другими устройствами. Всего портов у МП может быть 65536. Каждый порт имеет адрес – номер порта, соответствующий адресу ячейки памяти, являющейся частью устройства ввода/вывода, использующего этот порт, а не частью основной памяти компьютера.

Порт устройства содержит аппаратуру сопряжения и два регистра памяти – для обмена данными и обмена управляющей информацией. Некоторые внешние устройства используют и основную память для хранения больших объемов информации, подлежащей обмену. Многие стандартные устройства (НЖМД, клавиатура, принтер, сканер и др.) имеют постоянно закрепленные за ними порты ввода/вывода.

Схема управления шиной и портами выполняет следующие функции:

– формирование адреса порта и управляющей информации для него (переключение порта на прием или передачу и др.);

– прием управляющей информации от порта, информации о готовности порта и его состоянии;

– организацию сквозного канала в системном интерфейсе для передачи данных между портом устройства ввода/вывода и МП.

Схема управления шиной и портами использует для связи с портами кодовые шины инструкций (КШИ), адреса (КША) и данных (КШД) системной шины: при доступе к порту МП посылает сигнал по КШИ, который оповещает все устройства ввода/вывода, что адрес на КША является адресом порта, а затем посылает и сам адрес порта. То устройство, адрес порта которого совпадает, дает ответ о готовности, после чего по КШД осуществляется обмен данными.

ГЛАВА 5
АРХИТЕКТУРЫ МИКРОПРОЦЕССОРА И РАСПРЕДЕЛЕНИЕ ОПЕРАТИВНОЙ ПАМЯТИ КОМПЬЮТЕРА

5.1. Особенности архитектуры современных микропроцессоров

Одной из основных характеристик микропроцессора (МП) является тип его архитектуры.

К архитектуре МП относят систему команд и способы адресации, возможность совмещения выполнения команд во времени, наличие дополнительных устройств в составе микропроцессора, принципы и режимы его работы и др.

При попытке классифицировать архитектуры современных микропроцессоров, неизбежно возникает проблема, связанная со стремительным и бурным развитием технологий их производства. На каждом этапе развития микропроцессоров в силу различных возможностей элементной базы и предпочтений фирм-производителей доминирует определенная взаимосвязанная совокупность архитектурных идей. В дальнейшем эти идеи могут быть скомбинированы с другими, и представлять собой уже иную архитектуру. В силу указанных причин, предлагаемая ниже классификация не в полной мере будет отражать современный уровень развития архитектур МП.

Под архитектурой МП принято понимать совокупность представлений о составе его компонентов, организации обмена информацией внутри МП и с внешней средой, реализуемых системой команд.

Выделяют понятия: *микроархитектура*; *макроархитектура*.

Микроархитектура микропроцессора – это аппаратная организация и логическая структура микропроцессора, регистры, управля-

ющие схемы, арифметико-логические устройства, запоминающие устройства и связывающие их информационные магистрали.

Макроархитектура – это система команд, типы обрабатываемых данных, режимы адресации и принципы работы микропроцессора.

В целом, архитектуру микропроцессора можно классифицировать следующими признаками: *по количеству обрабатываемых потоков данных*; *по решаемым задачам*; *по типу обрабатываемых команд*; *по виду исполняемых команд*; *по возможности параллельной обработки данных*; *по виду используемой памяти (по количеству используемых шин)*.

По количеству обрабатываемых потоков данных все существующие микропроцессоры условно можно разделить на три класса. Первыми двумя являются *микропроцессоры с векторно-конвейерной архитектурой и ассоциативные процессоры с SIMD-архитектурой (single instruction, multiple data)*. В этих МП все обрабатывающие процессорные элементы выполняют команды одного потока, выдаваемые одним общим устройством управления. По большому счету, все ниже рассматриваемые архитектуры относятся именно к SIMD-архитектуре. К третьему классу МП, выделяемых по данному признаку классификации, можно отнести *мультипроцессорные системы с архитектурой MIMD (multiple instruction, multiple data)*. Такая архитектура обрабатывает одновременно много потоков данных множеством потоков команд.

По решаемым задачам различают регистровую архитектуру микропроцессора, стековую архитектуру микропроцессора, архитектуру микропроцессора, ориентированную на оперативную память.

Регистровая архитектура микропроцессора (архитектура типа «регистр – регистр») определяет наличие достаточно

большого регистрового файла внутри больших интегральных схем или даже сверхбольших интегральных схем микропроцессора. Этот файл образует поле памяти с произвольной записью и выборкой информации. Микропроцессоры с регистровой архитектурой имеют высокую эффективность решения научно-технических задач. Это связано с высокой скоростью *работы сверхоперативного запоминающего устройства (СОЗУ)*, которое позволяет эффективно использовать скоростные возможности арифметико-логического блока. Однако при переходе к решению задач управления эффективность таких микропроцессоров падает, так как при переключениях программ необходимо разгружать и загружать регистры СОЗУ.

Стековая архитектура микропроцессора дает возможность создать поле памяти с упорядоченной последовательностью записи и выборки информации. Эта архитектура эффективна для организации работы с подпрограммами, что необходимо для решения сложных задач управления, или при работе с языками высокого уровня. Хранение адресов возврата позволяет организовать в стеке эффективную обработку последовательностей вложенных подпрограмм. Однако стек на кристалле микропроцессора с малой информационной емкостью быстро переполняется, а стек большой емкости требует значительных ресурсов. Реализация стека в ОЗУ решает эти проблемы.

Архитектура микропроцессора, ориентированная на оперативную память (архитектура типа «память – память»), обеспечивает высокую скорость работы и большую информационную емкость рабочих регистров и стека при их организации в ОЗУ. Эта архитектура отнесена к типу «память – память», поскольку в МП с такой архитектурой все обрабатываемые числа после операции в микропроцессоре вновь возвращаются в память, а не хранятся в рабочих регистрах.

Наибольшее распространение получила именно последняя архитектура. Особенно это относится к системам, работающим в реальном масштабе времени, и к системам обработки данных, ориентированных на использование в системах управления.

По типу обрабатываемых команд выделяют два класса – *RISC (reduce instruction set computer)* – архитектура МП с сокращенным набором команд и *CISC (complete instruction set computer)* – архитектура МП с расширенным набором команд. К этому признаку можно отнести и архитектуру *EPIC (explicitly parallel instruction computing)* – архитектуру с вычислениями с явным параллелизмом команд.

Критерием оптимизации системы команд первых процессоров была минимизация длины программ для решения требуемых задач. При этом вводились команды, которые использовали в качестве операндов как регистры и ячейки памяти, а также сложные схемы формирования адресов с использованием индексных регистров. Набор команд ограничивался форматами «регистр – регистр, регистр», «память – регистр» и «регистр – память». В связи с тем, что компиляторы были не в состоянии эффективно использовать сложные команды, формировалась концепция процессоров с сокращенным набором команд – RISC-процессоров. Появлению RISC-процессоров так же способствовали особенности архитектуры конвейерных процессоров: наличие отдельных наборов команд для работы с памятью и отдельных наборов команд для преобразования информации в регистрах процессора. Каждая такая команда единообразно разбивается на небольшое количество этапов с одинаковым временем исполнения (выборка команды, дешифрация команды, исполнение, запись результата). Это, в свою очередь, позволяет построить эффективный конвейер процессора, способный каждый такт выдавать результат исполнения очередной команды.

Основными чертами RISC-концепции являются:

– одинаковая длина команд;

– одинаковый формат команд;

– операндами команд могут быть только регистры;

– команды выполняют только простые действия;

– наличие большого количества регистров общего назначения (РОН), которые могут быть использованы любой командой;

– наличие конвейеров;

– выполнение команды не дольше, чем за один такт;

– простая адресация.

К RISC процессорам относятся процессоры архитектуры MIPS, SPARC, Power PC, DEC Alpha, HP PA-RISC, Intel 960, AMD 29000 и др.

Очевидно, что RISC-процессоры эффективны там, где можно продуктивно использовать структурные способы уменьшения времени доступа к оперативной памяти. Если программа генерирует произвольные последовательности адресов обращения к памяти и каждая единица данных используется только для выполнения одной команды, то фактически производительность процессора определяется временем обращения к основной памяти. В этом случае использование сокращенного набора команд только ухудшает эффективность, и более оптимальными, с точки зрения производительности, являются CISC-процессоры.

В отличие от RISC-процессоров, в CISC-процессорах, как правило, команды имеют много разных форматов и требуют для своего представления различного числа ячеек памяти. Это обусловливает определение типа команды в ходе ее дешифрации при исполнении, и, в целом, усложняет устройство управления процессора и препятствует повышению тактовой частоты до уровня, достижимого в RISC-процессорах на той же элементной базе. Так, в наборе команд CISC часто присутствуют команды

организации циклов, команды вызова подпрограммы и возврата из подпрограммы, сложная адресация, позволяющая реализовать одной командой доступ к сложным структурам данных. Основной недостаток CISC – большая сложность реализации процессора при малой производительности. Примеры CISC процессоров – семейство Motorola 680x0 и процессоры фирмы Intel от 8086 до Pentium 4.

Несмотря на указанные недостатки, на рынке МП доминируют CISC-процессоры. Это связано с тем, что развитие микропроцессоров происходит при постоянном стремлении сохранения преемственности программного обеспечения (ПО).

Однако сохранение преемственности ПО и существующее требование повышения производительности МП противоречат друг другу. Для разрешения данного противоречия ведущими производителями МП была реализована успешная попытка решения проблемы повышения производительности в рамках CISC-архитектуры. Сохраняя преемственность по системе команд с CISC-микропроцессорами, разработчики создали новые устройства с использованием элементов RISC-архитектуры. В микропроцессор встраивается аппаратный транслятор, превращающий команды CISC-микропроцессора в команды внутреннего RISC-процессора. При этом одна команда из сложного набора может порождать до четырех команд RISC-процессора. Характерным примером таких «смешанных» МП являются МП фирмы AMD и фирмы Intel.

Особенностями EPIC архитектуры являются:

– большое количество регистров; масштабируемость архитектуры до большого количества функциональных устройств;

– явный параллелизм в машинном коде;

– поиск зависимостей между командами производит не процессор, а компилятор;

– предикация (Predication) – команды из разных ветвей условного ветвления снабжаются предикатными полями (полями условий) и запускаются параллельно;

– загрузка по предположению (данные из медленной основной памяти загружаются заранее).

К процессорам с EPIC-архитектурой относят Merced, работы над которым вела фирма Intel и McKinley, который разрабатывался в HP.

По виду исполняемых команд различают *скалярные процессоры* и *векторные процессоры*. Классификация по данному признаку находит свое обоснование по типу исполняемых команд. Если входными операндами команды МП и ее результат являются целочисленные операнды или операнды с плавающей точкой (скаляры), то такой МП относят к классу *скалярных МП*. При этом в МП предусмотрено более одного конвейера, позволяющего одновременно выполнять более одной разной инструкции (команды). Впервые эта возможность была реализована в МП Pentium (два конвейера из пяти стадий каждый), у МП Pentium Pro было уже три конвейера.

Если входные операнды обрабатываемой команды МП и результат являются вектором (массивом) чисел, то такой МП является *векторным (или векторно-конвейерным)*. Появление векторных команд и, соответственно, векторных процессоров обусловлено стремлением ускорить обработку массивов данных за счет исключения затрат времени на выборку и дешифрацию команд обработки, одинаковых для всех компонентов входных массивов.

По возможности параллельной обработки данных выделяют с*уперскалярные*, *с длинным словом команды (VLIW)*, *мультитредовые микропроцессоры*.

Современные микропроцессоры содержат десять и более обрабатывающих устройств, каждое из которых представляет

собой конвейер. В случае эффективной загрузки параллельно функционирующих устройств возможно получение в одном такте нескольких результатов операций, представленных скалярами.

Эффективная загрузка параллельно функционирующих конвейеров обеспечивается либо аппаратурой процессора, либо компилятором, либо совместно аппаратурой и компилятором. В компиляторах используется изощренная техника извлечения параллелизма из последовательных программ. Аппаратура микропроцессоров ориентирована на выделение более простых форм параллелизма, в том числе естественного.

При отображении внутреннего параллелизма обработки данных микропроцессора на архитектурном уровне в системе команд выделяют два крайних подхода.

Первый подход состоит в том, что никакого указания на параллельную обработку внутри процессора система команд не содержит. Такие процессоры относятся к классу *суперскалярных*. Такое название, с одной стороны, отличает эти процессоры от векторных процессоров, а с другой стороны, подчеркивает присущий этим процессорам внутренний параллелизм, обеспечивающий получение в одном такте нескольких скалярных результатов.

В отличие от *скалярных МП,* в данной архитектуре предусмотрен дополнительно хотя бы один конвейер, включающий на одной из стадий несколько параллельных исполнительных блоков.

В *суперскалярных процессорах* в рамках модели последовательных программ реализуется параллельное исполнение команд этих программ. После извлечения последовательного потока команд между командами устанавливаются только действительно необходимые зависимости по данным. При этом, чтобы сохранить порядок при наступлении прерывания, сохраняется

достаточно информации об очередности следования команд в исходной программе.

Типичный суперскалярный процессор выбирает команды и исследует их по мере выполнения. Это действие проводится с целью выявления и обработки команд перехода, идентификации типа команды для ее дальнейшего направления на соответствующий исполнительный блок или в буфер памяти. Эффективность использования суперскалярных архитектур, как правило, ограничивается наличием в программах условных переходов. Это приводит к резкому возрастанию требований к ресурсам и ограничивает количество исполняемых команд. Считается, что пределом распараллеливания при суперскалярной обработке является запуск одновременно на исполнение в каждом такте 7–8 команд.

На рисунке 7 показаны основные компоненты суперскалярного микропроцессора. Функциональными модулями такого МП являются модули выполнения операций с плавающей точкой (FPU) и фиксированной точкой (ALU), устройство загрузки/сохранения, файлы регистров, раздельная кэш-память команд и данных, а так же вспомогательные модули, обеспечивающие динамическое планирование вычислительного процесса, устройство связи с кэш-памятью второго уровня, блок переупорядочивания команд и блок предварительной дешифрации.

Второй подход к отображению присущего микропроцессору внутреннего параллелизма обработки данных на архитектурном уровне в системе команд полностью открывает пользователю все возможности параллельной обработки. В специально отведенных полях команды каждому из параллельно работающих обрабатывающих устройств предписывается действие, которое устройство должно совершить. Такие процессоры называются *процессорами с длинным командным словом (VLIW)*.

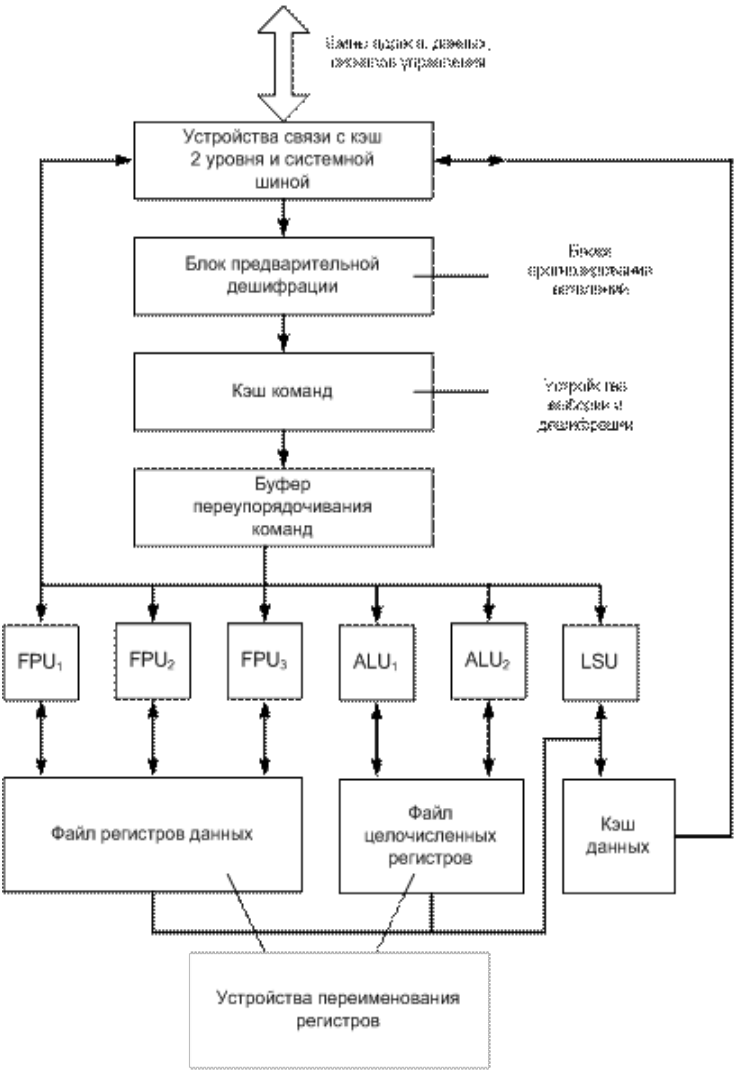


Рис. 7. Структура суперскалярного микропроцессора

Достоинства VLIW заключаются в следующем. Во-первых, компилятор может более эффективно исследовать зависимости между командами и выбирать параллельно исполняемые коман-

ды, чем это делает аппаратура суперскалярного процессора, ограниченная размером окна исполнения. Во-вторых, VLIW-процессор имеет более простое устройство управления и потенциально может иметь более высокую тактовую частоту.

Однако у VLIW-процессоров есть серьезный минус, снижающий их производительность. Это команды ветвления, зависящие от данных, значения которых становятся известны только в динамике вычислений. Очередь команд VLIW-процессора не может быть очень большой ввиду отсутствия у компилятора информации о зависимостях, формируемых динамически, в процессе выполнения. Этот недостаток препятствует возможности переупорядочивания операций в VLIW-процессоре. Кроме того, такой процессор требует большого размера памяти имен, многовходовых регистровых файлов, большого числа перекрестных связей. Возможна также остановка, когда во время выполнения возникла ситуация, отличающаяся от состояния в момент генерации плана выполнения (например, во время выполнения произошло неудачное обращение в кэш).

Суперскалярные микропроцессоры и *микропроцессоры с длинным командным* словом имеют один счетчик команд и, в силу этого, могут быть названы однотредовыми. Такой подход ограничивает производительность вычислительной структуры.

Дальнейшее повышение производительности МП в настоящее время связывается со статическим и динамическим анализом кода с целью выявления параллелизма уровня программных сегментов. При этом для параллельного исполнения команды одной или нескольких программ в микропроцессор вводится несколько счетчиков команд и другое дополнительное оборудование. Такие микропроцессоры с несколькими счетчиками команд получили название *мультитредовые*.

Мультитредовая архитектура решает проблему борьбы с простоями функциональных устройств процессора, возникающими из-за невозможности выполнения следующей команды. Это достигается путем переключения на другой регистровый файл. В результате процессор получает другие данные для продолжения вычислений, переходя к выполнению другого треда (процесса).

Мультитредовый процессор может исполнять процессы, принадлежащие одной или нескольким программам. Если процессор исполняет одну программу, то говорят о его производительности, если несколько программ – о пропускной способности.

В результате применения таких процессоров производительность МП, при прочих равных условиях, может возрасти в десятки раз.

5.2. Архитектура современных многоядерных процессоров

Компьютерная техника развивается быстрыми темпами. Еще в 1965 г. Гордон Мур – один из основателей Intel – пришел к выводу, что «*количество транзисторов, размещаемых на кристалле интегральной схемы, удваивается каждые 24 месяца*». Вычислительные устройства становятся мощнее, компактнее, удобнее, однако, в последнее время повышение производительности устройств стало большой проблемой.

Первые разработки в области создания многопроцессорных систем начались в 1970-х гг. Длительное время производительность привычных одноядерных процессоров повышалась за счет увеличения тактовой частоты (до 80 % производительности определяла только тактовая частота) с одновременным увеличением числа транзисторов на кристалле. Фундаментальные законы физики остановили этот процесс: чипы стали перегреваться,

технологический размер стал приближаться к размерам атомов кремния. Все эти факторы привели к тому, что:

– увеличились токи утечки, вследствие чего повысились тепловыделение и потребляемая мощность;

– процессор стал намного «быстрее» памяти. Производительность снижалась из-за задержки обращения к оперативной памяти и загрузке данных в кэш;

– возникает такое понятие, как «фон-неймановское узкое место». Оно означает неэффективность архитектуры процессора при выполнении какой-либо программы.

Многопроцессорные системы (как один из способов решения проблемы) не получили широкого применения, так как требовали дорогостоящих и сложных в производстве многопроцессорных материнских плат. Исходя из этого, производительность повышалась иными путями. Эффективной оказалась концепция многопоточности – одновременная обработка нескольких потоков команд.

Hyper-threading technology (HTT), или технология сверхпоточной обработки данных позволяла процессору на одном ядре выполнять несколько программных потоков. Именно HTT по мнению многих специалистов стала предпосылкой для создания многоядерных процессоров. Выполнение процессором одновременно нескольких программных потоков называется параллелизмом на уровне потоков (*TLP – thread-level parallelism*).

Для раскрытия потенциала многоядерного процессора исполняемая программа должна задействовать все вычислительные ядра, что не всегда достижимо. Старые последовательные программы, способные использовать лишь одно ядро, теперь уже не будут работать быстрее на новом поколении процессоров, поэтому в разработке новых микропроцессоров все большее участие принимают программисты.

В процессе развития полупроводниковые структуры (микросхемы) эволюционируют, поэтому принципы построения процессоров, количество входящих в их состав элементов, то, как организовано их взаимодействие, постоянно изменяются. Таким образом, процессоры (англ. аббревиатура – CPU) с одинаковыми основными принципами строения принято называть процессорами одной архитектуры. А сами такие принципы называют архитектурой процессора (или микроархитектурой).

Напомним, что *микропроцессор* – это главный компонент компьютера. Он обрабатывает информацию, выполняет программы и управляет другими устройствами системы. От мощности процессора зависит, насколько быстро будут выполняться программы.

Ядро – основа любого микропроцессора. Оно состоит из миллионов транзисторов, расположенных на кристалле кремния. Микропроцессор разбит на специальные ячейки, которые называются *регистрами общего назначения (РОН)*. Работа процессора в общей сложности состоит в извлечении из памяти в определенной последовательности команд и данных и их выполнении. Кроме того, ради повышения быстродействия компьютера, микропроцессор снабжен внутренней кэш-памятью. *Кэш-память* – это внутренняя память процессора, используемая в качестве буфера (для защиты от перебоев со связью с оперативной памятью).

Новейшие процессоры Intel, используемые в IBM-совместимых компьютерах, насчитывают уже более тысяч команд и относятся к процессорам с расширенной системой команд – CISC-процессорам.

Высокопроизводительные вычисления. Параллелизм. Темпы развития вычислительной техники легко проследить: от ENIAC (первая электронная цифровая машина общего назначения) с производительностью в несколько тысяч операций

в секунду до суперкомпьютера Tianhe-2 (1000 трлн операций с плавающей запятой в секунду). Это означает, что скорость вычислений увеличилась в триллион раз за 60 лет. Создание высокопроизводительных вычислительных систем – одна из самых сложных научно-технических задач. Притом, что скорость вычислений технических средств выросла всего лишь в миллионы раз, общая скорость вычислений выросла в триллионы раз. Этот эффект достигнут за счет применения параллелизма на всех стадиях вычислений. Параллельные вычисления требуют поиска рационального распределения памяти, надежных способов передачи информации и координации вычислительных процессов.

Симметрическая мультипроцессорность. *Symmetric multiprocessing (сокращенно SMP) или симметрическое мультипроцессирование* – это особая архитектура мультипроцессорных систем, в которой несколько процессоров имеют доступ к общей памяти. Это очень распространенная архитектура, достаточно широко используемая в последнее время.

При применении SMP в компьютере работает сразу несколько процессоров, каждый над своей задачей. SMP система при качественной операционной системе рационально распределяет задачи между процессорами, обеспечивая равномерную нагрузку на каждый из них. Однако возникает проблема к обращению памяти, ведь даже однопроцессорным системам требуется на это относительно большое время. Таким образом, обращение к оперативной памяти в SMP происходит последовательно: сначала один процессор, затем второй.

В силу перечисленных выше особенностей, SMP-системы применяются исключительно в научной сфере, промышленности, бизнесе, крайне редко в рабочих офисах. Кроме высокой стоимости аппаратной реализации, такие системы нуждаются в очень дорогом и качественном программном обеспечении,

обеспечивающем многопоточное выполнение задач. Обычные программы (игры, текстовые редакторы) не будут эффективно работать в SMP-системах, так как в них не предусмотрена такая степень распараллеливания. Если адаптировать какую-либо программу для SMP-системы, то она станет крайне неэффективно работать на однопроцессорных системах, что приведет к необходимости создания нескольких версий одной и той же программы для разных систем. Исключение составляет, например, программа Ableton Live (предназначена для создания музыки и подготовки DJ-сетов), имеющая поддержку мультипроцессорных систем. Если запустить обычную программу на мультипроцессорной системе, она все же станет работать немного быстрее, чем в однопроцессорной. Это связано с так называемым аппаратным прерыванием (остановка программы для обработки ядром), которое выполняется на другом свободном процессоре.

SMP-система (как и любая другая, основанная на параллельных вычислениях) предъявляет повышенные требования к такому параметру памяти, как полоса пропускания шины памяти. Это зачастую ограничивает количество процессоров в системе (современные SMP-системы эффективно работают вплоть до 16 процессоров).

Так как у процессоров общая память, то возникает необходимость рационального ее использования и согласования данных. В мультипроцессорной системе получается так, что несколько кэшей работают для разделяемого ресурса памяти. *Cache coherence (когерентность кэша)* – свойство кэша, обеспечивающее целостность данных, хранящихся в индивидуальных кэшах для разделяемого ресурса. Данное понятие – частный случай понятия когерентности памяти, где несколько ядер имеют доступ к общей памяти (повсеместно встречается в современных многоядерных системах). Если описать данные понятия в общих чертах, то кар-

тина будет следующей: один и тот же блок данных может быть загружен в разные кэши, где данные обрабатываются по-разному.

Если не будут использованы какие-либо уведомления об изменении данных, то возникнет ошибка. Когерентность кэша нужна для разрешения таких конфликтов и поддержки соответствия данных в кэшах.

SMP-системы являются подгруппой *MIMD (вычислительная система со множественным потоком команд и множественным потоком данных)* классификации вычислительных систем по Флинну[1]. Согласно данной классификации, практически все разновидности параллельных систем можно отнести к MIMD.

Разделение многопроцессорных систем на типы происходит на основе принципа использования памяти. Выделяются следующие типы многопроцессорных систем: *multiprocessors (мультипроцессорные системы с общей разделяемой памятью)* и *multicomputers (системы с раздельной памятью)*. Общие данные, используемые при параллельных вычислениях, требуют синхронизации. Задача синхронизации данных – одна из самых важных проблем и ее решение при разработке многопроцессорных и многоядерных систем и, соответственно, необходимого программного обеспечения является приоритетной задачей инженеров и программистов. Общий доступ к данным может быть произведен при физическом распределении памяти. Этот подход называется неоднородным доступом к памяти (*non-uniform memory access*, или *NUMA*).

Среди данных систем можно выделить:

– системы, где только индивидуальная кэш-память процессоров используется для представления данных (*cache-only memory architecture*);

---

[1] Майкл Флинн – профессор Стэнфордского университета, сооснователь Palyn Associates.

– системы с обеспечением когерентности локальных кэшей для различных процессоров (*cache-coherent NUMA*);

– системы с обеспечением общего доступа к индивидуальной памяти процессоров без реализации на аппаратном уровне когерентности кэша (*non-cache coherent NUMA*).

Упрощение проблемы создания мультипроцессорных систем достигается использованием *распределенной общей памяти (distributed shared memory)*, однако, этот способ приводит к ощутимому повышению сложности параллельного программирования.

Одновременная многопоточность. Исходя из перечисленных недостатков симметрической мультипроцессорности, имеют смысл разработка и развитие других способов повышения производительности. Если проанализировать работу каждого отдельного транзистора в процессоре, можно обратить внимание на очень интересный факт – при выполнении большинства вычислительных операций задействуются далеко не все компоненты процессора (согласно последним исследованиям – около 30 % всех транзисторов). Таким образом, если процессор выполняет, скажем, несложную арифметическую операцию, то большая часть процессора простаивает, следовательно, ее можно использовать для других вычислений. Так, если в данный момент процессор выполняет операции с вещественными числами, то в свободную часть можно загрузить целочисленную арифметическую операцию. Чтобы увеличить нагрузку на процессор, можно создать опережающее выполнение операций, хотя это требует большого усложнения аппаратной логики процессора. Если в программе заранее определить потоки (последовательности команд), которые могут выполняться независимо друг от друга, то это заметно упростит задачу (данный способ легко реализуется на аппаратном уровне). Эта идея, принадлежащая Дину Тулсену, разработанная им еще в 1955 г. в университете Вашингтона, по-

лучила название *одновременной многопоточности (simultaneous multithreading)*. Позднее она была развита компанией Intel под названием *гиперпоточности (hyper-threading)*. Так, один процессор, выполняющий множество потоков, воспринимается операционной системой Windows как несколько процессоров. Использование данной технологии опять-таки требует соответствующего уровня программного обеспечения. Максимальный эффект от применения технологии многопоточности составляет около 30 %.

Многоядерность. Технология многопоточности – это, прежде всего, реализация многоядерности на программном уровне. Дальнейшее увеличение производительности, как всегда, требует изменений в аппаратной части процессора. Усложнение систем и архитектур не всегда оказывается действенным. Существует обратное мнение: «все гениальное – просто!». Действительно, чтобы повысить производительность процессора вовсе необязательно повышать его тактовую частоту, усложнять логическую и аппаратную составляющие, так как достаточно лишь провести рационализацию и доработку существующей технологии. Такой способ весьма выгоден – не нужно решать проблему повышения тепловыделения процессора, разработку нового дорогостоящего оборудования для производства микросхем. Данный подход и был реализован в рамках технологии *многоядерности* – реализации на одном кристалле нескольких вычислительных ядер. Если взять исходный процессор и сравнить прирост производительности при реализации нескольких способов повышения производительности, то очевидно, что применение технологии многоядерности является оптимальным вариантом.

Если сравнивать архитектуры симметричного мультипроцессора и многоядерного, то они окажутся практически идентичными. Кэш-память ядер может быть многоуровневой (локальной и общей,

причем данные из оперативной памяти могут загружаться в кэш-память второго уровня напрямую). Исходя из рассмотренных достоинств многоядерной архитектуры процессоров, производители делают акцент именно на ней. Данная технология оказалась достаточно дешевой в реализации и универсальной, что позволило вывести ее на широкий рынок. Кроме того, эта архитектура внесла свои коррективы в закон Мура: «*Количество вычислительных ядер в процессоре будет удваиваться каждые 18 месяцев*».

Если посмотреть на современный рынок компьютерной техники, то можно увидеть, что доминируют устройства с 4- и 8-ядерными процессорами. Кроме того, производители процессоров заявляют, что в скором времени на рынке можно будет увидеть процессоры с сотнями вычислительных ядер. Как уже неоднократно говорилось ранее, весь потенциал многоядерной архитектуры раскрывается только при наличии качественного программного обеспечения. Таким образом, сфера производства аппаратного обеспечения и программного обеспечения очень тесно связаны между собой.

5.3. Тенденции развития современных микропроцессоров

В настоящее время существует множество типов архитектур. Исторически архитектуры процессоров возникли исходя из предполагаемого набора задач, которые должны были исполняться на этих процессорах, и преемственности решаемых программ. Сегодня имеющиеся аппаратные ресурсы позволяют собрать в одном процессоре все известные архитектурные приемы повышения производительности, сообразуясь только с взаимной совместимостью. Поэтому, как считают некоторые современные ученые, скорее всего до появления единой архитектуры процессора еще достаточно далеко.

Анализ конкретных семейств микропроцессоров разных производителей подтверждает общие тенденции их развития: повышение тактовой частоты, увеличение объема и пропускной способности подсистемы памяти, увеличение количества параллельно функционирующих исполнительных устройств. Совокупная реализация в одном микропроцессоре рекордных значений по всем этим тенденциям невозможна из-за фундаментальных физических ограничений, а также из-за ограничений технологического процесса изготовления и экономических ограничений на стоимость одного микропроцессора и микроэлектронного производства в целом. Поэтому каждый конкретный тип микропроцессора есть результат многих компромиссов, принятых его создателями.

5.4. Классическое распределение оперативной памяти

Логическая структура оперативной памяти компьютера ранее была обусловлена особенностями системы адресации процессоров семейства x86. Процессоры 8086/88, применявшиеся в первых моделях компьютера, имели доступное адресное пространство 1 Мбайт (20 бит шины адреса). Эти процессоры использовали сегментную модель памяти, унаследованную и следующими моделями в реальном режиме. Согласно этой модели исполнительный (линейный) адрес вычислялся по определенной формуле. Таким образом, обеспечивался доступ к адресному пространству Addr = 00000 – FFFFFh при помощи пары 16-битных регистров. Заметим, что при Seg = FFFFh и Offset = FFFFh данная формула дает адрес 10FFEFh, но ввиду 20-битного ограничения на шину адреса эта комбинация в физической памяти указывала на 0FFEFh. Таким образом, адресное пространство как бы сворачивалось в кольцо с небольшим «нахлестом».

Начиная с процессора 80286, шина адреса была расширена до 24 бит, а впоследствии (386DX, 486 и выше) – до 32 и даже 36 (P6).

P6 – компьютеры основаны на процессоре Pentium Pro и выше. В реальном режиме работы процессора, используемом в DOS, применялась та же сегментная модель памяти, и формально был доступен лишь 1 Мбайт памяти, что являлось недостаточным для большинства приложений. Однако выяснилось, что процессоры 80286 в реальном режиме эмулируют 8086 с ошибкой: та самая единица в бите A20, которая отбрасывалась в процессорах 8086/88, теперь попадала на шину адреса, и в результате максимально доступный линейный адрес в реальном режиме достигал 10FFEFh. За эту ошибку с радостью ухватились разработчики компьютеров, поскольку дополнительные (64K – 16) байты оперативной памяти, адресуемой в реальном режиме, оказались подарком, позволяющим освободить дефицитное пространство оперативной памяти для прикладных программ. В эту область (100000h – 10FFEFh), названную «*высокой памятью*» – *high memory area (HMA)*, стали помещать часть операционной системы и небольшие резидентные программы.

Однако для обеспечения полной совместимости с процессором 8086/88 в схему компьютера ввели вентиль линии A20 шины адреса – *GateA20*, который либо пропускал сигнал от процессора, либо принудительно обнулял линию A20 системной шины адреса. Старшие биты такой «заботы» не требовали, поскольку переполнение при суммировании 16-битных компонентов адреса по данной схеме до них не распространялось.

Управление этим вентилем подключили к свободному программно-управляемому выходному биту 1 контроллера клавиатуры 8042, ставшего стандартным элементом архитектуры ком-

пьютера, начиная с AT. Предполагалось, что этим вентилем часто пользоваться не придется. Но оказалось, что переключение вентиля в многозадачных операционных системах, часто переключающих процессор между защищенным режимом, реальным режимом и режимом V86, контроллером клавиатуры выполняется слишком медленно. Так, появились альтернативные методы быстрого переключения вентиля, специфичные для различных реализаций системных плат (например, через порт 92h). Кроме того, иногда использовали и аппаратную логику быстрого декодирования команды на переключение бита, поступающую к контроллеру клавиатуры. Для определения способа переключения в утилиту CMOS Setup ввели соответствующие параметры, позволяющие выбрать между стандартным, но медленным способом, и менее стандартизованным, но быстрым, в зависимости от используемого программного обеспечения.

32-разрядные процессоры позволяли организовать режим, иногда называемый «нереальным» или «большим реальным», в котором инструкции выполнялись как в реальном, но доступны были все 4 Гбайт памяти. Этот режим часто использовался в игровых программах, целиком захватывающих все ресурсы компьютера, не заботясь о «правилах хорошего тона» по отношению к другим исполняемым программам.

Основную часть адресного пространства до сих пор занимает оперативная память. Объем установленной памяти определяется тестом POST при начальном включении (перезагрузке) компьютера, начиная с младших адресов. Натолкнувшись на отсутствие памяти (ошибку), тест останавливается на достигнутом и сообщает системе объем реально работающей памяти.

Классическое распределение памяти компьютера, непосредственно адресуемой процессором, приведено на рис. 8 и представляется следующим образом.

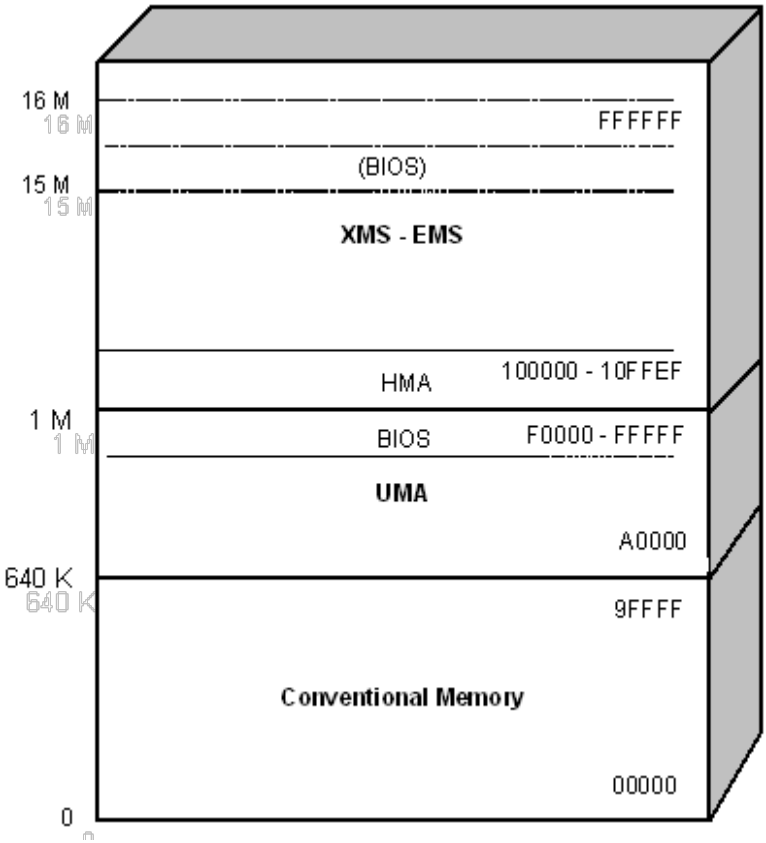


Рис. 8. Классическое распределение памяти компьютера

00000h – 9FFFFh – *conventional (base) memory,* 640 Кбайт – стандартная (базовая) память, доступная DOS и программам реального режима. В некоторых системах с видеоадаптером MDA верхняя граница сдвигалась к AFFFFh (704 Кбайт).

Иногда верхние 128 Кбайт стандартной памяти (область 80000h – 9FFFFh) называли *extended conventional memory*.

A0000h – FFFFFh – *upper memory area (UMA),* 384 Кбайт – верхняя память, зарезервированная для системных нужд. В ней размещались области буферной памяти адаптеров (например,

видеопамять) и постоянная память (BIOS с расширениями). Эта область, обычно использовалась не в полном объеме, ставила непреодолимый архитектурный барьер на пути непрерывной (нефрагментированной) памяти, о которой всегда мечтали программисты.

Память выше 100000h – *extended memory – дополнительная (расширенная) память*, непосредственно доступная только в защищенном (и в «большом реальном») режиме для компьютеров с процессорами 286 и выше. В ней выделялась область 100000h – 10FFEFh – *высокая память, HMA, – единственная область расширенной памяти*, доступная 286+ в реальном режиме при открытом вентиле *Gate A20.*

Область памяти выше первого мегабайта в различных источниках называлась по-разному. Ее английское название – *extended memory* – пересекалось с названием одной из спецификаций ее использования – *extended memory specification*. Но название другой спецификации использования – *expanded memory specification* – в прямом переводе на русский язык неотличимо от перевода предыдущего термина (и *extended*, и *expanded* переводятся как «расширенный»). Будем придерживаться терминологии, укрепившейся в литературе, и область всей физической памяти, расположенной в адресном пространстве выше 1 Мбайт, будем называть дополнительной памятью. Ее объем у ранних компьютеров указывался строкой *Extended Memory xxxxx Kbyte* в таблице, выводимой после прохождения теста POST, и в меню стандартной конфигурации CMOS Setup.

Вышеприведенное разделение памяти актуально только для приложений и операционных систем реального режима типа MS-DOS. Для ОС защищенного режима (в том числе Windows 9x/NT/2000) доступна была вся оперативная память, причем без каких-либо ухищрений вроде EMS и XMS, описанных ниже.

Однако область UMA, сохраняемая ради совместимости, оставалась барьером на пути к единой однородной памяти.

Для компьютеров класса AT-286 с 24-битной шиной адреса верхняя граница оперативной памяти была FDFFFFh (максимальный размер 15,9 Мбайт). Область FE0000h – FFFFFFh содержала ПЗУ BIOS (ROM BIOS Area), обращение к этой области было эквивалентно обращению к ROM BIOS по адресам 0E0000h – 0FFFFFh.

Для 386+ процессоров и 32-битной шины адреса теоретическая верхняя граница составляла 4 Гбайт, а для P6 – 64 Гбайт (36-битная шина адреса). В компьютерах с 32-разрядной шиной адреса образ BIOS дополнительно проецировался в адреса FFFE0000h – FFFFFFFFh, хотя для процессоров P6 это было необязательно.

Однако иногда использовалась и проекция BIOS в область FE0000h – FFFFFFh, что не позволяло задействовать более 16 Мбайт ОЗУ, поскольку система воспринимала только найденную непрерывную область оперативной памяти. Если 32-разрядный компьютер имел отображение области BIOS под границей 16 Мбайт, это отображение обычно можно было запретить установкой соответствующего параметра CMOS Setup.

Иногда для использования специфических адаптеров ISA, имеющих буфер с адресами в 16-м мегабайте памяти, предусматривали параметр *Memory Hole At 15M-16M*. Его установка также не позволяла использовать оперативную память свыше 16 Мбайт.

Ранее системные платы позволяли установить до 2048 Мбайт ОЗУ, для первых мощных серверных платформ это не было пределом. Обращение по адресам, превышающим границу установленной оперативной памяти (или максимально возможного объема), транслировалось на шину PCI, которая имела 32-битную адресацию.

Компьютеры, использующие режим системного управления *SMM (system management mode)*, имевшийся у большинства процессоров тех поколений, имел еще одно адресное пространство памяти – *SMRAM*. Это адресное пространство «параллельно» пространству обычной памяти и при работе было доступно процессору только в режиме обработки SMI. Память SMRAM могла представлять собой часть физической оперативной памяти (DRAM), а могла быть реализована и специальной микросхемой энергонезависимой памяти. Ее размер мог варьироваться в диапазоне от 32 Кбайт (минимальные потребности SMM) до 4 Гбайт. SMRAM располагалась, начиная с адреса SMIBASE (по умолчанию 30000h), и распределялась относительно адреса SMIBASE следующим образом.

SMI# (*system management interrupt*) являются сигналом на входе. В режим SMM процессор может войти только по этому сигналу. Сигнал SMI# для процессора является немаскируемым прерыванием с наивысшим приоритетом. FE00h – FFFFh (3FE00h – 3FFFFh) – область сохранения контекста распределяется, начиная со старших адресов по направлению к младшим. По прерыванию SMI сохраняются почти все регистры процессора, но сохранение регистров FPU не производится.

8000h (38000h) – точка входа в обработчик (SMI Handler).

0-7FFFh (30000h – 37FFFh) – свободная область.

Память SMRAM должна была быть схемотехнически защищена от доступа прикладных программ. Процессор генерировал специальный выходной сигнал SMIACT# во время обработки SMI, который и должен был быть «ключом» доступа к этой памяти. Если SMRAM не являлась энергонезависимой, то системная логика должна была обеспечить возможность ее инициализации (записи программного кода обработчика) процессором из обычного режима работы до разрешения появления сигнала SMI#.

Стандартная память – Conventional. Стандартная память являлась самой дефицитной в компьютере, когда речь шла о работе в среде операционных систем типа MS-DOS. На ее небольшой объем (типовое значение 640 Кбайт) претендовали и BIOS, и ОС реального режима, а остатки отдавались прикладному ПО. Стандартная память распределялась таким образом:

– 00000h – 003FFh – *Interrupt Vectors – векторы прерываний (256 двойных слов);*

– 00400h – 004FFh – *BIOS Data Area – область переменных BIOS;*

– 00500h – 00xxxh – *DOSArea – область DOS;*

– 00xxxh – 9FFFFh – *User RAM – память, предоставляемая пользователю (до 638 Кбайт).*

При использовании PS/2 Mouse область 9FC00h – 9FFFFh использовалась как расширение BIOS Data Area, и размер User RAM уменьшался.

Верхняя память – UMA. Верхняя память имеет области различного назначения, которые могли быть заполнены буферной памятью адаптеров, постоянной памятью или оставаться незаполненными. Раньше эти пустоты не использовали из-за сложности «фигурного выпиливания» адресуемого пространства. С появлением механизма страничной переадресации (у процессоров 386 и выше) их стали по возможности заполнять «островками» оперативной памяти, названными блоками верхней памяти *UMB (upper memory block)*. Эти области были доступны DOS для размещения резидентных программ и драйверов через драйвер EMM386, который отображал в них доступную дополнительную память.

Стандартное распределение верхней памяти выглядело так:

– A0000h – BFFFFh – *Video RAM, 128 Кбайт – видеопамять (обычно используется не полностью);*

– C0000h – DFFFFh – *Adapter ROM, Adapter RAM, 128 Кбайт – резерв для адаптеров, использующих собственные модули ROM BIOS или (и) специальное ОЗУ, разделяемое с системной шиной;*

– *E0000h – EFFFFh – свободная область, 64 Кбайт, иногда занятая под System BIOS;*

– F0000h – FFFFFh – *System BIOS, 64 Кбайт – системная BIOS;*

– FD000h – FDFFFh – *ESCD (Extended System Configuration Data) – область энергонезависимой памяти, используемая для конфигурирования устройств plug and play. Эта область имелась только при наличии PnP BIOS, ее положение и размер жестко не задавались*.

В области UMA практически всегда присутствовал графический адаптер. В зависимости от модели он занимал следующие области:

– MDA RAM – B 0000h – B0FFFh;

– CGA RAM – B8000h – BBFFFh;

– EGA ROM – C0000h – C3FFFh/C7FFFh;

– VGA ROM – C0000h – C7FFFh;

– EGA, VGA RAM – A0000h – BFFFFh, в зависимости от видеорежима использовались следующие области: Graphics – A0000h – AFFFFh; Color Text – B8000h – BFFFFh; Mono Text – B0000h – B7FFFh.

Также распространенным потребителем UMA являлись расширения ROM BIOS, расположенные на платах дисковых контроллеров, и микросхемы удаленной загрузки (Boot ROM) на платах адаптеров ЛВС. Обычно они занимали область C8000h – CBFFFh/C9FFFh/C8FFFh (для дисковых контроллеров), но могли и перемещаться при конфигурировании адаптеров. Размер области, занимаемой системной ROM BIOS, колебался от 8 Кбайт

у PC/XT до 128 Кбайт, однако разумное значение было 64 Кбайт. Большая область использовалась благодаря появлению микросхем ROM и флеш-памяти объемом 1 Мбит (128K×8), но при этом размер доступной UMA сокращался. Тогда же микросхемы того же (и большего) объема стали отображать только на область FOOOOh – FFFFFh (64 Кбайт), а иногда и меньшую. Это оказалось возможным, поскольку не все содержимое микросхемы ROM BIOS должно было быть доступно одновременно. Таким способом удавалось примирить интересы пользователей UMB с необходимостью расширения объема BIOS, связанной с усложнением технических средств.

Видеопамять графического адаптера являлась особой областью памяти, к которой во время непрерывного процесса регенерации экрана интенсивно обращались и центральный процессор, и графический акселератор (если таковой имелся). Видеопамять всегда традиционно являлась физически выделенной памятью сравнительно (по сравнению с ОЗУ) небольшого объема и для нее разными способами обеспечивали максимальную производительность – увеличивали разрядность до 128 бит, повышали частоту, применяли специализированные, в том числе и двухпортовые, микросхемы памяти. Это, конечно же, приводило к удорожанию компьютера.

Вместо предоставления локальной памяти адаптера была предложена *архитектура унифицированной памяти UMA (unified memory architecture)*. Тогда для видеопамяти (и других нужд акселератора) выделялась область в общем пространстве единой физической оперативной памяти. За этот способ снижения стоимости приходилось расплачиваться снижением производительности, как видеосистемы, так и основной памяти. Архитектура UMA применялась в чипсетах системной платы с интегрированной графикой для недорогих компьютеров. При этом

могла предоставляться возможность установки и дополнительного специализированного модуля видеопамяти, позволявшая за дополнительные деньги отказаться от UMA. Если с графического адаптера AGP снимали локальную память, то этот высокопроизводительный адаптер вырождался в систему с UMA.

Accelerated Graphic Port, ускоренный графический порт. Отображаемая и расширенная память – спецификации EMS и XMS. *Отображаемая память EMS (expanded memory specification)* – программная спецификация использования дополнительной памяти DOS-программами реального режима. Спецификация *LIM EMS* – соглашение фирм Lotus, Intel, Microsoft на использование EMS. С помощью специальных аппаратных или программных средств любая область дополнительной памяти могла быть отображена на небольшие страницы, расположенные в области UMA. В первоначальном варианте можно было использовать 4 страницы по 16 Кбайт, примыкающие друг к другу, обычно начиная с адреса D0000h (положение страниц можно менять в пределах свободных областей UMA). Обращение прикладных программ к памяти EMS осуществлялось через диспетчер памяти, вызываемый по прерыванию Int 67h. Программа, нуждающаяся в дополнительной памяти, должна была сначала запросить выделение области, указав ее размер в 16-килобайтных страницах. В ответ на этот запрос (если имелась свободная память) диспетчер сообщал программе номер дескриптора EMS (EMS handler), по которому программа в дальнейшем ссылалась на выделенную ей область при управлении отображением. Далее программа через диспетчер назначала отображение требуемой логической страницы из выделенной ей области дополнительной памяти на выбранную физическую страницу, расположенную в области UMA. После этого любые программные обращения процессора к физической странице, расположенной в пределах

первого мегабайта, могли в действительности работать с логической страницей дополнительной памяти, расположенной выше первого мегабайта, причем без переключения в защищенный режим. Для работы с иной логической страницей требовался вызов диспетчера для переназначения отображения. В EMS 4.0, эмулируемой на процессорах 386+, была возможность увеличения числа доступных физических страниц и отображения дополнительной памяти не только на фиксированные области UMA, и на любые области памяти.

Для поддержки EMS поначалу требовались специальные аппаратные средства. В компьютерах на процессорах 386 и выше появилась возможность программной эмуляции EMS, которую в MS-DOS 5+ выполнял драйвер EMM386.EXE.

Система EMS в основном была предназначена для хранения данных – для исполняемого в данный момент программного кода она оказалась неудобной, поскольку требовала программного переключения страниц через каждые 16 Кбайт. EMS использовалась в основном очень ранним программным обеспечением, фирма Lotus продвигала эту спецификацию для хранения своих больших электронных таблиц. Ее использовали для создания виртуальных дисков, хранения очередей заданий для печати, а также и для хранения данных и даже программного кода некоторых резидентных программ (в целях экономии стандартной памяти).

Расширенная память XMS (extended memory specification) – иная программная спецификация использования дополнительной памяти DOS-программами, разработанная компаниями Lotus, Intel, Microsoft и AST для компьютеров на процессорах 286 и выше. Эта спецификация позволяла программе получить в распоряжение одну или несколько областей дополнительной памяти, а также использовать область HMA. Распределением областей ведал диспетчер расширенной памяти – драйвер HIMEM.

SYS. Диспетчер позволял захватить или освободить область HMA (65 520 байт, начиная с 100000h), а также управлять вентилем линии адреса A20. Функции, собственно, XMS позволяли программе:

– определить размер максимального доступного блока памяти;

– захватить или освободить блок памяти;

– копировать данные из одного блока в другой, причем участники копирования могли быть блоками как стандартной, так и дополнительной памяти в любых сочетаниях;

– запереть блок памяти (запретить копирование) и отпереть его;

– изменить размер выделенного блока.

В ответ на запрос выделения области диспетчер выдавал номер дескриптора блока (16-битное число XMS handler), по которому выполнялись дальнейшие манипуляции с этим блоком. Размер блока мог достигать 64 Мбайт. Спецификация XMS позволяла программам реального режима устраивать «склады» данных в дополнительной памяти, которая им непосредственно была недоступна, копируя в нее и из нее данные доступных областей первого мегабайта памяти. Доступ к диспетчеру XMS осуществлялся через прерывание Int 2Fh. Заботу о переключении в защищенный режим и обратно для получения доступа к дополнительной памяти брал на себя диспетчер. По умолчанию HIMEM.SYS позволял использовать до 32 дескрипторов блоков, но это число можно было увеличить, задав параметр /NUMHANDLES=xx в строке загрузки драйвера HIMEM.SYS.

Кроме работы с дополнительной памятью спецификация XMS определяла пару функций и для работы с блоками UMB – захват блока требуемого размера (или определение максимально доступных блоков) и освобождение его.

Как видно, спецификации EMS и XMS отличались по принципу действия: в EMS для доступа к дополнительной памяти выполнялось отображение (страничная переадресация) памяти, а в XMS – копирование блоков данных. На компьютерах с процессорами 386+ эти спецификации мирно сосуществовали при использовании драйвера HIMEM.SYS, поверх которого мог быть загружен и драйвер EMM386.EXE, пользующийся памятью XMS для эмуляции EMS-памяти. Память, доступная EMS и XMS, могла выделяться динамически из числа дополнительной. Ключ NOEMS в строке запуска EMM386 запрещал выделение памяти под использование по спецификации EMS.

Теневая память – Shadow ROM и Shadow RAM. В области верхней памяти UMA обычно располагались устройства с медленной памятью: системная BIOS (*System ROM BIOS*), расширения BIOS на графическом адаптере (*Video ROM BIOS*), на контроллерах дисков и интерфейсов (*Adapter ROM*), ПЗУ начальной загрузки на сетевой карте (*Boot ROM*), видеопамять (*video memory buffer*). Они, как правило, были реализованы на 8-или 16-битных микросхемах с довольно большим временем доступа. Обращение к полноразрядному системному ОЗУ выполнялось гораздо быстрее. Для ускорения обращений к памяти этих устройств применялась теневая память (*Shadow Memory*) – подмена ее системным ОЗУ. Теневая память появилась на развитых моделях AT-286, где она была реализована аппаратно. Процессоры класса 386+ позволяли ее реализовать программно, с помощью страничной переадресации.

Затенение ОЗУ и ПЗУ выполнялось по-разному. При инициализации теневого ПЗУ (*Shadow ROM*) содержимое затеняемой области копировалось в ОЗУ и при дальнейшем чтении по этим адресам подставлялось в ОЗУ, а запись в эту область блокировалась.

При использовании теневого ОЗУ (*Shadow RAM*) запись производилась одновременно в физическую память затеняемой области и в системное ОЗУ, наложенное на эту область. При чтении затененной области обращение шло только к системной памяти, что происходило гораздо быстрее. Особенно велик был эффект от затенения видеопамяти старых графических адаптеров, которая по чтению была доступна только во время обратного хода развертки, и процессору приходилось долго ждать этого момента. Однако затенение областей разделяемой памяти, модифицируемых со стороны адаптеров, было недопустимо – эти изменения не воспринимались процессором. К разделяемой относилась буферная память сетевых адаптеров, видеопамять адаптеров с графическими сопроцессорами (акселераторами). Из этого следует, что затенение видеопамяти применимо было только к примитивным графическим картам, устанавливаемым в слот ISA, и то не во всех режимах.

Обычно теневая память включалась через CMOS Setup отдельными областями, размером по 16 Кбайт или более крупными, и для каждой области указывали режим затенения (*Shadow ROM* или *Shadow RAM*). Также было возможно ее включение и драйверами ОС (например, драйвером EMM386). На ранних системных платах затенение области системной BIOS выполнялось всегда, на самых старых платах затенением этой области можно было управлять. Затенение BIOS видеоадаптера (*Video BIOS Shadowing*) для работы в среде Windows с «родными» драйверами графического адаптера могло и не давать прирост производительности.

5.5. Система управления памятью

Организация распределения памяти компьютера. Запоминающие устройства являются одной из основных частей лю-

бого компьютера. Их работа, как отмечалось ранее, строится по иерархическому принципу. От того, насколько рационально организовано использование памяти на каждом из уровней иерархии и взаимодействие между ЗУ различных уровней, во многом зависит эффективность работы компьютера.

Ключевую роль в этой иерархии играет *оперативная память*. Именно в ней хранятся программы во время их исполнения, именно отсюда загружаются в регистры микропроцессора исходные данные для обработки. Сюда же передаются и окончательные результаты работы программ. Поэтому рациональное использование ОЗУ на протяжении всего времени работы компьютера чрезвычайно важно.

В оперативной памяти мультипрограммных компьютеров обычно постоянно хранится *ядро* операционной системы. Программы ядра операционной системы в процессе работы компьютера выполняются часто, время их выполнения невелико. Остальные части операционной системы, как правило, находятся во внешней памяти, в случае необходимости требуемые модули загружаются в оперативную *память*, занимая ее часть. В оставшейся части оперативной памяти хранятся несколько программ, выполняемых в мультипрограммном режиме, и используемые ими данные.

Распределение памяти предполагает удовлетворение потребностей, как пользователей, так и системных средств, но эти требования в большей части противоречивы.

Системные цели заключаются, прежде всего, в увеличении степени использования оперативной памяти при параллельном развитии нескольких процессов в мультипрограммном режиме, а также в реализации защиты информации при развитии этих *процессов, обеспечении* взаимодействия между процессами и т. д.

Требования пользователей к памяти весьма разнообразны: быстрое выполнение коротких программ, выделение оператив-

ной памяти в размерах, превышающих физически существующую, легкость и простота взаимодействия с другими программами при использовании, например, общих процедур и т. п.

Вследствие этого *распределение памяти* всегда носит компромиссный характер.

Система управления памятью выполняет следующие основные функции:

– учет состояния свободных и уже распределенных областей памяти и модернизация этой информации всякий раз, когда в распределении памяти производятся изменения;

– распределение памяти для выполнения задач (определение, какой задаче, когда и в каком количестве выделить оперативную память);

– непосредственное выделение задаче оперативной памяти; если свободные области оперативной памяти отсутствуют, то предварительное их освобождение путем сохранения информации во внешней памяти.

Все доступное множество адресов элементов хранения, упорядоченное по какому-либо признаку, называют адресным пространством памяти. Физическое *адресное пространство* организовано просто как *одномерный массив* ячеек, каждой из которых присвоен свой номер, называемый физическим адресом.

В общем случае, под адресом понимают некий *идентификатор*, однозначно определяющий расположение элемента хранения среди прочих элементов в составе среды хранения.

Для адресации данных в физическом адресном пространстве программы используют логическую адресацию. *Процессор* автоматически транслирует *логические адреса* в физические, выдаваемые на адресную шину и воспринимаемые схемами управления (контроллерами) памяти.

Существуют две стратегии распределения оперативной памяти, как и любого ресурса: *статическое и динамическое распределение*.

При *статическом распределении* вся необходимая оперативная память выделяется процессу в момент его порождения. При этом память выделяется единым блоком необходимой длины, начало которого определяется базовым адресом. Программа пишется в адресах относительно начала блока, а физический адрес команды или операнда при выполнении программы формируется как сумма базового адреса блока и относительного адреса в блоке. Значение базового адреса устанавливается при загрузке программы в оперативную память. Так как в разных программах используются блоки разной длины, то при таком подходе возникает проблема фрагментации памяти, т. е. возникают свободные участки памяти, которые невозможно без предварительного преобразования использовать для вычислительного процесса.

Проиллюстрируем это простым примером.

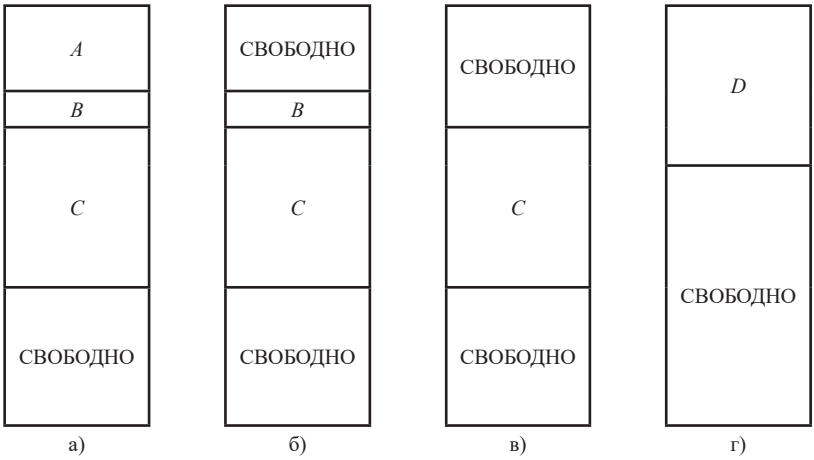


Рис. 9. Статическое распределение памяти:
а – начальное распределение; б – после завершения программы A;
в – после завершения программы B; г – после завершения программы C

Пусть ОП имеет объем 10 Мбайт, а для выполнения программ A, B, C, D требуются следующие объемы памяти: A – 2 Мбайт, B – 1 Мбайт, C – 4 Мбайт, D – 4 Мбайт. Начальное *распределение памяти* и *распределение памяти* после выполнения некоторых программ представлено на рис. 9.

Из рисунка видно, что *программа D* при *статическом распределении памяти* может быть загружена в оперативную *память* лишь после завершения выполнения всех предыдущих программ, хотя необходимый для нее объем памяти существовал уже после завершения работы программы A. В то же время для улучшения показателей работы мультипрограммного компьютера требуется, чтобы в оперативной памяти находилось постоянно возможно большее количество программ, готовых к выполнению.

Данную проблему можно частично решить перераспределением памяти после завершения выполнения каждой программы с целью формирования единого свободного участка, который может быть выделен новой программе, поступающей на обработку (*дефрагментация памяти*). Однако это требует трудоемкой работы системных средств и практически не используется.

Современные системы распределения памяти опираются на две концепции: динамического использования ресурсов и виртуализации.

При *динамическом распределении памяти* каждой программе в начальный момент выделяется лишь часть от всей необходимой ей памяти, а остальная часть выделяется по мере возникновения реальной потребности в ней. Такой подход базируется на следующих предпосылках.

Во-первых, при каждом конкретном исполнении в зависимости от исходных данных некоторые части программы (до 25 % ее длины) вообще не используются. Следует стремиться к тому,

чтобы эти фрагменты кода не загружались в оперативную память.

Во-вторых, *исполнение* программы характеризуется так называемым *принципом локальности* ссылок. Он подразумевает, что при исполнении программы в течение некоторого относительно малого интервала времени происходит обращение к памяти в пределах ограниченного диапазона адресов (как по коду программы, так и по данным). Следовательно, на протяжении этого времени нет необходимости хранить в оперативной памяти другие блоки программы.

При этом системные средства должны отслеживать возникновение требований на обращение к тем частям программы, которые в данный момент отсутствуют в ОЗУ, выделять этой программе необходимый *блок памяти* и помещать туда из внешнего запоминающего устройства требуемую часть программы. Для этого может потребоваться предварительное перемещение некоторых блоков информации из ОЗУ во внешнюю *память. Данные* перемещения должны быть скрыты от пользователя, и в наименьшей степени замедлять работу его программы.

Перемещение блоков информации из ОЗУ во внешнюю *память* с целью освобождения места для новой информации происходит обычно по одному из следующих алгоритмов:

– *LRU (least recently used)* – давно не использовавшийся;
– *FIFO* – самый давний по пребыванию в ОЗУ;
– *Random* – случайным образом.

Динамическое распределение памяти тесно переплетается с понятием *виртуальной памяти*.

Принцип *виртуальной памяти* предполагает, что пользователь при подготовке своей программы имеет дело не с физической оперативной памятью, действительно работающей в составе компьютера и имеющей некоторую фиксированную емкость,

а с виртуальной (кажущейся) одноуровневой памятью, емкость которой равна всему адресному пространству, определяемому размером адресной шины ($L_{ша}$) компьютера: если $V_{вирт} >> V_{физ}$, то $V_{вирт} = 2^{Lша}$.

Для *персонального компьютера* на основе 32-разрядных микропроцессоров: $V_{вирт} = 2^{32} = 4$ Гбайт.

При этом, естественно, в ЭВМ должен быть обеспечен достаточный объем внешней памяти для хранения всех программ, обрабатываемых на компьютере.

Программист имеет в своем распоряжении *адресное пространство*, ограниченное лишь разрядностью адресной шины, независимо от общего объема оперативной памяти компьютера и объемов памяти, используемых другими программами, параллельно обрабатываемыми в мультипрограммной ЭВМ.

Виртуальная *память*, обеспечивая возможность программисту обращаться к очень большому объему непрерывного адресного пространства, предоставляемого в его монопольное распоряжение, обладает обычными свойствами: побайтовая *адресация*, *время доступа*, сравнимое со временем доступа к оперативной памяти.

На всех этапах подготовки программ, включая загрузку в оперативную *память, программа* представляется в *виртуальных адресах*, и лишь при выполнении машинной команды *виртуальные адреса* преобразуются в физические. Для каждой программы, выполняемой в мультипрограммном режиме, создается своя *виртуальная память*. Каждая *программа* использует одни и те же *виртуальные адреса* от нулевого до максимально большого в данной архитектуре.

Для преобразования виртуальных адресов в физические физическая и *виртуальная память* разбиваются на блоки фиксированной длины, называемые страницами. Объемы виртуальной

и *физической страниц* совпадают. Страницы виртуальной и физической памяти нумеруются.

Виртуальный (логический) адрес в этом случае представляет собой номер виртуальной страницы и смещение внутри этой страницы. В свою очередь, физический *адрес* – это номер *физической страницы* и смещение в ней.

Сначала в оперативную память загружается первая страница программы и ей передается управление. Когда в ходе выполнения программы происходит обращение за пределы загруженной страницы, *операционная система* прерывает выполнение данной программы, загружает требуемую страницу в оперативную память, после чего передает управление прерванной программе.

Правила перевода номеров *виртуальных страниц* в номера *физических страниц* обычно задаются в виде таблицы *страничного преобразования*. Такие таблицы формируются системой управления памятью и модифицируются каждый раз при перераспределении памяти.

Перевод виртуальных адресов в физические проиллюстрирован на рис. 10.

Рассмотрим пример преобразования адреса *виртуальной страницы* в *адрес физической страницы*. Пусть *компьютер* имеет

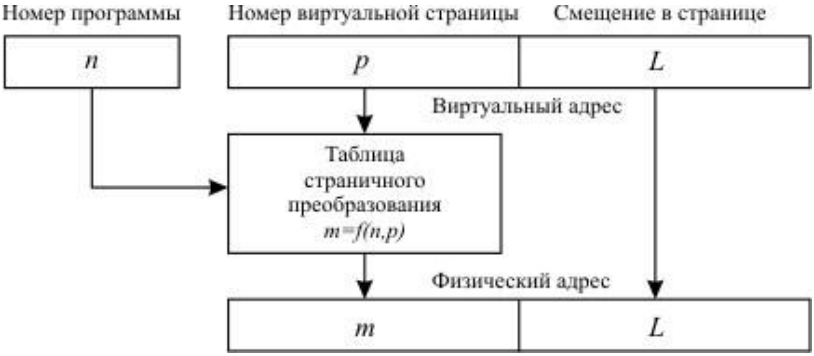


Рис. 10. Преобразование виртуального адреса в физический

оперативную *память* $V_{ОЗУ} = 3$ и *адресное пространство*, предполагающее *разбиение* на страницы объемом $V_{стр} = 1$. Каждая *программа*, в свою *очередь*, разбивается на *виртуальные страницы* того же объема. Пусть *коэффициент мультипрограммирования* данной ЭВМ равен четырем, т. е. на компьютере могут одновременно выполняться до четырех программ. Переключение между программами происходит через $t_k = 1$. *Время выполнения* каждой страницы любой программы составляет $t = 2t_k$. Полагаем, что страницы программ загружаются в оперативную *память* по мере их необходимости и, по возможности, в свободные области *ОЗУ*. Если вся *память* занята, то новая страница замещает ту, к которой дольше всего не было обращений (механизм замещения *LRU*).

Таблица страничного преобразования хранится в оперативной памяти. В связи с этим каждое обращение программы к памяти за командой или за операндом требует дополнительного обращения к оперативной памяти для *страничного преобразования*, что существенно снижает *производительность* компьютера. Для уменьшения влияния этого фактора используются различные подходы. Основной из них – метод, при котором после первого преобразования номера *виртуальной страницы* полученный номер *физической страницы* запоминается во внутренних служебных регистрах микропроцессора. При очередном обращении к памяти аппаратными средствами микропроцессора осуществляется проверка того, было ли уже обращение к данной *виртуальной странице*. Если было, то номер соответствующей ей *физической страницы* уже находится в микропроцессоре. В противном случае преобразование выполняется обычным образом с обращением к оперативной памяти. Так как *программа* может достаточно долго обращаться к адресам, находящимся в пределах одной страницы, такой подход существенно сокращает время на *страничное преобразование*.

5.6. Система управления памятью в компьютере на базе 32-разрядного микропроцессора

В компьютере на основе 32-разрядного микропроцессора при работе в так называемом защищенном режиме, поддерживающем *мультипрограммирование* и обеспечивающем адресацию операндов в максимально возможном для данной архитектуры диапазоне до 2^{32} *байт*, *виртуальная память* организуется на основе сегментно-страничного представления памяти. При этом *память* разбивается на *сегменты* переменной длины, выделяемые пользователю под *размещение* его программ и данных. *Сегменты*, в свою очередь, делятся на страницы фиксированной длины (4К = 2^{12} *байт*), используемые системой управления памятью для ее виртуализации.

Начало каждого сегмента устанавливается операционной системой через соответствующий сегментный *регистр* и скрыто от пользователя. *Пользователь* пишет свои программы в адресах относительно начала сегмента, полагая, что он располагает сегментом максимально возможной для данной архитектуры длины (2^{32} *байт*). *Аппаратные средства* микропроцессора сначала проводят сегментное преобразование адреса, а затем – страничное.

Механизм формирования физического адреса при сегментно-страничной организации памяти показан на рис. 11.

Основой получения физического адреса, выдаваемого на адресную шину микропроцессора, служит *логический адрес*. Он состоит из двух частей: ***селектора***, являющегося идентификатором сегмента, и смещения в сегменте.

Смещение в сегменте (32 разряда) – эффективный *адрес* вычисляется по задаваемому в команде режиму адресации операнда и является виртуальным адресом операнда. При обращении к команде в качестве смещения выступает *значение* регистра-указателя команд.

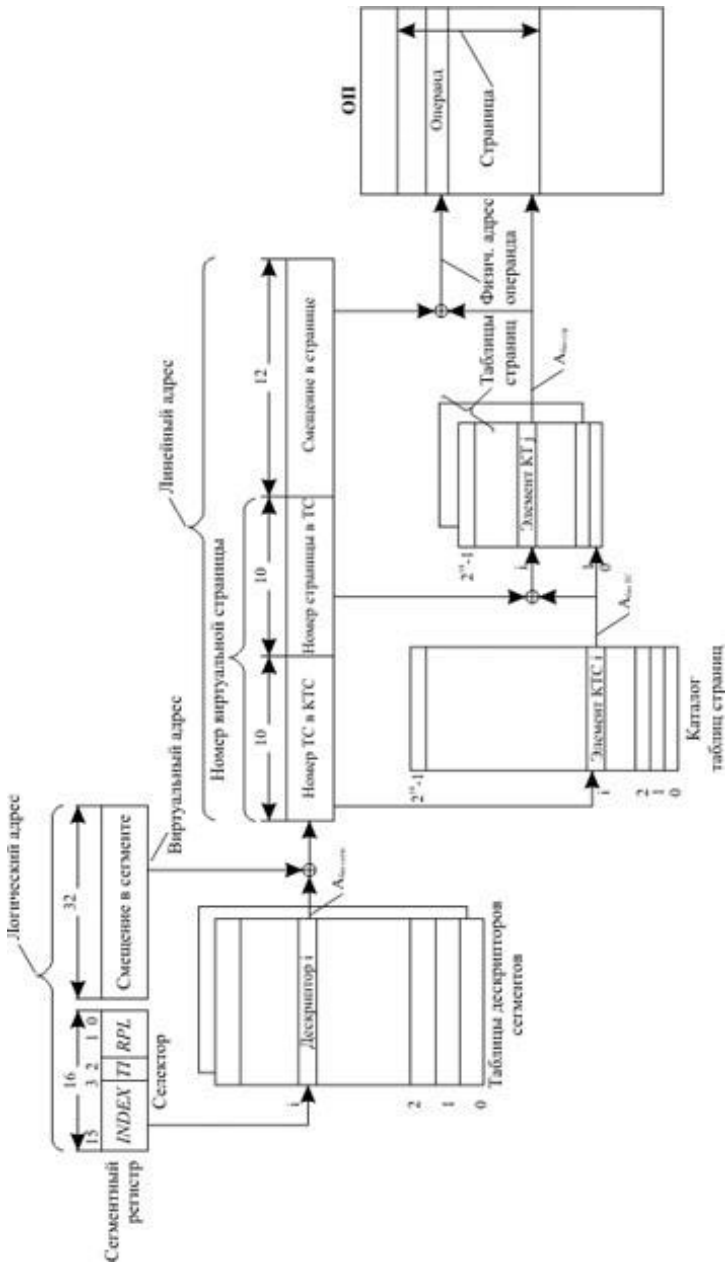


Рис. 11. Формирование физического адреса при сегментно-страничной организации памяти в 32-разрядном микропроцессоре

Селектор размещается в сегментном регистре (см. рис. 11). Основная его часть представляет собой номер (INDEX), по которому в одной из специальных таблиц дескрипторов можно найти *дескриптор (описатель)* данного сегмента. Вид используемой таблицы определяется битом TI (*table indicator*) *селектора*. *Селектор* содержит также двухразрядное *поле* RPL, используемое при организации защиты памяти по привилегиям.

Дескриптор (рис. 12) содержит сведения о сегменте. В одном из его полей содержится базовый *адрес* сегмента. В остальных полях записана дополнительная *информация* о сегменте: *длина*, допустимый уровень прав доступа к данному сегменту с целью защиты находящейся в нем информации, тип сегмента (сегмент кода, сегмент данных, специальный системный сегмент и т. д.) и некоторые другие атрибуты.

Рис. 12. Структура дескриптора сегмента

Сумма полученного из *дескриптора* базового адреса сегмента и вычисленного смещения дает линейный *адрес* операнда, который при включенном механизме *страничного преобразования* представляет собой номер *виртуальной страницы* (старшие 20 разрядов) и смещение операнда в странице (младшие 12 разрядов линейного адреса в соответствии с объемом страницы 4 Кбайт).

При преобразовании номера *виртуальной страницы* в номер физической используются следующие системные объекты: *ка-*

Рис. 13. Элемент каталога таблиц страниц (таблицы страниц)

талог таблиц страниц (КТС) и *таблицы страниц (ТС)*. Структуры этих таблиц сходны между собой (рис. 13).

Преобразование проводится в два этапа.

Сначала по разрядам A31–A22 линейного адреса в *КТС* выбирается нужный элемент. *Каталог таблиц страниц* всегда присутствует в оперативной памяти и содержит указания по размещению *таблицы страниц*, относящейся к тому или иному процессу. Элемент *КТС* содержит: адрес начала *таблицы страниц*; бит присутствия (P) *таблицы страниц* в оперативной памяти; бит разрешения чтения/записи (R/W); бит защиты страницы (пользователь/супервизор (U/S) и некоторые другие атрибуты.

После получения из выбранного элемента *КТС* начального адреса *таблицы страниц* происходит обращение к *ТС*. В выбранной *таблице страниц* находится элемент, номер которого определяется разрядами A21–A12 линейного адреса. Структура элемента *таблицы страниц* аналогична структуре элемента *КТС*. Элемент *ТС* в соответствующем *поле* содержит *адрес* начала требуемой *физической страницы* и другие атрибуты, аналогичные элементу *КТС*. При P = 0 возникает *прерывание*, необходимая страница подкачивается в оперативную память, ее *адрес* заносится в соответствующий элемент *ТС*, и *команда* выполняется повторно.

Сокращение потерь времени при использовании сегментно-страничной организации памяти в персональной ЭВМ. Преобразование *логического адреса* в физический при сегментно-страничной организации памяти требует, как минимум, трех обращений к системным таблицам, расположенным в оперативной памяти (таблице дескрипторов, *КТС* и *ТС*). Это может привести к существенному снижению производительности компьютера. Механизм сокращения потерь времени на такое преоб-

разование основывается на том факте, что изменение состояния сегментных регистров производится относительно редко, например, при переключении ЭВМ на новую задачу, а новое *страничное преобразование* требуется лишь при выходе программы за пределы загруженной в оперативную память страницы.

При сегментном преобразовании адреса после первого считывания *дескриптора* из таблицы дескрипторов, расположенной в оперативной памяти (например, после изменения состояния сегментного регистра при переключении на новую задачу), он запоминается в программно-недоступных («теневых») регистрах микропроцессора (рис. 14). При последующих обращениях к данному сегменту используется *дескриптор* из «теневого» регистра без обращения к ОП, поэтому на его вызов требуется минимальное время. Так как состояние сегментных регистров меняется относительно редко, то такой подход приводит к значительной экономии времени при сегментном преобразовании адреса.

Рис. 14. Сохранение дескрипторов сегментов в «теневых» регистрах микропроцессора

	Тег (A31–A15)	Физический адрес начала страницы	Атрибуты страницы	
Блок 0				Строка 0
				Строка 1
				Строка 2
				Строка 3
Блок 1				Строка 0
				Строка 1
				Строка 2
				Строка 3
…	…	…	…	…
Блок 7				Строка 0
				Строка 1
				Строка 2
				Строка 3

Рис. 15. Формат буфера ассоциативной трансляции адреса страницы

При *страничном преобразовании* номера *виртуальной страницы* в номер *физической страницы* используется ***кэш-буфер ассоциативной трансляции*** (TLB), содержащий физические адреса 32 наиболее активно используемых страниц (рис. 15) и расположенный непосредственно в микропроцессоре.

Номер *виртуальной страницы* представляет собой старшие 20 разрядов линейного адреса, полученного при сегментном преобразовании (A31–A12). По младшим разрядам (A14–A12) этого номера выбирается блок в *буфере ассоциативной трансляции*. Содержимое поля тэгов каждой из четырех строк этого блока ассоциативным образом (одновременно) сравнивается с разрядами

(A31–A15) линейного адреса. Если значения для одной из строк выбранного блока совпали, значит, номер этой *виртуальной страницы* уже преобразовывался в номер *физической страницы* и результат этого преобразования находится в найденной строке TLB. Если сравнение не было успешным, то преобразование номера *виртуальной страницы* в номер физической проходит обычным образом через обращения к каталогу таблиц страниц и к таблице страниц, а полученное значение заносится в TLB. При этом в поле тэгов заносятся старшие 17 разрядов линейного адреса этой страницы (A31–A15). Если нет свободной строки в блоке, определяемом разрядами A14–A12 линейного адреса, то из блока вытесняется строка, информация в которой дольше всего не использовалась (механизм LRU).

ГЛАВА 6
ИНТЕРФЕЙСЫ ЭВМ

Согласование между отдельными блоками выполняется с помощью устройств, называемых аппаратными интерфейсами.

Стандарты на аппаратные интерфейсы называются протоколами. Аппаратные интерфейсы разделятся на *последовательные* и *параллельные*.

Последовательный интерфейс обеспечивает передачу данных последовательно, бит за битом, поэтому имеет малую скорость передачи данных и простое устройство. Такие интерфейсы используют для подключения медленных устройств, а также в тех случаях, когда нет ограничений на продолжительность обмена данными, например, для подключения клавиатуры и мыши. Скорость передачи данных через последовательный интерфейс измеряется в битах в секунду.

Параллельный интерфейс обеспечивает передачу данных одновременно группами бит, что повышает скорость передачи данных. Количество бит группе называется разрядностью интерфейса. Существуют 8-, 16-, 32- и 64-разрядные интерфейсы. Они имеют более сложное устройство, чем последовательные интерфейсы. Их применяют тогда, когда важна скорость передачи данных: для подключения печатающих устройств, устройств ввода графических данных, устройств записи на внешние запоминающие устройства и т. п. Производительность параллельных интерфейсов измеряется в байтах в секунду.

6.1. Шинная организация ввода/вывода

Шинная организация ввода/вывода двумя преимуществами: низкой стоимостью и универсальностью. К общей шине легко подключаются новые устройства ввода/вывода. А одни и те же

периферийные устройства можно использовать в компьютерах разной архитектуры, но использующих однотипную шину.

Главным недостатком организации с общей шиной является то, что она ограничивает как максимальную пропускную способность системы ввода/вывода, так и ее масштабируемость. Количество внешних устройств, которые одновременно могут быть подключены к общей шине, фактически (физически) весьма ограниченно. В серверах, где ввод/вывод осуществляется очень часто, в системе применяется множество шин, способных удовлетворить необходимые запросы.

Одна из существенных особенностей шинной организации связана с тем, что любая общая шина является широковещательной (*broadcast*) средой передачи информации. Так, устройство может передавать информацию сразу нескольким или всем остальным устройствам на шине.

Рассмотрим несколько подробнее общие шины, имеющие широкое распространение.

В базовой версии (IEEE P1386.1) PCI представляла собой 32-битную синхронную шину с тактовой частотой 33 МГц и максимальной пропускной способностью 133 Мбайт/с. Адресное пространство составляло 32 бита (4 ГБ), использовались сигнальные уровни 3,3 и 5 вольт.

Базовая спецификация нового стандарта *PCI Express*, пришедшего на замену PCI, была опубликована 22 июля 2002 г. *PCI Express* представляет собой последовательную системную шину общего назначения, организованную как совокупность независимых самостоятельных последовательных каналов передачи данных. Каждый канал состоит из двух дифференциальных сигнальных пар (уровень напряжения 0,8 В), при этом используется избыточное защищенное от помех кодирование – каждый байт при передаче представляется десятью битами.

Пропускная способность *PCI Express* составляет 2,5 Гбит/с для одного канала в каждом направлении одновременно. Стандартизированы 1-, 2-, 4-, 8-, 16- и 32-канальные варианты. При передаче данные передаются параллельно (но не синхронно) по всем доступным каналам.

Вся контрольная информация передается по тем же линиям, что и данные. Стандарт предусматривает использование альтернативных носителей сигнала, таких как оптические волноводы.

6.2. Шины «малого» интерфейса

Параллельная шина ATA (*advanced technology attachment*) или *IDE* (*integrated drive electronics*, присоединение по продвинутой технологии) – была самым массовым интерфейсом, применяемым для устройств хранения.

Последовательная шина Serial ATA (SATA) – преемник своего параллельного предшественника ATA.

В 2000-х гг. был осуществлен переход на стандарт *Serial ATA (SATA)*, который, являясь логически совместимым с IDE, предусматривал не параллельную, а последовательную передачу данных. Работа SATA в полудуплексном режиме позволяла достичь пропускной способности до 1,5 Гбайт/с.

Для SATA повысилась скорость обмена с устройством, была решена проблема одновременной работы с несколькими устройствами, используется расширенная адресация. Кабели и разъемы последовательного интерфейса SATA компактны, «горячее» подключение реализуется естественным образом: в SATA каждое устройство подключается к собственному порту хост-контроллера, а не к общей шине. Поначалу интерфейс SATA предназначался только для подключения внутренних устройств, позже он стал и внешним интерфейсом. В SATA-II появились новые элементы: мультиплексоры, позволяющие подключать к одному порту

хоста несколько устройств, и селекторы портов, позволяющие подключать одно устройство (или мультиплексор) к двум хостам (но работать устройство может только с одним хостом).

Шина SCSI (*Small Computer System Interface)* – главный конкурент шины ATA для устройств хранения – является универсальным. Она предназначена для подключения устройств различных классов: дисковых, ленточных, оптических и других устройств хранения, принтеров, сканеров, коммуникационных и прочих устройств. В интерфейсе SCSI определена идеология взаимодействия хоста с устройствами, эффективная при работе с множеством устройств в многозадачных системах.

Интерфейс SCSI поначалу существовал только в виде параллельной шины. Согласно современным стандартам, протоколы интерфейса SCSI позволяют работать не только с параллельной шиной или последовательным интерфейсом SAS (недавно появившийся вариант SCSI), но и с другими средствами передачи данных: последовательной шиной IEEE 1394 (FireWire), Fibre Channel, SSA, а также любыми IP-сетями – iSCSI. Все варианты SCSI пригодны как для внутреннего, так и для внешнего подключения. Они имеют поддержку «горячего» подключения-отключения, необходимую в больших и ответственных системах хранения данных. Предел адресации данных для устройств SCSI первоначально составлял 2 Тбайт (32-разрядная адресация блоков), позже ввели возможность 64-разрядной адресации блоков (объем хранения – до 9 444 732 965 739 290 427 392 байт).

К настоящему времени стандарт SCSI исчерпал свои возможности. Логичным и естественным последовательным расширением технологии параллельного интерфейса SCSI является стандарт последовательного интерфейса ***Serial Attached SCSI (SAS)***.

Serial Attached SCSI (SAS). Стандарт SAS опирается на электрические и физические характеристики интерфейса

Serial ATA. Архитектурная схожесть с SATA не мешает SAS обладать наиболее востребованными чертами SCSI.

SAS – это полнодуплексный SATA с поддержкой двух портов, больших возможностей адресации, высокой надежностью, производительностью и логической совместимостью со SCSI.

В отличие от SCSI, SAS имеет последовательную архитектуру с непосредственным подключением контроллера к ВУ. SAS поддерживает до 128 ВУ различных типов, совместно подключенных более тонкими и длинными, нежели в случае SCSI, кабелями. Полнодуплексный сигнальный последовательный интерфейс SAS обеспечивает обмен данными на скорости 3–10 Гбайт/с на порт.

Среди преимуществ данного интерфейса стоит отметить в настоящее время следующие: использование общей шины всеми устройствами; последовательный протокол передачи данных, используемый SAS, позволяет задействовать меньшее количество сигнальных линий; отсутствует необходимость в терминации шины; практически неограниченное число подключаемых устройств; более высокая пропускная способность (до 12 Гбит/с). В будущих реализациях протокола SAS предполагается поддерживать скорость обмена данными до 24 Гбит/с.; возможность подключения к контроллеру SAS накопителей с интерфейсом Serial ATA.

FireWire (IEEE 1394). Интерфейс (высокоскоростной и последовательный), предназначенный для передачи информации со скоростью до 400 Мбит/с. Этот интерфейс также может применяться для подсоединения к компьютеру сканера.

Теперь рассмотрим широко распространенную универсальную последовательную шину USB.

USB (***Universal Serial Bus)*** – универсальная последовательная шина. USB является промышленным стандартом расширения архитектуры ПК (персонального компьютера), ориентиро-

ванным на интеграцию с телефонией и устройствами бытовой электроники.

Архитектура USB определялась следующими критериями:

– дешевое решение, поддерживающее скорость передачи до 12 Мбит/с;

– полная поддержка в реальном времени передачи аудио- и сжатых видеоданных;

– гибкость протокола для смешанной передачи изоморфных данных и асинхронных сообщений.

С появлением шин USB и FireWire в качестве характеристики интерфейса стала фигурировать топология соединения. Для интерфейсов RS-232C и Centronics практически всегда применялась двухточечная топология «компьютер – устройство» или «компьютер – компьютер». Интерфейсные шины USB и FireWire реализуют древовидную топологию, в которой внешние устройства могут быть как оконечными, так и промежуточными (разветвителями). Эта топология позволяет подключать множество устройств к одному порту USB или FireWire.

Изначально в стандарт USB было заложено несколько удачных принципов:

1. Универсальный интерфейс, по которому может быть подключено любое из устройств: сканер, принтер, модем, клавиатура, мышь, аудиосистема, DVD-CD-приводы, жесткий диск, флешнакопитель и др. Определенные классы подключаемых устройств могут получать питание от встроенного в USB источника 5 В.

2. Динамическое подключение ВУ, не требующее установки дополнительных плат, отдельных драйверов и выключения компьютера.

3. Возможность каскадного подключения: некоторые устройства могут выступать как узлы (концентраторы), к которым можно подключить дополнительные устройства.

Итак, USB как шина обеспечивает обмен данными между хост-компьютером и множеством одновременно доступных периферийных устройств. Распределение пропускной способности шины между подключенными устройствами планируется хостом и реализуется им с помощью посылки маркеров. Шина позволяет подключать, конфигурировать, использовать и отключать устройства во время работы хоста и самих устройств – динамическое («горячее») подключение и отключение.

Интерфейс USB обеспечивает поддержку протокола USB, выполнение стандартных операций (конфигурирование и сброс) и стандартное представление информации, описывающей устройство.

USB OTG (от *On-The-Go* – «на ходу») – дальнейшее расширение спецификации USB 2.0, предназначенное для соединения периферийных USB-устройств друг с другом без необходимости подключения к компьютеру. Например, цифровую камеру можно подключить к фотопринтеру напрямую, если они оба поддерживают стандарт USB OTG.

USB wireless позволяет организовать беспроводную связь с высокой скоростью передачи информации (до 480 Мбит/с на расстоянии 3 м и до 110 Мбит/с на расстоянии 10 м).

USB 3.0 (его также называют SuperSpeed USB). Спецификацией USB 3.0 предусмотрен режим SuperSpeed со скоростью передачи данных до 5 Гбит/с (640 Мбайт/с), т. е. более чем в 10 раз превышающей ту, что предусмотрена спецификацией USB 2.0.

При всем разнообразии интерфейсов в большинстве случаев сейчас используется последовательный вариант (SATA и SAS). Для внешних устройств широко применяются интерфейсы USB и FireWire.

6.3. Типы устройств и интерфейсы: SAS, SCSI, SATA, IDE (ATA, PATA)

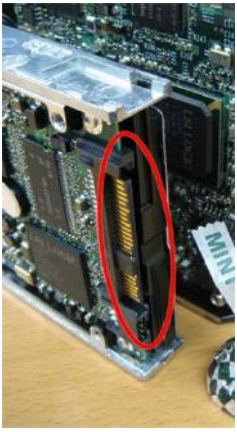
Серийно выпускаемые жесткие диски, предназначенные для внутренней установки, могут использовать интерфейсы ATA (он же IDE и PATA), SATA, eSATA, SCSI, SAS, FireWire, SDIO и Fibre Channel.

Интерфейсы жестких дисков не в меньшей степени, чем физические параметры накопителей, влияют на многие рабочие характеристики накопителей и на их производительность. В частности, интерфейсы накопителей определяют такие их параметры, как скорость обмена данными между жестким диском и материнской платой, количество устройств, которые можно подключить к компьютеру, возможность создания дисковых массивов, возможность горячего подключения, поддержка технологий NCQ (функцию очереди команд) и AHCI и т. д. Также от интерфейса жесткого диска зависит, какой кабель, шнур или переходник для его подключения к материнской плате потребуется.

SCSI – интерфейс, разработанный для объединения на одной шине различных по своему назначению устройств, таких как сканер, принтер, модем, клавиатура, мышь, аудиосистема, DVD-CD-приводы, жесткий диск, флеш-накопитель и др.

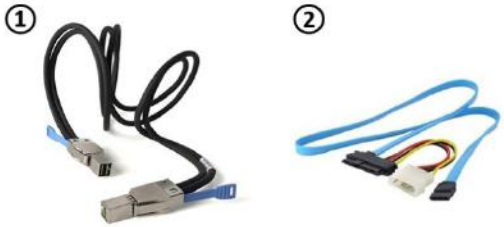
В качестве его замены был разработан более совершенный, последовательный стандарт SAS.

SAS-интерфейс разработан для обмена данными с такими устройствами, как жесткие диски, накопители на оптическом диске и т. д. SAS использует последовательный интерфейс для работы с непосредственно подключаемыми накопителями DAS (англ. *direct-attached storage devices*). SAS совместим с интерфейсом SATA. Хотя SAS использует последовательный интерфейс в отличие от параллельного интерфейса, используемого традиционным SCSI, для управления SAS-устройствами по-прежнему используются команды SCSI. Разъемы SAS гораздо меньше разъемов традиционного параллельного интерфейса SCSI, что позволяет использовать разъемы SAS для подключения компактных накопителей размером 2,5 дюйма. SAS поддерживает передачу информации со скоростью 3–10 Гбит/с.



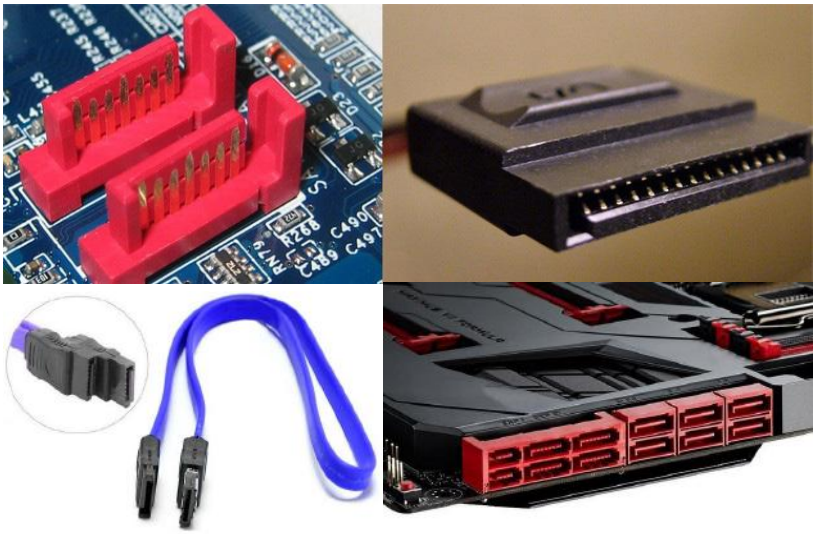
Разъемы SAS могут иметь различную форму и размер, в зависимости от его типа (внешний или внутренний) и от версий SAS. Ниже представлены внутренний разъем SFF-8482 (слева) и внешний разъем SFF-8644 с кабелем (справа), разработанный для SAS-3.

Несколько примеров внешнего вида шнуров и переходников SAS: шнур HD-Mini SAS (слева) и шнур-переходник SAS-Serial ATA (справа).



SATA. SATA использует 7-контактный разъем вместо 40-контактного разъема у PATA, вместо 80-контактного разъема у IDE. SATA-кабель имеет меньшую площадь, за счет чего уменьшается сопротивление воздуха, обдувающего комплектующие компьютера; улучшается охлаждение системы. SATA-кабель за счет своей формы более устойчив к многократному подключению. Питающий шнур SATA также разработан с учетом многократных подключений. Разъем питания SATA подает три напряжения питания: +12, +5 и +3,3 В. Однако современные устройства могут работать без напряжения +3,3 В, что дает возможность использовать пассивный переходник со стандартного разъема питания IDE на SATA. Ряд SATA-устройств поставляется с дву-

мя разъемами питания: SATA и Molex. Стандарт SATA отказался от традиционного для PATA подключения по два устройства на шлейф, т. е. каждому устройству полагается отдельный кабель, что снимает проблему невозможности одновременной работы устройств, находящихся на одном кабеле (и возникающих отсюда задержек), уменьшает возможные проблемы при сборке (проблема конфликта Slave/Master-устройств для SATA отсутствует), устраняет возможность ошибок при использовании нетерминированных PATA-шлейфов. Стандарт SATA предусматривает горячую замену устройств и функцию очереди команд (NCQ).



Суть технологии NCQ состоит в том, что она поддерживает систему упорядочивания операций чтения/записи, поступающих к нескольким накопителям, установленным в системе. Таким образом, NCQ способна значительно повысить производительность работы накопителей, в особенности массивов жестких дисков. Однако для использования NCQ необходимы поддерж-

ка со стороны операционной системы, поддержка технологии со стороны жесткого диска, а также поддержка технологии со стороны хост-адаптера материнской платы. Практически все адаптеры, поддерживающие AHCI, поддерживают и NCQ. Кроме того, NCQ поддерживают и некоторые старые проприетарные адаптеры.

Вывод по типам устройств. Интерфейсы жестких дисков представляют собой средство для связи между этими накопителями и материнской платой. На сегодняшний день существует немало различных типов интерфейсов жестких дисков, каждый из которых имеет свои достоинства, недостатки и характерные особенности.

Выбор современного жесткого диска во многом определяется не только его внутренними характеристиками, такими, как емкость, объем кэш-памяти, скорость доступа и вращения, но и тем интерфейсом, для которого он был разработан.

6.4. Система USB

С точки зрения пользователя привлекательны такие черты USB:

– легко реализуемое расширение периферии компьютера;

– открытие новых классов устройств, расширяющих возможности компьютера;

– интеграция в технологию выпускаемых устройств;

– простота кабельной системы подключений;

– изоляция подробностей электрических подключений от пользователя;

– самоидентифицирующаяся периферия, автоматическая связь устройств с драйверами и конфигурирование;

– возможность динамического подключения и реконфигурирования периферии.

Структура и взаимодействие системы USB. Устройства USB могут являться хабами, «функциями» или их комбинацией. Хаб обеспечивает дополнительные точки подключения устройств к шине. «Функции» USB предоставляют системе дополнительные возможности, например, подключение к ISDN, цифровой джойстик, акустические колонки с цифровым интерфейсом и т. д. Многие устройства, подключаемые к USB, имеют в своем составе и «функции», и хабы.

Устройство USB должно иметь интерфейс USB, обеспечивающий поддержку протокола USB, выполнение стандартных операций (конфигурирование и сброс) и стандартное представление информации, описывающей устройство. Работой всей системы USB управляет хост-контроллер, являющийся программно-аппаратной подсистемой хост-компьютера.

Физическое соединение устройств осуществляется по топологии многоярусной звезды. Центром каждой звезды является хаб, каждый кабельный сегмент соединяет две точки – хаб с другим хабом или хаб с «функцией». В системе USB имеется только один хост-контроллер, расположенный в вершине пирамиды устройств и хабов USB. Хост-контроллер интегрируется с корневым хабом (*root hub*), обеспечивающим одну или несколько точек подключения – портов. Контроллер USB, входящий в состав чипсетов многих современных системных плат, обычно имеет двухпортовый хаб.

Логически, устройство, подключенное к любому хабу и сконфигурированное, может рассматриваться как подключенное напрямую к хост-контроллеру.

«Функции» представляют собой устройства USB, способные принимать и передавать данные или управляющую информацию по шине. Физически в одном корпусе может быть несколько

«функций» со встроенным хабом, обеспечивающим их подключение к одному порту.

Каждая «функция» предоставляет конфигурационную информацию, описывающую возможности и требования к ресурсам устройства. Перед использованием «функция» должна быть сконфигурирована хостом – ей должна быть выделена полоса в канале, выбраны специфические опции конфигурации.

Хаб – ключевой элемент системы *plug and play* в архитектуре USB. Хаб является кабельным концентратором, точки подключения называются портами хаба. Каждый хаб преобразует одну точку подключения в их множество. Архитектура подразумевает возможность соединения нескольких хабов.

У каждого хаба имеется один восходящий порт (upstream port), предназначенный для подключения к хосту или к хабу верхнего уровня. Остальные порты являются нисходящими (downstream) и предназначены для подключения функций и хабов нижнего уровня. Хаб может распознавать подключение или отключение устройств к этим портам и управлять подачей питания на их сегменты. Каждый из этих портов индивидуально может быть разрешен или запрещен и сконфигурирован на полную или ограниченную скорость обмена. Хаб обеспечивает изоляцию сегментов с низкой скоростью от высокоскоростных.

Хабы могут иметь возможность управления подачей питания на нисходящие порты, в которых предусмотрена управляемая установка ограничения на ток, потребляемый каждым портом.

Система USB разделяется на три уровня с определенными правилами взаимодействия. Устройство USB делится на интерфейсную часть, часть устройства и функциональную часть. Хост тоже делится на три части – интерфейсную, системную и программное обеспечение устройства. Каждая часть отвечает только

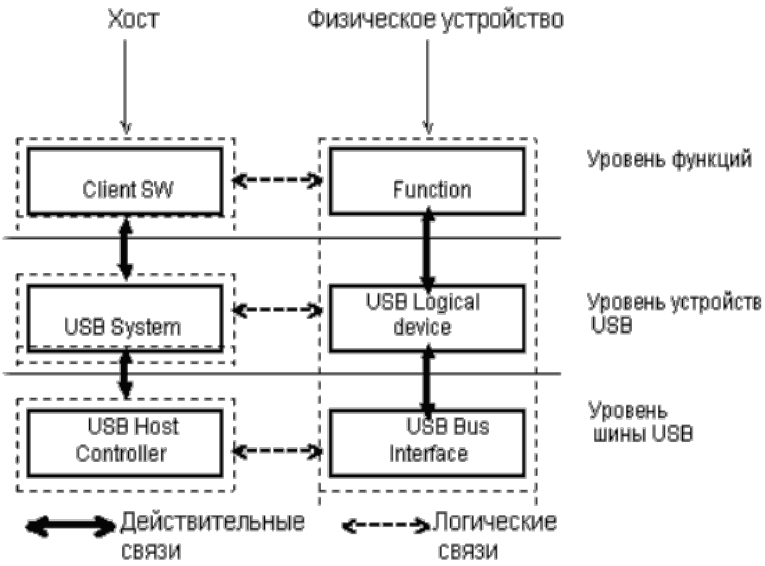


Рис. 16. Взаимодействие компонентов USB

за определенный круг задач, взаимодействие между ними показано на рисунке 16.

На рисунке 16 можно увидеть, что физическое устройство USB – устройство на шине, выполняющее функции, интересующие пользователя. Client SW – программное обеспечение, соответствующее конкретному устройству, исполняемое на хост-компьютере, может являться составной частью ОС или специальным продуктом. USB System SW – системная поддержка USB операционной системой, независимая от конкретных устройств и клиентского ПО. USB Host Controller – аппаратные и программные средства, обеспечивающие подключение устройств USB к хост-компьютеру.

Физический интерфейс системы USB. Информационные сигналы и питающее напряжение 5 В передаются по четырехпроводному кабелю. Для сигнала используется дифференциальный способ передачи по двум проводам D+ и D−. Уровни сигналов пе-

редатчиков в статическом режиме должны быть ниже 0,3 В (низкий уровень) или выше 2,8 В (высокий уровень). Приемники должны выдерживать входное напряжение в пределах −0,5...+3,8 В. Передатчики должны иметь возможность перехода в высоко импедансное состояние для обеспечения двунаправленной полудуплексной передачи данных по одной паре проводов.

Передача по двум проводам не ограничивается лишь дифференциальными сигналами. Кроме дифференциального приемника, каждое устройство имеет и линейные приемники сигналов D+ и D−, а передатчики этих линий управляются индивидуально. Это позволяет различать множество состояний линии, используемых для организации аппаратного интерфейса. Состояния Diff0 и Diff1 определяются по разности потенциалов на линиях D+ и D− более 200 мВ при условии, что на одной из них потенциал выше порога срабатывания VSE. Состояние, при котором на обоих входах D+ и D− присутствует низкий уровень, называется линейным нулем (*SE0 – single-ended zero*). Интерфейс определяет следующие состояния:

– Data J State и Data K State – состояния передаваемого бита (определяются через состояния Diff0 и Diff1);

– Idle State – пауза на шине;

– Resume State – сигнал «пробуждения» для вывода устройства из спящего режима;

– Start of Packet (SOP) – начало пакета (переход из «Idle» в «K»);

– End of Packet (EOP) – конец пакета;

– Disconnect – устройство отключено от порта;

– Connect – устройство подключено к порту;

– Reset – сброс устройства.

Состояния определяются сочетаниями дифференциальных и линейных сигналов, для полной и низкой скоростей состояния

Diff0 и Diff1 имеют противоположное назначение. В декодировании состояние Disconnect, Connect и Reset принимается во внимание и время нахождения линий (более 2,5 мс) в определенных состояниях.

Шина имеет два режима передачи. Полная скорость передачи сигналов USB составляет 12 Мбит/с. Низкая скорость передачи сигналов USB составляет 1,5 Мбит/с.

Для полной скорости используется экранированная витая пара с импедансом 90 Ом и длиной сегмента до 5 м. Для низкой скорости используется невитой и неэкранированный кабель при длине сегмента до 3 м.

Одна и та же система может использовать оба режима, переключение для устройств осуществляется прозрачно. Низкая скорость предназначена для работы с небольшим количеством устройств, не требующих высокой пропускной способности канала.

Скорость, используемая устройством, подключенным к конкретному порту, определяется хабом по уровням сигналов на линиях D+ и D−, смещаемых нагрузочными резисторами R2 приемопередатчиков (рис. 8, 9). Сигналы кодируются по методу NRZ(I) (*non-return-to-zero inverted*) – при переходе сигнала из 0 в 1 сигнал NRZ(I) не изменяется, а при переходе из 1 в 0 – изменяется на противоположный. Каждому пакету предшествует поле SYNC, позволяющее приемнику настроиться на частоту передатчика.

Кроме сигнальной пары кабель имеет линии VBus и GND для передачи питающего напряжения 5 В к устройствам.

Питание устройства USB возможно как от кабеля, так и от собственного блока питания.

Хост обеспечивает питанием непосредственно подключенные к нему устройства. Каждый хаб обеспечивает питание устройств, подключенных к его нисходящим портам.

USB имеет развитую систему управления энергопотреблением. Хост-компьютер может иметь собственную систему управления энергопотреблением, к которой логически подключается одноименная система USB. Программное обеспечение USB, взаимодействуя с этой системой, поддерживает такие события, как приостановка (*suspend*) или восстановление (*resume*). Кроме того, устройства USB могут сами являться источниками событий, отрабатываемых системой управления энергопотреблением.

Модель передачи данных системы USB. С точки зрения данных, каждое устройство USB представляет собой ряд независимых конечных точек, с которыми хост-контроллер может обмениваться информацией. Конечные точки описываются следующими параметрами:

– требуемая частота доступа к шине и допустимые задержки обслуживания;

– требуемая полоса пропускания канала;

– номер точки;

– требования к обработке ошибок;

– максимальные размеры передаваемых и принимаемых пакетов;

– тип обмена;

– направление обмена (для сплошного и изохронного обмена).

Каждое устройство обязательно имеет конечную точку с номером 0, используемую для инициализации и общего управления логическим устройством, а также опроса его состояния. Эта точка всегда сконфигурирована при включении питания и подключении устройства к шине и поддерживает передачи типа «управление».

Кроме нулевой точки, устройства-функции могут иметь дополнительные точки, реализующие полезные обмены данными.

Низкоскоростные устройства могут иметь до двух дополнительных точек, полноскоростные – до 16 точек ввода и до 16 точек вывода. Все эти точки не могут быть использованы до их конфигурирования.

Канал (*pipe*) – модель передачи данных между хост-контроллером и конечной точкой устройства. Имеются два типа каналов – потоки и сообщения. Каналы организуются при конфигурировании устройств USB. Один канал сообщений (Control Pipe 0), по которому передается информация конфигурирования, управления и состояния, обязательно существует для каждого включенного устройства.

Поток доставляет данные от одного конца канала к другому, он всегда однонаправленный. Один и тот же номер конечной точки может использоваться для двух поточных каналов – ввода и вывода. Поток может использовать следующие типы обмена – сплошной, прерывания и изохронный. Доставка всегда идет в порядке «первый вошел – первый вышел». Данные потока неструктурированные.

Сообщения имеют формат, определенный спецификацией USB. Обмен сообщениями отличается от потоков: хост отсылает запрос к конечной точке, после которого передается или принимается поток сообщения, за которым следует пакет с информацией о состоянии конечной точки. При обработке ошибок возможен сброс необработанных сообщений. Двусторонний обмен сообщениями адресуется к одной и той же конечной точке. Для доставки сообщений используется только обмен типа «управление».

Типы передачи данных системы USB. USB поддерживает несколько режимов связи, как однонаправленных, так и двунаправленных. Передача данных производится между ПО хоста и конкретной конечной точкой устройства. Устройство может

иметь несколько конечных точек, связь с каждой из них устанавливается независимо от других.

В архитектуре USB существуют четыре типа передаваемых данных:

1. Управляющие посылки (*control transfers*) используются для конфигурирования во время подключения и в процессе работы для управления устройствами. Протокол обеспечивает гарантированную доставку данных. Длина поля данных управляющей посылки не превышает 64 байт для полной скорости и 8 байт – для низкой.

2. Сплошные передачи (*bulk data transfer*) сравнительно больших пакетов без жестких требований ко времени доставки. Эти передачи занимают всю свободную полосу пропускания шины, не занятую другими классами передач. Пакеты имеют поле данных размером 8, 16, 32 или 64 байт. Приоритет этих передач самый низкий, они могут приостановиться при большой загрузке шины. Допускаются только на полной скорости передачи.

3. Прерывания (*interrupts*) – короткие (до 64 байт на полной скорости и до 8 – на низкой) передачи типа вводимых символов или координат. Прерывания имеют спонтанный характер и должны обслуживаться не медленнее, чем того требует устройство. Предел времени обслуживания устанавливается в диапазоне 1–255 мс для полной скорости и 10–255 мс для низкой.

4. Изохронные передачи – непрерывные передачи в реальном времени, занимающие предварительно согласованную часть пропускной способности шины и имеющие заданную задержку доставки. В случае обнаружения ошибки изохронные данные передаются без повтора – недействительные пакеты просто игнорируются.

Полоса пропускания шины делится между всеми установленными каналами. Выделенная полоса закрепляется за кана-

лом. Если установка нового канала требует такой полосы, которая не вписывается в уже существующее распределение, запрос на выделение канала отвергается.

Архитектура USB предусматривает внутреннюю буферизацию всех устройств. USB должна обеспечивать обмен с такой скоростью, чтобы задержка данных в устройстве, вызванная буферизацией, не превышала миллисекунд.

Изохронные передачи классифицируются по способу синхронизации конечных точек с системой: различают асинхронный, синхронный и адаптивный классы устройств, каждому из которых соответствует свой тип канала USB.

Протокол системы USB. Транзакции по USB состоят из трех пакетов. Каждая транзакция планируется и начинается по инициативе контроллера, который посылает пакет-маркер. Этот пакет описывает тип и направление передачи, адрес устройства и номер конечной точки.

В каждой транзакции возможен обмен между конечной точкой адресуемого устройства и хостом. Адресуемое маркером устройство USB распознает свой адрес и подготавливается к обмену. Источник данных передает пакет данных (или уведомление об отсутствии данных, предназначенных для передачи). После успешного приема пакета приемник посылает пакет подтверждения (или «рукопожатия»), подтверждающий прием информации. Планирование транзакций обеспечивает управление поточными каналами. На аппаратном уровне использование отказа от транзакции (NACK) при недопустимой интенсивности транзакций на шине предохраняет буферы от переполнения или от опустошения. Маркеры отвергнутых транзакций передаются вновь в свободное для шины время. Управление потоками позволяет гибко планировать обслуживание одновременных разнородных потоков данных.

Устойчивость к ошибкам обеспечивают следующие свойства USB:

1. Высокое качество сигналов, обеспечиваемое дифференциальными приемниками, передатчиками и экранированием кабелей.

2. Защита полей управления и данных CRC-кодами.

3. Обнаружение подключения и отключения устройств, конфигурирование ресурсов на системном уровне.

4. Самовосстановление протокола с использованием таймаута при потере пакетов.

5. Управление потоком для обеспечения изохронности и управления аппаратными буферами.

6. Независимость одних функций от неудачных обменов с другими функциями, обеспечиваемая конструкцией каналов.

Для обнаружения ошибок передачи каждый пакет использует контрольные поля CRC-кодов, позволяющие обнаруживать одиночные и двойные битовые ошибки. Аппаратные средства обнаруживают ошибки передачи, а контроллер производит трехкратную попытку передачи. Если эти попытки безуспешны, то сообщение об ошибке передается клиентскому ПО для программной обработки.

Конфигурирование системы USB. USB поддерживает подключение и отключение устройств во время работы шины. Нумерация устройств шины является постоянным процессом, отслеживающим динамические изменения физической топологии.

Все устройства USB подключаются через порты хабов. Хабы определяют подключение и отключение устройств к своим портам и сообщают состояние портов в ответ на запрос от контроллера. Хост разрешает работу порта и адресуется к устройству

через канал управления, используя свой нулевой адрес – USB default address. Все устройства адресуются этим адресом при начальном подключении или после сброса.

Хост определяет, является подключенное устройство хабом или функцией, и назначает ему уникальный адрес USB. Хост устанавливает с этим устройством канал управления, используя назначенный адрес и нулевой номер точки назначения.

Если новое устройство является хабом, то хост определяет подключенные к нему устройства, устанавливает каналы и назначает для них адреса. Если новое устройство является «функцией», уведомление о подключении предается диспетчером USB соответствующему ПО.

Когда устройство отключается, хаб автоматически запрещает использование соответствующего порта и сообщает об отключении контроллеру, который удаляет сведения о данном устройстве из всех структур данных. Если отключается хаб, то процесс удаления повторяется для всех подключенных к нему устройств.

Нумерация устройств, подключенных к шине, осуществляется динамически по мере подключения или отключения их питания без какого-либо вмешательства пользователя или клиентского ПО. Процедура нумерации выполняется следующим образом:

1. Хаб, к которому подключилось устройство, информирует хост о смене состояния своего порта ответом на опрос состояния. С этого момента устройство переходит в состояние «Attached» («присоединено»), а порт, к которому оно присоединено, в состояние *Disabled*.

2. Хост уточняет состояние порта.

3. Узнав порт, к которому подключилось новое устройство, хост дает команду сброса и разрешения порта.

4. Хаб формирует сигнал *Reset* для данного порта (10 мс) и переводит его в состояние *Enabled*. Подключенному устройству позволяется потреблять от шины ток питания в пределах 100 мА. Устройство переходит в состояние *Powered*, все его регистры переводятся в исходное состояние, и оно отзывается на обращение по нулевому адресу.

5. До тех пор, пока устройство не получит уникальный адрес, оно доступно по дежурному каналу, по которому хост-контроллер может определять максимально допустимый размер поля данных пакета.

6. Хост сообщает устройству его уникальный адрес, и оно переходит в состояние *Addressed*.

7. Хост считывает все конфигурации устройства, включая и заявленный ток потребления от шины.

8. Исходя из полученной информации, хост конфигурирует все имеющиеся конечные точки данного устройства, которое переводится в состояние *Configured*. Теперь хаб позволяет устройству потреблять от шины полный ток, заявленный в конфигурации. С точки зрения устройства оно становится готовым к использованию.

Когда устройство отделяется от шины, хаб уведомляет об этом хост и работа порта запрещается, а хост обновляет свою текущую топологическую информацию.

Хост-контроллер. Через хост-контроллер хост-компьютер общается с устройством. Хост-контроллер имеет следующее назначение:

– обнаружение подключения и отсоединения устройств USB;

– манипулирование потоком управления между устройствами и хостом;

– управление потоками данных;

– сбор информации о состоянии и статистике;

– обеспечение энергосбережения подключенными устройствами.

Системное ПО контроллера управляет взаимодействием между устройствами и их ПО, функционирующим на хост-компьютере. Области взаимодействия следующие:

– нумерация и конфигурация устройств;
– изохронные передачи данных;
– асинхронные передачи данных;
– управление энергопотреблением;
– информация об управлении устройствами и шиной.

По возможности, ПО USB в этих областях использует существующее системное ПО хост-компьютера.

Функции и хабы. Возможности шины USB позволяют ее использовать для подключения самых разнообразных устройств. Все устройства должны поддерживать нижеперечисленный набор общих операций.

Динамическое подключение и отключение подразумевает возможность этих действий в любой момент времени. Эти события отслеживаются хабом, который сообщает о них хост-контроллеру и выполняет сброс подключенного устройства. Устройство после сигнала сброса должно отзываться на нулевой адрес, при этом оно не сконфигурировано и не приостановлено.

После назначения адреса, за которое отвечает хост-контроллер, устройство должно отзываться только на свой уникальный адрес.

Конфигурирование устройств, выполняемое хостом, является необходимым шагом для их использования. Для конфигурации обычно используется информация, считанная из самого устройства. Устройство может иметь множество интерфейсов, каждому из них соответствует собственная конечная точка, пред-

ставляющая хосту функцию устройства. Кроме того, интерфейс в конфигурации может иметь альтернативные наборы характеристик, смена наборов поддерживается протоколом. Для поддержки адаптивных драйверов дескрипторы устройств и интерфейсов имеют поля класса, подкласса и протокола.

Передача данных возможна посредством одного или четырех типов передач. Для конечных точек, допускающих разные типы передач, после конфигурирования доступен только один из них.

Управление энергопотреблением является весьма развитой функцией USB. Для устройств, питающихся от шины, мощность является ограниченным ресурсом. Любое устройство при начальном подключении не должно потреблять ток больше 100 мА. Рабочий ток (не более 500 мА) заявляется в конфигурации, и, если хаб не сможет обеспечить устройству заявленный ток, то оно не конфигурируется.

Устройство USB должно поддерживать режим приостановки, в котором его потребляемый ток не должен превышать 500 мкА. Устройство должно автоматически приостанавливаться при прекращении деятельности шины.

Возможность удаленного пробуждения позволяет приостановленному устройству подать сигнал хост-компьютеру, который тоже может находиться в приостановленном состоянии. Возможность удаленного пробуждения описывается в конфигурации устройства, и при конфигурировании эта функция может быть и запрещена.

Хаб в USB выполняет функции коммутации сигналов и раздачи питающего напряжения, а также отслеживает состояние подключенных к нему устройств, уведомляя хост об изменениях. Хаб состоит из двух элементов – контроллера и повторителя. Повторитель представляет собой управляемый ключ, соединяю-

щий выходной порт со входным. Он также имеет средства поддержки сброса и приостановки передачи сигналов.

Нисходящие порты хабов могут находиться в следующих состояниях:

1. *Power Off* – состояние, в котором на порт не подается питание (возможно только для хабов, коммутирующих питание). Выходные буферы переводятся в высокоимпедансное состояние, входные сигналы игнорируются.

2. *Disconnected* – порт не передает сигналы ни в одном направлении, но способен обнаружить событие подключения устройства. Обнаружив подключение (по отсутствию состояния SE0 в течение 2,5 мкс), порт переходит в состояние *Disabled*, а по уровням входных сигналов он определяет скорость подключенного устройства.

3. *Disabled* – порт может передавать только сигнал сброса (по команде от контроллера), сигналы от порта, кроме обнаружения отключения, не воспринимаются. При обнаружении отключения порт переходит в состояние *Disconnect*, а если отключение обнаружено «спящим» хабом, контроллеру будет послан сигнал *Resume*.

4. *Enabled* – порт передает сигналы в обоих направлениях. По команде контроллера или при обнаружении ошибки кадра порт переходит в состояние *Disabled*, а при обнаружении отключения – в состояние *Disconnect*.

5. *Suspended* – порт передает сигнал перевода в состояние приостановки. Если хаб находится в активном состоянии, то сигналы через этот порт не пропускаются. Однако «спящий» хаб воспринимает сигналы смены состояния незапрещенных портов, обеспечивая возможность подачи «пробуждающих» сигналов от активизировавшегося устройства даже через цепочку «спящих» хабов.

Состояние каждого порта идентифицируется контроллером хаба с помощью отдельных регистров. Кроме того, имеется общий регистр, биты которого отражают факт изменения состояния каждого порта. Это позволяет хост-контроллеру быстро опрашивать состояние хаба, а в случае обнаружения изменений специальными транзакциями уточнять состояние.

Порт USB 3.0 (его также называют SuperSpeed USB). Для обеспечения полноценной обратной совместимости в концентратор USB 3.0 также интегрирован контролер USB 2.0. При этом сохраняется обратная совместимость с прежними устройствами USB 1.1. Кроме того, USB 3.0 при работе обеспечивает гораздо меньшее энергопотребление, а также большую эффективность протокола (уменьшается нагрузка на шину). Кабели и порты USB 3.0 обеспечивают обратную совместимость, а также имеют поддержку новых разработок, использующих оптические интерфейсы.

Следует отметить и другие важные улучшения порта USB:

– выделенные линии OUT и IN с четырьмя дополнительными проводниками (плюс земля), что предоставляет возможность работать в полнодуплексном режиме (запись и чтение данных осуществляется одновременно);

– более высокий предел максимальной мощности для устройств, использующих питание непосредственно от шины (4,5 Вт против 2,5 Вт, которые могла предоставить шина USB 1.1/2.0);

– улучшенное управление питанием – при простое шина переключается в режим низкого потребления энергии;

– устройства и контролеры осуществляют транзакции только при условии наличия данных.

Коннекторы и кабели USB 3.0 имеют пять дополнительных контактов для корректной работы функций SuperSpeed (зем-

ля и две сигнальные пары SuperSpeed), а контакты USB 1.1/2.0 (их четыре) оставлены для полной обратной совместимости. В разъеме новые контакты находятся под и над существующими. В гнездо USB 3.0 можно вставлять кабели и устройства USB 3.0, а также кабели и устройства USB 2.0/1.1.

Приведем сравнения интерфейсов (см. таблицы 8 и 9).

Таблица 8

Характеристики интерфейсов устройств хранения

Интерфейс	**FDD**	**ATA**	**SATA**	**SCSI**	**SAS**	**FC-SW**	**FC-AL**	**1394**	**USB 1.X/2.0**	**LPT**
Скорость, Мбайт/с	0,06	133	150	160 320 (640)	150 300	100 200	100 200	10-160	1,2/24	2
Число устройств	2	2	4 и выше	16	16 384	224	126	63	127	1
Длина кабеля, м	0,5	0,5	1	25	3	60 500 10 км	60 500 10 км	4,5 (100)	5	5
Максимальное удаление, м	0,5	0,5	1	25	10	10 км	10 км	72	25	5
Одновременный обмен	–	–	–	–	+	+	–	–	–	–
Работа с несколькими инициаторами				+	+	+	+	+		

Таблица 9

Сравнение интерфейсов устройств хранения

Интерфейс	Достоинства	Недостатки
ATA	Массовость. Низкая цена устройств и кабелей. Простота конфигурирования. Эффективность при малом числе устройств	Малое число устройств: до двух на шине (обычно есть две шины). Только для внутренних устройств. Низкая эффективность при работе с двумя устройствами на шине. Высокая загрузка ЦП и большое число прерываний при отработке запросов
SATA	То же. Независимость устройств. Высокая эффективность при поддержке NCQ и устройствами, и контроллером	Малая распространенность устройств и контроллеров с поддержкой NCQ
SCSI	Большое число подключаемых устройств, внутренних и внешних. Высокая эффективность в многозадачных системах при большом числе устройств	Высокая цена контроллера, устройств и аксессуаров (кабели, терминаторы). Сложность установки и конфигурирования
SAS	То же. Возможность физически одновременного обмена с несколькими устройствами. Простота подключения и конфигурирования	Высокая цена оборудования
USB	Удобство подключения. Распространенность. Средняя скорость (USB 2.0)	Загрузка ЦП, возрастающая с увеличением числа устройств. Низкая скорость (USB 1.0)
FireWire	Удобство подключения. Высокая скорость. Эффективная работа с множеством устройств	Малая распространенность в «мире PC»

ГЛАВА 7
ОРГАНИЗАЦИЯ ХРАНЕНИЯ ДАННЫХ НА ЭЛЕКТРОННЫХ (КОМПЬЮТЕРНЫХ) НОСИТЕЛЯХ ИНФОРМАЦИИ

Физическая организация данных на магнитных носителях. В большинстве персональных компьютеров для хранения информации используются дисковые магнитные носители, а именно жесткий магнитный диск[1] и магнитные дискеты. Организация хранения данных на жестком диске и на дискетах имеет много общего. Можно выделить следующие характерные особенности:

1. Поверхность диска разбивается на концентрические дорожки, каждая дорожка в свою очередь разбивается на некоторое количество секторов фиксированного размера. Размер одного сектора, как правило, равен 512 байт.

2. Объем устройства определяется такими характеристиками, как количество поверхностей, количество дорожек и количество секторов на дорожке.

---

[1] Накопитель жесткий магнитный диск (НЖМД) получил название «винчестер» (англ. Winchester) благодаря работавшему на фирме IBM Кеннету Хотону (Kenneth E. Haughton), руководителю проекта, результатом которого стал в 1973 г. выпущенный жесткий диск, впервые объединивший в одном неразъемном корпусе пластины диска и считывающие головки. Они не касались друг друга благодаря прослойке из набегающего потока воздуха, образуемого при быстром вращении.

При разработке новой модели жесткого магнитного диска инженеры использовали краткое внутреннее название модели диска «30-30», что означало два модуля (в максимальной компоновке) по 30 мегабайт каждый. По созвучию название модели совпало с обозначением популярного охотничьего оружия – винтовки Winchester Model 1894, использующего винтовочный патрон марки 30-30 Winchester. Продолжив аналогию, свою разработку инженеры тоже стали называть винчестером. В Европе и США название «винчестер» вышло из употребления в 1990-х гг., а в русском же языке сохранилось и получило полуофициальный статус (на компьютерном сленге сократилось до слова «винт», иногда «винч»).

3. Минимальный объем информации, который можно считать или записать за одну операцию чтения-записи – один сектор.

4. Для адресации сектора на диске требуется указать три значения: номер поверхности, номер дорожки и номер сектора на дорожке. Поверхности и дорожки нумеруются, начиная с 0, секторы – начиная с 1.

Первый сектор диска (сектор 1 на дорожке 0 поверхности 0) называется корневым сектором диска.

Все информационное пространство жесткого диска состоит из одного или нескольких разделов. Количество разделов, их тип и размещение на диске описывается в таблице разделов жесткого диска (*partition table*), которая расположена в корневом секторе жесткого диска.

Первый сектор раздела называется корневым сектором раздела, его содержимое аналогично корневому сектору дискеты. Корневой сектор дискеты или раздела содержит служебную информацию о носителе и часть операционной системы – программу, которая проверяет наличие файлов операционной системы на носителе и, если они обнаружены, загружает операционную систему.

Логическая организация данных на носителях. Прежде чем излагать логическую организацию данных на носителях, необходимо ввести термины «*файл*», «*форматы данных*», «*тип файла*».

Информация на носителях хранится в файлах. Итак, *файл – это поименованная целостная совокупность данных на внешнем носителе*.

В компьютерных системах хранится и обрабатывается информация самого разного характера: тексты, графические изображения, видео- и аудиозаписи, числовые данные, программы и т. д. Для представления каждого рода данных используются различные форматы.

Так как информация хранится на носителях в виде файлов, то можно говорить о типе файлов. В операционных системах (например, семейств MS DOS, Windows) принято указывать тип файла при помощи так называемого расширения имени файла. Таким образом, по расширению можно судить о характере информации, содержащейся в данном файле.

Файловая система. Размещением файлов на носителе управляет операционная система.

Операционная система – это совокупность программных средств, обеспечивающих управление ресурсами вычислительной системы и взаимодействие программных процессов с аппаратурой, другими процессами и пользователем.

Операционная система выполняет следующие функции: управление памятью, управление вводом-выводом, управление файловой системой, управление взаимодействием процессов, диспетчеризация процессов, защита, учет использования ресурсов, обработка командного языка.

Понятие файла одинаково для всех операционных систем, однако, различные операционные системы используют различные способы организации и управления доступом к данным на носителе. При этом используются различные управляющие структуры данных и программные методы обращения к данным.

Совокупность логических структур и программных компонентов операционных систем, управляющих доступом к данным на носителе, называется файловой системой.

Файловая система выполняет следующие функции:

– упорядоченное хранение файлов на носителе (каталоги или, как теперь говорят, – папки);

– хранение информации о размещении файлов на диске;

– хранение информации о файлах;

– контроль целостности данных файлов;
– разграничение доступа к файлам.

Минимальный объем дискового пространства, предусмотренный файловой системой для хранения информации, называется *единицей распределения* или *кластером*. Размер кластера составляет некоторое целое количество секторов.

При размещении файлов на диске каждый файл занимает целое количество кластеров. Если кластер занят файлом не целиком, то оставшаяся часть кластера является «потерянной» и не может быть использована для записи другого файла. Чем больше размер кластера, тем больше потери дискового пространства.

При удалении файла кластеры, занимаемые файлом, помечаются как свободные; физического удаления информации с диска не происходит. Это позволяет при помощи специальных программных средств прочитать информацию, содержавшуюся в удаленном файле.

Большинство операционных систем поддерживают несколько файловых систем. Например: операционная система Windows 10 поддерживает файловые системы FAT32, NTFS и ReFS. Операционная система Windows 95 поддерживала файловые системы FAT12, FAT16 и FAT32. Операционная система Windows NT – FAT12, FAT16 и NTFS. Операционная система MSDOS – только FAT12 и FAT16. Причем FAT12 применялась для разделов объемом менее 4 Мб и для гибких дискет. Цифры говорят о количестве битов, отведенных для адресации кластера в таблице размещения файлов (FAT).

Определение типа файловой системы является одним из первых шагов при исследовании носителей данных. Тип файловой системы можно определить по данным таблицы разделов жесткого диска. В таблице разделов есть поле «Тип раздела», в которое заносится код файловой системы.

Логическая структура гибкого диска (дискеты). Следует отметить, что гибкие диски, вставляемые в дисковод, приводились в движение только тогда, когда операционной системе надо было прочитать или записать данные. При обращении к дискете загорался индикатор и включался двигатель привода дискеты, обеспечивающий скорость вращения 300 об/мин (разброс в скорости вращения достигал не более 0,5 %). Таким образом, поддерживалась постоянная скорость вращения. В том случае, когда дискета вращалась неравномерно или с другой скоростью, данные, записанные на гибкий диск, например, на другом дисководе, не могли быть прочитаны.

Так как гибкие диски имели магнитный слой, то для чтения/записи данных в дисководе были установлены две магнитные головки, которые постоянно находились в контакте с поверхностью гибкого диска. Для перемещения головок от одной дорожки к другой использовался шаговый двигатель, который при подаче на него одного импульса напряжения поворачивался на строго определенный угол.

Данные на гибкие диски записывались блоками на концентрические дорожки (*tracks*), как показано на рис. 17. Самая первая, *нулевая дорожка,* записывалась у внешнего края гибкого диска. Количество стандартных дорожек, доступных операционной системе, на гибком диске объемом 1440 Кбайт было равно 80, но следует отметить, что всегда имелись запасные дорожки, расположенные у центра диска, которые иногда использовались для специальных нужд, например, защиты от копирования.

Объем стандартного блока – сектора (*sector*) – 512 байт. Количество секторов было равно 18. Соответственно, если умножить количество секторов на количество дорожек, две стороны и 512 байт, то получалось:

18 × 80 × 2 × 512 = 1 474 560 байт, или 1440 Кбайт, или 1,44 Мбайт.

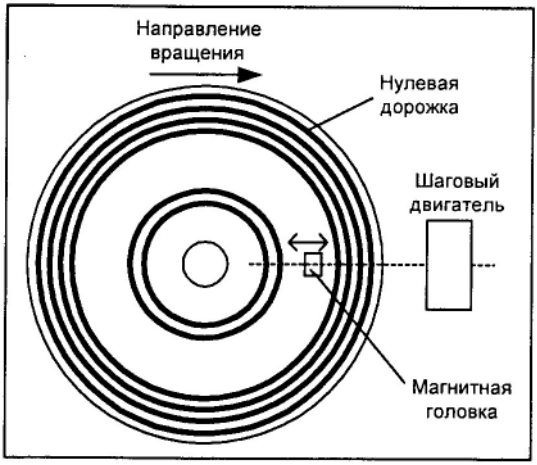


Рис. 17. Логическое разбиение гибкого диска

Секторы на дорожке записывались последовательно (рис. 18). Между каждым сектором оставался промежуток, предназначенный для синхронизации. Данные в секторе предварялись служебной информацией, которая, например, информировала контроллер дисковода о размере сектора.

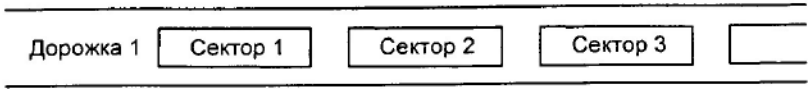


Рис. 18. Порядок секторов на дорожке

Так как гибкий диск представлял собой лавсановую поверхность, покрытую сплошным ферромагнитным слоем, то для создания информационных дорожек производилось его *форматирование*. То есть при первом использовании гибкого диска он должен был быть вставлен в дисковод и с помощью программы FORMAT размечен для работы в конкретной операционной системе.

Пользователь всегда мог отформатировать гибкий диск по своему усмотрению, например, создать меньше дорожек или изменить количество секторов на дорожке. Но пользоваться такой возможностью не рекомендовалось без веских на то оснований, так как такую дискету невозможно было прочитать на другом компьютере без дополнительных усилий, скажем, запуска специальной программы.

Например, программа FORMAT могла запускаться с ключами ***T*** (количество дорожек) и ***N*** (число секторов). Так, для стандартного формата дискеты 1,44 МБайт действительны следующие команды:

– стандартная команда MS-DOS – ***FORMAT A***;

– вариант с дополнительными ключами – ***FORMAT A: /T:80 /N:18***.

Чаще всего, не было необходимости форматировать гибкий диск сразу после покупки, потому что 3,5” гибкие диски форматировались при изготовлении на заводе. К их форматированию прибегали тогда, когда надо было удалить без следа информацию на дискете или восстановить структуру старой дискеты, когда на ней возникали испорченные блоки (bad-blocks).

В обозначении гибких дисков имелась информация о плотности записи: SD (*single density*) – одинарная плотность; DD (*double density*) – двойная плотность; QD (*quadro density*) – двойная плотность с удвоенным количеством дорожек; HD (*high density*) – высокая плотность.

Хранение данных на жестких магнитных дисках. Взаимосвязь физической и логической структур хранения данных. Наиболее важные для пользователей параметры жесткого магнитного диска – это объем дискового пространства и быстродействие. На оба параметра влияют ряд специфичных характеристик конструкции конкретного типа НЖМД, т. е., например, НЖМД

с большим числом дисков не всегда имеет объем больший, чем НЖМД с одним диском.

В общем виде объем жесткого магнитного диска рассчитывается по формуле:

$$\textit{Объем НЖМД} = C \cdot H \cdot S \cdot 512 \text{ байт},$$

где: C – количество цилиндров (дорожек) на одной поверхности ферромагнитного диска; H – количество головок, которое равно количеству рабочих плоскостей (одна поверхность диска может использоваться для служебных нужд – синхронизации); S – количество секторов на дорожке. Так как стандартный размер сектора равен 512 байт, то в формуле присутствует это число.

Если посчитать объем установленного в компьютере жесткого магнитного диска и сравнить его с паспортными данными, то можно заметить, что куда-то пропадает довольно значительное дисковое пространство. Дело в том, что в паспортах на НЖМД указываются два варианта объема – первый относится к неформатированному дисковому пространству (т. е. на диске не создана определенная структура для хранения данных), а второй – к форматированному.

Быстродействие жесткого магнитного диска – запись и чтение информации – зависит от множества факторов, определяемых как конструкцией диска и схемотехникой его контроллера, так и работой IDE-интерфейса. Например, скорость доступа к информации зависит от геометрии дискового пространства – распределения секторов по дорожкам и сторонам дисков, а так как НЖМД – это механическое устройство, движущиеся части его обладают значительной инерцией.

Поскольку пользовательские данные разбиты на маленькие секторы, которые могут размещаться произвольно, то чтение файла, части которого расположены на разных дорожках, занимает больше времени, чем когда все части файла находятся на

одной дорожке или на одном и том же номере дорожки, но на разных сторонах диска. Сказывается тот факт, что для перемещения головки с одной дорожки на другую требуется значительное время, порядка единиц и десятков миллисекунд, а это весьма большое время для современного компьютера.

Например, у НЖМД время перехода на соседнюю дорожку составляет около 1 мс, а в среднем (для случайного перехода на другую дорожку) – 8,5 мс.

При знакомстве с НЖМД полезно знать и понимать следующие термины.

Время доступа (access time) – это время от начала операции чтения до момента, когда начинается чтение данных.

Время поиска (seek time) – это время, которое необходимо для установки головок в нужную позицию (на дорожку, где будут производиться операции чтения/записи данных).

Среднее время поиска (average seek time) – усредненное время, требуемое для установки головок на случайно заданную дорожку.

Время поиска при переходе на соседний трек (track-to-track seek time) – время перехода головок с 1-й дорожки на 2-ю и т. д.

На быстродействие НЖМД оказывает сильное влияние и то, как размещены секторы на дорожках и соседних сторонах дисков. Если все секторы будут идти друг за другом и параллельно на каждой стороне диска, то скорость доступа к информации будет не слишком велика, так как электроника, которая считывает данные с диска, имеет ограниченное быстродействие. Тем более что в отличие от обычного магнитофона, данные на ферромагнитный диск записываются в закодированном виде, что позволяет повысить надежность хранения и уменьшить объем дискового пространства для хранения единиц информации. Соответственно, прочитав первый сектор, контроллер должен проверить

достоверность считанной информации и только потом начать читать следующий сектор, но за это время под головкой будет находиться не второй сектор, а какой-либо следующий. В этом случае приходится ждать, пока диск не сделает целый оборот, чтобы прочитать второй сектор. То же самое относится и к секторам на соседних плоскостях.

Длина дорожки, которая находится ближе к краю диска, значительно больше, чем длина дорожки, расположенной у центра диска. Таким образом, плотность информации на наружных дорожках ниже, чем на внутренних. С этим фактом вынуждены считаться производители НЖМД, которые делят все дисковое пространство на зоны (*intelligent zoned recording*), различающиеся различным числом секторов на дорожках.

Вообще, внутреннее распределение секторов по поверхности диска может быть самым разнообразным, о чем часто знает только производитель НЖМД, но для пользователей контроллер НЖМД преобразует внутреннюю геометрию к общепринятой для компьютеров.

Заметим, что к современным НЖМД неприменима операция низкоуровневого форматирования, как для гибких магнитных дисков и старых «классических винчестеров». Первоначальное форматирование производится на заводе-изготовителе, а попытка пользователя провести низкоуровневое форматирование чаще всего пресекается, хотя иногда такое сопротивление удается преодолеть, однако, это приводит к ухудшению временных параметров НЖМД.

Логическая структура диска. *Жесткие магнитные диски* относятся к магнитным носителям информации. В качестве запоминающей среды у них используются магнитные материалы со специальными свойствами (с прямоугольной петлей гистерезиса), позволяющими фиксировать два магнитных состояния –

два направления намагниченности. Каждому из этих состояний ставятся в соответствие двоичные цифры: 0 и 1. Эти накопители являются наиболее распространенными внешними запоминающими устройствами в компьютере.

Все диски: и магнитные, и оптические характеризуются своим диаметром или, иначе, *форм-фактором.* Наибольшее распространение получили диски с форм-факторами 3,5” (89 мм) и 5,25” (133 мм). Диски с форм-фактором 3,5” при меньших габаритах имеют большую емкость, меньшее время доступа и более высокую скорость чтения данных подряд (*трансфер*), более высокие надежность и долговечность.

Информация на носителе записывается и считывается *магнитными головками* вдоль концентрических окружностей – *дорожек (треков)* (рис. 19). Количество дорожек на носителе и их информационная емкость зависят от типа носителя, конструкции накопителя, качества магнитных головок и магнитного покрытия.

Каждая дорожка носителя разбита на *сектора.* В одном секторе дорожки может быть помещено 128, 256, 512 или 1024 байт, но обычно 512 байт данных. Обмен данными между носителем

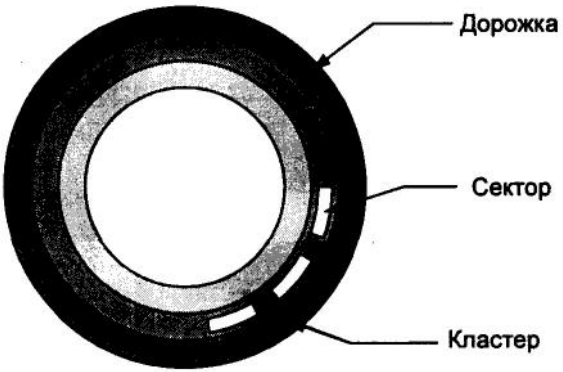


Рис. 19. Логическая структура поверхности магнитного диска

и ОП осуществляется последовательно целым числом секторов. *Кластер* – это минимальная единица размещения информации на диске, состоящая из одного или нескольких смежных секторов дорожки.

При записи и чтении информации носитель вращается вокруг своей оси, а механизм управления магнитной головкой подводит ее к дорожке, выбранной для записи или чтения информации.

Данные на дисках хранятся в *файлах,* которые обычно отождествляют с участком (областью, полем) памяти на этих носителях информации.

Поле памяти создаваемому файлу выделяется кратным определенному количеству кластеров. Кластеры, выделяемые одному файлу, могут находиться в любом свободном месте дисковой памяти и не обязательно являются смежными. Файлы, хранящиеся в разбросанных по диску кластерах, называются *фрагментированными.*

Для пакетов магнитных дисков (диски установлены на одной оси) и для двухсторонних дисков вводится понятие «цилиндр». *Цилиндром* называется совокупность дорожек МД, находящихся на одинаковом расстоянии от его центра.

Оптические компакт-диски. Стандарт компьютерных компакт-дисков CD-ROM (*compact disk read-only memory*) появился в 1985 г. Очень быстро он был дополнен стандартом для записываемых дисков CD-R (*compact disk recordable*), а в дальнейшем добавился режим CD-RW – *compact disk read/write*.

Предшественником стандарта CD-ROM является стандарт цифровой аудиозаписи на компакт-диски CD-DA (*compact disk digital audio*). Компакт-диски с аудиозаписями до сих пор выпускаются наибольшими тиражами. Соответственно, любой компьютерный CD-ROM может без использования центрального

процессора проигрывать музыкальные диски, а его отдельный звуковой выход (почти всегда снабженный регулятором громкости) не имеет никакого отношения к персональному компьютеру, так как аналоговое напряжение выходного сигнала формируется с помощью встроенного цифро-аналогового преобразователя.

По мере развития компьютерных технологий в 1995 г. появился стандарт DVD, предназначенный для записи на компакт-диски видеофильмов.

Кроме перечисленных стандартов, существует ряд других, которые разработали отдельные фирмы и группы фирм. Для воспроизведения записанных в соответствии с ними компакт-дисков требуется, чтобы CD-ROM поддерживал соответствующий стандарт, а также была установлена специальная программа для обработки информации, например, просмотра фотографий в формате Photo-CD или использования режима караоке при проигрывании музыки.

Наиболее популярный стандарт для CD-ROM – это ISO 9660 (ISO – *International Organization for Standardization*, Международная организация по стандартизации), который поддерживают все приводы, работающие с компьютерными компакт-дисками, в том числе и приводы DVD. Стандарт говорит о том, как размещаются данные (в том числе, файлы и каталоги) на компакт-диске.

Большинство стандартов записи данных на компакт-диски описаны в «радужной» серии книг, получивших свое название от цвета обложек:

1. Red Book (красная книга) – описание формата для CD-DA.
2. Yellow Book (желтая книга) – описание формата для CD-ROM.
3. Green Book (зеленая книга) – описание формата для CD-I.
4. Orange Book (оранжевая книга) – описание формата для CD-R и CD-RW.

5. White Book (белая книга) – описание формата для Video CD.

Для звуковых компакт-дисков и CD-ROM используется идентичная технология. Основное отличие состоит в том, что проигрыватели CD-ROM более прочные и содержат устройства для исправления ошибок, обеспечивающие корректность передачи данных с диска в ВМ. Диск изготавливается из пластмассы, например, поликарбоната, и покрыт окрашенным слоем с высокой отражающей способностью, обычно, алюминием. Цифровая информация заносится в виде микроскопических углублений в отражающей поверхности. Запись информации производится с помощью сильно сфокусированного луча лазера высокой интенсивности. Так, создается мастер-диск, с которого затем печатаются копии. Углубления на копии защищаются от пыли и повреждений путем покрытия поверхности диска прозрачным лаком.

Информация с диска считывается маломощным лазером, расположенным в проигрывателе. Лазер освещает поверхность вращающегося диска сквозь прозрачное покрытие. Интенсивность отраженного луча лазера меняется, когда он попадает в углубление на диске. Эти изменения фиксируются фотодетектором и преобразуются в цифровой сигнал.

Углубления, расположенные ближе к центру диска, перемещаются относительно луча лазера медленнее, чем более удаленные. Из-за этого необходимы меры для компенсации различий в скорости так, чтобы лазер мог считывать информацию с постоянной скоростью.

Одно из возможных решений, аналогичное применяемому для магнитных дисков, – увеличение расстояния между битами информации, в зависимости от ее расположения на диске. В этом случае диск может вращаться с неизменной скоростью и, соответственно, такие дисковые ЗУ известны как устройства с постоянной угловой скоростью (CAV, *constant angular velocity*).

Ввиду нерационального использования внешней части диска метод постоянной угловой скорости в CD-ROM не поддерживается. Вместо этого информация по диску размещается в секторах одинакового размера, которые сканируются с постоянной скоростью за счет того, что диск вращается с переменной скоростью. В результате, углубления считываются лазером с постоянной линейной скоростью (CLV, *constant linear velocity*). При доступе к информации у внешнего края диска скорость вращения меньше и возрастает при приближении к оси. Емкость дорожки и задержки вращения возрастают по мере смещения от центра к внешнему краю диска.

Выпускаются компакт-диски различной емкости. В типовом варианте расстояние между дорожками составляет 1,6 мкм, что, с учетом промежутков между дорожками, позволяет обеспечить 20344 дорожки. Фактически же, вместо множества концентрических дорожек, имеется одна дорожка в виде спирали, длина которой равна 5,27 км. Постоянная линейная скорость CD-ROM – 1,2 м/с, т. е. для «прохождения» спирали требуется 4391 с, или 73,2 мин. Именно эта величина составляет стандартное максимальное время проигрывания аудиодиска. Так как данные считываются с диска, со скоростью 176,4 Кбайт/с, емкость CD равна 774,57 Мбайт. Данные на CD-ROM организованы как последовательность блоков. Типичный формат блока показан на рис. 20. Блок включает в себя следующие поля:

Рис. 20. Формат блока CD-ROM

1. *Синхронизация.* Это поле идентифицирует начало блока и состоит из нулевого байта, десяти байт, содержащих только единичные разряды, и вновь байта из всех нулей.

2. *Идентификатор.* Заголовок, содержащий адрес блока и байт режима. Режим определяет пустое поле данных. Режим 1 отвечает за использование кода, корректирующего ошибки, и наличие 2048 байт данных. Режим 2 определяет наличие 2336 байт данных и отсутствие корректирующего кода.

3. *Данные.* Данные пользователя.

4. *Корректирующий код (КК).* Поле предназначено для хранения дополнительных данных пользователя в режиме 2, а в режиме 1 содержит 288 байт – код с исправлением ошибок.

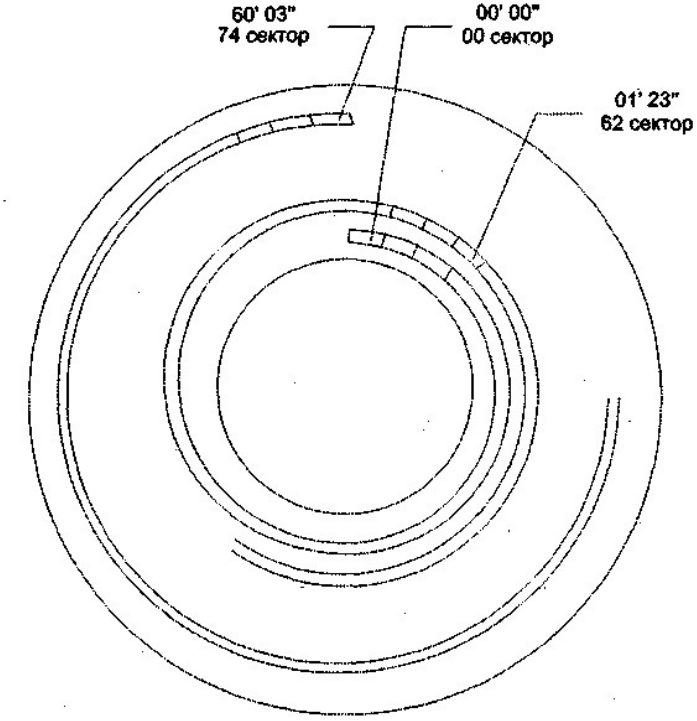


Рис. 21. Организация диска с постоянной линейной скоростью

Рисунок 21 иллюстрирует организацию информации на CD-ROM. Как уже отмечалось, данные расположены последовательно по спиралевидной дорожке. Для варианта с постоянной линейной скоростью произвольный доступ к информации становится более сложным.

На рис. 22 приведена геометрия стандартного компакт-диска, который может быть помещен в любой привод CD, устанавливаемый на данный момент в персональные компьютеры. Кроме того, существуют и становятся все более популярными малогабаритные компакт-диски диаметром в 80 мм, а также компакт-диски прямоугольного формата – компьютеризированные визитные карточки. Подобные компакт-диски читаются и записываются на стандартных приводах CD, у которых всегда в лотке

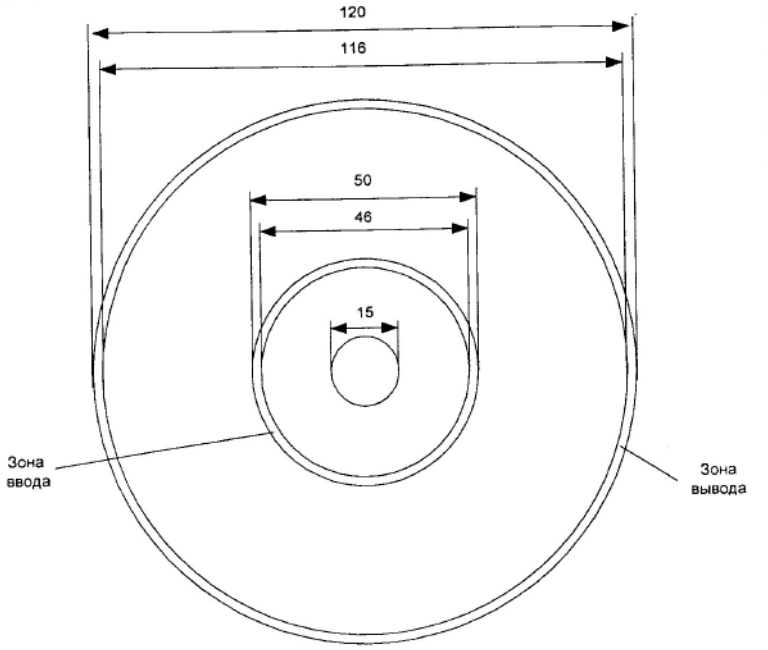


Рис. 22. Геометрические характеристики компакт-диска

для компакт-диска имеется углубление для укладывания 80 мм дисков.

Отметим, что компакт-диски разработаны в метрической системе измерений.

Способ изготовления музыкальных компакт-дисков и CD-ROM почти такой же, как у патефонных пластинок. На прозрачном диске из поликарбоната толщиной 1,2 мм с помощью металлической матрицы методом выдавливания создаются информационные дорожки. Для улучшения отражающих свойств на диск наносится слой алюминия (изредка золота), а для защиты от грязи диск покрывается слоем лака (рис. 23*а*). У компакт-дисков стандарта CD-R между металлическим слоем и диском находится термочувствительный красочный слой. При нагреве записывающим лазером, который имеет повышенную мощность, происходит разогрев металлического и красочного слоев, что приводит к химической реакции, изменяющей отражающие свойства нагретой точки (рис. 23*б*). Обычно в качестве красителя используется цианин или фталоцианин. Для возможности многократной записи для компакт-дисков стандарта CD-RW на поверхности пластмассового диска создается более сложная композиция из пленок различного состава, например, перед ме-

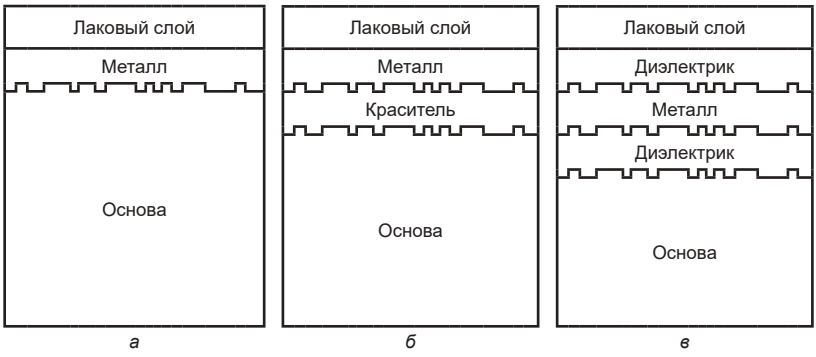


Рис. 23. Строение компакт-дисков: а – CD-ROM; б – CD-R; в – CD-RW

таллическим слоем помещается вещество, которое меняет свою кристаллическую структуру при нагревании, что приводит к изменению оптических свойств (рис. 23*в*).

Логическая структура. Данные на компакт-дисках стандарта CD-ROM записываются на одной спиральной дорожке с интервалом в 1,6 мкм между витками. Длина минимальной информационной точки (*пита*) 0,83 мкм. Для чтения информации используется инфракрасный лазер. Более объемные DVD-диски имеют интервал между витками информационной дорожки в 0,74 мкм, длину пита 0,4 мкм, а для чтения применяют красный лазер, обладающий меньшей длиной световой волны (для новых стандартов используют зеленый лазер).

Для того, чтобы на процесс чтения не влияли дефекты диска и царапины, для записи данных применяются коды с избыточной информацией. Для кодирования 1 байта (8 бит) используются 14 бит плюс 3 бита слияния (*merge bit*).

Минимальная структура данных на компакт-диске – *кадр* (*frame*), который состоит из 24 байт, представленных 588 битами (180 бит используются для коррекции ошибок). Из 3234 кадров формируется сектор, в котором 1352 байта относятся к данным, а 882 байта используются для коррекции ошибок и управления. Теоретически, контроллер привода ком-

Рис. 24. Структура сектора для Mode 00

Рис. 25. Структура сектора для Mode 01

Рис. 26. Структура сектора для Mode 02

пакт-диска может восстановить до 500 байт в секторе, что соответствует дефекту дорожки длиною до 2,5 мм.

На рисунках 24–26 показана структура сектора на компакт-диске в соответствии со стандартом ISO 9660.

Так как стандартный сектор для НЖМД содержит 512 байт, то на компакт-дисках в большинстве случаев в секторе размещается 2048 байт (512×4), а остальное пространство не используется.

В простейшем виде информационная поверхность компакт-диска разделена на три зоны – входной каталог (*lead in*), область данных и выходной каталог (*lead out*), как показано на рис. 27. Входной каталог содержит оглавление (*table of contents*, TOC), адреса записей, число заголовков, суммарное время за-

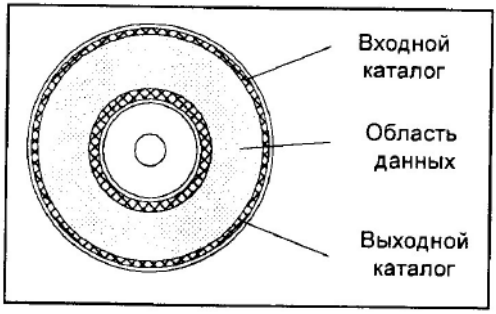


Рис. 27. Организация данных на компакт-диске

писи, название диска. В выходном каталоге присутствует метка конца диска.

Стандарт ISO 9690 регламентирует способы указания имен файлов и структуру каталогов. Используются три вложенных уровня. Первый уровень (*level 1*) – файловая система в стиле MS-DOS, в которой имя файла – 8 символов плюс трехсимвольное расширение, а глубина вложенности каталогов до 8. В качестве символов могут использоваться только заглавные латинские буквы, цифры и символ подчеркивания. Второй уровень (*level 2*) позволяет использовать имена файлов длиной до 31 символа и глубиной каталогов 32. Два первых уровня не допускают фрагментацию файлов, т. е. файл должен записываться на компакт-диск за один раз в виде единого массива. Только для третьего уровня (*level 3*) разрешена фрагментация файлов.

Скорость передачи данных. При проигрывании аудиозаписей используется метод считывания информации с *постоянной линейной скоростью*, вследствие чего скорость вращения диска плавно изменяется при изменении положения читающего лазера. Соответственно, проигрыватели Audio CD и первое поколение CD-ROM имели скорость передачи данных, равную 150 Кбит/с. В следующих поколениях CD-ROM скорость вращения постепенно увеличивали, а для классификации устройств использо-

вали кратность относительно первого поколения. В настоящее время достигнут практический потолок в повышении скорости для традиционных компакт-дисков, которая составляет 56×.

В табл. 10 приведено соотношение между кратностью привода CD-ROM и скоростью передачи данных.

Таблица 10

Скорость передачи данных при различной кратности привода CD-ROM и DVD

Кратность	**CD**	**DVD**
1×	150 Кбайт/с	1,32 Мбайт/с
4×	600 Кбайт/с	5,28 Мбайт/с
6×	900 Кбайт/с	7,93 Мбайт/с
8×	1,2 Мбайт/с	10,57 Мбайт/с
10×	1,5 Мбайт/с	13,21 Мбайт/с
12×	1,8 Мбайт/с	15,84 Мбайт/с
16×	2,4 Мбайт/с	21,13 Мбайт/с
20×	3,0 Мбайт/с	26,4 Мбайт/с
32×	4,8 Мбайт/с	42,26 Мбайт/с

Так как поддержание постоянной линейной скорости при больших оборотах диска – технически сложная задача, то для скоростных приводов используют метод *постоянной угловой скорости* вращения, а также комбинированный метод CLV-CAV. В последнем случае от самой внутренней дорожки до середины диска используется метод CLV, а для внешних дорожек – метод CAV. Заметим, что кратности, которые указываются на скоростных приводах CD-ROM, могут относиться к максимальному (достигается на внешней части диска) или среднему значению, которое опять-таки может рассчитываться по-разному, соответственно, из-за этого бывает сложно сравнивать приводы компакт-дисков различных фирм.

Даже на скорости в 48× существует опасность разрушения некачественного компакт-диска, что приводит к повреждению его осколками внутренней конструкции привода CD-ROM.

Методы записи. Появление приводов CD-R и CD-RW, встреченных с восторгом пользователями, особенно когда цена на них упала ниже 100 долларов, добавило весьма специфических проблем и условий, которыми обставляется запись на компакт-диск. Так, приводы CD-ROM, которые избавились от фирменных интерфейсов и получили поддержку в BIOS системных плат и операционной системе Windows, а также CD-R и CD-RW в режиме чтения необычайно просто устанавливаются, и работа с ними практически не требует каких-либо дополнительных знаний. А вот процесс записи болванок обставлен таким множеством условий, что пользователь, который первый раз пользуется записывающим устройством, обязательно напортит множество компакт-дисков прежде, чем у него будет получаться все так, как надо.

Тут сказалось аудиопрошлое компакт-дисков. Ведь даже используя один и тот же стандарт ISO 9660, компакт-диск можно записывать различными методами. Далее перечислены режимы записи компакт-дисков, которые чаще всего используются в настоящее время:

Disc-At-Once (DAO) – один из самых простых способов записи компакт-дисков, при котором вся информация записываются за один раз. Записывающий лазер работает в непрерывном режиме и выключается только после записи всего диска. Дополнительных данных на компакт-диск после этого записать нельзя. Основное назначение этого метода – создание точных копий компакт-дисков и записи Audio CD.

Track-At-Once (TAO) – в этом режиме запись данных на компакт-диск выполняется отдельными треками. Записывающий

лазер выключается после записи каждого трека. Данный режим позволяет оставить диск «открытым», разрешая в дальнейшем дописывание данных, или «закрыть сессию», защищая компакт-диск от записи новых данных. При записи аудиодиска между композициями образуются двухсекундные паузы.

Session-At-Once – при этом способе за один сеанс записи записывается один или несколько треков.

Компакт-диск, на котором записано более одной сессии, называется *мультисессионным*. Следует помнить, что подавляющее большинство музыкальных центров и CD-плееров не умеют читать мультисессионные компакт-диски (т. е. диски, состоящие из нескольких сессий). Надо также иметь в виду, что при записи нескольких треков теряется некоторое свободное пространство на диске. Так, после первой сессии вы недосчитаетесь около 22 Мбайт, после второй и последующих – около 13 Мбайт. В табл. 11 приведено распределение дискового пространства компакт-диска при различных количествах записанных сессий.

Таблица 11

Рекомендации компании Hewlett-Packard по записи данных на компакт-диск в формате ISO 9660 (CD 74 мин)

Число сеансов	Дисковое пространство для служебных целей	Дисковое пространство, отведенное для данных в расчете на один сеанс
15	Примерно 23 Мбайт Примерно 79 Мбайт (около 23 для первого, по 14 Мбайт для остальных)	Один сеанс 627 Мбайт, 114 Мбайт на каждый сеанс
1030	Примерно 149 Мбайт (около 23 для первого, по 14 Мбайт для остальных) Примерно 430 Мбайт (около 23 для первого, по 14 Мбайт для остальных)	Один сеанс 50 Мбайт, 7,4 Мбайт на каждый сеанс

Проще говоря, трек – это одна музыкальная запись.

Во всех трех приведенных выше методах записи есть серьезный недостаток – данные во время записи должны поступать с НЖМД непрерывно, так как любая приостановка в передаче данных портит записываемую болванку. То есть процессор и НЖМД не должны прерывать передачу данных в привод. Для исключения аварийных ситуаций используется буфер, иногда до 8 Мбайт, из которого данные поступают непрерывно на блок записи, но в ряде случаев Windows может прерваться на такое время, что и буфер также будет опустошен.

Для защиты от опустошения буфера ряд фирм предложили собственные технологии. Наиболее часто используемая технология – BURN-Proof (*Buffer UnderRun Proof*) – предложенная компанией Sanyo Electric, позволяет отключать лазер в тот момент, когда кончаются данные. Как только в буфере появляются новые данные, запись снова возобновляется. Аналогичная разработка от компании Ricoh называется JustLink (она является чуть более совершенной, в ней уменьшается стык между обрывом предыдущей записи и началом новой). Кроме того, существуют технологии: SeamlessLink от компании Acer, SafeBurn от компании Yamaha и др.

При создании полной копии компакт-диска на НЖМД формируется его образ (*image*) в виде файла, например, my_cd.iso или my_cd.c2d. Из файла образа при создании копии берутся данные. Раньше для расширения имен файлов образов применялась комбинация символов ISO, в настоящее время используется термин ISO-образ для файлов, хранящих образ какого-нибудь компакт-диска. Следует обратить внимание, что операционные системы Windows не дают доступ к файлам и каталогам, находящимся внутри ISO-файла, как это возможно, например, в операционной системе Linux.

Когда собираются файлы для сессии, которая будет записана на компакт-диск, программа записи не только записывает в проект имена файлов, но и собирает их в файл-образ, занимая место на НЖМД. Если быстродействие НЖМД выше скорости записи данных на компакт-диск, то можно использовать режим *on-the-fly* (на лету), при котором не создается предварительный образ записываемого компакт-диска. Отметим, когда на НЖМД недостаточно места, это единственный способ записать компакт-диск.

Стандарт на компакт-диски CD-RW появился уже после того, как у пользователей стали популярны приводы CD-ROM. Соответственно, компакт-диски CD-RW могут быть прочитаны только на приводах CD-ROM, совместимых со стандартом MultiRead. Можно считать, что если CD-ROM был выпущен в 1998 г. или позднее, а его скорость составляет 24× или выше, то он должен прочитать данные на компакт-диске CD-RW.

Еще один метод записи, который был назван Packet Writing, позволяет записывать данные небольшими частями – пакетами, которые занимают всего несколько Кбайт. А поскольку размер пакета значительно меньше буфера, то практически невозможно испортить болванку. При отсутствии следующего пакета в буфере лазер выключается до момента поступления нового пакета. Правда, скорость записи в режиме Packet Writing ниже, и до начала записи данных необходимо форматировать компакт-диск, что требует значительного времени (фактически, приходится заполнить метками весь компакт-диск). Для форматирования можно использовать различные программы, например, Adaptec DirectCD и CeQuadrat PacketCD. После форматирования данные на компакт-диск можно записывать обычными файловыми менеджерами, тем же ***Проводником***, правда, должен быть установлен специальный драйвер для записи. Чтение также произво-

дится обычными средствами операционной системы, если установлен драйвер UDF. Несмотря на удобство и эффективность записи данных пакетами, у этого метода есть ряд недостатков, которые ограничивают его применение. Во-первых, диски, записанные этим методом, могут читать только современные приводы CD-ROM, которые понимают режим MultiRead. Во-вторых, отформатированный для использования пакетного режима диск имеет всего 510 Мбайт для CD-RW и 618 Мбайт для CD-R, оставшихся от первоначальных 640 Мбайт. Ну, и самое существенное, для большинства версий операционной системы Windows требуется загрузка драйвера UDF (*universal disk format*). Правда, драйвер этот обычно записывается на тот же компакт-диск таким образом, что он может быть прочитан и установлен в стандартном режиме CD-ROM. Кроме того, как показывает практика, пользователь, не сталкивавшийся до этого с форматом UDF, скорее всего, посчитает диск испорченным или сочтет испорченным свой привод CD-ROM.

Твердотельные накопители. На замену магнитным жестким дискам приходят твердотельные накопители, сокращенно – SSD (*solid-state drive*). И хоть в сокращении упоминается слово drive – диск, новые устройства хранения информации трудно назвать дисками, так как в них нет ничего, напоминающего диск.

Преимущества SSD перед классическим жестким диском нельзя не оценить:

1. Самым главным преимуществом SSD перед HDD является то, что их быстродействие куда выше, чем «классических винчестеров». Дело в том, что SSD используют совсем иную технологию записи, хранения и считывания информации. Технология позаимствована у флеш-памяти, поэтому SSD можно назвать специализированной флешкой большой емкости.

2. Второе преимущество SSD – это отсутствие движущихся частей и деталей. Ни для кого не секрет, что магнитные жесткие диски очень чувствительны к вибрационным нагрузкам, особенно в рабочем состоянии. Случайное падение и с HDD можно распрощаться навсегда. Также нередок выход из стоя привода, который приводит в движение магнитные пластины. Так как в SSD попросту нет движущихся частей и деталей, то устойчивость их к вибрации и ударам значительно выше, чем обычных HDD.

3. Третьим и немаловажным для портативной техники качеством SSD является их малый вес. Если на одну ладонь положить 2,5” SSD, емкостью, например, 128 Гбайт, а на другую ладонь 2,5” HDD на 180 Гбайт, то твердотельный накопитель «*покажется вам просто пушинкой*». Они невероятно легкие.

4. Четвертым преимуществом SSD перед HDD является то, что они расходуют меньше энергии, а рабочая температура их намного ниже.

Технология твердотельных накопителей не идеальна, поэтому стоит сразу упомянуть о некоторых минусах. Все SSD имеют ограниченный ресурс по количеству записываемой информации. Из-за особенностей архитектуры флеш-памяти и методов записи происходит деградация ячеек памяти. Со временем это приводит к уменьшению доступного объема и отказу накопителя. Но не все так плохо, как кажется на первый взгляд. Даже относительно бюджетные модели могут обладать ресурсом перезаписи в районе 200 циклов, не говоря уже о более дорогих моделях.

Устройство SSD-диска. На рисунке 28 показан внешний вид среднестатистического SSD-диска. Существуют модели и в бескорпусном исполнении. Наиболее распространены SSD-накопители форм-фактора 2,5”.

Рядовой твердотельный накопитель представляет собой печатную плату с установленным на ней набором микросхем. Этот

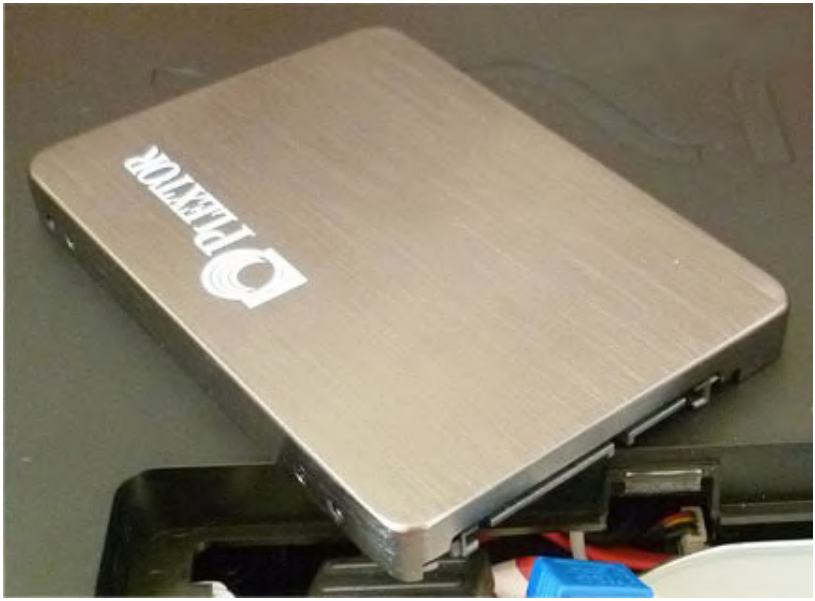


Рис. 28. Внешний вид SSD-диска

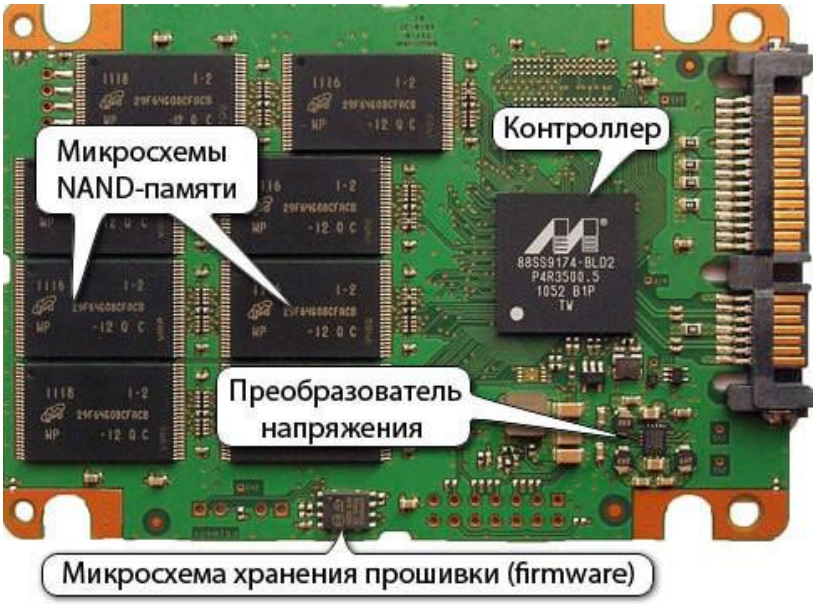
набор состоит из микросхемы *NAND-контроллера* и, собственно, микросхем *NAND-памяти*.

Площадь печатной платы твердотельного накопителя используется полностью. Большую ее часть занимают микросхемы NAND-памяти.

Как видим, в SSD-накопителе нет никаких механических частей и дисков – только микросхемы. Не зря в последнее время SSD все чаще называют «электронными» дисками или твердотельными.

Как уже говорилось, рядовой SSD состоит из двух взаимосвязанных частей: памяти и контроллера.

Типы памяти в SSD. Для хранения информации в SSD используется NAND-память, которая состоят из огромного количества MOSFET-транзисторов с плавающим затвором. Их еще называют ячейками (памяти). Ячейки объединяются в стра-


Микросхемы
NAND-памяти
Контроллер
Преобразователь
напряжения
Микросхема хранения прошивки (firmware)


Микросхемы
NAND-памяти
Преобразователь
напряжения
Буферная
память

ницы по 4 Кбайт (4096 байт), затем – в блоки по 128 страниц, а далее – в массив по 1024 блока. Один массив имеет объем 512 Мбайт и управляется отдельным контроллером. Такая многоуровневая модель устройства накопителя наносит определенные ограничения на его работу. Так, например, стирать информацию можно только блоками по 512 Кбайт, а запись возможна только по 4 Кбайт. Все это приводит к тому, что записью и чтением информации с микросхем памяти руководит специальный контроллер.

В SSD применяются три основных типа NAND-памяти: SLC, MLC и TLC. В памяти типа *SLC (single-level cell)* используются одноуровневые транзисторы. Это значит, что один транзистор может хранить 0 или 1. Одним словом, такой транзистор может запомнить только 1 бит информации.

В памяти типа *MLC (multi-level cell)* используются четырехуровневые транзисторы.

При этом каждый уровень представляет 2 бита информации. То есть на одном транзисторе можно записать одну из четырех комбинаций 0 и 1, а именно: 00, 01, 10, 11. 4 комбинации против 2 у SLC. В два раза больше, чем на SLC-ячейках!

Но у MLC-ячеек есть существенные недостатки. Срок жизни таких ячеек меньше, чем у SLC, и составляет в среднем 100000 циклов. У SLC-ячеек этот параметр составляет 1 млн циклов. Также стоит отметить, что время чтения и записи у MLC-ячеек больше, что уменьшает быстродействие твердотельного накопителя.

Сейчас наибольшее распространение получили SSD-накопители с памятью типа *TLC (triple-level cell* – трехуровневых ячеек). Память TLC имеет 8 уровней, а, следовательно, каждая ячейка может хранить уже 3 бита информации (000, 001, 011, 111, 110, 100, 101, 010).

Таблица 12

Типы флеш-памяти

Характеристика NAND	SLC	MLC	TLC
Бит в ячейке	1	2	3
Циклов перезаписи	100 000	3000	1000
Время чтения	25 мкс	50 мкс	~75 мкс
Время программирования	200–300 мкс	600–900 мкс	~900–1350 мкс
Время стирания	1,5–2 мс	3 мс	~4,5 мс

Из таблицы видно, что чем больше уровней используется в ячейке, тем медленнее работает память на ее основе. TLC-память явно проигрывает, как по скорости, так и по «времени жизни» – циклам перезаписи.

Производители NAND-памяти: Intel/Micron; Hynix; Toshiba/SanDisk; Samsung.

Контроллер и его функции. Каждый SSD включает в себя контроллер, соединяющий компоненты памяти NAND с системой. Контроллер представляет собой встроенный процессор, который выполняет код встроенного программного обеспечения и является одним из наиболее важных для производительности элементов твердотельного накопителя.

Контроллеры создаются как фирмами-производителями памяти и твердотельных накопителей (Intel, Samsung, Toshiba и др.), так и сторонними компаниями (Marvell, SandForce, SiliconMotion, Phison).

ГЛАВА 8
ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ПЕРСОНАЛЬНОГО КОМПЬЮТЕРА

8.1. Общие представления о программном обеспечении

Программное обеспечение является очень широким понятием, которое охватывает:

– системное программное обеспечение (СПО, или *system software*) работоспособности компьютеров и управления их работой;

– прикладное программное обеспечение (ППО) для решения задач в различных предметных областях пользователям в виде пакетов прикладных программ (ППП);

– программное обеспечение производства программ (ПОПП), разрабатываемое с помощью инструментария технологии программирования и реализуемое в виде инструментальных систем программирования (ИСП).

В настоящее время многообразие программного обеспечения представлено, как правило, *программными продуктами (иногда говорят – изделиями)*, которые предназначены для удовлетворения потребностей пользователей, широкого распространения и продажи.

Программный продукт должен быть соответствующим образом подготовлен к эксплуатации, иметь необходимую техническую документацию, предоставлять сервис и гарантию надежной работы программы, иметь товарный знак изготовителя, а также желательно наличие кода государственной регистрации. Только при этих условиях созданный программный комплекс может быть назван программным продуктом. Таким образом, *программный продукт* – это комплекс взаимосвязанных программ

для решения определенной проблемы (задачи) массового спроса, подготовленный к реализации как любой вид промышленной продукции.

Программный продукт разрабатывается на основе промышленной технологии выполнения проектных работ с применением современной инструментальной технологии программирования. Специфика заключается в уникальности процесса разработки алгоритмов и программ, зависящего от характера обработки информации и используемых инструментальных средств. На создание программных продуктов затрачиваются значительные ресурсы – трудовые, материальные, финансовые; требуется высокая квалификация разработчиков. Более того, программные продукты требуют сопровождения, которое осуществляется специализированными фирмами – распространителями программ (дистрибьютерами), реже – фирмами-разработчиками. Сопровождение программ массового применения сопряжено с большими трудозатратами – исправление обнаруженных ошибок, создание новых версий программ и т. п.

В силу этого существует также разнообразие программ или программных комплексов, которые условно можно обозначить как утилитарные. *Утилитарные программы («программы для себя»)* предназначены для удовлетворения нужд их разработчиков. Чаще всего утилитарные программы исполняют роль некоторого сервиса в технологии обработки данных либо являются программами решения функциональных задач, не предназначенных для широкого распространения. Но бывают и исключения, когда *«программы для себя»* становятся либо – freeware – бесплатными, свободно распространяемыми, поддерживаемыми в силу возможности самими пользователями, правомочными вносить в них необходимые изменения; либо – shareware – некоммерческими (условно-бесплатными), которые используются

в условиях регулярного взноса определенной суммы на счет разработчика.

Отдельную нишу создает ряд производителей средств вычислительной техники, комплектуя ее OEM-программами (*original equipment manufacturer*), т. е. встроенными программами, устанавливаемыми на компьютеры или поставляемые вместе с вычислительной техникой.

Итак, СПО состоит из двух компонентов: *базового программного обеспечения* (*base software*), обычно поставляемого вместе с компьютером; *сервисного программного обеспечения* (*utility software*), устанавливаемого дополнительно.

Базовое программное обеспечение содержит минимальный набор программ, обеспечивающих работу компьютера и их управление. В базовое ПО обычно входят: BIOS или UEFI, операционная система и операционные оболочки (текстовые и графические), драйверы внутренних и периферийных устройств компьютера.

Сервисное программное обеспечение предназначено для обслуживания и технического контроля компьютеров с внешней периферией; расширения возможности базового ПО и организации более удобной среды работы пользователя; защиты компьютера от вирусов, вредоносных программ и разнообразных сетевых атак.

8.2. Понятие и назначение операционной системы

Чтобы лучше понять *место и роль* операционной системы (ОС) в информационных процессах компьютера, рассмотрим компьютерную систему в целом и обратим внимание на отдельные ее компоненты:

1. *Аппаратура (hardware)* компьютера, основные части которой – *центральный процессор*, выполняющий *команды*

(инструкции) компьютера; *память (memory)*, хранящая данные и программы, *устройства ввода/вывода* или *внешние устройства (input-output devices – I/O devices)*, обеспечивающие ввод информации в компьютер и вывод результатов работы программ в форме, воспринимаемой пользователем-человеком или другими программами.

2. *Операционная система (operating system)* – системное программное обеспечение, управляющее использованием аппаратуры компьютера различными программами и пользователями.

3. *Прикладное программное обеспечение (applications software)* – программы, предназначенные для решения различных классов задач. К ним относятся, в частности, *компиляторы*, обеспечивающие трансляцию программ с языков программирования, например, C++, в *машинный код (команды)*; *системы управления базами данных (СУБД)*; *графические библиотеки, игровые программы, офисные программы*. Прикладное программное обеспечение образует следующий, более высокий уровень, по сравнению с операционной системой, и позволяет решать на компьютере различные прикладные и повседневные задачи.

4. *Пользователи (users)* – люди и другие компьютеры. Отнесение пользователя-человека к компонентам компьютерной системы – вовсе не шутка, а реальность, поскольку любой пользователь фактически становится частью вычислительной системы в процессе своей работы на компьютере, так как должен подчиняться определенным строгим правилам, нарушение которых приведет к ошибкам или невозможности использования компьютера, и выполнять большой объем типовых рутинных действий – почти как сам компьютер. Одна из важных функций ОС как раз и состоит в том, чтобы избавить пользователя от большей

части такой рутинной работы и позволить ему сосредоточиться на работе творческой. Другие компьютеры в сети также могут играть роль *пользователей (клиентов)* по отношению к данному компьютеру, выступающему в роли *сервера*, используемого, например, для хранения файлов или выполнения больших программ.

Заметим, что девизом фирмы *Sun Microsystems* еще в 1982 г., при ее создании, стал афоризм «*The network is the computer*» («*Сеть – это компьютер*»). Эту истину следует помнить всем пользователям компьютеров и шире использовать возможности компьютерных сетей, распределяя среди компьютеров в них различные задачи (функции). Изолированный от сети компьютер сегодня – это «*каменный век*». Отсюда – *неразрывная связь* операционных систем и сетей.

Понятие операционной системы. Определение ОС. *Операционная система (operating system)* – базовое системное *программное обеспечение* (комплекс управляющих и обрабатывающих программ), управляющее работой компьютера и являющееся посредником (*интерфейсом*) между *аппаратурой* (*hardware*), *прикладным программным обеспечением* (*application software*) и *пользователем* компьютера (*user*). Фактически *операционная система* с точки зрения пользователя – это как бы продолжение аппаратуры, надстройка над ней, обеспечивающая более удобное, надежное и безопасное использование компьютеров и компьютерных сетей.

Общее назначение ОС:

1. *Обеспечение удобства, эффективности, надежности, безопасности выполнения пользовательских программ*. Для пользователя самое главное – чтобы его программа работала, вела себя предсказуемо, выдавала необходимые ему правильные результаты, не давала сбоев, не подвергалась внешним атакам.

Вычислительную среду для такого выполнения программ и обеспечивает операционная система.

2. *Обеспечение удобства, эффективности, надежности, безопасности использования компьютера*. Операционная система обеспечивает максимальную полезность и эффективность использования компьютера и его ресурсов, обрабатывает прерывания, защищает компьютер от сбоев, отказов и хакерских атак. Эта деятельность ОС может быть не столь заметной для пользователя, но она осуществляется постоянно.

3. *Обеспечение удобства, эффективности, надежности, безопасности использования сетевых, дисковых и других внешних устройств, подключенных к компьютеру*. Особая функция операционной системы, без которой невозможно использовать компьютер, – это работа с внешними устройствами. Например, ОС обрабатывает любое обращение к жесткому диску, обеспечивая работу соответствующего драйвера (низкоуровневой программы для обмена информацией с диском) и контроллера (специализированного процессора, выполняющего команды ввода/вывода с диском). Любая флеш-карта, вставленная в USB-порт компьютера, распознается операционной системой, получает свое логическое имя (в системе Windows – в виде буквы, например, G:) и становится частью файловой системы компьютера на все время, пока она не будет извлечена (демонтирована).

Подчеркнем особую важность таких функций современных ОС, как обеспечение *безопасности, надежности и защиты данных*. Следует учитывать, что компьютер и операционная система работают чаще всего в сетевом окружении, в котором постоянно возможны и фактически происходят атаки хакеров и их вредоносных программ, ставящие своей целью нарушение работы компьютера, «взлом» конфиденциальных данных

пользователя, хранящихся на нем, похищение логинов, паролей, использование компьютера как «робота» для рассылки реклам или вирусов и др.

В связи с этим еще в 2002 г. фирма Microsoft объявила *инициативу по надежным и безопасным вычислениям (trustworthy computing initiative)*, целью которой является повышение надежности и безопасности всего программного обеспечения, прежде всего, операционных систем.

Основные компоненты операционной системы:

Ядро (kernel) – центральная часть операционной системы, низкоуровневая основа любой операционной системы, выполняемая аппаратурой в особом привилегированном режиме. Ядро загружается в память один раз и находится в памяти резидентно – постоянно, по одним и тем же адресам, управляет процессами.

Подсистема управления ресурсами (resource allocator) – часть операционной системы, управляющая вычислительными ресурсами компьютера – процессором и его временем, оперативной и внешней памятью, доступом к файловой системе, сетевым взаимодействием и др.

Управляющая программа (control program, supervisor) – подсистема ОС, управляющая исполнением других программ и функционированием устройств ввода/вывода.

Основные функции ОС:

– выполнение по запросу программ элементарных (низкоуровневых) действий, которые являются общими для большинства программ и часто встречаются почти во всех программах (ввод и вывод данных, запуск и остановка других программ, выделение и освобождение дополнительной памяти и др.);

– загрузка программ в оперативную память и данных, которые подлежат обработке;

– выполнение программ;

– стандартизованный доступ к периферийным устройствам (устройства ввода/вывода);

– управление оперативной памятью (распределение между процессами, организация виртуальной памяти);

– управление доступом к данным на энергонезависимых носителях (таких как жесткий диск, оптические диски и др.), организованным в той или иной файловой системе;

– обеспечение пользовательского интерфейса;

– сетевые операции, поддержка стека сетевых протоколов.

Дополнительные функции ОС:

– параллельное или псевдопараллельное выполнение задач (многозадачность);

– эффективное распределение ресурсов вычислительной системы между процессами;

– разграничение доступа различных процессов к ресурсам;

– организация надежных вычислений на основе разграничения доступа к ресурсам;

– взаимодействие между процессами: обмен данными, взаимная синхронизация;

– защита самой системы, а также пользовательских данных и программ от действий пользователей (злонамеренных или по незнанию) или приложений;

– многопользовательский режим работы и разграничение прав доступа.

Современные универсальные ОС можно охарактеризовать, прежде всего, как: *использующие файловые системы* (с универсальным механизмом доступа к данным), *многопользовательские* (с разделением полномочий), *многозадачные* (с разделением времени).

8.3. Обзор основных функций операционной системы

Управление процессами предполагает, что ОС отвечает за действия, связанные с управлением процессами.

Процесс (process) – это пользовательская программа при ее исполнении в компьютерной системе. Для выполнения процесса требуется ряд *ресурсов*, включая время процессора, память, файлы, устройства ввода/вывода, сетевые устройства и др.

Итак, ОС отвечает за нижеперечисленные действия, связанные с управлением процессами.

– *Создание и удаление процессов*. При *создании процесса* в памяти создаются соответствующие системные структуры (таблица страниц, стек и др.). При удалении процесса *память*, занимаемая ими, освобождается, а также выполняется закрытие всех файлов и освобождение всех других ресурсов, которые использовал процесс, если последний не сделал этого явно.

– *Приостановка и возобновление процессов*. Выполнение процесса приостанавливается при выполнении *синхронного ввода/вывода*, а также *системного вызова или команды* (типа *suspend*). Сразу отметим, что использовать подобные *операции явной приостановки процессов* следует с осторожностью, так как приостанавливаемый процесс может находиться в своей *критической секции* – выполнять обработку общего ресурса, к которому каждому процессу предоставляется *монопольный доступ*, так что при его приостановке возникает *ситуация тупика* (*deadlock*) – приостановленный процесс не может освободить *ресурс*, а конкурирующий процесс не может его получить. При приостановке процесса ОС сохраняет состояние его выполнения, а при возобновлении – восстанавливает.

– *Синхронизация процессов*. Процессы работают параллельно и при этом конкурируют за общие ресурсы, а также должны

в некоторые моменты вычислений ожидать наступления некоторых событий. Для предотвращения возможных конфликтов и несогласованностей, например, *race condition* – несогласованного доступа к общим данным, при котором один процесс читает старые данные, а другой их в этот же момент обновляет, ОС предоставляет средства *синхронизации* (например, *семафоры* и *мониторы*).

– *Взаимодействие процессов*. При своей параллельной работе процессам необходимо взаимодействие с целью согласованного решения различных частей одной и той же задачи. Процессы могут взаимодействовать с помощью передачи *сообщений* друг другу, а также с помощью так называемых *условных переменных* и *рандеву*. ОС предоставляет все эти средства в виде системных вызовов для организации адекватного и удобного взаимодействия процессов.

Отметим, что существуют средства синхронизации – *семафоры*, предложенные еще в 1966 г. профессором Эдсгером Дейкстра, как новый способ синхронизации процессов и ставший в последствии классическим, и *мониторы*, предложенные в 1974 г. профессором Чарльзом Хоаром, как более надежный способ синхронизации.

Управление основной памятью. Основную (оперативную) *память* компьютерной системы можно рассматривать как большой *массив слов или байт*, каждый из которых имеет свой *адрес*.

Таким образом, *память – это хранилище данных с быстрым доступом, совместно используемое процессором и устройствами ввода/вывода.*

Следует иметь в виду важную особенность основной памяти. В компьютерных архитектурах имеется два различных способа нумерации байт в слове. Традиционно *память* представляют как *линейный массив* расположенных слева направо слов, такой,

что адреса слов, находящихся левее, меньше, чем адреса слов, находящихся правее. Каждое *слово* делится на байты, имеющие в слове свои номера: 0, 1 и т. д. Например, в 64-разрядных системах в слове 8 байт, с номерами от 0 до 7, в более старых 16-разрядных системах (*×86*) – два байта, с номерами 0 и 1.

Если *нумерация байт в слове* начинается слева, т. е. начиная со старших бит, то такую архитектуру принято называть *big endian*, если же справа, т. е. начиная с младших бит, то *little endian*. Например, при *big endian* – архитектуре 32-разрядного процессора байты двух соседних слов памяти нумеруются так: ***0, 1, 2, 3, 0, 1, 2, 3***. При *little endian* архитектуре *нумерация* будет иной*:* ***3, 2, 1, 0, 3, 2, 1, 0***.

Представим теперь, что мы хотим рассматривать эти же два слова как *массив байт длиной 8* и записать туда байт за байтом 8 символов какой-либо строки. Такая операция при обеих архитектурах будет выполнена одинаково, т. е. последовательные байты получат именно эти значения. Затем рассмотрим результат снова, но уже как последовательность из двух слов. При архитектуре *big endian* сюрпризов не будет. Однако при архитектуре *little endian* результат будет совсем иным, поскольку при обработке целого слова в архитектуре *little endian* байты как бы «переставляются» в обратном порядке. Разумеется, это неудобно. Подобная операция типична для *системных программ*, например, *таблица* идентификаторов в компиляторе должна содержать как символы идентификатора (последовательность байт), так и другую информацию о нем (длину, ссылки в различные таблицы и т. д.). Поэтому при архитектуре *little endian* приходится хранить и обрабатывать *байтовые массивы и массивы слов* отдельно и нельзя изменять точку зрения на одну и ту же область памяти и рассматривать ее то как *массив байт*, то как *массив слов*.

При программировании на *языках высокого уровня* разработчику, как правило, не приходится учитывать это различие. Однако, если при реализации *распределения памяти* требуется одну и ту же область памяти рассматривать то как *массив слов*, то как *массив байт*, тогда для архитектур *little endian* могут быть «сюрпризы», связанные с тем, что при записи в *память* как в *массив слов* байты как бы «переставляются».

ОС отвечает за следующие действия, связанные с управлением памятью:

1. *Отслеживание, какие части памяти в данный момент используются и какими процессами*. Как правило, ОС организует для каждого процесса свою *виртуальную память* – расширение основной памяти путем хранения ее образа на диске (дампа) и организации подкачки в *оперативную память* фрагментов (страниц или сегментов) виртуальной памяти процесса и ее откачки по мере необходимости. Поэтому чаще дамп называют файлом подкачки.

2. *Стратегия загрузки процессов в оперативную память по мере ее освобождения*. При активизации процесса и его запуске или продолжении его выполнения процесс должен быть загружен в *оперативную память*, что и осуществляется операционной системой. При этом, возможно, какие-либо неактивные в данный момент процессы приходится откачивать на диск.

3. *Выделение и освобождение памяти по мере необходимости*. ОС обслуживает запросы типа «выделить область *оперативной памяти* длиной *n-байт*» и «освободить область памяти, начинающуюся с заданного адреса, длиной *m-байт*». Длина участков выделяемой и освобождаемой памяти может быть различной. ОС хранит список занятой и свободной памяти. При интенсивном использовании памяти может возникнуть ее *фрагментация* – дробление на мелкие свободные части, вследствие

того, что при запросах на выделение памяти длина найденного сегмента оказывается немного больше, чем требуется, и остаток сохраняется в списке свободной памяти как область небольшого размера (подчас всего 1–2 слова). Тогда приходится бороться с фрагментацией. При исчерпании *оперативной памяти* ОС выполняет «*сборку мусора*» – поиск неиспользуемых фрагментов, на которые потеряны ссылки, и *уплотнение (компактировку)* памяти – сдвиг всех используемых фрагментов по меньшим адресам, с корректировкой всех адресов.

Управление файлами. Файл (*file*) – совокупность логически взаимосвязанной информации, расположенная во внешней памяти.

Файлами могут быть:

– программы – исполнимые файлы, которые хранят машинные команды процессора;

– объектные коды – совокупности подпрограмм, не прошедшие связывание – линкование (link), представленные в двоичном коде машинные команды процессора;

– тексты, которые отражают алгоритмы решения самых разнообразных задач, составленные на каком-либо языке программирования – неисполнимые файлы, иногда говорят: исходный текст;

– тексты, которые хранят команды ОС – так называемые bat-файлы – исполнимые файлы;

– документы в текстовом виде;

– аудио- и видеоданные;

– специально созданные структуры баз данных и т. д.

ОС отвечает за следующие действия, связанные с управлением файлами:

1. *Создание и удаление файлов. Отображение файлов на внешнюю память*. ОС выделяет *внешнюю память* при создании

нового файла. *Файл* в большинстве файловых систем состоит из *заголовка* и *памяти*. В заголовке хранятся *атрибуты* файла, например, его *длина*, тип, *ссылка* на элементы файла во внешней памяти. *Память* файла может быть организована по-разному – как *список* смежных областей (*блоков, записей* или, чаще всего, *секторов***)**, одна смежная область, *список* индексных узлов, ссылающихся на блоки файла и т. д. Кроме создания и удаления файла, основные *операции* над ним – *открытие* и *закрытие*. *Открытие файла* – это считывание в *основную память* его заголовка и, возможно, одного или нескольких соседних блоков. Оно должно быть выполнено перед выполнением операций чтения из файла или записи в *файл*. *Закрытие файла* – это обратная операция: сброс всех копий блоков на *внешнюю память* и освобождение областей оперативной памяти, занятых открытым файлом. ОС закрывает файлы процесса при его завершении, если процесс не сделал этого явно (последнее рекомендуется). При отображении файлов на *внешнюю память* возникают проблемы, аналогичные проблемам распределения основной памяти, – фрагментация, возможность исчерпания внешней памяти или ее *раздела (partition)* – смежной области внешней памяти, имеющей определенное символьное обозначение.

2. *Создание и удаление папок (директорий). Поддержка примитивов (пользовательских команд и библиотечных вызовов) для управления файлами и папками (директориями)*. *Папка (директория – directory)* – это каталог (справочник) ссылок на группу файлов или других папок (директорий), каждый (каждая) из которых имеет в данной папке (директории) свое уникальное символьное *имя*. *Иерархия* папок (директорий) позволяет организовать *поиск* файла по его *символьному пути (path)*, например, в ОС *Windows*: «*c:\doc\plan.txt*» – текстовый документ «*plan.txt*», содержащий план моих текущих действий, *ссылка* на который

находится на диске «*C:*», в папке (директории) «*doc*». ОС управляет созданием и удалением директорий и поиском в них файлов *по* их путям. Следует иметь в виду, что на *файл* возможно несколько ссылок из разных директорий (хотя это и не рекомендуется), поэтому *удаление элемента* директории не означает и удаления файла – сам *файл* сохраняется, пока на него есть хотя бы одна *ссылка*. Более того, в некоторых файловых системах, например, *FAT* ОС *Windows* ошибочно удаленный *файл* можно восстановить, хотя и под другим именем. В *NTFS* ОС *Windows* – можно восстановить под собственным именем. В других же файловых системах, например, в *UNIX*, где используются индексные блоки, хранящие адреса блоков файла, *удаление файла* – фатальная операция, от ошибок в которой может спасти только вовремя сделанная резервная копия файловой системы на диске или флешке.

3. *Сброс или резервное копирование (backup) файлов* на *энергонезависимые* носители информации (*флеш-память*, *компакт-диск*, *USB-диск* и др.) с целью их последующего восстановления при сбое или при ошибке пользователя. *Значение резервного копирования файлов* для пользователей ОС трудно переоценить. Все результаты работы пользователя в виде наиболее важных документов, данных в папках (директориях) должны регулярно копироваться на *внешнюю память* (желательно делать не одну, а несколько копий на разные носители). Возможности ОС позволяют выполнять такое *копирование* автоматически, в определенное время.

В некоторых ОС реализованы файловые системы с *криптошифрованием* данных при записи в *файл*. Такой подход позволяет решить проблему *сохранения конфиденциальности информации (privacy)*, но опасен в случае отсутствии копии сертификата на внешнем съемном носителе, поскольку данные при работе с другими ОС не будут доступны.

Управление вторичной памятью. Так как размер оперативной памяти недостаточен для постоянного хранения всех программ и данных, в компьютерной системе должна быть предусмотрена вторичная (внешняя) *память* для откачки (*back up*, *swapping*) части содержимого оперативной памяти.

В большинстве компьютерных систем в качестве главной вторичной памяти для хранения программ и данных используются диски.

ОС отвечает за выполнение следующих действий, связанных с управлением дисками: *управление свободной дисковой памятью; выделение дисковой памяти; диспетчеризация дисков (disk scheduling)*.

При управлении вторичной памятью возникают проблемы, аналогичные проблемам распределения оперативной памяти. Всякая *память*, даже самая большая по объему, рано или поздно может исчерпаться, либо фрагментироваться на множество мелких областей свободной памяти. Реализуются в этих случаях разработанные под ОС *методы управления* оперативной и внешней памятью.

Управление сетевыми (распределенными) системами. Распределенная система – это совокупность компьютеров (рабочих станций), которые не используют *общую память* или частоты (такты процессора). Каждый имеет собственную *локальную память*. Рабочие станции в такой системе соединены в *сеть*. Сетевое взаимодействие выполняется по определенному *протоколу* (интерфейсу, набору операций). Наиболее распространенный *сетевой протокол – TCP/IP*, основанный на *IP-адресах* машин (*hosts*).

В распределенной системе *сетевая ОС* обеспечивает *доступ* пользователей к различным *общим сетевым ресурсам* – например, файловым системам или принтерам. Каждому обще-

му ресурсу сетевой ОС присваивает определенное сетевое имя и управляет возможностью доступа к нему с различных компьютеров сети. Сетевая ОС обеспечивает также *удаленный запуск программ на другом компьютере сети* – возможность входа на другой компьютер и работы на нем с использованием памяти, процессора и диска удаленного, как правило, более мощного компьютера и использованием клиентского компьютера в качестве терминала. В *Windows* такая возможность называется *удаленный рабочий стол (remote desktop connection)*, в операционных системах UNIX, LINUX, Solaris – *rsh (remote shell) и rlogin (remote login)*.

Доступ к общим ресурсам (*shared resource*) в распределенной системе позволяет: *ускорить вычисления; расширить границы доступа к данным; обеспечить более высокую надежность*.

Система защиты (protection). Термин *защита (protection)* используется для обозначения механизма управления доступом программ, процессов и пользователей к *системным* и *пользовательским ресурсам*.

Механизм защиты в ОС должен обеспечивать следующие возможности:

– различать *авторизованный, или санкционированный (authorized)* и *несанкционированный (unauthorized) доступ*. Под *авторизацией* понимается предоставление операционной системой пользователю или программе какого-либо определенного набора *полномочий (permissions)*, например, возможности чтения или изменения файлов в *файловой системе с общим доступом*;

– описывать предназначенные для защиты *элементы управления (конфигурации*). В ОС UNIX используются специальные текстовые конфигурационные файлы для представления информации о файловых системах, к которым возможен *сетевой до-*

ступ, с указанием списка машин (хостов), с которых возможен *доступ*, и набора действий, которые могут быть выполнены;

– обеспечивать средства выполнения необходимых для защиты действий (*сигналы, исключения, блокировка и др.*). Система защиты ОС должна фильтровать *сетевые пакеты*, получаемые извне локальной сети, выбирать и отсеивать «неблагонадежные» (получаемые с подозрительных IP-адресов), сообщать пользователю об обнаруженных и ликвидированных попытках *сетевых атак* с целью «взлома» Вашего компьютера (что и происходит на практике, например, при работе в *Windows*, когда Вы выходите в *Интернет* с Вашего компьютера). Если Вы нарушили условия защиты (например, Ваша *программа* попыталась обратиться к файлу, работать с которым у Вас нет полномочий), ОС должна выдать понятное сообщение и прекратить работу программы. В современных системах это делается с *помощью генерации исключений (exceptions)*, например, *Security Exception*.

Система поддержки командного интерпретатора. Большинство команд для ОС задаются с помощью специальных управляющих операторов, предназначенных для выполнения следующих основных функций:

– *создания процессов и управления процессами* (в UNIX команда «***ps –a***» выводит в стандартный вывод процесса информацию обо всех активных процессах в системе с указанием их номеров (PID);

– *выполнения ввода/вывода* (в системе MS DOS команда «***type file_name***» выполняет вывод на терминал содержимого заданного текстового файла);

– *управления вторичной памятью* (в UNIX команда «***share /mydir***» добавляет папку (директорию) «***/mydir***» к списку совместно используемых в локальной сети файловых систем;

– *управления основной памятью* (команда «***swap***» в ОС Solaris позволяет управлять размером пространства дисковой па-

мяти для реализации виртуальной памяти и выводить информацию о его состоянии;

– *доступа к файловой системе* (в большинстве ОС команда «***cd new_dir***» устанавливает заданную папку (директорию) в качестве текущей (рабочей);

– *защиты* (в системе UNIX команда «***chmod 700 my_home _dir***» защитит Вашу домашнюю папку (директорию) от непрошенных любопытных глаз – «лазутчик» не сможет даже выполнить команду «***cd***» для этой папки (директории) и, тем более, читать в ней какие-либо файлы;

– *управления сетью* (команды «***telnet host_name***» и «***rlogin host_name***» (последняя доступна в системе UNIX) служат для удаленного входа на другой компьютер сети).

Программа, которая читает и интерпретирует *операторы управления*, называется *командным интерпретатором*. В ОС Windows это *интерпретатор command.com*, доступный для выполнения команд в окне *MS DOS prompt*. В ОС UNIX, Linux, Solaris – это всевозможные «шеллы»: ***sh, csh, ksh, bash*** – процессоры для интерпретации мощных *командных языков*. *Функция командного процессора* состоит в том, чтобы прочесть и исполнить очередной управляющий оператор (команду).

Сервисы (службы) ОС. Операционная система предоставляет для пользователей целый ряд сервисных возможностей – сервисов (служб):

1. *Исполнение программ – загрузка* программы в *память* и ее выполнение (так, в Windows при запуске программы ОС находит в файле ее двоичного кода (.exe) так называемую *заглушку для исполнения*, содержащую ссылку на код головного метода main, и запускает его). В среде *.NET* этот же *execution stub* в файле двоичного кода используется системой для вызова не непосредственно исполняемой программы, а общего окружения времени выпол-

нения – *Common Language Runtime* (*CLR*), которое обеспечивает особый режим (managed *execution*) выполнения программы.

2. *Поддержка ввода/вывода* – обеспечение интерфейса для работы программ с *устройствами ввода/вывода*. Например, в ОС UNIX у каждой программы есть свой *стандартный ввод* и *стандартный вывод* (по умолчанию это *терминал*).

3. *Работа с файловой системой* – предоставление программам интерфейса для создания, именования, *удаления файлов*.

4. *Коммуникация – обмен информацией* между процессами, выполняемыми на одном компьютере или на других системах, связанных в *сеть*. В операционных системах реализуется с помощью *общей памяти (shared memory)* или передачи сообщений.

5. *Обнаружение ошибок* в работе *процессора, памяти*, *устройств ввода/вывода* и программах пользователей.

Дополнительные функции ОС. *Системные вызовы (system calls)* являются интерфейсом между выполняемой программой и операционной системой. Обычно системные вызовы доступны как специальные команды, созданные ассемблером – языком программирования низкого уровня. Однако некоторые языки программирования высокого уровня (C, C++, Java и др.) позволяют выполнять системные вызовы непосредственно, не «опускаясь» до ассемблерного уровня, с помощью вызовов специальных библиотечных функций (методов) типа System («***cd my_dir***»).

При системном вызове ОС из программы пользователя возникает проблема передачи параметров. Используются три основных способа передачи параметров исполняемой программой операционной системе:

Различаются следующие основные виды системных вызовов:

– *управление процессами* (в UNIX *системный вызов fork* создает новый *параллельный процесс* с новым пространством *виртуальных адресов)*;

– *управление файлами* (в UNIX *системный вызов open (f, «rw»)* осуществляет открытие заданного файла для чтения и записи);

– *управление устройствами* (*системный вызов rewind* осуществляет перемотку – позиционирование – магнитной ленты на начало);

– *сопровождающая информация* (*системный вызов env* выдает в *стандартный вывод* информацию о значениях *переменных окружения* – переменных с символьными значениями, например, *PATH*, задающими окружение исполняемого процесса;

– *коммуникации* (*системный вызов CreateSocket* создает новый *сокет* – системную структуру для обмена информацией клиента с сервером через TCP/IP – сеть).

Следует отметить, что многие из этих возможностей ОС доступны также в виде выполняемых команд.

8.4. Архитектура операционной системы

Операционная система – весьма сложная по архитектуре программная система, в которой можно выделить следующие основные компоненты (или подсистемы):

1. Подсистема «Управление процессами».
2. Подсистема «Управление основной памятью».
3. Подсистема «Управление файлами».
4. Подсистема «Управление системой ввода/вывода».
5. Подсистема «Управление внешней памятью».
6. Поддержка сетей (*networking*).
7. Система защиты (*protection*).
8. Система поддержки командного интерпретатора.
9. Графическая оболочка.

Рассмотрим эти компоненты архитектуры ОС подробнее.

Подсистема «*Управление процессами*». Процесс – это *программа* пользователя в ходе ее выполнения в компьютерной вычислительной системе. Сетевая ОС управляет работой процессов, их распределением по процессорам и ядрам системы, порядком их выполнения и размещения в памяти, их синхронизацией при параллельном решении частей одной и той же задачи разными процессами.

Подсистема «*Управление основной памятью*». Основная (оперативная) *память* может рассматриваться как большой *массив*. *Операционная система* распределяет ресурсы памяти между процессами, выделяет *память* по запросу, освобождает ее при явном запросе или по окончании процесса, хранит списки занятой и свободной памяти в системе.

Подсистема «*Управление файлами*». *Файл* – это логическая *единица* размещения информации на внешнем устройстве, например, на диске. ОС организует работу пользовательских программ с файлами, создает файлы, выполняет их открытие и закрытие и *операции* над ними (чтение и *запись*), хранит ссылки на файлы в директориях (папках) и обеспечивает их *поиск* по символьным именам.

Подсистема «*Управление системой ввода/вывода*». Как уже отмечалось, в компьютерной системе имеется большое число *внешних устройств* (принтеры, сканеры, устройства управления компакт-дисками и др.), управляемых специальными контроллерами (спецпроцессорами) и драйверами – низкоуровневыми программами управления устройствами, выполняемыми в привилегированном режиме. ОС управляет всеми этими *аппаратными и программными компонентами*, обеспечивая *надежность* работы *внешних устройств*, эффективность их использования, *диагностику* и *реконфигурацию* в случае их сбоев и отказов. Для этого ОС хранит и использует *таблицу состояния* устройств.

Подсистема «*Управление внешней памятью*». Как уже говорилось, внешняя (вторичная) *память* – это расширение оперативной памяти процессора более медленными, но более емкими и постоянно хранящими информацию видами памяти (диски, ленты и др.). При управлении внешней памятью ОС решает задачи, аналогичные задачам управления основной памятью, – выделение памяти по запросу, освобождение памяти, хранение списков свободной и занятой памяти и др. ОС поддерживает также использование ассоциативной памяти (кэш-памяти) для оптимизации обращения к внешней памяти.

Поддержка сетей. Как неоднократно подчеркивалось, любая современная компьютерная система постоянно или временно находится в различных локальных и глобальных сетях. *Операционная система* обеспечивает использование сетевого оборудования (*сетевых карт* или *адаптеров*), вызов соответствующих драйверов, поддержку удаленного взаимодействия с файловыми системами, находящимися на компьютерах сети, удаленный вход на другие компьютеры сети и использование их вычислительных ресурсов, отправку и получение сообщений по сети.

Система защиты. Согласно современным принципам надежных и безопасных вычислений, при работе ОС должны быть обеспечены *надежность* и *безопасность*, т. е. защита от внешних атак, *конфиденциальность* личной и корпоративной информации, *диагностика* и исправление ошибок и неисправностей и др. ОС обеспечивает защиту *компонентов* компьютерной системы, данных и программ, поддерживает фильтрацию *сетевых пакетов*, обнаружение и предотвращение атак (т. е. защиту от *сетевых атак*), хранит информацию обо всех действиях над системными структурами, полезную для анализа атак и борьбы с ними.

Система поддержки командного интерпретатора. Любая ОС поддерживает *командный язык* (или набор *командных языков*), состоящий из пользовательских команд, выполняемых с пользовательского терминала (из пользовательской консоли). Типичные команды – это получение информации об окружении, установка и смена текущей рабочей директории, пересылка файлов, компиляция и выполнение программ, получение информации о состоянии системы и выполнении своих процессов и др. В системе Windows для выполнения команд по традиции используется окно пользовательской консоли MS DOS (MS DOS Prompt), в системе Linux – специальное окно «Терминал» (Start/System Tools/Terminal). Наиболее мощные командные процессоры имеются в системах типа UNIX (UNIX, Solaris, Linux и др.). Их командные языки позволяют писать *скрипты* – командные файлы, содержащие часто используемые последовательности команд ОС. Что касается Windows, сравнительно недавно в ней появился мощный *командный интерпретатор PowerShell*, который и рекомендуется к использованию. Кроме того, для Windows имеется система *CygWin*, позволяющая выполнять команды и командные файлы UNIX в среде Windows.

Графическая оболочка – подсистема ОС, реализующая графический пользовательский *интерфейс* пользователей и системных администраторов с операционной системой. Разумеется, использование одного лишь командного языка и системных вызовов неудобно, поэтому простой и наглядный графический пользовательский *интерфейс* с ОС необходим. Имеется много известных графических оболочек для операционных систем, в частности, среди графических оболочек, используемых в *системах типа UNIX*, можно назвать *CDE, KDE, GNOME*. ОС *Windows* и *MacOS* имеют встроенные собственные удобные графические оболочки.

ГЛАВА 9
ФАЙЛОВЫЕ СИСТЕМЫ

9.1. Стандарты MBR и GPT

Разделы диска MBR и GPT. Даже в процессе установки операционная система предлагает выбрать стандарт MBR или GPT, определяющий структурирование информации на носителе, начальные и конечные точки разделов и содержание кода загрузки.

Стандарт MBR был представлен совместно с IBM PC DOS 2.0 в марте 1983 г. и используется до сих пор.

Стандарт GPT был разработан в конце 1990-х гг. как структурный элемент позже появившейся современной замены BIOS – UEFI и приобрел особую популярность лишь в последние годы.

Стандарт MBR, также известный как «главная загрузочная запись», – это традиционная структура для управления разделами диска. Реализуется на носителе путем записи в специальный сектор, расположенный в самом начале накопителя – в первом секторе жесткого магнитного диска. Запись содержит таблицу разделов – данные о разделах на диске, т. е. информацию об организации логических разделов на жестком магнитном диске. MBR также содержит исполняемый код, который сканирует разделы на предмет поиска активной операционной системы и инициализирует процедуру ее загрузки.

Диск со структурой MBR допускает только четыре основных раздела. Если необходимо иметь большее количество разделов, то можно назначить один из разделов «расширенным» и его разделить в свою очередь на подразделы или логические диски.

Преимущества структуры MBR заключаются в том, что она совместима с большинством операционных систем.

Недостатки структуры MBR в том, что она допускает только четыре раздела, но с возможностью создания дополнительных подразделов на одном из основных разделов, определенных как «расширенный». Другой недостаток, что информация о разделе хранится только в одном месте – в главной загрузочной записи. Если она повреждена, то весь диск становится нечитаемым и его невозможно использовать. При этом необходимо восстанавливать запись.

Стандарт GPT (GUID partition table) – новее, чем MBR (*master boot record*), для определения структуры разделов на диске. Постепенно GPT вытесняет MBR на носителях современных компьютеров, поскольку он обладает большим количеством преимуществ. При этом стоит отметить, что стандарт MBR до сих пор чрезвычайно актуален, поскольку master boot record обладает лучшей совместимостью и необходим в определенных случаях, когда устройство несовместимо со стандартом GPT. Более того, стандарт GPT не является эксклюзивным операционной системы Windows. Операционные системы Mac OS X и Linux также могут работать с носителями, которые используют структуру разметки GPT.

Для определения структуры диска стандарт GPT использует глобальные уникальные идентификаторы (GUID). Они называются таблицей разделов GUID, поскольку каждому разделу на диске присваивается «уникальный глобальный идентификатор» или GUID – случайная строка такой длины, что каждый GPT раздел на Земле, скорее всего, обладает уникальным идентификатором. GPT – это часть стандарта UEFI, т. е. систему на основе UEFI можно установить только на диск, использующий GPT.

Преимущества стандарта GPT заключаются в том, что он допускает создание неограниченного количества разделов, хотя некоторые операционные системы могут ограничивать их число,

в частности, ОС Windows – 128 разделами. Стандарт GPT хотя и имеет ограничение для дисков с секторами по 512 байт на максимальный размер раздела в 9,4 Збайта (один зеттабайт равен 1073741824 терабайт), но он пока больше, чем объем любых существующих дисков в настоящее время. Второе преимущество в том, что структура GPT хранит копию раздела и загрузочных данных и может восстановить данные в случае повреждения основного заголовка GPT. Более того, структура GPT хранит значения контрольной суммы по алгоритму циклического избыточного кода (CRC) для проверки целостности своих данных (используется для проверки целостности данных заголовка GPT). В случае повреждения GPT может заметить проблему и попытается восстановить поврежденные данные из другого места на диске.

На диске со структурой MBR данные о разделах и загрузочная информация хранятся в одном месте. Если эти данные повреждены или перезаписаны, то возникают проблемы. GPT же хранит несколько копий этих данных по всему диска, поэтому работает гораздо быстрее и позволяет восстановить поврежденную информацию. GPT так же хранит значения циклического избыточного кода (CRC), чтобы точно знать, что данные не тронуты. Если информация повреждена, GPT замечает проблему и пытается восстановить поврежденные данные с другого места на диске. MBR не может узнать о повреждении информации. Это можно увидеть только, если не загружается система или один из разделов диска исчезает.

Итак, прежде чем использовать диск, его необходимо разбить на разделы. MBR (*master boot record* – главная загрузочная запись) и GPT (таблица разделов GUID) представляют собой два различных способа хранения информации о разделах диска. На носителе необходимы записи – данные о начале и конце разделов, чтобы система знала, к какому разделу принадлежит

каждый сектор и какой раздел является загрузочным. Именно поэтому необходимо выбирать структуру MBR или GPT перед созданием разделов на диске.

На устройствах, использующих UEFI, 64-разрядные ОС семейства Windows могут запускаться только с раздела GPT. Помимо Windows таблицу разделов GUID поддерживают последние версии UNIX-систем.

Совместимость стандартов MBR и GPT с операционными системами. Диски со структурой GPT обычно включают «*защитный MBR*». Этот тип MBR сообщает системе, что GPT-диск представляет собой один большой раздел. Если Вы попытаетесь настроить GPT-диск старым инструментом, который может читать только MBR, он увидит один раздел, распространяющийся на весь диск. Таким образом, MBR предотвращает ситуацию, при которой старые инструменты посчитают GPT-диск неразмеченным и перепишут данные GPT информацией MBR. Другими словами, «защитный MBR» защищает данные GPT от перезаписи.

ОС Windows может загружаться с GPT-диска только на компьютерах с UEFI, работающих под управлением 64-битных версий и соответствующих серверных версий. Версии Windows, начиная с 8 и до Vista, могут читать GPT-диски и использовать их для хранения данных, но они не могут с них загружаться.

Другие современные операционные системы так же могут использовать GPT. ОС Linux имеет встроенную поддержку GPT.

Если необходима совместимость со старыми системами, например, возможность загружать Windows на компьютере с традиционным BIOS, придется пока остановиться на MBR.

Структура MBR- и GPT-дисков достаточно проста и представлена на рис. 29.

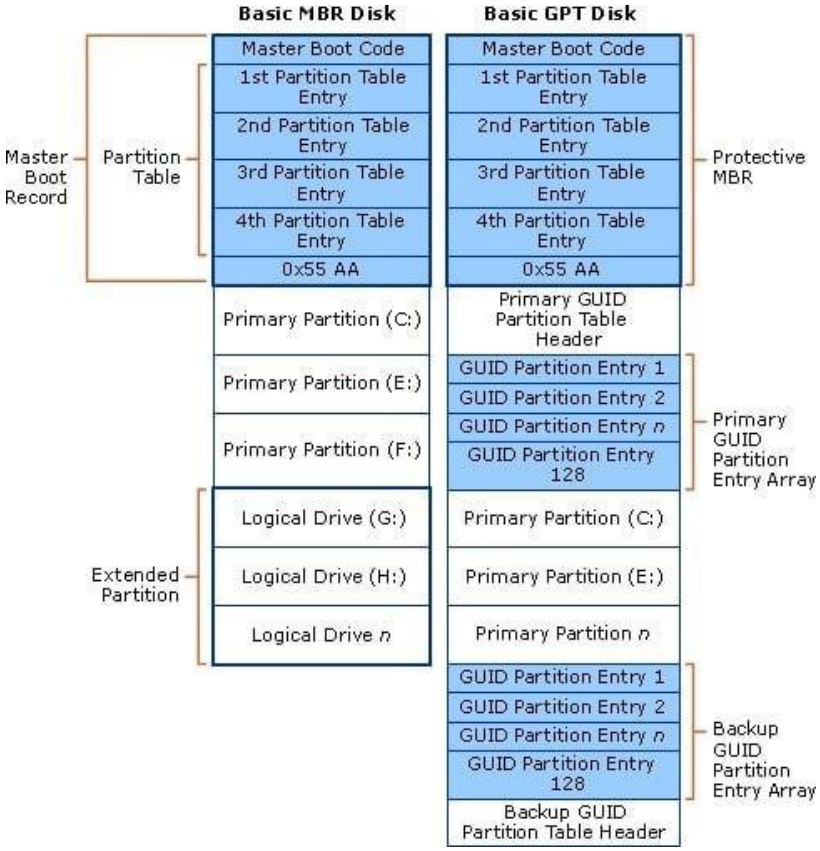


Рис. 29. Структура MBR- и GPT-дисков

Напомним, что MBR состоит из кода главного загрузчика, таблицы разделов жесткого диска и подписи диска (сигнатуры). При этом таблица разделов может иметь максимум четыре входа для основных разделов в ОС Windows. В структуру GPT входит так называемый «защитный MBR. Также туда входит первичный загрузчик таблицы разделов GUID (содержит информацию о своем размере и месте размещения, а также о размере и месте размещения второго загрузчика GPT); первичный вход в таблицу разделов GUID, backup (резервная копия) входа

в таблицу массивов GUID и backup загрузчика таблицы разделов GUID.

Большинство операционных систем Linux совместимы с GPT. При установке ОС Linux на диск в качестве загрузчика будет установлен, в частности, GRUB2.

Если материнская плата компьютера поддерживает только режим загрузки Legacy boot, то можно загрузить ОС Windows только с MBR-диска. Если необходимо установить ОС Windows на GPT-диск в этом режиме, то при попытке этого получите сообщение «*Windows не может быть установлен на этот диск. Выбранный диск имеет таблицу разделов GPT*». Однако, если материнская плата компьютера поддерживает только загрузку в UEFI, то можно загрузить ОС Windows только с GPT-диска. В ином случае получим ошибку, аналогичную уже упомянутой. Но, если материнская плата поддерживает оба режима (Legacy boot и UEFI boot) и при этом активировать CSM (Compatibility Support Module – модуль поддержки совместимости) в BIOSe, то возможно загрузить ОС Windows как с MBR-, так и с GPT-диска, или можно активировать UEFI, когда будет необходимость загрузить ОС с GPT-диска, или активировать Legacy BIOS, когда планируется загрузка ОС с MBR-диска.

Низкоуровневое и высокоуровневое форматирование носителей информации. Для НЖМД суть процедуры *низкоуровневого форматирования* (иногда говорят физического форматирования) состоит в том, что с помощью магнитных головок наносятся специальные магнитные метки на магнитные поверхности, которые и размечают дорожки. Кроме прочего дорожки отделяются друг от друга интервалами. В пределах каждой дорожки магнитными метками размечают сектора, которые также отделяются друг от друга интервалами. Сведения об обозначении дорожек и секторов в пределах дорожек заносятся в специально

выделяемые служебные области, создавая, таким образом, управляемые области диска, не подлежащие какому-либо форматированию.

Емкость сектора определяет наименьшее количество данных, которое может быть записано или считано за одну операцию ввода-вывода.

Напомним, что информация на магнитный диск записывается и считывается магнитными головками вдоль концентрических дорожек по секторам. При записи информации магнитный диск вращается вокруг своей оси, а головка с помощью специального механизма подводится к нужной дорожке и нужному сектору на ней. То же происходит и при чтении информации.

Необходимость низкоуровневого форматирования диска может потребоваться по нескольким причинам:

– *создание базовой разметки для дальнейшей работы с диском.* Выполняется после первого подключения нового диска к компьютеру, иначе его просто не будет видно среди локальных дисков;

– *удаление всех сохраненных файлов.* Так, за годы работы компьютера или ноутбука на диске скапливается огромное количество ненужных данных. Это не только пользовательские, но и системные файлы, которые уже не нужны, но при этом не удаляются самостоятельно. В результате может возникнуть переполнение накопителя, нестабильная и медленная работа. Самый простой вариант избавления от мусора – сохранить нужные файлы в специальное хранилище или на флеш-накопитель и отформатировать диск. Это в каком-то роде является радикальным методом оптимизации работы диска;

– *полная переустановка операционной системы.* Для более качественной и чистой установки операционной системы правильнее всего использовать чистый диск;

– *исправление ошибок.* Неустранимые вирусы и вредоносное программное обеспечение, поврежденные блоки, секторы и другие проблемы с диском нередко исправляются созданием новой разметки.

Низкоуровневое форматирование может использоваться как для новых накопителей, так и для бывших в употреблении. Форматировать новый НЖМД необходимо для создания разметки, без которой он не будет восприниматься операционной системой. Если на диске уже есть какая-либо информация, то она стирается.

Однако следует отметить, что низкоуровневое форматирование жесткого магнитного диска под операционной системой Linux невозможно. Хотя в этом нет особой необходимости, поскольку современные диски выпускаются отформатированными на низком уровне.

Во время процесса *высокоуровневого форматирования* носителей информации формируются файловая система и файловые таблицы. После этого носитель становится доступным для хранения данных. Форматирование на высоком уровне жесткого магнитного диска производится после разбиения на разделы, данные о местонахождении всех ранее записанных файлов стираются. Впоследствии можно полностью или частично восстановить данные в отличие от низкоуровневого форматирования, где происходит полное затирание информации.

Дополнительно к процедуре низкоуровневого форматирования добавляется создание одного раздела на НЖМД и далее осуществляется полное высокоуровневое форматирование (иногда говорят логическое форматирование) – процедура установки определенной файловой системы и создание при этом логического диска. Логическое форматирование заключается в оформлении диска соответственно стандартам операционной системы.

Целью логического форматирования является создание на диске управляющих таблиц, которые необходимы для учета использования имеющихся ресурсов. Они могут быть специально отведенным вполне определенным количеством секторов, например, для файловой системы FAT, или записываемыми впоследствии файлами, например, для файловой системы NTFS. В ходе установки файловой системы NTFS все пространство части раздела, созданного предварительно на НЖМД и выделенного под логический диск, делится на кластеры. В дальнейшем формируется собственно файловая система NTFS.

Форматирование на высоком уровне жесткого магнитного диска под операционной системой Linux заключается в создании на диске разделов и файловой системы. Для создания разделов под Linux используются программы fdisk, cfdisk и sfdisk. Программы fdisk и cfdisk позволяют создать качественную таблицу разделов, но имеются некоторые ограничения. Программа fdisk, хотя и позволяет произвести разбиение диска в большинстве случаев, но содержит несколько ошибок. Ее главное преимущество в том, что она поддерживает разделы DOS, BSD и других систем. Программа sfdisk работает более корректно, чем fdisk, и она гораздо мощнее fdisk и cfdisk, но имеет неудобный пользовательский интерфейс. Так что рекомендуется пытаться применять эти программы в следующем порядке: cfdisk, fdisk, sfdisk.

Под операционной системой Linux после создания файловой системы ее надо смонтировать в общее дерево каталогов. Делается это с помощью команды mount.

Итак, *полное высокоуровневое форматирование* предполагает полное удаление информации с жесткого магнитного диска – секторы перезаписываются нулями, вместе с этим файловая система проверяется на различные ошибки, исправляются плохие секторы (если точнее, они помечаются как непригодные для

дальнейшего хранения информации). Все это требует куда больше времени, вплоть до нескольких часов. Однако так информация будет надежно удалена, и ее потом не удастся восстановить даже специальными программами.

Быстрое высокоуровневое форматирование занимает не очень много времени, поскольку весь процесс сводится к удалению данных о местонахождении файлов в таблицах распределения файлов. При этом сами файлы никуда не исчезают и будут перезаписаны новой информацией – в результате операции происходит обозначение пустого места, куда в дальнейшем могут записываться новые файлы, «вытесняя» собой старые. Структура не оптимизируется, и если есть ошибки, то они пропускаются и не исправляются. Однако такой процесс занимает, как правило, до 1 мин в зависимости от объема, а данные могут быть восстановлены при помощи специального программного обеспечения частично или полностью.

Процедура установки операционной системы на логический диск. В этой ситуации осуществляется *системное форматирование*, отвечающее за дальнейшую установку операционной системы.

9.2. Принципы построения файловой системы

Организация файловой системы. *Файл* – понятие, привычное любому пользователю компьютера. Для пользователя каждый файл – это отдельный предмет, у которого есть начало и конец и который отличается от всех остальных файлов именем и расположением («как называется» и «где лежит»). Как и любой предмет, файл можно создать, переместить и уничтожить, однако, без внешнего вмешательства он будет сохраняться неизменным неопределенно долгое время. Файл предназначен для

хранения данных любого типа – текстовых, графических, звуковых, исполняемых программ и многого другого. Аналогия файла с предметом позволяет пользователю быстро освоиться при работе с данными в операционной системе.

Для операционной системы Linux файл – не менее важное понятие, чем для ее пользователя: все данные, хранящиеся на любых носителях, обязательно находятся внутри какого-нибудь файла, в противном случае они просто недоступны ни для операционной системы, ни для ее пользователей. Более того, все устройства, подключенные к компьютеру (начиная с клавиатуры и заканчивая любыми внешними устройствами, например, принтерами и сканерами), Linux представляет как файлы (так называемые *файлы-дырки*). Конечно, файл, содержащий обычные данные, сильно отличается от файла, предназначенного для обращения к устройству, поэтому в Linux определены несколько различных типов файлов. В основном, пользователь имеет дело с файлами трех типов: *обычными файлами*, предназначенными для хранения данных, *каталогами* и *файлами-ссылками*.

Файловая система с точки зрения пользователя – это «пространство», в котором размещаются файлы, наличие файловой системы позволяет определить не только, как называется файл, но и где он находится. Различать файлы лишь по имени было бы слишком неэффективным: про каждый файл приходилось бы помнить, как он называется, и при этом заботиться о том, чтобы имена никогда не повторялись. Более того, необходим механизм, позволяющий работать с группами тематически связанных между собой файлов (например, компонентов одной и той же программы или разных глав одной диссертации). Иначе говоря, файлы нужно *систематизировать*.

Файловая система определяет способ хранения и организации доступа к данным на носителе информации или его разделе.

Классическая файловая система имеет иерархическую структуру, в которой файл однозначно определяется **полным путем** к нему.

Большинство современных файловых систем используют в качестве основного организационного принципа *каталоги*. *Каталог* – это список ссылок на файлы или другие каталоги. Принято говорить, что каталог *содержит* в себе файлы или другие каталоги, хотя в действительности он только *ссылается* на них. Физическое размещение данных на диске обычно никак не связано с размещением каталога. Каталог, на который есть ссылка в данном каталоге, называется *подкаталогом* или *вложенным каталогом*. Каталог в файловой системе более всего напоминает библиотечный каталог, содержащий ссылки на объединенные по каким-то признакам книги и другие разделы каталога (файлы и подкаталоги). Ссылка на один и тот же файл может содержаться в нескольких каталогах одновременно, что может сделать доступ к файлу более удобным. В файловой системе Ext2 каждый каталог – это отдельный файл особого типа («d», от англ. *directory*), отличающийся от *обычного файла* с данными: в нем могут содержаться только ссылки на другие файлы и каталоги.

Довольно часто вместо термина *каталог* можно встретить термин *папка* (*folder*). Этот термин хорошо вписывается в представление о файлах как о предметах, которые можно раскладывать по папкам. Однако часть возможностей файловой системы, которая противоречит этому представлению, таким образом, становится непонятной. В частности, с термином «папка» плохо согласуется то, что ссылка на файл может присутствовать одновременно в нескольких каталогах, файл может быть ссылкой на другой файл и т. д. В Linux эти возможности файловой системы весьма важны для эффективной работы, поэтому будем всюду использовать более подходящий термин «каталог».

В файловой системе, организованной при помощи каталогов, на любой файл должна быть ссылка как минимум из одного каталога, в противном случае файл просто не будет доступен внутри этой файловой системы, иначе говоря, не будет существовать.

Имена файлов и каталогов. *Допустимые имена.* Главные отличительные признаки файлов и каталогов – их имена. В Linux имена файлов и каталогов могут быть длиной не более 256 символов и могут содержать любые символы кроме «/». Причина этого ограничения очевидна: этот символ используется как разделитель имен в составе пути, поэтому не должен встречаться в самих именах. Причем Linux всегда различает прописные и строчные буквы в именах файлов и каталогов, поэтому «methody», «Methody» и «METHODY» будут тремя *разными* именами.

Есть несколько символов, допустимых в именах файлов и каталогов, которые нужно использовать с осторожностью. Это – так называемые *спецсимволы* «*», «\», «&», «<», «>», «;», «(», «)», «|», а также пробелы и табуляции. Дело в том, что эти символы имеют особое значение для любой *командной оболочки*, поэтому надо специально позаботиться о том, чтобы командная оболочка воспринимала эти символы как часть имени файла или каталога.

Кодировки и русские имена. Как можно было заметить, пока во всех встречавшихся именах файлов и каталогов употреблялись только символы латинского алфавита и некоторые знаки препинания. Это не случайно и вызвано желанием обеспечить тот факт, чтобы приводимые примеры совершенно одинаково выглядели на любых системах. В Linux в именах файлов и каталогов допустимо использовать любые символы любого языка, однако, такая свобода связана с некоторыми ограничениями.

Дело в том, что каждый символ (буква) каждого языка традиционно представлялся в виде *одного* байта. Такое представ-

ление накладывает очень жесткие ограничения на *количество* букв в алфавите: их может быть не больше 256, а за вычетом управляющих символов, цифр, знаков препинания и прочего – и того меньше. Обширные алфавиты (например, иероглифические японский и китайский) пришлось заменять упрощенным их представлением. Вдобавок, первые 128 символов из этих 256 лучше всегда оставлять неизменными, соответствующими стандарту ASCII, включающему латиницу, цифры, знаки препинания и наиболее популярные символы из тех, что встречаются на клавиатуре печатной машинки. Интерпретация остальных 128 символов зависит от того, какая *кодировка* установлена в системе. Например, в русской кодировке KOI8-R 228-й символ такой таблицы соответствует букве «Д», а в западноевропейской кодировке ISO-8859-1 этот же символ соответствует букве «а» с двумя точками на ней (как у нашей буквы «е»).

Имена файлов, *записанные* на диск в одной кодировке, выглядят нелепо, если при *просмотре* каталога была установлена другая кодировка. Хуже того. Многие кодировки заполняют диапазон символов с номерами от 128 до 255 *не полностью*, поэтому соответствующего символа может вообще не быть! Это означает, что *ввести* такое искаженное имя файла с клавиатуры (для того, чтобы его переименовать) напрямую не удастся. Наконец, многие языки, в том числе и русский, исторически имеют *несколько* кодировок. К сожалению, в настоящее время нет стандартного способа указывать кодировку прямо в имени файла, поэтому в рамках одной файловой системы *стоит* придерживаться единой кодировки при именовании файлов.

Существует универсальная кодировка, включающая символы всех письменностей мира – UNICODE. Стандарт UNICODE в настоящее время получил большое распространение и имеет статус общего для всех текстов, хранящихся в электронной фор-

ме. Однако пока нет желаемой универсальности, особенно, в области имен файлов. *Один* символ в UNICODE может занимать *больше* одного байта – и в этом главный его недостаток, так как множество полезных прикладных программ, отлично работающих с *однобайтными* кодировками, необходимо основательно или даже полностью перерабатывать для того, чтобы «научить» их обращаться с UNICODE.

Это не означает, что, называя файлы, не следует использовать языки, отличные от английского. Пока точно известно, в какой кодировке задано имя файла – проблем не возникнет. Гораздо более легкий способ передать файл – использовать в его названии *только* символы ASCII.

Расширения. Многим пользователям знакомо понятие *расширение* – часть имени файла после точки, обычно ограничивающаяся несколькими символами и указывающая на тип содержащихся в файле данных. В файловой системе Linux нет никаких предписаний по поводу расширения: в имени файла может быть любое количество точек (в том числе и ни одной), а после последней точки может быть любое количество символов. Хотя расширения не обязательны и не навязываются технологией в Linux, они широко используются: расширение позволяет человеку или программе, не открывая файл, только по его имени определить, какого типа данные в нем содержатся. Однако нужно учитывать, что расширение – это только набор соглашений по наименованию файлов разных типов. Строго говоря, данные в файле могут не соответствовать заявленному расширению по той или иной причине, поэтому всецело полагаться на расширение нельзя.

Определить тип содержимого файла можно и на основании самих данных. Многие форматы предусматривают указание в начале файла, как следует интерпретировать дальнейшую информацию: как программу, исходные данные для текстового редак-

тора, страницу HTML, звуковой файл, изображение или что-то другое. В распоряжении пользователя Linux всегда есть утилита *file*, которая предназначена именно для определения типа данных, содержащихся в файле.

Утилита *file* умеет различать очень многие типы данных и почти наверняка выдает правильную информацию. Эта утилита никогда не «доверяет» расширению файла, если даже оно присутствует, и анализирует сами данные. Она различает не только разные данные, но и разные типы файлов, в частности, сообщает, если исследуемый объект не является *обычным файлом*, а, например, каталогом.

Дерево каталогов. Понятие каталога позволяет *систематизировать* все объекты, размещенные на носителе данных (например, на диске). В большинстве современных файловых систем используется иерархическая модель организации данных: существует один каталог, объединяющий все данные в файловой системе – это «корень» всей файловой системы, *корневой каталог*. Корневой каталог может содержать любые объекты файловой системы и, в частности, подкаталоги (каталоги первого *уровня вложенности*). Те, в свою очередь, также могут содержать любые объекты файловой системы и подкаталоги (второго уровня вложенности) и т. д. Таким образом, все, что записано на диске – файлы, каталоги и специальные файлы – обязательно «принадлежит» корневому каталогу: либо непосредственно (содержится в нем), либо на некотором уровне вложенности.

Иерархию вложенных друг в друга каталогов можно соотнести с иерархией данных в системе: объединить тематически связанные файлы в каталог, тематически связанные каталоги – в один общий каталог и т. д. Если строго следовать иерархическому принципу, то чем глубже будет *уровень вложенности* каталога, тем более частным признаком должны быть объединены

содержащиеся в нем данные. Если этому принципу *не* следовать, то вскоре окажется, что гораздо проще складывать *все* файлы в один каталог и искать нужный среди них, чем проделывать такой поиск по всем подкаталогам системы. Однако в этом случае, о какой бы то ни было *систематизации* файлов, говорить не приходится.

Структуру файловой системы можно представить наглядно в виде дерева, «корнем» которого является корневой каталог, а в вершинах расположены все остальные каталоги. На рис. 30 изображено дерево каталогов, курсивом обозначены имена файлов, прямым начертанием – имена каталогов.

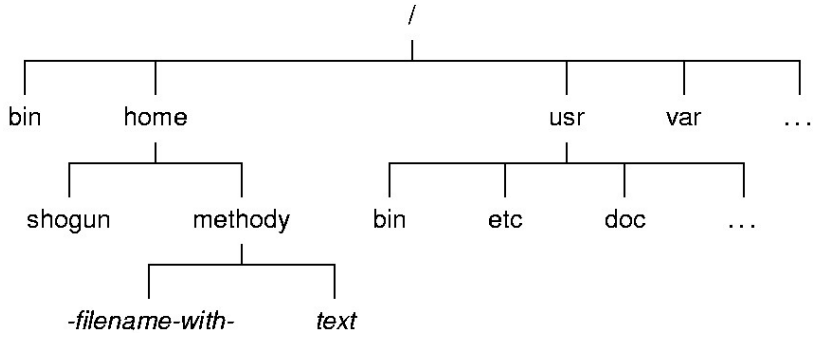


Рис. 30. Дерево каталогов в Linux

В любой файловой системе Linux всегда есть только один *корневой каталог*, который называется «/». Пользователь Linux всегда работает с единым деревом каталогов, даже если разные данные расположены на разных носителях: нескольких жестких или сетевых дисках, съемных дисках, CD-ROM и др. Для того, чтобы подключать файловые системы на разных устройствах в одно общее дерево и отключать их, используются процедуры *монтирования* и *размонтирования*. После того, как файловые системы на разных носителях подключены к общему дереву, содержащиеся на них данные доступны так, как если бы все они

составляли единую файловую систему: пользователь может даже не знать, на каком устройстве какие файлы хранятся.

Положение любого каталога в дереве каталогов точно и однозначно описывается при помощи *полного пути*. Полный путь всегда начинается от корневого каталога и состоит из перечисления всех вершин, встретившихся при движении по ребрам дерева до искомого каталога включительно. Названия соседних вершин разделяются символом «/» («слеш»). В Linux полный путь, например, до каталога «methody» в файловой системе, приведенной на рис. 30, записывается следующим образом: сначала символ «/», обозначающий корневой каталог, затем к нему добавляется «home», затем разделитель «/», за которым следует название искомого каталога «methody», в результате получается полный путь «/home/methody».

Расположение файла в файловой системе аналогичным образом определяется при помощи полного пути, только последним элементом в данном случае будет не название каталога, а название файла.

Организация каталогов файловой системы в виде дерева не допускает появления циклов: т. е. каталог не может содержать в себе каталог, в котором содержится сам. Благодаря этому ограничению полный путь до любого каталога или файла в файловой системе всегда будет *конечным*.

Размещение компонентов системы. Стандарт FHS (Filesystem Hierarchy Standard – стандартная структура файловых систем). Попробуем более подробно разобраться, как устроено дерево каталогов Linux, где и что в нем можно найти. Фрагмент дерева каталогов типичной файловой системы Linux (SomeLinux) приведен на рис. 30. В целях изучения файловой системы, начиная с *корневого каталога*, необходимо использовать команду: ls *каталог*, где *каталог* – это полный путь

к каталогу. Утилита *ls* выведет список всего, что в этом каталоге содержится.

Стандартные каталоги в корневом каталоге («/»). Утилита *ls* вывела список подкаталогов корневого каталога. В корневом каталоге Linux-системы обычно находятся только подкаталоги со *стандартными* именами. Более того, не только имена, но и *тип данных*, которые могут попасть в тот или иной каталог, также регламентированы стандартом *FHS*.

Опишем кратко, что находится в каждом из подкаталогов корневого каталога. Для этого используется команда: *ls имя каталога*, где *каталог* – это полный путь к каталогу.

Каталог /bin. Название этого каталога происходит от англ. *binaries* (двоичные, исполняемые). В этом каталоге находятся исполняемые файлы самых необходимых утилит. Сюда попадают такие программы, которые могут понадобиться системному администратору или другим пользователям для устранения неполадок в системе или при восстановлении после сбоя.

Каталог /boot. Boot – загрузка системы. В этом каталоге находятся файлы, необходимые для самого первого этапа загрузки: загрузки ядра и, обычно, само ядро. Пользователю практически никогда не требуется непосредственно работать с этими файлами.

Каталог /dev. В этом каталоге находятся все имеющиеся в системе *файлы-дырки*: файлы особого типа, предназначенные для обращения к различным системным ресурсам и устройствам (англ. *devices* – устройство, отсюда и сокращенное название каталога). Например, файлы /dev/tty*N* соответствуют *виртуальным консолям*, где *N* – номер виртуальной консоли. Данные, введенные пользователем на первой виртуальной консоли, система считывает из файла /dev/tty1, в этот же файл записываются данные, которые нужно вывести пользователю на эту консоль. В фай-

лах-дырках в действительности не хранятся никакие данные, при их помощи данные *передаются*.

Каталог /etc. Каталог для системных *конфигурационных файлов*. В нем хранится информация о специфических настройках данной системы: информация о зарегистрированных пользователях, доступных ресурсах, настройках различных программ.

Каталог /home. Здесь расположены каталоги, принадлежащие пользователям системы – *домашние каталоги*, отсюда и название *home*. Отделение всех файлов, создаваемых пользователями, от прочих системных файлов дает очевидное преимущество: серьезное повреждение системы или необходимость обновления не затронет наиболее ценной информации – пользовательских файлов.

Каталог /lib. Название этого каталога – сокращение от англ. *libraries* – библиотеки. *Библиотеки* – это собрания наиболее стандартных функций, необходимых многим программам: операций ввода/вывода, рисования элементов графического интерфейса и проч. Чтобы не включать эти функции в текст каждой программы, используются стандартные функции библиотек – это значительно экономит место на диске и упрощает написание программ. В этом каталоге содержатся библиотеки, необходимые для работы наиболее важных системных утилит (размещенных в /bin и /sbin).

Каталог /mnt. Каталог для *монтирования* (от англ. *mount*) – временного подключения файловых систем, например, на съемных носителях (CD-ROM и др.).

Каталог /proc. В этом каталоге все файлы «виртуальные» – они располагаются не на диске, а в оперативной памяти. В этих файлах содержится информация о программах (*процессах*), выполняемых в данный момент в системе.

Каталог /root. Домашний каталог администратора системы – пользователя root. Смысл размещать его отдельно от домашних каталогов остальных пользователей состоит в том, что /home может располагаться на отдельном устройстве, которое не всегда доступно (например, на сетевом диске), а домашний каталог /root должен присутствовать в любой ситуации.

Каталог /sbin. Каталог для важнейших системных утилит (название каталога – сокращение от «systembinaries»): в дополнение к утилитам, находящимся в /bin, здесь находятся программы, необходимые для загрузки, резервного копирования, восстановления системы. Полномочия на исполнение этих программ есть только у системного администратора.

Каталог /tmp. Этот каталог предназначен для *временных файлов*: в таких файлах программы хранят промежуточные данные, необходимые для работы. После завершения работы программы временные файлы теряют смысл и должны быть удалены. Обычно каталог /tmp очищается при каждой загрузке системы.

Каталог /usr. Каталог /usr – это «государство в государстве». Здесь можно найти такие же подкаталоги /bin, /etc, /lib, /sbin, как и в корневом каталоге. Однако в корневой каталог попадают только утилиты, *необходимые* для загрузки и восстановления системы в аварийной ситуации, *все остальные* программы и данные располагаются в подкаталогах каталога /usr. Прикладных программ в современных системах обычно установлено очень много, поэтому этот раздел файловой системы может быть очень большим.

Каталог /var. Название этого каталога – сокращение от «variable» («переменные» данные). Здесь размещаются те данные, которые создаются в процессе работы разными программами и предназначены для передачи другим программам и систе-

мам (очереди печати и электронной почты и др.) или для сведения системного администратора (*системные журналы*, содержащие протоколы работы системы). В отличие от каталога /tmp сюда попадают те данные, которые могут понадобиться после того, как создавшая их программа завершила работу.

Стандарт *FHS* регламентирует не только перечисленные каталоги, но и их подкаталоги, а иногда даже приводит список конкретных файлов, которые должны присутствовать в определенных каталогах. Этот стандарт последовательно соблюдается во всех Linux-системах.

Стандартное размещение файлов позволяет и пользователю, и программе предсказать, где находится тот или иной компонент системы. Для пользователя это означает, что он сможет быстро сориентироваться в любой системе Linux (где файловая система организована в соответствии со стандартом) и найти то, что ему нужно. Для программ стандартное расположение файлов – это возможность организации автоматического взаимодействия между разными компонентами системы.

Использование стандартного расположения файлов дает определенные преимущества: можно запускать утилиты, не указывая полный путь к исполняемому файлу, например, *cat* вместо /bin/cat. Командная оболочка «знает», что исполняемые файлы располагаются в каталогах /bin, /usr/bin и т. д. – именно в этих каталогах она ищет исполняемый файл *cat*. Благодаря этому каждая вновь установленная в системе программа немедленно оказывается доступна пользователю из командной строки, для этого не требуется ни перезагружать систему, ни запускать никаких процедур – достаточно просто поместить исполняемый файл в один из соответствующих каталогов.

Рекомендации стандарта по размещению файлов и каталогов основываются на принципе разносить в разные подкаталоги

файлы, которые по-разному используются в системе. По типу использования файлов их можно разделить на следующие группы:

1. *Пользовательские и системные файлы.* Пользовательские файлы – это все файлы, созданные пользователем и не принадлежащие ни одному из компонентов системы.

2. *Изменяющиеся и неизменные файлы.* К неизменным файлам относятся все статические компоненты программного обеспечения: библиотеки, исполняемые файлы и др. – все, что не изменяется само без вмешательства системного администратора. Изменяющиеся файлы – это те, которые изменяются без вмешательства пользователя в процессе работы системы: *системные журналы*, очереди печати и пр. Выделение неизменных файлов в отдельную структуру (например, /usr) позволяет использовать соответствующую часть файловой системы в режиме «только чтение», что уменьшает вероятность случайного повреждения данных и позволяет использовать для хранения этой части файловой системы CD-ROM и другие носители, доступные только для чтения.

3. *Разделяемые и неразделяемые файлы.* Это разграничение становится полезным, если речь идет о сети, в которой работает несколько компьютеров. Значительная часть информации при этом может храниться на одном из компьютеров и использоваться всеми остальными по сети (к такой информации относятся, например, многие программы и домашние каталоги пользователей). Однако часть файлов нельзя разделять между системами (например, файлы для начальной загрузки системы).

9.3. Организация управления файлами

Файловая система. Одной из основных задач ОС является предоставление удобств пользователю при работе с данными, хранящимися на дисках. Для этого ОС подменяет физическую структуру хранящихся данных некоторой удобной для

пользователя логической моделью, которая реализуется в виде дерева каталогов, выводимого на экран такими утилитами, как NortonCommander, FarManager или WindowsExplorer. Базовым элементом этой модели является *файл*, который так же, как и *файловая система* в целом, может характеризоваться как логической, так и физической структурой.

Управление файлами. Файлы хранятся в памяти, не зависящей от энергопитания. Исключением является электронный диск, когда в оперативной памяти (ОП) создается структура, имитирующая файловую систему.

Файловая система (ФС) – это компонент ОС, обеспечивающий организацию создания, хранения и доступа к именованным наборам данных – файлам.

Файловая система включает:

– совокупность всех фалов на диске;

– наборы структур данных, используемых для управления файлами (каталоги файлов, дескрипторы файлов, таблицы распределения свободного и занятого пространства на диске);

– комплекс системных программных средств, реализующих различные операции над файлами: создание, уничтожение, чтение, запись, именование, поиск.

Задачи, решаемые ФС, зависят от способа организации вычислительного процесса в целом. Самый простой тип – это ФС в однопользовательских и однопрограммных ОС. Основные функции в такой ФС нацелены на решение следующих задач:

– именование файлов;

– программный интерфейс для приложений;

– отображения логической модели ФС на физическую организацию хранилища данных;

– устойчивость ФС к сбоям питания, ошибкам аппаратных и программных средств.

Задачи ФС усложняются в однопользовательских многозадачных ОС, которые предназначены для работы одного пользователя, но дают возможность запускать одновременно несколько процессов. К перечисленным выше задачам добавляется новая задача – совместный доступ к файлу из нескольких процессов.

Файл в этом случае является разделяемым ресурсом, а значит, ФС должна решать весь комплекс проблем, связанных с такими ресурсами. В частности: должны быть предусмотрены средства блокировки файла и его частей, согласование копий, предотвращение гонок, исключение тупиков. В многопользовательских системах появляется еще одна задача: защита файлов одного пользователя от несанкционированного доступа другого пользователя.

Основное назначение *файловой системы* и соответствующей ей *системы управления файлами* – организация удобного управления файлами, организованными как файлы: вместо низкоуровневого доступа к данным с указанием конкретных физических адресов нужной пользователю записи, используется логический доступ с указанием имени файла и записи в нем.

Термины «файловая система» и «система управления файлами» необходимо различать: файловая система определяет, прежде всего, принципы доступа к данным, организованным как файлы. А термин «система управления файлами» следует употреблять по отношению к конкретной реализации файловой системы, т. е. это комплекс программных модулей, обеспечивающих работу с файлами в конкретной ОС.

Типы файлов. *Обычные файлы* содержат информацию произвольного характера, которую заносит в них пользователь или которая образуется в результате работы системных и пользовательских программ. Содержание обычного файла определяется приложением, которое с ним работает.

Обычные файлы могут быть двух типов:

1. *Программные (исполняемые)* – представляют собой программы, написанные на командном языке ОС, и выполняют некоторые системные функции (имеют расширения .exe, .com, .bat).

2. *Файлы данных* – все прочие типы файлов: текстовые и графические документы, электронные таблицы, базы данных и др.

Специальные файлы – это фиктивные файлы, ассоциированные с устройствами ввода/вывода, которые используются для унификации механизма доступа к файлам и внешним устройствам. Специальные файлы позволяют пользователю осуществлять операции ввода/вывода посредством обычных команд записи с файлов или чтения из файлов. Эти команды обрабатываются сначала программами ФС, а затем на некотором этапе выполнения запроса преобразуются ОС в команды управления соответствующим устройством (PRN, LPT1 – для порта принтера (символьные имена, для ОС – это файлы), CON – для клавиатуры).

Файловая структура. *Файловая структура* – вся совокупность файлов на диске и взаимосвязей между ними (порядок хранения файлов на диске).

Виды файловых структур:

– *простая* или *одноуровневая*: каталог представляет собой линейную последовательность файлов;

– *иерархическая* или *многоуровневая*: каталог сам может входить в состав другого каталога и содержать внутри себя множество файлов и подкаталогов. Иерархическая структура может быть двух видов: «Дерево» и «Сеть». Каталоги образуют «Дерево», если файлу разрешено входить только в один каталог (ОС MS-DOS, Windows), и «Сеть» – если файл может входить сразу в несколько каталогов (UNIX, LINUX).

Файловая структура может быть представлена в виде графа, описывающего иерархию каталогов и файлов (рис. 31).

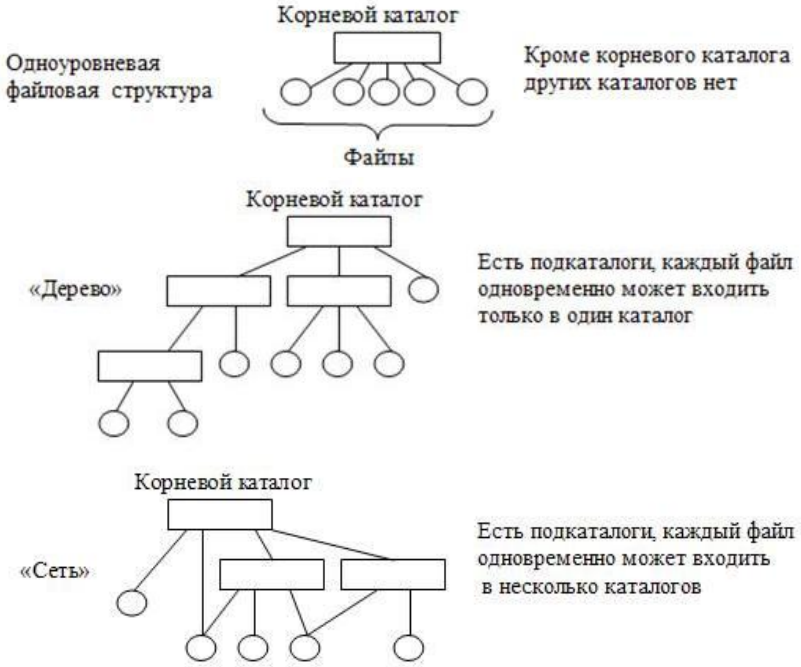


Рис. 31. Представление файловой структуры в виде графа

Типы имен файлов. Файлы идентифицируются именами. Пользователи дают файлам *символьные имена*, при этом учитываются ограничения ОС как на используемые символы, так и на длину имени. В ранних файловых системах эти границы были весьма узкими. Так, в ранней популярной *файловой системе FAT* длина имен ограничивалась известной схемой 8.3 (8 символов – собственно имя, 3 символа – расширение имени), а в ОС UNIX System V имя не могло содержать более 14 символов.

Однако пользователю гораздо удобнее работать с длинными именами, поскольку они позволяют дать файлу действительно

мнемоническое название, по которому даже через достаточно большой промежуток времени можно будет вспомнить, что содержит этот файл. Поэтому современные файловые системы, как правило, поддерживают длинные символьные имена файлов.

Например, в Windows NT в своей файловой системе NTFS устанавливалось, что имя файла может содержать до 255 символов, не считая завершающего нулевого символа.

При переходе к длинным именам возникает проблема совместимости с ранее созданными приложениями, использующими короткие имена. Чтобы приложения могли обращаться к файлам в соответствии с принятыми ранее соглашениями, файловая система должна уметь предоставлять эквивалентные короткие имена (псевдонимы) файлам, имеющим длинные имена. Таким образом, одной из важных задач становится проблема генерации соответствующих коротких имен.

Символьные имена могут быть трех типов: *простые*, *составные* и *относительные*:

1. *Простое имя* идентифицирует файл в пределах одного каталога, присваивается файлам с учетом номенклатуры символа и длины имени.

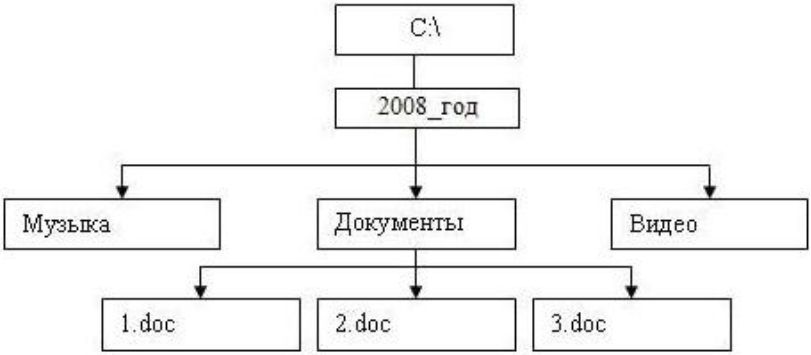
2. *Полное имя* представляет собой цепочку простых символьных имен всех каталогов, через которые проходит путь от корня до данного файла, имени диска, имени файла. Таким образом, полное имя является составным, в котором простые имена отделены друг от друга принятым в ОС разделителем.

3. Файл может быть идентифицирован также *относительным именем*. Относительное имя файла определяется через понятие «текущий каталог». В каждый момент времени один из каталогов является текущим, причем этот каталог выбирается самим пользователем по команде ОС. Файловая система фиксирует имя текущего каталога, чтобы затем использовать его как

дополнение к относительным именам для образования полного имени файла.

В древовидной файловой структуре между файлом и его полным именем имеется взаимно однозначное соответствие – «один файл – одно полное имя». В сетевой файловой структуре файл может входить в несколько каталогов, а, значит, может иметь несколько полных имен; здесь справедливо соответствие – «один файл – много полных имен».

Пример. *Для файла «2.doc» нужно определить все три типа имени, при условии, что текущим каталогом является каталог «2008_год».*



Простое имя: 2.doc

Полное имя: C:\2008_год\Документы\2.doc

Относительное имя: Документы\2.doc

Атрибуты файлов. Важной характеристикой файла являются атрибуты. *Атрибуты* – это информация, описывающая свойства файлов. Примеры возможных атрибутов файлов:

– признак «только для чтения» (*read-only*);
– признак «скрытый файл» (*hidden*);
– признак «системный файл» (*system*);
– признак «архивный файл» (*archive*);

– тип файла (обычный файл, каталог, специальный файл);

– владелец файла;

– создатель файла;

– пароль для доступа к файлу;

– информация о разрешенных операциях доступа к файлу;

– время создания, последнего доступа и последнего изменения;

– текущий размер файла;

– максимальный размер файла;

– признак «временный (удалить после завершения процесса)»;

– признак блокировки.

В файловых системах разного типа для характеристики файлов могут использоваться разные наборы атрибутов (так, в однопользовательской ОС в наборе атрибутов будут отсутствовать характеристики, имеющие отношение к пользователю и защите – создатель файла, пароль для доступа к файлу и проч.).

Пользователь может получать доступ к атрибутам, используя средства, предоставленные для этих целей файловой системой. Обычно разрешается читать значения любых атрибутов, а изменять – только некоторые, например, можно изменить права доступа к файлу, но нельзя изменить дату создания или текущий размер файла.

Права доступа к файлу. Определить права доступа к файлу – значит определить для каждого пользователя набор операций, которые он может применить к данному файлу. В разных файловых системах может быть определен свой список дифференцируемых операций доступа. Этот список может включать следующие операции: создание файла; уничтожение файла; запись в файл; открытие файла; закрытие файла; чтение из файла; дополнение файла; поиск в файле; получение атрибутов файла;

установление новых значений атрибутов; переименование; выполнение файла; чтение каталога и др.

В самом общем случае *права доступа* могут быть описаны матрицей прав доступа, в которой столбцы соответствуют всем файлам системы, строки – всем пользователям, а на пересечении строк и столбцов указываются разрешенные операции:

Имена пользователей	Имена файлов			
	dok.txt	prog.exe	baza.dbf	unix.ppt
User1	*читать*	*выполнять*	–	*выполнять*
User2	*читать*	*выполнять*	–	*выполнять* *читать*
User3	*читать*	–	–	*выполнять* *читать*
User4	*читать* *писать*	–	*создать*	–

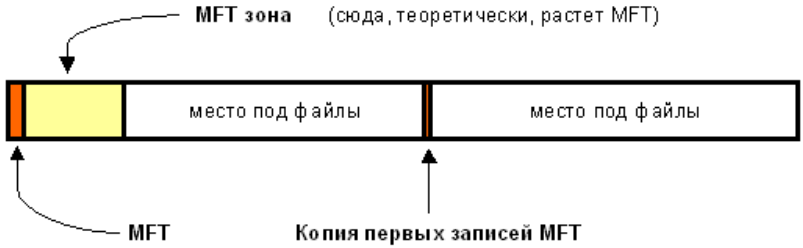
В некоторых системах пользователи могут быть разделены на отдельные категории. Для всех пользователей одной категории определяются единые права доступа. Так, в системе UNIX все пользователи подразделяются на три категории: владелец файла, члены его группы и все остальные.

9.4. Файловая система NTFS

Физическая структура NTFS. Раздел NTFS, теоретически, может быть почти любого размера. Предел, конечно, есть, но его с запасом хватит на последующее развитие вычислительной техники – при любых темпах роста. На практике максимальный размер раздела NTFS в данное время ограничен лишь объемами жестких дисков.

Структура раздела. Как и любая другая система, NTFS делит все полезное место на кластеры – блоки данных, используемые единовременно. NTFS поддерживает почти любые размеры кластеров – от 512 байт до 64 Кбайт, неким стандартом же считается кластер размером 4 Кбайт. Никаких аномалий кластерной структуры NTFS не имеет.

Диск NTFS условно делится на две части. Первые 12 % диска отводятся под так называемую MFT зону – пространство, в которое растет метафайл MFT. Запись каких-либо данных в эту область невозможна. MFT-зона всегда держится пустой – это делается для того, чтобы самый главный, служебный файл (MFT) не фрагментировался при своем росте. Остальные 88 % диска представляют собой обычное пространство для хранения файлов.



Свободное место диска, однако, включает в себя все физически свободное место – незаполненные куски MFT-зоны туда тоже включаются. Механизм использования MFT-зоны таков: когда файлы уже нельзя записывать в обычное пространство, MFT-зона просто сокращается (в текущих версиях операционных систем – ровно в два раза), освобождая, таким образом, место для записи файлов. При освобождении места в обычной области MFT-зона может снова расшириться. При этом не исключена ситуация, когда в этой зоне остались и обычные файлы: никакой аномалии тут нет. Система старалась оставить ее свободной, но ничего не

получилось. Метафайл MFT все-таки может фрагментироваться, хоть это и было бы нежелательно.

MFT и его структура. Файловая система NTFS представляет собой выдающееся достижение структуризации: *каждый* элемент системы представляет собой файл – даже служебная информация. Самый главный файл на NTFS называется MFT, или Master File Table – общая таблица файлов. Именно он размещается в MFT зоне и представляет собой централизованный каталог всех остальных файлов диска, а также себя самого. MFT поделен на записи фиксированного размера (обычно 1 Кбайт), каждая запись соответствует какому-либо файлу. Первые 16 файлов носят служебный характер и недоступны операционной системе – они называются метафайлами, причем самый первый метафайл – сам MFT. Эти первые 16 элементов MFT – единственная часть диска, имеющая фиксированное положение. Интересно, что вторая копия первых трех записей, для надежности – они очень важны – хранится ровно посередине диска. Остальной MFT-файл может располагаться, как и любой другой файл, в произвольных местах диска – восстановить его положение можно с помощью его самого, «зацепившись» за самую основу – за первый элемент MFT.

Метафайлы. Первые 16 файлов NTFS (метафайлы) носят служебный характер. Каждый из них отвечает за какой-либо аспект работы системы. Преимущество такого модульного подхода заключается в поразительной гибкости – например, на FAT-е физическое повреждение в самой области FAT фатально для функционирования всего диска, а NTFS может сместить, даже фрагментировать по диску, все свои служебные области, обойдя любые неисправности поверхности – кроме первых 16 элементов MFT.

Метафайлы находятся в корневом каталоге NTFS-диска – они начинаются с символа имени «$», но получить какую-либо информацию о них стандартными средствами сложно. Хотя и для

этих файлов указан вполне реальный размер – можно узнать, например, сколько операционная система тратит на каталогизацию всего вашего диска, посмотрев размер файла $MFT.

Таблица 13

Используемые метафайлы и их назначение

$MFT	Сам MFT
$MFTmirr	Копия первых 16 записей MFT, размещенная посередине диска
$LogFile	Файл поддержки журналирования
$Volume	Служебная информация – метка тома, версия файловой системы, т. д.
$AttrDef	Список стандартных атрибутов файлов на томе
$.	Корневой каталог
$Bitmap	Карта свободного места тома
$Boot	Загрузочный сектор (если раздел загрузочный)
$Quota	Файл, в котором записаны права пользователей на использование дискового пространства (начал работать лишь в NT5)
$Upcase	Файл – таблица соответствия заглавных и прописных букв в именах файлов на текущем томе. Нужен, в основном, потому что в NTFS имена файлов записываются в Unicode, что составляет 65 тысяч различных символов, искать большие и малые эквиваленты которых очень сложно.

Файлы и потоки. Прежде всего, обязательный элемент – запись в MFT, ведь, как было сказано ранее, все файлы диска упоминаются в MFT. В этом месте хранится вся информация о файле, за исключением собственно данных. Имя файла, размер, положение на диске отдельных фрагментов и др. Если для информации не хватает одной записи MFT, то используются несколько, причем не обязательно подряд.

Опциональный элемент – потоки данных файла. Может показаться странным определение «опциональный», но, тем не ме-

нее, ничего странного тут нет. Во-первых, файл может не иметь данных – в таком случае на него не расходуется свободное место самого диска. Во-вторых, файл может иметь не очень большой размер. Тогда идет в ход довольно удачное решение: данные файла хранятся прямо в MFT, в оставшемся от основных данных месте в пределах одной записи MFT. Файлы, занимающие сотни байт, обычно не имеют своего «физического» воплощения в основной файловой области – все данные такого файла хранятся в одном месте – в MFT.

Довольно интересно обстоит дело и с данными файла. Каждый файл на NTFS, в общем-то, имеет несколько абстрактное строение – у него нет как таковых данных, а есть потоки (streams). Один из потоков и носит привычный для нас смысл – данные файла. Но большинство атрибутов файла – тоже потоки! Таким образом, получается, что базовая сущность у файла только одна – номер в MFT, а все остальное опционально. Данная абстракция может использоваться для создания удобных объектов – например, файлу можно «прицепить» еще один поток, записав в него любые данные – информацию об авторе и содержании файла, как это было сделано в Windows 2000 (самая правая закладка в свойствах файла, просматриваемых из проводника). Но эти дополнительные потоки не видны стандартными средствами: наблюдаемый размер файла – это лишь размер основного потока, который содержит традиционные данные. Можно, к примеру, иметь файл нулевой длинны, при стирании которого освободится 1 Гбайт свободного места – просто потому, что какая-нибудь «хитрая» программа или технология «прицепила» к нему дополнительный поток (альтернативные данные) гигабайтового размера. На самом деле в текущий момент потоки практически не используются, так что опасаться подобных ситуаций не следует, хотя гипотетически они возможны. Просто имейте в виду, что файл на NTFS – это более глубо-

кое и глобальное понятие, чем можно себе вообразить, просто просматривая каталоги диска. Более того, имя файла может содержать любые символы, включая полый набор национальных алфавитов, так как данные представлены в Unicode – 16-битном представлении, которое дает 65535 разных символов. Максимальная длина имени файла – 255 символов.

Каталоги (папки). Каталог на NTFS представляет собой специфический файл, хранящий ссылки на другие файлы и каталоги, создавая иерархическое строение данных на диске. Файл каталога поделен на блоки, каждый из которых содержит имя файла, базовые атрибуты и ссылку на элемент MFT, который уже предоставляет полную информацию об элементе каталога. Внутренняя структура каталога представляет собой бинарное дерево. Вот что это означает: для поиска файла с данным именем в линейном каталоге, таком, например, как у файловой системы FAT, операционной системе приходится просматривать все элементы каталога, пока она не найдет нужный. Бинарное же дерево рас-

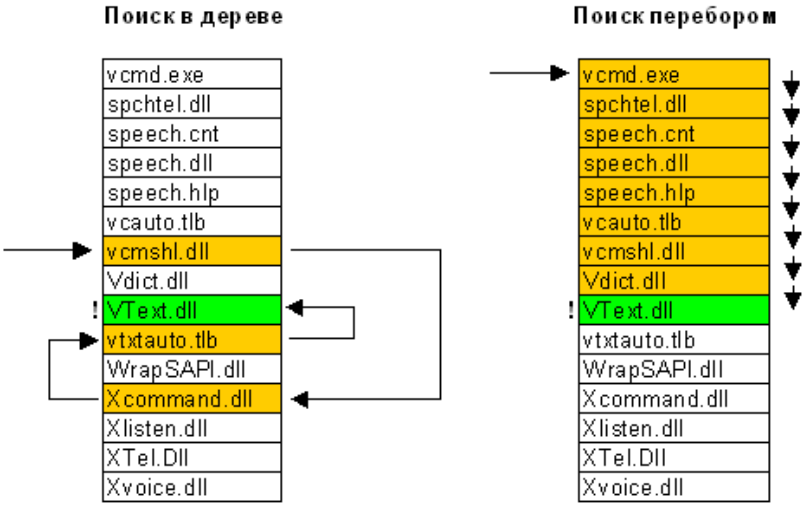


Рис. 32. Схема поиска в файловой системе NTFS

полагает имена файлов таким образом, чтобы поиск файла осуществлялся более быстрым способом – с помощью получения двухзначных ответов на вопросы о положении файла (рис. 32).

Вопрос, на который бинарное дерево способно дать ответ, таков: в какой группе относительно данного элемента находится искомое имя – выше или ниже? Поиск начинается с такого вопроса к среднему элементу, и каждый ответ сужает зону поиска в среднем в два раза. Файлы, скажем, просто отсортированы по алфавиту, и ответ на вопрос осуществляется очевидным способом – сравнением начальных букв. Область поиска, суженная в два раза, начинает исследоваться аналогичным образом, начиная опять же со среднего элемента.

Вывод: для поиска одного файла среди 1000, например, файловой системе FAT придется осуществить в среднем 500 сравнений (наиболее вероятно, что файл будет найден на середине поиска), а системе на основе дерева – всего около 12 ($2^{10} = 1024$). Экономия времени поиска налицо. Не стоит, однако, думать, что в традиционных системах (FAT) все так плохо: во-первых, поддержание списка файлов в виде бинарного дерева довольно трудоемко, а во-вторых – даже FAT в исполнении систем Windows 2000 и Windows 98 использовали сходную оптимизацию поиска. Необходимо развеять довольно распространенное заблуждение о том, что добавлять файл в каталог в виде дерева труднее, чем в линейный каталог: это сравнимые по времени операции. Дело в том, что для того, чтобы добавить файл в каталог, нужно сначала убедиться, что файла с таким именем там еще нет. И вот тут-то в линейной системе будут трудности с поиском файла, описанные выше, которые компенсируют саму простоту добавления файла в каталог.

Какую информацию можно получить, просто прочитав файл каталога? Только то, что выдает команда dir. Для выполнения простейшей навигации по диску не нужно обращаться в MFT за

каждым файлом, надо лишь читать самую общую информацию о файлах из файлов каталогов. Главный каталог диска – корневой – ничем не отличается от обычных каталогов, кроме специальной ссылки на него из начала метафайла MFT.

Журналирование. NTFS – отказоустойчивая система, которая вполне может привести себя в корректное состояние при практически любых реальных сбоях. Любая современная файловая система основана на таком понятии, как *транзакция* – действие, совершаемое целиком и корректно или не совершаемое вообще. У NTFS просто не бывает промежуточных (ошибочных или некорректных) состояний – квант изменения данных не может быть поделен на до и после сбоя, принося разрушения и путаницу – он либо совершен, либо отменен.

Пример 1. *Осуществляется запись данных на диск. Вдруг выясняется, что в то место, куда мы только что решили записать очередную порцию данных, писать не удалось – физическое повреждение поверхности. Поведение NTFS в этом случае довольно логично: транзакция записи отклоняется целиком – система осознает, что запись не произведена. Место помечается как сбойное, а данные записываются в другое место – начинается новая транзакция.*

Пример 2. *Более сложный случай – идет запись данных на диск. Вдруг возникает отключение питания, и система перезагружается. На какой фазе остановилась запись, где есть данные, а где ошибки записи? На помощь приходит другой механизм системы – журнал транзакций. Дело в том, что система, осознав свое желание писать на диск, пометила в метафайле $LogFile это свое состояние. При перезагрузке этот файл изучается на предмет наличия незавершенных транзакций, которые были прерваны аварией и результат которых непредсказуем, и все эти транзакции отменяются: место, в которое осу-*

ществлялась запись, помечается снова как свободное, индексы и элементы MFT приводятся в состояние, в котором они были до сбоя, и система, в целом, остается стабильна. А если ошибка произошла при записи в журнал? Тоже ничего страшного: транзакция либо еще и не начиналась (идет только попытка записать намерения ее произвести), либо уже закончилась – т. е. идет попытка записать, что транзакция на самом деле уже выполнена. В последнем случае при следующей загрузке система сама вполне разберется, что на самом деле все и так записано корректно, и не обратит внимания на «незаконченную» транзакцию.

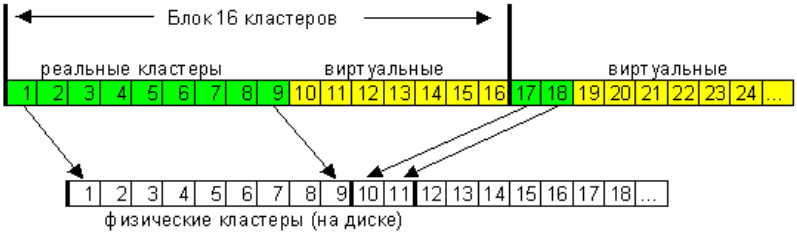
И все-таки помните, что журналирование – не абсолютная панацея, а лишь средство существенно сократить число ошибок и сбоев системы. Вряд ли рядовой пользователь NTFS хоть когда-нибудь заметит ошибку системы или вынужден будет запускать chkdsk. Опыт показывает, что NTFS восстанавливается в полностью корректное состояние даже при сбоях в очень загруженные дисковой активностью моменты. Пользователь может оптимизировать диск и в самый разгар этого процесса нажать reset – вероятность потерь данных даже в этом случае будет очень низка. Важно понимать, однако, что система восстановления NTFS гарантирует корректность файловой системы, а не пользовательских данных. Если производилась запись на диск и произошла «авария», то пользовательские данные могут и не записаться.

Сжатие. Файлы NTFS имеют один довольно полезный атрибут – «сжатый». Дело в том, что NTFS имеет встроенную поддержку сжатия дисков – то, для чего раньше приходилось использовать Stacker или DoubleSpace. Любой файл или каталог в индивидуальном порядке может храниться на диске в сжатом виде – этот процесс совершенно прозрачен для приложений. Сжатие файлов имеет очень высокую скорость и только одно

большое отрицательное свойство – огромная виртуальная фрагментация сжатых файлов, которая, правда, никому особо не мешает. Сжатие осуществляется блоками по 16 кластеров и использует так называемые «виртуальные кластеры» – опять же предельно гибкое решение, позволяющее добиться интересных эффектов – например, половина файла может быть сжата, а половина – нет. Это достигается благодаря тому, что хранение информации о компрессированности определенных фрагментов очень похоже на обычную фрагментацию файлов.

Например, типичная запись физической раскладки для реального, несжатого файла: кластеры файла с 1 по 43 хранятся в кластерах диска, начиная с 400. Кластеры файла с 44 по 52 хранятся в кластерах диска начиная с 8530 и т. д.

Тогда физическая раскладка типичного сжатого файла такова: кластеры файла с 1 по 9 хранятся в кластерах диска, начиная с 400. Кластеры файла с 10 по 16 нигде не хранятся. Кластеры файла с 17 по 18-й хранятся в кластерах диска начиная с 409. Кластеры файла с 19 по 36 нигде не хранятся.



Видно, что сжатый файл имеет «виртуальные» кластеры, реальной информации в которых нет. Как только система видит такие виртуальные кластеры, она тут же понимает, что данные предыдущего блока, кратного 16, должны быть разжаты, а получившиеся данные как раз заполнят виртуальные кластеры – вот, по сути, и весь алгоритм.

ГЛАВА 10
ПЕРВИЧНАЯ ИНИЦИАЛИЗАЦИЯ КОМПЬЮТЕРА И ЗАГРУЗКА ОПЕРАЦИОННОЙ СИСТЕМЫ

10.1. Первичная инициализация компьютера

Первичная инициализация компьютера является сложным процессом и происходит в несколько этапов. Пользователь нажимает кнопку POWER системного блока компьютера.

Первое устройство, которое запускается после нажатия кнопки включения компьютера, – блок питания. Если все питающие напряжения окажутся в наличии и будут соответствовать норме, на системную плату будет подан специальный сигнал *Power Good* (спустя 0,1–0,5 с), свидетельствующий об успешном тестировании блока питания и разрешающий запуск компонентов системной платы.

После этого на втором этапе чипсет (микросхема BIOS (*basic input/output system*) – базовая система ввода/вывода) автоматически подает сигнал сброса (*reset*) на специальный вход центрального процессора. Микропроцессор обнуляет содержимое своей микропроцессорной памяти (МПП) и запускается.

Системная память сконфигурирована так, что первая команда, которую считает процессор после сброса, будет находиться в микросхеме BIOS.

Таким образом, МП автоматически начинает выполнение команд, расположенных в ПЗУ – постоянной (или перезаписываемой) памяти (EEP ROM или Flash ROM), начиная с заданного адреса FFFF0h, принадлежащего пространству адресов BIOS. Данная команда просто передает управление программе инициализации BIOS и выполняет следующую команду (микропрограмму BIOS).

Эти микропрограммы не обладают всей функциональностью операционной системы, но ее (функциональности) достаточно для того, чтобы выполнить последовательную загрузку других программ, которые выполняются друг за другом до тех пор, пока последняя из них не загрузит операционную систему.

Итак, продолжает выполняться POST (*power-on self-test*). Осуществляется самотестирование при включении питания процессора и затем оперативной памяти. После чего возможно возникновение звуковых сигналов, автоматически генерируемых микросхемой BIOS и подаваемых встроенным динамиком в виде сочетаний коротких и длинных звуков, которые свидетельствуют о неисправностях процессора и оперативной памяти. И если возникают сигналы, свидетельствующие о неисправности, то они с некоторыми промежутками времени повторяются до тех пор, пока пользователь не выключит компьютер, запомнив эти сочетания сигналов и зная уже, таким образом, об обнаруженных неисправностях.

Если неисправности отсутствуют – процессор и оперативная память работоспособны, то реализуется следующий, третий этап инициализации компьютера – копирование микропрограмм BIOS в оперативную память и проверка контрольных сумм BIOS (CMOS) и состояния батарейки, первоначальная настройка чипсетов (набора микросхем, реализующих системную логику). Если контрольная сумма CMOS ошибочна, будут загружены значения по умолчанию из микросхемы BIOS.

Следующий, четвертый этап выполнения первичной инициализации компьютера – выполнение микропрограмм BIOS – на основе информации, хранимой в ПЗУ и содержащей данные о видеоадаптере, считывание с микросхемы BIOS уже видеокарты, запись в оперативную память ее драйвера, инициализация ее, осуществление ее тестирования.

Если определяется неисправность видеокарты, то подается соответствующая комбинация звуков, повторяющаяся во времени до тех пор, пока пользователь не запомнит звуковое предупреждение о неисправности и не выключит компьютер. Если видеокарта работоспособна, то в ходе инициализации видеоадаптера на экране появляется первое изображение, сформированное с помощью BIOS видеоадаптера.

Следующие многочисленные этапы выполнения первичной инициализации компьютера:

– расширенное тестирование процессора и оперативной памяти. Результаты тестирования выводятся на экран;

– подключение клавиатуры, мыши, тестирование портов ввода/вывода, других контроллеров и устройств;

– инициализация дисковых накопителей. Сведения об обнаруженных устройствах обычно выводятся на экран;

– распределение ресурсов между устройствами и вывод таблицы с обнаруженными устройствами и назначенными для них ресурсами на экран;

– поиск и инициализация иных устройств (так называемых *plug and play*), имеющих собственную микросхему BIOS.

В зависимости от конкретной версии BIOS порядок процедуры POST может немного различаться, но приведенные выше основные этапы выполняются при первичной инициализации любого компьютера.

В последующем программа POST завершается, передавая управление на вызов программного прерывания BIOS INT 19h, который ищет загрузочный сектор на устройствах, указанных в списке загрузки, для запуска загрузчика операционной системы (ОС).

Новые компьютеры используют вместо традиционного BIOS так называемые «прошивки» UEFI, содержащие програм-

мы первичной инициализации компьютера. И BIOS, и UEFI – это примеры программного обеспечения низкого уровня, запускающегося при старте компьютера перед тем, как загрузится операционная система.

Фактически BIOS был внедрен в 1980-х гг. на компьютерах тогда еще с операционной системой MS-DOS. Со временем BIOS менялся и улучшался. Разрабатывались его расширения, в частности ACPI – Advanced Configuration and Power Interface (усовершенствованный интерфейс управления конфигурацией и питанием). Это позволяло BIOS проще настраивать устройства и креативнее управлять питанием, например, уходить в спящий режим.

Но все же BIOS развивался не так стремительно, как другие компьютерные технологии со времен MS-DOS – эволюционировал он мало. У BIOS остались серьезные ограничения. Передача программного управления на загрузку ОС может осуществляться лишь для жестких дисков объемом не более 2,1 Тбайт. BIOS поддерживает 16-битный режим работы процессора, позволяющий совмещать и 16-битные, и 32-битные команды. BIOS доступен всего лишь 1 Мбайт оперативной памяти. У BIOS проблемы с одновременной инициализацией нескольких устройств, что ведет к замедлению процесса загрузки, во время которого инициализируются все аппаратные интерфейсы и устройства.

Давно пора было заменить BIOS. Еще в 1998 г. компания Intel начала работу над Extensible Firmware Interface (EFI). И в 2007 г. корпорации Intel, AMD, Microsoft и производители PC договорились о новой спецификации Unified Extensible Firmware Interface (UEFI) – унифицированный интерфейс расширяемой прошивки. Это уже индустриальный стандарт, который нашел практическую реализацию с появлением ОС Windows Vista Service Pack 1 и Windows 7.

Код микропрограммы UEFI пишется на языке программирования C и выполняется в современном 64-битном режиме работы процессора, использующем 32- или 64-битные команды. Адресное пространство UEFI больше, чем у BIOS. И в силу этих причин первичная инициализация проходит быстрее.

В настоящее время практически все изготавливаемые компьютеры используют UEFI вместо BIOS. Однако пока сохраняется обратная совместимость – большинство версий UEFI поддерживают эмуляцию BIOS, чтобы можно было устанавливать и работать с устаревшей ОС, ожидающей наличия BIOS.

UEFI активно развивается и дополняется другими функциями, в частности, реализацией поддержки графики, манипуляторов (мыши), сетевых адаптеров. Фактически UEFI превращается в небольшую операционную систему, способную на значительно большее, чем BIOS. По факту, UEFI имеет собственные драйверы, собственный сетевой стек, полный доступ ко всей памяти и устройствам, т. е. к оборудованию типа RAID/USB/SATA/Ethernet-контроллеры, управляемому драйверами.

Принципиально UEFI можно хранить во флеш-памяти на материнской плате либо загружать с жесткого диска или из сети. По причине возможности работы в сети и осуществления удаленной настройки и отладки разработчикам UEFI приходится встраивать безопасный запуск (*secure boot*), который реализует проверку загрузки ОС на отсутствие изменений ее вредоносными программами.

Secure boot – это специальный механизм защиты от запуска неподписанного кода (т. е. защищает от подмены загрузчика, а в случае с Microsoft – это еще и защита от нелицензионного использования ОС).

Можно отметить, что поспешность разработки UEFI проявляет отрицательные последствия. Так, в частности из-за «сырых»

прошивок – Secure boot, из-за традиционного подхода корпорации Microsoft к защите от нелицензионного использования ОС возможны проблемы дальнейшей загрузки ОС. Поэтому, в силу ряда причин, пользователи часто вынуждены его отключать (и даже иногда уже по умолчанию режим Secure boot в прошивке отключен). А поскольку UEFI, как операционная система, работает на неограниченном уровне доступа к ресурсам «ring 0», то существует возможность без включенного Secure Boot запуск процесса с администраторскими полномочиями, результатом которого будет удаление, модификация некоторых модулей и добавление вредоносных моделей в UEFI, что является крайне негативным.

10.2. Загрузка операционной системы

Независимо от аппаратуры и операционной системы, все компьютеры при загрузке используют или традиционный метод BIOS-MBR, или более современный UEFI-GPT.

Если материнская плата с прошивкой BIOS компьютера, то можно установить и загрузить операционную систему только для MBR-диска.

Если материнская плата с прошивкой UEFI компьютера поддерживает только режим загрузки Legacy boot, то можно загрузить операционную систему только с MBR-диска и нельзя установить операционную систему на GPT-диск.

Если материнская плата компьютера поддерживает загрузку исключительно в режиме UEFI boot, то можно загрузить операционную систему только с GPT-диска. В ином случае получим сообщение об ошибки.

Однако, если материнская плата с прошивкой UEFI компьютера поддерживает оба режима Legacy boot и UEFI boot, то необходимо будет активировать CSM (*Compatibility Support Module*) –

модуль поддержки совместимости. В таком случае можно загрузить операционную систему как с MBR-диска, так и GPT-диска.

Естественно, можно отдельно активировать режим загрузки UEFI boot для загрузки операционной системы с GPT-диска или отдельно активировать режим загрузки Legacy boot для загрузки операционной системы с MBR-диска.

Завершаясь, процесс первичной инициализации компьютера осуществляет программную передачу – старт процесса, который в итоге приведет к загрузке операционной системы в оперативную память. Первая команда зависит от того, какова структура разделов на жестком диске или на твердотельной памяти (flash-памяти).

Есть два вида структур разделов для этих носителей информации: MBR и GPT. Структура разделов на диске определяет: структуру данных на диске; код, который используется при загрузке, если раздел загрузочный; места, где начинаются и где заканчиваются разделы на диске.

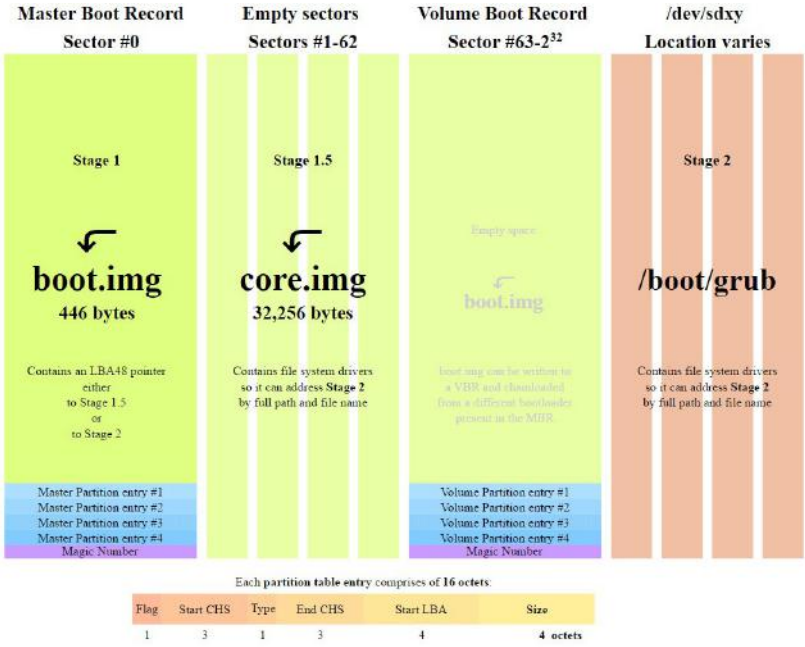
Процесс загрузки с MBR-диска. Базовая структура ввода-вывода включает в себя микропрограмму загрузчика, которая содержит код для загрузки начальной стадии загрузчика. По окончании первичной инициализации, используя BIOS Setup, а точнее системную конфигурацию, определяется загрузочное устройство. Как только BIOS определил загрузочное устройство, считывается первый дисковый сектор этого устройства в память. Первый сектор диска – это главная загрузочная запись (MBR) размером 512 байт. В этот размер помещаются три объекта: первая стадия загрузчика (446 байт); таблица разделов диска (16 байт на раздел × 4 раздела, поскольку MBR поддерживает только четыре раздела); подпись (2 байта).

На этом этапе в MBR сканируется таблица разделов и загружается в оперативную память загрузочный сектор – Volume Boot

Record (VBR). VBR обычно содержит начальный загрузчик программ – Initial Program Loader (IPL) и именно этот код инициирует процесс загрузки. Начальный загрузчик программ включает в себя вторую стадию загрузчика, который затем загружает операционную систему.

В частности, на системах семейства Windows NT, таких как Windows XP, начальный загрузчик программ сначала загружает другую программу под названием NT Loader (аббревиатура NTLDR), которая затем загружает операционную систему.

Для операционных систем на ядре Linux используется загрузчик GRUB (*GRand unified bootloader*). Процесс загрузки похож на описанный выше, единственная разница – в наименовании загрузчиков на первой и второй стадии. В GRUB первая стадия загрузчика называется GRUB Stage 1. Она за-



гружает вторую стадию, известную как GRUB Stage 2. Вторая стадия получает и загружает список операционных систем на жестких дисках, предоставляя пользователю список ОС для загрузки.

На следующей схеме демонстрируются для метода BIOS-MBR этапы работы загрузчика операционной системы.

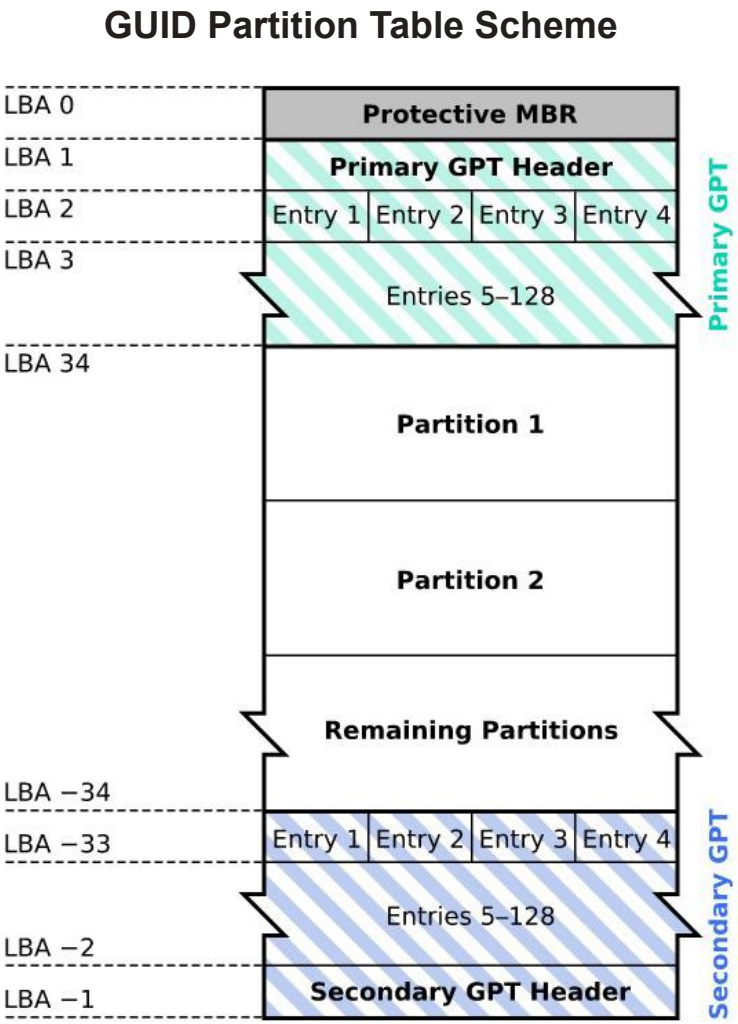
Процесс загрузки с GPT-диска. По окончании первичной инициализации, используя прошивку UEFI, считывается GPT (GUID Partition Table – таблица разделов GUID). GPT располагается в первых секторах диска, сразу после сектора 0, где по-прежнему хранится главная загрузочная запись для Legacy BIOS. На самом деле GPT определяет таблицу разделов на диске, на которой загрузчик UEFI распознает системный раздел EFI. Системный раздел содержит загрузчики для всех операционных систем, установленных на других разделах жесткого диска. Загрузчик инициализирует менеджер загрузки операционных систем, который затем загружает операционную систему, указанную пользователем в списке всех выявленных.

Для операционных систем семейства Linux существует версия загрузчика GRUB с поддержкой UEFI, которая загружает файл, такой как grub.efi, или загрузчик EFI, который загружает свой файл, такой как elilo.efi.

Следует заметить, что методы и UEFI-GPT, и BIOS-MBR передают управление загрузчику, но сами напрямую не грузят операционную систему. Однако в UEFI не требуется проходить через несколько стадий загрузчика, как в BIOS. Процесс загрузки происходит на самой ранней стадии, в зависимости от аппаратной конфигурации компьютера. В структуре разделов GPT происходит следующее. GPT использует UEFI, в котором нет такой, как у MBR процедуры хранения в загрузочном секторе первой стадии загрузчика с последующим вызовом второй стадии за-

грузчика. UEFI в отличии от BIOS может анализировать файловую систему и даже сам загружать файлы.

На следующей схеме демонстрируются таблицы разделов (Partition Table) GUID – глобального уникального идентификатора.



«Классическая» загрузка операционной системы. В конце завершения программы POST осуществляется программная передача управления на программу «Загрузчик ОС» (*boot loader*). При этом реализуются поиск и выполнение начального кода загрузки. В зависимости от установок BIOS осуществляется попытка доступа (в предварительно определенном порядке, который можно изменить) к первому сектору гибкого магнитного диска, заданного жесткого магнитного диска или оптического компакт-диска. На жестком диске первый сектор называют главной загрузочной записью (MBR).

Когда устройство найдено, BIOS копирует содержание его первого сектора в оперативную память (начиная из фиксированного физического адреса *0x00007c00*), выполняется команда перехода по этому адресу и начинает выполняться только что загруженный код. За все остальное отвечает операционная система.

Рассмотрим основные принципы работы самого простого, «классического», загрузчика в архитектуре PC.

Как упоминалось ранее, MBR содержит таблицу разделов и небольшую программу, которая загружает первый (загрузочный) сектор одного из разделов в память и начинает выполнять код, который находится в нем, – обычно этот код называют кодом загрузчика ОС. Код загрузочного сектора зависит от того, какая файловая система установлена для этого раздела: для FAT выполняют один вариант, для NTFS – другой. Однако код самого простого загрузчика все-таки сводится к поиску на диске ядра ОС (обычно это файл, который находится на фиксированном месте корневого каталога).

Таким образом, загрузчик ОС является программой, которая формирует (загружает) образ ядра операционной системы в оперативной памяти.

На самом деле выбор раздела, из которого нужно загрузить сектор, преимущественно осуществляют с помощью признака – флажка активного раздела, заданного для ячейки таблицы разделов. Таким способом можно загрузить только ту ОС, ядро которой находится на активном разделе.

Следует отметить, что использование загрузочного сектора для непосредственной загрузки ядра ОС имеет свои особенности:

– код загрузчика вынужденно является очень простым, а потому в нем нет возможности выполнять более сложные действия, например, руководить загрузкой нескольких ОС, а большинство других недостатков являются последствиями этого;

– нет возможности передавать параметры в загрузчик;

– процесс ограничен описанной схемой переключения из MBR на загрузочный сектор и при этом нет возможности руководить этим процессом;

– нет возможности загружать ядро из другого раздела диска или из подкаталога;

– ОС всегда запускается только в реальном режиме работы процессора.

Для того, чтобы решить эти проблемы, код загрузчика ОС должен быть усложнен. Естественно, что усложненный код не поместится в один сектор, потому был предложен подход, который получил название схемы двухэтапной загрузки.

Таким образом, загрузчик разбили на две части: загрузчик первого и второго этапов. Первый хранится в загрузочном секторе диска (но может также сохраняться и в MBR) и его основное задание – это поиск на диске и загрузка в память не ядра, а загрузчика второго этапа.

Загрузчик второго этапа – это программа, которая получает контроль над компьютером после выполнения начальной загруз-

ки. И только теперь она уже может быть выполнена или в реальном режиме работы процессора, или, переключившись, в привилегированном режиме работы процессора. В сущности, такой загрузчик является мини-ОС специализированного назначения.

Рассмотрим возможности двухэтапного загрузчика. Только в этом загрузчике возможно:

Руководить загрузкам нескольких операционных систем. Особенно удобно это делать в загрузчиках, которые принимают управление от MBR. При этом загрузчик берет на себя поиск активного раздела и загрузку системы из него. Конфигурацию такого загрузчика можно динамически изменять во время изменения разделов диска. Загрузчик может содержать код доступа к разным файловым системам, код загрузки разных ядер и тому подобное.

Предоставлять интерфейс пользователя, который обычно сводится к отображению меню выбора загружаемой операционной системы, а также осуществлять передачу введенных пользователем параметров в ядро перед его загрузкой.

Задавать пользователем его конфигурацию из загруженной ОС. Для этого, например, можно задать текстовый конфигурационный файл, сохранить его на диске и запустить специальную утилиту, которая сделает синтаксический разбор файла, превратит его во внутреннее отображение и сохранит на диске в фиксированном месте, известном загрузчику.

Загрузчик может работать со всеми дисками компьютера, загружать ядра, которые находятся в разных местах на диске (в частности, внутри иерархии каталогов), не ограничиваясь одним разделом и одним диском.

В свое время двухэтапные загрузчики были чрезвычайно распространены. Их поставляли вместе с ОС (например, lilo или GRUB для Linux; загрузчик Windows XP), а также они были ре-

ализованы как отдельные продукты – менеджеры загрузки (boot managers).

Пример загрузки ОС Windows XP. Итак, после того, как определено место нахождения файла ntldr в корневом каталоге этого раздела, осуществляется его загрузка в оперативную память и передача управления на его точку входа.

Файл ntldr можно рассматривать как загрузчик второго этапа. Он начинает свое выполнение в 16-битном режиме процессора. Прежде всего, переводит процессор в защищенный режим и включает поддержку страничной организации оперативной памяти. После этого считывает из корневого каталога файл boot.ini и делает его синтаксический разбор.

Вот фрагмент файла boot.ini:

```
[boot loader]
timeout=30
default=multi(0) disk(0) rdisk(0) partition(1)\WINDOWS
[operating systems]
multi (0) disk(0) rdisk(0) partition(1)\WINDOWS=Windows XP"
C:\="windows 98"
```

После дескриптора [boot loader] задается вариант загрузки по умолчанию и время, по завершении которого система автоматически будет загружаться в соответствии с этим вариантом.

После дескриптора [operating systems] определяется список возможных вариантов загрузки. Для каждого варианта может быть задан один из нескольких адресов загрузки:

– раздел с корневым каталогом WINDOWS (для загрузки Windows XP);

– литерное обозначение тома, на котором находится другая ОС;

– имя файла с указанием тома, на котором он находится.

В случае указания литерного имени раздела (как в примере) ntldr находит на диске файл bootsec.dos. В этом файле после уста-

новки (развертывания») операционной системы Windows XP хранят загрузочный сектор (DOS или Consumer Windows). Выполнение ntldr переключает процессор в реальный режим и начинает выполнять код этого загрузочного сектора, хранимого теперь после установки ОС в файле bootsec.dos.

Если задано имя файла, то ntldr будет загружать файл с таким именем. И, следовательно, если в файле сохранить загрузочный сектор другой ОС, например, Linux, то ntldr сможет загрузить и его.

Для реализации этого примера вариант загрузки будет иметь такой вид:

C: bootsec. lnx="linux"

Можно отметить также, что раздел с установкой Windows XP в boot.ini не обязан совпадать с разделом, из которого происходит загрузка, – таких разделов может быть несколько.

Когда есть один вариант загрузки, система сразу начинает загружаться, когда их больше – отображает меню загрузки. После выбора варианта из меню ntldr запускает программу ntdetect. com, что в реальном режиме определяет базовую конфигурацию компьютера (подобно тому, как это делает функция setup() для Linux – ни одна из систем не доверяет этот код BIOS). Собранную информацию хранят в системе. И позже она будет сохранена в реестре. Внизу экрана появляется текстовый индикатор процесса. В этой ситуации можно нажать на функциональную клавишу F8 и перейти в меню дополнительных возможностей загрузки (в частности, загрузка безопасного режима, загрузка операционной системы без установки сетевых драйверов и тому подобное).

Далее ntldr загружает в оперативную память исполнимый файл ntoskrnl.exe, который фактически реализует функции ядра и исполнительной подсистемы Windows XP; файл-оверлей bootvid.dll, подгружающий видеодрайвер по умолчанию, кото-

рый отвечает за отображение информации во время загрузки; файл-оверлей hat.dll, реализующий уровень абстрагирования от оборудования, а также основные файлы реестра. Затем ntldr определяет по данным реестра, какие драйверы установлены в режиме запуска во время загрузки (например, драйвер жесткого диска), и загружает их (без инициализации). При этом будет загружен также и драйвер корневой файловой системы.

На этом ntldr завершает свою роль в загрузке и вызывает главную функцию, реализуемую исполнимым файлом ntoskrnl.exe для продолжения загрузки.

Инициализация ntoskrnl.exe состоит из двух этапов: фаз 0 и 1. Множество подсистем исполнительной системы принимают параметр, который показывает, в какой фазе инициализации в данный момент находится система. Во время выполнения фазы 0 прерывания запрещены, на экране ничего не отображается. Основной целью этого этапа является подготовка начальных структур данных, необходимых для расширенной инициализации во время выполнения фазы 1. Отметим, что менеджер процессов на этом этапе будет инициализироваться почти полностью, с его помощью создают начальный объект-процесс с названием Idle, процесс System и системный поток для выполнения инициализации фазы 1. После завершения фазы 0 прерывания разрешены и начинает выполняться системный поток. Во время выполнения фазы 1 управление экраном осуществляет видеодрайвер bootvid.dll, что отображает загрузочный экран и графический индикатор процесса на нем (этот индикатор будет изменяться на протяжении всей фазы 1). Происходит окончательная инициализация разных подсистем исполнительной системы (менеджера объектов, планировщика, службы безопасности, менеджера виртуальной памяти, менеджера кэша и тому подобное). Во время инициализации подсистемы ввода-вывода (которая занимает до 50 % вре-

мени этой фазы) происходит подготовка необходимых структур данных, инициализация драйверов с запуском во время загрузки (boot-start), загрузка и инициализация драйверов с системным запуском (system-start). Фаза 1 завершается запуском менеджера сессий (smss.exe).

Последующую загрузку выполняют три системных процесса: менеджер сессий smss.exe, процесс регистрации в системе winlogon.exe и менеджер управления сервисами (SCM, services.exe). Основным заданием менеджера сессий является загрузка и инициализация всех компонентов подсистемы Win32 (как режиму пользователя, так и режиму ядра), а также окончательная инициализация реестра и запуск winlogon.exe.

Процесс регистрации в системе запускает менеджер управления сервисами и менеджер аутентификации, а также организует регистрацию пользователей в системе.

Менеджер сервисов (SCM) загружает и инициализирует сервисы режима пользователя, установленные в режиме автоматической загрузки, – в частности, интегрированную графическую среду. Этот процесс автоматической загрузки других сервисов может длиться уже после начала интерактивной работы пользователя. После инициализации всех сервисов загрузка операционной системы считается успешной.

Особенности загрузчика Linux. Во время загрузки операционной системы Linux используют двухэтапный загрузчик. Есть несколько программных продуктов, которые реализуют такие загрузчики. Самый известный из них lilo (от linux loader). Он может быть установлен как в MBR, заменив в записи код, который загружает первый сектор активного раздела, так и в загрузочном секторе обычно активного раздела диска. Второй подход является более безопасным при условии нескольких ОС, установленных на компьютере и выбираемых пользователем в режиме

мультизагрузки, поскольку некоторые ОС могут перезаписывать MBR по своей инициативе.

Первая часть lilo, записанная в загрузочный сектор или MBR, во время своего выполнения готовит оперативную память и загружает в нее вторую часть. Она считывает с диска двоичное отображение карты имеющихся на компьютере вариантов загрузки (разные ОС, разные установки Linux и тому подобное) и предлагает пользователю выбрать один из них (с помощью подсказки «LILO boot.:»). Отметим, что исходную версию карты вариантов загрузки создает пользователь с правами системного администратора в виде обычного текстового файла /etc/lilo.conf. После каждого изменения карты необходимо обновлять ее отображение на диске, используемое загрузчиком. Для этого выполняют команду (с правами root): # lilo.

После выбора пользователем одного из вариантов загрузки поведение загрузчика зависит от характера файловой системы раздела. В случае выбора раздела с другой ОС (например, Windows) считывается в оперативную память и выполняется загрузочный сектор этого раздела. Именно поэтому с помощью lilo можно загрузить любую ОС. Если же выбран раздел из Linux, в оперативную память загружают ядро системы, адрес которого на диске содержится в карте вариантов загрузки. После загрузки в оперативную память появляется сжатое ядро системы и пригодный для выполнения код двух функций загрузки: setup() и startup_32(). Код загрузчика переходит к выполнению функции setup(), начиная инициализацию ядра.

Первый этап инициализации ядра происходит в реальном режиме процессора. Преимущественно осуществляется инициализация аппаратных устройств (Linux не доверяет этого BIOS). Функция setup() определяет физический объем оперативной памяти в системе, инициализирует клавиатуру, видеокарту, кон-

троллер жесткого диска, некоторые другие устройства, переназначает таблицу прерываний, переводит процессор в защищенный режим и передает управление функции startup_32(), код которой также находится вне сжатого ядра.

Функция startup_32() задает сегментные регистры и стек, распаковывает образ ядра и располагает его в оперативной памяти. Дальше выполняется код распакованного ядра. Он в свою очередь формирует среду выполнения для первого потока ядра Linux («процесс 0»); создает его стек (с этого момента считается, что он есть); включает поддержку страничной организации оперативной памяти, задает начальные (пустые) обработчики прерываний, определяет модель процессора и переходит к выполнению функции start_kernel().

Функция start_kernel() выполняется в пределах потока ядра «процесс 0», завершая инициализацию ядра. Она доводит до рабочего состояния практически каждый компонент ядра, и в частности:

– инициализирует таблицы страниц и все дескрипторы страниц;

– окончательно инициализирует таблицу прерываний;

– инициализирует распределение оперативной памяти по сегментам;

– устанавливает системные дату и время;

– выполняет код инициализации драйверов устройств;

– делает доступной корневую файловую систему (где расположены файлы, необходимые для загрузки системы).

Кроме того, создается поток инициализации («процесс 1») с помощью функции kernel_thread(), которая выполняет код функции init(). Этот поток создает другие потоки ядра и выполняет программу /shin/init, превращаясь в первый в системе процесс пользователя init. Отметим, что для корректной загрузки init

должна быть доступной корневая файловая система с важнейшими библиотеками (каталог /lib должен быть на том же разделе, что и корневой каталог /).

После превращения этого потока в init инициализация ядра считается завершенной, функция start_kernel() переходит в бесконечный цикл простоя (*idle loop*), не занимая ресурсов процессора. Последующая инициализация системы происходит во время выполнения init.

ГЛАВА 11
СТРУКТУРА КОМПЬЮТЕРА. ПЕРИФЕРИЙНЫЕ УСТРОЙСТВА

11.1. Элементы конструкции компьютера

Конструктивно компьютер выполнен в виде центрального системного блока, к которому через разъемы подключаются внешние устройства: дополнительные устройства памяти, клавиатура, дисплей, принтер, сканер, модем и др.

Системный блок содержит те блоки, которые производят вычисления и обеспечивают связь с периферийными устройствами.

На заре эры вычислительной техники, то, что находится теперь в корпусе размерами 50×40×20 см, занимало несколько больших комнат, если вспомнить, например, вычислительную машину БЭСМ-6 или ЕС-1060.

Монитор (дисплей) обеспечивает вывод графической информации в том виде, какой требуется для удобного восприятия человеком. От качества изображения на экране монитора в первую очередь зависит, насколько комфортно и безопасно человеку общаться с компьютером.

Клавиатура – устройство для ручного ввода числовой, текстовой и управляющей информации в компьютер.

Эти элементы составляют минимальную (и достаточную) конфигурацию компьютера.

Последующие элементы, которые будут приведены, обеспечивают удобство работы с компьютером и увеличение возможностей компьютера.

Манипуляторы (устройства указания): *джойстик-рычаг*, *мышь*, *трекбол* – шар в оправе, *световое перо* и др. служат для

ввода графической информации на экран дисплея путем управления движением курсора по экрану с последующим кодированием координат курсора и вводом их в компьютер.

Мышь, например, позволяет человеку выбрать из набора одну из управляющих функций: открыть или закрыть приложение, изменить геометрию окна приложения, сохранить или удалить файл и т. д.

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

К средствам мультимедиа относятся:

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

– *принтеры* – печатающие устройства для регистрации текстовой и графической информации, в том числе и цветной, на бумажный носитель или пленку;

– *графопостроители (плоттеры)* – для вывода цветной или черно-белой графической информации (графиков, чертежей, рисунков) из компьютера на бумажный носитель;

– высококачественные акустические и видеовоспроизводящие системы с усилителями и звуковыми колонками, большими видеоэкранами;

– высококачественные видео- и звуковые платы, платы видеозахвата, снимающие изображение с видеомагнитофона или видеокамеры и вводящие его в компьютер;

– активно развивающиеся устройства речевого ввода и вывода информации.

Дополнительными устройствами являются *устройства связи и телекоммуникации*. Они используются для связи с приборами и другими средствами автоматизации (согласователи интерфейсов, адаптеры, цифро-аналоговые и аналого-цифровые преобразователи и т. п.) и для подключения компьютера к каналам связи, к другим компьютерам и разнообразным сетям (сетевые интерфейсные платы, «стыки», мультиплексоры передачи данных, модемы).

11.2. Периферийные устройства персонального компьютера

Периферийным устройством называется аппаратное обеспечение компьютера, которое конструктивно отделено от самого компьютера, но, в то же время, функционирует под его управлением. Предназначаются такие устройства для расширения функциональности персонального компьютера: организации ввода/вывода, распечатки набранных текстов и изображений, обмена данными с другими персональными компьютерами, для управления разнообразными внешними устройствами, включая бытовую технику, и т. д.

Таким образом, периферийные устройства персонального компьютера подключаются к компьютеру с помощью специальных разъемов.

Можно подчеркнуть, что в настоящее время активно развиваются беспроводные периферийные устройства, в частности: мышки, клавиатуры, принтеры и др.

Ниже будут приведены примеры основных устройств, которые можно часто встретить на компьютерах.

Монитор, дисплей, экран. Мониторы по технологии работы делятся на ЭЛТ (электронно-лучевая трубка) и ЖК (жидкокристаллический). Первый вид – это устройство, содержащее в себе кинескоп, такой же, как в старых телевизорах. Использование такого монитора довольно вредно для здоровья пользователя, к счастью, сегодня они мало применяются. Второй вид – жидкокристаллический монитор – это современное решение, его использование гораздо менее вредно для здоровья.

Второй важной характеристикой является размер экрана в мониторе. Его принято измерять по диагонали и указывать в дюймах. Жидкокристаллические мониторы бывают широкоформатными, это значит, что экран будет слегка вытянут по ширине, соотношение сторон такого экрана обычно 16:9 (у обычного квадратного 4:3).

Мониторы можно подключать через следующие интерфейсы VGA, DVI, HDMI и DisplayPort. В данное время на персональных компьютерах широко используются VGA и DVI интерфейсы, также существуют различные переходники, если в мониторе или в материнской плате не предусмотрены данные интерфейсы.

Клавиатура. Это устройство для ввода информации. Все клавиши разделены на несколько групп: буквенно-цифровые; управляющие (клавиши Enter, Backspace, Shift, Ctrl, Alt, Win,

Caps Lock, Tab, Print Screen, Scroll Lock, Pause Break, Num Lock); функциональные (клавиши F1, F2, …, F12); клавиши управления курсором (Стрелки, Insert, Delete, Home, End, Page Up, Page Down); малая цифровая клавиатура.

Кроме перечисленных выше, на клавиатуре может находиться набор мультимедийных клавиш самого разного назначения. Также обычно имеются индикаторы режима Num Lock, Caps Lock, Scroll Lock.

Устройство может подключаться по интерфейсу PS/2, USB. Существуют также переходники, которые позволяют подключить USB клавиатуру в порт PS/2 и наоборот.

На ноутбуках и нетбуках в целях экономии места могут отсутствовать некоторые группы клавиш. Также могут отсутствовать они и в обычных клавиатурах.

Плюсом USB клавиатуры является то, что ее можно подключать к включенному компьютеру, и через некоторое время операционная система автоматически опознает клавиатуру, тем самым вам не надо перезагружать компьютер, чтобы начать работать с ней. Если подключить клавиатуру PS/2 к включенному компьютеру, то система не сможет определить устройство и придется перезагрузить компьютер, чтобы начать использовать клавиатуру.

Мышь. Это устройство-манипулятор, которое преобразует движения руки пользователя в движения курсора на экране. Минимальный набор – это две клавиши и колесико прокрутки, некоторые модели могут иметь расширенный набор: более одного колесика и дополнительные клавиши по левой и правой стороне мышки.

Кнопки мыши обычно принято называть «левая кнопка мыши» и «правая кнопка мыши», под колесиком обычно тоже имеется третья дополнительная кнопка.

По принципу работы мышки бывают механическими, оптическими и лазерными. Механические содержат внутри прорезиненный шар, который при движении вращает маленькие валы, с которых и считывается информация о направлении и скорости движения манипулятора (устаревшая модель). Оптические мышки имеют направленный вниз светодиод. Отраженный от поверхности свет и дает возможность узнать направление и скорость перемещения. Лазерные мышки являются разновидностью оптических.

Разница состоит в том, что светодиод заменен миниатюрным лазером. Это позволило избавиться от свечения мыши и увеличило точность позиционирования. Механические мыши устарели и почти не используются, обычно применяются разновидности оптических манипуляторов.

Способы подключения мыши такие же, как и у клавиатуры: USB и PS/2. Как и с клавиатурами, USB-мышки определяются включенным компьютером.

Принтер. Это устройство для вывода (печати) информации на бумагу.

В первую очередь они различаются по технологии печати. Бывают лазерные (светодиодные принтеры), струйные, матричные и другие принтеры (твердочернильные, сублимационные).

Лазерные принтеры – наиболее практичные для работы устройства. У них наибольшая скорость печати, ресурс картрид-

жа и наименьшая стоимость обслуживания и заправки. Обычно они бывают черно-белые, хотя существуют и цветные. Для печати используется специальный порошок – тонер. Он наносится на лист бумаги в нужных местах, а затем закрепляется на ней путем нагрева и расплавления. Также есть светодиодный принтер, который является параллельной веткой развития технологии лазерной печати.

Струйный принтер – самый подходящий вариант для печати цветных изображений, в том числе фотографий. В качестве печатающего вещества используется жидкая краска четырех или шести цветов. Смешение этих красок в разных сочетаниях дает всю палитру при печати. Недостатком является опасность засыхания краски в картридже в случае длительного простоя и невысокая скорость печати. Однако такие принтеры дают наибольшее качество цветной печати, а также невысокую стоимость заправки при условии использования СНПЧ – системы непрерывной подачи чернил. Это система, при которой емкости с краской находятся рядом с принтером и подача в картриджи осуществляется по специальным трубкам.

Матричный принтер. Это наиболее старый и наименее удобный вариант. В нем для печати используется лента, пропитанная красящим веществом. Лента прижимается к бумаге специальными уголками в нужных местах и формирует из точек изображение. Главными минусами таких принтеров является: низкая ско-

рость печати, качество и повышенный шум при печати. Однако они по-прежнему используются во многих организациях потому, что некоторые старые программные продукты могут печатать только на таких принтерах.

Принтеры подключаются к компьютеру через интерфейс USB или LPT (старые модели).

Плоттер. Это устройство практически распространено у пользователей компьютеров, которые постоянно работают с чертежами или рисунками больших форматов. Подобно принтеру, плоттер способен выводить на бумагу выполненные на компьютере изображения – чертежи и рисунки очень больших форматов (вплоть до формата A0). Они незаменимы при печати больших, демонстрационных диаграмм, схем и рисунков для различных презентаций.

Первые модели плоттера отличались от принтера тем, что они не смешивали цвета – конструктивно они чертили заправленными каждый своим цветом «рапидографами». Можно было подбирать толщину линий, но вот цвет всегда оставался одним из установленного в плоттере набора. Сегодня, последние модели полностью лишены этого недостатка.

Сканер. Устройство для передачи информации с бумажного носителя в компьютер. Отсканировав изображение, мы получим картинку. В случае если сканируется текст и его нужно отредактировать, применяются специальные программы для распознавания текста. Одна из популярных программ, которая распознает текст со сканированного документа ABBYY FineReader, которая распространяется как платный программный продукт.

Сканеры подключаются через USB.

МФУ. Аббревиатура расшифровывается как многофункциональное устройство. Это очень практичное решение, представляющее собой комплектацию принтера и сканера. Современные МФУ дают возможность делать копии без включения компьютера (копировальный аппарат), а также могут выполнять функции факса.

Подключаются через USB и Ethernet (по сети).

Акустические колонки. Это устройства для воспроизведения звука.

Отличаются колонки, в первую очередь, мощностью. Подключать их необходимо в двух местах: к источнику сигнала – зеленый круглый разъем на материнской плате или дискретной звуковой карте; а также к источнику питания, чаще в обычную розетку, но бывают версии, питающиеся от USB.

К числу периферийных устройств можно отнести множество других девайсов: это и источники бесперебойного питания, и веб-камеры, и внешние модемы, и еще множество других полезных приспособлений, фотографии некоторых из которых представлены ниже.


Logitech


USB
On/Off

ГЛАВА 12
ОСНОВЫ РАБОТЫ УСТРОЙСТВ ВВОДА/ВЫВОДА

12.1. Логические принципы организации ввода/вывода для устройств

Структура системы ввода/вывода. Если поручить неподготовленному пользователю сконструировать систему ввода/вывода, способную работать со всем множеством внешних устройств, то, скорее всего, он окажется в ситуации, в которой находились биологи и зоологи до появления трудов Линнея. Все устройства разные, отличаются по выполняемым функциям и своим характеристикам, и кажется, что принципиально невозможно создать систему, которая без больших постоянных переделок позволяла бы охватывать все многообразие видов. Вот перечень лишь нескольких направлений (далеко не полный), по которым различаются устройства.

Скорость обмена информацией может варьироваться в диапазоне от нескольких байт в секунду (клавиатура) до нескольких гигабайт в секунду (сетевые карты).

Одни устройства могут использоваться несколькими процессами параллельно (являются разделяемыми), в то время как другие требуют монопольного захвата процессом.

Устройства могут запоминать выведенную информацию для ее последующего ввода или не обладать этой функцией. Устройства, запоминающие информацию, в свою очередь, могут дифференцироваться по формам доступа к сохраненной информации: обеспечивать к ней последовательный доступ в жестко заданном порядке или уметь находить и передавать только необходимую порцию данных.

Часть устройств умеет передавать данные только по одному байту последовательно (символьные устройства), а часть

устройств умеет передавать блок байт как единое целое (блочные устройства).

Существуют устройства, предназначенные только для ввода информации, устройства, предназначенные только для вывода информации, и устройства, которые могут выполнять и ввод, и вывод.

В области технического обеспечения удалось выделить несколько основных принципов взаимодействия внешних устройств с вычислительной системой, т. е. создать единый интерфейс для их подключения, возложив все специфические действия на контроллеры самих устройств. Тем самым конструкторы вычислительных систем переложили все хлопоты, связанные с подключением внешней аппаратуры, на разработчиков самой аппаратуры, заставляя их придерживаться определенного стандарта.

Похожий подход оказался продуктивным и в области программного подключения устройств ввода/вывода. Подобно тому, как Линнею удалось заложить основы систематизации знаний о растительном и животном мире, разделив все живое в природе на относительно небольшое число классов и отрядов, мы можем разделить устройства на относительно небольшое число типов, отличающихся по набору операций, которые могут быть ими выполнены, считая все остальные различия несущественными. Мы можем затем специфицировать интерфейсы между ядром операционной системы, осуществляющим некоторую общую политику ввода/вывода, и программными частями, непосредственно управляющими устройствами, для каждого из таких типов. Более того, разработчики операционных систем получают возможность освободиться от написания и тестирования этих специфических программных частей, получивших название драйверов, передав эту деятельность производителям

самих внешних устройств. Фактически мы приходим к использованию принципа уровневого или слоеного построения системы управления вводом-выводом для операционной системы (см. рис. 33).

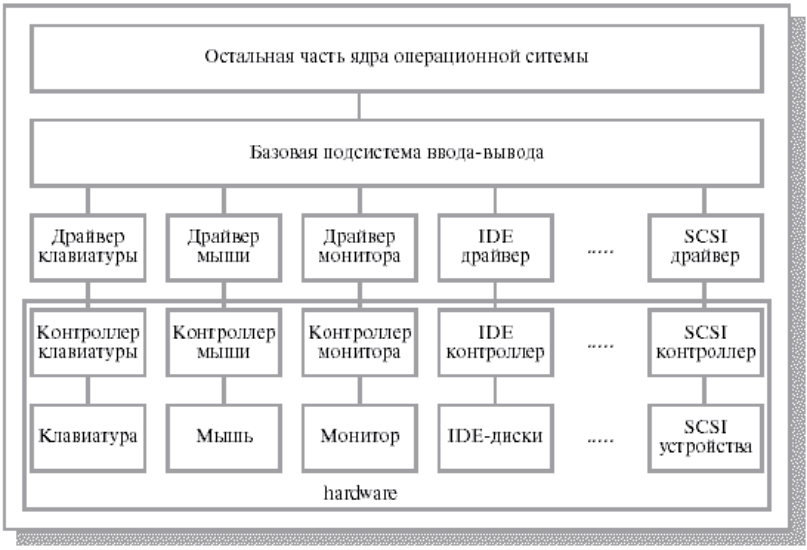


Рис. 33. Структура системы ввода/вывода

Два нижних уровня этой слоеной системы составляет hardware: сами устройства, непосредственно выполняющие операции, и их контроллеры, служащие для организации совместной работы устройств и остальной вычислительной системы. Следующий уровень составляют драйверы устройств ввода/вывода, скрывающие от разработчиков операционных систем особенности функционирования конкретных приборов и обеспечивающие четко определенный интерфейс между hardware и вышележащим уровнем – уровнем базовой подсистемы ввода/вывода, которая, в свою очередь, предоставляет механизм взаимодействия между драйверами и программной частью вычислительной системы в целом.

12.2. Шинная организация подсистемы ввода/вывода

Общая организация шин. Особенность данной организации ввода/вывода заключается в использовании общей информационной магистрали для передачи данных между процессором, памятью и контроллерами (устройствами) ввода/вывода (УВВ), подключенными к шине. Заметим, что термин «шина» применительно к системе ввода/вывода используется для обозначения информационных магистралей как со стороны «большого», так и со стороны «малого» интерфейса. Если речь идет о шинной архитектуре системы ввода/вывода, очевидно, имеется в виду «большой» интерфейс, организованный в виде общей шины.

Как правило, шины состоят только из пассивных элементов, и все управление передачами выполняется передающим и принимающим устройствами. Так как между разными устройствами возникает «состязание» за пользование общим ресурсом, то необходимо управление предоставлением шины передающему устройству.

Общая схема организации подсистемы ввода/вывода представлена на рисунке 34.

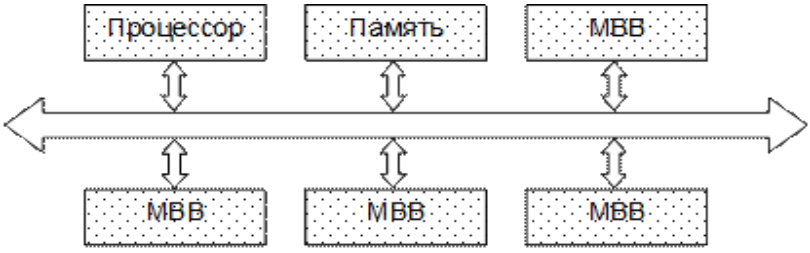


Рис. 34. Шинная организация подсистемы ввода/вывода

С целью снижения стоимости некоторые ЭВМ имеют единственную шину для памяти и устройств ввода/вывода. Такая шина называется *системной*. Примерами системных шин явля-

ются шины стандартов *ISA*, *EISA* или *MCA* персональных компьютеров недалекого прошлого. Необходимость сохранения баланса производительности по мере роста быстродействия процессоров привела к двухуровневой организации шин в персональных компьютерах на основе локальной шины. *Локальной шиной* называется шина, электрически выходящая непосредственно на контакты процессора. Она обычно объединяет процессор, память, схемы буферизации для системной шины и ее контроллер, а также некоторые вспомогательные схемы. Типичными примерами локальных шин являются *VL-Bus* и *PCI*.

Устройство, желающее передать данные, должно сначала получить доступ к шине, затем установить связь с адресатом и определить его способность к взаимодействию. После этого передающее устройство извещает принимающее о действиях, которые должно совершить принимающее устройство в ходе взаимодействия. Принимающее устройство распознает свой адрес на шине и отвечает на управляющие сигналы от передающего устройства. Только после этого происходит собственно передача данных.

Передача элемента данных по шине от одного МВВ к другому осуществляется в виде неделимой последовательности операций, называемой *транзакцией*. Шинная транзакция включает в себя два этапа: посылку адреса и прием (или посылку) данных. Во время интервала между посылкой адреса и получением данных процессор или МВВ, инициировавший обмен, могут простаивать.

Подключенное к общей шине устройство, которое может инициировать транзакции чтения или записи, называется *главным устройством шины*. ЦП, например, всегда является главным устройством шины. Шина может иметь несколько главных устройств, например, если имеется несколько ЦП или когда МВВ

могут инициировать транзакции на шине. Если имеется несколько таких устройств, то требуется *схема арбитража*, чтобы решить, кто следующий «захватит» шину.

По способу коммутации различают два типа шин: *шины с коммутацией цепей (circuit-switched bus)* и *шины с коммутацией пакетов (packet-switched bus)*, получившие свои названия по аналогии со способами коммутации в сетях передачи данных.

Шина с коммутацией пакетов обеспечивает большую пропускную способность по сравнению с шиной с коммутацией цепей за счет разделения транзакции на две логические части: запроса шины и ответа. Каждая часть транзакции помечается (тегируется) так, чтобы при их разделении можно было бы определить, к какой транзакции относится та или иная часть. Произвольное количество других пакетов или транзакций могут использовать шину между запросом и ответом, поэтому разделенная на две части транзакция называется *расщепленной*.

Шина с коммутацией цепей не делает расщепления транзакций, любая транзакция на ней есть неделимая операция. Главное устройство запрашивает шину, после арбитража помещает на нее адрес и блокирует шину до окончания обслуживания запроса. При этом часть времени обслуживания тратится на второстепенные операции на шине, например, на задержку выборки из памяти.

Расщепленные транзакции делают шину доступной для других главных устройств в то время, пока идет информационный обмен. Шины с расщеплением транзакций имеют более высокую пропускную способность, но при этом, как правило, и большую задержку, чем шины, которые используются в течение всего времени выполнения транзакции.

По типу синхронизации шины делятся на *синхронные* и *асинхронные*. *Синхронная шина* включает сигналы синхронизации,

которые передаются по линиям управления шины, и фиксированный протокол, определяющий расположение сигналов адреса и данных относительно сигналов синхронизации. Простота подобной организации позволяет получать производительные решения с невысокой стоимостью. Однако в синхронных шинах возникает проблема перекоса синхросигналов, ограничивающая физическую длину шины. Обычно синхронные шины используются для организации обмена между ЦП и памятью.

Асинхронная шина использует старт-стопный режим передачи и протокол «рукопожатия» (*handshaking*) между источником и приемником данных на шине. Эта схема менее быстродействующая, но позволяет использовать длинные шины. Выбор типа шины (синхронной или асинхронной) определяет не только пропускную способность, но также непосредственно влияет на емкость системы ввода/вывода в терминах физического расстояния и количества УВВ, которые могут быть подсоединены к шине.

Шинная организация ввода/вывода имеет два основных преимущества: низкую стоимость и универсальность. К общей шине легко могут быть подсоединены новые УВВ, и одни и те же периферийные устройства можно применять в ЭВМ разной архитектуры, но использующих однотипную шину. Стоимость такой организации получается достаточно низкой, поскольку для реализации множества путей передачи информации используется единственный набор линий шины, разделяемый множеством устройств.

Главным недостатком организации с общей шиной является то, что шина создает узкое место, ограничивая как максимальную пропускную способность системы ввода/вывода, так и ее масштабируемость. Количество внешних устройств, которые одновременно могут быть подключены к общей шине весьма ограниченно. В серверах, где ввод-вывод осуществляется очень

часто, одним из главных вопросов разработки является создание системы нескольких шин, способной удовлетворить все запросы.

Отметим одну из существенных особенностей шинной организации. Любая общая шина является широковещательной (broadcast) средой передачи информации. Одно устройство может предавать информацию сразу нескольким или всем остальным устройствам на шине. Заметим, что локальные сети Ethernet и Token Ring с протоколами, использующими общую среду распространения сигналов, также реализуют логический протокол шинной структуры.

12.3. Последовательный интерфейс RS-232

В последовательном интерфейсе биты передаются друг за другом по одной линии. Эта линия может быть как однонаправленной, так и двунаправленной.

Стандарт RS-232C описывает несимметричные передатчики и приемники. Интерфейс не обеспечивает гальванической развязки устройств. Логической единице (состояние MARK) на входе данных (сигнал RxD) соответствует диапазон напряжения от –12 до –3 В; логическому нулю – от +3 до +12 В (состояние SPACE). Для входов управляющих сигналов состоянию ON («включено») соответствует диапазон от +3 до +12 В, состоянию OFF («выключено») – от –12 до –3 В. Диапазон от –3 до +3 В – зона нечувствительности, обусловливающая гистерезис приемника: состояние линии будет считаться измененным только после пересечения порога. Уровни сигналов на выходах передатчиков должны быть в диапазонах от –12 до –5 В и от +5 до +12 В. Разность потенциалов между схемными землями (SG) соединяемых устройств должна быть менее 2 В, при более высокой разности потенциалов возможно неверное восприятие сигналов.

Интерфейс предполагает наличие защитного заземления для соединяемых устройств, если они оба питаются от сети переменного тока и имеют сетевые фильтры.

Реализацию интерфейса RS-232C можно увидеть на примере нуль-модемного кабеля (рис. 35).

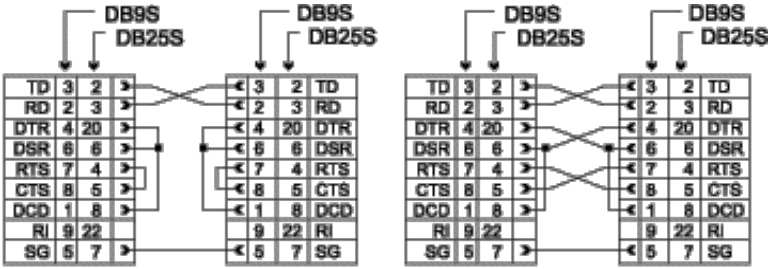


Рис. 35. Нуль-модемный кабель: а – минимальный, б – полный

Таблица 14

Назначение сигналов интерфейса RS-232C

Сигнал	Назначение
PG	*Protected Ground* – защитная земля, соединяется с корпусом устройства и экраном кабеля
SG	*Signal Ground* – сигнальная (схемная) земля, относительно которой действуют уровни сигналов
TD	*Transmit Data* – последовательные данные – выход передатчика
RD	*Receive Data* – последовательные данные – вход приемника
RTS	*Request To Send* – выход запроса передачи данных: состояние «включено» уведомляет модем о наличии у терминала данных для передачи. В полудуплексном режиме используется для управления направлением – состояние «включено» служит сигналом модему на переключение в режим передачи

Окончание табл. 14

Сигнал	Назначение
CTS	*Clear To Send* – вход разрешения терминалу передавать данные. Состояние «выключено» запрещает передачу данных. Сигнал используется для аппаратного управления потоками данных
DSR	*Data Set Ready* – вход сигнала готовности от аппаратуры передачи данных (модем в рабочем режиме подключен к каналу и закончил действия по согласованию с аппаратурой на противоположном конце канала)
DTR	*Data Terminal Ready* – выход сигнала готовности терминала к обмену данными. Состояние «включено» поддерживает коммутируемый канал в состоянии соединения
DCD	*Data Carrier Detected* – вход сигнала обнаружения несущей удаленного модема
RI	*Ring Indicator* – вход индикатора вызова (звонка). В коммутируемом канале этим сигналом модем сигнализирует о принятии вызова

Управление потоком данных. Для управления потоком данных (*flow control*) могут использоваться два варианта протокола – аппаратный и программный. Иногда управление потоком путают с квитированием. Квитирование (*handshaking*) подразумевает посылку уведомления о получении элемента, в то время как управление потоком предполагает посылку уведомления о возможности или невозможности последующего приема данных. Зачастую управление потоком основано на механизме квитирования.

Аппаратный протокол управления потоком ***RTS/CTS*** *(hardware flow control)* использует сигнал CTS, который позволяет остановить передачу данных, если приемник не готов к их

приему (рис. 36). Передатчик «выпускает» очередной байт только при включенной линии CTS. Байт, который уже начал передаваться, задержать сигналом CTS невозможно (это гарантирует целостность посылки). Аппаратный протокол обеспечивает самую быструю реакцию передатчика на состояние приемника. Микросхемы асинхронных приемопередатчиков имеют не менее двух регистров в приемной части – сдвигающий, для приема очередной посылки, и хранящий, из которого считывается принятый байт. Это позволяет реализовать обмен по аппаратному протоколу без потери данных.

Рис. 36. Аппаратное управление потоком

Аппаратный протокол удобно использовать при подключении принтеров и плоттеров, если они его поддерживают. При непосредственном (без модемов) соединении двух компьютеров аппаратный протокол требует перекрестного соединения линий RTS – CTS.

При непосредственном соединении у передающего терминала должно быть обеспечено состояние «включено» на линии CTS (соединением собственных линий RTS – CTS), в противном случае передатчик будет «молчать».

Программный протокол управления потоком ***XON/XOFF*** предполагает наличие двунаправленного канала передачи данных. Работает протокол следующим образом: если устройство, принимающее данные, обнаруживает причины, по которым оно не может их дальше принимать, оно по обратному последовательному каналу посылает байт-символ XOFF (13h). Противопо-

ложное устройство, приняв этот символ, приостанавливает передачу. Когда принимающее устройство снова становится готовым к приему данных, оно посылает символ XON (11h), приняв который противоположное устройство возобновляет передачу. Время реакции передатчика на изменение состояния приемника по сравнению с аппаратным протоколом увеличивается, по крайней мере, на время передачи символа (XON или XOFF) плюс время реакции программы передатчика на прием символа (рис. 37). Из этого следует, что данные без потерь могут приниматься только приемником, имеющим дополнительный буфер принимаемых данных и сигнализирующим о неготовности заблаговременно (имея в буфере свободное место).

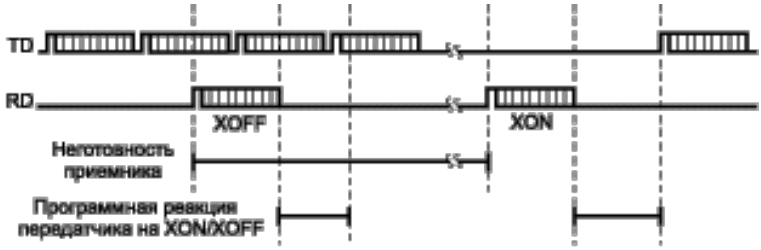


Рис. 37. Программное управление потоком XON/XOFF

Преимущество программного протокола заключается в отсутствии необходимости передачи управляющих сигналов интерфейса – минимальный кабель для двустороннего обмена может иметь только три провода. Недостатком, помимо обязательного наличия буфера и большего времени реакции (снижающего общую производительность канала из-за ожидания сигнала XON), является сложность реализации полнодуплексного режима обмена. В этом случае из потока принимаемых данных должны выделяться (и обрабатываться) символы управления потоком, что ограничивает набор передаваемых символов.

12.4. Параллельный интерфейс Centronics

В компьютерах традиционно используется параллельный интерфейс Centronics, реализуемый LPT-портами, шинами ATA, SCSI и всеми шинами расширения.

В параллельном интерфейсе все биты передаваемого слова (обычно одного, двух или четырех байт) выставляются и передаются по соответствующим параллельно идущим проводам одновременно.

Основным назначением интерфейса Centronics (аналог-ИР-ПР-М) является подключение к компьютеру принтеров различных типов. Поэтому распределение контактов разъема, назначение сигналов, программные средства управления интерфейсом ориентированы именно на это использование. В то же время с помощью данного интерфейса можно подключать к компьютеру и другие внешние устройства, имеющие разъем Centronics.

Основным достоинством использования Centronics по сравнению с ISA является значительно меньший риск вывести компьютер из строя. Главный недостаток этого подхода – меньшая скорость обмена. Назначение контактов разъема Centronics приведено в таблице 15.

Таблица 15

Назначение контактов разъема Centronics

Вывод	Наименование	Направление	Описание
1	STROBE	Out	Strobe (Строб)
2	D0	Out	Data Bit 0
3	D1	Out	Data Bit 1
4	D2	Out	Data Bit 2
5	D3	Out	Data Bit 3
6	D4	Out	Data Bit 4
7	D5	Out	Data Bit 5
8	D6	Out	Data Bit 6
9	D7	Out	Data Bit 7

Окончание табл. 15

Вывод	Наименование	Направление	Описание
10	ACK	In	Acknowledge (Подтверждение)
11	BUSY	In	Busy (Занято)
12	PE	In	Paper End (Конец бумаги)
13	SEL	In	Select (Выбор)
14	AUTOFD	Out	Autofeed (Перевод строки)
15	ERROR	In	Error (Ошибка)
16	INIT	Out	Initialize (Инициализация)
17	SELIN	Out	Select In (Выбор)
18	GND	–	Signal Ground (Корпус)
19	GND	–	Signal Ground (Корпус)
20	GND	–	Signal Ground (Корпус)
21	GND	–	Signal Ground (Корпус)
22	GND	–	Signal Ground (Корпус)
23	GND	–	Signal Ground (Корпус)
24	GND	–	Signal Ground (Корпус)
25	GND	–	Signal Ground (Корпус)

Сигналы Centronics имеют следующее назначение (тип выходных каскадов для всех сигналов – ТТЛ):

– D0...D7 – 8-разрядная шина данных для передачи из компьютера в принтер. Логика сигналов положительная;

– STROBE – сигнал стробирования данных. Данные действительно передаются как по переднему, так и по заднему фронту этого сигнала. Сигнал говорит приемнику (принтеру), что можно принимать данные;

– ACK – сигнал подтверждения принятия данных и готовности приемника (принтера) принять следующие данные. То есть здесь реализуется асинхронный обмен;

– BUSY – сигнал занятости принтера обработкой полученных данных и неготовности принять следующие данные. Сигнал активен также при переходе принтера в состояние off-line или при ошибке, а также при отсутствии бумаги. Компьютер начинает новый цикл передачи только после снятия -ACK и после снятия BUSY;

– AUTO FD – сигнал автоматического перевода строки. Получив его, принтер переводит каретку на следующую строку. Остальные сигналы не являются, вообще говоря, обязательными;

– PE – сигнал конца бумаги. Получив его, компьютер переходит в режим ожидания. Если в принтер вставить лист бумаги, то сигнал снимается;

– SLCT – сигнал готовности приемника. С его помощью принтер говорит о том, что он выбран и готов к работе. У многих принтеров имеется постоянно высокий уровень;

– SLCT IN – сигнал сообщает принтеру о том, что он выбран и последует передача данных;

– ERROR – сигнал ошибки принтера. Активен при внутренней ошибке, переходе принтера в состояние off-line или при отсутствии бумаги. Как видим, здесь многие сигналы дублируют друг друга;

– INIT – сигнал инициализации (сброса) принтера. Его длительность не менее 2,5 мкс. Происходит очистка буфера печати. Временная диаграмма цикла передачи данных представлена на рисунке 38.

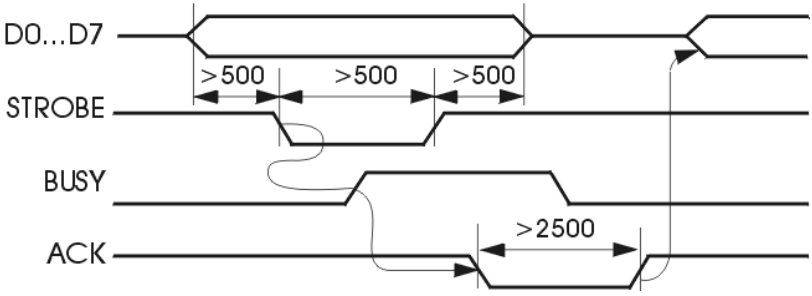


Рис. 38. Временные диаграммы цикла передачи данных в Centronics (все временные интервалы в наносекундах)

Перед началом цикла передачи данных компьютер должен убедиться, что сняты сигналы BUSY и -ACK. После этого выставляются данные, формируется строб, снимается строб и снимаются данные. Принтер должен успеть принять данные с выбранным темпом. При получении строба принтер формирует сигнал BUSY, а после окончания обработки данных выставляет сигнал -ACK, снимает BUSY и снимает -ACK. Затем может начинаться новый цикл.

Все сигналы интерфейса Centronics передаются в уровнях ТТЛ и рассчитаны на подключение одного стандартного входа ТТЛ. Максимальная длина соединительного кабеля по стандарту – 1,8 м.

Формирование и прием сигналов интерфейса Centronics производится путем записи и чтения, выделенных для него портов ввода/вывода. В компьютере могут использоваться три порта Centronics, обозначаемых LPT1 (базовый адрес 378h), LPT2 (базовый адрес 278h) и LPT3 (базовый адрес 3BCh). При этом LPT3 используется в том случае, когда контроллер принтера находится на плате графического адаптера Hercules или EGA. Прерывания портов принтеров (IRQ5 для LPT2 и IRQ7 для LPT1) используются очень редко.

Базовый адрес порта используется для передачи принтеру байта данных. Установленные на линиях данные можно считать из этого же порта. Следующий адрес (базовый + 1) служит для чтения бит состояния принтера (бит 3 соответствует сигналу -EEROR, бит 4 – сигналу PE, бит 6 – сигналу -ACK, бит 7 – сигналу BUSY). Последний используемый адрес (базовый + 2) предназначается для записи бит управления принтером (бит 0 соответствует сигналу -STROBE, бит 1 – сигналу -AUTO FD, бит 2 – сигналу -INIT, бит 3 – сигналу -SLCT IN и, наконец, бит 4, равный единице, разрешает прерывание от принтера).

Приведем упрощенную таблицу сигналов интерфейса Centronics.

Таблица 16

Упрощенная таблица сигналов интерфейса Centronics

Контакты DB-25 IEEE 1284-A	Контакты Centronics IEEE 1284-B	Обозначение	Примечание	Функция
1	1	Strobe	Маркер цикла передачи (выход)	Управление
2	2	Data 0	Сигнал 0 (выход)	Данные
3	3	Data 1	Сигнал 1 (выход)	Данные
4	4	Data 2	Сигнал 2 (выход)	Данные
5	5	Data 3	Сигнал 3 (выход)	Данные
6	6	Data 4	Сигнал 4 (выход)	Данные
7	7	Data 5	Сигнал 5 (выход)	Данные
8	8	Data 6	Сигнал 6 (выход)	Данные
9	9	Data 7	Сигнал 7 (выход)	Данные
10	10	Acknowledge	Готовность принять (вход)	Состояние
11	11	Busy	Занят (вход)	Состояние
12	12	Paper End	Нет бумаги (вход)	Состояние
13	13	Select	Выбор (вход)	Состояние
14	14	Auto Feed	Автоподача (выход)	Управление
15	32	Error	Ошибка (вход)	Состояние
16	31	Init	Инициализация (выход)	Управление
17	36	Select In	Управление печатью (выход)	Управление
18–25	16–17, 19–30	GND	Общий	Земля

Далее на рисунке 39, слева направо, приводятся интерфейс разъема Centronics; кабельный 36-контактный штекер Centronics для подключения внешнего устройства (IEEE 1284-B); 25-контактное гнездо DB-25 Centronics, используемое как LPT-порт на персональных компьютерах (IEEE 1284-A).

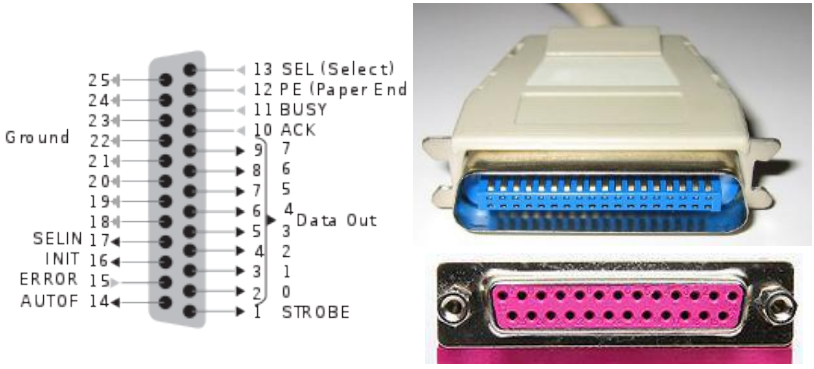


Рис. 39. Интерфейс разъема, штекер и гнездо Centronics

ГЛАВА 13
АРХИТЕКТУРА МОБИЛЬНЫХ УСТРОЙСТВ

13.1. Общее устройство мобильных телефонов

Принципы устройства компьютера и мобильного телефона очень схожи. Вместе с тем для мобильных телефонов используют немного другой подход к программному обеспечению и несколько иную терминологию.

Отметим, что мобильные телефоны, как и компьютеры, являются цифровыми программируемыми автоматами.

С одной стороны, и компьютер, и телефон собраны из вполне конкретных вещественных деталей: микросхем, конденсаторов, печатных плат, проводов, кнопок и т. д. Все это называют *аппаратной частью* мобильного телефона. С другой стороны, во внутренних элементах памяти мобильного телефона содер-

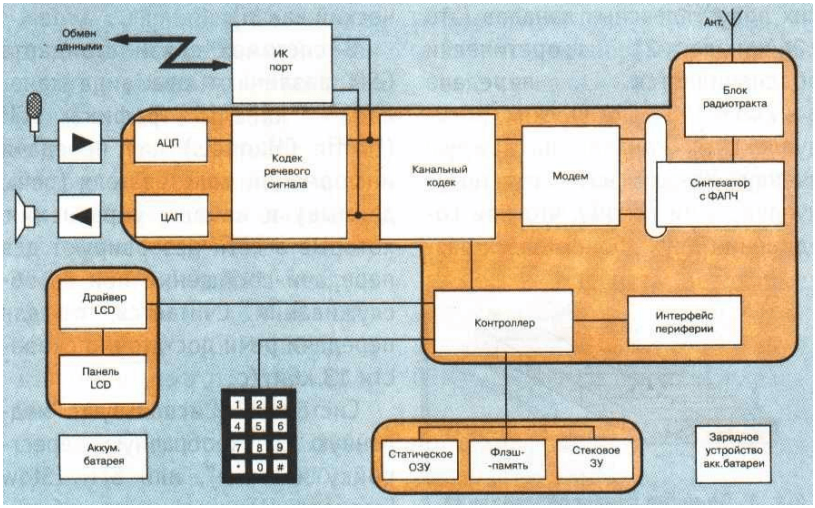


Рис. 40. Простейшая структурная схема мобильного телефона, отражающая его функционал

жится программное обеспечение. В мобильном телефоне все операции с данными (информацией в телефоне – прием, передача и расшифровка сигналов, идентификация телефона в сети и др.) и все компьютерные функции (телефонная книга, голосовой набор, игры и др.) зависят не только и не столько от аппаратной части, сколько от программного обеспечения.

Отражая, прежде всего, функциональность мобильного телефона, на структурной схеме приводят блок радиотракта, блоки приема и воспроизведение звуков (аудиосигналов), дисплей или экран в виде блока и прочие блоки (рис. 40).

Однако устройство мобильного телефона, как и компьютера, является модульно-блочным, т. е. состоит из блоков и модулей (рис. 41). К ним можно отнести следующие компоненты:

– блоки радиосвязи различного назначения (модуль GPRS/GSM с устанавливаемой в него SIM-картой, Wi-Fi-адаптер, Bluetooth-адаптер, GPS- или ГЛОНАСС-приемник и пр.);

– дисплей и связанная с ним сенсорная панель управления;

– блок кнопочного управления мобильным телефоном;

– блок видеокамеры;

– блок аккумуляторной батареи и ее подзарядки;

– блок для подключения внешних устройств в целях обмена разнообразными данными;

– блок, содержащий системную плату с микропроцессором, локальными и общими шинами, оперативную память, микросхемы памяти для хранения программы первичной инициализации телефона без возможности перезаписи, flash-память для хранения программного обеспечения телефона и самых различных данных владельца телефона и прочие блоки.

Естественно, существует отличие аппаратной части телефона от аппаратного обеспечения компьютера по размерам, но главное отличие в разнообразии производителей.

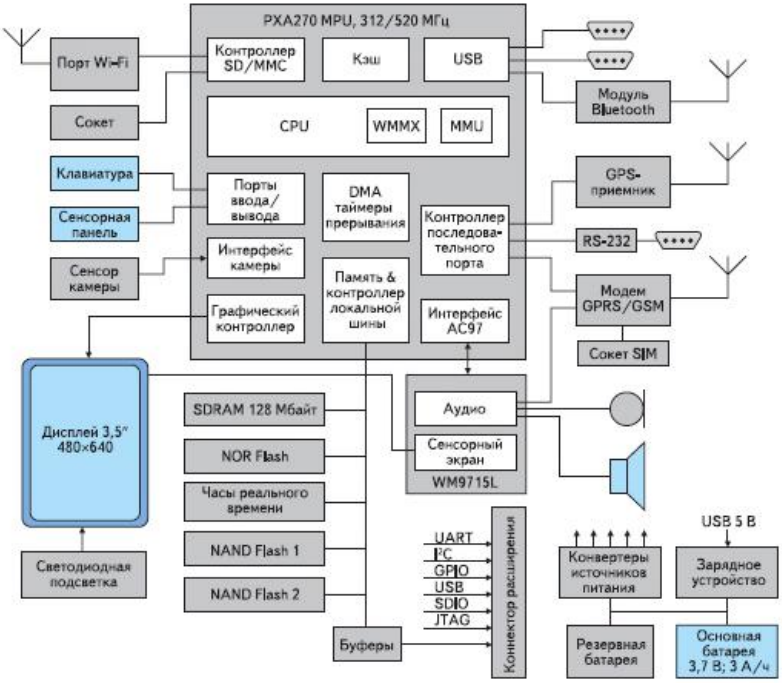


Рис. 41. Модульно-блочная структурная схема мобильного телефона

Для комплектующих компьютеров существует множество открытых стандартов и, таким образом, компьютер можно собрать из отдельных блоков – устройств, выпущенных самыми разными производителями, и при этом выбрать операционную систему и другие программы, являющиеся достаточно универсальными.

А вот мобильные телефоны изначально разрабатывались многими конкурирующими компаниями на различных нестандартизированных аппаратных платформах. Модификаций этих платформ еще больше. Стандартизация коснулась лишь внешних интерфейсов и то не всех. Наглядный пример тому – выбор data-кабеля. Из-за необходимости тесного взаимодействия аппа-

ратной и программной частей телефона программное обеспечение для него создают производители специально под конкретную модель мобильного телефона. И, таким образом, на других телефонах эти программы работать не будут: с «чужой начинкой» телефон даже не включится.

13.2. Взаимосвязь аппаратных и программных составляющих мобильного телефона

Отметим еще раз, что запоминающие устройства обязательно входят в состав компьютеров, мобильных телефонов, да и многих современных «умных» приборов: от кондиционера до телевизора. Они являются электронными компонентами, способными «запоминать» поступающие на них электрические сигналы, сохранять информацию, а потом вновь воспроизводить. Любое такое устройство можно представить как множество ячеек, в каждую из которых помещается элементарная порция информации – 1 байт. Именно в них физически записано программное обеспечение. При включении питания программа первичной инициализации передается процессору. В первый момент это электрический процесс: из строго определенных ячеек памяти информация в виде сигналов по шине поступает в процессор. Выполняя команды, процессор начинает опрашивать запоминающие устройства, считывает и выполняет записанные там программы, переходит к следующим инструкциям. В итоге осуществляется загрузка операционной системы мобильного телефона и интегрированной среды, позволяющей владельцу мобильного телефона использовать его по мере необходимости.

Отметим еще раз, что в персональном компьютере информация хранится на нескольких устройствах (рис. 42). Операционная система, все прикладные программы и пользовательские данные хранятся на жестком диске. Жестких дисков может быть несколь-

ко. Кроме них, существуют и другие, например, компакт-диски, flash-диски и карты памяти, которые физически дисками вообще не являются. Объединяет их одно: информация на любых дисках организована в форме файлов. Другими словами, почти все программное обеспечение компьютера заключено в файловой системе. В принципе, какую-либо программу можно записать не на жесткий диск, а на съемный носитель (компакт- или flash-диск) и запускать непосредственно с этого носителя.

Файловую систему связывают с разметкой диска. Разметка диска напоминает оглавление книги или таблицу, в которой

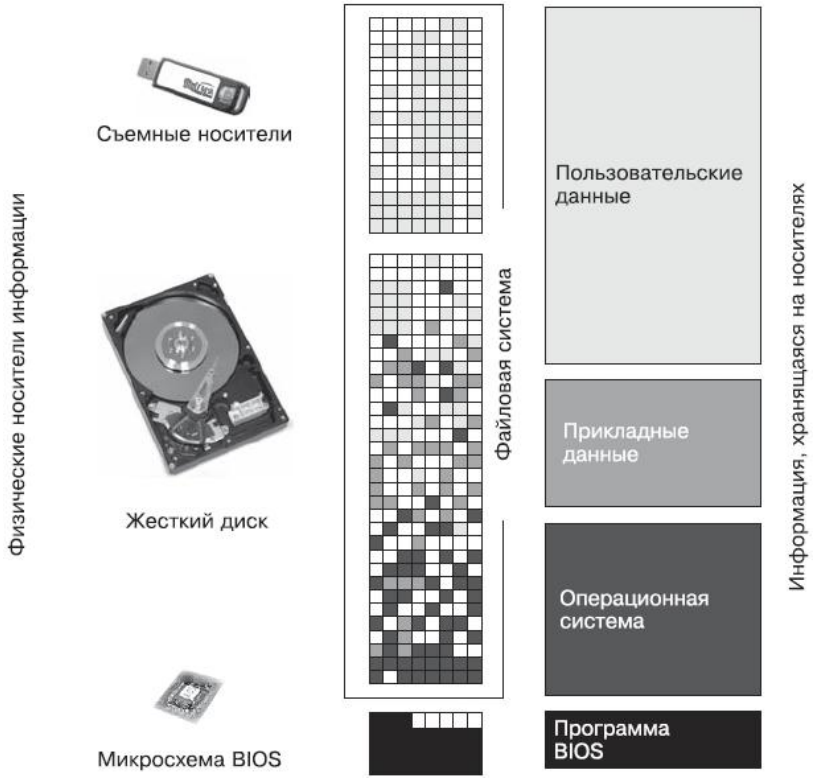


Рис. 42. Физическая и логическая структура постоянной памяти компьютера мобильного телефона

с именами файлов сопоставлены определенные области запоминающего устройства. Благодаря файловой системе осуществляется обращение к конкретным порциям информации «по имени». Вводится имя файла и в ответ компьютер считывает с диска последовательность байт. При этом не имеет значения, записаны ли эти байты в виде намагниченных участков жесткого диска или содержатся в виде электрических зарядов в ячейках flash-диска.

За пределами файловой системы существует только программа BIOS (базовая система ввода/вывода) или UEFI. Она записывается в микросхему, находящуюся на материнской плате, в процессе ее изготовления. Средства BIOS или UEFI включают в себя загрузчик, исполняемый код программы ввода/вывода и значения настроек. Основная ее задача – начальная загрузка компьютера. Когда она выполнена, управление переходит к операционной системе.

В отличие от компьютера в мобильном телефоне запоминающие устройства представлены только микросхемами. В самом общем случае таких микросхем три.

Первая микросхема содержит ячейки памяти, в которых содержится загрузчик, физически находятся в микросхеме процессора. Загрузчик – микропрограмма, при выполнении которой процессор начинает считывать содержимое остальных областей памяти.

Вторая микросхема называется EEPROM (электрически стираемое программируемое постоянное запоминающее устройство). В ее ячейках памяти хранятся все индивидуальные настройки телефона, связанные с аппаратной частью: параметры аудиотракта, радиопередатчика, уникальный номер аппарата (IMEI), параметры аккумулятора и т. д. Эти данные очень напоминают настройки BIOS компьютера.

Третья микросхема – микросхема flash-памяти (*flash memory*); емкость ее в разных аппаратах варьируется от 1 до 32 Мб. В ней записана *прошивка* (*firmware*) телефона, состоящая из двух частей. Первая часть – программный код (*flash* или *main code*) – это собственно программа, которую выполняет аппаратная часть телефона. Ее принято сравнивать с операционной системой компьютера. Операционная система состоит из множества файлов (программ, их компонентов и файлов настроек), каждый из которых выполняет свою достаточно узкую функцию. Такие действия, как просмотр файлов мультимедиа, в компьютере возложены на отдельные прикладные программы. Программный код телефона – единое целое, он включает в себя все функции данного аппарата. Поэтому, если на компьютере отдельные прикладные программы можно устанавливать и удалять, то прошивку заменяют только целиком. Языковой пакет (*lang*) неразрывно связан с программным кодом и обеспечивает поддержку различных языков интерфейса. Кроме того, языковой пакет содержит словари для быстрого набора SMS (поддержку функций iTap или T9). Вторая часть прошивки Flex – файловая часть. Она содержит файлы системных настроек (*seems*), значков, стандартных мелодий, картинок, шаблонов сообщений, а также файлы расположения меню, конфигурации подсветки и др. Иначе говоря, Flex является файловой системой телефона. Здесь же содержатся записи телефонной книги и ежедневника (*user data*), хотя эти данные явно и не отображаются файловой системой.

Такова классическая схема запоминающих устройств или областей памяти мобильного телефона (рис. 43). Она совпадает с перечнем составляющих программного обеспечения. Во многих моделях запоминающие устройства организованы иначе, хотя функциональное назначение составляющих в целом сохраняется. Например, в большинстве современных телефонов

микросхема EEPROM физически отсутствует, а под нее выделяют область в микросхеме flash-памяти. В некоторых телефонах Motorola в единственной микросхеме памяти размещены и загрузчик, и EEPROM, и прошивка со всей файловой системой. Из всего перечисленного файловой структурой обладает только Flex, а остальное программное обеспечение хранится в запоми-

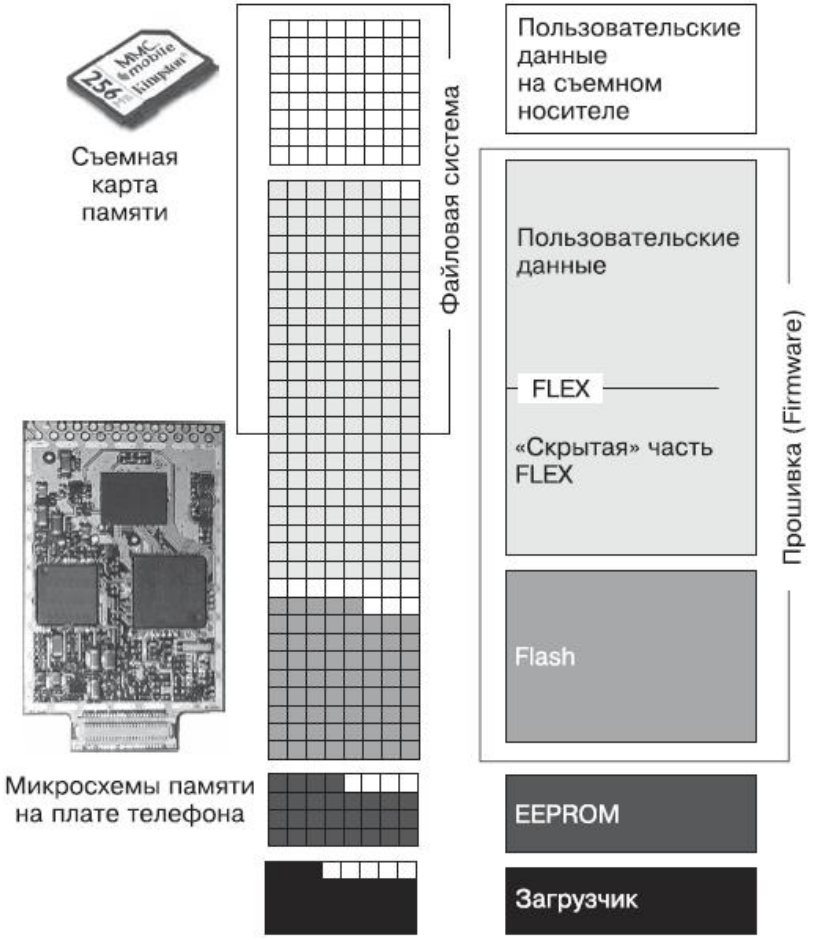


Рис. 43. Физическая и логическая структура памяти телефона

нающих устройствах телефона в виде безымянной непрерывной последовательности байт. Этим телефон очень напоминает BIOS компьютера. Не случайно BIOS тоже называют прошивкой.

Понятием SEEM обозначают отдельные записи в памяти телефона, в которых содержатся настройки телефона, IMEI, удаленные SMS и телефонная книга. Это не область памяти, а лишь собирательное название ячеек, хранящих информацию определенного рода. Часть ячеек SEEM входит в EEPROM, а часть – во FLEX.

В памяти телефонов Motorola особо выделяют область PDS – это уникальная для каждого телефона зона безопасности, содержащая часть SEEM, адрес для сетевого соединения Bluetooth, сведения об операторской блокировке, кодах и паролях, историю прошивок и другую служебную информацию. Функционально PDS соответствует EEPROM других телефонов. Со стертым или неправильным PDS телефон включается только во flash-режим.

Panics – это область памяти, идущая сразу за PDS, в которую телефон записывает информацию об ошибках и сбоях программного обеспечения. Panics не затирается при перепрошивке, уничтожить эту информацию можно только принудительно. Прочитать сообщения Panics можно средствами самого телефона через меню.

SIM-карта мобильного телефона занимает обособленное место в ряду карт памяти. SIM-карта, несмотря на крошечные размеры, является сложным и вполне самостоятельным устройством, которое сообщает телефону отдельные данные для входа в сеть оператора. Часть своей памяти SIM-карта может предоставить для хранения записей телефонной книги, но эта память никоим образом не является запоминающим устройством самого телефона. Карты памяти представляют собой обычные съемные носители информации с собственной файловой системой и к те-

лефону непосредственного отношения также не имеют. Когда они подключены к телефону, их содержимое отображается вместе с пользовательскими файлами, хранящимися во Flex.

Важно напомнить, что файловая информация в компьютере не привязана к каким-либо определенным ячейкам или адресам: она записывается на первое свободное место, а в файловой системе отмечается, что содержимое такого-то файла находится в таком-то месте на диске.

Совершенно иначе обстоит дело с программным обеспечением телефона: для каждого компонента отведено строго определенное число ячеек (байт) по заранее определенным адресам памяти. Для Flex тоже выделен четкий диапазон адресов, а уже внутри этого диапазона файлы произвольного размера могут располагаться как угодно. Именно содержимое Flex просматривают и изменяют программы-менеджеры.

Ячейки памяти принято нумеровать в шестнадцатеричной (hex) системе. Диапазон адресов обозначают начальным и конечным адресом, например 10B1F800 – 10B1FFFF. Можно встретить и другое обозначение диапазона: «начальный адрес 10B1F800, смещение 7FF». Смещение – всего лишь количество ячеек после начального адреса, записанное шестнадцатеричным числом. В данном случае количество ячеек равно 2047.

О загрузчике и EEPROM необходимо помнить следующее: эти составляющие программного обеспечения остаются неизменными на протяжении всей «жизни» телефона. Именно загрузчик и EEPROM отвечают за включение телефона и связь с компьютером через data-кабель. Пока они целы, телефон можно попытаться «реанимировать»: подключить к компьютеру или программатору и восстановить или заменить прошивку. Если поврежден загрузчик, то мобильный телефон окончательно теряет работоспособность.

Поскольку ни загрузчик, ни EEPROM «трогать» владельцу телефона не следует, а организация хранения файлов на телефоне уже известна, рассмотрим операции с программным обеспечением, касающиеся именно прошивки.

13.3. Установка и прошивка мобильного телефона

Вспомним, что для компьютера имеются все возможности для установки, настройки и изменения своего программного обеспечения. Вполне достаточно, чтобы в распоряжении пользователя был системный блок с монитором и клавиатурой, программа BIOS или UEFI, которая всегда «зашита» в системном блоке. Она всегда позволит выполнить начальную загрузку компьютера и установить на жесткий диск операционную систему с внешнего носителя. Настроить операционную систему можно как угодно, пользуясь средствами самой системы. Можно обновить ее, установить или удалить отдельные компоненты. Установка прикладных программ не вызывает затруднений даже у начинающего пользователя: в сущности, необходимо скопировать все программы (файлы) из дистрибутива, находящегося на внешнем носителе, на жесткий диск компьютера. Разумеется, с помощью операционной системы и прикладных программ можно выполнять любые действия с пользовательскими данными (документами): создавать новые, копировать их с одного диска на другой и т. д. Все перечисленное, начиная с установки операционной системы и заканчивая работой с документами, сводится к операциям с файлами.

Однако в телефон программное обеспечение попадает несколько иначе, а он сам играет в этом процессе достаточно пассивную роль. Этот процесс называется прошивкой или перепрошивкой. Чтобы внести ясность в русско-английскую терминоло-

гию, скажем, что в процессе прошивки в память телефона (*flash*) записывают прошивку (*firmware*), которая состоит из программного кода (*flash*) и файловой части (*flex*) (рис. 44).

Для прошивки в телефон программного обеспечения необходимы следующие действия:

1. Выполнить установку на компьютер программы для прошивки мобильного телефона. Эти приложения называют флешерами (flasher) или прошивальщиками.

2. Полностью зарядить аккумулятор телефона. Еще лучше, если его питание осуществляется через кабель или от зарядного устройства.

3. Подключить телефон к компьютеру data-кабелем, установить необходимые драйверы.

4. Запустить программу для прошивки телефона. Открыть этой программой файл, в котором содержится образ прошивки.

5. Перевести телефон в режим обновления программного обеспечения, в так называемый flash-режим (*flash mode*). В некоторых случаях это делает сама программа для прошивки. Другие телефоны переводят в данный режим вручную одновременным нажатием нескольких кнопок телефона.

6. Записать в память телефона программное обеспечение, содержащееся в файле-образе прошивки. Следует указать, в какие ячейки памяти должна быть записана прошивка. Многие программы для прошивки предлагают нужные значения исходя из модели телефона или данных, содержащихся в файле-образе. Процесс записи занимает достаточно много времени.

7. Перезагрузить телефон. Многие программы делают это автоматически, закончив прошивку.

Такие программы позволяют выполнить и обратную операцию: полностью или частично считать из телефона содержимое

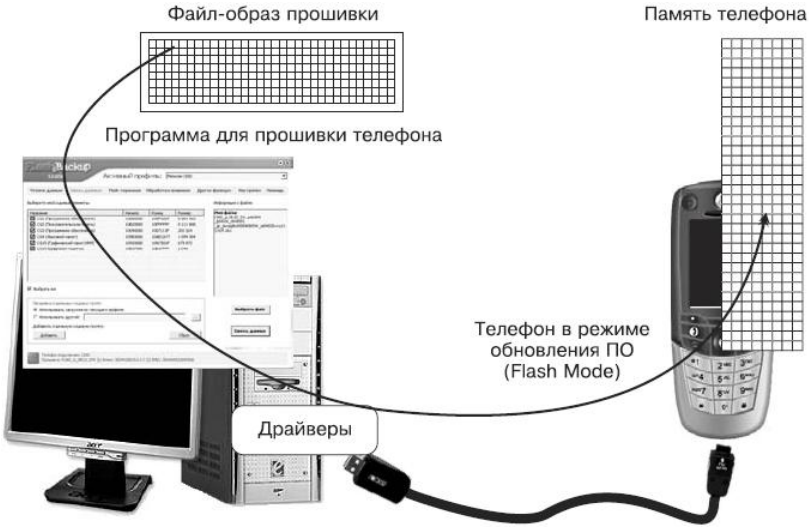


Рис. 44. Общая схема прошивки телефона

его памяти, а затем сохранить его в виде файла-образа прошивки или отдельных его частей. Эта процедура называется чтением и резервным копированием прошивки (*flash backup*) или дампом (*dump*).

ГЛАВА 14
ОСНОВЫ ОРГАНИЗАЦИИ КОМПЬЮТЕРНЫХ СЕТЕЙ

14.1. Назначение и классификация компьютерных сетей

Обобщенная структура компьютерных сетей. Объединение в один комплекс средств вычислительной техники, аппаратуры связи и каналов передачи данных предъявляет специфические требования со стороны каждого элемента ассоциации, а также требует формирования специальной терминологии.

Абоненты сети – объекты, генерирующие или потребляющие информацию в сети.

Физическая передающая среда – линии связи или пространство, в котором распространяются электрические сигналы, и аппаратура передачи данных.

На базе физической передающей среды строится коммуникационная сеть, которая обеспечивает передачу информации между абонентами.

Такой подход позволяет рассматривать любую компьютерную сеть как совокупность абонентов и коммуникационной сети.

Классификация компьютерных сетей. В зависимости от территориального расположения сети можно разделить на три основных класса:

– глобальные сети (*WAN – wide area network*);
– региональные сети (*MAN – metropolitan area network*);
– локальные сети (*LAN – local area network*).

Глобальная сеть объединяет абонентов, расположенных в различных странах, на различных континентах. Взаимодействие между абонентами такой сети может осуществляться

на базе телефонных линий связи, радиосвязи и систем спутниковой связи. Глобальные сети позволяют решить проблему объединения информационных ресурсов всего человечества и организации доступа к этим ресурсам.

Региональная сеть связывает абонентов, расположенных на значительном расстоянии друг от друга. Она может включать абонентов внутри большого города, экономического региона, отдельной страны. Обычно расстояние между абонентами региональной сети составляет десятки – сотни километров.

Локальная сеть объединяет абонентов, расположенных в пределах небольшой территории. В настоящее время не существует четких ограничений на территориальный разброс абонентов локальной сети. Обычно такая сеть привязана к конкретному месту. К классу локальных сетей относятся сети отдельных министерств, организаций, предприятий, фирм, банков, офисов и т. д. Протяженность такой сети во многих случаях можно ограничить пределами 2–2,5 км.

Объединение глобальных, региональных и локальных сетей позволяет создавать *многосетевые иерархии*. Они обеспечивают мощные, экономически целесообразные средства обработки огромных информационных массивов и доступ к неограниченным информационным ресурсам. На рис. 45 приведена одна из возможных иерархий сетей. Локальные сети могут входить как компоненты в состав региональной сети, региональные сети – объединяться в составе глобальной сети и, наконец, глобальные сети могут также образовывать сложные структуры.

Пример. Компьютерная сеть Интернет является наиболее популярной глобальной сетью. В ее состав входит множество свободно соединенных сетей. Внутри каждой сети, входящей в Internet, существуют конкретная структура связи и определенная дисциплина управления. Внутри Internet структура и ме-

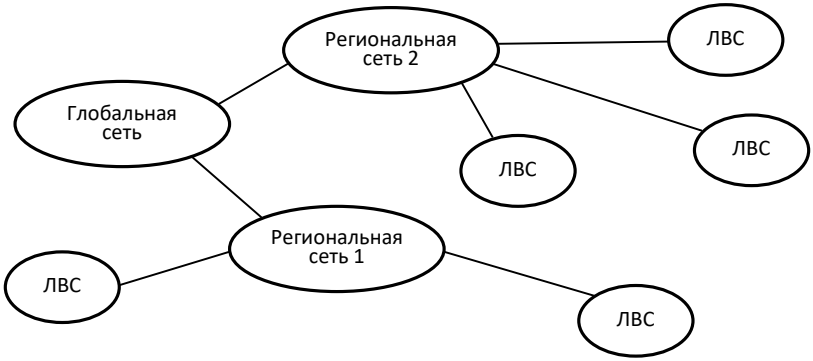


Рис. 45. Иерархия компьютерных сетей

тоды соединений между различными сетями для конкретного пользователя не имеют никакого значения.

14.2. Характеристика процесса передачи данных

Режимы передачи данных. Любая коммуникационная сеть должна включать следующие основные компоненты: *передатчик*, *сообщение*, *средства передачи*, *приемник*.

Передатчик – устройство, являющееся источником данных.

Приемник – устройство, принимающее данные. Приемником могут быть компьютер, терминал или какое-либо цифровое устройство.

В качестве передатчиков и приемников чаще всего выступают отдельные абоненты.

Сообщение – цифровые данные определенного формата, предназначенные для передачи. Это может быть файл базы данных, таблица, ответ на запрос, текст или изображение.

Средства передачи – физическая передающая среда и специальная аппаратура, обеспечивающая передачу сообщений.

Для передачи сообщений в сетях используются различные типы каналов связи. Наиболее распространены выделенные те-

лефонные каналы и специальные каналы для передачи цифровой информации. Применяются также радиоканалы и каналы спутниковой связи.

Особняком в этом отношении стоят ЛВС, где в качестве передающей среды используются витая пара проводов, коаксиальный кабель и оптоволоконный кабель.

Для характеристики процесса обмена сообщениями в сети по каналам связи используются следующие понятия: *режим передачи, код передачи, тип синхронизации*.

Режим передачи. Существуют три режима передачи: симплексный, полудуплексный и дуплексный.

Симплексный режим – передача данных только в одном направлении.

Полудуплексный режим – попеременная передача информации, когда источник и приемник последовательно меняются местами.

Дуплексный режим – одновременные передача и прием сообщений. Является наиболее скоростным режимом работы.

Коды передачи данных. Для передачи информации по каналам связи используются специальные коды. Коды эти стандартизованы и определены рекомендациями Международной организации по стандартизации (МОС) или Международного консультативного комитета по телефонии и телеграфии (МККТТ).

Наиболее распространенным кодом передачи по каналам связи является код ASCII, принятый для обмена информацией практически во всем мире (отечественный аналог – код КОИ-7).

Следует обратить внимание еще на один способ связи между ЭВМ, когда ЭВМ объединены в комплекс с помощью интерфейсного кабеля и с помощью двухпроводной линии связи.

Интерфейсный кабель – это набор проводов, по которым передаются сигналы от одного устройства компьютера к другому. Чтобы обеспечить быстродействие, для каждого сигнала выделен отдельный провод. Сигналы передаются в определенной последовательности и в определенных комбинациях друг с другом.

Для передачи кодовой комбинации используется столько линий, сколько бит эта комбинация содержит. Каждый бит передается по отдельному проводу. Это *параллельная передача* или *передача параллельным кодом*. Предпочтение такой передаче отдается при организации локальных МВК, для внутренних связей ЭВМ и для небольших расстояний между абонентами сети. Передача параллельным кодом обеспечивает высокое быстродействие, но требует повышенных затрат на создание физической передающей среды и обладает плохой помехозащищенностью. В сетях передача параллельными кодами практически не используется.

Для передачи кодовой комбинации по двухпроводной линии группа бит передается по одному проводу бит за битом. Это передача информации *последовательным кодом*. Она, вполне естественно, медленнее, но экономически более выгодна для передачи сообщений на большие расстояния.

Типы синхронизации данных. Процессы передачи или приема информации в сетях могут быть привязаны к определенным временным отметкам, т. е. один из процессов может начаться только после того, как получит полностью данные от другого процесса. Такие процессы называются синхронными.

В то же время существуют процессы, в которых нет такой привязки, и они могут выполняться независимо от степени полноты переданных данных. Такие процессы называются асинхронными.

Синхронизация данных – согласование различных процессов во времени. В системах передачи данных используются два способа передачи данных: синхронный и асинхронный.

При *синхронной передаче* (рис. 46) информация передается блоками, которые обрамляются специальными управляющими символами. В состав блока включаются также специальные синхросимволы, обеспечивающие контроль состояния физической передающей среды, и символы, позволяющие обнаруживать ошибки при обмене информацией. В конце блока данных при синхронной передаче в канал связи выдается контрольная последовательность, сформированная по специальному алгоритму. По этому же алгоритму формируется контрольная последовательность при приеме информации из канала связи. Если обе последовательности совпадают – ошибок нет. Блок данных принят. Если же последовательности не совпадают – ошибка. Передача повторяется до положительного результата проверки. Если повторные передачи не дают положительного результата, то фиксируется состояние аварии.

Синхронная передача характеризуется как высокоскоростная и почти безошибочная. Она используется для обмена сооб-

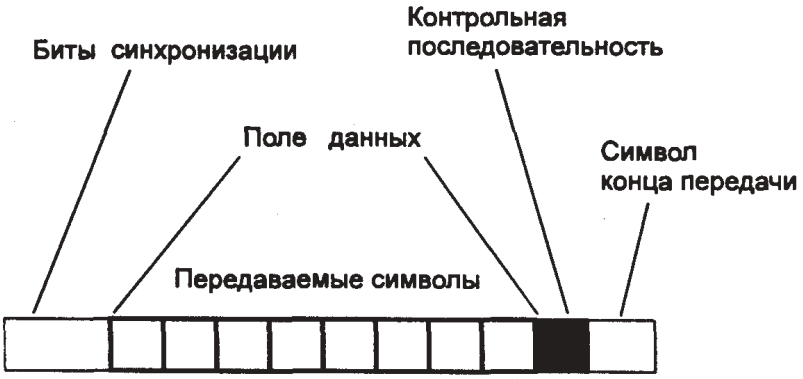


Рис. 46. Синхронная передача данных

щениями между ЭВМ в сетях. Синхронная передача требует дорогостоящего оборудования.

При *асинхронной передаче* (рис. 47) данные передаются в канал связи как последовательность бит, из которой при приеме необходимо выделить байты для последующей их обработки. Для этого каждый байт ограничивается стартовым и стоповым битами, которые и позволяют произвести выделение их из потока передачи. Иногда в линиях связи с низкой надежностью используется несколько таких бит. Дополнительные стартовые и стоповые биты снижают эффективную скорость передачи данных и, соответственно, пропускную способность канала связи. В то же время асинхронная передача не требует дорогостоящего оборудования и отвечает требованиям организации диалога в сети при взаимодействии персональных ЭВМ.

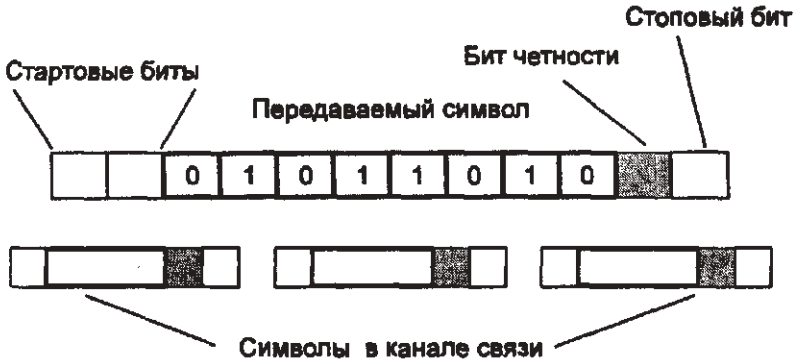


Рис. 47. Асинхронная передача данных

14.3. Аппаратная реализация передачи данных

Способы передачи цифровой информации. Цифровые данные по проводнику передаются путем смены текущего напряжения: нет напряжения – «О», есть напряжение – «I». Существуют

два способа передачи информации по физической передающей среде: *цифровой* и *аналоговый*.

Если все абоненты компьютерной сети ведут передачу данных по каналу на одной частоте, такой канал называется узкополосным (пропускает одну частоту).

Если каждый абонент работает на своей собственной частоте по одному каналу, то такой канал называется широкополосным (пропускает много частот). Использование широкополосных каналов позволяет экономить на их количестве, но усложняет процесс управления обменом данными.

При *цифровом* или *узкополосном способе передачи* (рис. 48) данные передаются в их естественном виде на единой частоте. Узкополосный способ позволяет передавать только цифровую информацию, обеспечивает в каждый данный момент времени возможность использования передающей среды только двумя пользователями и допускает нормальную работу только на ограниченном расстоянии (длина линии связи не более 1000 м). В то же время узкополосный способ передачи обеспечивает высокую скорость обмена данными – до 10 Мбит/с и позволяет создавать легко конфигурируемые сети. Подавляющее число локальных сетей использует узкополосную передачу.

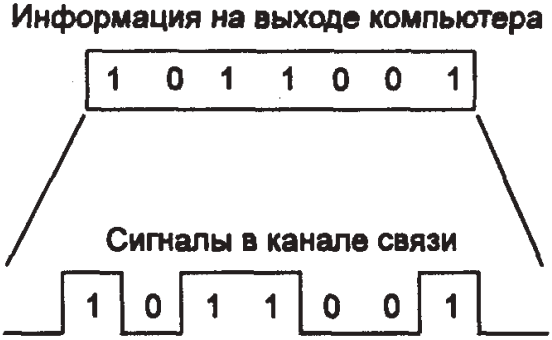


Рис. 48. Цифровой способ передачи

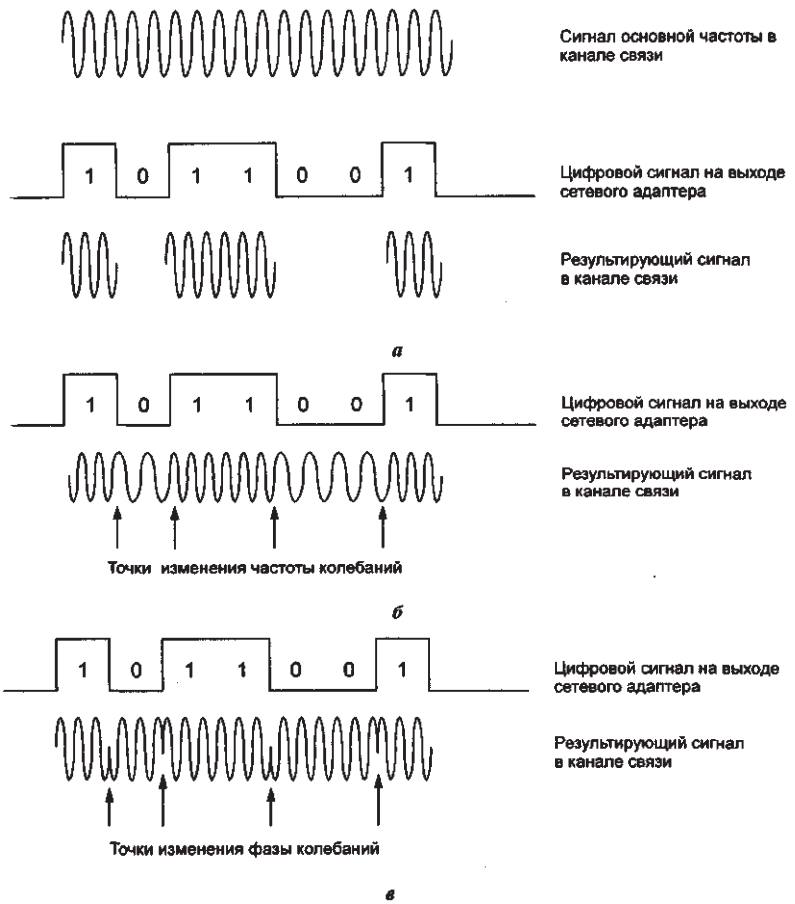


Рис. 49. Способы передачи цифровой информации по аналоговому сигналу: а – амплитудная модуляция; б – частотная модуляция; в – фазовая модуляция

Аналоговый способ передачи цифровых данных (рис. 49) обеспечивает широкополосную передачу за счет использования в одном канале сигналов различных несущих частот.

При аналоговом способе передачи происходит управление параметрами сигнала несущей частоты для передачи по каналу связи цифровых данных.

Сигнал несущей частоты представляет собой гармоническое колебание, описываемое уравнением:

$$X = X_{max} sin(\omega t + \varphi_o),$$

где: X_{max} – амплитуда колебаний; ω – частота колебаний; t – время; φ_o – начальная фаза колебаний.

Передать цифровые данные по аналоговому каналу можно, управляя одним из параметров сигнала несущей частоты: амплитудой, частотой или фазой. Так как необходимо передавать данные в двоичном виде (последовательность единиц и нулей), то можно предложить следующие способы управления (модуляции): амплитудный, частотный, фазовый.

Проще всего понять принцип *амплитудной модуляции*: «0» – отсутствие сигнала, т. е. отсутствие колебаний несущей частоты; «1» – наличие сигнала, т. е. наличие колебаний несущей частоты. Есть колебания – единица, нет колебаний – нуль (рис. 49*а*).

Частотная модуляция предусматривает передачу сигналов 0 и 1 на разной частоте. При переходе от 0 к 1 и от 1 к 0 происходит изменение сигнала несущей частоты (рис. 49*б*).

Наиболее сложной для понимания является *фазовая модуляция*. Суть ее в том, что при переходе от 0 к 1 и от 1 к 0 меняется фаза колебаний, т. е. их направление (рис. 49*в*).

В сетях высокого уровня иерархии – глобальных и региональных используется также и *широкополосная передача*, которая предусматривает работу для каждого абонента на своей частоте в пределах одного канала. Это обеспечивает взаимодействие большого количества абонентов при высокой скорости передачи данных. Кроме того, широкополосная передача позволяет совмещать в одном канале передачу цифровых данных, изображения и звука, что является необходимым требованием современных систем мультимедиа.

Пример. Типичным аналоговым каналом является телефонный канал. Когда абонент снимает трубку, то слышит равномерный звуковой сигнал – это и есть сигнал несущей частоты. Так как он находится в диапазоне звуковых частот, то его называют тональным сигналом. Для передачи по телефонному каналу речи необходимо управлять сигналом несущей частоты – модулировать его. Воспринимаемые микрофоном звуки преобразуются в электрические сигналы, а те, в свою очередь, и модулируют сигнал несущей частоты. При передаче цифровой информации управление производят информационные байты – последовательность единиц и нулей.

Аппаратные средства. Как уже говорилось ранее, для передачи цифровой информации по каналу связи необходимо поток бит преобразовать в аналоговые сигналы, а при приеме информации из канала связи в ЭВМ выполнить обратное действие – преобразовать аналоговые сигналы в поток бит, которые может обрабатывать ЭВМ. Такие преобразования выполняет специальное устройство – *модем*.

Модем – устройство, выполняющее модуляцию и демодуляцию информационных сигналов при передаче их из ЭВМ в канал связи и при приеме в ЭВМ из канала связи.

Однако, чтобы обеспечить передачу информации из ЭВМ в коммуникационную среду, необходимо согласовать сигналы внутреннего интерфейса ЭВМ с параметрами сигналов, передаваемых по каналам связи. При этом должно быть выполнено как физическое согласование (форма, амплитуда и длительность сигнала), так и кодовое.

Технические устройства, выполняющие функции сопряжения ЭВМ с каналами связи, называются *адаптерами* или *сетевыми адаптерами*. Один адаптер обеспечивает сопряжение с ЭВМ одного канала связи. Кроме одноканальных адаптеров

используются и многоканальные устройства – *мультиплексоры передачи данных* или просто *мультиплексоры*.

Мультиплексор передачи данных – устройство сопряжения ЭВМ с несколькими каналами связи.

Мультиплексоры использовались в системах телеобработки данных – первом шаге на пути к созданию сетей. В дальнейшем при появлении сетей со сложной конфигурацией и с большим количеством абонентских систем для реализации функций сопряжения стали применяться специальные связные процессоры.

Наиболее дорогим компонентом сети является канал связи. Поэтому при построении ряда сетей стараются сэкономить на каналах связи, коммутируя несколько внутренних каналов связи на один внешний. Для выполнения функций коммутации используются специальные устройства – *концентраторы*.

Концентратор – устройство, коммутирующее несколько каналов связи на один путем частотного разделения.

В ЛВС, где физическая передающая среда представляет собой кабель ограниченной длины, для увеличения протяженности сети используются специальные устройства – *повторители*.

Повторитель – устройство, обеспечивающее сохранение формы и амплитуды сигнала при передаче его на большее, чем предусмотрено данным типом физической передающей среды, расстояние.

Существуют *локальные* и *дистанционные* повторители. Локальные повторители позволяют соединять фрагменты сетей, расположенные на расстоянии до 50 м, а дистанционные – до 2 км.

Характеристики коммуникационной сети. Для оценки качества коммуникационной сети можно использовать следующие характеристики:

– скорость передачи данных по каналу связи;

– пропускную способность канала связи;
– достоверность передачи информации;
– надежность канала связи и модемов.

Скорость передачи данных по каналу связи измеряется количеством битов информации, передаваемых за единицу времени – секунду.

Часто используется единица измерения скорости – бод. Бод – число изменений состояния среды передачи в секунду. Так как каждое изменение состояния может соответствовать нескольким битам данных, то реальная скорость в битах в секунду может превышать скорость в бодах.

Скорость передачи данных зависит от типа и качества канала связи, типа используемых модемов и принятого способа синхронизации.

Так, для асинхронных модемов и телефонного канала связи диапазон скоростей составляет 300–9600 бит/с, а для синхронных – 1200–19200 бит/с.

Для пользователей сетей значение имеют не абстрактные биты в секунду, а информация, единицей измерения которой служат байты или знаки. Поэтому более удобной характеристикой канала является его пропускная способность, которая оценивается количеством знаков, передаваемых по каналу за единицу времени – секунду. При этом в состав сообщения включаются и все служебные символы. Теоретическая пропускная способность определяется скоростью передачи данных. Реальная пропускная способность зависит от ряда факторов, среди которых и способ передачи, и качество канала связи, и условия его эксплуатации, и структура сообщений.

Существенной характеристикой коммуникационной системы любой сети является *достоверность* передаваемой информации. Так как на основе обработки информации о состоянии

объекта управления принимаются решения о том или ином ходе процесса, то от достоверности информации может зависеть судьба объекта. Достоверность передачи информации оценивают как отношение количества ошибочно переданных знаков к общему числу переданных знаков. Требуемый уровень достоверности должны обеспечивать как аппаратура, так и канал связи. Нецелесообразно использовать дорогостоящую аппаратуру, если относительно уровня достоверности канал связи не обеспечивает необходимых требований.

Для сетей этот показатель должен лежать в пределах 10^{-6}–10^{-7} ошибок/знак, т. е. допускается одна ошибка на миллион переданных знаков или на десять миллионов переданных знаков.

Наконец, надежность коммуникационной системы определяется либо долей времени исправного состояния в общем времени работы, либо средним временем безотказной работы. Вторая характеристика позволяет более эффективно оценить надежность системы.

Для сетей среднее время безотказной работы должно быть достаточно большим и составлять, как минимум, несколько тысяч часов.

Основные формы взаимодействия абонентских ЭВМ. Самое существенное в работе сети – определение набора функций, доступных ее абоненту.

Так как пользователи сети работают в определенных предметных областях и используют сеть для решения своих прикладных задач, напомним, что такое процесс, и определим понятие прикладной процесс.

Процесс – некоторая последовательность действий для решения задачи, определяемая программой.

Прикладной процесс – некоторое приложение пользователя, реализованное в прикладной программе.

Отсюда следует, что взаимодействие абонентских ЭВМ в сети можно рассматривать как взаимодействие прикладных процессов конечных пользователей через коммуникационную сеть.

Коммуникационная сеть обеспечивает физическое соединение между абонентскими ЭВМ – передачу сообщений по каналам связи. Для того, чтобы могли взаимодействовать процессы, между ними должна существовать и логическая связь (процессы должны быть инициированы, файлы данных открыты).

Анализ работы сетей позволяет установить следующие формы взаимодействия между абонентскими ЭВМ:

– терминал – удаленный процесс;
– терминал – доступ к удаленному файлу;
– терминал – доступ к удаленной базе данных;
– терминал – терминал;
– электронная почта.

Взаимодействие «*терминал – удаленный процесс*» предусматривает обращение с терминала одной из абонентских ЭВМ к процессу, находящемуся на другой абонентской ЭВМ сети. При этом устанавливается логическая связь с процессом и проводится сеанс работы с ним. Можно запустить удаленный процесс, получить результаты обработки данных этим процессом. Возможна также работа в режиме консоли – трансляция команд сетевой операционной системы на удаленную ЭВМ.

При взаимодействии «*терминал – доступ к удаленному файлу*» можно открыть удаленный файл, модифицировать его или произвести транспортировку этого файла на любое внешнее устройство абонентской ЭВМ для дальнейшей работы с ним в локальном режиме.

Работа в режиме «*терминал – доступ к удаленной базе данных*» аналогична предыдущей форме взаимодействия. Только в этом случае производится работа с базой данных в ее полном

объеме в соответствии с правами доступа, которыми обладает данный пользователь сети.

Взаимодействие «*терминал – терминал*» предусматривает обмен сообщениями между абонентами сети в диалоговом режиме. Сообщения могут посылаться как отдельным абонентам, так и группам абонентов сети. Длина сообщения не должна превышать некоторой установленной для данной сети величины (обычно – строка на экране терминала).

Форма взаимодействия «*электронная почта*» в последнее время стала очень распространенной. Каждый абонент имеет на своей ЭВМ «почтовый ящик». Это специальный файл, в который записываются все поступающие в его адрес сообщения. Конечный пользователь может проверять в начале работы свой «почтовый ящик», выводить сообщения на печать и передавать сообщения в адрес других абонентов сети.

14.4. Архитектура компьютерных сетей

Эталонные модели взаимодействия систем. Модель взаимодействия открытых систем. Для определения задач, поставленных перед сложным объектом, а также для выделения главных характеристик и параметров, которыми он должен обладать, создаются общие модели таких объектов. Общая модель сети определяет характеристики сети в целом и характеристики, и функции входящих в нее основных компонентов.

Архитектура сети – описание ее общей модели.

Многообразие производителей сетей и сетевых программных продуктов создало проблему объединения сетей различных архитектур. Для ее решения МОС разработала модель архитектуры открытых систем.

Открытая система – система, взаимодействующая с другими системами в соответствии с принятыми стандартами.

Предложенная модель архитектуры открытых систем служит базой для производителей при разработке совместимого сетевого оборудования. Эта модель не является неким физическим телом, отдельные элементы которого можно осязать. Модель представляет собой самые общие рекомендации для построения стандартов совместимых сетевых программных продуктов. Эти рекомендации должны быть реализованы как в аппаратуре, так и в программных средствах сетей.

В настоящее время модель взаимодействия открытых систем (ВОС) является наиболее популярной сетевой архитектурной моделью. Модель рассматривает общие функции, а не специальные решения, поэтому *не все реальные сети* абсолютно точно ей следуют. Модель взаимодействия открытых систем состоит из семи уровней (рис. 50).

Рис. 50. Эталонная модель архитектуры открытых систем

7-й уровень – *прикладной* – обеспечивает поддержку прикладных процессов конечных пользователей. Этот уровень определяет круг прикладных задач, реализуемых в данной сети.

Он также содержит все необходимые элементы сервиса для прикладных программ пользователя. На прикладной уровень могут быть вынесены некоторые задачи сетевой операционной системы.

6-й уровень – *представительный* – определяет синтаксис данных в модели, т. е. представление данных. Он гарантирует представление данных в кодах и форматах, принятых в данной системе. В некоторых системах этот уровень может быть объединен с прикладным.

5-й уровень – *сеансовый* – реализует установление и поддержку сеанса связи между двумя абонентами через коммуникационную сеть. Он позволяет производить обмен данными в режиме, определенном прикладной программой, или предоставляет возможность выбора режима обмена. Сеансовый уровень поддерживает и завершает сеанс связи.

Три верхних уровня объединяются под общим названием – *процесс* или *прикладной процесс*. Эти уровни определяют функциональные особенности сети как прикладной системы.

4-й уровень – *транспортный* – обеспечивает интерфейс между процессами и сетью. Он устанавливает логические каналы между процессами и обеспечивает передачу по этим каналам информационных пакетов, которыми обмениваются процессы. Логические каналы, устанавливаемые транспортным уровнем, называются транспортными каналами.

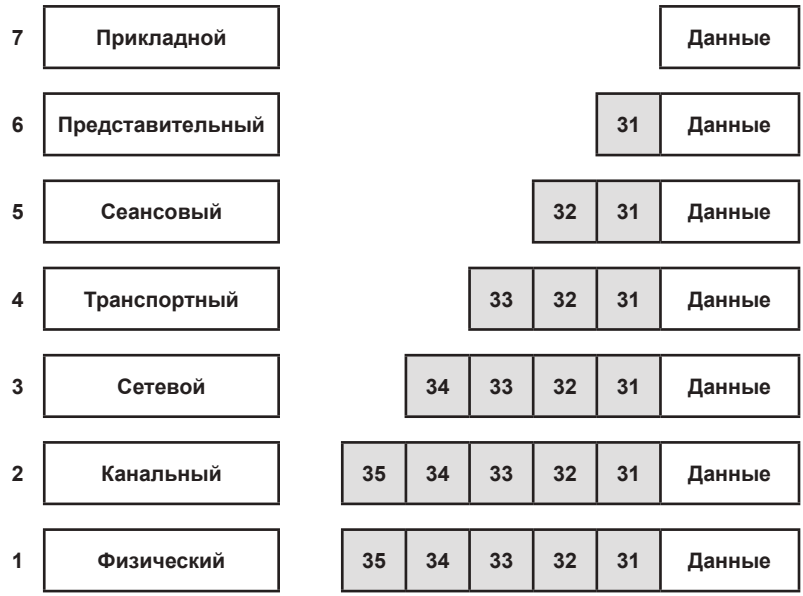
Пакет – группа байт, передаваемых абонентами сети друг другу.

3-й уровень – *сетевой* – определяет интерфейс оконечного оборудования данных пользователя с сетью коммутации пакетов. Он также отвечает за маршрутизацию пакетов в коммуникационной сети и за связь между сетями – реализует межсетевое взаимодействие.

В технике коммуникаций используется термин оконечное оборудование данных. Он определяет любую аппаратуру, подключенную к каналу связи, в системе обработки данных (компьютер, терминал, специальная аппаратура).

2-й уровень – *канальный* – уровень звена данных – реализует процесс передачи информации по информационному каналу. Информационный канал – логический канал, он устанавливается между двумя ЭВМ, соединенными физическим каналом. Канальный уровень обеспечивает управление потоком данных в виде кадров, в которые упаковываются информационные пакеты, обнаруживает ошибки передачи и реализует алгоритм восстановления информации в случае обнаружения сбоев или потерь данных.

1-й уровень – *физический* – выполняет все необходимые процедуры в канале связи. Его основная задача – управление аппаратурой передачи данных и подключенным к ней каналам связи.



При передаче информации от прикладного процесса в сеть происходит ее обработка уровнями модели взаимодействия открытых систем (рис. 51).

Смысл этой обработки заключается в том, что каждый уровень добавляет к информации процесса свой *заголовок* – служебную информацию, которая необходима для адресации сообщений и для некоторых контрольных функций. Канальный уровень кроме заголовка добавляет еще и *концевик* – контрольную последовательность, которая используется для проверки правильности приема сообщения из коммуникационной сети.

Физический уровень заголовка не добавляет. Сообщение, обрамленное заголовками и концевиком, уходит в коммуникационную сеть и поступает на абонентские ЭВМ сети. Каждая абонентская ЭВМ, принявшая сообщение, дешифрирует адреса и определяет, предназначено ли ей данное сообщение.

При этом в абонентской ЭВМ происходит обратный процесс – чтение и отсечение заголовков уровнями модели взаимодействия открытых систем. Каждый уровень реагирует только на свой заголовок. Заголовки верхних уровней нижними уровнями не воспринимаются и не изменяются – они «прозрачны» для нижних уровней. Так, перемещаясь по уровням модели ВОС, информация, наконец, поступает к процессу, которому она была адресована.

В чем же основное достоинство семиуровневой модели ВОС? В процессе развития и совершенствования любой системы возникает потребность изменять ее отдельные компоненты. Иногда это вызывает необходимость изменять и другие компоненты, что существенно усложняет и затрудняет процесс модернизации системы.

Здесь и проявляются преимущества семиуровневой модели. Если между уровнями определены однозначно интерфейсы,

то изменение одного из уровней не влечет за собой необходимости внесения изменений в другие уровни. Таким образом, существует относительная независимость уровней друг от друга.

Необходимо сделать и еще одно замечание относительно реализации уровней модели ВОС в реальных сетях. Функции, описываемые уровнями модели, должны быть реализованы либо в аппаратуре, либо в виде программ.

Функции физического уровня всегда реализуются в аппаратуре. Это адаптеры, мультиплексоры передачи данных, сетевые платы и т. д.

Функции остальных уровней реализуются в виде программных модулей – драйверов.

Модель взаимодействия для ЛВС. Для того, чтобы учесть требования физической передающей среды, используемой в ЛВС, была произведена некоторая модернизация семиуровневой модели взаимодействия открытых систем для локальных

Рис. 52. Эталонная модель для локальных компьютерных сетей

сетей (рис. 52). Необходимость такой модернизации была вызвана тем, что для организации взаимодействия абонентских ЭВМ в ЛВС используются специальные методы доступа к физической передающей среде. Верхние уровни модели ВОС не претерпели никаких изменений, а канальный уровень был разбит на два подуровня. Подуровень LLC (*logical link control*) обеспечивает управление логическим звеном, т. е. выполняет функции собственно канального уровня. Подуровень MAC (*media access control*) обеспечивает управление доступом к среде.

14.5. Протоколы компьютерной сети

Понятие протокола. Как было показано ранее, при обмене информацией в сети каждый уровень модели ВОС реагирует на свой заголовок. Иными словами, происходит взаимодействие между одноименными уровнями модели в различных абонентских ЭВМ. Такое взаимодействие должно выполняться по определенным правилам.

Протокол – набор правил, определяющий взаимодействие двух одноименных уровней модели взаимодействия открытых систем в различных абонентских ЭВМ.

Протокол – это не программа. Правила и последовательность выполнения действий при обмене информацией, определенные протоколом, должны быть реализованы в программе. Обычно функции протоколов различных уровней реализуются в драйверах для различных сетей.

В соответствии с семиуровневой структурой модели можно говорить о необходимости существования протоколов для каждого уровня.

Концепция открытых систем предусматривает разработку стандартов для протоколов различных уровней. Легче всего поддаются стандартизации протоколы трех нижних уровней модели

архитектуры открытых систем, так как они определяют действия и процедуры, свойственные для сетей любого класса.

Труднее всего стандартизовать протоколы верхних уровней, особенно прикладного, из-за множественности прикладных задач и в ряде случаев их уникальности. Если по типам структур, методам доступа к физической передающей среде, используемым сетевым технологиям и некоторым другим особенностям можно насчитать примерно десяток различных моделей сетей, то по их функциональному назначению пределов не существует.

Основные типы протоколов. Проще всего представить особенности сетевых протоколов на примере протоколов канального уровня, которые делятся на две основные группы: *байт-ориентированные* и *бит-ориентированные*.

Байт-ориентированный протокол обеспечивает передачу сообщения по информационному каналу в виде последовательности байт. Кроме информационных байт в канал передаются также управляющие и служебные байты. Такой тип протокола удобен для ЭВМ, так как она ориентирована на обработку данных, представленных в виде двоичных байт. Для коммуникационной среды байт-ориентированный протокол менее удобен, так как разделение информационного потока в канале на байты требует использования дополнительных сигналов, что, в конечном счете, снижает пропускную способность канала связи.

Наиболее известным и распространенным байт-ориентированным протоколом является протокол двоичной синхронной связи BSC (*binary synchronous communication*), разработанный фирмой IBM. Протокол обеспечивает передачу двух типов кадров: управляющих и информационных. В управляющих кадрах передаются управляющие и служебные символы, а в информационных – сообщения (отдельные пакеты, последовательность пакетов). Работа протокола BSC осуществляется в три фазы: уста-

новление соединения, поддержание сеанса передачи сообщений, разрыв соединения. Протокол требует на каждый переданный канал посылки квитанции о результате его приема. Кадры, переданные с ошибкой, передаются повторно. Протокол определяет максимальное число повторных передач.

Квитанция представляет собой управляющий кадр, в котором содержится подтверждение приема сообщения (положительная квитанция) или отказ приема из-за ошибки (отрицательная квитанция). Передача последующего кадра возможна только тогда, когда получена положительная квитанция на прием предыдущего. Это существенно ограничивает быстродействие протокола и предъявляет высокие требования к качеству канала связи.

Бит-ориентированный протокол предусматривает передачу информации в виде потока бит, не разделяемых на байты. Поэтому для разделения кадров используются специальные последовательности – флаги. В начале кадра ставится флаг открывающий, а в конце – флаг закрывающий.

Бит-ориентированный протокол удобен относительно коммуникационной среды, так как канал связи как раз и ориентирован на передачу последовательности бит. Для ЭВМ он не очень удобен, потому что из поступающей последовательности бит приходится выделять байты для последующей обработки сообщения. Впрочем, учитывая быстродействие ЭВМ, можно считать, что эта операция не окажет существенного влияния на ее производительность. Бит-ориентированные протоколы являются более скоростными по сравнению с байт-ориентированными протоколами, что обусловливает их широкое распространение в современных сетях.

Типичным представителем группы бит-ориентированных протоколов являются протокол HDLC (*High-level Data Link Control* – высший уровень управления каналом связи) и его под-

множества. Протокол HDLC управляет информационным каналом с помощью специальных управляющих кадров, в которых передаются команды. Информационные кадры нумеруются. Кроме того, протокол HDLC позволяет без получения положительной квитанции передавать в канал до трех – пяти кадров. Положительная квитанция, полученная, например, на третий кадр, показывает, что два предыдущих приняты без ошибок, и необходимо повторить передачу только четвертого и пятого кадров. Такой алгоритм работы и обеспечивает высокое быстродействие протокола.

Из протоколов верхнего уровня модели ВОС следует отметить протокол X.400 (электронная почта) и FTAM (*File Transfer, Access and Management* – передача файлов, доступ к файлам и управление файлами).

Стандарты протоколов сетей. Для протоколов физического уровня стандарты определены рекомендациями МККТТ. Цифровая передача предусматривает использование протоколов X.21 и X.21-бис.

Канальный уровень определяют протокол HDLC и его подмножества, а также протокол X.25/3.

Широкое распространение локальных сетей потребовало разработки стандартов для этой области. В настоящее время для ЛВС используются стандарты, разработанные Институтом инженеров по электротехнике и радиоэлектронике – ИИЭР (*IEEE – Institute of Electrical and Electronics Engineers*).

Комитеты IEEE 802 разработали ряд стандартов, часть из которых принята МОС и другими организациями. Для ЛВС разработаны следующие стандарты:

– 802.1 – верхние уровни и административное управление;

– 802.2 – управление логическим звеном данных (LLC);

– 802.3 – случайный метод доступа к среде (CSMA/CD – *carrier-sense multiple access with collision detection* – множествен-

ный доступ с контролем передачи и обнаружением столкновений);

– 802.4 – маркерная шина;
– 802.5 – маркерное кольцо;
– 802.6 – городские сети.

Обмен информацией между одноименными уровнями определяется протоколами, речь о которых шла выше.

14.6. Локальные сети

Особенности организации ЛВС. Функциональные группы устройств в сети. Основное назначение любой компьютерной сети – предоставление информационных и вычислительных ресурсов подключенным к ней пользователям.

С этой точки зрения локальную сеть можно рассматривать как совокупность серверов и рабочих станций.

Сервер – компьютер, подключенный к сети и обеспечивающий ее пользователей определенными услугами.

Серверы могут осуществлять хранение данных, управление базами данных, удаленную обработку заданий, печать заданий и ряд других функций, потребность в которых может возникнуть у пользователей сети. Сервер – источник ресурсов сети.

Рабочая станция – персональный компьютер, подключенный к сети, через который пользователь получает доступ к ее ресурсам.

Рабочая станция сети функционирует как в сетевом, так и в локальном режиме. Она оснащена собственной операционной системой, обеспечивает пользователя всеми необходимыми инструментами для решения прикладных задач.

Особое внимание следует уделить одному из типов серверов – файловому серверу (*file server*). В распространенной терминологии для него принято сокращенное название – файл-сервер.

Файл-сервер хранит данные пользователей сети и обеспечивает им доступ к этим данным. Это компьютер с большой емкостью оперативной памяти, жесткими дисками большой емкости.

Он работает под управлением специальной операционной системы, которая обеспечивает одновременный доступ пользователей сети к расположенным на нем данным.

Файл-сервер выполняет следующие функции: хранение данных, архивирование данных, синхронизацию изменений данных различными пользователями, передачу данных.

Для многих задач использование одного файл-сервера оказывается недостаточным. Тогда в сеть могут включаться несколько серверов. Возможно также применение в качестве файл-серверов мини-ЭВМ.

Управление взаимодействием устройств в сети. Информационные системы, построенные на базе компьютерных сетей, обеспечивают решение следующих задач: хранение данных, обработка данных, организация доступа пользователей к данным, передача данных и результатов обработки данных пользователям.

Компьютерные сети реализуют распределенную обработку данных. Обработка данных в этом случае распределена между двумя объектами: клиентом и сервером.

Клиент – задача, рабочая станция или пользователь компьютерной сети.

В процессе обработки данных клиент может сформировать запрос на сервер для выполнения сложных процедур, чтение файла, поиск информации в базе данных и т. д.

Сервер, определенный ранее, выполняет запрос, поступивший от клиента. Результаты выполнения запроса передаются клиенту. Сервер обеспечивает хранение данных общего пользования, организует доступ к этим данным и передает данные клиенту.

Клиент обрабатывает полученные данные и представляет результаты обработки в виде, удобном для пользователя. В принципе, обработка данных может быть выполнена и на сервере. Для подобных систем приняты термины – системы клиент-сервер или архитектура клиент-сервер.

Архитектура клиент-сервер может использоваться как в одноранговых локальных сетях, так и в сети с выделенным сервером.

Одноранговая сеть. В такой сети нет единого центра управления взаимодействием рабочих станций и нет единого устройства для хранения данных. Сетевая операционная система распределена по всем рабочим станциям. Каждая станция сети может выполнять функции, как клиента, так и сервера. Она может обслуживать запросы от других рабочих станций и направлять свои запросы на обслуживание в сеть.

Пользователю сети доступны все устройства, подключенные к другим станциям (диски, принтеры).

Достоинства одноранговых сетей: низкая стоимость и высокая надежность.

Недостатки одноранговых сетей:

– зависимость эффективности работы сети от количества станций;

– сложность управления сетью;

– сложность обеспечения защиты информации;

– трудности обновления и изменения программного обеспечения станций.

Наибольшей популярностью пользуются одноранговые сети на базе сетевых операционных систем LANtastic, NetWare Lite.

Сеть с выделенным сервером. В сети с выделенным сервером один из компьютеров выполняет функции хранения данных, предназначенных для использования всеми рабочими станция-

ми, управления взаимодействием между рабочими станциями и ряд сервисных функций.

Такой компьютер обычно называют сервером сети. На нем устанавливается сетевая операционная система, к нему подключаются все разделяемые внешние устройства – жесткие диски, принтеры и модемы.

Взаимодействие между рабочими станциями в сети, как правило, осуществляется через сервер. Логическая организация такой сети может быть представлена топологией звезда. Роль центрального устройства выполняет сервер. В сетях с централизованным управлением существует также возможность обмена информацией между рабочими станциями, минуя файл-сервер.

Достоинства сети с выделенным сервером:

– надежная система защиты информации;

– высокое быстродействие;

– отсутствие ограничений на число рабочих станций;

– простота управления по сравнению с одноранговыми сетями.

Недостатки сети:

– высокая стоимость из-за выделения одного компьютера под сервер;

– зависимость быстродействия и надежности сети от сервера;

– меньшая гибкость по сравнению с одноранговой сетью.

Сети с выделенным сервером являются наиболее распространенными у пользователей компьютерных сетей.

Типовые топологии и методы доступа ЛВС. Физическая передающая среда ЛВС. Физическая среда обеспечивает перенос информации между абонентами сети. Как уже упоминалось, физическая передающая среда ЛВС представлена тремя типами кабелей: *витая пара проводов*, *коаксиальный кабель*, *оптоволоконный кабель*.

Витая пара состоит из двух изолированных проводов, свитых между собой (рис. 53). Скручивание проводов уменьшает влияние внешних электромагнитных полей на передаваемые сигналы. Самый простой вариант витой пары – телефонный кабель. Витые пары имеют различные характеристики, определяемые размерами, изоляцией и шагом скручивания. Дешевизна этого вида передающей среды делает ее достаточно популярной для ЛВС.

Рис. 53. Витая пара проводов

Основной недостаток витой пары – плохая помехозащищенность. Технологические усовершенствования позволяют повысить скорость передачи и помехозащищенность (экранированная витая пара), но при этом возрастает стоимость этого типа передающей среды.

Коаксиальный кабель (рис. 54) по сравнению с витой парой обладает более высокой механической прочностью, помехозащищенностью и обеспечивает скорость передачи информации до 10–50 Мбит/с. Для промышленного использования выпускаются два типа коаксиальных кабелей: толстый и тонкий. Толстый кабель более прочен и передает сигналы нужной амплитуды на большее расстояние, чем тонкий. В то же время тонкий кабель значительно дешевле. Коаксиальный кабель так же, как и витая

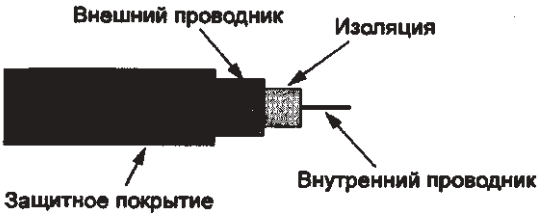


Рис. 54. Коаксиальный кабель

пара, является одним из популярных типов передающей среды для ЛВС.

Оптоволоконный кабель – идеальная передающая среда (рис. 55). Он не подвержен действию электромагнитных полей и сам практически не имеет излучения. Последнее свойство позволяет использовать его в сетях, требующих повышенной секретности информации.

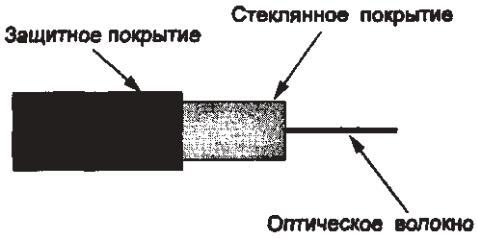


Рис. 55. Оптоволоконный кабель

Скорость передачи информации по оптоволоконному кабелю более 50 Мбит/с (около 1 Гбит/с в настоящее время). По сравнению с предыдущими типами передающей среды он дороже, менее технологичен в эксплуатации.

ЛВС, выпускаемые различными фирмами, либо рассчитаны на один из типов передающей среды, либо могут быть реализованы в различных вариантах на базе различных передающих сред.

Основные топологии ЛВС. Вычислительные машины, входящие в состав ЛВС, могут быть расположены самым случайным образом на территории, где создается сеть. Следует заметить, что для способа обращения к передающей среде и методов управления сетью небезразлично, как расположены абонентские ЭВМ. Поэтому имеет смысл говорить о топологии ЛВС.

Топология ЛВС – это усредненная геометрическая схема соединений узлов сети.

Топологии сетей могут быть самыми различными, но для локальных сетей типичными являются всего три: *кольцевая*, *шинная*, *звездообразная*.

Иногда для упрощения используют термины – кольцо, шина и звезда. Не следует думать, что рассматриваемые типы топологий представляют собой идеальное кольцо, идеальную прямую или звезду.

Любую компьютерную сеть можно рассматривать как совокупность узлов.

Узел – любое устройство, непосредственно подключенное к передающей среде сети.

Топология усредняет схему соединений узлов сети. Так, и эллипс, и замкнутая кривая, и замкнутая ломаная линия относятся к кольцевой топологии, а незамкнутая ломаная линия – к шинной.

Кольцевая топология предусматривает соединение узлов сети замкнутой кривой – кабелем передающей среды (рис. 56). Выход одного узла сети соединяется с входом другого. Информация по кольцу передается от узла к узлу. Каждый промежуточный узел между передатчиком и приемником ретранслирует посланное сообщение. Принимающий узел распознает и получает только адресованные ему сообщения.

Кольцевая топология является идеальной для сетей, занимающих сравнительно небольшое пространство. В ней отсутствует центральный узел, что повышает надежность сети. Ретрансля-

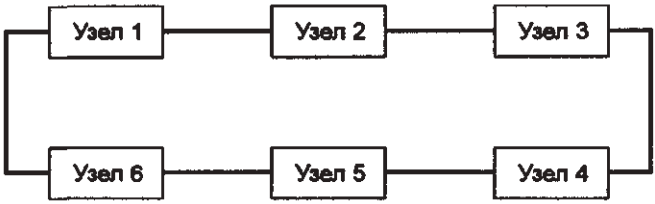


Рис. 56. Сеть кольцевой топологии

ция информации позволяет использовать в качестве передающей среды любые типы кабелей.

Последовательная дисциплина обслуживания узлов такой сети снижает ее быстродействие, а выход из строя одного из узлов нарушает целостность кольца и требует принятия специальных мер для сохранения тракта передачи информации.

Шинная топология – одна из наиболее простых (рис. 57). Она связана с использованием в качестве передающей среды коаксиального кабеля. Данные от передающего узла сети распространяются по шине в обе стороны. Промежуточные узлы не транслируют поступающих сообщений. Информация поступает на все узлы, но принимает сообщение только тот, которому оно адресовано. Дисциплина обслуживания параллельная.

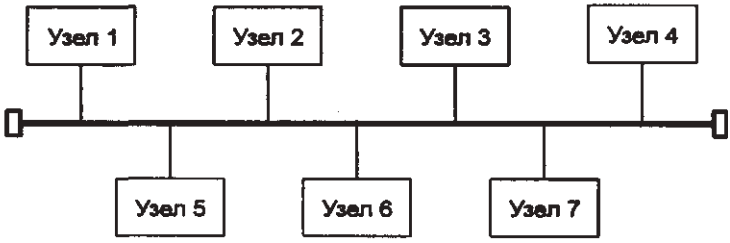


Рис. 57. Сеть шинной топологии

Это обеспечивает высокое быстродействие ЛВС с шинной топологией. Сеть легко наращивать и конфигурировать, а также адаптировать к различным системам. Сеть шинной топологии устойчива к возможным неисправностям отдельных узлов.

Сети шинной топологии наиболее распространены в настоящее время. Следует отметить, что они имеют малую протяженность и не позволяют использовать различные типы кабеля в пределах одной сети.

Звездообразная топология (рис. 58) базируется на концепции центрального узла, к которому подключаются периферий-

Рис. 58. Сеть звездообразной топологии

ные узлы. Каждый периферийный узел имеет свою отдельную линию связи с центральным узлом. Вся информация передается через центральный узел, который ретранслирует, переключает и маршрутизирует информационные потоки в сети.

Звездообразная топология значительно упрощает взаимодействие узлов ЛВС друг с другом, позволяет использовать более простые сетевые адаптеры. В то же время работоспособность ЛВС со звездообразной топологией целиком зависит от центрального узла.

В реальных сетях могут использоваться более сложные топологии, представляющие в некоторых случаях сочетания рассмотренных.

Выбор той или иной топологии определяется областью применения ЛВС, географическим расположением ее узлов и размерностью сети в целом.

Методы доступа к передающей среде. Передающая среда является общим ресурсом для всех узлов сети. Чтобы получить возможность доступа к этому ресурсу из узла сети, необходимы специальные механизмы – методы доступа.

Метод доступа к передающей среде – метод, обеспечивающий выполнение совокупности правил, по которым узлы сети получают доступ к ресурсу.

Существуют два основных класса методов доступа: *детерминированные*, *недетерминированные*.

При детерминированных методах доступа передающая среда распределяется между узлами с помощью специального механизма управления, гарантирующего передачу данных узла в течение некоторого, достаточно малого интервала времени.

Наиболее распространенными детерминированными методами доступа являются метод опроса и метод передачи права. Метод опроса рассматривался ранее. Он используется преимущественно в сетях звездообразной топологии.

Метод передачи права применяется в сетях с кольцевой топологией. Он основан на передаче по сети специального сообщения – маркера.

Маркер – служебное сообщение определенного формата, в которое абоненты сети могут помещать свои информационные пакеты.

Маркер циркулирует по кольцу, и любой узел, имеющий данные для передачи, помещает их в свободный маркер, устанавливает признак занятости маркера и передает его по кольцу. Узел, которому было адресовано сообщение, принимает его, устанавливает признак подтверждения приема информации и отправляет маркер в кольцо.

Передающий узел, получив подтверждение, освобождает маркер и отправляет его в сеть. Существуют методы доступа, использующие несколько маркеров.

Недетерминированные – случайные методы доступа предусматривают конкуренцию между всеми узлами сети за право передачи. Возможны одновременные попытки передачи со стороны нескольких узлов, в результате чего возникают коллизии.

Наиболее распространенным недетерминированным методом доступа является множественный метод доступа с контролем несущей частоты и обнаружением коллизий (CSMA/CD). В сущности, это описанный ранее режим соперничества. Кон-

троль несущей частоты заключается в том, что узел, желающий передать сообщение, «прослушивает» передающую среду, ожидая ее освобождения. Если среда свободна, узел начинает передачу.

Следует отметить, что топология сети, метод доступа к передающей среде и метод передачи тесным образом связаны друг с другом. Определяющим компонентом является топология сети.

Назначение ЛВС. Локальные сети за последнее время получили широкое распространение в самых различных областях науки, техники и производства.

Особенно широко ЛВС применяются при разработке коллективных проектов, например, сложных программных комплексов. На базе ЛВС можно создавать автоматизированные информационные системы, а также реализовывать новые информационные технологии.

В учебных лабораториях университетов ЛВС позволяют повысить качество обучения и внедрять современные интеллектуальные технологии обучения.

Объединение ЛВС. Причины объединения ЛВС. Созданная на определенном этапе развития системы ЛВС с течением времени перестает удовлетворять потребности всех пользователей, и тогда встает проблема расширения ее функциональных возможностей. Может возникнуть необходимость объединения внутри организации различных ЛВС, появившихся в различных ее отделах и филиалах в разное время, хотя бы для организации обмена данными с другими системами. Проблема расширения конфигурации сети может быть решена как в пределах ограниченного пространства, так и с выходом во внешнюю среду.

Стремление получить выход на определенные информационные ресурсы может потребовать подключения ЛВС к сетям более высокого уровня.

В самом простом варианте объединение ЛВС необходимо для расширения сети в целом, но технические возможности существующей сети исчерпаны, новых абонентов подключить к ней нельзя. Можно только создать еще одну ЛВС и объединить ее с уже существующей, воспользовавшись одним из нижеперечисленных способов.

Способы объединения ЛВС. Мост. Самый простой вариант объединения ЛВС – объединение одинаковых сетей в пределах ограниченного пространства. Физическая передающая среда накладывает ограничения на длину сетевого кабеля. В пределах допустимой длины строится отрезок сети – сетевой сегмент. Для объединения сетевых сегментов *используются мосты*.

Мост – устройство, соединяющее две сети, использующие одинаковые методы передачи данных.

Сети, которые объединяет мост, должны иметь одинаковые сетевые уровни модели взаимодействия открытых систем, нижние уровни могут иметь некоторые отличия.

Для сети персональных компьютеров мост – отдельная ЭВМ со специальным программным обеспечением и дополнительной аппаратурой. Мост может соединять сети разных топологий, но работающие под управлением однотипных сетевых операционных систем.

Мосты могут быть *локальными* и *удаленными*.

Локальные мосты соединяют сети, расположенные на ограниченной территории в пределах уже существующей системы.

Удаленные мосты соединяют сети, разнесенные территориально, с использованием внешних каналов связи и модемов.

Локальные мосты, в свою очередь, разделяются на *внутренние* и *внешние*.

Внутренние мосты обычно располагаются на одной из ЭВМ данной сети и совмещают функцию моста с функцией абонент-

ской ЭВМ. Расширение функций осуществляется путем установки дополнительной сетевой платы.

Внешние мосты предусматривают использование для выполнения своих функций отдельной ЭВМ со специальным программным обеспечением.

Маршрутизатор (роутер). Сеть сложной конфигурации, представляющая собой соединение нескольких сетей, нуждается в специальном устройстве. Задача этого устройства – отправить сообщение адресату в нужную сеть. Называется такое устройство *маршрутизатором.*

Маршрутизатор или *роутер* – устройство, соединяющее сети разного типа, но использующее одну операционную систему.

Маршрутизатор выполняет свои функции на сетевом уровне, поэтому он зависит от протоколов обмена данными, но не зависит от типа сети. С помощью двух адресов – адреса сети и адреса узла маршрутизатор однозначно выбирает определенную станцию сети.

Пример. Необходимо установить связь с абонентом телефонной сети, находящимся в другом городе. Сначала набирается адрес телефонной сети этого города – код города. Затем – адрес узла этой сети – телефонный номер абонента. Функции маршрутизатора выполняет аппаратура АТС.

Маршрутизатор также может выбрать наилучший путь для передачи сообщения абоненту сети, фильтрует информацию, проходящую через него, направляя в одну из сетей только ту информацию, которая ей адресована.

Кроме того, маршрутизатор обеспечивает балансировку нагрузки в сети, перенаправляя потоки сообщений по свободным каналам связи.

Шлюз. Для объединения ЛВС различных типов, работающих по существенно отличающимся друг от друга протоколам, предусмотрены специальные устройства – шлюзы.

Шлюз – устройство, позволяющее организовать обмен данными между двумя сетями, использующими различные протоколы взаимодействия.

Шлюз осуществляет свои функции на уровнях выше сетевого. Он не зависит от используемой передающей среды, но зависит от используемых протоколов обмена данными. Обычно шлюз выполняет преобразование между двумя протоколами.

С помощью шлюзов можно подключить локальную сеть к главному компьютеру, а также локальную сеть подключить к глобальной.

Пример. Необходимо объединить локальные сети, находящиеся в разных городах. Эту задачу можно решить с помощью глобальной сети передачи данных. Такой сетью является сеть коммутации пакетов на базе протокола X.25. С помощью шлюза локальная сеть подключается к сети X.25. Шлюз выполняет необходимые преобразования протоколов и обеспечивает обмен данными между сетями.

Мосты, маршрутизаторы и даже шлюзы конструктивно выполняются в виде плат, которые устанавливаются в компьютерах. Функции свои они могут выполнять как в режиме полного выделения функций, так и в режиме совмещения их с функциями рабочей станции сети.

14.7. Глобальная сеть Интернет

Представление о структуре и системе адресации. Структура Интернет. Интернет представляет собой глобальную компьютерную сеть. Само ее название означает «между сетей». Это сеть, соединяющая отдельные сети.

Логическая структура Интернет представляет собой некое виртуальное объединение, имеющее свое собственное информационное пространство.

Интернет обеспечивает обмен информацией между всеми компьютерами, которые входят в сети, подключенные к ней. Тип компьютера и используемая им операционная система значения не имеют. Соединение сетей обладает громадными возможностями. С собственного компьютера любой абонент Интернет может передавать сообщения в другой город, просматривать каталог библиотеки Конгресса в Вашингтоне, знакомиться с картинами на последней выставке в музее Метрополитен в Нью-Йорке, участвовать в конференции IEEE и даже в играх с абонентами сети из разных стран. Интернет предоставляет в распоряжение своих пользователей множество всевозможных ресурсов.

Основные ячейки Интернет – локальные сети. Это значит, что Интернет не просто устанавливает связь между отдельными компьютерами, а создает пути соединения для более крупных единиц – групп компьютеров. Если некоторая локальная сеть непосредственно подключена к Интернет, то каждая рабочая станция этой сети также может подключаться к Интернет.

Рис. 59. Подключение локальной сети к Интернет

Существуют также компьютеры, самостоятельно подключенные к Интернет. Они называются хост-компьютерами (host – хозяин). Каждый подключенный к сети компьютер имеет свой адрес, по которому его может найти абонент из любой точки света.

Схема подключения локальной сети к Интернет приведена на рис. 59.

Важной особенностью Интернет является то, что она, объединяя различные сети, не создает при этом никакой иерархии – все компьютеры, подключенные к сети, равноправны. Для иллюстрации возможной структуры некоторого участка сети Интернет приведена схема соединения различных сетей (рис. 60).

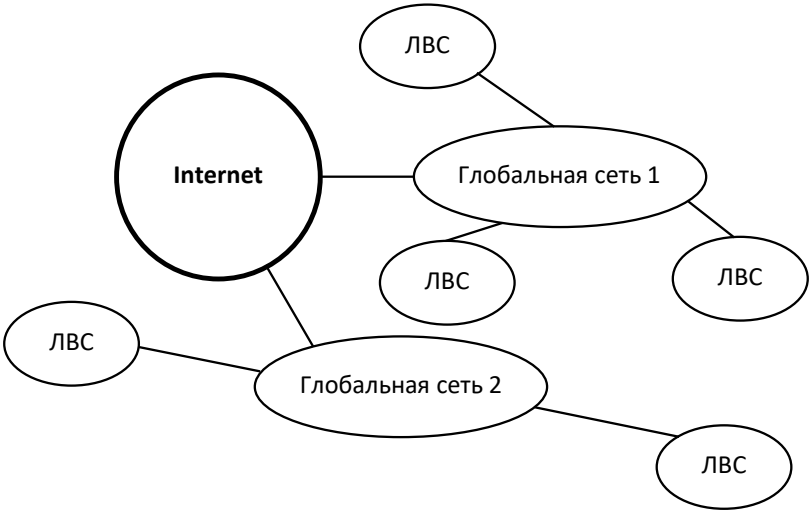


Рис. 60. Подключение различных сетей к Интернет

Система адресации в Интернет. Интернет самостоятельно осуществляет передачу данных. К адресам станций предъявляются специальные требования. Адрес должен иметь формат, позволяющий вести его обработку автоматически, и должен нести некоторую информацию о своем владельце.

С этой целью для каждого компьютера устанавливаются два адреса: цифровой IP-адрес (*IP – Internet Protocol* – межсетевой протокол) и доменный адрес.

Оба эти адреса могут применяться равноценно. Цифровой адрес удобен для обработки на компьютере, а доменный адрес – для восприятия пользователем.

Цифровой адрес имеет длину 32 бита. Для удобства он разделяется на четыре блока по 8 бит, которые можно записать в десятичном виде. Адрес содержит полную информацию, необходимую для идентификации компьютера.

Два блока определяют адрес сети, а два другие – адрес компьютера внутри этой сети. Существует определенное правило для установления границы между этими адресами. Поэтому IP-адрес включает в себя три компонента: адрес сети, адрес подсети, адрес компьютера в подсети.

Пример. В десятичном коде цифровой адрес записывается следующим образом: 192.45.9.200. Адрес сети – 192.45; адрес подсети – 9; адрес компьютера в подсети – 200.

Доменный адрес определяет область, представляющую ряд хост-компьютеров. В отличие от цифрового адреса он читается в обратном порядке. Вначале идет имя компьютера, затем имя сети, в которой он находится.

Чтобы абонентам Интернет можно было достаточно просто связаться друг с другом, все пространство адресов разделяется на области – домены. Возможно также разделение по определенным признакам и внутри доменов.

В системе адресов Интернет приняты домены, представленные географическими регионами. Они имеют имя, состоящее из двух букв.

Пример. Географические домены некоторых стран: Франция – fr; Канада – ca; США – us; Россия – ru. Существуют и до-

мены, разделенные по тематическим признакам. Такие домены имеют трехбуквенное сокращенное название. Учебные заведения – edu. Правительственные учреждения – gov. Коммерческие организации – com.

Компьютерное имя включает, как минимум, два уровня доменов. Каждый уровень отделяется от другого точкой. Слева от домена верхнего уровня располагаются другие имена. Все имена, находящиеся слева, – поддомены для общего домена.

Пример. Существует имя tutor.sptu.edu. Здесь edu – общий домен для школ и университетов. Tutor – поддомен sptu, который является поддоменом edu.

Для пользователей интернет-адресами могут быть просто их регистрационные имена на компьютере, подключенном к сети. За именем следует знак @. Все это слева присоединяется к имени компьютера.

Пример. Пользователь, зарегистрировавшийся под именем victor на компьютере, имеющем в Интернет имя tutor.sptu.edu, будет иметь адрес: victor@tutor. sptu.edu.

В Интернет могут использоваться не только имена отдельных людей, но и имена групп. Для обработки пути поиска в доменах имеются специальные серверы имен. Они преобразовывают доменное имя в соответствующий цифровой адрес.

Локальный сервер передает запрос на глобальный сервер, имеющий связь с другими локальными серверами имен. Поэтому пользователю просто нет никакой необходимости знать цифровые адреса.

Для выхода в Интернет необходимо знать адрес домена, с которым устанавливается связь.

Способы организации передачи информации. Электронная почта. Электронная почта (англ. *e-mail – electronic mail*) выполняет функции обычной почты. Она обеспечивает передачу

сообщений из одного пункта в другой. Главным ее преимуществом является независимость от времени. Электронное письмо приходит сразу же после его отправления и хранится в почтовом ящике до получения адресатом. Кроме текста оно может содержать графические и звуковые файлы, а также двоичные файлы – программы.

Электронные письма могут отправляться сразу по нескольким адресам. Пользователь Интернет с помощью электронной почты получает доступ к различным услугам сети, так как основные сервисные программы Интернет имеют интерфейс с ней. Суть такого подхода заключается в том, что на хост-компьютер отправляется запрос в виде электронного письма. Текст письма содержит набор стандартных формулировок, которые и обеспечивают доступ к нужным функциям. Такое сообщение воспринимается компьютером как команда и выполняется им.

Для работы с электронной почтой создано большое количество программ. Их можно объединить под обобщающим названием mail. Эти программы выполняют следующие функции:

– подготовку текста;
– чтение и сохранение корреспонденции;
– удаление корреспонденции;
– ввод адреса;
– комментирование и пересылку корреспонденции;
– импорт (прием и преобразование в нужный формат) других файлов.

Сообщения можно обрабатывать собственным текстовым редактором программы электронной почты. Из-за ограниченности его возможностей обработку текстов большого размера лучше выполнять внешним редактором. При отправке такого текста программа электронной почты дает возможность его обработать.

Обычно программы электронной почты пересылают тексты в коде ASCII и в двоичном формате. Код ASCII позволяет записывать только текст и не дает возможности передавать информацию об особенностях национальных шрифтов.

В двоичных файлах сохраняется любая информация. Поэтому для передачи комбинированных сообщений (графика и текст), а также для передачи программ используются двоичные файлы.

При отправлении сообщений по электронной почте необходимо указывать в адресе не только имя хост-компьютера, но и имя абонента, которому сообщение предназначено.

Формат адреса электронной почты должен иметь вид:

имя пользователя@адрес хост-компьютера

Для каждого пользователя на одном хост-компьютере может быть заведен свой каталог для получения сообщений по электронной почте.

Специальный стандарт MIME (*Multipurpose Internet Mail Extension*) – многоцелевое расширение почты Интернет – позволяет вкладывать в символьные сообщения любые двоичные файлы, включая графику, аудио- и видеофайлы.

Пользователь, имеющий выход в Интернет, может также отправлять электронную почту и по адресам других сетей, подключенных к ней с помощью шлюзов.

В этом случае необходимо учитывать, что различные сети применяют различную адресацию пользователей. Отправляя сообщение по электронной почте в другую сеть, следует использовать принятую там систему адресов.

World Wide Web (Всемирная информационная сеть). WWW является одной из самых популярных информационных служб Интернет. Две основные особенности отличают WWW: использование гипертекста и возможность клиентов взаимодействовать с другими приложениями Интернет.

Гипертекст – текст, содержащий в себе связи с другими текстами, графической, видео- или звуковой информацией.

Внутри гипертекстового документа некоторые фрагменты текста четко выделены. Указание на них с помощью, например, мыши позволяет перейти на другую часть этого же документа, на другой документ в этом же компьютере или даже на документы на любом другом компьютере, подключенном к Интернет.

Все серверы WWW используют специальный язык HTML (*Hypertext Markup Language* – язык разметки гипертекста). HTML-документы представляют собой текстовые файлы, в которые встроены специальные команды.

WWW обеспечивает доступ к сети как клиентам, требующим только текстовый режим, так и клиентам, предпочитающим работу в режиме графики. Отображенный на экране гипертекст представляет собой сочетание алфавитно-цифровой информации в различных форматах и стилях и некоторые графические изображения – картинки.

Связь между гипертекстовыми документами осуществляется с помощью ключевых слов. Найдя ключевое слово, пользователь может перейти в другой документ, чтобы получить дополнительную информацию. Новый документ также будет иметь гипертекстовые ссылки.

Работать с гипертекстами предпочтительнее на рабочей станции клиента, подключенной к одному из Web-серверов, чем на страницах учебника, поэтому изложенный материал можно считать первым шагом к познанию службы WWW.

Работая с Web-сервером, можно выполнить удаленное подключение Telnet, послать абонентам сети электронную почту, получить файлы с помощью FTP-анонима и выполнить ряд других приложений (прикладных программ) Интернет. Это дает возможность считать WWW интегральной службой Интернет.

Создание страниц WWW. Так как создание собственного сервера WWW является сложным и дорогостоящим, то многие пользователи сети Интернет могут размещать свою информацию на уже существующих серверах. Собственные страницы WWW можно создавать со специальных редакторов, реализующих набор макрокоманд, на базе которого создаются документы HTML.

В диалоговом режиме пользователь может создать свой документ. Редактор при этом обеспечивает:

– ввод заголовка документа;
– вставку графического изображения или видеофрагмента;
– вставку гипертекстовой ссылки;
– вставку закладки;
– просмотр страниц WWW.

Современные редакторы содержат средства для работы с языком JAVA. Этот язык позволяет интерпретировать программы, полученные из сети, на локальном компьютере пользователя. JAVA – язык объектно-ориентированного программирования. Он используется для передового способа создания приложений для Интернет – программирования аплетов (*аплет* – небольшое приложение). С помощью аплетов можно создавать динамичные веб-страницы.

Служба Gopher. Эта служба Интернет выполняет функции, аналогичные WWW. Вся информация на Gopher-сервере хранится в виде дерева данных (или иерархической системы меню). Начальный каталог Gopher является вершиной этого дерева, а все остальные каталоги и файлы представляются элементами меню. Строка главного меню представляет собой либо подменю, либо файл. Gopher поддерживает разные типы файлов – текстовые, звуковые, программные и др.

Телеконференции Usenet. Система Usenet была разработана для перемещения новостей между компьютерами по всему миру.

В дальнейшем она практически полностью интегрировалась в Интернет, и теперь Интернет обеспечивает распространение всех ее сообщений. Серверы Usenet имеют средства для разделения телеконференций по темам.

Телеконференции – дискуссионные группы, входящие в состав Usenet.

Телеконференции организованы по иерархическому принципу и для верхнего уровня выбраны семь основных рубрик. В свою очередь, каждая из них охватывает сотни подгрупп. Образуется древовидная структура, напоминающая организацию файловой системы. Из числа основных рубрик следует выделить: comp – темы, связанные с компьютерами; sci – темы из области научных исследований; news – информация и новости Usenet; soc – социальная тематика; talk – дискуссии.

Существуют, кроме того, специальные рубрики и региональное разделение телеконференций.

Управляют доступом к службе Usenet специальные программы, позволяющие выбирать телеконференции, работать с цепочками сообщений и читать сообщения и ответы на них. Эти программы выполняют такую функцию, как подписка на телеконференции. Если пользователь не вводит никаких ограничений, то по умолчанию производится подписка на все телеконференции, с которыми имеет связь его хост-компьютер. Программа также позволяет сделать тематический выбор и обеспечит пользователя сообщениями по интересующему его направлению.

При участии в какой-либо телеконференции любой абонент может направить свое сообщение по интересующей его теме.

Существуют два способа выполнения этой процедуры:

– посылка непосредственного ответа автору статьи по адресу его электронной почты;

– предоставление своего сообщения в распоряжение всех участников телеконференции.

Второй способ обозначается термином «Follow-up».

После электронной почты Usenet является самой популярной службой глобальной сети Интернет.

Передача файлов с помощью протокола FTP. Назначение электронной почты – это, прежде всего, обмен текстовой информацией между различными компьютерными системами. Не меньший интерес для пользователей сети Интернет представляет обмен отдельными файлами и целыми программами.

Для того, чтобы обеспечить перемещение данных между различными операционными системами, которые могут встретиться в Интернет, используется протокол FTP (*File Transfer Protocol*), работающий независимо от применяемого оборудования. Протокол обеспечивает способ перемещения файлов между двумя компьютерами и позволяет абоненту сети Интернет получить в свое распоряжение множество файлов. Пользователь получает доступ к различным файлам и программам, хранящимся на компьютерах, подключенных к сети.

Программа, реализующая этот протокол, позволяет установить связь с одним из множества FTP-серверов в Интернет.

FTP-сервер – компьютер, на котором содержатся файлы, предназначенные для открытого доступа.

Программа FTP-клиент не только реализует протокол передачи данных, но и поддерживает набор команд, которые используются для просмотра каталога FTP-сервера, поиска файлов и управления перемещением данных.

Для установки связи с FTP-сервером пользователь при работе должен ввести команду ftp, а затем адрес или его доменное имя.

Если связь установлена, появится приглашение ввести имя пользователя. Пользователь, не зарегистрированный на серве-

ре, может представиться именем «anonymus» и получит доступ к определенным файлам и программам. Если будет запрошен пароль, можно ввести свой адрес электронной почты. Поступившее после выполнения этих процедур приглашение позволяет работать с FTP-сервером.

Так как большинство FTP-серверов работает под управлением операционной системы Unix, то технология работы в этой системе требует введения команд из командной строки компьютера и несколько затрудняет действия пользователя в этом режиме.

Взаимодействие с другим компьютером (Telnet). Telnet обеспечивает взаимодействие с удаленным компьютером. Установив такую связь через Telnet, пользователь получает возможность работать с удаленным компьютером, как со «своим», т. е. теоретически получить в свое распоряжение все ресурсы, если к ним разрешен доступ. Реально Telnet предоставляет открытый доступ, но организация взаимодействия полностью определяется удаленным компьютером. Два вида услуг Интернет требуют подключения к серверам через Telnet: библиотечные каталоги и электронные доски объявлений (BBS).

Программа Telnet в использовании очень проста. Для установки с ее помощью связи с каким-либо компьютером, подключенным к сети, необходимо знать его полный адрес в Интернет. При установлении соединения с нужным компьютером следует указать в команде его адрес. В процессе соединения хост-компьютер запрашивает имя пользователя. Для работы в удаленной системе пользователь должен иметь там права доступа. После успешного подключения к хост-компьютеру пользователь должен указать тип используемого терминала. Для удобства работы пользователя хост-компьютер обычно указывает ему способ вызова справочной информации.

Работа с удаленной системой может вестись в «прозрачном» режиме, когда программы на сервере и у клиента только обеспечивают протокол соединения, и в командном, когда клиент получает в свое распоряжение набор команд сервера.

Следует заметить, что из соображений безопасности намечается тенденция сокращения числа узлов Интернет, позволяющих использовать Telnet для подключения к ним.

Электронные доски объявлений (BBS). Независимо от Интернет существуют маленькие диалоговые службы, предоставляющие доступ к BBS (*bulletin board system* – система электронных досок объявлений).

Это компьютеры, к которым можно подсоединиться с помощью модемов через телефонную сеть. BBS содержат файлы, которые можно переписывать, позволяют проводить дискуссии, участвовать в различных играх и имеют свою систему электронной почты.

Самой крупной и известной системой электронных досок объявлений является система CompuServe. Она насчитывает около двух миллионов пользователей. Для расширения своих возможностей CompuServe подключается к Интернет и предоставляет своим пользователям право доступа к службам Интернет.

Несмотря на относительную дешевизну обслуживания, ни одна из диалоговых систем BBS не может дать пользователям тех возможностей, которые предоставляет Интернет.

ГЛАВА 15
АРХИТЕКТУРА И СТРУКТУРА ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ

Создать высокопроизводительную ЭВМ по современной технологии на одном микропроцессоре не представляется возможным ввиду ограничения, обусловленного конечным значением скорости распространения электромагнитных волн (300 000 км/с), ибо время распространения сигнала на расстояние несколько миллиметров (линейный размер стороны МП) при быстродействии 100 млрд операций в секунду становится соизмеримым со временем выполнения одной операции. Поэтому ЭВМ создаются на базе параллельных многопроцессорных вычислительных систем (МПВС).

Параллельные МПВС имеют несколько разновидностей:

– магистральные (конвейерные) МПВС, в которых процессоры одновременно выполняют разные операции над последовательным потоком обрабатываемых данных; по принятой классификации такие МПВС относятся к системам с многократным потоком команд и однократным потоком данных (МКОД или *MISD – multiple instruction, single data*);

– векторные МПВС, в которых все процессоры одновременно выполняют одну команду над различными данными – однократный поток команд с многократным потоком данных (ОКМД или *SIMD – single instruction, multiple data*);

– матричные МПВС, в которых все микропроцессоры одновременно выполняют разные операции над несколькими последовательными потоками обрабатываемых данных – многократный поток команд с многократным потоком данных (МКМД или *MIMD – multiple instruction, multiple data*).

В ЭВМ используются все три варианта архитектуры МПВС:

– структура MIMD в классическом ее варианте (например, в суперкомпьютере BSP фирмы Burroughs);

– параллельно-конвейерная модификация, иначе, MMISD, т. е. многопроцессорная (*multiple*) MISD-архитектура (например, в суперкомпьютере «Эльбрус 3»);

– параллельно-векторная модификация, иначе, MSIMD, т. е. многопроцессорная SIMD-архитектура (например, в суперкомпьютере Cray 2).

Условные структуры однопроцессорной (SISD – *single instruction, single data*) и названных многопроцессорных вычислительных систем показаны на рис. 61.

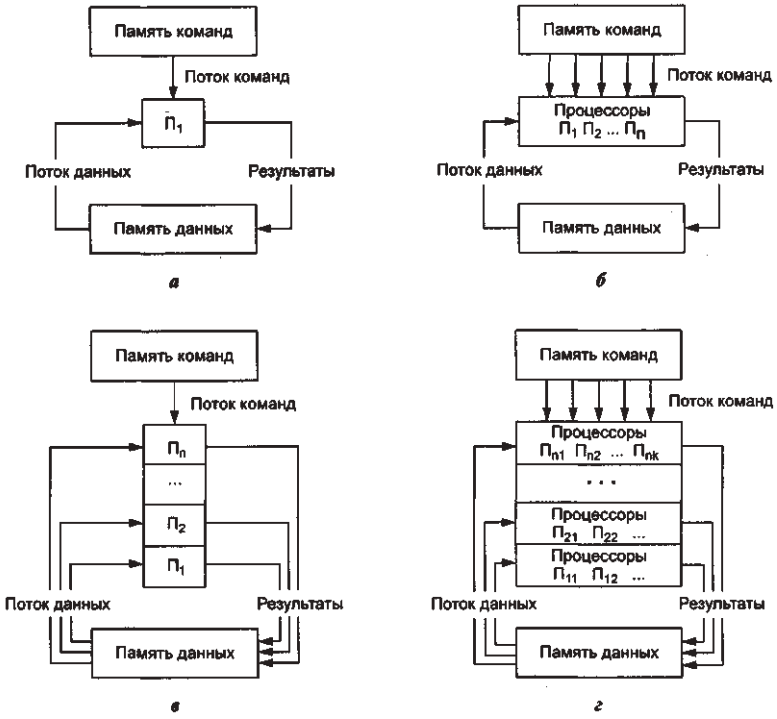


Рис. 61. Условные структуры вычислительных систем: а – SISD (однопроцессорная); б – MISD (конвейерная); в – SIMD (векторная); г – MIMD (матричная)

ЗАКЛЮЧЕНИЕ ПЕРСПЕКТИВЫ РАЗВИТИЯ ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ

Главной тенденцией развития вычислительной техники в настоящее время является дальнейшее расширение сфер применения компьютеров и, как следствие, переход от отдельных компьютеров к их системам – вычислительным системам и комплексам разнообразных конфигураций с широким диапазоном функциональных возможностей и характеристик.

Другой любопытной сферой развития вычислительной техники является конструирование современных самых разнообразных электронных безделушек: гаджетов и информеров.

Наиболее перспективные, создаваемые на основе персональных ЭВМ, территориально распределенные многомашинные вычислительные системы – вычислительные сети – ориентируются не столько на вычислительную обработку информации, сколько на коммуникационные информационные услуги: электронную почту, системы телеконференций и информационно-справочные системы.

Специалисты точно спрогнозировали, что в начале XXI в. в цивилизованных странах произойдет смена основной информационной среды.

Уже сегодня пользователям глобальной вычислительной сети Интернет стала доступной практически любая информация, находящаяся в хранилищах знаний этой сети, в частности, не относимая к конфиденциальным сведениям. Можно почитать или посмотреть, например, любую из нескольких сотен религиозных книг, рукописей или картин в библиотеке Ватикана, оформленных в виде файлов, послушать музыку в Карнеги-холл, «заглянуть» в галереи Лувра или в кабинет президента США в Белом

доме. Пользователи этой суперсети могут получить для изучения интересующую их статью или подборку статей по нужной тематике, «опубликовать» в сети свою новую работу, обсудить ее с заинтересованными специалистами.

В сети Интернет реализован принцип гипертекста, согласно которому абонент, выбирая встречающиеся в читаемом тексте ключевые слова, может получить необходимые дополнительные пояснения и материалы для углубления в изучаемую проблему. Используя этот принцип, абонент может прочитать электронную газету, персонифицированную на интересующую его тематику, с любой степенью подробности и достоверности. Электронная почта Интернет позволяет получить почтовое отправление из любой точки Земного шара (где есть терминалы этой сети) через 5 с, а не через неделю или месяц, как это имеет место при использовании обычной почты.

В Массачусетском университете (США) создана электронная книга, куда можно записывать любую информацию из сети; читать эту книгу можно отключившись от сети, автономно, в любом месте. Сама книга в твердом переплете, содержит тонкие жидкокристаллические индикаторы – страницы с бумагообразной синтетической поверхностью и высоким качеством «печати».

При разработке и создании собственно ЭВМ существенный и устойчивый приоритет в последние годы имеют сверхмощные компьютеры – суперЭВМ и миниатюрные, и сверхминиатюрные. Уже полным ходом ведутся поисковые работы по созданию компьютеров 6-го поколения и появляются образцы, базирующиеся на распределенной нейронной архитектуре, – нейрокомпьютеры. В частности, в нейрокомпьютерах могут использоваться уже имеющиеся специализированные сетевые микропроцессоры – транспьютеры – микропроцессоры сети со встроенными средствами связи.

Повсеместное использование мультиканальных широкополосных радио-, волоконно-оптических, а в пределах прямой видимости и инфракрасных каналов обмена информацией между компьютерами обеспечивает практически неограниченную пропускную способность (трансферт до сотен миллионов байт в секунду).

Широкое внедрение средств мультимедиа, в первую очередь аудио- и видеосредств ввода и вывода информации, позволяет уже сейчас общаться с компьютером на естественном языке. Мультимедиа нельзя трактовать узко, только как мультимедиа на компьютере. Можно говорить о бытовом (домашнем) мультимедиа, включающем в себя и компьютер, и целую группу потребительских устройств (тех самых гаджетов и виджетов), доводящих потоки информации до потребителя и активно забирающих информацию у него.

Этому уже сейчас способствуют:

– развитые технологии медиасерверов, способных собирать и хранить огромнейшие объемы информации и выдавать ее в реальном времени по множеству одновременно приходящих запросов;

– системы сверхскоростных широкополосных информационных магистралей, связывающие воедино все потребительские системы.

Названные и уже высокоразвитые технологии и характеристики устройств компьютеров совместно с их общей миниатюризацией сделали всевозможные вычислительные средства и системы вездесущими, привычными, обыденными, естественно насыщающими нашу повседневную жизнь.

Специалисты точно предсказали возможность создания компьютерной модели реального мира, такой виртуальной системы, в которой мы можем активно жить и манипулиро-

вать виртуальными предметами. Простейший прообраз такого кажущегося мира уже сейчас существует в сложных компьютерных играх. И теперь говорят уже не об играх, а о виртуальной реальности в нашей повседневной жизни: когда нас в комнате, например, окружают сотни активных компьютерных устройств – автоматически включающиеся и выключающиеся по мере надобности, активно отслеживающие наше местоположение, постоянно снабжающие нас ситуационно необходимой информацией, активно воспринимающие нашу информацию и управляющие многими бытовыми приборами и устройствами.

Информационная революция затронула все стороны жизнедеятельности, появились системы, создающие виртуальную реальность:

– компьютерные системы с «дружественным интерфейсом», которые позволяют пользователям по видеоканалу видеть друг друга или даже виртуального собеседника, активно общаться на естественном речевом уровне с аудио- и видеоразъяснениями, советами, подсказками;

– системы автоматизированного обучения – при наличии обратной видеосвязи пользователь общается с персональным виртуальным учителем, учитывающим психологию, подготовленность, восприимчивость ученика;

– торговля – любой товар сопровождается не магнитным кодом, а электронным штрих-кодом или QR-кодом, нанесенным на торговый ярлык, дистанционно общающийся с потенциальным покупателем и сообщающий ему всю необходимую информацию – что, где, когда, как, сколько и почем. И так далее, и тому подобное.

Уже создано техническое обеспечение, необходимое для таких виртуальных систем:

– дешевые, простые, портативные компьютеры со средствами мультимедиа;

– программное обеспечение для «вездесущих» приложений;

– миниатюрные приемопередающие радиоустройства (трансиверы) для связи компьютеров друг с другом и с сетью;

– распределенные широкополосные каналы связи и сети.

Но есть и проблемы. Важнейшая из них – обеспечение прав интеллектуальной собственности и конфиденциальности информации, поскольку теперь уже личная жизнь каждого из нас стала всеобщим достоянием, что неприемлемо на любой стадии технического прогресса.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. SSD диски – принцип работы, скорость, надежность : статья // Storelab. – URL: http://storelab-rc.ru/ssd-review.htm.

2. Аксенова, А. А. Перспективы развития современных систем обработки информации / А. А. Аксенова // Моделирование, оптимизация и информационные технологии. – 2015. – № 4 (11). – 6 с.

3. Богданов, А. В. Архитектуры и топологии многопроцессорных вычислительных систем : учебник / А. В. Богданов, В. В. Корхов, В. В. Мареев и др. – М. : Интернет-университет информационных технологий (ИНТУИТ) ; Саратов : Ай Пи Ар Медиа, 2020. – 135 с. // Электронно-библиотечная система «IPR Books». – Режим доступа: для авторизированных пользователей.

4. Баранникова, И. В. Вычислительные машины, сети и системы: Функционально-структурная организация вычислительных систем : учебное пособие / И. В. Баранникова, А. Н. Гончаренко. – М. : МИСиС, 2017. – 103 с. // Электронно-библиотечная система «IPR Books». – Режим доступа: для авторизированных пользователей.

5. Батенькина, О. В. Программное и техническое обеспечение информационных систем : учебное пособие / О. В. Батенькина. – Омск : ОмГТУ, 2014. – 200 с.

6. Бекман, И. Н. Компьютеры в информатике : курс лекций / И. Н. Бекман // Docplayer. – URL: https://docplayer.ru/42095382-kompyutery-v-informatike-kurs-lekciy.html.

7. Берлин, А. Н. Телекоммуникационные сети и устройства : учебное пособие / А. Н. Берлин. – М. : Интернет-университет информационных технологий (ИНТУИТ) ; Саратов : Ай Пи Ар Медиа, 2020. – 395 с. // Электронно-библиотечная система

«IPR Books». – Режим доступа: для авторизированных пользователей.

8. Бройдо, В. Л. Вычислительные системы, сети и телекоммуникации : учебник для вузов / В. Л. Бройдо, О. П. Ильина. – 4-е изд. – СПб. : Питер, 2011. – 560 с.

9. Воройский, Ф. С. Информатика. Энциклопедический словарь-справочник: введение в современные информационные и телекоммуникационные технологии в терминах и фактах / Ф. С. Воройский. – М. : Физматлит, 2006. – 767 с.

10. Гольцман, В. И. Компьютер + мобильник: эффективное взаимодействие (+CD) / В. И. Гольцман. – СПб. : Питер, 2008. – 188 с.

11. Гуров, В. В. Архитектура и организация ЭВМ : учебный курс / В. В. Гуров, В. О. Чуканов. – М. : Интернет-университет информационных технологий, 2006. – 380 с.

12. Елесина, С. И. ЭВМ и периферийные устройства: Устройства ввода-вывода информации : учебник для вузов / С. И. Елесина, Е. Р. Муратов, М. Б. Никифоров. – М. : Курс, 2018. – 205 с.

13. Кандаурова, Н. В. Вычислительные системы, сети и телекоммуникации: (Курс лекций и лабораторный практикум) : учебное пособие / Н. В. Кандаурова [и др.]. – 2-е изд., стереотип. – М. : Флинта, 2013. – 339 с.

14. Колесниченко, О. В. Аппаратные средства PC / О. В. Колесниченко, И. В. Шишигин, В. Г. Соломенчук. – 6-е изд., перераб. и доп. – СПб. : БХВ-Петербург, 2010. – 800 с.

15. Колкер, А. Б. Управление исполнительными устройствами по USB в режиме реального времени / А. Б. Колкер, Н. О. Горбунов // Автоматика и программная инженерия. – 2013. – № 2 (4). – С. 48–59.

16. Касперски, К. Файловая система NTFS извне и изнутри / К. Касперски // Системный администратор. – 2004. – № 11 (24). – С. 72–77.

17. Куль, Т. П. Основы вычислительной техники : учебное пособие / Т. П. Куль. – Минск : Республиканский институт профессионального образования (РИПО), 2018. – 244 с. // Электронно-библиотечная система «IPR Books». – Режим доступа: для авторизированных пользователей.

18. Курячий, Г. В. Операционная система Linux : курс лекций : учебное пособие / Г. В. Курячий, К. А. Маслинский. – 2-е изд., испр. – М. : ALT Linux : ДМК Пресс, 2016. – 348 с.

19. Макаренко, С. И. Операционные системы, среды и оболочки : учебное пособие / С. И. Макаренко. – Ставрополь : СФ МГГУ имени М.А. Шолохова, 2008. – 210 с.

20. Макарова, Н. В. Информатика : учебник / Н. В. Макарова ; под ред. проф. Н. В. Макаровой. – 3-е изд., перераб. и доп. – М. : Финансы и статистика, 2009. – 768 с.

21. Назаров, С. В. Современные операционные системы : учебное пособие / С. В. Назаров, А. И. Широков. – М. : Интернет-университет информационных технологий (ИНТУИТ) ; Саратов : Ай Пи Ар Медиа, 2020. – 351 с. // Электронно-библиотечная система «IPR Books». – Режим доступа: для авторизированных пользователей.

22. Нужнов, Е. В. Операционные системы : учебное пособие : в 2 ч. / Е. В. Нужнов. – Ч. 1: Основы операционных систем. – Таганрог : Издательство Южного федерального университета, 2013. – 143 с.

23. Орлов, С. П. Организация вычислительных машин и систем / С. П. Орлов, Н. В. Ефимушкина. – [2-е изд., перераб. и доп.]. – Самара : Самарский государственный технический университет, 2016. – 280 с.

24. Архитектура персонального компьютера // Helpiks.org. – URL: https://helpiks.org/9-33878.html.

25. Партыка, Т. Л. Вычислительная техника : учебное пособие / Т. Л. Партыка, И. И. Попов. – 3-е изд., перераб. и доп. – М. : Форум : Инфра-М, 2020. – 445 с.

26. Пятибратов, А. П. Вычислительные системы, сети и телекоммуникации : учебное пособие / А. П. Пятибратов, Л. П. Гудыно, А. А. Кириченко ; под ред. А. П. Пятибратова. – М. : Кнорус, 2017. – 352 с.

27. Сафонов, В. О. Основы современных операционных систем : учебное пособие / В. О. Сафонов. – М. : Интернет-университет информационных технологий (ИНТУИТ) : Бином. Лаборатория знаний, 2011. – 584 с.

28. Семененко, В. А. Арифметико-логические основы компьютерной схемотехники : учебное пособие для студентов вузов, обучающихся по гуманитарным специальностям / В. А. Семененко, Э. К. Скуратович. – М. : Академический проект, 2004. – 140 с.

29. Семенов, Ю. А. Алгоритмы телекоммуникационных сетей : учебное пособие / Ю. А. Семенов. – Ч. 1: Алгоритмы и протоколы каналов и сетей передачи данных. – М. : Интернет-университет информационных технологий (ИНТУИТ) ; Саратов : Ай Пи Ар Медиа, 2020. – 757 с. // Электронно-библиотечная система «IPR Books». – Режим доступа: для авторизированных пользователей.

30. Соломенчук, В. Г. Железо ПК 2011 / В. Г. Соломенчук, П. В. Соломенчук. – СПб. : БХВ-Петербург, 2011. – 384 с.

31. Степина, В. В. Архитектура ЭВМ и вычислительные системы : учебник / В. В. Степина. – М. : Курс : Инфра-М, 2017. – 384 с.

32. Тихонов, В. А. Организация ЭВМ и систем / В. А. Тихонов, А. В. Баранов. – М. : Гелиос АРВ, 2008. – 384 с.

33. Хахаев, И. А. Вычислительные машины, сети и системы телекоммуникаций в таможенном деле : учебное пособие / И. А. Хахаев. – СПб. : Университет ИТМО, 2015. – 86 с.

34. Хлебников, А. А. Информационные технологии : учебник / А. А. Хлебников. – М. : Кнорус, 2016. – 466 с.

35. Цай, Д. В. Аппаратное обеспечение вычислительных систем : учебное пособие / Д. В. Цай, Р. Р. Файзрахманова. – М. : Фолиант, 2019. – 320 с.

36. Чекмарев, Ю. В. Вычислительные системы, сети и телекоммуникации / Ю. В. Чекмарев. – Саратов : Профобразование, 2019. – 184 с.

37. Чередов, А. Д. Организация ЭВМ и систем : учебное пособие / А. Д. Чередов, А. Н. Мальчуков. – 4-е изд., перераб. и доп. – Томск : Издательство Томского политехнического университета, 2016. – 236 с.

38. Черпаков, И. В. Теоретические основы информатики : учебник и практикум для академического бакалавриата / И. В. Черпаков. – М. : Юрайт, 2017. – 353 с.

Учебное пособие

Борзунов Константин Константинович,
кандидат технических наук

Пименова Оксана Владимировна,
кандидат технических наук

Лустин Владимир Иванович

Сребродольская Светлана Александровна

СРЕДСТВА ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ

Редактор *Чеботарева С. О.*
Корректор *Васильевых Е. М.*
Компьютерная верстка *Киселева М. Е.*

Московский университет МВД России имени В.Я. Кикотя
117997, г. Москва, ул. Академика Волгина, д. 12

Подписано в печать	Формат 60×84 1/16	Тираж 80 экз.
15.12.2020	Цена договорная	Объем 13,08 уч.-изд. л.
Заказ № 261		22,08 усл. печ. л.