### Оглавление

[Предисловие 7](#_TOC_250063)

Тема 1. Информатика - история становления 9

Тема 2. Информация 18

* 1. [Понятие информации 18](#_TOC_250062)
  2. [Свойства информации 20](#_TOC_250061)
  3. [Количество информации 22](#_TOC_250060)
  4. Информация, данные и знания - модель DIКW 25
  5. [Информация и данные как объект управления 31](#_TOC_250059)

[Контрольные вопросы 35](#_TOC_250058)

Тема 3. Представление информации в компьютерных

системах 37

* 1. [Числовое кодирование данных 37](#_TOC_250057)
  2. [Системы счисления 38](#_TOC_250056)
     1. [Аддитивные системы счисления 39](#_TOC_250055)
     2. [Позиционные системы счисления 41](#_TOC_250054)
  3. [Двоичное кодирование текстовой информации 49](#_TOC_250053)
     1. Стандарт ASCII 50
     2. Стандарт UNICODE 53
  4. [Двоичное кодирование десятичных чисел 56](#_TOC_250052)
     1. [Кодирование натуральных чисел 57](#_TOC_250051)
     2. [Кодирование целых чисел со знаком 58](#_TOC_250050)
     3. [Кодирование вещественных чисел 64](#_TOC_250049)

[Контрольные вопросы 69](#_TOC_250048)

[Тема 4. Аппаратное обеспечение ЭВМ 71](#_TOC_250047)

* 1. [Базовые принципы функционирования ЭВМ 71](#_TOC_250046)
  2. [типовая архитектура простейшей ЭВМ 74](#_TOC_250045)
  3. [Центральный процессор 77](#_TOC_250044)
  4. ЗапоминаJОщие устройства 80
  5. [Периферийное оборудование 87](#_TOC_250043)
  6. [Адресное пространство ПК 88](#_TOC_250042)
     1. Сегментная организация адресного пространства

основной памяти 89

* + 1. Стандартное распределение базового адресного

пространства 93

* + 1. [Адресное пространство ввода-вывода 97](#_TOC_250041)

[Контрольные вопросы 101](#_TOC_250040)

3

Тема 5. Программное обеспечение ЭВМ 102

* 1. [Классификация программного обеспечения 102](#_TOC_250039)
  2. Системное ПО 103
     1. Программная структура MS DOS 103
     2. Процесс загрузки MS DOS 105
     3. Функциональная структура MS DOS 107

[Контрольные вопросы 111](#_TOC_250038)

Тема б. Файловые системы 112

* 1. [Трехуровневая модель дискового пространства 112](#_TOC_250037)
  2. [Пользовательский уровень представления файловых систем 116](#_TOC_250036)
     1. [Классификация команд 116](#_TOC_250035)
     2. [Формат команд 118](#_TOC_250034)
     3. Примеры использования команд 119
     4. [Использование групповых имен файлов 120](#_TOC_250033)
     5. [Перенаправление вывода 121](#_TOC_250032)
     6. [Программирование пакетных (.bat) файлов 122](#_TOC_250031)
  3. [Файловые FАТ-системы 127](#_TOC_250030)
     1. [Структура системной области тома 128](#_TOC_250029)
     2. [Алгоритмы выполнения файловых операций 135](#_TOC_250028)
     3. Недостатки FАТ-систем 146
  4. Файловые системы NТFS 149
     1. Структура тома NТFS 149
     2. Мета-файлы NТFS 150
     3. [Схемы хранения файлов 151](#_TOC_250027)
     4. [Схемы хранения каталогов 154](#_TOC_250026)
     5. [Атрибуты файлов и каталогов 157](#_TOC_250025)

[Контрольные вопросы 158](#_TOC_250024)

Тема 7. Обмен данными с периферийными устройствами 160

* 1. [Система обработки прерываний 160](#_TOC_250023)
     1. [Аппаратные прерывания 161](#_TOC_250022)
     2. [Программные прерывания 161](#_TOC_250021)
     3. [Контроллер прерываний 162](#_TOC_250020)
     4. [Таблица векторов прерываний 166](#_TOC_250019)
     5. [Процедура обработки прерывания 167](#_TOC_250018)
  2. Клавиатура персонального компьютера 168
     1. [Контроллер клавиатуры 169](#_TOC_250017)
     2. [Алгоритм обработки прерывания клавиатуры 171](#_TOC_250016)
  3. Видеосистема персонального компьютера 182
     1. Аппаратный комплекс 183
     2. [Программное обеспечение 185](#_TOC_250015)
     3. Структуры данных 186
     4. [Кодирование данных в видеопамяти 188](#_TOC_250014)
     5. [Знакогенераторы 191](#_TOC_250013)

[Контрольные вопросы 193](#_TOC_250012)

4

[**Практикум**](#_TOC_250011)

**Сборник задач с комментариями и примерами решений**

[Общие методические указания 197](#_TOC_250010)

[Задания по теме 2 «Информация» 198](#_TOC_250009)

Задания по теме 3 «Представление информации в компьютерных системах» 201

[Позиционные системы счисления 201](#_TOC_250008)

[Двоичное кодирование текстовых (символьных) данных 202](#_TOC_250007)

[Двоичное кодирование целых десятичных чисел 203](#_TOC_250006)

Двоичное кодирование вещественных десятичных чисел

(стандарт IEEE 754) 204

Задания по теме 4 «Адресное пространство персонального

компьютера» 205

[Расчетные задания 205](#_TOC_250005)

Лабораторно-исследовательские задания 205

Задания по теме 6 «Файловые системы персонального

компьютера» 206

[Командный пользовательский интерфейс 206](#_TOC_250004)

[Исследование алгоритмов выполнения типовых файловых операций 208](#_TOC_250003)

Задания по теме 7 «Обмен данными с периферийными

устройствами» 213

[Система обработки прерываний 213](#_TOC_250002)

Клавиатура персонального компьютера 213

[Видеосистема персонального компьютера 215](#_TOC_250001)

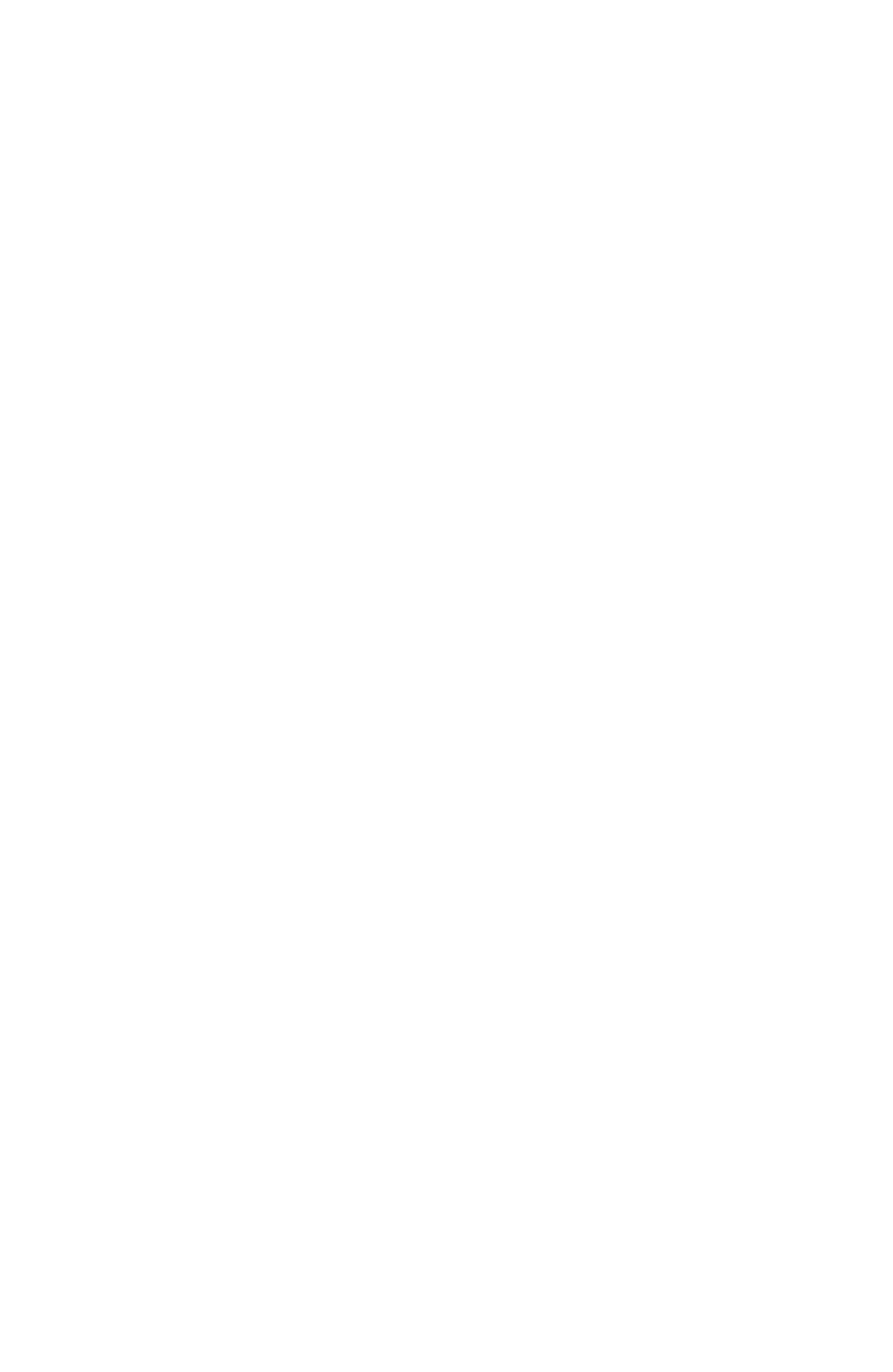
[Примеры решений 217](#_TOC_250000)

Приложение. Инструкция по установке виртуальной

DОS-машины 222

Новые издания по дисциWiине «Информатика»

и смежным дисциWiинам 224



https://urait.ru

# Предисловие

Дисциплина «Информатика» прочно закрепилась в вузовских образовательных программах как технических, так и гуманитар­ ных направлений и занимает в них достойное место в одном ряду с другими общеобразовательными математическими и естествен­ но-научными дисциплинами. Давно упти в прошлое те времена, когда основной задачей этой дисциплины считались «ликвидация компьютерной безграмотности» и формирование «базовых навыков работы на компьютере» -у сегодняшних студентов-первокурсни­ ков, как правило, все в порядке и с компьютерной грамотностью, и с базовыми навыками.

Существует множество вузовских учебников и учебных пособий по информатике, рекомендованных студентам различных специаль­ ностей или групп специальностей. В этих учебниках информатика рассматривается как общеобразовательная дисциплина, задачами которой являются формирование у студентов базовых представле­ ний об информатике, вычислительной технике и машинной ариф­ метике, а также практическое освоение студентами информаци­ онных технологий и пользовательских программных приложений, состав которых определяется профилем образовательной програм­

**мы.**

При всем многообразии учебников по информатике ни один их них не может быть рекомендован студентам IТ-специальностей в качестве основного, и это вполне объяснимо, так как для таких студентов профессиональное освоение информатики не может ограничиться какой-либо одной дисциплиной - различные аспек­ ты этой науки и ее многочисленных приложений рассматриваются комплексом взаимосвязанных профессиональных дисциплин в те­ чение всего периода обучения.

Вводный курс информатики для студентов IТ-специальностей по­ зиционируется как практическое введение в прикладные разделы этой науки, в котором рассматриваются ее базовые понятия, мето­ ды двоичного кодирования и машинного представления текстовых и числовых данных, типовая структура, принципы функционирова­ ния и алгоритмы взаимодействия аппаратных и программных ком­ понентов простейшего персонального компьютера (центральный процессор, адресное пространство, система обработки прерываний, клавиатура, видеосистема и файловая система).

**7**

В результате освоения дисциплины студент должен:

***знать***

* базовые понятия информатики, ее место в ряду фундамен­ тальных наук;
* стандартные определения и трактовку понятий «данные»,

«информация», «знания» и взаимосвязи между ними;

* типовой жизненный цикл информации как информацион­ ного ресурса организации, функциональную структуру системы управления этим ресурсом;
* свойства информации и методы ее количественной оценки;
* основы двоичной и шестнадцатеричной арифметики, спосо­ бы двоичного кодирования и представления числовых и текстовых данных в вычислительных устройствах;
* базовые принципы функционирования компьютера, состав и назначение его аппаратных компонентов и программного обе­ спечения;
* основы организации адресного пространства персонального компьютера и типовые алгоритмы обмена данными с периферий­ ными устройствами;
* структуры данных файловых систем и алгоритмы работы ос­ новных файловых функций;

***уметь***

* использовать инструментальные программные средства для анализа работы компонентов компьютера;

***владеть***

* командным языком управления файловой системой.

Автор выражает благодарность Аркадию Андреевичу Медведеву, внимательно прочитавшему рукопись и сделавшему ряд профес­ сиональных комментариев и замечаний методического характера, несомненно способствовавших повышению качества учебного по­ собия, и Артуру Германовичу Рабушко, оказавшему помощь в фор­ мировании программного обеспечения лабораторных работ, вошед­ ших в практикум.

## Тема 1

**ИНФОРМАТИКА - ИСТОРИЯ СТАНОВЛЕНИЯ**

Идея создания автоматического программно-управляемого вы­ числительного устройства была высказана английским матема­ тиком Чарльзом Бэббиджем в 1833 г., но только через 100 лет эти идеи нашли практическое воплощение. Основы построения вычис­ лительных машин в их современном понимании бьmи заложены английским математиком А. ТЬюрингом и американцем Дж. Ней­ маном. А. ТЬюринг сформулировал (1936 г.) понятие абстрактной вычислительной машины, получившей название *«машины Тьюрин­ г.а»,* которая, хотя и не была реализована в качестве действующего вычислительного устройства, до настоящего времени использует­ ся для моделирования алгоритмов и вычислительных процессов. Дж. Нейман предложил свою модель вычислительного устройства *(«автомат Неймана»),* позволяющую, в отличие от машины ТЬю­ ринга, естественно моделировать параллельные вычислительные процессы.

Первым реально работавшим автоматическим вычислительным устройством считается электромеханический вычислитель Zl, со­ бранный немецким инженером Конрадом Цузе в 1938 г. В 1944 г. в США была создана расчетно-механическая машина Марк-1. В этих конструктивно сложных машинах в качестве элементной базы были использованы различные электромеханические устройства, что не позволило их разработчикам добиться желаемого уровня надеж­ ности работы этих машин и высокой производительности вычисле­ **ний.**

Новейшая история развития вычислительной техники связана с достижениями в области вычислительной математики, промыш­ ленной электроники и схемотехники. Элементная база ЭВМ перво­ го поколения - это электронные лампы. Первый ламповый триг­ гер был создан российским ученым Бонч-Бруевичем в 1918 г., годом позднее аналогичная схема была продемонстрирована американца­ ми Икклзом и Джорданом.

Проекты первых вычислительных машин на электронных радио­ лампах были разработаны в Германии и США - это проект Z2 Кон­ рада Цузе и проект ENIAC Джона Моучли. Машина Z2 была введена в эксплуатацию в 1940 г. и использовалась на одном из берлинских авиационных заводов для расчетов геометрических параметров ави-

**9**

ационных бомб, а опытная эксШiуатация ENIAC началась в 1944 г. в одной из лабораторий Пенсильванского университета.

Среди отечественных разработок ЭВМ первого поколения мож­ но отметить проекты МЭСМ (малая электронно-счетная машина) и БЭСМ-1 (1949-1952), ЭВМ «Стрела» (1953) и серия ЭВМ «Урал».

В середине 1950-х гг. на смену ламповым ЭВМ пришли вычисли­ тельные машины второго поколения, элементная база которых была построена на полупроводниковых приборах. Существенно снизи­ лась энергоемкость вычислений, машины стали более компактны­ ми и быстродействующими, они стали оснащаться достаточно ем­ кими (для своего времени) запоминающими устройствами - все это привело к тому, что ЭВМ превратилась из быстродействующего программируемого калькулятора в мощный инструмент для надеж­ ного хранения и высокопроизводительной обработки больших объ­ емов информации.

В результате сформировалось новое научно-техническое направ­ ление, получившее свое название от французского *L'informatique* (на немецком - *Die Informatik,* на испанском - *La informatica,* на русском - *информатика).* Англоязычное название информати­ ки - *The computer science* - подчеркивает важную роль вычисли­ тельной техники в информатике.

Информатика - это комШiексное междисциШiинарное направ­ ление, в рамках которого изучаются процессы получения, передачи, хранения и обработки информации средствами электронно-вычис­ лительной техники и связи. В широком смысле информатика - это наука об информационных процессах **и их** организации в человеко­ машинных (социотехнических) системах.

Информатика - относительно молодая, но динамично развива­ ющаяся наука, интегрирующая и использующая передовые дости­ жения многих фундаментальных и прикладных наук - таких как *теория информации,* изучающая способы восприятия, преобразо­ вания и передачи информации; *теория автоматов,* изучающая специальный класс дискретных систем переработки информации; *теория программирования,* изучающая математические модели и алгоритмы нахождения решений и способы кодирования этих алгоритмов в форме, воспринимаемой вычислительной системой; *теория управления,* изучающая процессы управления объектами разных классов и принципы построения систем, осуществляющих целенаправленную переработку информации.

Информатика составляет основу информационных технологий (ИТ) - инженерных методов автоматизированной обработки и ис­ пользования информации. Повышение вычислительной мощности ЭВМ, интенсивное развитие научной базы информатики и практи­ ки компьютерного программирования привели к тому, что за от­ носительно небольшой исторический период ИТ нашли достойное

**10**

место среди других (более традиционных) инженерных отраслей и сегодня эффективно применяются в различных сферах деятель­ ности человека, в том числе и весьма далеких от инженерии.

Вслед за сменой поколений вычислительной техники ИТ прош­ ли в своем развитии ряд этапов, на каждом из которых перед ними ставились определенные цели и вырабатывались технологические инструменты для их достижения.

*Первый этап* развития ИТ (1950-е - начало 1960-х гг.) харак­ теризуется дефицитом машинных ресурсов. Основная технологиче­ ская проблема в этот период - низкая скорость вычислений и вы­ сокая стоимость самих вычислителей. В 1953 г. один из основателей теории информации Клод Шеннон писал: *«Наши ЭВМ выглядят, как ученые-схоласты* - *при вычислении длинной цепи арифметиче­ ских действий они значительно обгоняют человека, когда же их пы­ таются использовать для выполнения неарифметических операций, они оказываются неуклюжими и неприспособленными для такой работы».*

Основной целью ИТ на этом этапе было повышение эффектив­ ности компьютерной обработки данных по легко формализуемым алгоритмам, и эта цель достигалась путем повышения быстродей­ ствия ЭВМ и оптимизации программ по критериям длины машин­ ного кода и затрат оперативной памяти на программную реализа­ цию вычислительных алгоритмов.

В 1950-е гг. были сделаны важные шаги в направлении перехо­ да от аппаратно-арифметического подхода к программированию на низкоуровневых машинно-ориентированных языках к алгорит­ мическим языкам высокого уровня, что привело к повышению про­ изводительности разработки программ1.

В этот период были разработаны четыре базовых высокоуров­ невых языка: FORTRAN-1, нацеленный на программирование ма­ тематических задач, COBOL, эффективно работавший с текстами и записями и предназначенный для разработки бизнес-приложе­ ний и решения экономических задач, язык ALGOL-58, в котором впервые реализован блочный (структурный) подход к построению программы, и язык функционального программирования LISP, ори­ ентированный на обработку списковых структур данных и специа­ лизированный для решения задач не численного характера.

1 Идея использования высокоуровневых команд управления вычислениями была высказана Конрадом Цузе, им же был разработан первый высокоуровневый язык программирования Планкалкюль (Plankalkiil- «исчисление планов»), реали­ зованный в проекте ZЗ 1941 г. и опубликованный только в 1948 г. Этот язык был свободно-переносимым, т. е. независимым от системы команд машины, включал операторы условных переходов и циклов, обеспечивал возможность работы с под­ программами и многомерными массивами.

**11**

Возможно, главной чертой *второго этапа* развития ИТ (с сере­ дины 1960-х до конца 1980-х гг.) следует считать появление микро­ процессорной техники и, как следствие - создание мини-ЭВМ и массовое распространение персональных компьютеров. Вот крат­ кая история «миниатюризации» компьютеров:

1969 год - компания Honeywell выпускает на рынок первый до­ машний компьютер;

1971 год - выпущен первый *микропроцессор* - 4-разрядный *Intel 4004;*

1973 год - первый ПК Xerox Alto с графическим интерфейсом

и прообразом «рабочего стола»;

1974 год - выпущен первый В-разрядный микропроцессор *Intel 8080* (архитектурный аналог которого выпускался в СССР в составе микропроцессорного комплекта *К580);*

1977 год - первый компьютер для массового пользователя - серийный моноблок Apple II с цветным графическим дисплеем и встроенным Ваsiс-интерпретатором;

1978 год - выпущен первый 16-разрядный микропроцессор *Intel 8086* (отечественный аналог - *микропроцессор К1810ВМ80),* ко­ торый имел 20-разрядную адресную шину, что позволило в 16 раз увеличить размер адресного пространства (до 1 Мб). Архитектура этого микропроцессора, получившая обозначение *х86,* фактически стала стандартом на длительный период развития персональных компьютеров;

1982 год - выпущен 16-разрядный микропроцессор *Intel 80286* с увеличенной до 24 разрядов адресной шиной, что позволило еще в 16 раз расширить емкость адресуемой памяти;

1983 год - IВМ Р80С ХТ с увеличенным объемом оперативной памяти и жестким диском емкостью 10 Мбайт (!);

1984 год - первый серийно выпускаемый отечественный персо­ нальный компьютер АГАТ с В-разрядным процессором отечествен­ ного производства;

1985 год - начало выпуска 32-разрядных микропроцессоров *Intel 80386* (впоследствии выпускались процессоры Intel 80486 с интегрированным арифметическим сопроцессором, Intel Pentium и их модификации);

1986 год - первый ноутбук IВМ РС ConvertiЫe с блоком питания от батареи, жестким диском и двумя 3-дюймовыми дисководами.

Еще одна важная черта второго этапа развития информацион­ ных технологий - использование межкомпьютерных коммуника­ ций. Попытки создания компьютерных сетей предпринимались еще в начале 1960-х гг., когда были разработаны первые близкие к со­ временным способы соединения компьютеров в сети. Началом эпо­ хи компьютерных сетей считается 1969 г., когда в США была созда-

**12**

на сеть ARPANET, объединившая по телефонному кабелю несколько компьютеров, установленных в четырех университетах.

В 1971 г. к этой сети бьmи подключены еще 12 терминалов, а еще через два года к ней подключились и иностранные организации из Норвегии и Великобритании - так ARPANET стала прообразом современной глобальной сети интернет.

С переходом на новую элементную базу компьютеры стали де­ шевле и производительнее, они стали оснащаться быстродействую­ щими дисковыми накопителями большой емкости, что дало толчок к развитию сетевых технологий и технологий баз данных. В ре­ зультате области применения компьютеров вышли далеко за рам­ ки научных и инженерных расчетов, и вслед за развитием рынка персональных компьютеров появился и рынок компьютерных про­ грамм, ориентированных на массового потребителя, а стоимость программного обеспечения компьютера стала больше стоимости его аппаратных компонентов.

Вскоре спрос на рынке программного обеспечения начал суще­ ственно превышать предложение по причине высокой трудоемко­ сти процесса программирования. Прогнозы конца 1960-х гг. показы­ вали, что при сохранении достигнутых к тому времени технологий разработки программ уже к середине 1990-х гг. программировани­ ем должно *бъто* бы заниматься все население планеты. Создавшая­ ся ситуация бьта определена, как «кризис программирования».

На фоне радикальных изменений в аппаратуре компьютерных систем произошло смещение критериев оценки ИТ - от эффек­ тивности исполнения программного кода вычислительным устрой­ ством к эффективности процессов разработки программ и их сопро­ вождения на стадии эксплуатации.

Основным результатом радикального пересмотра критериев ка­ чества ИТ стало развитие концепций модульного программирова­ ния и структурного подхода к проектированию и программирова­ нию сложных программных систем.

В 1968 г. в профессиональный оборот *бьт* введен термин *software engineering* - *инженерия про2раммно20 обеспечения,* рассматриваю­ щая программный продукт как технически сложное промышлен­ ное изделие, к которому применимо понятие «жизненного цикла», включающего программирование лишь как один из этапов процес­ са производства программного продукта.

Бьта выработана новая концепция операционных систем, обе­ спечивающих повышение эффективности труда программистов за счет отказа от примитивного «пакетного режима» выполнения и отладки программ. В качестве примера можно привести ОС UNIX, успешно функционировавшую на различных мини-ЭВМ с начала 1970-х гг., а также РС DOS с развитым «командным» языком, уста­ навливаемую на IВМ-совместимые персональные компьютеры.

**13**

Были разработаны новые версии языков программирования (ALGOL-69, FORTRAN-66/77) и созданы новые языки - Pascal (1970) и Си (1972), позволяющие программисту оперировать слож­ ными структурами данных и поддерживающие концепции струк­ турного и модульного программирования, за счет чего улучшалась

«читабельность» программного кода и сокращалось время отладки и модификации программ. Был создан первый непроцедурный вы­ сокоуровневый язык программирования PROLOG, в основе которо­ го лежит язык предикатов математической логики.

Управление большими объемами информации было выделено в отдельную ИТ-отрасль, получившую название *«технологии баз данных».* В практику проектирования баз данных введено инфор­ мационное моделирование бизнес-процессов предметной области, были разработаны первые проектные модели и системы графиче­ ского моделирования. Введено понятие *«модели данных»,* разработа­ ны теоретические основы и специализированные языки управления данными, была выработана концепция СУБД - систем управления базами данных.

В 1968 г. компания IВМ выпустила на рынок первую СУБД *IMS,* поддерживающую иерархическую модель данных. В 1969 г. Кон­ ференцией по языкам систем данных *(Conference оп Data Systems Languages)* была разработана *сетевая модель данных,* получившая название *«модели CODASYL».* В 1970 г. сотрудник компании IВМ Э. Кодд предложил реляционную модель данных и детально разра­ ботал математическую основу этой модели - реляционную алге­ бру, а также теорию нормальных форм и технологию нормализации баз данных. К середине 1970-х гг. идеи Э. Кодда и его последовате­ лей были реализованы в первой реляционной СУБД System R, вы­ пущенной компанией IВМ.

*Третий этап* развития информационных технологий, начав­ шийся в 1990-е гг. и продолжающийся в настоящее время, харак­ теризуется переходом к объектно-ориентированной методологии разработки программных систем, дальнейшим развитием сетевых технологий и распределенных систем хранения и обработки данных, а также получением практически значимых результатов в сфере ис­ кусственного интеллекта и *Data Mining* - мультидисциплинарной области, в рамках которой рассматриваются методы и компьютер­ ные технологии извлечения ранее неизвестных знаний из больших объемов слабоструктурированных данных.

Достижения второго этапа развития ИТ позволили (как позже выяснилось - временно) преодолеть кризис программирования, но дальнейшее расширение сферы применения компьютерной тех­ ники в условиях динамично развивающегося рынка программного обеспечения потребовало повышения мобильности процессов раз­ работки: стала объективной реальностью ситуация, когда заказ-

**14**

чик программного продукта вносит *изменения в требования* к его функционированию не только на стадии эксплуатации (с чем раз­ работчики уже успели смириться), но и *на стадиях проектирования* и *программирования.*

В этих условиях разработка программной системы превраща­ ется в ее постоянное перепроектирование и перепрограммирова­ ние, и потребовалась методология, обеспечивающая возможность оперативного внесения изменений в программу без существенных изменений ранее написанного кода. В результате была предложе­ на *объектно-ориентированная* методология разработки программ­ ных систем, которая, используя достижения структурного подхода, радикально меняет как технологию проектирования системы, так и структуру программного кода ее компонентов.

Согласно структурному подходу решение задачи представляется в виде последовательности вызовов функций, реализующих опре­ деленные алгоритмы. Объектно-ориентированная архитектура про­ граммной системы представляется множеством взаимосвязанных объектов, взаимодействующих друг с другом путем передачи сооб­ щений и (или) предоставления ресурсов. Состояние программной системы определяется свойствами (значениями атрибутов) всех ее объектов, а функционирование системы рассматривается как процесс смены ее состояний в результате взаимодействия объектов. Внедрение объектно-ориентированной методологии потребо­ вало разработки соответствующего этой методологии языка гра­ фического моделирования, и такой язык был создан - он полу­ чил название UML *(Unified Modeling Language).* Были разработаны и поддерживающие этот язык САSЕ-средства *(Computer Aided Soft Engineering),* обеспечивающие процессы разработки и документиро­

вания программных проектов.

На смену языкам структурного программирования пришли объ­ ектно-ориентированные языки, наиболее известные из которых - С++, С#, Visual Basic, Java, Python.

Еще одна особенность ИТ рассматриваемого периода связана с прогрессом в области сетевых технологий и массовых телекомму­ никаций: высокоскоростные (в том числе и беспроводные) каналы передачи данных, WеЬ-технологии доступа к информационным ре­ сурсам, облачные сервисы и распределенные хранилища данных, мобильная связь, IР-телефония и мультимедиа, мессенджеры и со­ циальные сети - благодаря этим новым возможностям области применения ИТ существенно расширились, и самая разнородная информация стала массово доступной.

Доступность информационных ресурсов создала и новые угро­ зы - потребовалось защищать информацию от несанкциониро­ ванного доступа со стороны пользователей компьютерных систем, а также защищать самих пользователей от разглашения, модифи-

**15**

кации или утраты принадлежащей им персональной информации. В этот период сформировалась отдельная ИТ-отрасль - информа­ ционная безопасность, в рамках которой рассматриваются про­ граммно-технические, нормативно-правовые и организационные аспекты защиты информации в компьютерных системах.

Характеризуя третий этап развития ИТ, нельзя не упомя­ нуть и о системах искусственного интеллекта (ИИ, англ. *Artificial Intelligence, Al).* Теоретические основы и методологические подходы к созданию таких систем были выработаны еще в 1960-х гг. трудами философов и нейробиологов, специалистов в области математиче­ ской логики, прикладной статистики и теории алгоритмов, но для их практической реализации требовались вычислительные системы высокой производительности, которые появились только к началу 1990-х гг.

Несмотря на отсутствие общепризнанного определения искус­ ственного интеллекта, существует понимание того, что искусствен­ ный интеллект - это способность искусственных, т. е. созданных человеком, программно-технических систем автоматически вы­ полнять те виды человеческой деятельности, которые традиционно считаются творческими (интеллектуальными).

Технологии ИИ интенсивно развиваются в двух основных на­ правлениях: *семиотическое* направление занимается созданием *баз знаний* и *экспертных систем,* имитирующих психические процессы мыптения, рассуждения, логического вывода, и *биологическое,* в ко­ тором разрабатываются алгоритмы эволюционных вычислений, ис­ кусственные *нейронные сети* и *биокомпъютеры.*

Оба направления напти свое применение при создании систем ИИ в самых различных областях: робототехника и управление бес­ пилотными транспортными средствами, распознавание текстов и графических образов, компьютерные игры, обучающие образо­ вательные системы и системы самообучения машин, системы под­ держки принятия решений в маркетинге, бизнесе и в медицине, многомерная аналитическая обработка данных, поиск неявных за­ кономерностей в больших массивах статистических данных и мно­ гие другие.

Приведенный выше краткий исторический очерк дает представ­ ление об информатике как о комплексном научно-техническом направлении со своей теоретической базой, аппаратным и про­ граммным обеспечением и множеством практических приложений. Прикладные разделы информатики создают методологическую ос­ нову информационных технологий и программной инженерии, за­ нимающейся разработкой и эксплуатацией автоматизированных информационных, информационно-аналитических, информаци­ онно-управляющих и информационно-телекоммуникационных си­ стем.

**16**

Вычислительная система - это неразрывный комru1екс аппарат­ ного и программного обеспечения, в котором аппаратные компо­ ненты работают под управлением компьютерных программ, а про­ граммы выполняются на аппаратной платформе, обрабатывают данные, хранимые на аппаратных устройствах, и взаимодействуют друг с другом, реагируя на сигналы, поступающие от аппаратных компонентов. При этом программа остается практически бесполез­ ной без реализующей ее аппаратуры, а аппаратура без управляю­ щей программы - не более, чем мертвое (хотя и весьма дорогосто­ ящее) «железо».

Изучение теоретической информатики и информационных тех­ нологий, а также процессов функционирования компонентов элек­ тронно-вычислительных машин и системного программного обе­ спечения выходит за рамки вводного курса информатики - теория информации, технологии программирования, хранения данных и управления данными, схемотехнические основы и архитектура ЭВМ, низкоуровневое программирование и функционирование опе­ рационных систем рассматриваются в соответствующих профиль­ ных дисциru1инах, включенных в учебные ru1аны практически всех IТ-специальностей.

В курсе рассмотрены основы двоичного кодирования информа­ ции, принципы функционирования ЭВМ и примеры реализации этих принципов в персональном компьютере простейшей маги­

стральной архитектуры. Основное внимание уделено изучению и экспериментальному исследованию алгоритмов функционирова­ ния файловой системы, системы обмена данными с периферийным оборудованием и видеосистемы ПК, а также изучению структур данных, обслуживающих эти алгоритмы.

## Тема2

**ИНФОРМАЦИЯ**

##### Понятие информации

Латинское понятие *informatio,* давшее свое «имя» информатике как целому научно-техническому направлению, может быть пере­ ведено на русский язык как «разъяснение», «изложение» или «ос­ ведомленность», при этом слова, обозначающие это понятие, су­ ществовали во многих естественных языках (как, например, слово

«информация» в русском) задолго до появления информатики. Клод Шеннон ввел в употребление технический термин «информация» применительно к *теории передачи кодов,* получившей впоследствии название *теории информации,* но само понятие информации все еще остается интуитивным и получает разные толкования в различ­ ньrх сферах деятельности человека.

В широком смысле слова информация - это *отображение (ин­ формационная модель) реального мира,* в узком техническом смыс­ ле *информация* - *это сведения, являющиеся объектом хранения, передачи и преобразования.*

Информация - это определенный набор свойств (параметров) реального объекта, и проявление этих свойств и их восприятие дру­ гими объектами становятся возможными только в процессе инфор­ мационного обмена между объектами при их взаимодействии.

В материальном мире существует два фундаментальных вида материального взаимодействия: вещественное и энергетическое, которые подчиняются законам сохранения и в этом смысле являют­ ся симметричными: сколько один объект отдал, столько же другие и получили. В отличие от материальных, информационное взаимо­ действие является *несимметричным* и не подчиняется законам со­ хранения - объекты-приемники получают переданную им инфор­ мацию, но при этом объект-источник ее не теряет.

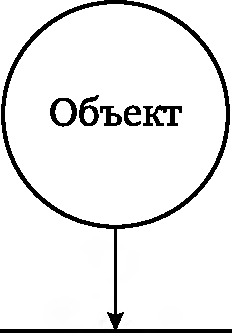
Естественно, что информационное взаимодействие сопрово­ ждается энергетическим и (или) вещественным, обеспечивающим перенос информационных сигналов посредством изменения пара­ метров какой-либо материальной среды (например, акустической или электромагнитной).

**18**

С учетом сказанного сформулируем следующее определение.

Любое взаимодействие между объектами, в процессе которого один из них приобретает некоторую субстанцию, а другой ее не теряет, будем называть *информационным взаимодействием,* а передавае­ мую при этом субстанцию будем называть *информацией.*

Информация представляется в форме *информационного сообще- ния,* передаваемого от источника сообщения к получателю по кана­ лу связи (рис. 2.1).



Источник сообщения

Кодирующее устройство

Канал связи

Декодирующее устройство

Получатель сообщения

*Рис. 2.*7. **Схема передачи информационноrо сообщения**

*Источник сообщения* получает информацию об *объекте* и фор­ мирует информационное сообщение, пользуясь некоторой систе­ мой знаков, которую будем называть *алфавитом.* Таким алфавитом может быть алфавит русского или английского языков, азбука Мор­ зе, азбука Брайля, двоичный или десятичный цифровые алфавиты.

*Кодирующее устройство* производит преобразование формы представления информационного сообщения: при этом могут изме-

**19**

пяться как исходный алфавит сообщения, так и физическая форма представления сообщения. На выходе кодирующего устройства со­ общение представлено в форме сигнала, соответствующего параме­ трам канала связи.

*Канал связи* представляет собой некоторую физическую среду, которая изменяет свои энергетические параметры под воздействи­ ем передаваемого сигнала.

*Декодирующее устройство* регистрирует изменение этих параме­ тров канала связи и преобразует принятые сигналы к физической форме и алфавиту, принятым у получателя сообщения. Таким обра­ зом, сообщение поступает к получателю в удобной для него форме, позволяющей получателю производить интерпретацию и обработку сообщения.

Перечислим основные факторы, присущие информационному взаимодействию.

1. Информационные взаимодействия возможны только в ре­ зультате симметричных взаимодействий, связанных с передачей вещества и энергии. Формы вещества или энергии, с помощью ко­ торых переносится информация, называются *информационными ко­ дами.*
2. Информационное взаимодействие может происходить только при взаимном соответствии свойств передающего и принимающе­ го объектов. От свойств принимающего объекта зависит то, какую информацию он принимает, получая конкретный набор кодов. Ком­ плекс свойств принимающего объекта, позволяющих ему воспри­ нимать получаемые коды как некоторую информацию, называют *аппаратом интерпретации.*
3. *Информация, принимаемая объектом, является целесо­ образной для принимающего объекта,* а нецелесообразная для этого объекта информация не будет им принята по причине отсутствия у него соответствующего аппарата интерпретации.
4. *Информация, передаваемая объектом, является для него зна­ чимой,* а процесс передачи информации является *целенаправленным.*

#### Свойства информации

Информационные технологии - это способы целенаправленной обработки информации, реализация которых приводит к измене­ нию ее свойств. Рассматривая свойства информации как объекта ИТ, выделяют два класса таких свойств: внутренние и внешние. Внутренние свойства информации - это свойства, органически присущие самому отображаемому объекту, а внешние ее свойства проявляются во взаимодействии объектов, участвующих в процессе информационного обмена.

**20**

Рассмотрим внешние свойства информации, оцениваемые полу­ чателем информационного сообщения, который, как правило, имеет некоторое представление о возможном его содержании и оценивает качество, полезность и возможность использования полученной ин­ формации в соответствии со своими ожиданиями.

При этом возможны следующие типовые ситуации:

а) полученная информация может соответствовать, не соответ­ ствовать или частично соответствовать запросу получателя;

б) информация может соответствовать запросу пользователя, но ее оказалось недостаточно для решения соответствующей за­ дачи;

в) информация оказалась недостоверной по причине наличия скрытых ошибок в информационном сообщении;

г) полученная информация несвоевременна (уже устарела или, наоборот, поступила преждевременно);

д) информация представлена в неудобной для получателя фор­ ме (например, использован незнакомый получателю алфавит);

е) информация недоступна (например, по причине аппаратной или программной несовместимости технических устройств или от­ сутствия прав доступа к ней у получателя сообщения).

В соответствии с рассмотренными ситуациями можно сформули­ ровать следующие *внешние свойства информации,* проявляющиеся в процессе ее обработки получателем сообщения:

* ***релевантность*** - степень соответствия информации запро­ су получателя;
* ***достоверность*** - свойство информации, определяющее на­ личие (и количество) скрытых ошибок;
* ***полнота*** - свойство информации исчерпывающе (для полу­ чателя) характеризовать отображаемый объект;
* ***эргономичность*** - свойство, характеризующее удобство формы представления информации в информационном сообщении;
* ***доступность*** - свойство, характеризующее возможность получения информационного сообщения;
* ***своевременность*** - свойство, характеризующее время полу­ чения информационного сообщения.

Если рассматривать другую группу свойств информации, связан­ ных с взаимодействием источника информации с отображаемым объектом, то здесь важнейшим свойством информации является ее ***адекватность*** - *свойство информации однозначно соответ­ ствовать отображаемому объекту.*

С точки зрения получателя информационного сообщения адек­ ватность является внутренним свойством информации, проявляю­ щимся через такие ее внешние свойства, как релевантность, досто­ верность и полнота.

**21**

https://urait.ru

##### Количество информации

Реализация базовых процессов информационных технологий, обеспечивающих обработку, передачу или хранение информации, требует количественной оценки информационных сообщений, по­ этому количество информации - важнейшее ее внутреннее свой­ ство. Для количественной оценки информации необходимо выбрать единицы ее измерения, что позволит приписать определенной пор­ ции информации некоторое числовое значение.

Существует три основных метода оценки количества информа­ ции: алгоритмический, объемный и энтропийный.

*Алгоритмический* метод применяется в теории информации и теории алгоритмов. Согласно этому методу, *количество инфор­ мации, содержащееся в информационном сообщении, определяется алгоритмической сложностью компьютерной программы, воспро­ изводящей это сообщение.* Например, сообщение вида «0000» будет содержать меньше информации, чем сообщение вида «0101», так как программа, генерирующая первое сообщение, очевидно проще (короче) программы, генерирующей второе. Для того, чтобы реаль­ но оценить количество информации по алгоритмическому методу, необходимо задаться некоторым единым алгоритмическим языком, на котором следует записывать программы-генераторы оценивае­ мых информационных сообщений. Для этих целей используется язык *машины Тьюринга* - абстрактной модели простейшего вы­ числительного устройства, уже упоминавшейся во вводной части курса.

*Объемный метод* оценки количества информации - самый про­ стой и очевидный: согласно этому методу, количество единиц ин­ формации, содержащихся в информационном сообщении, равно длине (количеству символов) самого сообщения. При всей простоте такого метода оценки он оказывается чувствительным к форме за­ писи (алфавиту) сообщения. Следующий пример иллюстрирует тот факт, что при использовании объемного подхода одно и то же тек­ стовое сообщение, записанное в разных алфавитах, будет иметь раз­ личную количественную оценку: «21»- 2 единицы, «XXI» - 3 еди­ ницы, «twenty one» - 10 единиц, «двадцать один» - 13 единиц.

*Энтропийный метод* оценки количества информации, принятый в теории информации и кодирования, использует следующую модель:

1. получатель сообщения имеет определенное представление о его возможном содержании - это представление выражается ве­ роятностями, с которыми он ожидает тот или иной вариант полу­ чаемого сообщения;
2. общая мера неопределенности (энтропия) характеризуется некоторой математической зависимостью от совокупности этих ве­ роятностей;

**22**

1. количество информации, содержащейся в информационном сообщении, определяется тем, насколько уменьшится энтропия (мера неопределенности) после получения данного сообщения.

Рассмотрим простой пример. Из колоды игральных карт (32 кар­ ты) наугад выбирается одна карта. Всего имеется 32 равновероят­ ных варианта получения информационного сообщения «выбрана конкретная карта», таким образом, общая мера неопределенности может быть оценена числом 32. После получения сообщения (вы­ бора определенной карты) неопределенность полностью снимает­ ся, следовательно, число 32 определенным образом характеризует количество информации, содержащейся в полученном сообщении.

Очевидно, что чем больше вариантов сообщения, тем больше мера неопределенности и, соответственно, больше информации со­ держит полученное сообщение. Если бы в рассмотренном примере колода содержала не 32, а 256 различных игральных карт, то же са­ мое сообщение «выбрана конкретная карта» содержало бы больше информации, чем в предыдущем случае.

Пусть *N* - количество равновероятных вариантов сообщения, а *I* - количество единиц информации, содержащейся в этом со­ общении. В рассмотренных выше примерах предлагалась простей­ шая оценочная формула *I* = *N,* однако в теории информации при­ нята другая (производная от рассмотренной выше) количественная оценка:

*l=* logiN". (2.1)

Согласно этой оценке, количество информации, содержащейся в сообщениях «выбрана одна карта из возможных 32» и «выбрана одна карта из возможных 256», будет, соответственно, 5 и 8 единиц. Обобщим приведенные рассуждения. Пусть информационное со­ общение длиной *Х* знаков задано на некотором Р-ичном (Р-арном) алфавите (т. е. на алфавите, содержащем *Р* различных знаков). Тог­ да количество различных вариантов такого сообщения определится

по следующей формуле:

*N=PX.* (2.2)

Подставив выражение (2.2) в формулу (2.1), получим формулу для оценки количества информации:

 (2.3)

Как видно из формулы (2.3), количество информации пропорци­ онально длине сообщения *Х,* а коэффициент пропорциональности логарифмически возрастает с увеличением арности *Р* используемо­ го алфавита. Таким образом, сравнивая информативность различ-

**23**

ных алфавитов, можно сделать вывод о том, что чем «богаче» ал­ фавит, тем больше информации будет содержать заданное на этом алфавите сообщение (по сравнению с сообщениями такой же дли­ ны, заданными на более «бедных» алфавитах).

Учитывая тот факт, что количество информации, содержащейся в сообщении любой длины, заданном на унарном алфавите *(Р* = 1), тождественно равно нулю *(I =Х*- log21 = О), самым простым являет­ ся двоичный *(Р* = 2) алфавит, для которого формула (2.3) существен­

но упрощается:

 (2.4)

Как видим, количество информации, содержащейся в сообще­ нии, заданном на двоичном алфавите, равно длине этого сообще­ ния (отметим, что в случае с двоичным алфавитом энтропийный метод оценки сводится, по существу, к рассмотренному ранее объ­ емному методу).

*За единицу измерения информации принято количество инфор­ мации, содержащееся в самом коротком сообщении* (Х = 1), *заданном на самом простом (двоичном, Р* = 2) *алфавите:*

 (2.5)

Эта единица получила название *бит* (англ. ***Ьit*** - сокращение от двух английских слов: *Ьinary digit* - *двоичная цифра,* или *дво­ ичный разрядl).*

*Для оценки размеров информационных массивов, хранимых в за­ поминающих устройствах компьютеров, а также для оценки ин­ формационной емкости самих этих устройств используется другая (техническая) единица количества информации, называемая бай­* ***том.***

*Байт (byte)* - это минимальный размер ячейки памяти компью­ тера, имеющей уникальный адрес. В современных компьютерах 1 байт = 8 бит.

Соответственно, используются и более крупные единицы, про­ изводные от байта: килобайт, мегабайт, гигабайт, терабайт и пета­ байт:

1 Кбайт= 210 байт= 1024 байт;

1 Мбайт = 220 байт= 1048576 байт;

1 Гбайт = 2зо байт= 1 073 741 821 байт;

1 Существует и «настоящее» английское слово ***Ьit,*** переводимое на русский язык, как *«частица, малое количество чего-либо»* - очевидно, этот факт также был принят во внимание при выборе наименования единицы измерения информации.

**24**

1 Тбайт = 240 байт = 1 099 511 627 776 байт;

1 Пбайт = 250 байт =1125 899 906 842 624 байт.

В 1999 г. Международной электротехнической комиссией (МЭК) было введено иное наименование производных от байта единиц, чтобы отличать принятый в СИ префикс «кило-» (103 = 1000) от при­ нятого в информатике 210 = 1024. Согласно определению МЭК, еди­ ница, равная 210 байт = 1024 байта, называется «кибибайт» (КиБ, КiВ): префикс «киби-» получен от слов «кило» и «бинарный». Со­ ответственно, 210 КиБ = 1 МиБ (мебибайт), 210м иБ = 220 КиБ =

= 230 Б = 1 ГиБ (гибибайт) и т. д.

На практике применяются как традиционные наименования производных от байта единиц количества информации, так и наи­ менования МЭК.

##### 2.4. Информация, данные и знания -

**модель DIКW**

В программировании и прикладной информатике широко ис­ пользуются такие популярные в профессиональной среде слово­ сочетания, как *тип данных (data type), обработка данных (data processing), формат представления данных (data presentationformat), база данных (data base), большие данные (big data), интеллектуаль­ ный анализ данных (data mining),* в которых термин *данные* заменя­ ет близкий к нему по смыслу термин *информация.*

Данные - это набор разрозненных фактов, необработанный ма­ териал, который может стать источником некоторой информации. Используя термин *данные,* мы подчеркиваем техническую сторону *информации* и не затрагиваем ее содержательный и семантический (смысловой) аспекты.

Данные - это то, что может быть получено в результате измере­ ний или выполнения логико-математических операций и при этом представлено в форме, пригодной для хранения и последующей об­ работки. Например, в программировании отнесение каких-либо *данных* к определенному *типу* предполагает определенный *способ хранения* этих данных в памяти компьютера и определенный набор допустимых *операций* над этими данными.

«Данные» - это еще не «информация», информацию из данных надо еще извлечь, применяя для этого определенные методы аппа­ рата интерпретации. Одни и те же данные могут быть источником самой различной, в том числе и противоречивой, информации в за­ висимости от того, какие методы были применены для ее извлече­ ния. Несоответствие данных методам, применяемым для их обра-

**25**

ботки, или же низкое качество самих этих методов могут приводить и к тому, что объективно корректные данные могут стать источни­ ком недостоверной информации.

В качестве примера можно привести низкую достоверность ин­ формации прогнозного характера, полученной при обработке вы­ борки экспериментальных данных не вполне адекватным методом, что в результате привело к большой погрешности прогноза - либо по причине малого объема этой выборки, либо потому, что метод не учитывал наличия в выборке случайных выбросов.

Данные по своей природе могут быть как объективными - на­ пример, температура и влажность воздуха, атмосферное давление, направление и сила ветра, полученные с соответствующих метео­ датчиков, так и субъективными - например, оценки в школьном журнале, выставленные учителем. Информация же всегда субъек­ тивна, так как она получена путем обработки данных кем-то раз­ работанными (т. е. субъективными) алгоритмами интерпретации.

Информация формируется в результате анализа и выявления со­ отношений и взаимосвязей между разрозненными фактами (эле­ ментами «данных») и позволяет описать соответствующие события, процессы или явления, т. е. дать ответы на вопросы типа «Что?»,

«Кто?», «Где?», «Когда?», «Сколько?» и «Почему?». Например, анализ метеоданных за длительный период времени может позволить вы­ явить зависимости между направлением ветра, атмосферным дав­ лением и температурой воздуха, а сравнительный анализ данных школьного журнала позволит сформировать рейтинговые списки учеников, учителей, учебных предметов или их отдельных тем.

Информация, полученная в результате интерпретации данных, может послужить основанием для принятия какого-либо решения только в том случае, если она будет преобразована в соответствую­ щие знания.

Знания - это совокупность фактов, выявленных закономерно­ стей и правил, с помощью которых может быть решена поставлен­ ная задача. Знания получаются в результате синтеза полученной информации и человеческого разума, они формируются в процессе восприятия и усвоения результатов анализа разнородной взаимо­ связанной информации, полученной из множества источников дан­ ных, с учетом практического опыта людей, их способностей, интуи­ ции, убежденности и мотиваций.

Отметим основные свойства знаний, отличающие их от инфор­ мации:

* *структурированность* - знания должны быть представле­ ны множеством *взаимосвязанных* компонентов с целью обеспече­ ния их *эффективного усвоения* человеком (понять, запомнить или вспомнить забытое) или *эффективного доступа* к ним компьютер­ ными средствами;

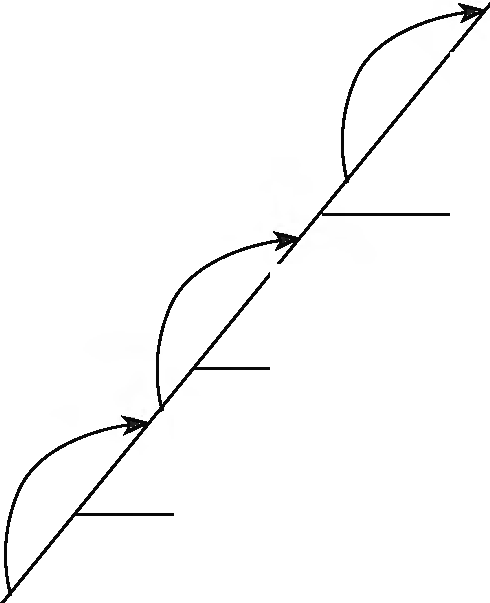
**26**

* *непротиворечивость* - знания не должны противоречить друг другу;
* *лаконичность* - знания должны быть минимально избы­ точными и не должны быть зашумленными, что позволяет быстро их осваивать и перерабатывать, повышая «коэффициент полезного содержания»;
* *процедурность* - знания нужны для того, чтобы их использо­ вать, т. е. применять к ним процедуры принятия решений, а также процедуры хранения, вывода и передачи знаний другим субъектам.

Существуют определенные трудности в понимании отличий и трактовке терминов *данные, информация* и *знания,* что вызвано их кажущейся синонимичностью. Даже известное в инженерии зна­ ний направление *Data Mining* по-разному переводится на русский язык, причем в различных переводах используются все три этих по­ нятия: «интеллектуальный анализ *данных»,* «извлечение *информа­ ции»* или «добыча *знаний».*

Несмотря на отмеченные различия, все эти понятия тесно связа­ ны и являются элементами единого процесса обработки данных, ил­ люстрируемого «информационной пирамидой» (рис. 2.2) и извест­ ного как «Модель *DIКW»: Data* - *Information* - *Knowledge* - *Wisdom.*

Wisdom Понимание­ объяснение, прогнозы поведения объекта и принятие решений



Кnowledge

Знания - инструкции по исследованию объекта

Information

Информация - описание взаимосвязей и отноше­ ний между элементами данных

Data

Данные - множество объективных фактов об исследуемом объекте

*Рис. 2.2.* **Информационная пирамида (модеnь DIКW)**

В основании пирамиды находятся *данные (data)* - факты о ре­ альном мире, полученные в результате наблюдений и измерений и сохраненные в форме, удобной для компьютерного хранения и последующей обработки.

Следующий уровень пирамиды занимает *информация (informa­ tion),* формируемая в процессе интерпретации данных путем выявле-

**27**

ния и осмысления взаимосвязей между их отдельными элементами.

*Информация,* в отличие от данных, несет в себе некоторый смысл.

Еще более высокий уровень - это *знания (knowledge),* которые получаются в результате целенаправленного восприятия и осмыс­ ления информации. Знания по своей природе процедурны и могут использоваться в процессах выработки и принятия решений.

Вершину информационной пирамиды занимают *глубокие зна­ ния, понимание* или *мудрость (wisdom).* На этом этапе обработки данных к знанию добавляется понимание того, где и как можно ис­ пользовать полученные знания, в том числе и за границами иссле­ дуемого процесса или явления.

На каждом новом уровне обрабатываемые данные становят­ ся более структурированными, процедурными и пригодными для использования в процессе принятия решений. Если информация *описывает* исследуемый объект и дает ответы на вопросы «Что?»,

«Сколько?» и «Почему?», знание - позволяет выработать техноло­ гические *инструкции* и получить ответы на вопрос «Как?», то пони­ мание дает нам *объяснение* и позволяет ответить на более сложные вопросы - «Зачем?», «Как лучше всего?».

Три нижних уровня пирамиды представляют прошлое и настоя­ щее - *данные, информация* и *знания* позволяют описать существу­ ющие процессы, явления, связи, и только *понимание* на верхнем уровне дает возможность «предсказания будущего» - т. е. позволя­ ет сделать обоснованные прогнозы.

При движении вверх по информационной пирамиде *объемы данных* преобразуются в *ценность знаний,* однако гарантией высо­ кой ценности знаний является не столько большой объем данных, сколько высокое качество методов их интерпретации, процедур об­ работки и использования информации.

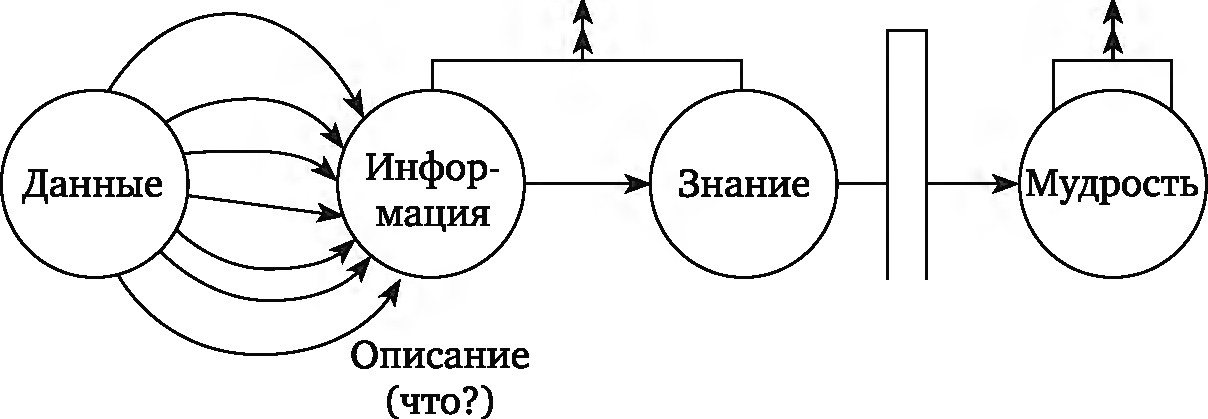
Еще одна иллюстрация модели DIКW приведена на рис. 2.3. Три нижних уровня пирамиды представляют прошлое и настоящее - данные, информация и знания позволяют описать существующие процессы, явления и связи между ними, и только понимание *(му­ дрость)* на верхнем уровне дает возможность сделать обоснован­ ные прогнозы на будущее.

Не всякие данные могут быть преобразованы в информацию, и не всякая информация сможет превратиться в знания. В каче­ стве примера можно рассмотреть текст научной статьи, написан­ ный на незнакомом читателю языке. С формальной точки зрения текст статьи - это данные, закодированные символами некоторого алфавита. При отсутствии аппарата интерпретации такие данные не смогут быть восприняты и осмыслены нашим читателем, т. е. не смогут быть преобразованы в информацию. С помощью аппара­ та интерпретации (в данном примере - словаря или переводчика, знающего оба языка) можно сделать корректный перевод статьи,

**28**

превратив тем самым данные в информацию, пригодную для вос­ приятия. Теперь статью можно прочитать, однако не факт, что эта информация сможет получить статус знаний и в дальнейшем будет использована для принятия решений - возможно, читатель недо­ статочно профессионально подготовлен для осмысления получен­ ной информации, или для понимания содержания статьи необходи­ мо привлечение дополнительной информации.

Делать правильно Делать то, что правильно



Инструкция

(как?)

Объяснение

(

зачем? что лучше

всего?)

Прошлое/настоящее

Будущее

*Рис. 2.3.* **Модеяь DIКW - от описания и анализа к прогнозированию**

Информация - это философская категория, и ее определение не может быть дано в рамках какой-то одной науки. Приведем сле­ дующее неформальное описание этого понятия.

Информация:

* это сведения о каком-то объекте, неизвестные ранее субъек­ ту - получателю информации;
* извлекается субъектом из соответствующих *данных;*
* носит субъективный характер и определяется предшествую­ щим уровнем *знаний* субъекта и способностью восприятия им но­ вой информации;
* пополняет знания субъекта, подтверждает, уточняет или опровергает его *понимание* определенных аспектов некоторой пред­ метной области.

В приведенном описании подчеркивается субъективный харак­ тер информации, ее целесообразность и значимость для получателя, а также взаимосвязь с другими (почти синонимичными) понятия­ ми - *данные, знания* и *понимание.*

Более формализованные определения данных, информации и знаний, а также определения некоторых производных понятий дает терминологический стандарт 1S0/IEC 23821:

1 ГОСТ 33707-2016 (ISOЛEC 2382:2015). Межгосударственный стандарт. **Ин­**

формационные технологии. Словарь.

**29**

**Данные** - представление информации в формальном виде, при­ годном для передачи, интерпретации или обработки людьми или компьютерами.

**Информация** - любой факт, понятие или значение, полученные из *данных,* а также контекст, выбранный из *знаний,* или контекст, ассоциированный со *знаниями.*

**Знания** - совокупность фактов, событий, убеждений, а также правил, организованных для систематического применения.

**Извлечение информации** - действия, методы и процедуры для получения *информации* о предмете из хранимых *данных.*

**База данных** - совокупность *данных,* хранимых в соответствии со схемой (моделью) данных, манипулирование которыми выпол­ няется в соответствии с правилами средств моделирования данных. **База знаний** - *база данных,* содержащая правила логичес ких выводов и *информацию* о человеческом опыте и *знаниях* экспертов

в некоторой предметной области.

**Экспертная система** - система обработки данных, которая обе­ спечивает экспертное решение проблем в заданной области приме­ нения, строя выводы на основе *базы знаний,* в которой формализо­ ван человеческий опыт.

Модель DIКW (см. рис. 2.2 и 2.3) наглядно демонстрирует тот факт, что данные - это «информационное сырье», а информация - это «обработанные данные в контексте». Пирамидальный образ мо­ дели наглядно демонстрирует общую концепцию преобразования данных в нечто действительно полезное, однако такое упрощенное представление создает ряд трудностей в понимании технологий практической реализации процедур такого преобразования:

* во-первых, данные считаются объективно существующими, но это не так, потому что данных в природе не существует, они должны быть созданы (или получены);
* во-вторых, данные - это средство представления фактов, они «заслоняют собой» реальные факты и в отрыве от контекста бесполезны, при этом контекст - это тоже данные (метаданные, данные о данных);
* в-третьих, линейная последовательность преобразования

*«данные* - *информация* - *знание* - *понимание»* игнорирует тот факт, что еще до начала такого преобразования нужны знания о том, как получить данные, затем - знания о том, как преобразо­ вать данные в информацию и т. д. до вершины пирамиды;

* в-четвертых, предполагается, что данные и информация отде­ лены друг от друга, тогда как в действительности два этих понятия тесно переплетены, взаимозависимы и по отдельности не существу­ ют: данные - это форма представления информации о реальном объекте, и состав накапливаемых данных зависит от того, какая ин­ формация об этом объекте будет востребована.

**30**

Взаимосвязи и соотношения между приведенными выше поняти­ ями в процессе их обработки иллюстрирует рис. 2.4

1 До обработки 1 Обработка 1 После обработки 1

tЭ 11 !, 11 9

**f**

11 Выборка 11 11 Организация 11

**t**  t

Обработка информации



Информация

11 Представление 11



Интеграция данных

**t t**

(данные Е---- \_о\_б \_

t

11 Измерение 11

\_даанбн\_ыох\_т\_к\_а ----- анные (

*Рис. 2.4.* **Данные, информация и знания в процессе их обработки**

##### 2.5. Информация и данные как объект управления

Информация является жизненно важным корпоративным акти­ вом, требующим непрерывного управления для извлечения из него практической выгоды - аналогично тому, как это делается и с про­ чими активами- материальными, кадровыми и финансовыми. Цен­ ность информации определяется эффективностью бизнес-решений, принимаемых по результатам ее использования, поэтому основная *цель управления информацией и данными (data management)* - *это обеспечение высокого качества данных и повышение ценности ин­ формационных активов на протяжении всего их жизненного цикла.* Продолжим цитировать определения из стандарта 1S0/IEC 2382.

**Управление информацией** - функции управления получением, анализом, сохранением, обновлением и распределением информации.

**31**

**Управление данными** - функции, обеспечивающие доступ к данным, реализацию и мониторинг хранения данных и управле­ ние операциями ввода-вывода.

Как видим, стандарт не дает четкого разграничения функций этих двух групп, указывая лишь на то, что функции управления дан­ ными реализуют задачи более низкого уровня и «вложены» в соот­ ветствующие высокоуровневые функции управления информацией. Сфера управления данными широка и разнообразна и не ограни­ чивается лишь ИТ-областью - к ней относится все, начиная с при­ нятия решений о способах извлечения ценности из имеющихся данных до технического развертывания и администрирования баз данных. При этом работы по управлению данными и ответствен­ ность за их результаты распределяется между менеджерами сферы бизнеса и ИТ-специалистами - распорядителями данных, аналити­ ками, администраторами и программистами, которые должны эф­ фективно сотрудничать и оперативно взаимодействовать, исполь­

зуя понятную и однозначно трактуемую терминологию.

В помощь профессионалам всех этих категорий Международной ассоциацией по управлению данными (Data Management Association, DАМА) в 2009 г. был разработан Свод знаний по управлению данны­ ми (Data Management Body of Кnowledge, DАМА DМВОК). В 2017 г. было выпущено второе издание свода знаний, в 2019 г. был опубли­ кован его русский перевод1. Книга написана ведущими аналитика­ ми в области управления данными и предоставляет обширный мате­ риал для специалистов, имеющих отношение к любому из аспектов управления данными (менеджерам, ИТ-специалистам, руководите­ лям, исследователям, преподавателям и студентам).

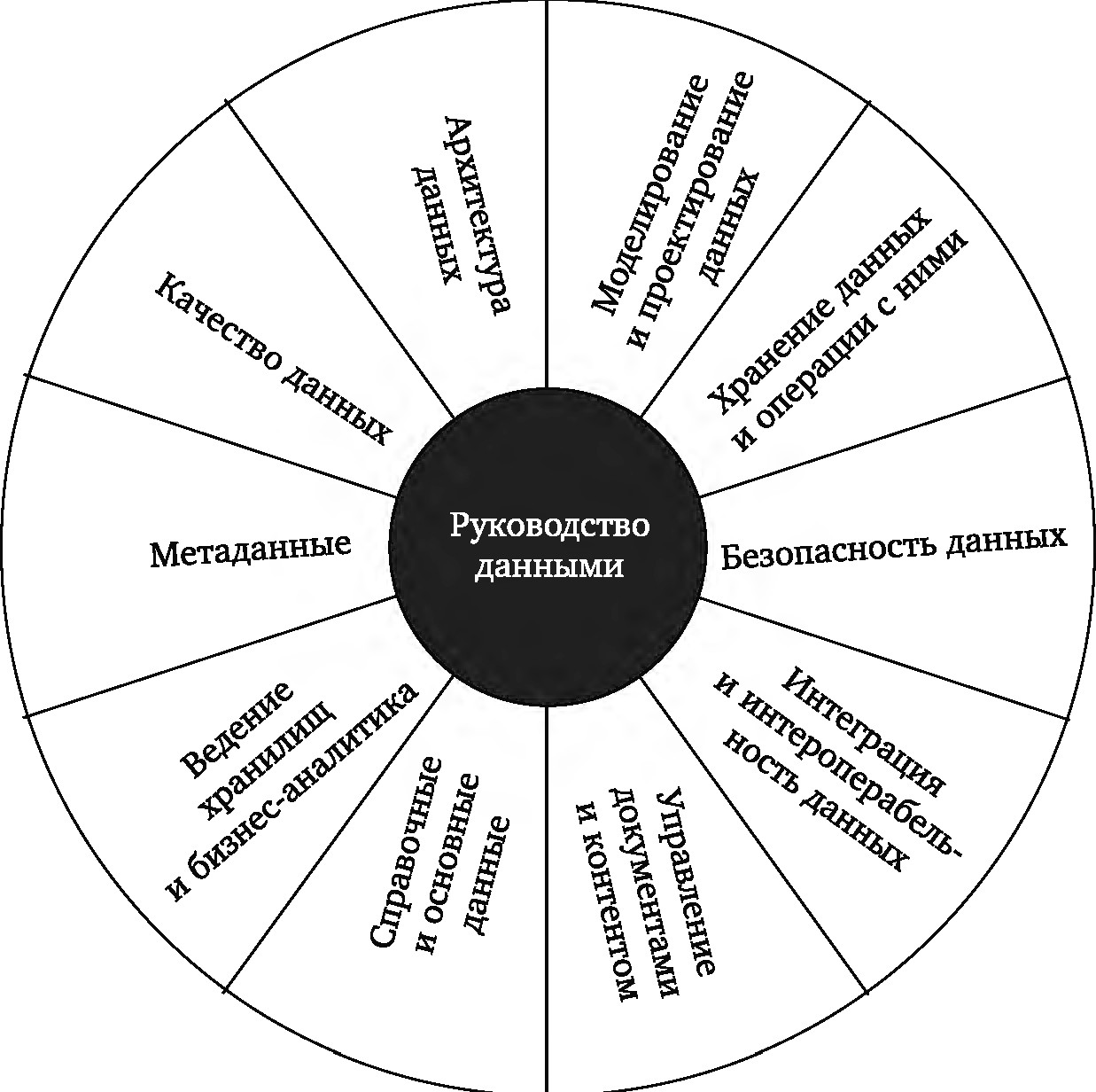
Согласно DAMA-DMBOK, система управления данными иллю­ стрируется представлением *«Колесо DАМА»* (рис. 2.5), определяю­ щим 11 функциональных областей, для каждой из которых описа­ ны цели и задачи, базовые понятия, а также используемые методы, инструменты и рекомендации по внедрению.

Центральное место в сфере управления данными отводится обла­ сти ***Руководство данными*** *(data government)* - осуществление ру­ ководящих и контрольных полномочий (планирование, мониторинг и обеспечение выполнения) в отношении всех остальных функцио­ нальных областей управления информационными активами.

**Архитектура данных** *(data architecture)* - определение концепту­ альных представлений об управлении информационными актива­ ми в соответствии со стратегией организации, а также выработка стратегических требований к данным и проектным решениям в об­ ласти данных.

1 DAМA-DMBOK: Свод знаний по управлению данными. 2-е изд. М. : Олимп-Биз­ нес, 2019.

**32**



*Рис. 2.5.* **«Копесо DAMA» - функциональная структура системы управпения данными**

**Моделирование и проектирование данных** *(data modeling* & *design)* - выявление, анализ и формулирование основных требова­ ний к данным с последующим их представлением и распростране­ нием в определенной форме.

**Хранение данных и операции с ними** *(data storage* & *opera­ tions)* - проектирование и реализация решений для хранения дан­ ных с целью получения максимальной выгоды от них на протяже­ нии всего их жизненного цикла.

**Безопасность данных** *(data security)* - планирование, разработ­ ка и осуществление политик и процедур, обеспечивающих аутен­ тификацию, авторизацию и доступ пользователей, а также аудит информационных ресурсов.

**Интеграция и интероперабельность данных** *(data integration* & *interoperaЬility)* -процессы, связанные с перемещением и консоли­ дацией данных как внутри хранилищ, приложений и организаций, так и в рамках обеспечения их взаимодействия.

**Управление документами и контентом** *(document* & *content ma­ nagement)* - планирование, реализация и контроль мероприятий по управлению жизненным циклом неструктурированных данных.

**33**

**Справочные и основные данные** *(referense* & *master data)* - со­ гласование и ведение критически важных совместно используемых данных для обеспечения скоординированного применения всеми информационными системами организации наиболее точной, пол­ ной и актуальной «версии правды» о ключевых бизнес-сущностях.

**Ведение хранилищ данных и бизнес-аналитика** *(data ware­ housing* & *business intelligence)* - планирование, реализация и кон­ троль процессов, обеспечивающих управление данными, исполь­ зуемыми для поддержки принятия решений, а также позволяющих специалистам извлекать ценность из данных.

**Метаданные** *(meta-data)* - планирование, реализация и кон­ троль деятельности по обеспечению доступа к высококачествен­ ным, интегрированным метаданным, включая определения, моде­ ли, описания потоков данных и другой информации, необходимой для понимания данных, а также систем, используемых для созда­ ния, ведения данных и доступа к ним.

**Качество данных** *(data quality)* - планирование и внедрение методических решений по управлению качеством данных, обеспе­ чивающих измерение, оценку и повышение качества, включая кон­ троль практической пригодности данных к использованию в орга­ низации.

Представление «Колесо DАМА» описывает состав функциональ­ ных областей системы управления данными, но не показывает ни связей между ними, ни хронологической последовательности разработки и внедрения соответствующих компонентов. Другое представление этой системы приведено на рис. 2.6, который ил­ люстрирует жизненный цикл данных в контексте решения задач управления данными.

Управление жизненным циклом

|  |  |
| --- | --- |
| Планирование и проектиро- вание  IA I | |
|  | |
| Моделирова- ние и про- ектирование данных |  |
|  | |

|  |  |  |
| --- | --- | --- |
| Обеспечение доступности и обслуживание  Хранение Ведение  и данных хранилищ  операции данных  сними  Интеграция Хранение  и операбель- больших  ность данных данных |  | Использование и улучшение  Бизнес- Наука  аналитика о данных  Использова- Монетизация ние основ- данных ныхданных  Управление Предиктив- документами ная анали- иконтентом тика |

*Рис. 2.6.* **Функции управnения данными на разных стадиях их жизненноrо цикnа**

|  |  |  |
| --- | --- | --- |
| Управление основными данными |  | Управление справочными данными |

**34**

Система управления данными организации не рождается вне­ запно - она целенаправленно планируется, проектируется и об­ служивается, постепенно наращивая свои возможности на пути к состоянию, в котором появляется возможность эффективного использования накопленной информации с применением передо­ вых информационных технологий, базирующихся на достижениях науки о данных *(data science),* - бизнес-аналитика, предиктивная аналитика в системах поддержки принятия управленческих реше­ ний и пр.

##### Контрольные вопросы

1. Перечислите внешние свойства информации, проявляющиеся в про­ цессе ее обработки получателем информационного сообщения. Приведите примеры недостоверной, нерелевантной, неполной и неэргономичной информации, а также возможные причины недоступности информации.
2. Какое из внешних свойств информации представлено следующим определением: «свойство информации, определяющее степень ее соответ­ ствия запросу получателя информационного сообщения»?
3. Какое из внешних свойств информации представлено следующим определением: «свойство информации, определяющее наличие (и количе­ ство) скрытых ошибок»?
4. Почему информационное взаимодействие называют несимметрич­ ным, в отличие от дРуrих (вещественных и энергетических) взаимодействий? Перечислите четыре основных фактора, присущих информационному вза­ имодействию.
5. Какие функции выполняют кодирующее и декодирующее устройства в процессе информационного взаимодействия (см. рис. 2.1)? Приведите

примеры.

1. Что общего между объемным и энтропийным методами оценки количества информации, содержащемся в информационном сообщении, и в чем их различие? Приведите примеры.
2. Что является основной единицей измерения количества информации?
3. Как влияет мощность (арность) используемого алфавита на инфор­ мативность сообщений? Сравните (по информативности) два сообщения одинаковой длины, заданных в бинарном и 16-арном алфавитах.
4. Расшифруйте аббревиатуру DIКW (см. рис. 2.2 и 2.3):

а) Как трактует стандарт 1S0/IEC 2382 понятия «данные», «информация» и «знания» и как связаны эти понятия?

б) Верно ли утверждение о том, что информация всегда субъективна, а данные могут иметь как субъективный, так и объективный характер? Ответ обоснуйте соответствующими примерами.

в) Перечислите свойства знаний, отличающие их от информации.

1. Как могут быть связаны данные, информация и знания в процессе их обработки (см. рис. 2.4)? Приведите примеры, иллюстрирующие ответы на следующие вопросы:

а) могут ли данные быть источником других данных; б) может ли информация быть источником знаний;

в) может ли информация быть источником данных, а данные - источ­ ником информации.

**35**

1. К какому из уровней информационной пирамиды DIКW следует от­ нести:

а) множество оценок, полученных студентами во время экзаменацион­ ной сессии;

б) рейтинговые списки студентов, построенные по результатам экзаме­ национной сессии;

в) принятие решения о поощрении студентов за отличные достижения в учебе, спорте или научной работе;

г) принятие решения об отчислении плохо успевающих студентов;

д) принятие решений о внесении изменений в учебные планы подготовки ИТ-специалистов (состав изучаемых дисциплин);

е) принятие решений о внесении изменений в программы изучаемых студентами дисциплин (содержание дисциплин и уровень требований к ка­ честву их освоения студентами)?

1. Расшифруйте аббревиатуру DMBOK (см. рис. 2.5 и 2.6):

а) Какая из функциональных областей занимает центральное место в сфере управления информацией/данными и какова роль этой функцио­ нальной области?

б) В рамках каких функциональных областей решаются задачи проек­ тирования и поддержки средств хранения данных?

в) В рамках каких функциональных областей решаются задачи защиты информационных ресурсов от несанкционированного доступа?

1. Как в ИТ-сфере трактуются следующие понятия:

а) структурированные и неструктурированные данные; б) метаданные;

в) большие данные *(big data);*

г) база данных *(data Ъаsе),* хранилище данных *(data warehou.se),* витрина данных *(data mart);*

д) наука о данных, бизнес-аналитика, предиктивная аналитика?

## Тема3 ПРЕДСТАВЛЕНИЕ ИНФОРМАЦИИ В КОМПЬЮТЕРНЫХ СИСТЕМАХ

##### Числовое кодирование данных

С точки зрения пользователя, не обремененного глубокими зна­ ниями в области информатики, современный компьютер - это универсальный гаджет, на который можно «скачать и установить» множество различных «приложений», которые помогут ему (поль­ зователю) писать, читать и рисовать, воспроизводить, обрабаты­ вать и записывать аудио-, фото- и видеоматериалы, обеспечивать телефонную и видеосвязь, осуществлять поиск информации в гло­ бальной сети и, разумеется, играть в разнообразные игры. Длина этого списка возможностей с каждым годом увеличивается, и где­ то в конце списка пользователь, конечно, упомянет и возможность считать (зачем-то же нужны калькулятор или MS Excel?).

Не будем разочаровывать не слишком продвинутого пользова­ теля, пусть он и дальше считает, что компьютер - это и писатель, и художник, и музыкант, и т. д., хотя на самом деле компьютер - это, в первую очередь, *computer,* т. е. *вычислитель,* и главная его возможность - это возможность выполнять арифметико-логиче­ скую обработку чисел.

В соответствии с одним из принципов фон Неймана (которые нам предстоит детально рассмотреть в следующей теме курса), лю­ бая информация (если более строго, то не информация, а данные) должна быть представлена в памяти компьютера в числовом двоич­ ном коде, а процесс обработки этой информации компьютерными программами (также представленными в виде двоично-закодиро­ ванных машинных команд) сводится к вычислительным процеду­ рам, реализуемым аппаратными компонентами компьютера.

Так, например, при редактировании текстового файла АSСП­ формата для преобразования заглавной буквы в строчную надо всего лишь прибавить 32 к числовому коду этой заглавной буквы, а процедура сортировки текстовых строк (расстановки их в алфа­ витном порядке) сводится к операции сравнения значений число­ вых кодов символов текста.

**37**

Другой пример: чтобы перекрасить текстовый символ, изобра­ женный на экране монитора, из синего цвета в зеленый, достаточно прибавить единицу к числовому коду атрибута этого символа и за­ писать измененный код в соответствующую ячейку видеопамяти компьютера. Чтобы перекрасить фон, на котором изображен тек­ стовый символ, из синего в зеленый цвет, к числовому коду атрибу­ та этого символа надо прибавить не единицу, а число 16.

Разумеется, для того, чтобы корректно обрабатывать информа­ цию, надо хорошо понимать систему ее числового кодирования (т. е. преобразования *информации* в *данные)* и представления дан­ ных в памяти компьютера.

Аппаратную основу электронных вычислительных устройств со­ ставляет двустабильный логический элемент - *mpuzzep,* выходной сигнал которого может принимать одно из двух устойчивых состо­ яний, одно из которых ассоциируется с числом О, а другое - с чис­ лом 1. Вследствие этого двоичный алфавит является естественным алфавитом для хранения и обработки информации в вычислитель­ ных устройствах, а двоичная система счисления положена в основу машинной арифметики и используется в качестве базовой системы счисления для технической реализации вычислительных процессов. Двоичная запись числа является крайне неэкономичной, поэто­ му для записи двоичных чисел часто используют шестнадцатерич­ ную систему счисления, в которой запись числа получается в четыре раза короче, а перевод чисел из двоичной системы в шестнадца­ теричную и обратно производится по очень простому алгоритму,

удобному даже для устного счета.

#### Системы счисления

**Определение 1. Система счисления** - это множество правил за­ писи и именования чисел и правил выполнения базовых арифмети­ ческих операций с этими числами.

**Определение 2. Базисные числа.** В любой системе счисления опре­ делен ограниченный набор базисных чисел, таких, что операции над ними позволяют вычислить значение любого другого числа в данной системе счисления.

**Определение 3. Цифра** - это знак (графический символ), исполь­ зуемый для записи базисного числа в некоторой системе счисления. В соответствии с данным определением количество цифр и количе­ ство базисных чисел одной системы счисления должны совпадать.

**38**

Цифровая запись числа в определенной системе счисления одно­ значно определяет его значение, однако способы вычисления значе­ ний чисел, записанных в системах счисления разных типов, также будут различными.

* + 1. Аддитивные системы счисления

В ***аддитивных*** системах счисления (от англ. *add* - прибавить, сложить) значение числа определяется суммированием и (или) вы­ читанием базисных чисел, представленных в записи этого числа соответствующими цифрами. Классические примеры - унарная и римская системы счисления.

В ***унарной*** системе счисления, хорошо знакомой каждому под названием «счетные палочки», определены единственное базисное число «один» и соответствующая ему цифра «единица», обозначае­ мая символом **«1».** Значение десятичного числа 8 в унарной системе

будет записано как 11111111; операция сложения 2 + 3 = 5 - как 11 +

+ 111 = 111111; операция вычитания 5 - 4 = 1-как 11111- 1111 =

= 1; операция умножения 2 • 3 = 6 - как 11 + 11 + 11 = 111111 или как 111 + 111 = 111111. Такая система счисления очень легка в осво­ ении, алгоритмы выполнения арифметических операций - просты

и очевидны, что следует даже из их наименований («прибавить»,

«отнять», «умножить»). Очевиден также и существенный недоста­ ток унарной системы счисления - она непригодна для оперирова­ ния с большими числами.

Другая аддитивная система счисления, получившая название

***«римской»,*** обеспечивает более компактную (по сравнению с унар­ ной системой) запись чисел, так как использует большее количество базисных чисел и, соответственно, большее количество цифр для их обозначения. В качестве цифр система использует буквы латин­ ского алфавита: цифра I обозначает единицу, V - пять, Х - десять, L - пятьдесят, С - сто, D - пятьсот, М - тысячу. При записи чисел в римской системе счисления значением числа является алгебраи­ ческая сумма базисных чисел, представленных в записи числа со­ ответствующими цифрами. При этом не любая последовательность цифр в записи числа является корректной, например:

* не разрешается записывать рядом более трех одинаковых цифр;
* если три левые крайние цифры в записи числа не одинаковы, то крайняя левая цифра должна иметь большее значение, чем со­ седняя с ней правая цифра;
* цифры записывают, как правило, в порядке убывания значе­ ний базисных чисел слева направо.

Если число записано корректно, то вычисление его значения производится по следующему алгоритму (табл. 3.1).

**39**

1. Начальное значение числа полагают равным нулю. Начинают последовательную обработку записи числа с крайней левой цифры.
2. Если очередная обработанная цифра - последняя (крайняя правая) в записи числа, то:
   * если вычисленное ранее текущее значение числа равно нулю, значение числа полагают равным значению очередной обработан­ ной цифры (в этом случае эта цифра - единственная в записи чис­ ла) и работа алгоритма заканчивается;
   * в противном случае вычисленное ранее текущее значение числа считается окончательным результатом и заканчивается рабо­ та алгоритма.
3. В противном случае сравнивают значения текущей цифры и соседней с ней правой цифры.
4. Если значения цифр в паре одинаковы, вычисляют новое те­ кущее значение числа как сумму предыдущего текущего значения и значения правой цифры в паре и далее переходят к п. 2 алгоритма (другими словами, значения всех одинаковых соседних цифр сум­ мируются).
5. Если значение очередной (правой) цифры не совпадает со значением левой (уже обработанной) цифры, то:
   * если очередная цифра - последняя (крайняя правая) в запи­ си числа, ее значение суммируется с накопленным текущим значе­ нием числа, и заканчивается работа алгоритма;
   * в противном случае очередная цифра вместе с соседней с ней правой цифрой образуют новую пару.
6. В случае, когда левая цифра пары представляет меньшее базисное число, чем правая, ее значение вычитается из значения правой цифры, в противном случае оба значения складываются, и результат вычисления суммируется с предыдущим текущим зна­ чением числа.
7. Далее выполняется п. 2 алгоритма.

*ТаблицаЗ.1*

**Примеры «римской» записи десятичных чисеn**

|  |  |  |
| --- | --- | --- |
| **Запись числа** | | **Алгоритм вычисления десятичного значения числа** |
| **Римская система** | **Десятичная система** |
| с | 100 | [100] |
| *N* | 4 | [5 - 1] |
| CCLXIV | 264 | [100 + 100] + [50 + 10] + [5 - 1] |
| CCCXCVII | 397 | [100 + 100 + 100] + [100 - 10 ] + [5 + 1] + [1] |
| DCXLIX | 649 | [500] + [100] + [50 - 10] + [10 - 1] |
| MCCVII | 1207 | [1000 + 100] + [100 + 5] + [1 + 1] |

**40**

|  |  |  |
| --- | --- | --- |
| **Запись числа** | | **Алгоритм** *вычислен-Nffнчаниетабл.З.1*  **десятичного значения числа** |
| **Римская система** | **Десятичная система** |
| IN | Запись является некорректной, так как не выполняется ограни- чение № 2, корректная запись - III ([1 + 1 + 1]) | |
| CCLXIV  хххх | Запись является некорректной, так как не выполняется ограни- чение № 1, корректная запись - XL ([50 - 10]) | |

Недостатком римской системы является отсутствие строго фор­ мализованных правил записи и интерпретации чисел, что затруд­ няет выполнение арифметических действий с многозначными числами. Например, число 649 можно записать двумя способами, не нарушая установленных правил записи: и как DCXLIX ([500] +

+ [100] + [50 - 10] + [10 - 1]), и как DCIL ([500] + [100] + [50 - 1]).

* + 1. Позиционные системы счисления

В позиционных системах счисления, называемых также *адди­ тивно-мулътипликативными* (от англ. *Add* - прибавить, сложить, *Миltiрlу-умножить),* значение числа определяется комбинацией операций сложения и умножения базисных чисел, представленных в записи этого числа соответствующими цифрами.

Классические примеры - десятичная, двоичная, восьмеричная и шестнадцатеричная системы счисления. Все эти системы являют­ ся ***позиционными,*** так как вклад каждого базисного числа, пред­ ставленного в записи числа соответствующей цифрой, определяется не только его значением, но и номером позиции (разряда), в кото­ рой записана эта цифра.

**Десятичная система счисления**

Формула (3.1) иллюстрирует алгоритм вычисления значения десятичного числа *N*10 по его цифровой записи. Значение числа получается суммированием произведений базисных чисел, пред­ ставленных в записи этого числа соответствующими цифрами, на их позиционные веса:

 (3.1)

где *i* - номер позиции, в которой записана цифра, представляющая базисное число. Позиции записи целой части числа нумеруются справа налево, начиная от разряда единиц *(i* = О), затем - десятков

*(i* = 1) и т. д. до *i* = *п* - 1, где *п* - количество разрядов в целой части

числа. Позиции записи дробной части числа нумеруются слева на­ право, начиная от разряда десятых *(i* = -1), затем - сотых *(i* = -2)

**и** т. д. до *i* = *-т,* где *т* - количество разрядов в дробной части чис­

ла; *bi* - значение базисного числа (от О до 9), записанное в i-й по-

**41**

зиции; 10 - основание десятичной системы счисления; *10i* - «по­ зиционный вес» - числовое значение каждой единицы базисного числа, записанного в *i-й* позиции.

Рассмотрим пример записи и интерпретации значения числа 1234,56 в десятичной системе счисления:

1234,56 = 1 • 103 + 2 • 102 + 3 • 101 + 4 • 10° + 5 • 10-1 + 6 • 10-2.

Число 10 является *основанием* десятичной системы счисления (от которого эта система и получила свое название):

* в десятичной системе каждые 10 *единиц младшего разряда объединяются в одну единицу соседнего старшего разряда:* 10 еди­ ниц= 1 десяток, 10 десятков= 1 сотня и т. д.;
* в десятичной системе определено 10 *базисных чисел:* «ноль»,

«один», «два», ... «девять»;

* в десятичной системе используется 10 *цифр* (графических символов «О», «1», «2», ... , «9») для записи соответствующих базис­ ных чисел.

Все сказанное выше для десятичной системы можно распростра­ нить и на любую другую позиционную k-ичную систему счисления:

(3.2)

Здесь:

* основание системы счисления равно числу *k;*
* *k* единиц младшего разряда числа объединяются в одну еди- ницу старшего разряда;
* определено ровно *k* базисных чисел со значениями от О

ДО *k-1;*

* для записи базисных чисел используется ровно *k* цифр;
* позиционный вес *Pi* для *i-й* позиции *(п* - 1 *i -т)* вычис- ляется как *ki,* где *пи m* - количество разрядов в целой и дробной частях k-ичной записи числа;
* значение числа по его записи вычисляется как сумма произ­ ведений базисных чисел, представленных в записи числа соответ­ ствующими цифрами, на их позиционные веса.

**Двоичная система счисления**

Основание двоичной системы - число 2 (заметим, что это число здесь записано не в двоичной, а в привычной нам десятичной систе­ ме счисления).

Базисные числа в этой системе счисления - число «ноль» и чис­ ло «один», обозначаемые цифрами О и 1 в записи двоичных чисел. При поразрядном сложении 2 единицы младшего разряда объеди­ няются в одну единицу старшего (действуют все правила сложения/

**42**

вычитания «в столбик» - перенос единицы в старший разряд при сложении или заем из старшего разряда при вычитании):



Для вычисления десятичного значения двоичного числа приме­ няется описанная ранее общая формула, например:

100110011,1012 =

=1-28 +О-27 +О-26 +1-25 +1-24 +

+О-23 +О-22 +1-21 +1-2° +1-2-1 +0-2-2 +1-2-з =

= 256+32+ 16+2 +1 +О,5+0,125 = 307,62510.



Для двоичной системы счисления расчетную формулу для определения числового значения можно упростить - суммированию подлежат пози­ ционные веса только тех позиций, в которых записана единица, а осталь­ ные позиции - игнорируются.

Заметим, что все вычисления производились в десятичной систе­ ме счисления, и полученный результат вычислений - десятичное число. Следовательно, рассмотренная выше формула может быть положена в основу алгоритма перевода чисел из двоичной системы счисления в десятичную и обратно.

* **ФiФi+М**

Переведем число 1234,12510 в двоичную систему счисления. Для этого

представим исходное число в виде суммы слагаемых - максимально воз­ можных целочисленных степеней двойки, начиная с наибольшего:

1234,125 = 1024(210) + 128(27) + 64(26) + 16(24) + 2(21) + 0,125 (2-3),

и затем сформируем двоичную запись числа, записав единицы в десятом, седьмом, шестом, четвертом, третьем, первом и «минус третьем» разря­ дах и нули - во всех остальных промежуточных разрядах. В результате

получим: 1234,125 = 10011010001,0012.

**Восьмеричная система счисления**

Основание восьмеричной системы - число 8.

В этой системе счисления восемь базисных чисел: ноль, один, два, три, четыре, пять, шесть и семь, и восемь цифр, заимствован­ ных из десятичного алфавита: О, 1, 2, ... , 7.

**43**

При поразрядном сложении 8 единиц младшего разряда объ­ единяются в одну единицу старшего, при поразрядном вычитании заемная единица старшего разряда рассматривается как 8 единиц младшего:

lв +2в = 3в; 48 +58 =118;

38 +48 =78;

238 + 568 =1018;

48 +48 =108;

568 - 78 = 4 7 *в·*

Для вычисления десятичного значения восьмеричного числа (пе­ ревод из восьмеричной системы счисления в десятичную) применя­ ется описанная ранее общая для всех позиционных систем формула:

1234,568 =

=1-83 +2-в2 +3-81+4-8° +5-в-1+6-в-2 =

=1-512+2-64+3·8+4·1+5/8+6/64=

= 512+128+24+4+0,625+0,09375= 668,7187510•

Для обратного перевода из десятичной в восьмеричную систему надо представить исходное десятичное число в виде суммы произ­ ведений базисных чисел восьмеричной системы счисления на цело­ численные степени восьмерки и затем записать эти числа в соот­ ветствующих позициях. Например:

58,12510= 7 • *в1* + 2 • *в0* + 1 • в-1,

следовательно, в первой позиции восьмеричного числа будет запи­ сана цифра 7, в нулевой- цифра 2, а в «минус первой» позиции - цифра 1:

58,12510= 72,18.

Для перевода из двоичной системы счисления в восьмеричную можно использовать уже знакомую формулу, однако при этом сле­ дует нумеровать позиции и выполнять все вычисления в восьмерич­ ной системе счисления:

100110011,1012 = 210 + 2s + 24 + 21 + 20 + 2-1 + 2-з =

= 4008 + 408 + 208 + 28 + 18 + 4/108 + 1/108 = 463,58.

**Шестнадцатеричная система счисления**

Основание шестнадцатеричной системы - число 16 (записан­ ное здесь в десятичной системе счисления).

В этой системе шестнадцать базисных чисел: *ноль, один, два, три, четыре,* ... , *девять, десять, одиннадцать,* ... , *пятнадцать* (также записанных здесь в десятичной системе), и шестнадцать

**44**

цифр, первые десять из которых заимствованы из десятичного циф­ рового алфавита: О, 1, 2, 3, ... , 9, а недостающие шесть- из латин­ ского буквенного: А, В, С, D, Е и F.

При поразрядном сложении 16 единиц младшего разряда объ­ единяются в одну единицу старшего, при поразрядном вычитании

«заемная» единица старшего разряда рассматривается как 16 еди­ ниц младшего:

116 +216 = 316;

Е16 + 216 =1016;

816 +216 =А16;

fi6 +fi6 = lE16;

В16 +216 = D16; 2016 -116 =lfi6•

Для вычисления десятичного значения шестнадцатеричного чис­ ла (перевод из шестнадцатеричной в десятичную систему) приме­ няется описанная ранее общая формула (все числа и вычисления - в десятичной системе).

Пример перевода шестнадцатеричного числа 1234,CF16 в деся­ тичную систему счисления:

1234,CFi.6 =

=1-163 +2-162 +3-161+4·16° +12-16-1+15-16-2 =

=1·4096+ 2·256+3·16 + 4·1 +12/16+15/ 256= 4096+512+48+4+0,3125+0,01953125 = 4660,8085937510·

Для обратного перевода из десятичной в шестнадцатеричную си­ стему надо представить исходное десятичное число в виде суммы произведений базисных чисел шестнадцатеричной системы счисле­ ния на целочисленные степени числа 16 и затем записать эти числа в соответствующих позициях.

Например:

58,12510 = **3** • 161+ 10 • 160+ 2 • 16-1,

следовательно, в первой позиции шестнадцатеричного числа будет записана цифра 3, в нулевой позиции - цифра А, а в «минус пер­ вой» позиции -цифра 2. В результате получим: 58,12510 = 3А,216. Для перевода из двоичной системы счисления в шестнадцатерич­ ную можно использовать уже знакомую формулу, однако при этом следует иметь в виду, что нумеровать позиции и выполнять все вы­

числения следует в шестнадцатеричной системе счисления:

100110011,1012= *2в+* 2s + 24+ 21+ 20+ 2-1+ 2-з =

= 10016+ 2016+ 1016+ 216+ 116+ 8/1016+ 2/1016 = 133,А16.

Таблица 3.2 иллюстрирует представление чисел в позиционных системах счисления: десятичной, двоичной, восьмеричной и шест­ надцатеричной.

**45**

*ТаблицаЗ.2*

**Представление чисея впозиционных системах счисяения1**

|  |  |  |  |
| --- | --- | --- | --- |
| **Десятичная** | **Двоичная** | **Восьмеричная** | **Шестнадцатерич- ная** |
| о | 0000 | о | о |
| 1 | 0001 | **1** | **1** |
| 2 | 0010 | 2 | 2 |
| 3 | 0011 | 3 | 3 |
| 4 | 0100 | 4 | 4 |
| 5 | 0101 | 5 | 5 |
| 6 | 0110 | 6 | 6 |
| 7 | 0111 | 7 | 7 |
| 8 | 1000 | 10 | 8 |
| 9 | 1001 | 11 | 9 |
| 10 | 1010 | 12 | **А** |
| 11 | 1011 | 13 | в |
| **12** | 1100 | 14 | с |
| 13 | 1101 | 15 | D |
| 14 | 1110 | 15 | Е |
| 15 | 1111 | 17 | F |
| 21= 2 | 10 | 2 | 2 |
| 22=4 | 100 | 4 | 4 |
| 23=8 | 1000 | 10 | 8 |
| 24= 16 | 10000 | 20 | 10 |
| 25=32 | 100000 | 40 | 20 |
| 26=64 | 1000000 | 100 | 40 |
| 27= 128 | 10000000 | 200 | 80 |
| 28= 256 | 100000000 | 400 | 100 |
| 29= 512 | 1000000000 | 1000 | 200 |
| 210= 1024 | 10000000000 | 2000 | 400 |
| 211= 2048 | 100000000000 | 4000 | 800 |
| 212=4096 | 1000000000000 | 10000 | 1000 |
| 220 = **1** 048 576 |  | 4 ООО ООО | 100 ООО |
| 2зо=1 073 741 824 |  | 10 ООО ООО ООО | 40 ООО ООО |

1 Содержимое этой таблицы (как минимум, ее первые 17 строк) придется вы­ учить наизусть и помнить так же (хорошо), как школьную таблицу умножения.

**46**

Анализ числовых данных, приведенных в табл. 3.2, позволяет сделать ряд полезных замечаний:

1. чем больше основание системы счисления, тем компактнее в ней запись числа;
2. десятичные числа, представляющие целые степени двойки, записываются в двоичной, восьмеричной и шестнадцатеричной си­ стемах круглыми числами, что является следствием того, что осно­ вания восьмеричной и шестнадцатеричной систем счисления сами являются целыми степенями двойки;
3. двоичная запись десятичных чисел, представляющих целые степени двойки, всегда содержит единицу в старшем бите и нули во всех остальных битах, количество которых равно степени двойки соответствующего десятичного числа;
4. для двоичной записи любого одноразрядного (базисного) восьмеричного числа потребуется не более трех битов (8 = 23);
5. для двоичной записи любого одноразрядного (базисного) десятичного или шестнадцатеричного числа потребуется не более четырех битов (23 < 10 < 24, 16 = 24).

Смешанные системы счисления

Смешанная система, по существу, не является системой счисле­ ния - скорее, это просто удобный способ записи чисел одной си­ стемы счисления с использованием набора цифр другой системы.

Смешанными (Р-Q-ми) называют системы, в которых для записи чисел, заданных в одной позиционной системе счисления (с основа­ нием, равным Q), используют набор цифр другой позиционной си­ стемы счисления (с основанием, равным *Р).* При этом предполага­ ется, что *Р* < *Q,* и каждое базисное число, представленное в Q-ичной системе счисления, записывается фиксированным количеством раз­ рядов числа в Р-ичной системе с сохранением порядка следования разрядов.

**МФ\ЩЩ.f -11 ------------------------**

**представление десятичных чисел**

**в двоичной и двоично-десятичной системах**

Например, десятичное число 307,62510 будет представлено в двоично­ десятичной *(Р* = 2, *Q* =10) системе следующим образом:

307,62510 = 00110000 0111, 0110 0010 01012-10·

Как видим, для записи каждого базисного десятичного числа исполь­ зуется соответствующее ему четырехразрядное двоичное число.

Процедура восстановления десятичного числа по его двоично-десятич­ ной записи очень проста: необходимо последовательно (справа налево для целой части числа и слева направо - для дробной) каждую четверку битов преобразовать в соответствующее одноразрядное десятичное число.

**47**

Заметим, что двоичный эквивалент исходного десятичного числа не совпадает с его двоично-десятичной записью:

307,62510 =100110011,1012 *-:f.*110000 0111,0110 0010 01012-10·

Рассмотрим еще несколько примеров с двоично-восьмеричными и двоично-шестнадцатеричными числами.

### ФiФШ-f◄

**Представление восьмеричных чисел**

**в двоичной и двоично-восьмеричной системах**

Переведем двоичное число 100110011,1012в восьмеричную систему счисления:

100110011,1012 = 210+ 2s + 24 + 21+ 20 + 2-1+ 2-з =

= 4008 + 408 + 208 + 28 + 18 + 4/108 + 1/108 = 463,58•

Запишем полученное восьмеричное число 463,58 в двоично-восьме­ ричной системе счисления - каждый разряд исходного числа будем запи­ сывать соответствующим трехразрядным двоичным числом:

463,58 =100 110 011,1012\_8 .

Заметим, что двоичная и двоично-восьмеричная записи восьмерич­ ного числа 463,58 оказались одинаковыми (в отличие от рассмотренного выше примера с десятичным числом).

* **,iФФУ.i**◄

**Представление шестнадцатеричных чисел**

**в двоичной и двоично-шестнадцатеричной системах**

Представим двоичное число 100110011,1012в шестнадцатеричной системе счисления:

100110011,1012= 2в + 2s + 24 + 21 + 20 + 2-1 + 2-з =



Запишем полученное шестнадцатеричное число 133,А16 в двоично­ шестнадцатеричной системе счисления - каждый разряд исходного числа будем записывать соответствующим четырехразрядным двоичным чис­ лом:

133,А16= 000100110011,10102\_16.

Как видим, двоично-шестнадцатеричная и двоичная записи этого шестнадцатеричного числа совпали, так же, как это было и в случае с восьмеричной системой, и это не простое совпадение, а следствие определенного соотношения оснований рассмотренных

**48**

систем счисления (2, 8 и 16). Примем (без доказательства) следую­ щее утверждение.

Если в паре позиционных систем счисления основание одной из них (Q) является целочисленной степенью основания другой *(Р),* то Q-ичная запись любого Р-ичного числа будет совпадать с его за­ писью в смешанной (Р-Q)-ичной системе.

В случае с двоично-десятичной записью число 10 не является целой степенью числа 2, а для двоично-восьмеричной и двоично­ шестнадцатеричной систем указанное соотношение выполняется, следовательно, можно смело применять простую процедуру записи двоичных чисел в смешанных двоично-восьмеричной и двоично­ шестнадцатеричной системах счисления для перевода восьмерич­ ных и шестнадцатеричных чисел в двоичную систему и обратного перевода двоичных чисел в восьмеричную и шестнадцатеричную системы.

Учитывая простоту такого перевода, а также тот факт, что по сравнению с двоичной шестнадцатеричная запись чисел в 4 раза более компактна, становится понятной популярность шестнадца­ теричной системы счисления среди компьютерных специалистов (а также среди разработчиков инструментальных средств, предна­ значенных для компьютерных специалистов) для использования в качестве удобного средства краткой записи двоичных чисел.

Дело в том, что любая информация, в какой бы форме она не предъявлялась пользователю различными программными прило­ жениями, в памяти компьютера представлена в числовом двоичном коде и обрабатывается вычислительными устройствами в двоичной системе счисления. Однако двоичный код крайне неэффективен для визуализации компьютерных данных, их анализа и обработки че­ ловеком, в том числе и по причине большой длины такой записи.

Шестнадцатеричная запись широко применяется для представле­ ния двоичных чисел в технической документации, научных публи­ кациях и учебной литературе по компьютерной тематике, многие инструментальные программные средства (в том числе и програм­ мы-анализаторы памяти компьютера, рекомендованные к примене­ нию в данном курсе при выполнении лабораторных работ) исполь­ зуют шестнадцатеричный формат представления двоичных чисел.

##### Двоичное кодирование текстовой информации

Необходимость стандартизации в области обмена текстовыми документами очевидна и явно осознавалась человечеством с древ­ нейших времен: любой письменный язык с его алфавитом, набо-

**49**

ром синтаксических правил и правил орфографии - это своеобраз­ ный стандарт, обеспечивающий возможность информационного обмена между людьми, владеющими этим языком. С появлением технических средств приема-передачи текстов (например, телегра­ фа), а позднее - и компьютерных систем обработки и хранения информации, интенсивность информационных обменов возросла тысячекратно, и проблема стандартизации в этой области получила международный характер.

Наличие стандарта числового кодирования текстовых символов позволяет хранить тексты в памяти компьютеров, обрабатывать (редактировать) и визуализировать их программными средства­ ми, т. е. преобразовывать коды символов в их графические образы. Один из способов визуализации *(знаког.енерации)* символов будет рассмотрен в следующей теме курса при обсуждении алгоритмов работы видеосистемы ПК в текстовых видеорежимах.

* + 1. **Стандарт ASCII**

Стандарт **ASCII** *(American Standard Code for Information Interchange)* широко используется разработчиками компьютерной техники, си­ стемного и прикладного программного обеспечения.

Современные версии стандарта базируются на 8-битной системе кодирования символов некоторого алфавита (1 символ - 1 байт), что позволяет одновременно поддерживать алфавит мощностью

не более 256 символов (28 = 25610 или lООь), символы которого мо­

гут иметь коды в диапазоне от 00 до FFь. Такого количества симво­

лов достаточно для представления алфавитов любых двух европей­ ских языков (строчные и прописные буквы), арабских цифр, знаков пунктуации, знаков математических операций и некоторых специ­ альных символов.

Стандарт предлагает набор из 256 символов (табл. 3.3), каждо­ му из которых поставлен в соответствие двоичный код. Все кодовое пространство разделено на два равных диапазона, называемых *ко­ довыми таблицами: основная кодовая таблица* содержит 128 сим­ волов с кодами в диапазоне от 00 до 7F; *дополнительная кодовая таблица* содержит 128 символов с кодами в диапазоне от 80ь до FFь. При этом предполагается, что основная таблица присутствует всег­ да, а дополнительных таблиц может быть много, все они спроециро­ ваны на единое кодовое пространство, но лишь одна из них может быть активной.

Основная кодовая таблица содержит коды букв латинского алфа­ вита, цифр, знаков пунктуации знаков математических операций, а также некоторых специальных символов.

Основное назначение дополнительной кодовой таблицы - коди­ рование символов какого-либо (одного!) национального алфавита и так называемых символов псевдографики - специальных зна-

**50**

ков, предназначенных для отображения границ таблиц (здесь надо не забывать о том, что стандарт активно использовался во времена матричных принтеров и текстовых режимов работы видеоадапте­ ров и экранов компьютерных мониторов).

*ТаблицаЗ.З*

**Кодовые таблицы стандарта 8-битного кодирования символов**

|  |  |  |  |
| --- | --- | --- | --- |
| **Основная кодовая таблица**  **ASCII** | | **Дополнительная кодовая таблица**  **(кирИJDiица)** | |
| **Дна- пазов**  **КОДОВ** | **Набор символов** | **ДИа-**  **пазов**  **КОДОВ** | **Набор символов** |
| 00-lF | Управляющие символы:  *нуль-символ,* Q, • ' ., ♦, .,  •, •, **а,** 0, ■, *c"S,* ' ***f,* /.1,** ◊,  ►, ◄, t,i, !!, !!, ,r, §, -, l,  i, Ф, , f-, L, Н | 80---9F | Прописные буквы (кириллица):  А, Б, В, Г, Д, Е, Ж,... **Я**  (кроме Ё) |
| 20-2F | Знаки пунктуации:  *пробел,* !, ", #, $, %, &, ',  (, ), \*, +, " -, ., / | AO-AF | Строчные буквы (кириллица  от адо п):  а, б, в, г, д, е, ж,... п (кроме буквы ё) |
| 30-39 | Арабские цифры:  о, 1, 2, 3, ..., 9 | B0---DF | Символы псевдографики:  ..,,1 =J  - ' '1 идр. |
| 3А--40 | Знаки:  :, ;, <, =, >, ?, @ | EO---EF | Строчные буквы (кириллица  от **рдоя):**  р, с, т, у, ф, х, ц, ... я |
| 41-SA | Прописные буквы (латиница):  А, В, С, D, ... , **Z** | FO---Fl | Буквы Ёи ё (кириллица) |
| 5В-60 | Знаки:  [, \, ], **л** '\_, ' | F2-F7 | Буквы славянских алфавитов, отсутствующие  в русском алфавите :  €, €, 1, i, У, у |
| 61-7А | Строчные буквы (латиница):  а, Ь, с, d, ... , z |
| 7B-7F | Знаки:  {, 1, }, ~, | *FB-FF* | Знаки:  a, ., **·,V,№,):{,** • |

Информацию о содержании кодовых таблиц можно легко полу­ чить, используя любой текстовый редактор: для этого надо последо­ вательно вводить коды символов (в десятичной системе счисления), нажимая на клавиши «цифровой клавиатуры» и удерживая нажатой

**51**

клавишу Alt - после отпускания клавиши Alt на экран будет выве­ ден символ, соответствующий набранному коду.

Американский национальный стандарт ASCII, ставший де-факто международным (по вполне понятным причинам), был далеко не единственным стандартом кодирования текстовых данных - одновременно с ним использовались и многие другие стандарты 8-битного кодирования символов (например, советский стандарт КОИ-В).

Все эти стандарты структурно были одинаковы, отличаясь только соответствием символов алфавитов их двоичным кодам, и при этом все они имели общий недостаток - этих кодов (и, соответствен­ но - символов) было всего 256 (28), из которых половина всегда была «занята» символами латиницы, цифрами и другими «между­ народными» знаками, и только 128 кодов оставалось для символов какого-то другого алфавита.

Несомненным достоинством всех этих 8-битных стандартов была высокая экономичность хранения текстовых данных, что было совсем немаловажно в эпоху, «когда компьютеры были большими, а запоминающие устройства - маленькими», но со временем про­ явились два существенных недостатка 8-битного кодирования сим­ волов.

Во-первых, компьютер оказывался только «двуязычным», и было весьма проблематично хранить и обрабатывать на одном компью­ тере текстовые документы, написанные более, чем на двух языках (точнее - более, чем на одном не-английском языке). Еще большие проблемы возникали в ситуации, когда в один документ требова­ лось включать разноязыковые фрагменты.

Второй недостаток связан с ограничением мощности используе­ мого алфавита. Носителям языков европейской группы (как, впро­ чем, и некоторых других языков) сильно повезло - в алфавитах этих языков всего порядка 30 знаков, с учетом регистров - поряд­ ка 60, что хорошо укладывается в предельно допустимое значение 128 символов для дополнительной кодовой таблицы.

Совсем другая ситуация возникает при кодировании иероглифи­ ческих алфавитов. Так, например, разные словари китайских иеро­ глифов содержат *от 50 до 85 тысяч* знаков, в большом японском словаре приведено *50 тысяч* иероглифов, из которых в современ­ ном языке их чуть больше *трех тысяч,* и только(!) *1945* иерогли­ фов утверждены в качестве повседневно необходимых.

Разумеется, можно ограничиться техническим китайским или японским алфавитами и успешно вести на таких усеченных языках деловой документооборот, писать письма друзьям, газетные репор­ тажи и научные статьи, можно даже написать руководство по экс­ плуатации велосипеда или персонального компьютера, но вряд

**52**

ли удастся записать текст поэтического произведения без использо­ вания всего богатства этих великих восточных языков.

Естественным выходом из такой ситуации является отказ от 8-битной системы кодирования, что и было сделано разработчи­ ками стандарта UNICODE.

* + 1. **Стандарт UNICODE**

Стандарт *Unicode* был разработан в 1991 г. с целью создания единой кодировки символов всех современных и многих древних письменных языков. Каждый символ в этом стандарте представля­ ется 16-разрядным двоичным числом, что позволяет закодировать 65 536 символов (что явно должно удовлетворить высокообразован­ ных носителей китайского и японского языков).

В одном текстовом документе, закодированном в стандарте Unicode, могут одновременно присутствовать китайские иерогли­ фы, математические символы, буквы греческого алфавита, лати­ ницы и кириллицы, при этом становится ненужным переключение кодовых страниц.

Другим существенным отличием стандарта Unicode является то, что он не только приписывает каждому символу уникальный код, но и определяет различные характеристики этого символа, например:

* тип символа (прописная или строчная буква, цифра, знак препинания);
* атрибуты символа (отображение слева направо или справа налево, разрыв строки и т. д.);
* соответствующая прописная или строчная буква (для строч­ ных и прописных букв);
* соответствующее числовое значение (для цифровых симво­ лов).

Стандарт состоит из двух основных разделов: универсальный набор символов UCS *(Universal Character Set),* определяющий одно­ значное соответствие символов их двоичным кодам, и семейство кодировок UTF *(Unicode Transformation Format),* определяющее ма­ шинное представление последовательности кодов UCS.

Все кодовое пространство в диапазоне кодов от 0000 до FFFF раз­ бито на несколько *стандартных подмножеств,* каждое из которых соответствует либо алфавиту какого-то языка, либо группе специ­ альных символов, сходных по своим функциям:

* диапазон с кодами от 0000 до 00FF содержит символы основ­ ной и дополнительной кодовых таблиц ASCII;
* далее расположены знаки различных письменностей, знаки пунктуации и технические символы;
* под символы кириллицы выделен диапазон с кодами от 0400 до 04FF;

часть кодов зарезервирована для использования в будущем.

**53**

В табл. 3.4 приведен (исключительно для демонстрации возмож­ ностей стандарта) перечень подмножеств Unicode 3.0 с указанием диапазонов кодов.

*ТаблицаЗ.4*

**Подмножества стандарта UNICODE**

|  |  |  |  |
| --- | --- | --- | --- |
| **Диапазон**  **КОДОВ** | **Описание подмножества** | **Диапазон**  **КОДОВ** | **Описание подмножества** |
| 00-7F | Базовая латиница | 2150-218F | Числовые символы |
| 80-FF | Латиница-1 ДОПОЛНИ-  тельная | 2190-21FF | Стрелки |
| 100-17F | Расширенная латини- ца-А | 2200-22FF | Математические опе- раторы |
| 180-24F | Расширенная латини- ца-В | 2300-23FF | Технические символы |
| 250-2AF | Фонетические знаки | 2400-24ЗF | Значки управляющих кодов |
| 2B0-2FF | Символы изменения пробела | 2440-245F | Оптическое распозна- вание образов |
| 300-ЗбF | Диакритические знаки | 2460-24FF | Буквы и цифры в кру- жочках |
| 370-ЗFF | Греческое и коптское письмо | 2500-257F | Символы рамок |
| 400-4FF | Кириллица | 2580-259F | Символы заполнения |
| 530-58F | Армянское письмо | 25A0-25FF | Геометрические **сим-**  волы |
| 590-SFF | Ивритское письмо | 2600-26FF | Различные значки |
| 600-бFF | Арабское письмо | 2700-27BF | Значки Dingbats |
| 700-74F | Сирийское ПИСЬМО | 2800-28FF | Шрифт Брайля |
| 780-7BF | Тана (мальдивское письмо) | 2E80-2EFF | Допоплнительные ключи к иероглифам |
| 900-97F | Деванагари | 2F00-2FDF | Кандзи (ключи к ие- роглифам) |
| 980-9FF | Бенгали | 2FF0-2FFF | Описания идеограмм |
| A00-A7F | Гурмукхи | 3000-ЗОЗF | Идеографические символы и знаки пре- пинания |
| *ABO-AFF* | Гуджарати | 3040-309F | Хирагана (японское письмо) |
| B00-B7F | Ория | ЗОАО-ЗОFF | Катакана (японское письмо) |

**54**

|  |  |  |  |
| --- | --- | --- | --- |
| **Диапазон**  **КОДОВ** | **Описание подмножества** | **Диапазон**  **КОДОВ** | *'&М9 табл. ЗА*  **подмножества** |
| B80-BFF | Тамили | 3100-312F | Китайское слоговое письмо |
| C00-C7F | Телугу | 3130-318F | Корейское совмести- мое письмо |
| C80-CFF | Каннада | 3190-319F | Камбун |
| D00- D7F | Малаялам | ЗlАО-ЗlВF | Китайское расширен- ное слоговое письмо |
| D80- DFF | Сингальское письмо | 3200-32FF | Дальневосточные буквы и месяцы в кру- жочках |
| E00-E7F | Тайское (сиамское) письмо | 3300-ЗЗFF | Дальневосточные со- вместимые символы |
| E80-EFF | Лаосское письмо | 3400-4OВ5 | Унифицированные иероглифы, расшире- ниеА |
| F00-FFF | тибетское письмо | 4E00-9FFF | Унифицированные иероглифы |
| 1000-  109F | Бирманское письмо | A000-A48F | Слоговое письмо ицзу (и) |
| l0A0- l0FF | Грузинское письмо | A490-A4CF | Ключи письма ицзу (и) |
| 1100-  llFF | Корейское ПИСЬМО | АСОО-D7АЗ | Корейское слоговое письмо |
| 1200-  137F | Эфиопское письмо | D800-DB7F | Первый байт суррогат- ных символов |
| lЗАО- lЗFF | Чероки ПИСЬМО | DB80-DBFF | Первый байт личных суррогатных символов |
| 1400-  167F | Письмо алгонкинских народов | DC00-DFFF | Второй байт суррогат- ных символов |
| 1680-  169F | Огамическое письмо | E000-F8FF | Область личных сим- волов |
| lбАО- lбFF | Руническое письмо | F900-FAFF | Совместимые иеро- глифы |
| 1780-  17FF | Кхмерское ПИСЬМО | FB00-FB4F | Декоративные вариан- ты букв |
| 1800-  18AF | Монгольское письмо | FBS0-FDFF | Арабское декоратив- ное письмо - А |
| lE00- lEFF | Дополнительная лати- ница | FE20-FE2F | Диакритические знаки половинной ширины |

**55**

|  |  |  |  |
| --- | --- | --- | --- |
| **Диапазон**  **КОДОВ** | **Описание подмножества** | **Диапазон**  **КОДОВ** | *ol/.f МlJl. maбл.ЗА*  **подмножества** |
| lF00- lFFF | Греческое расширен- ное ПИСЬМО | FEЗ0-FE4F | Декоративные дальне- восточные формы |
| 2000-  206F | Знаки пунктуации | FES0-FEбF | Малые варианты символов |
| 2070-  209F | Верхние и нижние индексы | FE70-FEFE | Арабское декоратив- ное письмо - В |
| 20АО-  20CF | Денежные СИМВОЛЫ | FEFF | Неразрывный пробел нулевой ширины |
| 20D0-  20FF | Диакритические знаки символов | FF00-FFEF | Символы полной и по- ловинной ширины |
| 2100-  214F | Буквенные символы | FFF0-FFFD | Специальные символы |

Обсуждая стандарты двоичного кодирования текстовых сим­ волов, мы оцениваем качество и возможности этих стандартов по мощности поддерживаемых ими алфавитов, затратам памяти на хранение закодированных текстов и при этом обращаем внима­ ние на удобство интерпретации кодов символов в процессе их ви­ зуализации.

При всем разнообразии таких стандартов алгоритм интерпрета­ ции будет примерно одинаков и весьма прост:

* транслятор, получив для обработки код очередного символа, последовательно просматривает бинарную кодовую таблицу (похо­ жую на табл. 3.3 или 3.4);
* находит в левом столбце одной из ее строк искомый код и «видит» в правом столбце этой строки соответствующий этому коду символ;
* «рисует» этот символ, используя, например, соответствую­ щую пиксельную матрицу его графического образа.

##### Двоичное кодирование десятичных чисел

Основным критерием оценки качества кодов числовых данных является эффективность выполнения вычислительных операций с этими кодами. Не вдаваясь в детали аппаратной реализации ариф­ метических операций, заметим, что процессор компьютера выпол­ няет элементарные арифметические операции совсем не так, как это делает человек, и эффективность машинной реализации опе­ раций во многом зависит от формы представления операндов - числовых данных, хранимых в памяти компьютера в определенных форматах.

**56**

В контексте рассматриваемого вопроса *числовыми данными* бу­ дем называть операнды и результаты вычисления математических выражений - константы базовых (скалярных) числовых типов или переменные, получившие числовые значения таких типов.

Если переменная получила числовое значение (например, в ре­ зультате выполнения операции присваивания), то это значение бу­ дет сохранено в определенной «ячейке» памяти, формат которой соответствует *типу данных* этой переменной.

*Форматом хранения* значения переменной числового типа будем называть двоичный код, соответствующий фактическому значению модуля кодируемого числа, его знаку (для «знаковых» типов дан­ ных) и (или) точности представления (для вещественных чисел), а также диапазону допустимых значений переменной.

* + 1. Кодирование натуральных чисел

Если для переменной определен тип данных «целое без знака» (например, тип unsigned int в языке Си или типы Byte/Word вязы­ ке Pascal), значит, программист уверен в том, что при всем много­ образии значений этой переменной (на то она и *переменная),* лю­ бое из них будет, во-первых, целым числом и, во-вторых, это число не будет отрицательным.

**Определение 1.***Машинным представлением натурального десятич­ ного числа* является его ***прямой двоичнь,й код,*** получаемый в резуль­ тате перевода этого числа в двоичную систему счисления по прави­ лам арифметики.

Для определения формата хранения натурального десятичного числа *N* потребуется последовательно выполнить ряд простых опе­ раций.

1. *Определить диапазон допустимых значений* этой переменной в соответствии с содержательной постановкой решаемой задачи (точнее - определить только ее наибольшее возможное значение Nmax).
2. *Вычислить разрядность* (в битах) прямого двоичного кода, достаточную для хранения максимального значения Nmax: *Lыt* =

= logiNmax, округляя результат вычисления до ближайшего больше­

го целого.

1. *Определить размер ячейки памяти* (в байтах): Lьyte = *Lыt/8,*

округляя результат вычисления до ближайшего большего целого.

1. Выбрать соответствующий *тип данных,* поддерживаемый ис­ пользуемым языком программирования, например, тип unsignedlnt, Byte или Word. Заметим, что использование языков с динамической типизацией (например, Visual Basic, Python или Erlang) избавляет

**57**

программиста от необходимости «типизации» переменных - соот­ ветствующие типы данных автоматически («динамически») опреде­ ляются (не для переменных, а для их числовых значений) в момент выполнения операции присваивания.

Если для хранения натурального числа выделена ячейка памяти размером в m байтов, то максимальное значение этого числа Nmax =

= 2в•m - 1 (табл. 3.5).

*Таблица3.5*

**Примеры кодирования натурапьных десятичных чисеп**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Десятич- ноечисло** | **Однобайтовый формат** | | **Двухбайтовый формат** | |
| **Binary** | **Hexadecimal** | **Binary** | **Hexadecimal** |
| о | 0000 0000 | 00 | 0000 0000 0000 0000 | 0000 |
| 128 | 1000 0000 | 80 | 0000 0000 1000 0000 | 0080 |
| 255 | 11111111 | FF | 0000 0000 1111 1111 | 00FF |
| 65535 | - | - | 1111111111111111 | FFFF |

* + 1. Кодирование целых чисел со знаком

Для целочисленных переменных, которые могут получать при выполнении программы как положительные, так и отрицательные значения, предусмотрены специальные форматы двоичного пред­ ставления, поддерживаемые соответствующими числовыми типами данных, например Integer или Long.

В таких форматах обеспечиваются эффективные способы хране­ ния информации как о модуле, так и о знаке кодируемого десятич­ ного числа, при этом под эффективностью здесь понимается «удоб­ ство» машинной реализации арифметических операций, человеку же такие форматы покажутся крайне неудобными и непривычными. Модели вычислений, удобные человеку, хорошо знакомы всем нам с начальной школы. Очевидно, самой простой является *анало­ говая модель,* в основе которой понятие *числовой оси* с линейной шкалой. В этой модели аналогом числа является отрезок числовой оси: модуль числа представлен длиной отрезка, а знак числа - его направлением: положительные числа ассоциируются с отрезками, направленными вправо, а отрицательные - с отрезками, направ­ ленными влево. На такой модели алгоритм вычисления алгебраиче­

ской суммы двух слагаемых реализуется «геометрически»:

1. отрезок, представляющий первое слагаемое, «откладывается» от нулевой точки оси в направлении, соответствующем его знаку;
2. от конца первого отрезка откладывается отрезок второго сла­ гаемого в соответствии с его знаком;
3. результат вычисления суммы будет представлен отрезком, на­ правленным от нулевой точки числовой оси к концу второго отрез-

**58**

ка: если результирующий отрезок направлен вправо, то сумма по­ ложительна, в противном случае - отрицательна, при этом длина результирующего отрезка представляет модуль полученной суммы. Ограничения такой аналоговой модели очевидны1, гораздо бо­ лее эффективна для «ручного» счета дискретная (цифровая) модель поразрядных вычислений, использующая такие базовые свойства позиционных k-ичных систем счисления, как «позиционные веса» базисных чисел и объединение *k* единиц младшего разряда в одну

единицу следующего старшего разряда.

Цифровая модель положена в основу «ручных» алгоритмов по­ разрядных вычислений, известных как «сложение/вычитание в столбик». Машинная реализация «ручной» операции сложения не создает особых проблем: циклически, начиная с младшего раз­ ряда, складываем соответствующие базисные числа слагаемых, ана­ лизируем разрядную сумму и, если она превышает основание систе­ мы счисления, вычитаем из суммы основание, переносим единицу в следующий старший разряд и используем ее в качестве третьего слагаемого на следующем проходе по циклу.

Операция поразрядного вычитания реализуется аналогично (с использованием «заемных» единиц старших разрядов), но только в том случае, когда уменьшаемое больше вычитаемого, и конечный результат операции оказывается положительным. В противном слу­ чае (когда большее число вычитается из меньшего) алгоритм суще­ ственно усложняется: уменьшаемое и вычитаемое меняются места­ ми, выполняется противоположная заданной операция вычитания, и к полученному (положительному) результату слева добавляется специальный знак «минус», показывающий, что разность операндов отрицательна.

Заметим, что машинная реализация такого алгоритма вычита­

ния привела бы к двукратной потере производительности вычисле­ ний, так как для определения большего из двух п-разрядных операн­ дов потребуется выполнить *п* операций поразрядного сравнения, суммарная стоимость которых соизмерима со стоимостью основной операции поразрядного вычитания.

И остается еще одна проблема, требующая решения при разра­ ботке компьютерного формата кодирования «чисел со знаком» - это проблема знака «минус» перед обозначением модуля отрица­ тельного числа.

1 При всей ограниченности этой модели на ее основе была создана самая мас­ совая и некогда популярная в инженерной среде аналоговая вычислительная маши­ на *-логарифмическая линейка,* в которой использовались не линейные, а логариф­ мические шкалы числовых осей, что позволило реализовать операции умножения/ деления чисел через операции сложения/вычитания их логарифмов (log[a • *Ь]* =

= log[a] + log[b] и log[a/b] = log[a] - log[b]).

**59**

Стандартный формат, используемый для компьютерного пред­ ставления целых чисел со знаком, предусматривает хранение мо­ дулей отрицательных чисел в *дополнительном двоичном коде,* что исключает необходимость выполнения машинной операции вычи­ тания, заменяя ее операцией сложения.

**Определение 2. Дополнительный код** п-разрядного числа, заданно­ го в любой позиционной системе счисления - это такое п-разрядное число, которое дополняет кодируемое число до *(п+*1)-разрядного числа, содержащего единицу в старшем разряде, и нули - во всех остальных разрядах.

Примеры формирования дополнительных кодов п-разрядных чи­ сел, заданных в десятичной и двоичной системах счисления, при­ ведены в табл. 3.6.

*ТаблицаЗ.6*

**Примеры формирования допопнитепьныхкодов чисел**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Десятичная система счисления** | | | **Двоичная система счисления** | | |
| **Число в прямом**  **коде** | **Дополнитель- ный код числа** | | **Число в прямом**  **коде** | **Дополнительный код числа** | |
| **при**  **n=З** | **при**  **n=S** | **при**  ***n=7*** | **при**  ***n=* 15** |
| 1 | 999 | 99999 | 1 | 1111111 | 111111111111111 |
| 100 | 900 | 99900 | 1010 | 1110110 | 111111111110110 |
| 500 | 500 | 99500 | 1010101 | 0101011 | 111111110101011 |
| 999 | 1 | 99001 | 1111111 | 0000001 | 111111110000001 |
| 50 ООО | - | 50000 | 1111111111 | - | 111110000000001 |

Как следует из определения 2 и примеров из табл. 3.6, сум­ ма прямого и дополнительного кодов п-разрядного числа равна (п+l)-разрядному числу, содержащему единицу в старшем разряде и нули - во всех остальных. Из этого следует простой способ фор­ мирования дополнительного кода числа по его прямому коду путем вычитания п-разрядного прямого кода числа из (п+l)-разрядного

«круглого» числа.

Такой способ применим для любой позиционной системы счис­ ления, однако для двоичной системы существует и другой способ, более удобный для устного счета и реализуемый двумя последова­ тельными шагами:

1. п-разрядный *прямой код* преобразуется в *обратный код* (на­ зываемый также *инверсным)* путем замены в прямом коде всех ну­ лей на единицы, а единиц - на нули;

**60**

1. *дополнительный код* формируется из полученного *обратного кода* путем прибавления к нему числа «1» по правилам арифметики.

Приведем правила компьютерного кодирования целых чисел со знаком:

1. для хранения информации о знаке числа используется стар­ ший бит из общего количества битов, выделенных для хранения двоичного кода числа. Этот бит называется «знаковым» и содержит

«1» для отрицательных чисел и «О» - для положительных чисел и числа «О»;

1. остальные (младшие) биты используются для хранения дво­ ичного кода модуля числа. Например, если для хранения целого числа со знаком выделена ячейка памяти размером в **1** байт, то для кодирования кода модуля этого числа можно использовать только 7 младших битов;
2. если кодируемое число положительно, то модуль числа ко­ дируется ***прямым двоичным кодом*** подобно тому, как это делает­ ся при кодировании натуральных чисел. Например, использование ячейки памяти размером в 1 байт не позволит хранить в ней поло­ жительные числа, превышающие значение +12710 (011111112);
3. если кодируемое число отрицательно, то модуль числа коди­

руется ***дополнительным двоичным кодом.***

Рассмотрим несколько примеров, иллюстрирующих систему ко­ дирования целых чисел со знаком.

***Первый пример*** иллюстрирует тот факт, что использование до­ полнительного кода для представления отрицательного числа по­ зволяет заменить операцию вычитания операцией сложения. Рас­ смотрим хорошо знакомую операцию поразрядного вычитания («в столбик») двух трехразрядных десятичных чисел:

782 - 234 = 548.

Здесь уменьшаемое больше вычитаемого, порядок следования операндов в операции оставлен без изменения, и результат опера­ ции положителен.

Рассмотрим операцию вычитания, противоположную предыду­ щей:

234- 782 = -(782 - 234) = -548.

Здесь уменьшаемое меньше вычитаемого, и поэтому вместо исходной операции вычитания «234 - 782» выполнена противо­ положная ей операция «782 - 234» и к полученному промежуточ­ ному результату «548» добавлен специальный знак «минус» перед модулем числа для обозначения того факта, что результат операции

«-548» - отрицателен.

**61**

Сделаем в связи с этим несколько полезных замечаний:

а) число «-548» с определенной долей условности обозначает отрицательный результат выполнения исходной операции вычита­ ния «234 - 782»;

б) в записи отрицательного результата мы использовали специ­ альный символ - знак «минус» перед модулем числа;

в) для получения этого результата нам пришлось предваритель­ но выполнить операцию поразрядного сравнения операндов, сто­ имость которой соизмерима со стоимостью основной операции, и по результатам сравнения принять решение о замене порядка следования операндов;

г) реализация вычислительным устройством такого алгоритма выполнения операции вычитания (перед каждой операцией - по­ разрядное сравнение операндов) привела бы к двукратной потере производительности вычислений.

Попытаемся выполнить исходную операцию вычитания «234 -

- 782» без предварительного сравнения и замены порядка следова­ ния операндов по алгоритму поразрядного вычитания «в столбик», начиная с младших разрядов, занимая при необходимости единицу в старшем разряде и превращая ее в десяток в младшем:

4-2=2

[10]+3-8=5

[10]+1-7=4

Заметим, что все действия выполняются в строгом соответствии с принципом позиционности десятичной системы счисления. Полу­ чен результат \*452, где «452» -числовой код, представляющий мо­ дуль результата операции, а знак «\*» - специальный символ, обо­ значающий тот факт, что результат операции отрицателен, так как при вычитании старших разрядов нам пришлось занять единицу в несуществующем *(п+***1)-м** разряде уменьшаемого.

Сравним результаты выполнения этой операции: числа «-548» и «\*452» - это две разные формы условной записи одного и того же числа. Заметим, что число 452 *дополняет* число 548 до числа 1000, т. е. в сумме эти два числа дают 1000. Другими словами, число 452 - это дополнительный код числа 548.

Продолжим арифметический эксперимент - выполним опера­ цию алгебраического сложения некоторого числа с результатом предыдущей операции:

1. вычитание прямого кода числа:

800 - 548 = 252;

1. сложение с дополнительным кодом числа:

800 + 452 = (1)252;

**62**

1. вычитание прямого кода числа:

300 - 548 = -248;

1. сложение с дополнительным кодом числа:

300 + 452 = 752.

В 1-й и 2-й операциях получен одинаковый результат- трехраз­ рядное положительное число 252. При сложении с дополнительным кодом в старшем (несуществующем) разряде числа получена едини­ ца - это разряд переполнения, который будет отброшен, однако его наличие может использоваться в качестве признака положительно­ го результата и, как следствие, это признак того, что полученный результат представлен в прямом коде.

В результате выполнения 4-й операции разряд переполнения результата отсутствует, что может служить признаком отрицатель­ ного результата, и, как следствие, признаком того, что число 752 представляет полученный результат в дополнительном коде, соот­ ветствующем результату «-248» 3-й операции.

Следующие примеры иллюстрируют реализацию машинных операций вычитания в двоичном коде (для краткости используется однобайтовый формат хранения данных).

***Второй пример:*** вычитание с положительным результатом. Опе­ рация вычитания десятичных чисел 710 - 110 = 610 в ее машинном представлении будет заменена операцией сложения двоичных ко­

дов модулей операндов 710 + (-110), в которой модуль положитель­ ного операнда будет представлен в прямом коде, а модуль отрица­ тельного - в дополнительном (старшие знаковые биты двоичных кодов подчеркнуты).

Первое слагаемое: машинное представление числа «+710»: QOOO 01112 (нулевой знаковый бит и прямой семиразрядный двоич­ ный КОД модуля ЭТОГО числа).

Второе слагаемое:

* прямой ?-разрядный двоичный код модуля числа 110: ООО 00012;
* обратный ?-разрядный двоичный код модуля числа 110:

11111102;

* дополнительный двоичный код модуля числа 110: 11111112;
* машинное представление числа «-110» (единичный знаковый

бит и дополнительный ?-разрядный код модуля): 111111112.

Операция *вычитания* (7 - 1) заменена операцией *сложения* ма­

шинных кодов ее операндов: положительное слагаемое «+7», мо­ дуль которого представлен прямым кодом, и отрицательное слага­ емое «-1», модуль которого представлен дополнительным кодом: 000001112 + 111111112 = Q00001102.

**63**

Как видим, в знаковом бите результата записан О, т. е. ответ по­ ложителен, и, следовательно, ?-разрядное число 00001102представ­ ляет модуль результата в прямом коде, т. е. этот результат - деся­

тичное число «+6».

***Третий пример:*** вычитание с отрицательным результатом (опе­ рация в десятичной системе счисления: 110 - 710 = -610).

Первое слагаемое: машинное представление числа «+110»:

Q00000012.

Второе слагаемое:

прямой ?-разрядный двоичный код числа 710: ООО 01112;

* обратный ?-разрядный двоичный код числа 710: 11110002;
* дополнительный ?-разрядный двоичный код числа 710:

11110012;

* машинное В-битовое предстамение числа «-710»:111110012 (дополнительный код модуля числа 710 с единицей в знаковом бите).

Операция *вычитания* (1 - 7) заменяется операцией *сложения*

машинных кодов чисел +1 и -7:



Как видим, результат выполнения такой операции содержит единицу в старшем (знаковом) бите, следовательно, младшие семь битов представляют дополнительный код модуля некоторого отри­ цательного числа.

Чтобы определить модуль этого числа, надо восстановить пря­ мой код из ?-разрядного дополнительного кода (путем вычитания из него единицы и последующего поразрядного инвертирования, т. е. замены нулей на единицы и единиц на нули):

* обратный код из дополнительного: 111 10102- ООО 00012=

= 11110012;

* прямой код из обратного: ООО 01102.

Полученный результат - код десятичного числа 6, т. е. результа­

том операции является число «-6», что и требовалось доказать.

* + 1. Кодирование вещественных чисел

В отличие от целых, вещественные числа используются для при­ ближенных вычислений - они всегда представляют некоторое количество с определенной степенью точности. Эту точность при­ нято обозначать количеством знаков, записываемых в дробной ча­ сти числа. Например, два числа: 3,14 и 3,14159 описывают одну и ту же константу *п* с разной степенью точности.

Используется и другая *(экспоненциальная)* форма записи веще­ ственных чисел: *M·kP,* где М называют *мантиссой* числа, *р* - *по­ рядком* числа, а *k* - это основание системы счисления, в которой записано число.

64

Например, ту же константу *п* можно записать множеством раз­ личных способов: 0,314·101, 3,14·100, 314·10-2 или 314159·10-5• Та­ ким образом, вещественное число может быть однозначно задано тремя числами: ***мантиссой, порядком* и *основанием системы счисления.***

Если принять тот факт, что компьютер использует исключитель­ но двоичную систему счисления, то для представления веществен­ ного числа достаточно двух целых чисел - мантиссы и порядка это­ го числа. При этом количество разрядов, отводимых для мантиссы, определяет достижимую точность представления результата вычис­ ления, а разрядность порядка определяет масштаб (диапазон допу­ стимых значений) числа.

Ассоциацией IEEE *(Institute of Electrical and Electronics Engineers)* в 1985 г. был разработан стандарт для представления в двоичном коде вещественных чисел (чисел с плавающей точкой). Стандарт получил обозначение **IEEE 754** и широко используется при аппарат­ ной и низкоуровневой программной реализации вычислительных процедур. Стандарт определяет:

* правила машинного представления нормализованных и де­ нормализованных положительных и отрицательных чисел с плава­ ющей точкой;
* правила машинного представления числа «нуль» и специаль­ ных величин «бесконечность» и «не число»;
* правила (режимы) округления действительных чисел.

Стандартом IEEE 754-85 определены два основных формата представления двоичных вещественных чисел, отличающиеся раз­ мером ячейки памяти, используемой для хранения мантиссы и по­ рядка числа, и, соответственно, допустимыми диапазонами значе­ ний чисел и достижимой точностью вычислений:

* одинарная точность *(single-precision,* Ьinary-32) - 32 бита;
* двойная точность *(douЫe-precision,* Ьinary-64) - 64 бита. Основные форматы поддерживаются большинством языков про­

граммирования: например, Visual Basic поддерживает типы данных single и douЫe, в языке С аналогичные типы получили название float (32 бита) и douЫe, а Python поддерживает тип float половин­ ной точности *(half-precision,* Ьinary-16), дополнительно введенный в стандарт IEEE 754 при его пересмотре в 2008 г.

Стандартом IEEE 754-2008 введены и другие базовые форматы представления вещественных двоичных и десятичных чисел (кроме уже упомянутого выше формата Ьinary-16):

двоичные числа четырехкратной точности (Ьinary-128); двоичные числа восьмикратной точности (Ьinary-256); десятичные числа одинарной точности (decimal-32); десятичные числа двойной точности (decimal-64); десятичные числа четырехкратной точности (decimal-128).

**65**

Рассмотрим представление нормализованных двоичных чисел в формате *single-precision* (Ьinary-32), так как остальные форматы яв­ ляются, по сути, его увеличенными (или уменьшенными) копиями. Для пояснения понятия нормализации/денормализации чисел вернемся к рассмотренному ранее примеру с экспоненциальной за­ писью константы *п* и обратим внимание на то, что существует мно­ жество эквивалентных записей этого числа, например: 0,0314 • 102, 0,314 • 101, 3,14 • 10°, 314 • 10-2, 3140 • 10-3. Отличаются эти записи формой представления *мантиссы* и, соответственно, значением *по­*

*рядка* (называемого также *экспонентой)* числа.

**Определение 3.** Число, представленное в позиционной системе счис­ ления, называют **нормализованным,** если его мантисса не меньше единицы и не больше самого большого базисного числа этой систе­ мы счисления.

Для десятичной системы счисления мантисса нормализованного числа должна находиться в диапазоне значений: 9 *М* 1. В нашем примере нормализованной записью константы *п* является число 3,14 • 10°.

Для двоичной системы счисления это неравенство примет вид:

1 *М* 1, из чего следует, *что в старшем разряде мантиссы нор­ мализованного двоичного числа всегда будет записана единица.* На­ пример, мантисса двоичного числа 11,0010012 в нормализованной форме будет представлена числом 1,10010012 (при этом, естествен­ но, порядок числа будет равен +1).

**Определение 4.** Число называют **денормализованным,** если его мантисса находится в диапазоне значений: **1** > **М О,1,** т. е. в стар­ шем разряде мантиссы денормализованного числа - всегда ноль, а в соседнем с ним младшем разряде - всегда не ноль.

В наших примерах: 0,314 •101 - денормализованная запись кон­ станты *п,* а двоичное число 0,110010012 • 22 -денормализованное представление числа 11,0010012 (основание и порядок двоичного числа условно записаны в десятичной системе счисления).

Чтобы представить десятичное число в формате *single-precision,* необходимо привести его к нормализованному двоичному виду и записать определенным образом код знака числа, его порядок и мантиссу в ячейку размером 32 бита.

**Представление знака числа.** Для кодирования знака числа ис­ пользуется старший (31-й) бит: О - для положительных чисел, 1 - для отрицательных.

**Представление порядка числа.** Для хранения порядка числа ис­ пользуются следующие 8 битов (с 23-го по 30-й), причем порядок

**бб**

хранится в специальном *смещенном* формате: к значению порядка числа прибавляется двоичный эквивалент числа 127 (011111112), что исключает необходимость хранения отрицательных порядков (и, соответственно, необходимость использования дополнительных

кодов). Например, если порядок (после перевода числа в двоичную систему и его последующей нормализации) оказался равным +1112 (+7), то храниться он будет, как число 134 (7+127), или, в двоичной

системе счисления, как 111+0111111 = 10000110.

При таком формате хранения порядка и положительные, и от­ рицательные его значения будут представлены положительными числами: например, порядок «-7» получит значение «120» (-7+127)

или «011110002» (-1112+011111112),

Самый большой порядок числа «+128» будет храниться, как чис­ ло 255 (111111112), а самый маленький «-127» - как число О. Что­ бы вычислить фактическое значение порядка числа, надо вычесть число 127 из числа, хранимого в этих восьми битах.

**Представление нормализованной мантиссы числа.** Нормали­ зованная мантисса хранится в 23 младших битах (с О-го по 22-й) в прямом двоичном коде, независимо от знака числа. Учитывая, что старший разряд (целая часть) нормализованной мантиссы двоич­ ного числа всегда содержит единицу, ее можно вообще не хранить (в целях экономии, так как каждый дополнительный бит удваивает диапазон значений двоичного числа), а «подразумевать» и «дописы­ вать» при реализации вычислительных операций. Так и поступили разработчики стандарта: в младших 23 битах хранится только дроб­ ная часть нормализованной мантиссы (как целое положительное число в прямом двоичном коде).

Рассмотрим пример стандартного 32-битового двоичного коди­ рования десятичного числа 155,625.

1. представим это число в двоичной системе счисления: 10011011,1012;
2. число положительно, следовательно, старший бит равен О;
3. запишем полученное двоичное число в нормализован­ ном экспоненциальном формате 1,00110111012 • 27 - мантисса 1,00110111012, порядок 1112;
4. дробная часть мантиссы: 0011011101{0000000000000} -

в фигурных скобках показаны незначащие нули, дополняющие дробную часть мантиссы *справа* до 23 разрядов;

1. порядок числа (в «смещенном» формате): 7 + 127 = 134, или,

в двоичной системе счисления: 1112 + 011111112 = 1000 01102.

В результате получим следующее машинное представление ис­

ходного десятичного числа 155,625 в стандартном формате одинар­ ной точности:

0100 0011 0001 1011 1010 0000 0000 00002,

**67**

или, в шестнадцатеричной системе счисления:

431ВАООО16.

Машинное представление отрицательного числа -155,625 будет отличаться от рассмотренного ранее только значением старшего (знакового) бита:

1100 0011 0001 1011 1010 0000 0000 00002

или, в шестнадцатеричной системе счисления:

С31ВАООО16.

Для восстановления исходного десятичного числа F по его 32-битной двоичной записи в соответствии со стандартом IEEE 754 можно воспользоваться следующей формулой:

*F=* (-l)S. 2СЕ-127) . (1 +М/223),

где S - знаковый бит (О или 1); *Е* - смещенный порядок; *М* -

дробная часть нормализованной мантиссы.

Отметим, что *(Е* - 127) - это фактический порядок нормали­ зованного числа, (1 + М/223) - нормализованная мантисса, где М/223- ее дробная часть, полученная из целочисленной записи М,

а единица - это та самая целая часть мантиссы, которая была «от­ брошена» при записи (конечно же, не из экономии, а для повыше­ ния точности представления результатов вычислений).

Проверим приведенную формулу на нашем примере +155,625:

1. S=0,(-1)0=1;
2. *Е* = 1000 01102= 134;
3. 2(134-127) = 27 = 128;
4. *М* = 00110111010000000000000 = 1810432;
5. М/223 = 0,2158203125;
6. **1** + М/223 = 1,2158203125;
7. **1** • **128** • 1,2158203125 = 155,625.

Операции умножения нормализованных чисел выполняются процессором компьютера по простому алгоритму:

* + восстанавливаются мантиссы сомножителей (добавляются единицы целой части);

после чего мантиссы перемножаются;

* + результат нормализуется и, при необходимости, округляется;
  + восстанавливаются порядки сомножителей (убираются сме- щения);

затем порядки складываются;

результат сложения порядков смещается; знаковые биты перемножаются.

**68**

Алгоритм выполнения операций сложения вещественных чисел:

* + перед выполнением операций сложения чисел они приводят­ ся к единому (оптимальному) порядку;
  + мантиссы восстанавливаются (добавляются единицы целой части);
  + восстановленные мантиссы приводятся к новому значению порядка;

и затем складываются (с учетом их знаков);

* + результат нормализуется и, при необходимости, округляется.

##### Контрольные вопросы

Системы счисления

1. Что называют системой счисления? Определите понятия «цифра» и «базисное число».
2. Почему римскую систему называют аддитивной, а десятичную, дво­ ичную и шестнадцатеричную - аддитивно-мультипликативными?
3. Определите понятия «основание» и «позиционный вес», используемые при описании позиционных систем счисления.
4. Что обозначено переменными *Ь,* в формуле *Nk* = *I(b;* • *ki),* используе­

мой для вычисления значения числа *Nk,* заданного в k-ичной позиционной системе счисления?

1. Что обозначено выражением *ki* в формуле *Nk* = *I(b;* • *ki),* используе­

мой для вычисления значения числа *Nk,* заданного в k-ичной позиционной системе счисления?

1. Сколько единиц в записи двоичного числа, соответствующего деся­ тичному числу, вычисленному по формуле: 21sз + 2sз + 2з?

Стандарты двоичного кодирования текстовых символов

1. Какова максимально допустимая мощность алфавита, символы ко­ торого кодируются по стандарту ASCII?
2. Какова максимально допустимая мощность алфавита, символы ко­ торого кодируются по стандарту UNICODE?
3. Текстовый файл АSСП-формата содержит слово ИНФОРМАТИКА, за­ писанное в одной строке. Сколько памяти займет этот файл в буфере ОЗУ?
4. Текстовый файл АSСП-формата содержит слово ИНФОРМАТИКА, за­ писанное в один столбец (по одной букве в строке). Сколько памяти займет этот файл в буфере ОЗУ?
5. Текстовый файл формата UNICODE содержит слово ИНФОРМАТИКА, записанное в одной строке. Сколько памяти займет этот файл в буфере ОЗУ?
6. Текстовый файл АSСП-формата содержит неупорядоченный список именованных объектов, при этом имя объекта состоит из любой (одной) цифры или буквы основной или дополнительной кодовой таблицы (напри­ мер: 7, d, W, я или Д). В каком порядке эти объекты будут расположены после сортировки списка (в алфавитном порядке имен, по возрастанию)?
7. Список фамилий русскоязычных студентов, представленный в ко­ дировке Windows Ср1251, программно сортируется в алфавитном порядке (по возрастанию). При очередном редактировании списка была допущена опечатка при написании фамилии студента Афанасьева - он был записан

**69**

как афанасьев. Как изменится положение этого студента в отредактирован­ ном списке после его сортировки?

1. Студентка Арбузова сменила фамилию на Watermelon, и этот факт бьm официально зарегистрирован в списке студентов, после чего список

бьm заново отсортирован в алфавитном порядке (по возрастанию). Как изменилось положение фамилии этой (бывшей Арбузовой) студентки в об­ новленном списке?

1. Программа-«шифратор» англоязычного текста АSСП-формата уве­ личивает на 32 код каждой прописной (заглавной) буквы этого текста. Как изменится текст после такого «шифрования»?
2. Программа-«шифратор» англоязычного текста АSСП-формата умень­ шает на 32 код каждой буквы этого текста. Как изменится текст после такого

«шифрования» ?

Двоичное кодирование целых десятичных чисел

1. Дайте определение дополнительного кода п-разрядного числа, за­ данного в k-ичной позиционной системе счисления, и опишите алгоритм преобразования прямого кода числа в дополнительный код.
2. Опишите алгоритм получения обратного и дополнительного кодов двоичного числа, заданного в прямом двоичном коде.
3. Каков диапазон допустимых значений переменной беззнакового целого типа длиной в один байт?
4. Каков диапазон допустимых значений переменной беззнакового целого типа длиной в два байта?
5. Как кодируется знак целого числа в его машинном представлении?
6. В каком двоичном коде (прямом, обратном или дополнительном) будет представлен модуль положительного целого числа?
7. В каком двоичном коде (прямом, обратном или дополнительном) будет представлен модуль отрицательного целого числа?
8. Каков диапазон допустимых значений переменной типа «целое со знаком» длиной в один байт?

Двоичное кодирование вещественных десятичных чисел (стандарт IEEE-754)

1. Что называют мантиссой Мk-ичного вещественного числаZ, значение которого вычисляется по формуле *Z* = М • *Е?*
2. Что называют порядкомР k-ичного вещественного числаZ, значение которого вычисляется по формуле *Z* = М • *Е?*
3. В каких диапазонах значений должна находиться мантисса нормали­ зованного и денормализованного представлений вещественного числа?
4. Какая из трех записей числа *п* (3,14159; 31,4159 • 10-1 или 0,314159

* 10) представляет это число в нормализованной форме?

1. Какая из трех записей числа *п* (3,14159; 31,4159 • 10-1 или 0,314159

* 10) представляет это число в денормализованной форме?

1. Как кодируется знак вещественного числа (по стандарту IEEE-754)?
2. Как кодируется знак порядка вещественного числа и для чего ис­ пользуется хранение порядка числа (по стандарту IEEE-754)?
3. В каком из кодов (прямом, обратном или дополниельном) представ­ ляются мантисса и порядок вещественного числа (по стандарту IEEE-754)?

## Тема4 АППАРАТНОЕ ОБЕСПЕЧЕНИЕ ЭВМ

Компьютерная программа - это языковое описание алгоритма решения задачи, и если алгоритм описан на языке высокого уров­ ня, то программа (так называемый *исходный код)* - не более, чем текст, содержащий описание последовательности выполнения алго­ ритмических операций. Такую программу невозможно выполнить и, следовательно, нельзя ощутить какую-либо ее полезность для по­ требителя (исключая, разумеется, полезность для тренировки ума читателя исходного кода или доход от использования интеллекту­ альной собственности для ее разработчика).

Для того чтобы даже широко востребованная и хорошо напи­ санная компьютерная программа могла стать практически полез­ ной, необходимы как минимум два условия: во-первых, программа должна быть представлена *исполнимым кодом,* т. е. должна быть написана на *языке низкого уровня* - языке машинных команд, вос­ принимаемых аппаратными устройствами, и, во-вторых, необходи­ ма аппаратура - комплекс электронно-вычислительных устройств, способных выполнять машинные команды.

##### Базовые принципы функционирования ЭВМ

Базовые принципы функционирования ЭВМ были сформулиро­ ваны американским ученым и инженером фон Нейманом1 в 1946 г. Эти принципы были выработаны в рамках проекта вычислительной машины EDVAC - одной из первых ЭВМ, использующих программу, хранимую во внутренней памяти, а не на внешнем устройстве (на­ пример, на наборной электрической панели, как это было у предше­ ственника EDVAC - машины ENIAC, перепрограммирование кото­ рой требовало длительного и трудоемкого процесса переустановки перемычек).

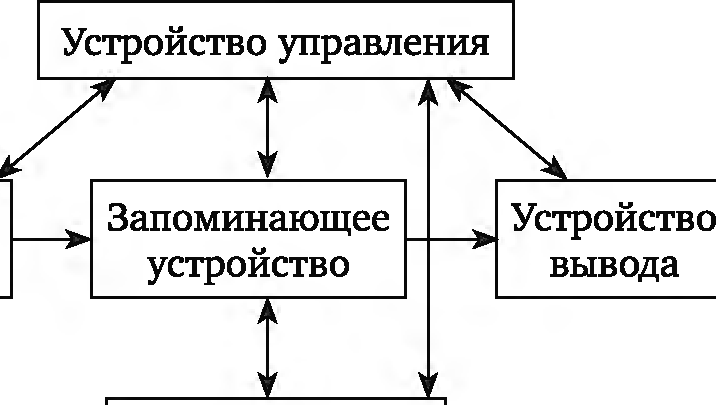
1. ***й принцип*** - *определяет структуру ЭВМ и минимально необ­*

*ходимый состав* ее *функциональных компонентов* (рис. 4.1): ариф-

1 Эти принципы, впоследствии названные *«принципами фон Неймана»,* впервые были опубликованы в статье «Предварительное рассмотрение логического констру­ ирования электронного вычислительного устройства», написанной Джоном фон Нейманом в соавторстве с Артуром Бёрксом и Германом Голдстайном.

**71**

метико-логическое устройство (АЛУ), запоминающее устройство (ЗУ), устройства ввода и вывода (УВВ) и устройство управления (УУ).



Устройство ввода

Арифметико­

логическое устройство

*Рис. 4.1.* **Обобщенная структура фон-неймановской ЭВМ**

**АЛУ** - это основное операционное устройство ЭВМ, производя­ щее арифметическую и логическую обработку данных.

**ЗУ** предназначено для хранения данных и машинных программ. **УВВ** обеспечивают ввод информации от устройств подготовки данных и вывод результатов вычислений на устройства отображе­

ния информации.

**УУ** обрабатывает информацию о состоянии всех прочих устройств и вырабатывает сигналы управления этими устройствами.

Одной из основных функций УУ является вычисление адресов ячеек памяти, из которых читается или в которые записывается информация, обрабатываемая в АЛУ. В современных компьютерах АЛУ и УУ вместе обозначают термином *центральный процессор.*

1. ***й принцип*** - *принцип двоичного кодирования:* любая инфор­ мация, обрабатываемая в ЭВМ, представлена в числовом двоичном коде и разделена на единицы, называемые машинными словами. Машинное слово - это единица данных, обрабатываемая в АЛУ за один цикл. Длина машинного слова (разрядность двоичного кода) - одна из основных характеристик АЛУ. В первых персональ­ ных компьютерах длина машинного слова составляла 8 или 16 би­ **тов.**
2. ***й принцип*** - *принцип адресуемости памяти.* Согласно этому принципу, запоминающее устройство состоит из последователь­

ности пронумерованных ячеек, при этом номер ячейки использу­ ется в качестве ее адреса, указывающего на расположение ячейки в адресном пространстве компьютера. Реализация доступа к памяти по адресам ее ячеек дала возможность использования в программи­ ровании переменных, которые, по существу, выполняют роль име­ нованных ссылок на определенные ячейки памяти.

**72**

1. ***й принцип*** - *принцип однородности памяти:* и программа, и обрабатываемые этой программой данные хранятся в едином запоминающем устройстве в виде последовательности машинных слов.

Алгоритм обработки данных представляется в памяти компьюте­ ра последовательностью *машинных команд,* каждая из которых за­ дает некоторую операцию над данными. Множество операций, вы­ полняемых АЛУ, называется *системой команд1.* Машинная команда содержит *код операции* и *адресные* поля2: код операции указывает на действие, которое следует выполнить в АЛУ над операндами ко­ манды, а адресные поля могут содержать значения этих операндов или адреса ячеек памяти, выделенные для операндов или результа­ тов операции.

Множество взаимосвязанных машинных команд, совместно ре­ ализующих какой-либо алгоритм обработки данных, называется *машинной программой.* Уже давно никто не занимается програм­ мированием на языке машинных команд - машинная программа формируется специальной программой-транслятором3, которая преобразует *исходный код* программы, написанной программистом на языке высокого уровня, в *исполнимый код* программы на языке машинных команд соответствующего процессора.

У принципа однородности памяти имеется два важных послед­ ствия:

1. над машинными командами, при необходимости, можно вы­ полнять те же действия, что и над данными;
2. только программа может определить, какая информация за­ кодирована в определенном машинном слове, и, следовательно, только программа может интерпретировать содержимое этого ма­ шинного слова.

Хранение программы в памяти ЭВМ с возможностью программ­ ной модификации машинных команд создало основу программиро­ вания в его современном понимании.

1. ***й принцип*** - *принцип программного управления* процессом выполнения самой машинной программы.

1 Система команд микропроцессора Intel 8086, на базе которого строились пер­ вые IВМ-совместимые ПК, включала 98 команд длиной от одного до четырех бай­ тов, системы команд современных микропроцессоров существенно богаче.

2 Имеются и *безадресные* команды - в этом случае адреса операндов команды считаются заданными по умолчанию (как правило, это адреса внутренних реги­ стров процессора).

3 Программа-транслятор тоже должна быть написана программистом на каком­ либо языке и затем транслирована в машинный код. Написание трансляторов - узкоспециальная область системного программирования, в которой используются низкоуровневые машинно-ориентированные языки или низкоуровневые возможно­ сти, имеющиеся у некоторых языков программирования высокого уровня.

**73**

АЛУ последовательно выполняет машинную программу, предва­ рительно записанную в память с помощью устройства ввода, по­ следовательно считывая из памяти очередные машинные коман­ ды, начиная с ее первой команды (адрес которой предполагается известным). При этом, в зависимости от результата выполнения текущей команды, управляющее устройство автоматически вы­ числяет адрес машинного слова, содержащего следующую команду программы - это может быть или адрес следующей ячейки, если выполняется линейный участок программы, или адрес какой-либо другой ячейки, на который указывает команда передачи управле­ ния (команда перехода).

Машинная команда, в соответствии с ее «кодом операции», мо­ жет инициировать выполнение АЛУ следующих действий:

* ввод данных из внешнего устройства в ячейки памяти;
* чтение из ячеек памяти данных, используемых в качестве операндов арифметических или логических операций;

выполнение логических или арифметических операций;

* запись результатов операции в ячейку памяти;
* вывод данных из памяти на внешнее устройство.

Программа закончит свою работу в результате выполнения спе­ циальной команды, которая будет интерпретирована, как «конец программы».

###### Типовая архитектура простейшей ЭВМ

Персональные компьютеры построены на базе простейшей *ма­ гистральной* архитектуры (рис. 4.2), называемой также архитек­ турой «с общей шиной». Все модули компьютера (модули памяти и периферийные устройства) подключены к общей магистрали (си­ стемной шине), обеспечивающей их взаимодействие путем обмена данными, адресной и управляющей информацией.

Магистральная архитектура - самая дешевая в реализации, что существенно при массовом производстве ПК, но далеко не самая эф­ фективная по критерию производительности обмена данными, так как временной ресурс общей шины физически делят между собой три логически автономных канала, обеспечивающих прием и пере­ дачу информации между центральным процессором, модулями па­ мяти и периферийными устройствами, а также прием и передачу сигналов, управляющих всеми этими устройствами:

* ***шина адреса*** (ША), по которой центральный процессор пере­ дает модулям памяти *адреса ячеек,* из которых требуется прочитать очередную машинную команду или данные для их последующей обработки командой, или в которые требуется записать данные - результаты выполнения команды;

**74**

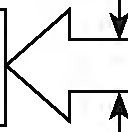
* ***шина данных*** (ШД) используется для двусторонней передачи различных данных (машинных команд, их операндов или результа­ тов выполнения) между центральным процессором и остальными модулями;
* ***шина управления*** (ШУ), по которой центральный процессор передает всем остальным модулям управляющие сигналы и (или) принимает сигналы об их состоянии. На этой шине основной яв­ ляется *линия операции,* на которую устройство-задатчик передает и с которой устройство-исполнитель принимает код операции об­ мена данными *(чтение* или *запись).* Кроме линии операции, шина управления включает *линию занятости* и *линию синхронизации.*

|  |  |
| --- | --- |
|  |  |
|  | ОЗУ  (RАМ) |

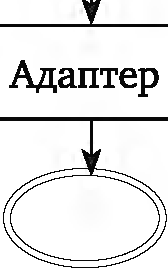
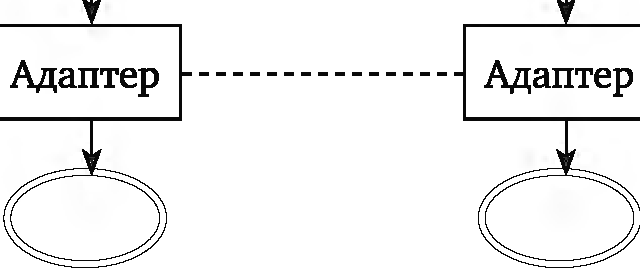
|  |  |
| --- | --- |
|  |  |
|  | ПЗУ  (ROM) |

Видео­ адаптер

Центральный процессор



Общая магистраль (системная шина)

*Рис. 4.2.* **Маrистрапьная архитектура ЭВМ**

Периферийные устройства подключаются к системной шине через специальные программируемые устройства (называемые адаптерами или контроллерами), обеспечивающие интерфейсы взаимодействия с внешним оборудованием. Среди них есть универ­ сальные, например, последовательные (СОМ), параллельные (LPT), и специальные, например, контроллеры дисковых запоминающих устройств, видеоадаптеры, контроллер прерываний, обслуживаю­ щий внешние устройства, взаимодействующих с системой в асин­ хронном временном режиме, или контроллер прямого доступа к па­

**мяти.**

Каждый контроллер имеет собственный набор встроенных реги­ стров, называемых *портами ввода/вывода* и связанных с системной шиной компьютера. Различают *порты данных* (или *буферные пор­ ты),* предназначенные для накопления (буферизации) данных при их приеме/передаче, *порты состояния,* используемые для хранения

**75**

информации о состоянии контроллера или сопряженного с ним пе­ риферийного устройства, и *порты управления,* используемые для получения команд от центрального процессора.

Информация, передаваемая по системной шине, представлена в числовой форме (вспомним 2-й принцип фон-Неймана) - в па­ раллельном двоичном коде, что обеспечивает возможность пере­ дачи за один цикл двоичного числа, разрядность которого не пре­ вышает разрядности соответствующей шины. От характеристик системной шины зависят объем памяти компьютера и максималь­ ное количество подключаемых периферийных устройств.

Разрядность *шины данных* должна соответствовать разрядности центрального процессора, т. е. длине машинного слова, обрабаты­ ваемого в АЛУ за один цикл. Если, например, В-разрядный процес­ сор «работает» в компьютере с 16-разрядной шиной данных, то та­ кая разрядность шины явно избыточна, так как ее старшие 8 битов вряд ли когда-либо будут использоваться. Противоположная си­ туация (например, 32-разрядный процессор и 16-разрядная шина данных) может привести к двукратной потере производительности вычислений, так как «доставка» процессору 4-байтового операнда для выполнения какой-либо операции потребует двукратного обра­ щения к памяти.

Минимальная адресуемая ячейка памяти имеет размер в 1 байт.

Разрядность *адресной шины* определяет максимально возможное ко­ личество однобайтовых ячеек памяти, к которым процессор может обратиться для чтения или записи, передав по этой шине двоичное число, представляющее адрес (порядковый номер) соответствую­ щей ячейки.

В компьютере сп-разрядной адресной шиной можно использо­ вать до *2п* ячеек памяти - с адресами от О до *2п* - 1. Например, В-разрядная адресная шина обеспечит доступ к памяти объемом не больше 256 байтов, 16-разрядная - уже 65 536 (64 Кбайт), а 20-разрядная - 1 Мбайт. Заметим, что один дополнительный бит адресной шины удваивает предельный размер адресного простран­ ства.

Для иллюстрации и исследования алгоритмов взаимодействия программных и аппаратных компонентов компьютера при выпол­ нении соответствующих лабораторных работ будет использовать­ ся виртуальная «DОS-машина» (аналог IВМ-совместимого ПК 1983 года выпуска), работающая под управлением операционной систе­ мы MS DOS и построенная на базе 16-разрядного микропроцессора Intel 8086 (рис. 4.3). АЛУ этого процессора способно обрабатывать за один цикл машинные слова разрядностью не более 16 бит, и все его внутренние регистры, в том числе и адресные, также 16-раз­ рядные. При этом в конструкции процессора предусмотрено спе­ циальное схемотехническое решение - *сумматор адреса,* обеспе-

**76**

чивающий возможность работы с 20-разрядной адресной шиной, что позволило 16-кратно (до 1 Мб) увеличить объем адресного про­ странства.

Отметим еще одну особенность системы адресации, реализо­ ванную в IВМ-совместимых ПК. Адаптеры, через которые перифе­ рийные устройства подключены к системной шине (см. рис. 4.2), имеют собственные «ячейки памяти» - последовательно прону­ мерованные однобайтовые регистры (называемые *портами ввода­ вывода),* через которые центральный процессор производит обмен данными с периферийными устройствами. Номер порта - это его адрес в адресном пространстве ввода-вывода (п. 4.6.2), по которому процессор получает доступ к порту через общую или автономную адресную шину, аналогично тому, как это делается для ячеек моду­ лей памяти.

##### Центральный процессор

На рис. 4.3 представлена упрощенная структурная схема микро­ процессора Intel-8086. Этот микропроцессор - 16-разрядный, т. е. АЛУ процессора способно обрабатывать за один цикл двухбайтовые машинные слова, и все регистры этого микропроцессора - также 16-разрядные. В структуре микропроцессора можно выделить три основных функциональных блока:

* *устройство шинного интерфейса,* обеспечивающее взаимо­ действие с шиной адреса/данных (ШАД) и выборку из памяти оче­ редных машинных команд и их операндов;
* *операционное устройство,* исполняющее эти команды;
* *устройство управления,* получающее информацию о теку- щем состоянии блоков процессора и внешнего оборудования и син­ хронизирующее процессы их взаимодействия.

Блоки могут работать параллельно, совмещая во времени про­ цессы выборки команд из памяти и их исполнения в операционном устройстве.

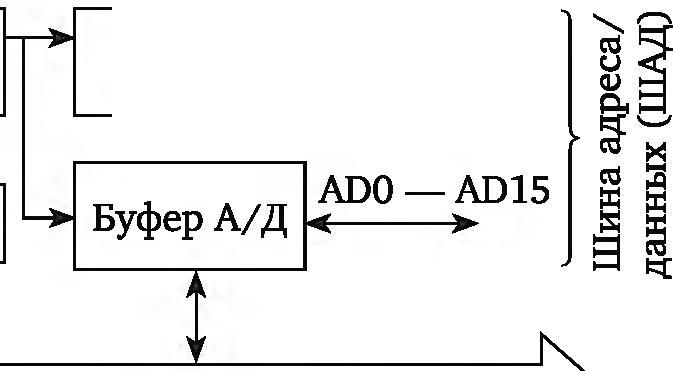
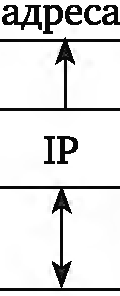
***Шинный интерфейс*** обеспечивает взаимодействие микропро­ цессора с системной шиной компьютера: формирует физические адреса ячеек памяти и портов ввода-вывода, производит выборку данных из этих ячеек (машинных команд и (или) их операндов), формирует очередь команд и записывает результаты их выполне­ ния в соответствующие ячейки памяти или порты ввода-вывода.

В состав шинного интерфейса входят следующие компоненты.

*Очередь команд* - регистровая память объемом 6 байт, предна­ значенная для временного хранения выбранных из памяти команд, ожидающих исполнения в операционном устройстве. Очередь мо­ жет содержать от одной до шести машинных команд (в зависимо-

**77**

сти от их суммарной длины) и обслуживается по конвейерной схеме в порядке поступления команд (FIFO- англ. *First In, First Out).*



1 Сумматор

Буфер А

Аlб-А19

Сегментные регистры

cs ss DS

ES

АЛУ

Р.О.Н. АХ/АН/АL BX/DH/BL CX/CH/CL DX/DH/DL

Регистры указатели SP

ВР SI DI

Регистр флагов

Очередь команд

6 х 8 бит

Устройство управления



M/IO INTR INTA

Шина управления (ШУ)

*Рис. 4.3.* **Схема 16-разрядного микропроцессора lntel 8086**

*Модуль формирования физического адреса,* включающий:

- четыре сегментных регистра, содержащих *номера сегментов памяти,* выделяемых машинным программам при их выполнении:

1. регистр *CS* (Code *Segment)* содержит номер сегмента памяти, используемого для хранения программного кода;
2. регистр *DS (Data Segment)* содержит номер сегмента памяти, используемого для хранения данных, обрабатываемых программой; З) регистр *SS (Stack Segment)* содержит номер сегмента, в кото­ ром организован *стек* - область памяти, обслуживаемая в порядке, обратном прядку поступления в стек данных (LIFO - англ. *Last In, First Out);* стек используется для временного сохранения содержи­ мого внутренних регистров процессора при прерывании выполне­ ния машинной программы (например, при вызове подпрограммы

или программы обработки прерывания);

4) регистр *ES* содержит номер дополнительного сегмента дан­ ных;

указатель команд *IP (Instruction Pointer),* содержащий началь­ ный адрес очередной машинной команды (точнее - только часть

**78**

адреса, а именно, смещение от начала сегмента, номер которого указан в регистре CS). Значение регистра IP автоматически инкре­ ментируется (увеличивается на определенное число) после завер­ шения обработки текущей команды;

* сумматор адреса, выполняющий арифметическую операцию

«суммирования со сдвигом» содержимого двух 16-разрядных реги­ стров (например, регистра-указателя IP и сегментного регистра CS), в результате которой формируется 20-разрядный физический адрес, выставляемый на 20-разрядную адресную системную шину;

* два *буфера* для временного хранения данных или физическо­ го адреса перед их передачей на системную шину адреса/данных (ШАД).

***Операционное устройство,*** обеспечивающее последовательное выполнение машинных команд, предварительно записанных в оче­ редь команд, содержит следующие модули.

*Арифметико-логическое устройство* (АЛУ), предназначенное для выполнения арифметических и логических операций.

*Регистры общего назначения* (РОН: А, В, С и D), иногда называ­ емые «сверх-оперативной памятью» и используемые для временно­ го хранения операндов и промежуточных результатов выполнения команд1. Эти регистры допускают раздельное использование своих младших *(L* - *Low)* и старших (Н - *High)* байтов, обеспечивая тем самым возможность хранения и извлечения как 16-разрядных ма­ шинных слов (Х), так и отдельных байтов данных:

* *АХ (AL* + *АН),* называемый *регистром-аккумулятором,* ис­ пользуется в операциях обмена данными с устройствами ввода/вы­ вода (IN и OUT) и в операциях умножения и деления;
* *ВХ (BL* + *ВН),* называемый *базовым регистром,* используется в вычислениях адреса, указывая на начальный адрес структуры дан­ ных в памяти;
* *СХ (CL* + СН) используется в качестве счетчика *(count)* циклов, определяющего количество повторов некоторой операции;
* *DX (DL* + *DН)* используется для хранения данных *(data),* пере­ даваемых в подпрограммы для обработки.

*Адресные регистры-указатели (SP, ВР, SI* и *DI),* используемые со­ вместно с сегментными регистрами при вычислении физического адреса, аналогично тому, как это делается с указателем команды *IP:*

1 Программист (точнее, компилятор, генерирующий машинный код из транс­ лируемого исходного кода, написанного программистом) сам решает, как исполь­ зовать регистры общего назначения, однако каждый из них имеет еще и свое специ­ фическое назначение, и в ряде случаев машинные команды требуют использования строго определенных регистров: так, например, команды IN и ОUТ обеспечивают прием/передачу данных между портом ввода-вывода, адрес которого указан в ко­ манде, и регистром-аккумулятором.

**79**

* *SP (Stack Pointer):* указывает на вершину стека, позволяет всегда производить выборку данных, записанных в стек послед­ ними;
* *ВР (Base Pointer):* указывает на данные, хранимые в стеке, об­ легчает доступ к данным, передаваемым через стек;
* *SI (Source Index):* индекс источника данных, обычно связан с сегментным регистром *DS;*
* *DI (Destination Index):* индекс назначения, обычно связан с сегментным регистром *ES.*

*Регистр флагов* (F), в котором каждый бит (флаг) содержит при­ знак результата выполненных команды или управляющую инфор­ мацию, используемую при выполнении операций:

* *ZF (Zero Flag):* устанавливается равным «1» при нулевом ре­ зультате операции;
* *SF (Sign Flag):* устанавливается равным старшему биту ре­ зультата выполнения арифметической операции («1» - при отри­ цательном результате, «О» - при равном или большем нуля);
* *CF (Carry Flag):* фиксирует факт переноса единицы из старше­ го бита в результате выполнения арифметической операции;
* *OF (Overfl.ow Flag):* устанавливается в «1» при получении ре­ зультата вне допустимого диапазона чисел;
* *PF (Parity Flag):* устанавливается в «1», если младший байт ре­ зультата операции содержит четное число единиц;
* *IF (Interrupt-enaЫe Flag):* если флаг установлен в «1», аппарат­ ное прерывание разрешено, если в «О» - прерывание игнорируется;
* *DF (Direction Flag):* применяется в командах обработки по­ следовательности байтов в памяти: если флаг установлен в «О», по­ следовательность обрабатывается от младшего адреса к старшему, в противном случае - наоборот;
* *ТF (Trap Flag):* единичное значение флага устанавливает по­ шаговый (отладочный) режим выполнения программы, вырабаты­ вая соответствующее прерывание после завершения выполнения каждой машинной команды.

***Устройство управления*** выполняет дешифрацию команды и вырабатывает соответствующие управляющие сигналы, синхро­ низующие работу всех блоков микропроцессора, а также обеспечи­ вает взаимодействие процессора с периферийными устройствами через системную шину управления.

###### Запоминающие устройства

Согласно одному из принципов фон Неймана, запоминающее устройство компьютера определено как техническое средство хра­ нения машинных программ и обрабатываемых ими данных, что

**80**

и определяет основные требования, предъявляемые к таким устрой­ ствам:

* большая емкость накопителя, позволяющая хранить сложные программы и большие массивы обрабатываемых ими данных и ре­ зультатов обработки;
* малое время доступа к памяти для чтения и записи данных, что должно обеспечить высокопроизводительную работу компью­ тера;
* низкое энергопотребление, а в идеале - полная энергонеза- висимость;
* малые габаритные размеры;
* разумно низкая стоимость запоминающего устройства.

Идеальный вариант - это когда малогабаритный персональный компьютер укомплектован высокопроизводительной памятью ги­ гантской емкости, информация в которой сохраняется неограни­ ченно долго при выключенном питании, и при этом стоимость за­ поминающего устройства не выходит за пределы, ограничивающие возможность массовых продаж компьютерной техники.

Очевидно, что в ближайшей перспективе такой вариант так и останется идеальным и практически не достижимым - объек­ тивные технические, технологические и финансовые ограничения не позволяют выполнить все эти, часто взаимоисключающие, тре­ бования в устройстве какого-то одного типа. Также очевидно и ре­ шение проблемы - вместо одного универсального запоминающего устройства компьютер комплектуется несколькими специализиро­ ванными модулями памяти, каждый из которых предназначен для решения задач определенного класса и удовлетворяет соответству­ ющим требованиям.

Изучение физических принципов работы запоминающих устройств, их возможностей и областей эффективного примене­ ния выходит за рамки вводного курса информатики - эти вопро­ сы детально рассматриваются в соответствующих разделах физики, электроники и схемотехники. Историю и перспективы развития технологий хранения информации также оставим за пределами рассмотрения в настоящем курсе, приведем лишь не очень строгую классификацию (рис. 4.4) запоминающих устройств по их функци­ ональному назначению и специфике использования в программно­ аппаратном комплексе ЭВМ.

Деление запоминающих устройств на «основную» и «внешнюю» память отражает различные способы адресации к этим устрой­ ствам: ячейки основной памяти расположены в основном адресном пространстве (п. 4.6.1), и доступ к ним осуществляется по систем­ ной адресной шине, а внешние запоминающие устройства доступ­ ны по шине ввода-вывода (п. 4.6.2), подобно тому как это делается для прочих периферийных устройств.

**81**

***Основная память*** компьютера представлена *оперативным за­ поминающим устройством* (ОЗУ), доступ к которому возможен как для чтения, так и для записи данных, что и подчеркивается ан­ глоязычным названием таких устройств - *Random Access Memory* (RAM), и *постоянным запоминающим устройством* (ПЗУ), до­ ступным центральному процессору только для чтения *(Read Only Memory* - ROM).

,-

-

Запоминающие устройства

-

-

-

HMDD

CMOS

ПЗУ (ROM)

Основная память

Внешние ЗУ

CD

DVD

1

1 1 1 1

Blu-ray

1 1

--1

R/RW /RE

1

SRAМ

SSD

Оптические диски

ОЗУ(RАМ)

>--

**RAМSSD**

>--

DRAМ

**FmmSSD**

*Рис. 4.4.* **Классификация запоминающих устройств по функциональному назначению**

*Постоянное запоминающее устройство* (ПЗУ) - это энергонеза­ висимая память, не требующая затрат энергии для хранения дан­ ных. Информация, записанная в ПЗУ, сохраняется в нем после вы­ ключения питания компьютера.

ПЗУ программируется («прошивается») при изготовлении ми­ кросхемы и предназначено для хранения неизменяемых в процессе работы компьютера программ и данных: постоянные программные компоненты базовой системы ввода-вывода (ROM BIOS) и обслу­ живающие их и структуры данных; программа POST *(Power Оп Self Testing),* автоматически запускаемая при включении питания и те­ стирующая память и подключенное внешнее оборудование; про­ грамма начальной загрузки операционной системы.

**82**

ПЗУ - относительно медленное устройство, поэтому, если тре­ буется оперативный доступ к данным, записанным в ПЗУ, операци­ онная система копирует эти данные в оперативную память, работа­ ющую существенно быстрее.

*Оперативное запоминающее устройство* (ОЗУ), как следует из его названия, используется в оперативном режиме - для запи­ си и чтения машинных программ и обрабатываемых ими данных. Часть ОЗУ резервируется операционной системой для хранения системного программного обеспечения и данных справочного ха­ рактера (см. табл. 4.2 далее), другая часть отдана в распоряжение прикладному программному обеспечению.

ОЗУ компьютера может быть построено на базе модулей динами­ ческой *(Dynamic* RАМ) или статической *(Static* RАМ) памяти.

В основе DRAМ - схема, состоящая из транзистора и конденса­ тора, которая обеспечивает хранение одного бита информации: для установки бита в единичное состояние требуется зарядить конден­ сатор, а для сброса в ноль - соответственно, разрядить. Преимуще­ ством DRAМ являются компактность и дешевизна технической реа­ лизации, а недостатком - относительно низкое быстродействие, так как, во-первых, требуется время на зарядку/разрядку конденса­ торов, а во-вторых, для поддержания заряженного состояния кон­ денсатора требуется его периодическая «подзарядка». Контроллер модуля DRAМ периодически приостанавливает операции доступа к памяти для регенерации её содержимого.

Статическая память *SRAМ* собрана на триггерах и существенно выигрывает у DRAМ по производительности: во-первых, изменение напряжения на входе триггера приводит к изменению его состояния практически без задержки, и, во вторых, отсутствие в схеме SRAМ конденсатора исключает необходимость периодической регенера­ ции памяти. Недостатками SRAМ являются его высокая стоимость и низкая (по сравнению с DRAМ) информационная плотность.

Обычно ОЗУ персональных компьютеров строят на базе моду­ лей динамической регенерируемой памяти DRAМ, а статическую память SRAМ используют, например, в кэш-памяти микропроцес­ соров.

***Внешние запоминающие устройства*** (ВЗУ) персональных ком­ пьютеров представлены большим разнообразием типов устройств и их конструктивных исполнений. Общее свойство всех типов ВЗУ - энергонезависимость, что, собственно, и позволяет таким устройствам выполнять свое главное предназначение - долговре­ менное и надежное хранение больших объемов информации.

ВЗУ выигрывают у ОЗУ и ПЗУ по такому параметру, как стои­ мость хранения единицы информации, но существенно проигрыва­ ют им по производительности. Последнее не так критично, так как режим доступа к ВЗУ предполагает эпизодическое обращение к ним

**83**

для загрузки в ОЗУ фрагментов файлов или, наоборот, для записи в файлы содержимого оперативной памяти.

Наиболее распространенным типом ВЗУ в большинстве персо­ нальных компьютеров является *накопитель на жестких магнитных дисках* (НЖМД или HMDD - *Hard Magnetic Disk Drive),* основанный на принципе электромагнитной записи. Является RАМ-устройством, т. е. позволяет производить как чтение, так и запись данных.

Накопитель на жестких магнитных дисках - технически слож­ ное электромеханическое устройство, управляемое специализи­ рованным контроллером. Основу конструкции HMDD составляет пакет металлических дисков, закрепленных на общем валу, враща­ ющемся с высокой скоростью. Рабочие поверхности дисков *(sides)* покрыты магнито-жестким ферромагнитным сплавом, важным свойством которого является сохранение состояния намагниченно­ сти участков поверхности после снятия напряженности магнитного поля. Высокий или низкий уровень намагниченности элементов ра­ бочей поверхности диска используется для двоичного кодирования записанной информации.

Источником магнитного поля при записи информации на диск является магнитная головка *(head),* она же является и приемником электромагнитного сигнала от намагниченной области поверхно­ сти диска при чтении информации. Блок магнитных головок (по од­ ной головке на каждую рабочую поверхность диска) совершает воз­ вратно-поступательное перемещение в радиальном направлении под управлением прецизионного дискретного электропривода, обе­ спечивающего точное позиционирование блока на заданном ради­ усе. При этом механический контакт магнитной головки с рабочей поверхностью вращающегося диска отсутствует - между ними под­ держивается постоянный воздушный зазор величиной в несколько нанометров.

Блок магнитных головок имеет несколько фиксированных ради­ альных позиций, каждая головка «описывает» на соответствующей рабочей поверхности диска множество концентрично расположен­ ных кольцевых поверхностей, называемых «дорожками» или «тре­ ками» *(track).* Множество треков, имеющих одинаковый радиус, но расположенных на различных рабочих поверхностях, называют­ ся «цилиндром» *(cylinder).*

Каждый трек разделен на множество секторов *(sectors),* при стандартном форматировании количество секторов на всех треках одинаково, одинакова также и информационная емкость сектора - обычно она составляет 512 байтов (1/2 Кбайт).

Сектор является минимально адресуемой «ячейкой» дискового накопителя (для сравнения - минимально адресуемая ячейка ОЗУ и ПЗУ имеет размер в 1 байт), а информационная емкость накопи­ теля *М* зависит от количества магнитных головок (рабочих поверх-

**84**

ностей) и плотности магнитной записи, определяемой количеством дорожек на рабочей поверхности и количеством секторов на каж­ дой дорожке:

*М* = *Heads* • *Tracks* • *Sectors* • 512. (4.1) Основные характеристики серийно выпускаемых жестких дис­

ков:

* *форм-фактор.* Включает физический размер (диаметр диска, обычно равный 3,5 или 2,5 дюйма, реже - 1,8, 1,3, 1 и 0,85 дюй­ ма), количество головок / рабочих поверхностей диска *(heads/ sides)* и параметры плотности магнитной записи *(cylinders/tracks* и *sectors);*
* *скорость вращения (spindle speed)* - угловая скорость враще­ ния пакета дисков, измеряемая в оборотах в минуту и влияющая на время доступа к данным, хранимым на HMDD. Стандарты ско­ ростей вращения: для ноутбуков - 4200, 5400 и 7200 об/мин; для ПК - 5400, 7200 и 10 ООО об/мин; для серверов и высокопроизво­ дительных рабочих станций - 10 ООО и 15 ООО об/мин;
* *время доступа (random access time).* Определяет время гаран­ тированного выполнения операции чтения или записи; у разных накопителей может составлять от 2,5 до 16 мс;
* *емкость (capacity).* Определяет максимальный объем храни­ мых данных, может достигать нескольких терабайт;
* *интерфейс.* Определяет правила (протокол) обмена данны­ ми. Используются интерфейсы АТА, SATA, SCSI, SAS, FireWire, USB, SDIO, Fibre Channel.

Следующая категория внешних запоминающих устройств - это ***оптические диски*** *(optical disc)* - оптико-механические устройства в форме многослойного диска: основа диска изготовлена из по­ ликарбоната, на нее методом напыления нанесены тонкие свето­ отражающие слои, поверх которых нанесены слои, чувствительные к воздействию луча лазера, и все это покрыто слоем прозрачного защитного материала.

Кодирование информации при ее записи на оптический диск происходит за счет изменения оптических свойств рабочего слоя диска, на котором под воздействием лазерного луча формируются мельчайшие выемки, называемые *питами (pit).* При считывании данных интенсивность отраженного луча изменяется при попада­ нии на очередной пит, что позволяет декодировать записанную ин­ формацию. Множество питов образуют на поверхности диска спи­ ральную дорожку (а не множество концентрических дорожек, как это имеет место в магнитных дисках).

Используются различные технологии и форматы записи ин­ формации на оптические носители. Первыми появились (1979 г.)

**85**

CD-ROM *(compact disk)* - RОМ-устройства, предназначенные толь­ ко для чтения исходно записанной на них информации объемом, не превышающим 750-800 Мб. Позднее были разработаны тех­ нологии однократной записи на оптический диск (CD-R, 1988 г.), а также многократной перезаписи информации на оптические диски (CD-RW) - однако, такие диски остаются, по существу, все теми же RОМ-устройствами, так как количество циклов перезаписи на них ограничено.

На смену CD пришел формат DVD1, обладающий большей плот­ ностью размещения данных как за счет уменьшения расстояния между дорожками спирали (CD -1,б мкм, DVD - 0,74 мкм), так и за счет возможности двухслойной записи. Среди множества типов DVD-дисков наиболее распространенным является DVD-5 (односто­ ронний однослойный диск емкостью 4,7 Гб), а наиболее емким - DVD-18 (двухсторонний двухслойный диск емкостью 17 Гб).

Более совершенный формат многослойной записи ***Blu-ray*** полу­ чил свое название от цвета луча коротковолнового (405 нм) сине­ го лазера, используемого для чтения-записи информации. Диски этого формата (называемые *ВD-дисками)* обеспечивают еще более высокую плотность записи и, соответственно, обладают большей информационной емкостью: однослойный ВD-диск - 25 Гб, двух­ слойный - 50 Гб, четырехслойный - 128 Гб. Имеются и реализа­ ции ВD-дисков еще большей емкости.

Оптические носители информации долгое время являлись самым дешевым средством хранения и распространения программного обеспечения, однако сегодня эти устройства используются все реже, уступив место более современным твердотельным запоминающим устройствам.

**SSD** *(Solid State Drive,* твердотельная статическая память) - запо­ минающее устройство, соизмеримое по емкости с традиционными магнитными дисками и многократно превосходящее их по произво­ дительности за счет отсутствия движущихся механических узлов.

Накопители *RАМ SSD* характеризуются сверхвысокой скоростью чтения и записи информации и используются в качестве ОЗУ для ускорения работы систем упрамения базами данных и мощных гра­ фических станций. RАМ SSD являются энергозависимыми устрой­ ствами, они, как правило, оснащаются аккумуляторами, а более до­ рогие модели - системами оперативного копирования. Основным недостатком RАМ SSD является их чрезвычайно высокая стоимость.

1 Первоначально формат DVD разрабатывался для хранения и комфортного вос­ произведения видеофильмов, откуда и его название - *Digital Video Disk.* Позднее этот формат был оптимизирован и для хранения компьютерной информации, при этом его название DVD сохранилось, но расшифровывается уже как *Digital Versatile Disk* (цифровой универсальный диск).

**86**

Энергонезависимые накопители *Flash SSD* также являются RАМ­ устройствами, позволяющими как читать, так и записывать инфор­ мацию. Эти устройства появились относительно недавно, но се­ годня они прочно закрепились на рынке сменных запоминающих устройств и уверенно завоевывают рынок стационарных ВЗУ, по­ степенно вытесняя с него традиционные магнитные дисковые нако­ пители. Преимуществами Flash SSD по сравнению с HMDD являются небольшие размеры, позволяющие использовать их в портативных устройствах, низкое энергопотребление и высокая производитель­ ность. Основным недостатком Flash SSD является их относительно высокая стоимость, существуют также ограничения по количеству циклов перезаписи (около миллиона циклов в современных устрой­ ствах).

В современных моделях ноутбуков Flash SSD часто является основным (и единственным) стационарным ВЗУ, а многие персо­ нальные компьютеры комплектуются SSD-памятью в качестве до­ полнительного внешнего накопителя, на который, как правило, устанавливаются системное ПО и пользовательские приложения, требующие интенсивного обмена с внешней памятью.

**СМОS-память** - это небольшое по емкости энергозависи­ мое ЗУ, конструктивно выполненное на специальных полупрово­ дниковых элементах СМОS-структуры *(Complementar Metal Oxide Semiconductor).* Особенностями СМОS-структуры являются относи­ тельно низкая производительность, что не является критичным для условий ее использования, и низкое энергопотребление, что гораз­ до важнее, так как фактически обеспечивает энергонезависимый режим ее работы при условии питания от встроенного аккумуля­ тора небольшой емкости. При включенном питании компьютера от сети аккумулятор заряжается, а при выключенном - позволяет достаточно долго сохранять в актуальном состоянии информацию, записанную в СМОS-память.

В СМОS-памяти хранится информация о конфигурации компью­ тера, способах загрузки операционной системы, а также текущие дата и время, которые в процессе загрузки считываются из СМОS­ памяти и записываются в область данных BIOS, организованную в основной оперативной памяти (см. табл. 4.3 далее).

##### Периферийное оборудование

Периферией принято называть все то, что расположено вдали от центра, и в этом смысле периферийными будут считаться все устройства, доступ к которым со стороны центрального процессора осуществляется через дополнительные устройства-адаптеры, под­ ключаемые к системной шине (см. рис. 4.2).

**87**

К периферийным устройствам относятся внешние ЗУ, рассмо­ тренные в подтеме 4.4, а также множество различных устройств, предназначенных для ввода, вывода и передачи информации и обе­ спечивающих коммуникации компьютера с внешним миром.

Известное утверждение о том, что периферийные устройства (в отличие от центральных) можно отключить от компьютера, и он после этого сможет продолжить свою работу, далеко от ре­ альности - если с отсутствием принтера или сканера еще можно как-то смириться, то представить себе работающим современный компьютер, лишенный внешнего ЗУ, клавиатуры, видеомонитора и сетевой карты, весьма проблематично.

Асинхронный режим взаимодействия центрального процессо­ ра с периферийными устройствами обеспечивает контроллер пре­ рываний (п. 7.1.3), обслуживающий множество периферийных устройств и составляющий аппаратную основу *системы обработки прерываний,* рассмотренной в теме 7 данного курса.

Алгоритмы взаимодействия центрального процессора ПК с кла­ виатурой и видеомонитором, а также структуры данных, используе­ мые этими алгоритмами, рассматриваются в подтемах 7.2 и 7.3 дан­ ного курса и исследуются при выполнении ряда лабораторных работ. Структура адресного пространства ввода-вывода и способы адре­

сации периферийного оборудования обсуждаются далее в п. 4.6.3.

##### Адресное пространство ПК

Минимальный размер автономно адресуемой ячейки памяти равен одному байту, при этом, в зависимости от разрядности цен­ трального процессора, бывает необходимо (и весьма полезно для повышения производительности обработки данных) хранить, запи­ сывать и считывать не только однобайтовые ячейки, но и ячейки большего размера - 2, 4 или 8 байтов, называемые *машинными словами.* Машинное слово включает ячейки с соседними адресами. Возможности передачи многобайтовых машинных слов ограни­ чиваются разрядностью системной шины данных, а максимальное количество прямо адресуемых однобайтовых ячеек *N* определяется

разрядностью *п* системной адресной шины: *N* = 2п.

Адресное пространство компьютера - это множество ячеек запоминающих устройств, доступных центральному процессору1 через системную шину адреса. При этом под «ячейками» здесь по-

1 Существует также возможность прямого взаимодействия периферийных устройств, минуя центральный процессор. Прямой доступ к памяти управляется контроллером *DМА (Direct Memory Access)* и используется для высокоскоростной передачи больших блоков данных между памятью и внешними устройствами без использования регистров процессора для промежуточного хранения данных.

**88**

нимаются как *ячейки модулей памяти* (оперативной - *RАМ,* до­ ступной как для чтения, так и для записи, и постоянной - *ROM,* до­ ступной только для чтения), так и *порты ввода-вывода* - регистры различных контроллеров и адаптеров (см. рис. 4.2), связывающих периферийные устройства с системной шиной.

* + 1. Сегментная организация адресного пространства основной памяти

Микропроцессор Intel 8086 поддерживает простейшую модель сегментирования адресного пространства, позволяющую расши­ рить объем физического адресного пространства и сделать машин­ ные программы независимыми от места их физической загрузки в память компьютера.

Процессор обслуживает совмещенные в системной шине адре­ са/данных (ШАД) 16-разрядную шину данных (ШД) и 20-разрядную адресную шину (ША), взаимодействуя с ними через два специаль­ ных буфера (см. рис. 4.3):

* буфер адреса/данных, связывающий внутреннюю магистраль микропроцессора с линиями (ADO - ADlS) ШАД: все 16 битов ШД и совмещенные с ними 16 младших битов ША;
* буфер адреса связан со старшими битами (А16 - А19) шины адреса.

В условиях, когда все регистры микропроцессора, включая адресные регистры, являются 16-разрядными, сегментация адрес­ ного пространства позволила расширить адресную шину на 4 бита

и, соответственно, 16-кратно увеличить объем памяти компьютера (с 216 = 64 Кбайт до 220 = 1 Мбайт).

Адресное пространство основной памяти (рис. 4.6) представле­ но множеством последовательно пронумерованных и перекрываю­ щихся сегментов фиксированного размера 64 Кбайт1. Номера сег­ ментов записываются в соответствующие 16-разрядные *сегментные регистры* (см. рис. 4.3). Например, номер сегмента, выделенного

для хранения кода программы, записывается в регистр *CS,* для хра­ нения данных - в регистр *DS,* а для стека - в регистр *SS.*

16-разрядный адрес ячейки памяти (называемый *эффективным адресом)* задается величиной ее смещения относительно начала не­ которого сегмента. Для хранения эффективных адресов в процессо­ ре также предусмотрены соответствующие 16-разрядные адресные регистры-указатели, например:

* регистр *IP,* используемый для хранения эффективных адресов машинных команд, задает смещение начального адреса команды от­ носительно начала сегмента, номер которого записан в регистр *CS;*

1 В поздних моделях процессоров х86 появилась возможность управления раз­ мерами сегментов в диапазоне от 16 байт до 64 Кбайт.

**89**

* регистр *SI* задает смещение источника данных относительно начала сегмента, номер которого записан в регистр *DS;*
* регистр *SP* указывает на вершину стека, организованного в сегменте памяти, номер которого записан в регистр *SS.*

Формирование 20-разрядного линейного адреса *LA* ячейки па­ мяти иллюстрирует следующая арифметическая модель: на вход *сумматора адреса* подаются два 16-разрядных числа - номер сег­ мента и эффективный адрес; сумматор выполняет над ними опера­ цию *сложения* со *сдвиг.ом* по следующей формуле (приведенной для случая вычисления адреса машинной команды):

*LA* = *CS* • *10000ь* + *IP.* (4.2)

При умножении номера сегмента *CS* на двоичное число lООООь (операция сдвига на 4 разряда) получается 20-разрядный *базовый адрес сегмента* - линейный адрес его начальной ячейки. После сложения базового адреса сегмента с эффективным адресом форми­ руется 20-разрядный линейный адрес, который и передается на си­ стемную шину: старшие 4 бита передаются через буфер на линии А16 -А19 ША, а младшие 16 битов - через другой буфер на линии AD0 - ADlS ШАД. Таким образом, линейный адрес ячейки памяти определяется номером сегмента, задающим его положение в адрес­ ном пространстве, и величиной смещения ячейки от начала этого сегмента.

Учитывая, что размер сегмента равен 64 Кбайт, а смещение со­ седних сегментов составляет 16 байтов, сегменты могут пересекать­ ся в адресном пространстве, т. е. одна и та же ячейка памяти может

«принадлежать» нескольким сегментам. Так, например, положение ячейки с линейным адресом 00040ь (рис. 4.6) может быть задано смещением 0040ь относительно начала О-го сегмента, смещением ООЗОь относительно начала 1-го сегмента или смещением 0020ь от­ носительно начала 2-го.

*Сегментные адреса* принято записывать двумя четырехразряд­ ными шестнадцатеричными числами, разделенными двоеточием: например: в записи адреса [0040:ООlА]ь слева записан номер сег­ мента (40ь), а справа - смещение (1Аь) от начала этого сегмента.

На рис. 4.7 приведен экранный образ фрагмента оперативной памяти объемом 160 байтов в диапазоне адресов от [0040:0000] h до [0040:009F]ь, выведенный на экран одной из инструменталь­ ных программ-анализаторов памяти.

В левой области экрана отображается сегментный адрес неко­ торой ячейки памяти, а в правой области - данные, прочитанные программой из этой и последующих пятнадцати ячеек.

Данные отображаются построчно - каждая экранная строка отображает 16-байтовый блок памяти. Так, в однобайтовой ячейке

**90**

с адресом [0040:0000] h записано двоичное число, эквивалентное шестнадцатеричному числу F8h, а в ячейке [0040:00SA] h - числу овh.

,'":s:

ro

\О,....\_

FFFFFь FFFFEь

...

FООО4ь FОООЗь FООО2ь

FOOOlь

,...,� *Q)*�

FООООь

**о** о.

'" ro

u

�

...

ro:i: .,q

o..,:i::s:

*'" Q)* lООЗОь

u**о** :i::s: 10020ь

§'

о;

*Q) �*

,'":s:

**о** .,q

:i: :i:

10010ь ro

� о.� lООООь

о. м

ro� ro

*Q)* l;l<

OFFFFь

**оо**

,'":s:

,'":s:

\О

ro

ro '"

'" \О '°

;::i::: s;J-

\О

;::i::: s;J-

'°

�N**'-'** ...

м**о**

,:s: ;::i:::

*Q)*:i:

ro ro '°s;J- '" �

\О

ОООSОь ;::i::: '"

*Q)*:i:

*Q)*�

u

'°v

м

:i:

� ,:s:

00040ь

'" u

*Q) Q)*�

�

:i: *Q)*� ,:s:

ОООЗОь

*Q)* u

1. :i

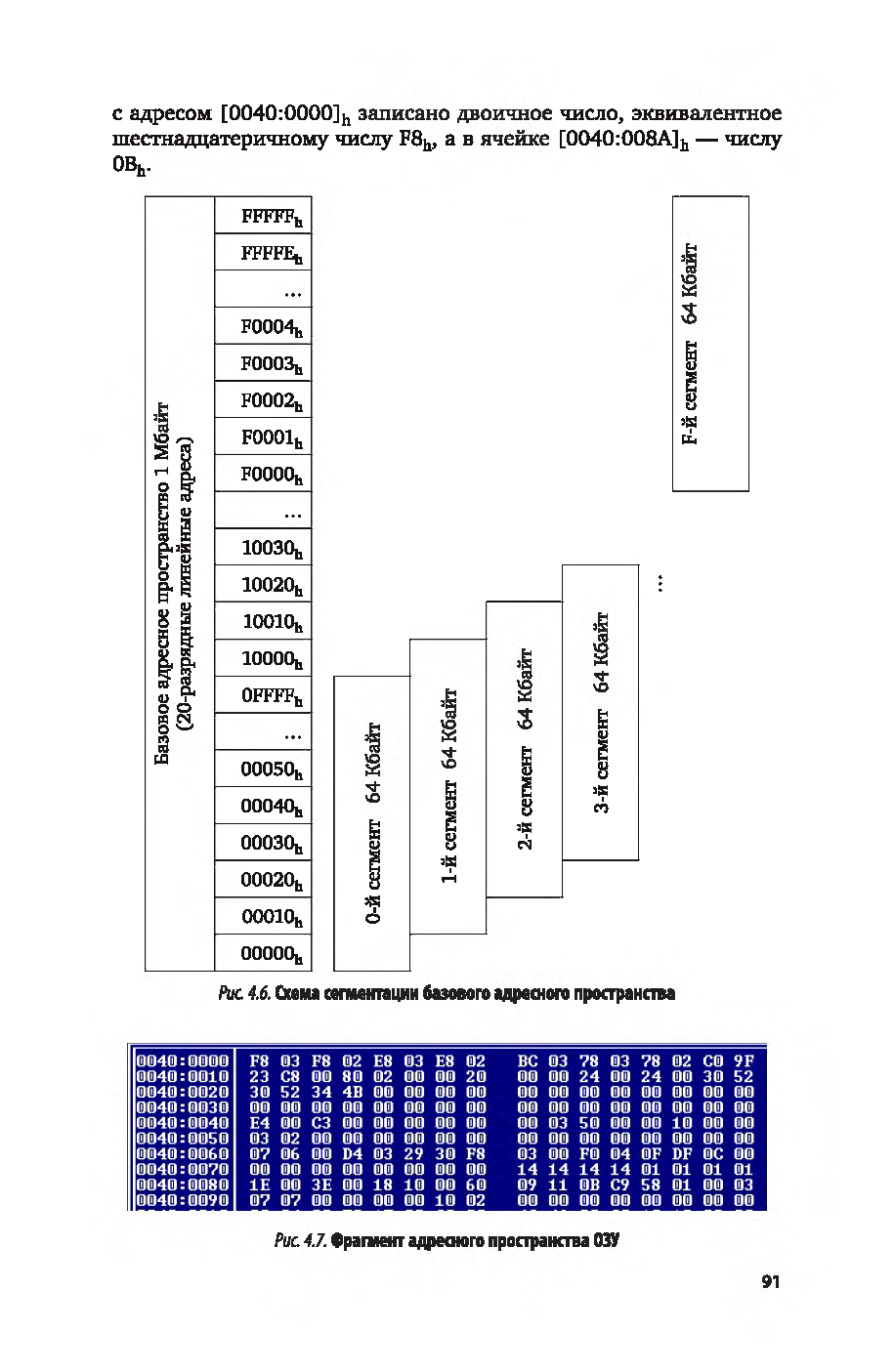
00020ь ОООIОь ОООООь

�

*Q)*�

,u:s:

',...,7

*Рис. 4.6.* **Схема сеrментации базовоrо адресноrо пространства**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| D D • D | (!J(!J(!J(!J | **F8** | m3 **F8** | 1!12 | **Е8** | m3 | **Е8** | 1!12 | вс | m3 | **?8** m3 | **?8** | 1!12 | cm | **9F** |
| D D • D | 1!11!111!1 | 23 | **С8** (!J(!J | 81!1 | 1!12 | (!J(!J | (!J(!J | 21!1 | l!J(!J | (!J(!J | **24** 1!11!1 | **24** | (!J(!J | 31!1 | 52 |
| D D • D | 1!11!121!1 | 31!1 | 52 **34** | **4В** | (!J(!J | (!J(!J | (!J(!J | (!J(!J | (!J(!J | (!J(!J | (!J(!J (!J(!J | (!J(!J | (!J(!J | (!J(!J | (!J(!J |
| D D • D | 1!11!131!1 | (!J(!J | (!J(!J (!J(!J | (!J(!J | (!J(!J | (!J(!J | (!J(!J | (!J(!J | (!J(!J | (!J(!J | (!J(!J (!J(!J | (!J(!J | (!J(!J | (!J(!J | (!J(!J |
| D D • D | 1!11!141!1 | Е4 | l!J(!J С3 | (!J(!J | 1!11!1 | (!J(!J | (!J(!J | (!J(!J | l!J(!J | m3 | 51!1 1!11!1 | (!J(!J | 11!1 | (!J(!J | 1!11!1 |
| D D • D | 1!11!151!1 | m3 | 1!12 (!J(!J | (!J(!J | (!J(!J | (!J(!J | (!J(!J | (!J(!J | (!J(!J | (!J(!J | (!J(!J (!J(!J | (!J(!J | (!J(!J | (!J(!J | (!J(!J |
| D D • D | 1!11!161!1 | 1!1? | 1!16 (!J(!J | **D4** | m3 | **29** | 31!1 | **F8** | m3 | (!J(!J | Fl!I **1!14** | l!IF | **DF** | mc | (!J(!J |
| D D • D | 1!11!1?1!1 | 1!11!1 | l!J(!J (!J(!J | (!J(!J | 1!11!1 | (!J(!J | (!J(!J | (!J(!J | 14 | **14** | **14** 14 | 1!11 | 1!11 | 1!11 | 1!11 |
| D D • D | 1!11!181!1 | 1Е | (!J(!J | (!J(!J | **18** | 11!1 | (!J(!J | 61!1 | 1!19 | 11 | l!IB | **58** | 1!11 | (!J(!J | m3 |
| D D • D | 1!11!191!1 | 1!1? | 1!1? (!J(!J | (!J(!J | (!J(!J | (!J(!J | 11!1 | 1!12 | (!J(!J | (!J(!J | (!J(!J (!J(!J | (!J(!J | (!J(!J | (!J(!J | (!J(!J |

*Рис. 4.7.* **Фраrмент адресноrо пространства ОЗУ**

3Е

**С9**

**91**

https://urait.ru

За один цикл обмена с памятью по шине данных передается 16-разрядное машинное слово, сформированное из содержимого пары «соседних» однобайтовых ячеек памяти, линейные адреса ко­ торых отличаются на 1. При этом в Intеl-совместимых компьюте­ рах младшие 8 битов (D0 - D7) машинного слова хранятся в байте с меньшим (четным) номером, а следующий за ним нечетный байт содержит старшие 8 битов (D8 - Dl5) машинного слова.

На рис. 4.7 байт с линейным адресом [00400]ь содержит число F8, следующий за ним байт с адресом [00401]ь - число 03, байт [00402]ь - число F8, а байт [00403]ь - число 02. При этом пер­ вая пара байтов образует 16-разрядное машинное слово 03F8ь, а не F803ь, как это визуально представлено на экране, а вторая - 02F8ь, а не F802ь, как может показаться человеку, привыкшему чи­ тать числа слева направо от старших разрядов к младшим.

Анализируя рассмотренную систему сегментной организации памяти микропроцессора Intel 8086, можно сделать следующие вы­ воды.

* 1. Процессор может поддерживать не более 216 = 65536 раз­

личных сегментов памяти, так как все его сегментные регистры - 16-разрядные.

* 1. Размер сегмента не может превышать 216 = 65536 байтов

(64 Кбайт), так как регистры, используемые для хранения эффек­ тивных адресов (смещений от начала сегментов), также 16-разряд­ ные.

* 1. Сегменты выровнены по 16-байтовым границам: это, в част­ ности, означает, что если номера сегментов отличаются на единицу, то их базовые адреса отличаются на 16 (или на lОь в шестнадцате­ ричной системе счисления).
  2. Формально существует 4096 альтернативных способов сег­ ментной записи одного и того же адреса, однако это утвержде­ ние будет справедливым только для «срединной части» адресно­ го пространства. Например, два сегментных адреса [0000:0408] h и [0040:0008]ь определяют один и тот же линейный адрес 00408ь, а для линейных адресов [ООООl]ь и [FFFFF]ь существует единствен­ ный такой способ- соответственно, [0000:ОООl]ь и [FFFF:000F]ь.
  3. Используемый способ формирования линейного адреса пу­ тем суммирования базового адреса сегмента со смещением позво­ ляет определить линейные адреса, находящиеся за пределами 1-ме­ габайтной границы. Так, например, сегментный адрес [FFFF:FFFF] соответствует линейному адресу [l0FFEF]ь, что требует увеличения на единицу разрядности адресной шины. В процессоре Intel 8086 игнорируется «лишний» 20-й бит вычисленного линейного адреса, который для рассмотренного выше примера будет иметь значение [0FFEF]ь.

**92**

Завершая обсуждение сегментной организации памяти, следует отметить, что в последующих моделях микропроцессоров Intel х86 отпала необходимость «искусственного» расширения адресного про­ странства за счет ее сегментации: микропроцессор Intel 80386DX, выпущенный в конце 1985 г., уже имел 32-разрядную адресную шину, что обеспечивало возможность прямой адресации к памяти объемом до 4 Гб, а с появлением 64-разрядных процессоров размер адресного пространства увеличился (теоретически) до 16 миллио­ нов терабайт.

При этом разработчики процессоров не только не отказались от сегментной модели памяти (хотя и были неудавшиеся попытки такого отказа), но и сделали ее еще более гибкой-например, рас­ ширился набор сегментных регистров, появилась возможность ра­ боты с сегментами памяти переменной длины.

* + 1. **Стандартное распределение базового адресного пространства**

Процессор с 20-разрядной ША обеспечивает размер адресного пространства в 1 Мбайт с диапазоном линейных адресов однобай­ товых ячеек памяти от [00000]h до [FFFFF]h. Распоряжается этим адресным пространством операционная система - главная «про­ грамма» компьютера, управляющая работой всех его аппаратных компонентов и прикладных пользовательских программ.

В табл. 4.1 приведено стандартное распределение базового адресного пространства ПК (включая ОЗУ, ПЗУ и видеопамять), по­ строенного на базе микропроцессора Intel 8086.

*Таблица4.1*

**Стандартное расnредеnение адресноrо пространства ПК**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **64-килобай- товые блоки памяти** | | **Области памяти** | | |
| **Нач. адрес** | **Имя** | **Размер, Кбайт** | **Имя** | **Использование** |
| [FОООО]ь | F | 256 | ПЗУ (ROM) | Постоянная память: ПЗУ, ROM - *Read Оп/у Memory,* доступная только для чтения.  Используется для хранения неизменя- емых программ и данных: программа самотестирования POST *(Power Оп\_ Self Testing),* автоматически запуска- емая при включении питания; про- грамма начальной загрузки MS DOS; компоненты ROM BIOS; знакогенера- торы |
| [ЕОООО]ь | Е |
| [DОООО]ь | D |
| [СОООО]ь | с |

**93**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **64-килобай- товые блоки памяти** | | **Области памяти** *Окончание табл. 4.*i | | |
| **Нач. адрес** | **Имя** | **Размер, Кбайт** | **Имя** | **Использование** |
| [ВОООО]ь | в | 128 | Video  **RАМ** | Специализированная ОЗУ - буфер обмена данными между программами и видеосистемой.  Программа, выполняемая централь- ным процессором, записывает данные в активную страницу видеопамяти.  Программа, выполняемая видеоадап- тером, сканирует активную страницу видеопамяти, считывает записанные  в нее данные и передает их видеоадап- теру.  Видеоадаптер принимает прочитан- ные данные и вырабатывает сигналы управления дисплеем, формирующие экранный образ активной страницы  видеопамяти |
| [АОООО]ь | А |
| [90000]ь | 9 | 640 | ОЗУ  **(RАМ)** | Оперативная память: ОЗУ, RАМ -  *Random Access Memory* - *память*  с *произвольным доступом,* доступная как для чтения, так **и** для записи.  В ОЗУ загружаются системные и поль- зовательские программы и обра- батываемые ими данные: таблица векторов прерываний, область данных ВIOS, данные и программные компо- ненты MS DOS, драйверы внешних устройств, резидентные программы; пользовательские программы и дан- ные.  Детали распределения адресного про-  странства ОЗУ между системными данными и программами приведены в табл. 4.2 и 4.3 |
| [ВОООО]ь | *в* |
| [70000]ь | 7 |
| [бОООО]ь | 6 |
| [SОООО]ь | 5 |
| [4ОООО]ь | 4 |
| [ЗОООО]ь | 3 |
| [20000]ь | 2 |
| [lОООО]ь | 1 |
| [ООООО]ь | о |

В табл. 4.2 показано распределение оперативной памяти ПК, а в табл. 4.3 представлена информационная структура *области дан­ ных ВIOS* - специальной области ОЗУ размером 256 байтов, в ко­ торой операционная система хранит служебные структуры данных и адресную информацию, используемую в операциях ввода-выво­ да со стандартными устройствами, такими, как клавиатура, видео­ адаптер, последовательный и параллельный порты ввода-вывода.

*Таблица4.2*

**94**

**Стандартное распредеnение оперативной памяти**

|  |  |  |
| --- | --- | --- |
| **Нач. адрес** | **Размер** | **Назначение области ОЗУ** |
| [0000:ОООО]ь | 1 Кбайт | *Таблица векторов прерываний* - справочник начальных адресов программ обработки аппа- ратных и программных прерываний |
| [0040:ОООО]ь | 2566 | *Область данных BIOS* (детали - в табл. 4.3) |
| [0050:ОООО]ь | -128  Кбайт | *Область MS DOS:*   * *программы обработки прерываний,* обслужи- вающие системный ввод/вывод; * *стеки* - используются программами обра- ботки прерываний MS DOS; * *переменные окружения* операционной систе- мы - задаются командами SET, РАТН, PROMPT идр.; * *буферы ввода/вывода* дисковых накопителей; * *дескрипторы открытых файлов.* Каждый дескриптор содержит таблицу доступа к файлу; * *драйверы* дополнительных устройств; * резидентная часть командного процессора (интерпретатора командной строки) *COMМAND. СОМ;* * *резидентные программы* |
| [0050:ООВО]ь | -512  Кбайт | Выполняемая прикладная программа |

*Таблица4.3*

**Структура обnасти данных BIOS**

|  |  |  |
| --- | --- | --- |
| **Начальный адрес** | **Длина, байт** | **Назначение** |
| [0040:ОООО]ь | 2 | Базовый адрес порта первого адаптера последо- вательного канала (COMl) |
| [0040:0002]ь | 2 | То же для СОМ2 |
| [0040:0004]ь | 2 | То же для СОМЗ |
| [0040:ОООб]ь | 2 | То же для СОМ4 |
| [0040:ОООВ]ь | 2 | Базовый адрес порта для 1-ro адаптера парал- лельного канала (LPTl) |
| [0040:ОООА]ь | 2 | То же для LPT2 |
| [0040:ОООС]ь | 2 | То же для LРТЗ |
| [0040:ОООЕ]ь | 2 | То же для LPT4 |
| [0040:ООlО]ь | 2 | Список установленного оборудования |
| [0040:ООlЗ]ь | 2 | Общая память (в килобайтах) |

**95**

|  |  |  |
| --- | --- | --- |
| **Начальный адрес** | **Длина, байт** | **Назначение** *Окончание табл. 4.3* |
| [0040:0017]ь | 2 | Флаги клавиатуры |
| [0040:0019]ь | 1 | Текущее(накопленное)значение ввода кода символа  **(Alt** + цифра) |
| [0040:ООlА]ь | 2 | Адрес «головы» буфера клавиатуры |
| [0040:ООlС]ь | 2 | Адрес «хвоста» буфера клавиатуры |
| [0040:ООlЕ]ь | **32** | Буфер клавиатуры |
| [0040:0049]ь | 1 | Номер текущего видеорежима |
| [0040:004А]ь | 2 | Число символов в строке |
| [0040:004С]ь | 2 | Размер видеостраницы (в байтах) |
| [0040:004Е]ь | 2 | Начальный адрес (смещение в видеосеrменте) активной видеостраницы |
| [0040:ООSО]ь | **8·2** | Координаты курсора (номера строк и столбцов знакомест) для каждой из 8 видеостраниц: млад- ший байт - номер столбца, старший байт - но- мер строки |
| [0040:ООбО]ь | 2 | Размер курсора (номера строк пикселей «вну- три» знакоместа): младший байт - последняя (верхняя) строка; старший байт - начальная (нижняя) строка |
| [0040:0062]ь | 1 | Номер текущей активной видеостраницы |
| [0040:ООбЗ]ь | 4 | Адреса регистров видеоконтроллера |
| [0040:ООб?]ь | 5 | Область данных программы POST |
| [0040:ООбС]ь | 4 | Счетчик тикав таймера (текущее время в 55-мил- лисекундных единицах) |
| [0040:ООВО]ь | 2 | Эффективные адреса (смещения относительно начала сегмента 40ь) начала и конца буфера клавиатуры.  Стандартно - соответственно, [ООlЕ]ь и [ООlС]ь |
| [0040:0082]ь | 2 | ... |
| ... | ... | ... |
| [0040:ООFО]ь | 16 | Область для обмена данными между приложе-  **ниями** |

Назначение, структура и использование *таблицы векторов пре­ рываний области данных ВIOS* рассматриваются в теме 7 данного курса и исследуются при выполнении ряда лабораторных работ.

**96**

* + 1. Адресное пространство ввода-вывода

Это адресное пространство представлено множеством адресов портов ввода-вывода - однобайтовых регистров контроллеров (адаптеров), через которые производится обмен данными с подклю­ ченными к ним периферийными устройствами. Все порты ввода-вы­ вода определенным образом пронумерованы, и номер порта - это и есть его адрес, по которому к порту могут обращаться различные машинные команды.

Существует два основных подхода к организации доступа к пор­ там ввода-вывода: *«Memory mapped 1/0»* и *«l/0 mapped 1/0».*

Первый подход (рис. 4.8) предполагает отображение портов вво­ да-вывода на адресное пространство основной памяти.

|  |  |  |  |
| --- | --- | --- | --- |
| ,:s:  С\1  \О  .,-.;\_,  о  l,Q  :uz:  Ср\1..  оu  р..  Q)  :оz:  u  Q)  р..  Q)  Q)  S"  \*о*О | FFFFFь | Адресное пространство модулей памяти |  |
| FFFFEь |
|  |
| BFFFFь | Адресное пространство ввода-вывода | Port№n |
| ... |  |
|  |
| ВООООь |  |
| AFFFFь |  |
| ... | Port№ 3 |
| Port№ 2 |
| АООООь | Port№ 1 |
| ... | Адресное пространство модулей памяти |  |
| ООООlь |
| ОООООь |

*Рис. 4.8.* **Отображение портов ввода-вывода**

**на адресное пространство модуnей памяти *(Memory mapped VO)***

При этом адресное пространство становится «общим», часть его линейных адресов (на рисунке показаны условно) резервируются для портов ввода-вывода, а оставшиеся адреса используются для об­ ращения к модулям памяти, объем которых при этом, естественно, сокращается.

Преимуществами такого подхода являются простота програм­ мирования и возможность использования для портов ввода-вывода всего многообразия машинных команд, оперирующих с памятью, а основным недостатком (кроме сокращения основного адресного

**97**

пространства) - потери производительности за счет увеличения времени ожидания откликов периферийного оборудования.

Второй подход (рис. 4.9) предполагает автономное использова­ ние двух изолированных друг от друга адресных пространств - 1 Мбайт для модулей памяти и 64 Кбайт для портов ввода-вывода. В такой ситуации два адресных пространства пересекаются, и адре­ са портов ввода-вывода совпадают с адресами всех ячеек нулевого сегмента модулей памяти.

При автономном использовании двух адресных пространств и ячейки модулей памяти, и порты ввода-вывода адресуются по об­ щей системной шине, сопряженной с внутренней шиной микро­ процессора через два буфера (см. рис. 4.3). При этом для адресации ячеек памяти используются все 20 адресных линий [ADO + AD15] +

+ [А16 + А19], а для адресации портов ввода-вывода - только 16 младших линий [ADO + AD15].

16-разрядная адресация портов ограничивает объем адресного пространства ввода-вывода величиной в 64 Кбайт, что позволяет отказаться от сегментирования этого адресного пространства (или условно считать, что все оно расположено в единственном нулевом сегменте). Следует заметить, что 65 536 восьмибитных каналов свя­ зи с внешним оборудованием - это более, чем достаточно для пер­ сонального компьютера.

|  |  |
| --- | --- |
| ,:s:  (!)  о  ::;: ';::;'  о ,:s:  !uЗ \ ctS  :zО:  ctS ..-1  '-'  оu:s:  §<1  :z: t:::  u  (р!)..  < | F000: FFFFь |
| ... |
| 0001: ООООь |
| 0000: FFFFь |
| ...  ...  ... |
| 0000: ОООlь |
| 0000: ООООь |

|  |  |  |
| --- | --- | --- |
| FFFFь | Port№n | tll ,....\_ |
|  |  | )  (!) l'Q \О  о o::i:::  Q l'Q V  (!) \О  ';' |
| 1 i<ас  ОООlь Port № 2 оu :l:'аQ  ООООь Port № 1 §< l'Q | | |

*Рис. 4.9.* **Автономное адресное пространство ввода-вывода (1/0 mapped 1/0)**

Проблема пересечения двух адресных пространств в микропро­ цессорах Intel х86 решена программно-аппаратным способом. В си­ стеме команд этого процессора для чтения/записи ячеек модулей памяти используется команда MOV (в ее многочисленных моди­ фикациях), а для обмена данными с портами ввода-вывода пред­ усмотрены две специальные одноадресные команды **IN и** OUT, по-

**98**

зволяющие прочитать (IN) данные из порта, адрес которого указан в команде, с их последующей записью в регистр-аккумулятор про­ цессора, или записать (OUT) содержимое регистра-аккумулятора в адресуемый порт.

Дешифратор команд, входящий в состав устройства управления, идентифицирует очередную команду и, в зависимости от запи­ санного в ней кода операции, устанавливает выходную линию М/ 1О устройства управления (см. рис. 4.3) в соответствующее состоя­ ние.

Если идентифицирована команда MOV (или другая команда об­ мена данными с памятью), выходная линия М/1O устройства управ­ ления устанавливается в состояние «М», и в результате контроллер системной шины переключает ее в режим доступа к основному адресному пространству для последующего обмена данными с ячей­ ками модулей памяти.

Если идентифицирована команда IN или OUT, выходная линия М/1O устанавливается в состояние «10», и контроллер системной шины переключает ее в режим доступа к адресному пространству ввода-вывода, после чего на эту шину из буфера передается задан­ ный в команде адрес порта.

Микропроцессор Intel 8086 позволяет использовать оба режима доступа к адресному пространству ввода-вывода, и эта возможность была реализована в IВМ РС и его последующих аналогах для взаи­ модействия с видеоадаптерами. В этих ПК управляющие регистры видеоадаптера доступны центральному процессору по автономной шине ввода-вывода *(I/0 mapped 1/0),* а весь объем видеопамяти (128 Кбайт в видеоадаптерах ранних моделей) отображается на ос­ новное адресное пространство *(Memory mapped I/0),* занимая в нем диапазон ячеек с адресами от [АООО:ОООО]ь до [B000:FFFF]ь,

16-разрядная адресная шина ввода-вывода обеспечивает воз­ можность доступа к 65536 однобайтовым портам ввода-вывода в диапазоне адресов от [ОООО]ь до [FFFF]ь, однако в первых IВМ­ совместимых ПК стандартно использовалось только 10 младших би­ тов этой адресной шины, что уменьшало верхнюю границу адресно­ го пространства до [03FF]ь, а максимальное количество адресуемых портов ввода-вывода - до 1024.

Часть адресного пространства ввода-вывода зарезервирована микропроцессором и используется им в своих «внутренних ин­ тересах», остальные порты ввода-вывода могут использоваться для подключения адаптеров различных периферийных устройств. В табл. 4.4 приведены примеры закрепления адресов за портами ввода-вывода, обслуживающими клавиатуру, дисковые запоминаю­ щие устройства, видеоадаптеры, звуковые карты, устройства, под­ ключаемые через стандартные последовательные и параллельные интерфейсы (принтеры, сканеры и другое периферийное оборудо-

**99**

вание), а также за контроллером прерываний и контроллером пря­ мого доступа к памяти.

Один контроллер может использовать несколько портов ввода­ вывода, назначение которых определяется производителем обору­ дования или соответствующим стандартом. Например, контроллер *параллельного* интерфейса, к которому обычно подключались ма­ тричные принтеры, использует три порта ввода-вывода (регистр *вывода данных,* регистр *состояния* и регистр *управления),* а кон­ троллер *последовательного* интерфейса занимает семь портов в адресном пространстве ввода-вывода.

Адрес младшего из портов контроллера называют его *базо­ вым адресом.* В процессе начальной загрузки системы базовые адреса контроллеров записываются в *область данных BIOS,* (см. рис. 4.7 и табл. 4.3) для оперативного использования программами обмена данными с периферийным оборудованием.

*Таблица4.4*

**Примеры исnоnьзования адресов портов ввода-вывода**

|  |  |
| --- | --- |
| **Диапазон адресов** | **Контроллер** |
| 0000-000F | Контроллер DМА |
| 0020-0021 | Контроллер прерываний |
| 0040-0043 | Системный таймер |
| 0060 | Контроллер клавиатуры |
| 0070 007F | СМОS-память |
| 0278-027F | Параллельный порт LРТЗ |
| 02C0-02DF | Видеоадаптер EGA |
| 02F8-02FF | Последовательный порт СОМ2 |
| 0320-032F | Контроллер жесткого диска |
| 0378-037F | Параллельный порт LPT2 |
| ОЗВО-ОЗВВ | Монохромный адаптер (MDA) |
| ОЗВС---ОЗВF | Параллельный порт LPTl |
| ОЗСО-ОЗСF | Видеоадаптер EGA |
| ОЗСО ОЗDF | Видеоадаптер VGA |
| 0ЗD0-0ЗDF | Видеоадаптер CGA |
| ОЗЕВ-ОЗЕF | Последовательный порт СОМЗ |
| ОЗFО ОЗF7 | Контроллер гибкого диска |
| 0ЗF8-0ЗFF | Последовательный порт COMl |

**100**

##### Контрольные вопросы

1. Что декларирует принцип двоичного кодирования фон-неймановской ЭВМ?
2. Что декларирует принцип адресуемости памяти фон-неймановской ЭВМ?
3. Что декларирует принцип программного управления процессом вы­ полнения машинной программы фон-неймановской ЭВМ?
4. Что декларирует принцип однородности памяти фон-неймановской ЭВМ?
5. Как влияет разрядность (в битах) адресной шины компьютера на мак­ симально допустимый объем (в байтах) адресного пространства? Приведите формулу.
6. Что такое «машинное слово»? Как влияет разрядность АЛУ, раз­ рядностъ шины данных и адресной шины на производительностъ работы и на объем памяти компьютера?
7. Каково основное назначение регистра флагов и регистров общего назначения в микропроцессорах х86?
8. Как в 16-разрядном микропроцессоре Intel 8086 решена проблема расширения адресного пространства до 1 Мб? Какие регистры микропро­ цессора Intel 8086, участвуют в формировании линейного адреса ячейки памяти?
9. Каков размер (в байтах) сегмента памяти и чем ограничивается этот размер?
10. Как определить базовый адрес сегмента памяти по номеру этого сегмента?
11. Как определить линейный адрес ячейки памяти по заданным номеру сегмента и величине смещения от его начала?
12. Какую функцию выполняет сумматор адреса в микропроцессоре Intel 8086?
13. В какой из областей памяти (RАМ, VideoRAМ или ROM) размещена таблица векторов прерываний (см. табл. 4.1-4.3)?
14. **В** какой из областей памяти (RАМ, VideoRAМ или ROM) размещена область данных BIOS (см. табл. 4.1 - 4.3)?
15. В какой из областей памяти (RАМ, VideoRAМ или ROM) размещена программа POST (см. табл. 4.1-4.3)?
16. В какой из областей памяти (RАМ, VideoRAМ или ROM) размещена программа начальной загрузки операционной системы (см. табл. 4.1- 4.3)?
17. Какие функции выполняют адаптеры периферийных устройств ПК (см. рис. 4.2)?
18. Что называют портом ввода-вывода и базовым адресом порта ввода­ вывода?
19. В какой области памяти (RАМ, VideoRAМ или ROM) хранится инфор­ мация об адресах базовых портов ввода-вывода (см. табл. 4.2, 4.3)?
20. В какой структуре данных (таблица векторов прерываний, область данных BIOS или VideoRAM) хранится информация об адресах базовых портов ввода-вывода (см. табл. 4.2, 4.3)?
21. Каково назначение линии «M/IO» управляющего устройства микро­ процессора Intel 8086 (см. рис. 4.3)?

## Темаs ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ЭВМ

##### Классификация программного обеспечения

Вычислительная система - это единый программно-аппаратный комruiекс, компоненты которого неразрывно связаны и функцио­ нируют в тесном взаимодействии друг с другом. Программное обе­ спечение (ПО) - необходимый компонент любой компьютерной системы, обеспечивающий ее гибкость, многофункциональность и возможность специализации на решение определенного класса задач в соответствии с потребностями пользователей компьютера.

По назначению и характеру использования различают *систем­ ное* и *прикладное* ПО. *Системное ПО* объединяют понятием *опера­ ционной системы* (ОС), компоненты которой обеспечивают управ­ ление аппаратурой компьютера, хранение данных, управление процессом выполнения программ, взаимодействие с пользователя­ ми. ОС современного компьютера - весьма сложный программный комШiекс, изучению которого посвящены специальные дисциШiины учебных Шiанов подготовки IТ-специалистов.

Все остальное ПО относят к категории *прикладного,* подчерки­ вая этим его вторичность по отношению к системному ПО. Вто­ ричность прикладного ПО (или просто *приложений* - *applications)* по отношению к системному следует понимать в том смысле, что приложения не могут функционировать вне операционной систе­ мы: во-первых, процессы их сохранения, загрузки и исполнения управляются операционной системой и, во-вторых, приложения используют различные компоненты системного ПО (например, для обмена данными с периферийными устройствами компьютера).

Компоненты прикладного ПО специализируют компьютер в определенной области и разрабатываются с учетом потребно­ стей определенных категорий пользователей, для которых соответ­ ствующее ПО является, по существу, инструментом, используемым в их профессиональной деятельности, в бытовой сфере или в сфе­ ре развлечений. Примеры использования прикладного ПО можно взять практически из любой области.

В отдельную категорию выделяют так называемое *инструмен­ тальное ПО,* занимающее промежуточное положение между си-

**102**

стемным и прикладным и ориентированное на особую категорию пользователей - специалистов в области разработки ПО и эксплу­ атации компьютерных систем (программисты, системные админи­ страторы и др.).

К инструментальному ПО относят программы, осуществляющие управление ресурсами вычислительных систем и ориентированные на решение определенного (как правило - весьма широкого) клас­ са задач: САSЕ-средства *(Computer Aided Soft Engineering-* компью­ терные средства разработки ПО, включающие средства поддержки программных проектов, трансляторы с языков высокого уровня, отладчики программного кода), сетевые сканеры и анализаторы памяти компьютера, серверы баз данных и коммуникационные серверы, средства обеспечения информационной безопасности ит.д.

С одной стороны, инструментальное ПО близко к прикладному, так как не работает напрямую с первичными ресурсами, а исполь­ зует для этого сервисы, предоставляемые системным ПО. Но с точки зрения технологий разработки и эксплуатации, инструментальное ПО ближе к системному, так как создается для многоцелевого ис­ пользования, управляет прикладными программами и использует низкоуровневые системные ресурсы в существенно большей степе­ ни, чем прикладное ПО.

Далее в этой теме будет рассмотрена функциональная структура MS DOS - одной из первых операционных систем, использовавших­ ся в IВМ-совместимых ПК, и достаточно простой для первоначаль­ ного ознакомления. Лабораторный практикум также базируется на виртуальной DОS-машине, а выполнение практических заданий экспериментального характера потребует освоения и использова­ ния специализированного инструментального ПО для исследования алгоритмов функционирования различных компонентов MS DOS и структур данных, используемых этими алгоритмами.

###### S.2. Системное ПО

* + 1. **Программная структура MS DOS**

Дистрибутив (комплект поставки) операционной системы MS DOS *(Microsoft Disc Operation System)* включал файлы собствен­ но операционной системы *10.sys* **и** *MSDOS.sys,* конфигурационный файл *CONFIG.sys,* командный процессор *COMМAND.com,* программ­ ные утилиты, реализующие ряд внешних команд ОС, драйверы пе­ риферийных устройств и другие файлы.

*Файл 10.sys* содержит компоненты базовой системы ввода/вы­ вода BIOS - расширение ROM BIOS, хранимой в ПЗУ компьютера и используемой для взаимодействия с аппаратурой компьютера.

**103**

*Файл MSDOS.sys* - это тело операционной системы, содержит набор программ обработки прерываний, в частности, программу обработки DОS-прерывания INT 21ь.

*CONFIG.sys* - файл текстового формата, включающий набор *ко­ манд конфигурации,* которые интерпретируются в процессе загруз­ ки системы и оказывают воздействие на определенные аспекты ее функционирования. Например, команды FCBS и FILES устанав­ ливают максимально допустимое количество одновременно «от­ крытых» файлов, команда DRIVERS определяет перечень подключа­ емых драйверов внешних устройств, а командой BUFFERS можно установить количество 512-байтовых буферов (от двух до 99), ре­ зервируемых в ОЗУ для хранения содержимого секторов системной области дисковых устройств1 с целью сокращения частоты обраще­ ний к внешней памяти при выполнении операций доступа к файлам и каталогам.

*Командный процессор COMМAND.com* предназначен для орга­ низации диалога с пользователем и интерпретации содержимо­ го командной строки2• Он анализирует вводимые пользователем команды и организует их выполнение, вызывая соответствующие функции DOS или внешние программы (исполнимые файлы форма­ тов *.сот,* .ехе или .bat). Поиск программных файлов в файловой си­ стеме ПК, их загрузку в ОЗУ и запуск на выполнение также органи­ зует командный процессор, используя для этого соответствующие компоненты системного ПО, обслуживающие файловую систему.

*Драйверы* - это программы, обслуживающие периферийные устройства. Как правило, программы взаимодействуют с устрой­ ствами не непосредственно, а через драйверы (рис. 5.1), поэтому применение драйверов решает проблемы использования в компью­ терной системе нового периферийного оборудования - достаточно написать для нового устройства драйвер (если его не поставил раз­ работчик устройства) и подключить его к операционной системе.

*Файлы внешних команд* операционной системы содержат про­ граммные утилиты, предназначенные для выполнения различных

«системных» операций (форматирование дисков, дефрагментация дискового пространства, специальное копирование файлов и ряд других).

1 Рекомендуется устанавливать количество буферов, как минимум, достаточное для хранения в ОЗУ всей таблицы FAT (подтема 6.1), так как именно к этой структу­ ре данных наиболее часто обращаются функции файловой системы.

2 В MS DOS отсутствовал графический «оконный» интерфейс, привычный совре­ менному пользователю ПК, и командный интерфейс был единственным способом общения пользователя с ОС. В ОС семейства Windows, пришедших на смену MS DOS, командный пользовательский интерфейс поддерживается Windоws-приложением CMD.exe, которое, по существу, является функциональным аналогом командного процессора COMМAND.com.

**104**

Например, утилита FDISK предназначена для подготовки к ра­ боте жесткого диска. Она разбивает диск на участки, называемые *логическими разделами* или *томами,* при этом некоторые разделы объявляются «загружаемыми»1 *(ЬооtаЫе)* - с одного из этих разде­ лов (в соответствии с установленными приоритетами) будет произ­ водиться загрузка операционной системы. Структура логического раздела диска является объектом исследования при выполнении одной из лабораторных работ, связанных с изучением файловой си­ стемы ПК.

*Файл AUТOEXEC.bat* - файл текстового формата, каждая строка которого содержит DОS-команду или комментарий. Этот файл (при его наличии) будет автоматически выполнен (т. е. будут выполнены все включенные в него команды) в процессе начальной загрузки ОС.

* + 1. **Процесс загрузки MS DOS**

MS DOS - это *дисковая* операционная система, представленная множеством программных и конфигурационных файлов, которые хранятся на системном диске, следовательно, процедура ее загруз­ ки при включении питания компьютера должна предусматривать предварительную инициализацию этого диска для последующего считывания с него системных файлов. Проблема в том, что компо­ ненты ОС, обеспечивающие доступ к файловой системе, также хра­ нятся в системных программных файлах и поэтому не могут быть использованы до завершения ее загрузки. Устранение этой пробле­ мы (точнее - ее обход) базируется на двух принципиальных реше­ **ниях.**

Во-первых, не все системные компоненты хранятся в файлах­ часть функций ввода-вывода (ROM BIOS) реализуется компонента­ ми, записанными в постоянную энергонезависимую память, там же хранятся и две важные программы, запуск которых не требует участия операционной системы.

Во-вторых, процесс загрузки системы реализуется поэтапно - на каждом этапе, исключая последний, выполняются программы­ загрузчики, размещение которых фиксировано и не требует привле­ чения сервисов файловой системы: каждая программа-загрузчик, выполнив «порученную» ей подготовительную работу, загружает следующую программу-загрузчик, а последняя из них загружает си­ стемный файл с программными сервисами файловой системы, с по­ мощью которых затем загружаются и остальные системные файлы.

Процедура загрузки MS DOS при включении питания компьюте­ ра реализуется следующими последовательными шагами.

1 Установка иерархии (приоритетов) активных разделов, с которых будет про­ изводиться загрузка ОС, производится соответствующим программированием СМОS-памяти.

**105**

**Шаг 1.** Управление передается программе *POST (Power Оп Self Testing),* которая хранится в ПЗУ по фиксированному адресу (в раз­ деле ROM BIOS). Основное назначение программы - тестирование аппаратных узлов компьютера и их инициализация, предполага­ ющая сохранение параметров конфигурации устройств в соответ­ ствующих ячейках *области данных BIOS.*

Программа POST при включении питания выполняет следующие действия:

тестирование процессора и постоянной памяти; инициализация таймера и портов ввода-вывода; инициализация контроллера прямого доступа к памяти; проверка системы регенерации ОЗУ;

тестирование ОЗУ;

загрузка в ОЗУ части таблицы векторов прерываний; инициализация видеоконтроллера, после чего все диагности-

ческие сообщения будут выводиться на экран;

инициализация клавиатуры, СМОS-памяти и системных ча-

сов;

инициализация СОМ- и LРТ-портов ввода/вывода; инициализация контроллеров внешних ЗУ.

**Шаг 2.** Если процесс тестирования и инициализации завершился успешно, запускается хранимая в ПЗУ *программа начальной загруз­ ки (Bootstrap Loader1),* которая:

* обращается к СМОS-памяти (которая уже инициализирова­ на на первом шаге начальной загрузки, и, следовательно, известен адрес порта СМОS-памяти) для чтения таблицы приоритетов загру­ зочных устройств;
* по таблице приоритетов определяет загрузочный диск (кон­ троллер которого также уже прошел процедуру инициализации);
* обращается к этому диску, загружает из его нулевого сектора *главную загрузочную запись MBR (Master Boot Record),* в состав кото­ рой входит *таблица разделов диска (Partition ТаЫе)2* и следующая (уже вторая) программа-загрузчик;
* передает управление второму загрузчику.

1 Существует версия происхождения названия *Bootstrap Loader,* которым при­ нято обозначать процесс начальной загрузки операционной системы. Казалось бы, какая связь между ботинками *(boots)* и загрузкой операционной системы? Одно из значений *bootstrap* - *«петля на заднике ботинка, облегчающая его натягивание на ногу»,* другое - *«добиваться чего-либо самостоятельно, без посторонней помо­ щи»,* есть еще и третье - *«вытащить себя за шнурки собственных ботинок».* Следу­ ет согласиться, что у всех перечисленных значений *bootstrap* (особенно у последне­ го) просматривается явная аналогия с процессом поэтапной загрузки ОС.

2 Структура главной загрузочной записи, таблицы разделов и таблицы параме­ тров форматирования разделов диска (используемых на третьем и четвертом шагах процедуры загрузки ОС) детально рассматривается в подтемах 6.1 и 6.2.

**106**

**Шаг 3.** Вторая программа-загрузчик:

* читает *таблицу разделов,* содержащую адреса (диапазоны но­ меров секторов), выделенные разделам диска при его форматирова­ нии;
* определяет адрес начального сектора загрузочного раздела диска;
* загружает из этого сектора таблицу параметров форматиро­ вания раздела и следующую (третью по порядку) программу-загруз­ чик;
* передает управление третьему загрузчику.

**Шаг 4.** Третья программа-загрузчик загружает системные файлы IO.sys и MSDOS.sys (они должны быть записаны на диск и доступны через корневой каталог) и передает управление программе IO.sys.

**Шаг 5.** С системного диска загружается файл CONFIG.sys и опи­ санные в этом файле драйверы.

**Шаг 6.** С системного диска загружается резидентная часть ко­ мандного процессора COMMAND.com (содержащего, в частности, ссылки на системные функции - обработчики внутренних команд MS-DOS), и ему передается управление.

Шаг 7. С системного диска загружается файл AUTOEXEC.bat (если он зарегистрирован в корневом каталоге), и последователь­ но выполняются содержащиеся в нем внутренние и (или) внешние команды.

**Шаг 8.** После загрузки командного процессора и выполнения

«автовыполняемых» команд подготовка системы к работе считает­ ся завершенной, COMМAND.com выводит на экран так называемое

«приглашение DOS» и ожидает от пользователя ввода очередной команды для ее последующей интерпретации и исполнения.

* + 1. **Функциональная структура MS DOS**

Состав функциональных компонентов ОС и схема взаимодей­ ствия системного ПО с аппаратным комплексом и прикладными программами показаны на рис. 5.1.

Ядро ОС включает несколько подсистем, каждая из которых от­ вечает за выполнение определенного класса задач. Эти подсистемы общаются с аппаратурой ПК через BIOS, через драйверы или напря­ мую.

*Прикладное ПО* может обращаться к драйверам через соответ­ ствующую подсистему ОС, работать с BIOS или непосредственно с аппаратурой. Очевидно, что чем выше уровень интерфейса при­ кладной программы и аппаратуры, тем меньше программа будет зависеть от особенностей аппаратуры.

*Файловая система* является одной из важнейших подсистем MS DOS, она используется как в процессе загрузки ОС, так и в про­ цессе ее работы. Сама операционная система хранится в файловом

**107**

формате, все прикладные программы и обрабатываемые ими дан­ ные хранятся в виде файлов, поэтому можно смело утверждать, что файловая система - ключевая подсистема ОС. Файловая система включает множество программ, доступных через функции соответ­ ствующих программных прерываний, а также используемые этими программами структуры данных, организованные на устройствах внешней памяти.

Прикладная программа

BIOS

Подсистемы MS DOS: Файловая система

Система управления памятью Система управления программами

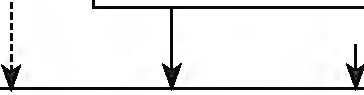


----, **Система связи с драйверами**

--- -

Драйверы

Система обработки ошибок Служба времени

Система ввода-вывода консоли

----t оператора

Аппаратура

*Рис. 5.1.* **Схема взаимодействия ПО с аппаратурой ПК**

Файловая система работает с дисками через драйверы, а драйве­ ры, в свою очередь, пользуются сервисами BIOS. На уровне BIOS вы­ полняются элементарные дисковые операции, такие, например, как чтение/запись секторов, форматирование диска и др. Такой низ­ коуровневый доступ к диску возможен и непосредственно из при­ кладной программы, но обычно она пользуется соответствующими функциями прерывания DOS. Программы защиты от несанкциони­ рованного доступа или копирования файлов вынуждены обращать­ ся к средствам более низкого уровня, вызывая прерывания BIOS, или даже работать с контроллером диска через порты ввода/вывода. Рассмотрению двух популярных файловых систем посвящена следующая тема курса, в одной из лабораторных работ исследуются алгоритмы выполнения файловых операций, реализованных в FАТ­

ориентированных файловых системах ПК.

*Система управления памятью* используется для выделения па­ мяти, необходимой выполняемым программам. Вся доступная про­ граммам память разбивается на блоки, каждому блоку предшеству­ ет управляющая структура МСВ *(Memory Control Вlock),* в которой записаны характеристики блока. Все блоки располагаются друг за другом, адрес первого блока хранится в специальной структу-

**108**

ре данных - организуемой в ОЗУ векторной таблице связи CVT

*(Communications Vector ТаЫе).*

Для каждой вновь запускаемой прикладной программы MS DOS создает определенное количество МСВ-блоков. Программа может

«заказать» для себя дополнительные блоки памяти, используя со­ ответствующие функции DОS-прерывания 21h, обеспечивающие доступ к системе управления памятью. При освобождении памяти или при поступлении от программ запросов на получение дополни­ тельной памяти содержимое таблицы CVT соответственно коррек­ тируется.

*Система управления программами* при запуске прикладной про­ граммы последовательно выполняет типовой набор операций:

* обращается к *системе управления памятью* для подготовки блоков памяти для запускаемой программы;
* обращается к *файловой системе* и загружает в ОЗУ файл (или фрагмент файла), содержащий программу;
* передает управление программе.

Другая задача, решаемая системой управления программами - это запуск программ из других программ и организация программ­ ных перекрытий *(overlay),* используемых в условиях недостаточно­ го объема оперативной памяти. Если не все компоненты большого программного комплекса нужны одновременно, можно разбить комплекс на несколько оверлейных модулей, загружаемых последо­ вательно в один и тот же блок памяти.

Еще одна функция системы управления программами связана с организацией работы *резидентных программ.* Если после завер­ шения работы программы необходимо оставить ее в памяти для по­ следующих оперативных запусков (т. е. сделать эту программу ре­ зидентной), следует обратиться к системе управления программами через соответствующую функцию DОS-прерывания 21h. Классиче­ скими примерами резидентных программ являются интерпрета­ тор команд COMМAND.com и анализатор памяти *Peek-Pook-Resident* (файл Peek.com), который будет использоваться в качестве основно­ го инструмента при выполнении ряда лабораторных работ.

*Система связи* с *драйверами* периферийных устройств обеспечи­ вает интерфейс между программным обеспечением и аппаратурой компьютера и существенно продлевает срок жизни прикладного и системного ПО.

На рынке регулярно появляется новое периферийное оборудова­ ние и новые модификации уже существующих устройств, и это объ­ ективная реальность, бороться с которой бессмысленно и беспер­ спективно. Но что делать с уже разработанным ПО, рассчитанным на старую аппаратуру? Если ПО еще не устарело, вряд ли целесо­ образно переделывать его только из-за того, что появилось устрой­ ство, поддерживающее новый протокол обмена. Интуитивно понят-

109

но, что в этих условиях должна существовать какая-то программная прослойка между аппаратным и программным обеспечением, вы­ полняющая согласующие функции и избавляющая прикладные (а в ряде случаев - и системные) программы от необходимости знать технические подробности устройства периферийного обору­ дования.

Проблемы устранения зависимости программного обеспечения от аппаратуры решаются по-разному в различных операционных системах. MS-DOS, как и многие другие ОС, использует для этих це­ лей механизм драйверов, однако драйверы MS-DOS не всегда обра­ щаются напрямую к аппаратуре, а вызывают функции BIOS, и уже BIOS выполняет операции ввода/вывода.

При этом BIOS обслуживает только стандартные устройства, не­ стандартные же устройства обслуживаются драйверами напрямую. Использование BIOS в качестве интерфейса между драйверами стандартных устройств и аппаратурой существенно повышает живу­ честь MS-DOS на ПК, не вполне совместимых с классическими IВМ РС. Для управления состоянием устройства ввода/вывода и обмена информацией между прикладной программой и драйвером устрой­ ства используется специальная функция 44ъ прерывания DOS 21h.

*Система обработки ошибок* MS DOS использует флаг перено­ са (CARRY FLAG, CF) регистра флагов центрального процессора. Если после обращения к прерыванию CF установлен в 1, произо­ шла ошибка. Интерпретацию ошибки выполняет соответствующая функция прерывания DOS. Например, если произошла критическая ошибка ввода/вывода, вызывается стандартная процедура DOS, вы­ водящая на экран запрос о дальнейших действиях.

*Служба времени.* Компьютер оборудуется системными часами с питанием от аккумулятора, содержимое которых не сбрасывает­ ся при выключении питания, и системным таймером, регулярно (каждые 55 мс) вырабатывающим соответствующее прерывание. MS DOS содержит драйвер устройства CLOCK$, который постоянно ведет подсчет времени и хранит текущие время и дату в соответ­ ствующем буфере, доступном прикладным программам через одну из функций прерывания 21h. Прикладная программа может также перехватывать прерывания системного таймера (INT 8h) и исполь­ зовать эти прерывания для регулярного выполнения каких-либо функций.

*Ввод/вывод на консоль оператора.* Консоль оператора состоит из двух устройств - клавиатуры и дисплея, которые обслуживаются одним драйвером консоли CON. Операционная система обслужива­ ет консоль с помощью функций прерывания 21h, обеспечивающих ввод и вывод символов на устройство CON. Для работы с клавиату­ рой и дисплейным адаптером на низком уровне этот драйвер ис­ пользует соответствующие прерывания ВIOS.

**110**

В качестве практического примера использования консоли опе­ ратора можно рассмотреть применение команды копирования фай­ лов СОРУ: команда *СОРУ CON <путь к файлу>* запишет в указанный файл текстовые данные, введенные пользователем с клавиатуры, а команда *СОРУ <путь к файлу> CON* прочитает содержимое ука­ занного в команде текстового файла и выведет его на экран видео­ монитора.

##### Контрольные вопросы

1. Какие из компонентов программного обеспечения ПК относятся к категории «системное ПО»?
2. Какие из компонентов программного обеспечения ПК относятся к категории «прикладное ПО»?
3. Какие из компонентов программного обеспечения ПК относятся к категории «инструментальное ПО»?
4. К какой из перечисленных выше категорий относятся и для чего могут быть использованы следующие программные продукты:

- Adode Reader;

IDLE;

HeliconFilter;

MS Word;

MS SQL Server;

OpenOffice Calc;

StarUML;

Zoom?

1. К какой категории ПО относятся САSЕ-средства? Какие из перечислен­ ных в вопросе 4 программных продуктов следует отнести к САSЕ-средствам?
2. Какие программы называют драйверами? К какой категории ПО сле­ дует отнести такие программы?
3. Какие функции выполняют файловые системы? Какие файловые системы используются на внешних ЗУ вашего компьютера?
4. Какое место занимает в файловой системе MS DOS программа COMМAND.com
5. Какие из функций 21-го прерывания DOS имеют отношение к фай­ ловой системе (для поиска ответа используйте электронный справочник HELP)?
6. На каких запоминающих устройствах хранятся компоненты систем­ ного ПО, участвующие в процессе загрузки MS DOS?
7. Какие функции выполняет программа POST?

## Темаб

**ФАЙЛОВЫЕ СИСТЕМЫ**

Файловая система является важнейшим компонентом опера­ ционной системы компьютера, ее основное назначение - обе­ спечение надежного хранения файлов на внешних запоминающих устройствах и предоставление прикладным программам унифици­ рованных и эффективных средств доступа к файлам, наличие кото­ рых избавляет прикладную программу от необходимости знать тех­ нические подробности устройства дисковой системы компьютера.

Файловая система включает программные компоненты, обеспе­ чивающие реализацию типовых операций доступа к файлам, и спе­ циальные структуры данных (организуемые как в оперативной, так и в дисковой памяти), обслуживающие работу соответствующих программных компонентов.

###### Трехуровневая модель дискового пространства

Файловые системы поддерживают трехуровневую модель памя­ ти, обеспечивающую интерфейс прикладных программ с аппарату­ рой ВЗУ.

*Верхний (пользовательский)* уровень доступен прикладным про­ граммам, которые отправляют файловой системе запросы на вы­ полнение операций доступа к файлам и получают от нее результаты выполнения таких запросов.

Дисковое пространство ПК на пользовательском уровне представ­ ляется (и визуально поддерживается) множеством иерархических (древовидных) структур: в каждом из «деревьев» роль «корня» выпол­ няет *том* (логический диск), ассоциируемый с корневым каталогом тома, роли «ветвей» - *подчиненные* (дочерние) *каталог.и* различных уровней подчиненности, а *файлам* в этой «древесной» терминологии отведена роль «листьев», которые всегда являются дочерними объек­ тами, т. е. привязаны к какому-либо (единственному) родительскому каталогу, и при этом сами не могут иметь дочерних объектов.

Программы «общаются» с файловой системой на языке пользова­ тельского представления и оперируют привычными пользователю терминами для обозначения объектов: диск (или том), папка (или каталог), файл определенного типа (приложение, документ), каж-

**112**

дый из которых должен быть поименован в соответствии с установ­ ленными соглашениями об именовании объектов. Базовым поняти­ ем такого представления является *«путь к файлу»* (или к каталогу), который представляется текстовой строкой стандартного формата, практически одинакового в различных файловых системах. В нача­ ле строки указывается имя тома, а далее - имена каталогов в по­ рядке их подчиненности.

Если прикладной программе требуется получить доступ к файлу с целью его последующей обработки, программист должен предус­ мотреть в исходном коде этой программы оператор открытия фай­ ла, например такой:

ореп file C:\my\_files\documents\doc123.txt

При этом прикладной программист не должен заботиться о реа­ лизации операции доступа к файлу и не обязан знать1, что указан­ ный в команде путь к файлу будет передан для обработки соответ­ ствующим программным компонентам файловой системы, которые преобразуют его в аппаратный адрес файла и передадут этот адрес для исполнения контроллеру диска, обеспечивающему чтение фраг­ мента файла с диска и запись его в буфер ОЗУ для последующей обработки программой.

Пользовательский интерфейс операционной системы предостав­ ляет возможность выполнения типового набора операций над объ­ ектами файловой системы, например:

* *диск* можно «открыть», просмотреть его свойства (например, тип файловой системы или объем свободного пространства) или от­ форматировать, в результате чего все ранее созданные на нем папки и файлы куда-то безвозвратно (как думает пользователь) исчезнут;
* *папку* можно переименовать, открыть и просмотреть ее со­ держимое, удалить или создать новую папку, подчиненную создан­ ной ранее;
* *файл* можно переименовать, удалить, скопировать или пере­ местить в другую папку, выполнить (если этот файл - приложе­ ние) или обработать соответствующим приложением, например: поместить в архив (сжать), отредактировать (если этот файл-до­ кумент), просмотреть графический файл, выполнить вычисления в электронной таблице или запрос в базе данных.

На *нижнем (аппаратном) уровне* модели программные компо­ ненты файловой системы «общаются» (обмениваются данными) с контроллерами запоминающих устройств. Язык такого обще­ ния - это, по существу, описание устройства дискового накопителя с позиций управляющего им контроллера.

1 Конечно же, профессиональному прикладному программисту непозволитель­ но не знать основ организации файловых систем.

**113**

Контроллер диска имеет дело с физической структурой дисково­ го пространства, которая описывается такими терминами, как ***диск*** (или ***пакет дисков),*** понимаемый как физическое устройство с ав­ тономным приводом, ***рабочая поверхность*** диска *(Side),* связанная с магнитной головкой чтения-записи (Head), магнитная ***дорожка*** *(Track)* и ***сектор*** *(Sector).* Контроллер управляет приводом вращения пакета дисков, приводом возвратно-поступательного перемещения блока магнитных головок с высокоточной системой позиционирова­ ния на заданном радиусе диска, а также процессом последователь­ ного чтения-записи секторов диска магнитными головками1.

Все рабочие поверхности (головки), дорожки (цилиндры) и сек­ торы последовательно пронумерованы, так что номер головки, но­ мер цилиндра и номер сектора вместе однозначно определяют *ап­ паратный адрес* сектора (называемый *относительным адресом).* Используют также и *абсолютную* нумерацию секторов - начиная с нулевого сектора нулевой дорожки нулевой поверхности диска. Если для диска известно количество головок, цилиндров и секторов на каждой дорожке и определен порядок их нумерации, относитель­ ные адреса секторов легко пересчитываются в их абсолютные адре­ са и обратно.

Для однозначного определения места расположения файла на диске необходимо связать с этим файлом упорядоченную после­ довательность пронумерованных секторов.

*Промежуточный уровень модели* обеспечивает интерфейсы вза­ имодействия с пользовательским и аппаратным уровнями. На этом уровне поддерживается *ленточная* модель дискового пространства, представляющая его в форме условной ленты, на которой последо­ вательно размещены секторы в порядке их абсолютной нумерации. Нулевой (в абсолютной нумерации) сектор содержит *главную за­ грузочную запись (Master Boot Record,* MBR), содержащую *исполни­ мый код программы начальной загрузки ОС* (см. п. 5.2.2) и *таблицу разделов диска (Partition ТаЫе),* в которой каждый том представлен

16-байтовым элементом (табл. 6.1).

Сектор - это весьма малая единица емкости дискового накопи­ теля, и использование номера сектора в качестве логического адре­ са файла снижает эффективность использования дисков большой емкости. Поэтому для адресации файлов на *промежуточном уровне* модели дискового пространства используется другая, более крупная единица, называемая *кластером.*

1 Накопитель при этом может быть совсем не *дисковым* и не *магнитным,* а на­ пример, твердотельным (SSD), в котором трудно отыскать *магнитные головки, до­ рожки* и *цилиндры,* но все эти технические детали надежно скрываются от файловой системы контршmером накопителя, который может найти *сектор,* номер которого ему передан, прочесть его содержимое или записать в этот сектор переданную кон­ тршmеру порцию данных.

**114**

Кластер - это блок из нескольких «соседних» секторов (в смысле их абсолютной адресации), размер кластера задается при формати­ ровании тома. Все кластеры тома последовательно пронумерованы, при этом номер кластера однозначно определяет номера всех вхо­ дящих в него секторов.

*Таблицаб.1*

**Структура табnицы раздеnов диска**

|  |  |  |
| --- | --- | --- |
| **Смещение, байт** | **Длина, байт** | **Содержимое** |
| 1-й раздел диска | | |
| +00 | 1 | Флаг загрузки: О - не загружаемый; 80h - загру- жаемый *(ВооtаЫе)* |
| +01 | 1 | Начало раздела: **HdS** (номер головки) |
| +02 | 2 | Начало раздела:  **Sec** (номер сектора- 6 младших битов);  **Cyl** (номер цилиндра - 10 старших битов) |
| +04 | 1 | Код системы: 1- DOS (FAT-12); 4-DOS (FAT-16) |
| +05 | 1 | Конец раздела: **HdE** (номер головки) |
| +Об | 2 | Конец раздела: **Sec** и **Cyl** (аналогично началу раз- дела) |
| +08 | 4 | Абсолютный номер начального сектора раздела (соответствует номерам сектора, головки и цилин- дра начала раздела):  **Cyl** • сект,/дор. • дор./цил. + **Hds** • сект,/дор. +  + **(Sec-1)** |
| +ОСЬ | 4 | Общее количество секторов раздела |
| 2-й раздел диска | | |
| ... | ... | ... |
| Последний раздел диска | | |
| ... | ... | ... |

Модели логических разделов диска реализуются по-разному в различных файловых системах. Далее будут рассмотрены две та­ ких реализации: файловая система типа FAT, разработанная перво­ начально для обслуживания накопителей на гибких магнитных дис­ ках малой емкости, и более современная файловая система NTFS.

Одна из лабораторных работ предусматривает проведение экс­ периментального исследования алгоритмов выполнения типовых файловых операций, реализованных соответствующими программ­ ными компонентами FАТ-систем.

**115**

##### Пользовательский уровень представления файловых систем

В операционной системе MS DOS отсутствовал графический

«оконный» интерфейс, привычный современному пользователю ПК, и единственным способом общения пользователя с ОС бьm так называемый командный интерфейс, базовым элементом которого является понятие «команда».

**Команда** - это средство текстового общения пользователя с операционной системой компьютера, она вводится с клавиатуры и отображается в *командной строке* экрана. В процессе записи ко­ манды ее можно редактировать - до тех пор, пока не нажата кла­ виша *Enter,* после чего команда записывается в специальный буфер ОЗУ1 и начинается процесс ее обработки интерпретатором команд­ ной строки. В MS DOS функции интерпретации команд выполняла системная программа Command.com, в ОС Windows - приложение Cmd.exe.

Процесс интерпретации команд кратко описан в п. 5.2.1 (см. тему 5). Если команда введена корректно, будет выполнена соот­ ветствующая команде системная функция или прикладная програм­ ма, в противном случае будет выдано диагностическое сообщение.

* + 1. Классификация команд

По способу исполнения различают внутренние и внешние ко­ манды, по функциональному назначению - команды управления томами, каталогами и файлами, а также служебные команды. Пол­ ный перечень команд с их краткими описаниями можно получить, выполнив внутреннюю команду *Help,* примеры некоторых команд различных категорий приведены в табл. 6.2.

*Внутренние команды* исполняются соответствующими системны­ ми функциями, доступ к которым получает интерпретатор команд­ ной строки по имени команды, записываемой первой в командной строке. Количество внутренних команд ограничено, их имена - это зарезервированные слова, обозначающие операции, для некоторых внутренних команд допускается использование их сокращенных имен, например: DIR (Directory), DEL (Delete), REN (Rename).

1 В буфере командной строки хранятся несколько введенных ранее команд в по­ рядке их исполнения. Для извлечения команд из буфера в командную строку ис­ пользуются клавиши-стрелки: «стрелка вверх» и «стрелка вниз» - для выбора оче­ редной команды из списка исполненных команд и клавиша «стрелка вправо» для ПОСИМВОЛЬНОГО выбора предыдущей команды. Если команды ИСПОЛНЯЮТСЯ при ак­ тивной программной оболочке Norton Commander (или любом из ее современных аналогов), извлечение очередной команды из буфера осуществляется комбинацией клавиш *Ctrl* + *Е.*

**116**

*Таблицаб.2*

**Примеры команд разnичных типов икатеrорий**

|  |  |  |  |
| --- | --- | --- | --- |
| **Кате­**  **rории** | **Имя команды** | **Тип команды** | **Выполняемая функция** |
|  | имя\_тома: | Внутренняя | Активизация тома (имя\_тома - одна  ИЗ букв ОТ А ДО Z) |
| LAВEL | Внешняя | Отображение и (или) редактирова­ ние метки тома |
| VOL | Внутренняя | Отображение метки тома |
| CHКDSK | Внешняя | Проверка состояния структуры диска (файлы, каталоги, FAT) |
| FDISK | Внешняя | Разбиение жесткого диска на логиче­ ские разделы (тома) |
| FORМAT | Внешняя | Форматирование тома |
|  | DIR | Внутренняя | Вывод оглавления каталога |
| CD | Внутренняя | Изменение текущего каталога |
| MD | Внутренняя | Создание нового каталога |
| RD | Внутренняя | Удаление каталога |
| TREE | Внешняя | Вывод «дерева каталогов» |
|  | СОРУ | Внутренняя | Копирование файлов |
| RENAМE | Внутренняя | Переименование файлов |
| ТУРЕ | Внутренняя | Просмотр текстового файла |
| **MORE** | Внешняя | Постраничный просмотр файлов |
| EDIT | Внешняя | Редактирование текстовых файлов |
| DEL | Внутренняя | Удаление файлов |
| ERASE | Внутренняя | Удаление файлов |
| PRINТ | Внешняя | Печать файлов |
| ХСОРУ | Внешняя | Копирование групп файлов вместе со структурой (деревом) каталогов |
| REPLACE | Внешняя | Замена файлов каталога одноимен­ ными файлами из друrого каталога |
| Q)  i:i! | DATE | Внутренняя | Установка системной даты |
| TIME | Внутренняя | Установка системного времени |
| PROMPT | Внутренняя | Установка формы «приглашения DOS» |
| РАТН | Внутренняя | Установка пути поиска программных файлов |
| VER | Внутренняя | Вывод версии ОС |

**117**

В качестве *внешней команды* используется имя исполнимого фай­ ла *(.сот,* .ехе или .bat). При выполнении внешней команды произ­ водятся поиск указанного в команде файла, загрузка его в память компьютера и запуск на выполнение.

Имя команды включает спецификацию исполнимого файла в со­ ответствии с принятым стандартом: имя\_тома:\путь\имя\_файла. расширение. При этом обязательным является только параметр имя\_файла:

* имя\_тома может быть опущено, если исполнимый файл на­ ходится на активном томе;
* путь к файлу1 может не указываться в случае, если исполни­ мый файл зарегистрирован в текущем каталоге или если путь к ро­ дительскому каталогу этого файла был предварительно задан ко­ мандой РАТН;
* расширение имени файла (.com, .ехе или .bat) может не ука­ зываться, если в целевом каталоге отсутствуют одноименные испол­ нимые файлы.
  + 1. Формат команд

Команда - это текстовая строка, содержащая три компонента:

<имя команды> <параметры команды> /<модификаторы команды> Первым (и единственным обязательным) компонентом являет­

ся *имя команды,* правила написания которого бьmи рассмотрены

выше.

*Параметры команды* - это, как правило, имена объектов (фай­ лов и каталогов), над которыми выполняется операция. Количество, назначение и порядок записи параметров уникальны для каждой команды2. Первый параметр отделяется от имени команды пробе­ лом, если параметров несколько, они также должны быть разделены пробелами. Некоторые команды (например, команда VER) не име­ ют параметров, некоторые команды имеют параметры, но их зна­ чения могут задаваться «по умолчанию»: например, команда VOL, выполненная без параметра «имя тома», выведет метку активного тома, а команда DIR без параметра выведет оглавление текущего каталога.

*Модификаторы команды,* называемые также *ЮLючами,* исполь­ зуются для указания конкретных условий ее применения. Мо­ дификаторы могут располагаться в командной строке как перед

1 Формат строки «путь к файлу» детально описан ниже при описании алгоритма поиска файла. Путь к текущему каталогу в командах не указывается - он известен операционной системе. Если путь начинается с символа«\» (обратная косая черта), он считается заданным от корневого каталога активного диска, если с символов «..» (две точки), то путь задан от родительского каталога.

2 Формат команды и полную инструкцию по ее применению можно вывести на экран путем выполнения этой команды без параметров с ключом «/?».

**118**

параметрами команды, так и после них. В качестве разделителя мо­ дификаторов используется символ «/», который записывается перед очередным модификатором.

Состав модификаторов (так же как и состав параметров) уни­ кален для каждой команды, единственным исключением является модификатор«/?», присутствующий во всех внутренних и многих внешних командах: выполнение команды с таким модификатором приведет к выводу на экран подробной инструкции по применению этой команды.

* + 1. **Примеры использования команд**

Команда **DIR** читает и выводит на экран оглавления каталогов. Единственным параметром этой команды является путь к каталогу:

* DIR (без параметров) - текущий каталог;
* DIR \ - корневой каталог;
* DIR .. - родительский каталог;
* DIR D:\DOC - каталог DOC, подчиненный корневому каталогу;
* DIR С:\ /р - постраничный просмотр корневого каталога;
* DIR Q\Windows /p/s - постраничный просмотр каталога\ Windows и всех подчиненных ему каталогов;
* DIR \*.ехе - просмотр части текущего каталога, содержащей только файлы с расширением .ехе.

Команда **MD** *(Make Directory)* создает подчиненные каталоги.

* MD ddd - создание каталога, подчиненного текущему;
* MD "ЬЬЬ- создание каталога, подчиненного родительскому каталогу;
* MD \first\_d - создание каталога, подчиненного корневому;
* MD D:\first\_d\next\_dir - создание каталога, подчиненного каталогу first\_d.

Команда **CD** *(Change Directory)* - изменение текущего каталога.

* CD \ - переход к корневому каталогу из текущего;
* CD .. - переход к родительскому каталогу из текущего;
* CD next\_dir - переход к дочернему каталогу next\_dir;
* CD next\_dir\ddd\qqq - переход к каталогу qqq (указан путь к новому каталогу, начиная от текущего);
* CD \First\_dir\next\_dir\ddd\qqq - переход к каталогу qqq (указан путь к новому каталогу, начиная от корневого).

Команда **СОРУ** - копирование файлов. Команда использует два параметра: первый указывает спецификацию исходного файла (что и откуда копируется), а второй - спецификацию его копии (куда копируется и под каким именем). Команда может использоваться также для слияния нескольких файлов, печати и вывода на экран или создания (вводом с клавиатуры) текстового файла:

* СОРУ E:\TXT\File\_l.txt C:\TEXТ\File\_2.txt - полноформат­ ная команда копирования файлов, в результате выполнения кото-

119

рой в каталоге ТЕХТ тома С будет создан файл File\_2.txt - копия файла Filel.txt, расположенного в каталоге ТХТ тома Е;

* СОРУ File\_l.txt E:\TEXТ\File\_2.txt - копирование файла под новым именем из текущего каталога в каталог ТЕХТ, подчиненный корневому каталогу тома Е;
* СОРУ Filel.txt D:\- копирование файла из текущего катало­ га в корневой каталог тома D (имя копии совпадает с именем копи­ руемого файла);
* СОРУ Filel + File2 + FileЗ File.All - объединение (слияние) трех файлов в один файл File.All (все файлы - в текущем каталоге);
* СОРУ File.txt PRN - печать файла («копирование» на прин­ тер);
* СОРУ File.txt CON -просмотр файла («копирование» на экран). Здесь в качестве параметра команды используется имя CON (сокра­ щение от Console), зарезервированное операционной системой для стандартных устройств ввода-вывода. При *вводе* данных устройство CON - это клавиатура, при *выводе* - экран видеомонитора. Анало­ гичного результата можно достичь и командой ТУРЕ File.txt;
* СОРУ CON File.txt - создание (копирование с клавиатуры) в текущем каталоге нового текстового файла File.txt. После выпол­ нения такой команды можно вводить с клавиатуры произвольный текст. Для завершения процесса создания файла следует ввести спе­ циальный управляющий символ «конец файла» (*Ctrl* + *Z)* и нажать клавишу *Enter.*

Команда **РАТИ** задает список путей к каталогам, содержащим файлы с исполнимыми программами, имена которых указаны в ка­ честве имен внешних команд. Команда использует единственный параметр- *список путей поиска,* разделенных символом«;» (точка с запятой). После выполнения команды ее параметр (строка со спи­ ском путей к целевым каталогам) помещается в специальный буфер ОЗУ и используется для последовательного поиска соответствующих каталогов в порядке их расположения в списке (слева направо).

Если во внешней команде путь к исполнимому файлу не указан и этот файл отсутствует в текущем каталоге, поиск будет произво­ диться в соответствии со списком путей, введенным командой РАТИ и сохраненным в соответствующем буфере.

Команда РАТИ с параметром «;» очищает буфер, т. е. *отменяет*

все ранее установленные пути поиска исполнимых файлов.

Команда РАТИ, введенная *без параметра,* выводит на экран те­ кущее состояние буфера.

* + 1. Использование групповых имен файлов

В приведенных выше примерах команд предполагалось, что од­ ной командой выполняется соответствующая операция над одним файлом (или файлом-каталогом), имя которого указано в параме-

**120**

тре команды. Использование в имени файла специальных шаблонов (подстановочных символов) позволяет одной командой произвести операцию над целой группой файлов.

Допускается использование в именах файлов двух типов ша­ блонов: символ«?» обозначает «один любой символ», а символ«\*» (звездочка) - «любое количество любых символов» в имени или расширении файла (каталога).

Например, команда ТУРЕ \*.txt последовательно выведет на экран содержимое всех файлов текущего каталога, имеющих расширение

.txt, а команда DEL ??DUM.\* удалит из текущего каталога все файлы с любыми расширениями и именами, состоящими из пяти симво­ лов, оканчивающимися на «DUM».

* + 1. Перенаправление вывода

Многие команды выводят на экран основные результаты своей работы (например, команды DIR и ТУРЕ) и (или) диагностические сообщения (например, команды СОРУ и FORМAT). С помощью спе­ циальных символов «>» или «>>», записываемых в конце командной строки, можно перенаправить выводимую командой информацию на другие устройства или записать ее в текстовые файлы.

Например, команда DIR D:\ > PRN выведет оглавление корневого каталога диска D на принтер, а команда DIR D:\ > dir.lst запишет его в файл dir.lst текущего каталога. Если файл с именем dir. 1st отсут­ ствует, он будет создан в результате выполнения команды.

Если файл уже существует, он будет замещен новым файлом с этим же именем.

Для добавления сообщений в существующий файл используется пара символов «>>», например, команда DIR С:\ >> dir.lst, выпол­ ненная после приведенной выше команды, «допишет» оглавление каталога в конец файла dir.lst.

Фиктивное (не существующее) внешнее устройство с систем­ ным именем NULL используется для подавления вывода сообщений на экран. Например, при успешном выполнении команды СОРУ Filel.txt File2.txt > NULL будет заблокирован вывод стандартного сообщения этой команды «Один файл скопирован» (при этом, есте­ ственно, выполнение команды заблокировано не будет).

Неразумное использование блокировки вывода сообщений ко­ манд может приводить к негативным последствиям, например если в предыдущем примере файл File1. txt не будет найден в текущем каталоге, пользователь не получит соответствующего уведомления (типа «File not found») и будет надеяться, что копия файла создана. Гораздо более неприятная ситуация может возникнуть, напри­ мер, при массовом удалении файлов командой DEL \*.\*>NULL, так как процедура удаления файлов в этом случае сделает паузу **и** за­ просит у пользователя подтверждения выполнения операции «Are

**121**

you sure? Yes/No». Неопытный пользователь, не увидев на экране этого сообщения, может долго ждать, но так и не дождется завер­ шения операции, так как не подозревает о необходимости своего участия в ее продолжении.

* + 1. Программирование пакетных (.bat) файлов

Несколько команд могут быть построчно записаны в текстовый АSСП-файл, который должен иметь стандартное расширение .bat (от англ. *batch-* пакет, пачка). Такой файл является «исполнимым файлом», т. е. имеет статус программы, которую можно выполнить, используя имя файла в качестве внешней команды. При этом каж­ дая команда должна быть записана в отдельной строке файла, и ин­ терпретатор будет последовательно выполнять содержащиеся в па­ кетном файле команды в порядке их записи, подобно тому как если бы они были введены пользователем непосредственно в командной строке.

Пакетные файлы полезны тогда, когда необходимо выполнить часто повторяющуюся последовательность команд. Например, в ре­ зультате выполнения пакетного файла 111.bat (листинг 6.1) в теку­ щем каталоге будет создан новый каталог NewDir, затем в этот ката­ лог будет скопирован из текущего каталога файл qqq.txt под именем ppp.txt, затем каталог NewDir будет установлен текущим и на экран будет выведен текст, содержащийся в файле ppp.txt.

**листинг 6.1. Текст простейшего пакетного файла 111.bat**

MD NewDir

СОРУ qqq.txt NewDir\ppp.txt CD NewDir

ТУРЕ ppp.txt

**Использование переменных.** В приведенном выше примере (см. листинг 6.1) параметры всех команд пакетного файла пред­ ставлены константами, что позволяет усомниться в его полезности (так же как и в полезности любой другой компьютерной програм­ мы, генерирующей одинаковые результаты при каждом ее выпол­ нении: попытайтесь оценить пользу от калькулятора, который за­

программирован на выполнение единственной операции 2 + 3 = 5).

Команды пакетного файла могут содержать имена переменных вместо своих фактических параметров, вместо модификаторов и даже вместо имен команд, что (вместе с наличием условных опе­ раторов и операторов циклов) позволяет считать пакетный файл

«настоящей» компьютерной программой, написанной на языке ко­ мандной строки.

Язык командной строки позволяет использовать в тексте пакет­ ного файла не более 10 различных переменных, имена которых должны состоять из двух символов: первый символ - это всегда знак процента, а второй символ - одна из десятичных цифр. Таким

**122**

образом, переменные оказываются «пронумерованными» в диапа­ зоне от %0 до %9.

В языке командной строки *отсутствует оператор присваива­ ния* - единственным способом присвоения переменным их факти­ ческих значений является передача этих значений через список па­ раметров, которые записываются после имени пакетного файла при выполнении соответствующей внешней команды (подобно тому как это делается при вызовах функций и процедур в языках программи­ рования). Значение параметра списка (отделенного от соседних па­ раметров символом «пробел») автоматически подставляется в текст пакетного файла вместо переменной с соответствующим номером. Например, переменная %2 получит перед выполнением файла зна­ чение второго по порядку параметра из списка, а переменная %5 - значение 5-го параметра.

Пакетный файл 222.bat (листинг 6.2) - функциональный аналог файла 111.bat, но он более универсален, так как содержит имена переменных вместо параметров команд.

**листинг 6.2. Текст пакетного файла 222.bat**

MD %1

СОРУ %2 %1\%3

CD %1

ТУРЕ %3

Результаты выполнения команд

222.bat NewDir qqq.txt ppp.txt

**и**

111.bat

будут одинаковыми, так как в процессе интерпретации текста 222. bat (см. листинг 6.2) будут выполнены соответствующие подстанов­ ки: переменная %1 получит значение 1-го параметра NewDir, вме­ сто переменной %2 будет подставлено значение 2-го параметра qqq. txt, а переменная %3 будет заменена на ррр.txt.

Используя переменные при программировании пакетных фай­ лов, следует учитывать ряд специфических особенностей языка ко­ мандной строки.

1. Все переменные имеют строковый тип данных, что не позво­ ляет применять к ним арифметические операции.
2. Все переменные имеют статус локальных, область их «види­ мости» ограничена пакетным файлом, в котором они использованы. Одноименные переменные, используемые в «головном» и «подчи­ ненном» (вызываемом из головного) пакетных файлах - это раз­ ные переменные.
3. Переменная %0 не соответствует никакому параметру - она получает значение имени пакетного файла (т. е. той части команд­ ной строки, которая расположена левее первого пробела).

**123**

1. Соответствие имен переменных номерам параметров коман­ ды может быть изменено командой SHIFT в процессе исполнения пакетного файла: однократное выполнение этой команды сдвигает список «номеров» переменных на одну позицию вправо относитель­ но списка введенных параметров (см. далее листинг 6.4).
2. Применение команды SHIFT также позволяет при выполне­ нии пакетного файла использовать количество параметров, пре­ вышающее количество используемых в нем переменных (см. далее листинг 6.5).
3. Если список параметров команды оказался короче списка

«номеров» переменных, использованных в тексте пакетного файла (с учетом п. 5), все неопределенные переменные получают значе­ ние «пустая строка».

**Специальные команды пакетных файлов.** Специальные ко­

манды существенно повышают эффективность применения пакет­ ных файлов, делая их полноценными программами. В табл. 6.3 при­ ведены некоторые из таких команд с их кратким описанием. Для получения полного перечня команд с детальными инструкциями по их применению можно воспользоваться командой *Help* или вы­ полнить требуемую команду с модификатором «/?».

*Таблицаб.3*

**Специальные команды пакетных файлов**

|  |  |  |
| --- | --- | --- |
| **Команда** | **Выполняемая функция** | **Формат** |
| ЕСНО | Вывод сообщений, блокировка отображения команд | ЕСНО <произвольный текст> ЕСНО OFF, ЕСНО ON |
| GOTO | Переход на метку | GOTO <метка строки> |
| IF | Условное выполнение команды | IF <условие> <команда\_l> [ELSE < команда\_2>] |
| FOR | Циклическое выполнение команды | FOR %%<переменная цик- ла> IN <список значений>  DO <команда> |
| CALL | Вызов подчиненного Ьаt-файла | CALL <имя> <параметрЫ> |
| SНIFT | Сдвиг списка фактических параметров Ьаt-файла относи- тельно списка используемых переменных | SНIFT |
| **REM** | Записывается в начале строки, блокирует выполнение коман- ды. Обычно используется для записи комментариев в тексте пакетного файла или для вре- менноrо блокирования команд | RЕМ<текст> |

**124**

|  |  |  |
| --- | --- | --- |
| **Команда** | **Выполняемая функция** | *Фор!ШJ. чание табл.6.3* |
| : | Записывается в начале строки (не более 8 символов), при- сваивает этой строке статус метки,которая может исполь- зоваться в командах GOTO | :<Текст> |
| @ | Записывается в начале строки, подавляет отображение этой строки на экране при выполне-  нии пакетного файла | @<команда> |

Листинги 6.3 - 6.7 иллюстрируют возможности специальных команд.

При выполнении пакетного файла, представленного на листин­ ге 6.3, оглавления каталога ТЕХТ в каждом из трех его состояний будут последовательно записаны в файл Dir.lst, расположенный в текущем каталоге. Содержимое этого файла затем будет выведе­ но на экран командой **ТУРЕ.** Стандартные сообщения команд **СОРУ и DEL** выводиться не будут- вместо них на экран будет выводиться текст, указанный в параметрах команд **ЕСНО.**

**Листинг 6.3. Примеры использования команды ЕСНО**

@ЕСНО OFF

**MD ТЕХТ**

DIR ТЕХТ > Dir.lst

ЕСНО Оглавления всех каталогов - в файле Dir.lst

ЕСНО Копирование текстовых файлов из текущего каталога СОРУ \*.ТХТ ТЕХТ\\*.\* > NULL

ЕСНО Копирование завершено DIR ТЕХТ >> Dir.lst

ЕСНО Удаление текстовых файлов из текущего каталога DEL ТЕХТ\\*.ТХТ > NULL

ЕСНО Удаление завершено DIR ТЕХТ >> Dir.lst

ЕСНО Просмотр оглавлений каталогов ТУРЕ Dir.lst

Команда **Shift** (листинг 6.4) сдвигает список параметров пакет­ ного файла на одну позицию влево относительно списка перемен­ ных. Первая команда **ЕСНО** выведет на экран значение первого па­ раметра, а вторая (точно такая же) - значение второго параметра.

**Листинг 6.4. Пример использования команды Shift**

ЕСНО %1

Shift ЕСНО %1

Листинг 6.5 иллюстрирует возможность использования практи­ чески неограниченного количества параметров при ограниченном

**125**

количестве переменных. В этом примере единственная переменная

%1 последовательно получает значения всех параметров, начиная с первого, и каждое ее значение выводится на экран до тех пор, пока список параметров не будет исчерпан (т. е. пока переменная

%1 не получит значения «пусто»).

**листинг 6.5. Пример использования команд IF и Shift**

:Loop ЕСНО %1 SHIFT

IF -%1==- GOTO Exit

GOTO Loop

:Exit

ЕСНО Список параметров исчерпан

Листинг 6.6 представляет команду **FOR,** обеспечивающую ци­ клическое выполнение других команд. Переменные цикла %%а,

%%Ь, %%с и %%d последовательно получают значения из списка, заданного в разделе IN, и используются в качестве параметров ко­ манд, заданных в разделе DO.

**листинг 6.6. Примеры использования команды FOR**

FOR %%а IN (Dirl Dir2 Dir3) DO MD %%а

FOR %%Ь IN (Dirl Dir2 Dir3) DO СОРУ Filel %%Ь\ FOR %%с IN (\*.%1) DO ЕСНО %%с

FOR %%d IN (txt doc xls) DO IF exist %1.%%d СОРУ %1.%%d DIRl\

Листинг 6.7 иллюстрирует использование команды **CALL,** с по- мощью которой из головного пакетного файла (QQQ.bat) вызывает­ ся подчиненный (PPP.bat) с передачей ему четырех параметров, два из которых - это локальные переменные головного файла, а два других - текстовые константы.

**листинг 6.7. Пример использования команды CALL *Файл***

QQQ.bat @ЕСНО OFF СОРУ %2 %1\%3

CALL PPP.bat File\_l %3 %1 File\_2 ТУРЕ %1\File\_2 Файл PPP.bat @ЕСНО OFF

ЕСНО Исходные файлы %1 и %2 CD %3

СОРУ %1+%2 %4 >Null

ЕСНО Слияние файлов завершено

Пусть оба файла размещены в одном каталоге. При выполнении команды

QQQ.bat Dirl Namel Name2

переменные %1, %2 и %3 головного файла QQQ.bat получат значе­ ния соответственно Dirl, Namel и Name2.

Вторая команда этого файла скопирует файл Namel под именем Name2 в подкаталог Dirl. Следующая команда **CALL** выполнит файл PPP.bat, передав ему в качестве первого параметра текстовую кон­ станту File\_l, в качестве второго параметра - значение перемен­ ной %3 головного файла (т. е. Name2), в качестве третьего пара-

**126**

метра - значение первого параметра головного файла (т. е. Dirl), а в качестве четвертого параметра - текстовую константу File\_2.

В результате переменные %1, %2, %3 и %4 подчиненного фай­ ла PPP.bat получат значения соответственно File\_l, Name2, Dirl и File\_2, команда **СОРУ** соединит два файла File\_l и Name2 и сохра­ нит результат соединения в файле File\_2.

После завершения работы подчиненного пакетного файла РРР. bat будет выполнена команда **ТУРЕ** головного файла, которая вы­ ведет на экран содержимое файла File\_2.

Можно вызвать подчиненный пакетный файл из головного и без команды **CALL:** вместо строки

CALL PPP.bat File\_l %3 **%1** File\_2

записать строку

PPP.bat File\_l %3 %1 File\_2

Однако в этом случае после завершения работы подчиненного файла не произойдет возврата к головному файлу, и вся программа завершит работу последней командой подчиненного файла.

##### Файловые FАТ-системы

Как уже отмечалось, жесткий диск может быть разделен на не­ сколько логических разделов (томов), каждый из которых представ­ ляется пользователю как автономный диск. Каждый том занимает определенный ему при форматировании диапазон секторов дис­ ка, номера которых указаны в соответствующей строке таблицы разделов. При рассмотрении структуры тома будем использовать внутреннюю нумерацию секторов в пределах тома: первый из вы­ деленных тому секторов будем называть *нулевым* сектором, следу­ ющий - *первым* и т. д.

В файловой системе FAT том разделен на две расположенные по­ следовательно области - системную и рабочую.

*Рабочая область* занимает основную часть тома и расположена непосредственно после системной области. Рабочая область (в от­ личие от системной) разделена на последовательно пронумеро­ ванные *кластеры* и предназначена для хранения файлов и подчи­ ненных каталогов (которые, по существу, также являются файлами специального формата).

Кластер используется в качестве минимальной единицы, выде­ ляемой операционной системой одному файлу, и этот файл будет единственным «владельцем» каждого из выделенных ему кластеров, независимо от того, осталось ли в кластере незанятое пространство после записи в него фрагмента файла.

**127**

Каждый кластер имеет уникальный номер и содержит несколь­ ко расположенных подряд секторов (как правило, от двух до 16, но допускаются и кластеры, состоящие из одного сектора). Номера кластеров используются в качестве адресов расположения файлов, между номером кластера и списком номеров входящих в него сек­ торов существует взаимно-однозначное соответствие.

*Системная область* (существенно меньшая по размеру) зани­ мает на томе несколько начальных секторов, начиная с нулевого, и используется для хранения служебных структур данных, необхо­ димых для организации доступа к файлам и загрузки операционной системы.

Секторы системной области не объединяются в кластеры, запись файлов любого типа в эту область запрещена.

* + 1. Структура системной области тома

Системная область содержит три служебные структуры данных: блок загрузки *(Boot Record);*

* таблица расположения файлов *(File Allocation ТаЫе,* FAT);
* корневой каталог *(Root Directory).*

*Блок загрузки* (табл. 6.4) занимает нулевой сектор и содержит та­ блицу параметров формата диска и короткую программу загрузки ОС (п. 5.2.2). Все данные блока загрузки записываются в том в про­ цессе его форматирования соответствующей программой, инфор­ мация о которой сохраняется в первой записи таблицы параметров (В-байтовый текстовый блок во второй строке табл. 6.4).

*Таблицаб.4*

**Структура б11ока заrрузки *(Ьооt record)***

|  |  |  |
| --- | --- | --- |
| **Смещение,**  байт | **Длина, байт** | **Содержимое элемента блока загрузки** |
| +00 | з | Команда перехода на программу загрузки (JMP) |
| +03 | 8 | Служебная информация программы форматирова-  **ния** |
| +ОВь | 2 | Размер сектора в байтах |
| +ОDь | 1 | Размер кластера в секторах |
| +ОЕь | 2 | Количество секторов перед первой таблицей FAT |
| +lОь | 1 | Количество таблиц FAT |
| +llь | 2 | Максимальное число элементов корневого каталога |
| +1Зь | 2 | Общее количество секторов на томе |
| +15ь | 1 | Дескриптор носителя (то же, что 1-й байт FAT) |
| +lбь | 2 | Количество секторов, выделенных одной FAT |

**128**

|  |  |  |
| --- | --- | --- |
| **Смещение, байт** | **Длина, байт** | **Содержимое элемента блока *:lfP 1l'/.5f106Л,*** *б.4* |
| +18ь | 2 | Количество секторов на дорожке |
| +1Аь | 2 | Количество головок чтения/записи (поверхностей) |
| +1Сь | 2 | Количество спрятанных секторов |
| +lEH |  | Начало программы начальной загрузки ОС |

*Таблица расположения файлов* (FAT) занимает в системной обла­ сти фиксированный набор секторов (см. табл. 6.4), расположенных непосредственно после загрузочного сектора, и используется для хранения номеров свободных кластеров и номеров кластеров, вы­ деленных файлам при их записи.

FAT - это информационная модель рабочей области тома, пред­ ставленная в форме *списка,* каждый элемент которого содержит числовой (двоичный) код, представляющий некоторую характери­ стику кластера рабочей области, соответствующего этому элементу (табл. 6.5).

*Таблицаб.5*

**Чисповые коды эпементов FAT**

|  |  |  |
| --- | --- | --- |
| **Значение**  **FAT[i]** | **Состояние**  **i-го кластера** | **Комментарии** |
| (О)ОООь | Кластер свобо- ден и доступен для записи | При форматировании тома (в любом из ре- жимов - *quick илиfиll)* все кластеры, кроме  «сбойных», объявляются свободными, т. е. все элементы FAT получают нулевые значе- ния |
| от (0)002ь до (F)FEFь | Кластер занят и содержит не последний  фрагмент файла | Код FAT[i] - это номер следующего класте- ра, в котором продолжается (или заканчи- вается) файл, очередной фрагмент которого расположен в i-том кластере |
| от (F)FFBь до (F)FFFь | Кластер занят и содержит по- следний фраг- мент файла | FAТ[i] - это код окончания «цепочки» кла- стеров, занятых файлом *(End Of File,* EOF) |
| от (F)FF0ь до (F)FF7ь | Кластер «сбой- ный», не досту- пен для чтения и записи | При *полном* форматировании тома *(full format)* весь кластер объявляется «сбойным» *(Bad),* если «сбойным» оказывается хотя  бы один его сектор |

Если при форматировании тома в его рабочей области создано *п* кластеров, то в системной области будет сформирована структура FAT (точнее - две идентичных копии этой структуры), содержа­ щая список из *п* элементов, при этом каждый элемент этого списка

**129**

FAT[i] представляет соответствующий *(i-й)* кластер рабочей обла­ сти тома.

Размер элемента FAT (разрядность соответствующего двоичного кода) ограничивает количество кластеров, создаваемых в рабочей области тома при его форматировании. Если, например, попытаться отформатировать том в 12-битной системе FAT-12 (когда-то приме­ нявшейся для гибких дисков), то на этом томе нельзя будет сформи­

ровать более 212 = 4096 кластеров, а фактически - чуть меньше, так

как 18 кодов зарезервированы для хранения служебной информации и не могут использоваться для «нумерации» кластеров (см. табл. 6.5). Если при этом емкость тома не превышает 2 Мбайт (221 бит),

то его удастся отформатировать с минимально возможным раз­ мером кластера, равным одному сектору (221/212 = 29 = 512 бит), но для томов большей емкости, форматируемых в FAT-12, создание

таких «мелких» кластеров будет технически (и «арифметически») невозможным.

На томе, отформатированном в системе FAT-16, станет воз­ можным создание 65 518 кластеров, а в системе FAT-32 - уже 4 294 967 278, что практически снимает ограничение на мини­ мальный размер кластера: например, для тома емкостью 2 Тбайт

(241бит), форматируемого в FAT-32, минимальный размер кластера составит один сектор (241/232 = 29 = 512 бит) - так же, как и для

тома емкостью 2 Мбайт, отформатированного в системе FAT-12.

Какой же размер кластера следует выбрать при форматирова­ нии тома? Все зависит от его предназначения. Если, например, том используется для хранения файлов - коротких сообщений на по­ чтовом сервере, то эффективность использования дискового про­ странства будет выше при кластерах минимального размера, так как даже самому короткому файлу не может быть выделено меньше одного кластера. Если же том используется на сервере баз данных, файлы которых, как правило, имеют гигабайтные размеры, эффек­ тивнее будут крупные кластеры, так как при этом ускоряется обра­ ботка FAT и сокращается размер системной области тома.

Программы форматирования предлагают (по умолчанию) разме­ ры кластеров от 4 до 8 секторов (соответственно, от 2 до 4 Кбайт) в зависимости от емкости форматируемых дисков.

Какую же разрядность FAT (12, 16 или 32 бита) выбрать при форматировании тома? Повышение разрядности экспоненциально увеличивает допустимое количество кластеров, но при этом увели­ чивается и размер системной области, которая «отнимает» часть дискового пространства у рабочей области тома, используемой для хранения файлов. При этом в ряде случаев, особенно при форма­ тировании томов небольшой емкости, увеличение разрядности FAT бывает избыточным и создает негативный эффект.

**130**

Размер фрагмента системной области тома, резервируемого для хранения FAT, определяется количеством ее элементов (равным ко­ личеству кластеров рабочей области), и размером каждого такого элемента.

В первом из рассмотренных выше примеров том размером 2 Мб был отформатирован в системе FAT-12 с 512-байтовыми кла­ стерами. При этом в рабочей области было создано приблизительно 4096 кластеров (221/29 = 212), и столько же 12-битных элементов

было сформировано в FAT. В этих условиях каждая из двух копий FAT будет занимать в системной области тома по 12 секторов (4096 • 12 / 8 / 512), отнимая этот объем у рабочей области тома.

Если этот же том отформатировать в системе FAT-16 с таким же раз­ мером кластеров, то FAT увеличится до16 секторов (4096 • 16 / 8 / 512),

а при форматировании в FAT-32 - до 32 секторов (4096 • 32 / 8 / 512). Как уже отмечалось, структура FAT первоначально разрабаты­ валась для дисковых устройств небольшой емкости, когда главным критерием качества была эффективность использования дискового пространства при хранении файлов. Принятая в FAT система коди­ рования кластеров (см. табл. 6.5), безусловно, соответствует этому критерию, но она противоречит другому, не менее важному, крите­ рию, так как способствует фрагментарному расположению файлов, что увеличивает время доступа к секторам диска на аппаратном

уровне.

Для иллюстрации основных концепций FAT рассмотрим неболь­ шой ее фрагмент, описывающий диапазон рабочей области тома из 32 кластеров - с 10-го по 41-й (рис. 6.1). На рисунке *курсивом* обозначены номера элементов FAT, а **жирным шрифтом** - число­ вые коды, записанные в этих элементах. Разумеется, все числовые коды FAT записаны в двоичной системе счисления, но для удобства прочтения все номера кластеров представлены на рисунке в деся­ тичной системе, а специальные коды EOF и BAD - в шестнадцате­ ричной.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *10*  о | *11*  о | | *13* | *14* | *15* | | *16* | | *17* | |  | *19* | *20* | *21* | *22* | *23* | *24* | *25* |
| **14** | **15** | **FFF** | | **О** | | **О** | | **19** | **20** | **21** | **31** | **О** | **О** | **О** | **О** |
|  |  | | |  |  | | | | | | |  |  |  |  |  |  |  |
| *26* | *27* | *28* | *29* | *30* | *31* | *32* | | *33* | | *34* | | *35* | *36* | 37 | *38* | *39* | *40* | *41* |
| о | о | о | о | о | 32 | **33** | | 35 | | FF7 | | **36** | 37 | FFF | о | о | о | о |

*Рис. 6.1.* **Фраrмент таблицы FAT**

Как следует из табл. 6.5, кластеры с номерами 10, 11, 16, 17, с 22-го по 30-й и с 38-го по 41-й свободны и доступны для записи, так как в соответствующих этим кластерам элементах FAT записан код «О».

**131**

Кластер № 33 объявлен «сбойным» (bad-cluster), так как код **FF7ь,** записанный в 33-м элементе FAT, блокирует возможность до­ ступа к этому кластеру программным компонентам FAT. Разумеет­ ся, программа форматирования диска такой доступ получит беспре­ пятственно, как, впрочем, и любая другая программа, работающая с контроллером диска напрямую без использования соответству­ ющих драйверов и функций DOS или BIOS (как это показано пун­ ктирной линией на рис. 5.1).

В кластерах № 15 и № 37 заканчиваются какие-то два файла, на что указывает код FFFь (EOF) в соответствующих элементах FAT. Цепочка ссылок в элементах «12-13-14-15» FAT позволяет предположить, что кластеры с номерами с 12-го по 15-й содержат последний фрагмент какого-то файла (размер этого фрагмента

не превышает суммарного размера четырех кластеров тома)..

Второй файл, заканчивающийся в 37-м кластере, записан на диск фрагментарно, и его последний фрагмент последовательно зани­ мает три фрагмента «цепочки» кластеров: [18 + 21] + [31 + 33] +

+ [35 + 37]. Последние два фрагмента разделены ВАD-кластером

№ 34, а первые два разделены девятью свободными кластерами - возможно, в момент записи второго файла эти кластеры были за­ няты другими файлами, которые позднее были удалены, что и при­ вело к «обнулению» соответствующих элементов FAT.

Вероятно, по этой же причине между 1-м и 2-м файлами присут­ ствуют свободные кластеры № 16 и № 17, а также свободные кла­ стеры № 10 и № 11, почему-то не выделенные первому файлу при его записи на диск1.

Структура FAT позволяет эффективно отследить всю цепочку за­ нятых файлом кластеров, при условии, что известно начало этой цепочки. Конечно, начальные элементы всех цепочек кластеров можно определить алгоритмически (как элементы, ссылающиеся на другие элементы, но на которые не ссылается ни один из дру­ гих элементов), но разработчиками FAT было принято другое, более простое решение: номер начального кластера для каждого файла хранится не в FAT, а в специальной структуре данных, называемой *каталог.ом* (корневым или любым из подчиненных), наряду с дру­ гими свойствами файла.

Именно с просмотра каталога и начинает свою работу рассмо­ тренный ниже алгоритм поиска файла.

*Корневой каталог.* - это последняя из трех структур данных, соз­ даваемых в системной области тома при его форматировании. Кор-

1 Алгоритмы поиска свободного дискового пространства, реализуемые функци­ ями FAT при записи файла на диск, рассматриваются в п. 6.3.2 - они неоднократно изменялись при обновлении версий ОС. Экспериментальному исследованию этих алгоритмов посвящена одна из лабораторных работ, а их программной реализа­ ции - одно из контрольных заданий.

**132**

невой каталог ограничен в размере соответствующим параметром, хранимым в блоке загрузки (см. табл. 6.4), и занимает последова­ тельность секторов, расположенных непосредственно после второй копии FAT.

Корневой каталог состоит из множества регистрационных за­ писей, каждая из которых содержит информацию об одном файле (в том числе - о файле типа «подчиненный каталог»), зарегистри­ рованном в корневом каталоге.

Запись корневого каталога имеет фиксированную длину в 32 бай­ та, ее структура представлена в табл. 6.6.

*Таблицаб.6*

**Структура записи корневоrо катаnоrа**

|  |  |  |
| --- | --- | --- |
| **Смещение** | **Длина, байт** | **Содержимое** |
| +00 | 8 | Имя файла |
| +08 | 3 | Расширение файла |
| +ОВь | 1 | Атрибуты файла |
| +ОСь | ОАь | Резерв |
| +lбi, | 2 | Время создания/модификации (в спец. формате) |
| +18ь | 2 | Дата создания/модификации (в спец. формате) |
| +1Аь | 2 | Номер начального кластера |
| +1Сь | 4 | Размер файла в байтах |
| +20ь | 1 | Размер элемента каталога (как правило, равно 20ь) |

*Имя* и *расширение* имени файла. Как показано в табл. 6.6, под имя и расширение имени файла отведены два первых поля записи суммарной длиной всего 11 байтов, что соответствует 8-битному АSСП-кодированию текстовых символов и соглашению об именова­ нии файлов, принятому в MS DOS (не более 8 знаков в имени файла и не более трех - в расширении.

Уже в ранних версиях ОС Windows это ограничение было снято, и сегодня полное имя Windows-фaйлa формально может содержать до 255 знаков (с возможностью использования почти всех симво­ лов, в том числе нескольких символов «точка», а также символов дополнительных кодовых таблиц), что восьмикратно превышает размер всей регистрационной записи каталога.

Хранение «длинных» имен файлов в «коротких» записях катало­ гов необходимо для обеспечения совместимости файловых систем MS DOS и Windows, и эта проблема была решена следующим об­ разом:

* из «длинного» Windows-имeни по специальному алгоритму вы­ деляются 11 символов (8 + 3), соответствующих требованиям MS DOS;

**133**

* это программно сформированное «короткое» имя вместе с остальными параметрами файла хранится стандартным для MS DOS способом в 32-байтовой регистрационной записи каталога;
* исходное «длинное» Windows-имя файла хранится как не­ структурированный текст, занимая необходимое количество допол­ нительных 32-байтовых записей этого же каталога.

Короткое DОS-имя файла формируется из длинного Windows­ имeни по следующему алгоритму, приводимому с некоторыми упрощениями:

* из исходного длинного имени файла удаляются все запре­ щенные правилами MS DOS символы, в том числе и пробелы, а так­ же все точки, кроме одной (самой правой) точки, которая получает статус разделителя между именем файла и расширением имени;
* из числа оставшихся символов, расположенных правее точки, отбрасываются все крайние правые символы, кроме трех прижатых к точке символов, которые становятся расширением имени файла;
* из числа символов, расположенных левее точки, оставляются только 6 крайних левых символов, остальные отбрасываются, по­ сле чего справа добавляется седьмой символ тильда (~) и восьмой цифровой символ (1-9) в зависимости от количества одноимен­ ных файлов, уже зарегистрированныхв каталоге (т. е. файлов с оди­ наковыми шестью первыми символами в именах и тремя одинако­ выми символами в расширениях);
* если после такого «усечения» имени в нем остались строч­ ные буквы дополнительной кодовой таблицы (например, символы кириллицы), все они заменяются соответствующими прописными буквами.

Если, например, файл имеет имя *<<Длинное* . *Имя* . *Файла* . *русское»,*

то будет сформировано его короткое имя *<<ДЛИННО~ 1.РУС»* или

*«.ДЛИНН0~2.РУС»,* если в каталоге уже имеется один файл с такими же именем и расширением.

*Атрибуты* файла. Третье поле записи длиной в 1 байт отведено для хранения *атрибутов* файла- шести битовых флагов, единич­ ное значение каждого из которых задает определенное свойство файла:

1. й бит= 1 - файл доступен только для чтения **(R/0** - Read Only);
2. й бит= 1 - файл объявляется *спрятанным* **(Hid** - Нidden), т. е. штатные программы просмотра каталогов не должны его по­ казывать в списке файлов;
3. й бит = 1 - системный файл ***(Sys*** - System);
4. й бит = 1 - фиктивный файл, имя и расширение которого ис­ пользуются в качестве *метки тома* ***(Vol*** - Volume);
5. й бит = 1 - файл специального формата - подчиненный ката­ лог ***(Dir-*** Directory);

**134**

1. й бит= 1 - файл, требующий архивирования при создании очередной резервной копии диска ***(Arc).***

Информация, хранимая в атрибутах файлов, используется опера­ ционной системой при выполнении файловых операций.

*Время* и *дата* создания файла описываются двухбайтовыми бло­ ками данных в специальных форматах.

*Номер начального кластера* - это та самая ссылка, с которой на­ чинается сканирование FAT в процедуре поиска номеров кластеров, выделенных файлу при его записи на диск.

*Размер файла* - числовая характеристика файла, определяемая объемом хранимых в файле данных. Учитывая тот факт, что файлу при его записи на диск выделяется целое число кластеров, его раз­ мер практически всегда будет меньше, чем размер занятого им дис­ кового пространства.

* + 1. Алгоритмы выполнения файловых операций

Стандартный набор операций над объектами файловой системы (дисками, каталогами и файлами) уже обсуждался выше в п. 6.1, все эти операции доступны пользователям Windows-cиcтeм через множество «контекстных меню», а пользователям MS DOS - через интерфейс «командной строки». Независимо от реализации поль­ зовательского интерфейса, файловые операции выполняются со­ ответствующими программными компонентами ОС, доступными также и прикладным программам через вызовы соответствующих системных функций.

Программная реализация любого алгоритма требует определен­ ной организации обрабатываемых данных. Системные функции, реализующие файловые операции, используют структуры данных, сформированные в служебных областях дисковых устройств *(MBR, Boot Record, FAT* и *Root Directory),* подчиненные каталоги, хранимые в рабочей области в файлах специального формата, а также слу­ жебные структуры данных, организованные в СМОS-памяти. Для ускорения работы файловых функций дисковые структуры данных копируются (и регулярно обновляются) в специальные буферы опе­ ративной памяти, количество которых определяется соответствую­ щей командой системного конфигурационного файла CONFIG.sys (см. п. 5.2.1).

Обсуждая алгоритмы работы файловых функций, будем предпо­ лагать, что этим функциям доступны все структуры данных файло­ вой системы, и при вызове функции ей передается текстовая стро­ ка, содержащая соответствующую исходную информацию - как правило, это имена объектов файловой системы и различные мо­ дификаторы. При этом неважно, введена ли эта информация поль­ зователем в «командной строке» для ее последующей обработки интерпретатором COMМAND.com, программно сформирована при-

135

ложением «контекстное меню» графического пользовательского ин­ терфейса или сгенерирована прикладной программой - в любом случае она будет передана файловой системе через программный интерфейс функций DOS.

Алгоритм активизации тома

Для активизации тома требуется загрузить в ОЗУ структуры дан­ ных из его системной области, что сделает этот диск оперативно доступным.

Очевидно, это самая алгоритмически простая из файловых опе­ раций, для выполнения которой пользователю достаточно ввести в командной строке имя тома с двоеточием в конце, например, D: или F:, или выбрать диск из списка.

Соответствующая DОS-функция, получив имя тома, который следует активизировать, определяет тип устройства (сменный или фиксированный) и адрес порта ввода-вывода, связанного с этим устройством.

Если получено имя сменного устройства, на котором при форма­ тировании создается единственный том, достаточно загрузить в бу­ фер ОЗУ его нулевой Ьооt-сектор, определить по таблице параме­ тров форматирования номера секторов системной области, занятых таблицей FAT и корневым каталогом, и загрузить в буфер ОЗУ эти системные структуры данных.

Если полученное имя идентифицирует один из разделов фик­ сированного диска, алгоритм немного усложняется: обращением к *таблице разделов* диска (которая уже скопирована в буфер ОЗУ в процессе начальной загрузки ОС) определяется адрес Ьооt-сектора этого раздела и далее - аналогично описанному выше алгоритму для сменного запоминающего устройства.

Алгоритм поиска файла

Алгоритм поиска файла «вложен» в алгоритмы реализации многих файловых функций - прежде, чем файл прочитать, ско­ пировать, удалить или выполнить, надо, как минимум, его найти в файловой системе, т. е. определить последовательность номеров кластеров, занятых этим файлом в томе (или установить факт от­ сутствия искомого файла).

Алгоритм поиска файла проиллюстрируем на примере команды ТУРЕ, используемой для вывода на экран монитора содержимого текстовых файлов АSСП-формата, при этом процесс визуализации текста в уже «найденном» файле рассматривать не будем.

Задача заключается в том, чтобы найти файл, «путь» к которому задан соответствующим значением параметра команды, например, такой:

ТУРЕ \my\_docs\my\_file.txt.

**136**

Единственный параметр команды ТУРЕ (часть текста командной строки, расположенная правее первого пробела) - это *путь к фай­ лу,* содержащий список имен подкаталогов с именем искомого фай­ ла в конце списка.

Строка «путь к файлу» формируется по следующим правилам:

- имена каталогов располагаются в списке слева направо в по­ рядке их подчиненности: от родительских каталогов к дочерним;

в качестве разделителя имен в списке используется символ

«\»;

если строка начинается с символа«\» (как в нашем примере),

то путь считается заданным от корневого каталога текущего тома;

* если строка начинается с двух символов «точка»(..), то путь считается заданным от каталога, являющегося родительским для ка­ талога, имя которого указано первым;
* в остальных случаях путь считается заданным от текущего каталога, «открытого» предшествующими командами.

1. **й шаг** алгоритма (для любой файловой операции) - это ин­ терпретация строки параметров команды. В случае с командой ТУРЕ параметр всего один, для более сложных команд параметры в строке (если их несколько) разделены символом «пробел». В даль­ нейшем при описании алгоритмов реализации файловых операций этот «нулевой» шаг алгоритма будем опускать, предполагая извест­ ными все компоненты «пути к файлу» - имена подкаталогов в по­ рядке их подчиненности.

В результате интерпретации строки с параметром команды полу­

**чим:**

* родительский подкаталог файла подчинен корневому ката­ логу тома (так как текущим каталогом по условию нашей задачи является корневой каталог);

имя родительского подкаталога искомого файла-mу\_dосs;

* имя искомого фaйлa-my\_file;
* расширение имени файла - txt.

1. **й шаг** алгоритма - поиск текущего каталога, т. е. каталога,

«открытого» предшествующими командами, от которого и начина­ ется путь к файлу, заданный в команде. Как правило, открытый ка­ талог (или только первый кластер, выделенный этому каталогу) уже загружен в соответствующий буфер ОЗУ, и «искать» его на диске не требуется, так как он ранее *бьт* найден процедурой, рекурсивно использовавшей описываемый здесь алгоритм, начиная от корнево­ го каталога активного диска.

Адрес корневого каталога (номера его начального и конечного секторов системной области диска) достаточно просто определя­ ются по данным таблицы параметров форматирования тома (см. табл. 6.4), гарантированно загруженной в буфер ОЗУ в составе ну­ левого Ьооt-сектора активного тома.

**137**

1. й шаг алгоритма - сканирование текущего каталога (см. табл. 6.6) с целью поиска объекта, подчиненного текущему ката­ логу. В рассматриваемом примере объект поиска - файл с име­ нем my\_docs (без расширения), являющийся подчиненным катало­ гом (4-й бит байта атрибута равен единице). Если такого объекта не найдено, поиск будет прекращен с выводом диагностического сообщения, в противном случае - запоминается *номер начального кластера* из множества кластеров, занятых объектом.
2. **й шаг** алгоритма - формирование списка номеров кластеров, занятых подкаталогом *my\_docs,* путем сканирования таблицы FAT (она тоже загружена в буфер ОЗУ), начиная с элемента, равного но­ меру начального кластера, и далее по ссылкам, пока не будет най­ ден элемент с кодом *EOF* (см. табл. 6.5).
3. **й шаг** алгоритма - преобразование списка номеров кластеров в список номеров секторов и передача этого списка контроллеру диска для загрузки содержимого этих секторов в буфер ОЗУ.

Далее рекурсивно выполняются шаги со 2-го по 4-й до тех пор, пока не будет найден объект, имя которого указано последним в строке «путь к файлу».

**Алгоритм переименования файла**

Для изменения имени файла используется команда *RENAМE* (или *REN).* У этой команды есть два параметра (две разделенных пробе­ лом текстовых подстроки): первый параметр - путь к переимено­ вываемому файлу, а второй - новое имя файла, например:

RENAМE \my\_docs\my\_file.txt new\_name.doc

Вначале запускается рассмотренный выше алгоритм поиска фай­ ла - с той разницей, что остановить поиск надо чуть раньше, так как для переименования файла его содержимое не требуется, до­ статочно «найти» только его имя.

Если при сканировании подкаталога *ту\_docs* будет найдена ре­ гистрационная запись объекта с именем *ту\_Ji.le* и с расширением *txt,* и при этом переименование объекта не запрещено (т. е. 0-й бит байта атрибутов объекта будет иметь нулевое значение), то поля

«имя» и «расширение» в регистрационной записи будут заменены на новые, в нашем примере - на *new\_пате* и *doc.*

**Алгоритм поиска свободного пространства на томе**

Рассмотрим два варианта использования этого алгоритма: в пер­ вом случае алгоритм используется для оценки общего объема сво­ бодного пространства тома, а во втором - для поиска свободных кластеров, необходимых для записи на том файла заданного размера. В обоих случаях в основе алгоритма лежит процедура сканирова­ ния таблицы FAT, определяющая количество (и, соответственно, но-

**138**

мера) элементов FAT, содержащих нулевые значения, так как каж­ дый такой элемент определяет свободный кластер рабочей области тома. Зная размер кластера в байтах (см. табл. 6.4), легко определя­ ется и размер свободного пространства.

Для оценки общего объема свободного пространства тома ска­ нируется *вся таблица FAT* и определяется количество ее нулевых элементов, которое затем умножается на размер кластера в байтах. Во втором случае известен размер копируемого файла (в бай­ тах), и требуется определить список номеров кластеров, количество которых будет минимально достаточным для записи этого файла на том. Алгоритм может быть реализован следующей последова­

тельностью шагов.

1. **й шаг** - вычисляется количество требуемых кластеров (деле­ нием размера файла на размер кластера с округлением до большего целого).
2. **й шаг** - последовательным просмотром таблицы FAT опреде­ ляется необходимое количество нулевых элементов с сохранением списка номеров найденных элементов.
3. **й шаг** - в каждом (кроме последнего) элементе сформирован­ ного списка вместо нуля записывается номер следующего за ним элемента, а в последнем элементе записывается код окончания фай­ ла FFFь (EOF).
4. **й шаг** - сохраняется номер первого элемента списка для его последующей вставки в новую регистрационную запись родитель­ ского каталога (поле «номер начального кластера», см. табл. 6.6).

Так как FAT обеспечивает возможность фрагментарного хране­ ния файла, существует множество вариантов реализации второго шага рассмотренного выше алгоритма. В ранних версиях ОС, когда ПК оснащались дисковыми накопителями малой емкости, главным критерием качества алгоритма была эффективность использования дискового пространства. В этих условиях функции поиска свобод­ ных кластеров использовали «скупой»1 алгоритм, который, к тому же, самый экономичный в реализации: в процессе сканирования та­ блицы FAT, начиная с ее первого элемента, номер каждого нулевого элемента FAT сохраняется в списке номеров кластеров, пока длина этого списка не станет достаточной.

Такой алгоритм минимизирует количество мелких фрагментов свободных кластеров, но увеличивает количество фрагментарно хранимых файлов, что снижает производительность дисковых опе­ раций и вынуждает пользователя ПК чаще проводить операцию дефрагментации дискового пространства.

1 Намеренно не используем здесь слово «жадный», так как название *«жадный алгоритм» (greedy algorithm)* уже занято - этим термином принято обозначать определенную группу алгоритмов принятия оптимальных решений.

**139**

Другие, менее «скупые», алгоритмы поиска свободных кластеров при сканировании таблицы FAT пренебрегают теми непрерывными фрагментами свободных кластеров, длины которых недостаточны для хранения всего файла, и формируют список номеров кластеров из состава кластеров, принадлежащих какому-то одному крупному фрагменту. При отсутствии крупных фрагментов выполняются раз­ личные модификации «скупого» алгоритма, минимизирующие ко­ личество непрерывных фрагментов кластеров, выделяемых файлу.

Алгоритм удаления файла

Какую цель преследует пользователь ПК, удаляя файл1, и какими будут последствия успешного завершения этой процедуры? Очевид­ но, что главная цель связана с необходимостью освобождения места2 для записи других файлов, а видимых пользователю последствий уда­ ления файла всего два: во-первых, стандартные утилиты просмотра каталогов (например, DОS-команда *DIR)* не показывают удаленные файлы в списках объектов их родительских каталогов, и, во-вторых, свободного места в томе стало больше, причем ровно на то количе­ ство кластеров, которое занимал удаленный файл (в чем легко убе­ диться, анализируя свойства тома до и после удаления файла).

При рассмотрении алгоритма удаления файла следует иметь в виду, что технологии записи двоично-кодированной информации на компьютерные носители не требуют физического удаления дан­ ных с носителя перед тем, как на их место будут записаны новые, также двоично-кодированные, данные.

И еще одна деталь, уже не технологического, а, скорее, психоло­ гического характера: кто из нас не попадал в ситуацию, когда уда­ ленный вчера файл, казавшийся нам совершенно ненужным, уже сегодня оказывается жизненно необходимым и требующим опера­ тивного восстановления (и при этом, желательно - без потери по­ лезной информации)?

Очевидно, примерно так была поставлена задача разработчи­ кам алгоритма удаления файла, реализованного ими при разра­ ботке файловой функции, связанной с DОS-командой *DEL «путь-к­ удаляемому-файлу».*

Алгоритм реализуется следующими последовательными шагами. **1-й шаг-** выполняется рассмотренный выше алгоритм поиска удаляемого файла, в результате которого отыскивается регистраци­ онная запись этого файла в родительском каталоге и определяется

1 Здесь не идет речь о привычной современному пользователю ПК «корзине» - специально организованной папке, предназначенной для временного хранения уда­ ленных файлов.

2 Разумеется, могут быть и другие цели, например, связанные с необходимо­ стью удаления файлов с конфиденциальной информацией или с истечением уста­ новленных сроков хранения деловой документации.

**140**

список элементов таблицы FAT, представляющий последователь­ ность кластеров, занятых файлом.

1. **й шаг** - все элементы таблицы FAT, входящие в список номе­ ров кластеров, занятых удаляемым файлом, заполняются нулевы­ ми значениями - тем самым соответствующие кластеры рабочей области тома объявляются свободными для последующей записи в них новых файлов.
2. **й шаг** - регистрационная запись удаленного файла в роди­ тельском каталоге специальным образом *маркируется* - первый из восьми символов поля «имя файла» заменяется специальным символом с АSСП-кодом E5h1.

АSСП-код ESh определяет один из символов дополнительной ко­ довой таблицы, которые, согласно правилам MS DOS, нельзя было использовать для именования файлов. Наличие маркированных за­ писей позволяет программам просмотра каталогов игнорировать все записи об удаленных файлах, а программам восстановления удаленных файлов - наоборот, выбирать только те записи, которые начинаются с такого «специального» символа.

Заметим, что *файл физически не удаляется* из занятых им кла­ стеров, и запись об удаленном файле также *не удаляется* из ро­ дительского каталога, что делает доступной всю ее информацию, включая размер файла, время его создания, расширение и послед­ ние семь символов имени файла, а также номер начального класте­ ра. Все это сохраняет потенциальную возможность восстановления удаленного файла - по крайней мере, до тех пор, пока условно ос­ вобожденные этим файлом кластеры не будут заняты каким-либо другим файлом.

Выбор способа маркировки регистрационных записей удален­ ных файлов путем модификации имен файлов кажется труднообъ­ яснимым - почему бы не использовать для этой цели один из двух свободных битов байта атрибутов?

К тому же, такая маркировка создала проблему при переходе на новое соглашение об именовании файлов, разрешившее исполь­ зовать в именах символы национальных алфавитов. Так как АSСП­ код ESh определяет строчную букву алфавита, пришлось все строч­ ные символы имени файла преобразовывать в соответствующие прописные2, чтобы файлы с именами, начинающимися со ставшего

1 Причину, по которой для маркировки удаленных файлов был выбран символ с АSСП-кодом ЕSь, а не какой-то другой из 128 символов дополнительной кодовой таблицы, теперь уже вряд ли удастся определить достоверно, остается только пред­ полагать, что в команде разработчиков MS DOS были русскоязычные программи­ сты, не лишенные чувства юмора.

2 Алгоритм преобразования «длинных» Windows-имeн файлов в «короткие» DОS-имена обсуждался выше в п. 6.3.1 при рассмотрении структуры записей кор­ невого каталога.

**141**

разрешенным символа с кодом ЕSь, не получали статус удаленных файлов в момент их создания.

**Алгоритм создания подчиненного каталога**

Подчиненный каталог - это промежуточный узел иерархиче­ ской (древовидной) структуры данных, непосредственно связанный с единственным *родительским узлом* (корневым или другим под­ чиненным каталогом) и множеством *дочерних узлов* (подчиненных каталогов или файлов).

По своей структуре подчиненный каталог подобен корневому - он содержит множество 32-байтовых записей, используемых для регистрации дочерних объектов, но, в отличие от корневого, под­ чиненный каталог является файлом специального формата и хра­ нится, как и все другие файлы, в рабочей области тома, занимая в ней определенную последовательность кластеров. Количество кластеров, занятых подчиненным каталогом, динамически изменя­ ется и в любой момент времени должно быть минимально достаточ­ ным для регистрации всех его дочерних объектов.

Для создания подчиненного каталога используется DОS-команда MD (или MKDIR, *Make Directory),* единственным параметром ко­ торой является «путь», заканчивающийся именем создаваемого каталога: например, в результате выполнения команды ***MD docs\* my\_docs** будет создан каталог **my\_docs,** подчиненный своему роди­ тельскому каталогу ***docs*** (который, в свою очередь, подчинен ката­ логу, установленному текущим к моменту выполнения команды).

На первом шаге реализуется *алгоритм поиска об-ьекта* - роди­ тельского каталога ***docs,* и** в нем создается новая регистрационная запись для дочернего объекта ***ту\_docs*** (если, разумеется, в этом ка­ талоге еще нет дочернего объекта с таким же именем). В этой за­ писи заполняются все поля (в том числе - устанавливается атрибут *dir,* позволяющий отличить файлы типа «каталог» от файлов других типов), кроме адресного поля - пока еще неизвестного *номера на­ чального кластера.*

Следующий шаг алгоритма - *поиск свободных кластеров,* не­ обходимых для хранения создаваемого подкаталога **my\_docs.** В мо­ мент создания каталога ему выделяется единственный кластер, но­ мер этого кластера записывается в адресном поле регистрационной записи родительского каталога, а в соответствующем элементе та­ блицы FAT записывается код FFFь (EOF).

На завершающем этапе алгоритма оформляется содержимое кла­ стера, выделенного каталогу.

Во-первых, кластер «обнуляется» - в каждый из его байтов за­ писывается ноль, в результате чего физически удаляются все дан­ ные, которые ранее были записаны в этом кластере (это могли быть данные удаленных файлов, ранее занимавших этот кластер, или

**142**

данные программы форматирования тома, тестирующей каждый сектор путем записи/чтения специальных кодовых последователь­ ностей).

Во-вторых, в кластере создаются две служебные записи для фик­ тивных объектов типа «каталог» (их атрибут *dir* установлен веди­ ницу). Эти записи подобны другим записям каталогов, за исключе­ нием имен объектов:

* первый фиктивный объект получает имя«.» (точка), а в ка­ честве адресной ссылки этого объекта указывается номер первого (и пока единственного) кластера, выделенного создаваемому ката­ логу («ссылка на себя»);
* имя второго фиктивного объекта - «..» (две точки, не путать с двоеточием), а в поле адресной ссылки этого объекта указывается номер начального кластера родительского каталога («ссылка на ро­ дителя»); если родительским является корневой каталог, то в поле ссылки указывается ноль (что допустимо, так как число «ноль» не используется для нумерации кластеров).

Такие «точечные» имена не могут быть именами реальных объ­ ектов файловой системы, они выполняют роль маркеров служебных записей, используемых для оптимизации алгоритмов поиска и про­ смотра подчиненных каталогов.

Казалось бы, зачем хранить ссылку на начальный кластер ката­ лога в самом этом начальном кластере? Причина - во фрагментар­ ности расположения файлов, причем вероятность фрагментарного расположения подчиненных каталогов существенно выше, чем обычных файлов.

Покажем это на простом примере: пусть на томе достаточно много непрерывных фрагментов свободных кластеров, тогда при создании каталога ему будет выделен один кластер в начале перво­ го из таких фрагментов. Далее в этот каталог массово копируются файлы.

Если кластер имеет размер 1024 байта (в нашем примере), а каждая регистрационная запись занимает 32 байта, одного кла­ стера каталога хватит для регистрации только первых 32 дочерних объектов (точнее, только первых 30, так как в каталоге уже созданы две служебные записи, а фактически - еще меньше, если в каталог копируются объекты с «длинными» Windows-имeнaми, для хране­ ния каждого из которых может потребоваться от одной до восьми дополнительных 32-байтовых записей).

Для продолжения копирования файлов потребуется расширение каталога, и будет запущен алгоритм поиска еще одного свободно­ го кластера, в котором можно будет зарегистрировать еще 32 до­ черних объекта, и т. д., до тех пор пока не будет исчерпан список копируемых файлов. Какова вероятность того, что при очеред­ ном расширении каталога ему будет выделен кластер с номером,

**143**

на единицу большим номера предыдущего кластера? Очевидно, что близкая к нулю, так как следующие (после начального) кластеры, бывшие свободными в момент создания каталога, к моменту его расширения с высокой вероятностью уже будут заняты дочерними объектами этого каталога.

А теперь представим себе реализацию алгоритма поиска до­ чернего объекта (см. выше алгоритм поиска файла) в таком силь­ но фрагментированном каталоге. На предыдущем шаге алгоритма по ссылке из родительского каталога был найден и загружен в бу­ фер ОЗУ начальный кластер этого каталога, после чего запускает­ ся процедура его сканирования с целью поиска дочернего объекта. Если в начальном кластере каталога искомого объекта нет, потре­ буется найти следующий кластер каталога, затем, возможно, еще один и т. д. по всей последовательности номеров кластеров, задан­ ной в таблице FAT для этого каталога.

Проблема в том, что ссылка на начало этой последовательно­ сти хранится в родительском каталоге, и сохранение ее дубликата в виде «ссылки на себя» в начальном кластере каталога (который га­ рантированно загружен в буфер ОЗУ) позволяет исключить опера­ цию чтения родительского каталога при поиске дочерних объектов. Если «ссылка на себя» позволяет оптимизировать алгоритм по­ иска дочерних объектов (спуск по дереву каталогов), то «ссылка на родителя» во второй служебной записи подчиненного каталога позволяет оптимизировать алгоритм поиска родительского катало­ га (подъем на один уровень по дереву каталогов), исключая необ­ ходимость доступа к родительскому каталогу своего родительского

каталога для поиска соответствующей адресной ссылки.

Например, DОS-команда CD *(Change Directory)* применяется для открытия каталога, путь к которому задается единственным па­ раметром этой команды. Алгоритм поиска элементов заданного

«пути» будет последовательно отыскивать в каталогах имена дочер­ них объектов и соответствующие адресные ссылки. В команде вида

CD ..

путь включает имя фиктивного объекта «.. », регистрационная за­ пись которого гарантированно присутствует в текущем каталоге и содержит ссылку на его родительский каталог.

Следующие три примера также иллюстрируют полезность хране­ ния ссылки на родителя непосредственно в текущем каталоге.

В результате выполнения DОS-команды

MD ..\my-brother-dir

будет создан каталог, подчиненный родителю текущего каталога (т. е. каталог одного уровня с текущим). Команда

СОРУ ..\old\_file new\_file

**144**

создаст в текущем каталоге копию файла родительского каталога, а команда

СОРУ old\_file ..\new\_file

скопирует файл из текущего каталога в родительский.

Алгоритмы копирования и перемещения файлов

Операция копирования файла реализуется последовательным выполнением рассмотренных выше алгоритмов поиска файла и по­ иска свободного пространства для записи файла. В результате созда­ ется копия исходного файла в том же или другом каталоге.

Операция перемещения файла реализуется в двух вариантах, в зависимости от того, куда перемещается файл: на другой том или в другой каталог того же тома. В первом случае файл сначала копируется, а затем удаляется в соответствии с описанными выше алгоритмами. Второй вариант не предусматривает создания копии исходного файла - файл остается в тех же кластерах, но его реги­ страционная запись перемещается в другой каталог.

Алгоритмы восстановления удаленных файлов

Анализ рассмотренных выше алгоритмов записи и удаления фай­ лов показывает, что в ряде случаев сохраняется возможность вос­ становления удаленного файла (до тех пор, пока на его место физи­ чески не записан другой файл).

Для восстановления логически удаленного файла достаточно *из­ менить в регистрационной записи каталога первый символ его име­ ни на любой из допустимых символов,* что сделает «видимой» запись об удаленном файле, и *восстановить в таблице FAT список номеров кластеров, занятых этим файлом,* что даст возможность доступа к его содержимому.

Проблема в том, что все элементы этого списка таблицы FAT при удалении файла получили нулевые значения и стали формально не­ отличимыми от множества других элементов FAT, представляющих свободные кластеры (в том числе и те, которые ранее бьти заняты впоследствии удаленными файлами).

В этих тяжелых для программиста условиях определенный опти­ мизм вселяет тот факт, что процедура удаления файла только мар­ кирует его регистрационную запись, не удаляя ее из соответству­ ющего каталога, что делает доступной информацию об удаленном файле:

* если регистрационная запись об удаленном файле присут­ ствует в каталоге (в чем можно убедиться по последним семи сим­ волам его имени, так как имена всех удаленных файлов начинаются одинаково - с символа с АSСП-кодом ЕSь), значит можно продол­ жить попытку его восстановления, так как в этом случае начальный

**145**

кластер удаленного файла еще не занят другими файлами, записан­ ными на диск после того, как этот файл бьm удален;

* сохранена адресная ссьmка (номер начального кластера), т. е. известно, откуда следует начинать поиск номеров кластеров, заня­ тых файлом;
* известен размер удаленного файла, что позволяет вычислить количество занятых им кластеров;
* известны дата и время создания удаленного файла, что может быть полезным в случае наличия в томе множества других удален­ ных файлов (время создания которых также известно), в том числе и не требующих восстановления;
* известен алгоритм поиска свободного дискового простран­ ства для записи файла, реализованный соответствующей системной функцией файловой системы.

Очевидно, что главная проблема восстановления удаленных файлов связана с фрагментарностью их расположения, поэтому для повышения надежности работы процедур восстановления рекомен­ дуется периодически дефрагментировать диск с помощью специ­ альных программных утилит, которые перераспределяют дисковое пространство, закрепляя за файлами последовательности из непре­ рывно расположенных кластеров. Процедура дефрагментации дис­ ка предполагает создание временных копий файлов, требует нали­ чия на диске свободного пространства и является весьма затратной по времени.

* + 1. **Недостатки FАТ-систем**

FАТ-системы успешно применялись многие годы в составе раз­ личных ОС, прежде всего - в DOS и Windows. Первоначально они создавались для НГМД - накопителей на гибких магнитных дис­ ках (первыми такими устройствами бьmи В-дюймовые односто­ ронние гибкие диски информационной емкостью 160 Кбайт), и од­ ним из основных требований, предъявляемых к файловой системе в те годы, бьmа экономичность хранения служебной информации.

Следует отметить, что разработчики системы FAT-12 успешно справились с реализацией этого требования, и основные концеп­ ции, заложенные при ее создании, оказались работоспособными и в применении к жестким дискам существенно большей (по срав­ нению с НГМД) емкости. Правда, это потребовало увеличения раз­ рядности таблицы FAT вначале до 16, а затем и до 32 битов, что неизбежно привело к увеличению объема хранимых на диске си­ стемных данных.

Со временем состав требований, предъявляемых к файловым си­ стемам, существенно расширился, в новых условиях FАТ-системы перестали справляться с задачами управления файлами, а отдель-

**146**

ные технологические и алгоритмические решения этих систем по­ теряли свою актуальность. Тем не менее, и сегодня FАТ-системы вполне успешно используются на персональных компьютерах - в основном для временного хранения информации на сменных твердотельных накопителях небольшой емкости.

Основная критика в адрес FАТ-систем нацелена на те их недо­ статки, которые связаны с организацией хранения и доступа к фай­ лам.

1. *Неэффективное использование дисковой памяти.* При форма­ тировании дисков большой емкости приходится решать нетриви­ альную задачу выбора размера кластера: большие размеры класте­ ров уменьшают их количество, соответственно уменьшается размер таблицы FAT и увеличивается рабочая область тома, но при этом снижается эффективность хранения небольших файлов; уменьше­ ние размеров кластеров позитивно сказывается на эффективности использования рабочей области тома, но приводит к уменьшению ее общего размера.
2. *Низкая эффективность таблицы FАТ.* Таблица FAT - это ос­ новная структура данных, используемая алгоритмами поиска фай­ лов и алгоритмами поиска свободного пространства. Казавшийся когда-то эффективным способ хранения такой разнородной инфор­ мации в единой структуре данных фиксированного размера с уве­ личением разрядности FAT стал проявлять свои негативные сторо­ ны. Хранение в элементах FAT как адресной информации (номера

«следующих» кластеров файлов), так и информации о свободных кластерах (число «О») привело к тому, что, например, в FAT-32 для хранения числа «О» расходуется *8 байтов* вместо *одного бита,* кото­ рого вполне хватило бы для того, чтобы отличить занятый кластер от свободного. В результате размер таблицы FAT возрастает и оста­ ется максимальным независимо от количества занятых кластеров рабочей области.

1. *Существенная фрагментация файлов.* При длительной экс­ плуатации дисковых накопителей операции удаления файлов (так же, как и операции редактирования, уменьшающие их размеры) объективно приводят к фрагментации свободного пространства, а используемые алгоритмы поиска свободного пространства при копировании (или увеличении размеров) файлов способствуют их фрагментации. В результате снижается быстродействие системы и увеличивается износ оборудования.
2. *Ограничение длины имени файла.* Ограничение длины имени файла восемью байтами при его хранении в соответствующем поле записи каталога приводит к необходимости дублирования записей и снижает эффективность хранения подчиненных каталогов, спо­ собствуя увеличению их фрагментарности.

**147**

1. *Низкая надежность хранения данных.* Надежность работы любой технической системы оценивается по двум основным пара­ метрам: длительности работы без сбоев (так называемая «наработ­ ка на отказ») и времени восстановления системы после сбоя.

Разработчики FАТ-систем предусмотрели различные средства повышения надежности работы файловой системы, однако эффек­ тивность этих средств явно не соответствует современным требова­ **ниям.**

В ситуациях с «жестким сбоем», связанных с разрушением носи­ теля информации (например, магнитного слоя рабочей поверхно­ сти диска):

* + фрагментарность файлов, приводящая к преждевременному износу оборудования, косвенно уменьшает время наработки на от­ каз;
  + дублирование таблицы FAT, основного «адресного справоч­ ника» файловой системы, снижает время восстановления системы, так как при утере основной FAT вся адресная информация может быть восстановлена по ее зеркальной копии; однако следует учесть, что обе FAT размещены в системной области тома, занимая в ней соседние секторы, поэтому вероятность того, что при разрушении основной FAT ее зеркальная копия останется доступной, крайне низка.

В ситуациях с «мягким сбоем», не связанных с разрушением но­ сителя информации, например, при ошибочном удалении файла:

* + FАТ-системы позволяют блокировать попытки ошибочного удаления особо ценных файлов путем установки в «1» атрибута *Read Опlу* в регистрационных записях этих файлов в соответствующих каталогах, однако такой прием одновременно заблокирует возмож­ ность выполнения ряда других файловых операций;
  + FАТ-системы сохраняют возможность восстановления удален­ ных файлов, но они ограничиваются фрагментарностью расположе­ ния файлов и по времени восстановления не конкурируют с «корзи­ ной» Windows.

1. *Отсутствие средств разграничения доступа к файлам.* В то время, когда разрабатывались FАТ-системы, проблема инфор­ мационной безопасности не стояла так остро, как сегодня, и поэто­ му средства защиты от несанкционированного доступа к файлам проработаны в этих системах весьма слабо. Защита от несанкцио­ нированного копирования или модификации файла обеспечивается установкой в «1» его атрибута *Read Only.* Для защиты файла с кон­ фиденциальной информацией от несанкционированного просмотра можно установить в «1» его атрибут *Hidden,* что придаст ему статус

«скрытого файла», однако, во-первых, это не разграничение, а огра­ ничение доступа, и, во-вторых такие ограничения легко обходятся каждым, кто знаком с DОS-командой *attrib.*

**148**

##### Файловые системы NTFS

По сравнению с FАТ-системами файловая система NTFS *(New Technology File System)* обладает лучшими показателями по произ­ водительности и надежности, а также обеспечивает полнофунк­ циональную защиту данных, позволяя гибко управлять доступом к данным и разграничивать доступ пользователей как к отдельным файлам, так и к каталогам.

NTFS поддерживает объектно-ориентированную файловую мо­ дель, рассматривая все файлы как объекты, которые имеют опреде­ ляемые пользователем и системой атрибуты-свойства. Минималь­ ный набор стандартных атрибутов файлов приведен в табл. 6.7. Форматы атрибутов и способы их хранения рассматриваются в п. 6.4.5.

*Таблицаб.7*

**Минимаnьный набор стандартных атрибутов файnов**

|  |  |  |
| --- | --- | --- |
| **Обозна- чение** | **Имя**  **атрибута** | **Назначение атрибута** |
| *н* | Header | Заголовок МFТ-записи (содержит, в частности, ее номер) |
| *SI* | Standard Information | Стандартная информация |
| *FN* | File Name | Имя файла |
| *Data* | Data | Данные файла |
| *SD* | Security Descriptor | Дескриптор безопасности |

Понятие *атрибута файла,* которое в FАТ-системах ограничи­ валось фиксированным набором из шести бинарно-кодируемых свойств, в NTFS существенно расширено: в частности, наряду со стандартными атрибутами появилась возможность создания пользовательских атрибутов произвольного назначения.

* + 1. **Структура тома NТFS**

NTFS, так же как и FAT, использует понятие кластера - мини­ мального блока, выделяемого файлам при их записи на том. В отли­ чие от FAT, NTFS делит на кластеры *все пространство тома,* в том числе и его системную область, называемую в NTFS *МFТ-зоной.* При этом поддерживаются кластеры практически любых размеров - от 512 байт до 64 Кбайт, стандартом считается размер 4 Кбайт.

Том NTFS условно делится на две зоны: примерно 12 о/о класте­ ров, расположенных в начале тома, отводятся под так называемую *МFТ-зону,* в которой располагается самый главный служебный файл

**149**

MFT *(Master File ТаЫе-* главная таблица файлов), а остальные при­ мерно 88 о/о кластеров образуют рабочую зону, используемую для хранения как служебных, так и пользовательских файлов. Запись в МFТ-зону других файлов, кроме самого MFT, запрещена - это сде­ лано для того, чтобы исключить фрагментацию MFT при его рас­ ширении.

При этом не все так просто с МFТ-зоной: свободными для записи считаются все свободные кластеры - в том числе и кластеры МFТ­ зоны, не занятые МFТ-файлом. Если объем рабочей зоны становит­ ся недостаточным для записи файлов, МFТ-зона сокращается (в два раза), предоставляя рабочей зоне дополнительные кластеры, а при освобождении рабочей зоны МFТ-зона может снова расшириться. При этом не исключена ситуация, когда в этой зоне остались и обыч­ ные файлы - файл MFT в этом случае может фрагментироваться.

* + 1. **Мета-файлы NТFS**

Главный «лозунг» файловой модели NТFS: *«Файлы, и ничего, кро­ ме Файлов».* **Любой** элемент файловой системы представляет собой файл, в том числе и элементы, содержащие служебную информа­ цию: корневой каталог, загрузочная запись, журнал транзакций, список прав пользователей и другие, так называемые *мета-файлы.* Состав метафайлов (табл. 6.8), их структура и назначение фик­ сированы, их имена начинаются с символа $, и все они в обязатель­ ном порядке регистрируются в корневом каталоге, который сам

также является метафайлом.

*Таблицаб.8*

**Метафайлы NTFS**

|  |  |  |
| --- | --- | --- |
| **Иденти- фикатор файла** | **Имя мета- файла** | **Назначение файла** |
| о | $MFT | **СамМFТ** |
| 1 | $MFТmirr | Зеркальная копия первых 16 записей MFT |
| 2 | $LogFile | Журнал транзакций, который используется для восстановления файловой системы после сбоев |
| 3 | $Volume | Метка тома и прочая служебная информация |
| 4 | $AttrDef | Список стандартных атрибутов файлов |
| 5 | $ | Корневой каталог |
| 6 | $Bitmap | Битовая карта занятости кластеров:  О - кластер свободен, 1 - кластер занят |
| 7 | $Boot | Загрузочная запись. Сдержит стандартный набор параметров BIOS, а также начальные номера кла- стеров основного МFТ и его зеркальной копии |

**150**

|  |  |  |
| --- | --- | --- |
| **Иденти- фикатор файла** | **Имя мета- файла** | **Назначение файла** *Окончание табл.* 6. |
| 8 | $BadClus | Список сбойных кластеров |
| 9 | $Quota | Описание прав доступа пользователей |
| 10 | $Upcase | Таблица соответствия Unicode-кoдoв заглавных и строчных букв в именах файлов |
| 11-15 |  | Резерв |

Основной метафайл в системе NTFS - это MFT, централизован­ ный каталог всех файлов тома (в некотором смысле MFT выполняет ту же роль, что и корневой каталог вместе с таблицей расположения файлов в FАТ-системах).

MFT состоит из записей фиксированного размера (обычно 2 Кбайт), каждая из которых содержит информацию о каком-либо одном файле. Каждый файл (в том числе и сам MFT) представлен в МFТ-файле, по крайней мере, одной записью, позиционный номер которой является уникальным *идентификатором файла.* Зная этот идентификатор, можно восстановить по ссылкам список номеров всех остальных МFТ-записей этого файла и далее, также по ссыл­ кам - список номеров всех кластеров, занятых этим файлом.

Первые 16 записей MFT содержат данные *метафайлов,* причем самая первая (нулевая) запись представляет сам MFT. Учитывая ценность информации, хранимой в метафайлах, зеркальная копия первых 16 записей MFT хранится в отдельном метафайле $MFTmirr, расположенном в рабочей зоне - примерно посередине дискового пространства тома (как видим, разработчиками NTFS учтен недо­ статок FАТ-систем, в которых зеркальная копия таблицы FAT хра­ нится в системной области, практически рядом с основной табли­ цей, что снижает надежность дисковой системы компьютера при жестких сбоях).

* + 1. Схемы хранения файлов

Основные претензии к FАТ-системам высказывались в связи с низкой эффективностью использования дискового пространства этими системами при хранении файлов радикально различных раз­ меров, а также в связи с потерями производительности операций доступа к файлам по причине их фрагментации. В NTFS система хранения файлов оптимизирована по этим критериям, что позво­ лило эффективно использовать кластеры небольшого размера без увеличения затрат на хранение адресной информации.

Во-первых, все файлы в NTFS классифицируются по размерным категориям, и для каждой категории используется своя система хра-

**151**

пения, оmимальная как по критерию фрагментарности файлов, так и по критерию производительности выполнения операций их поиска. Во-вторых, система хранения адресной информации о фрагмен­ тированных файлах (а фрагментация файлов больших размеров объективно неизбежна) сделана существенно более компактной по сравнению с таблицей FAT, в которой приходилось хранить упо­ рядоченные списки номеров всех (кроме первого) кластеров, заня­

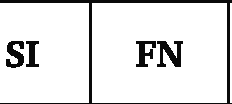
тых файлом.

В-третьих, для поиска файлов в NTFS применена технология индексации, широко используемая в системах управления базами данных. Каталоги, которые в FАТ-системах были представлены ли­ нейными списками имен файлов с адресными ссылками на началь­ ные кластеры этих файлов, получили в NTFS статус индексов - ие­ рархических структур данных, позволивших отказаться при поиске от операций полного перебора элементов списков в пользу опера­ ций «спуска по дереву», существенно ускоряющих поиск файлов по значениям их атрибутов.

**Схема хранения *небольших (small)* файлов.** Если файл имеет небольшой размер (как правило, не более 1,5 Кбайт), он может по­ лучить статус *резидентного файла* и будет целиком располагаться

«внутри» атрибута *Data* (рис. 6.2) своей единственной МFТ-записи. При этом возможны ситуации, когда файл малого размера не по­ лучает статус резидентного из-за того, что суммарный размер всех других его атрибутов (состав и размеры которых могут быть переменными), не позволяет ему разместиться внутри одной МFТ­ записи длиной 2 Кбайт.

Резидентные файлы не занимают кластеров в рабочей зоне тома, и доступ к ним осуществляется наиболее эффективно.



**н**

**Data**

**SD**

*Рис. 6.2.* **Схема хранения небопьших *(small)* файпов**

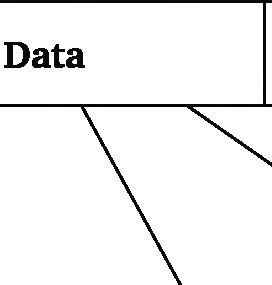
**Схема хранения *больших (large)* файлов.** Если файл не поме­ щается внутри своего атрибута *Data,* он становится *нерезидентным* и располагается вне МFТ-зоны, занимая в рабочей зоне один или более кластеров. При этом статус *резидентного* получает атрибут *Data* этого файла, который будет содержать уже не данные файла, а адресные ссьтки на фрагменты кластеров рабочей зоны, занятых файлом (рис. 6.3).

Каждая такая ссылка определяет один фрагмент непрерывно расположенных кластеров и содержит три параметра: *LN* - поряд­ ковый номер начального кластера фрагмента, *VN* - виртуальный номер начального кластера фрагмента (смещение начального кла-

**152**

стера фрагмента относительно начала файла) и *CL* - количество кластеров во фрагменте.

Таким образом, содержимое атрибута *Data* начальной (и един­ ственной) МFТ-записи нерезидентного файла позволяет определить упорядоченный список номеров кластеров, занятых этим файлом.



Фрагмент данных

**SD**

**н**

**SI**

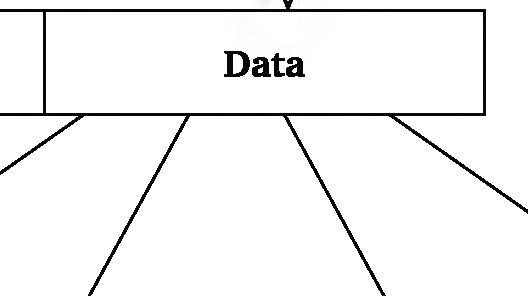
**FN**

Фрагмент данных

Фрагмент данных

Фрагмент данных

*Рис. 6.3.* **Схема хранения бопьwих (/arge) файпов**



**Схема хранения *очень больших (huge)* файлов.** Если файл на­ столько сильно фрагментирован, что его атрибут *Data* (со ссылка­ ми на фрагменты кластеров, занятых этим файлом) не помещается в начальной МFТ-записи файла, то *нерезидентным* становится сам этот атрибут, и для его хранения резервируется вторая МFТ-запись (рис. 6.4), ссылка на которую помещается в начальную МFТ-запись файла в качестве *внешнего атрибута ЕА (External Attribute).* При этом нерезидентный атрибут *Data,* помещенный во вторую МFТ­ запись файла, содержит ссылки на фрагменты кластеров, занятых файлом, точно так же, как и в случае с Lаrgе-файлами.

**н SI FN ЕА SD**

н

Фрагмент данных

Фрагмент данных

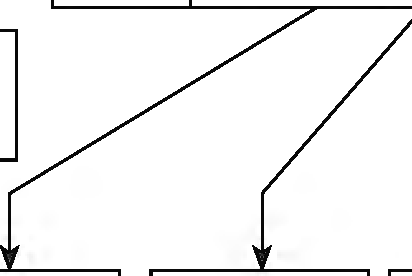
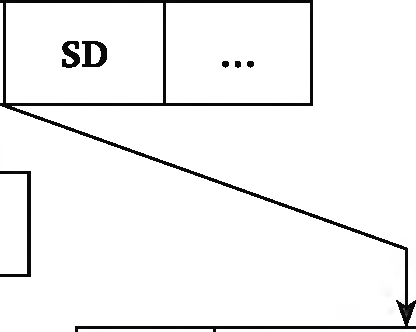
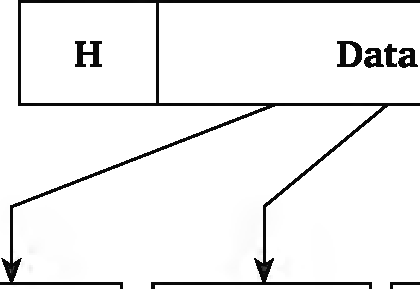
Фрагмент данных

Фрагмент данных

*Рис. 6.4.* **Схема хранения очень бопьwих *(huge)* файпов**

**153**

**Схема хранения *сверхбольших (extremely huge)* файлов.** Если нерезидентный атрибут *Data* не помещается и во вторую МFТ­ запись файла, то внешний атрибут *ЕА,* хранимый в начальной МFТ­ записи файла, может содержать несколько упорядоченных ссылок на дополнительные МFТ-записи этого файла, в каждой из которых хранится свой нерезидентный атрибут *Data* (рис. 6.5) с множеством ссылок на фрагменты кластеров, занятых файлом.



**н**

**SI**

**FN ЕА**

**н**

**Data**

Фрагмент Фрагмент Фрагмент данных данных данных

Фрагмент данных

Фрагмент Фрагмент данных данных

*Рис. 6.5.* **Схема хранения сверхбоnьwих *(extremely huge)* файnов**

Разумеется, в такой ситуации увеличивается размер внешнего атрибута *ЕА,* и когда ему перестанет хватать места в начальной МFТ-записи файла, он также станет нерезидентным.

Заметим, что и многие другие атрибуты файла могут хранить­ ся в нерезидентной форме, что практически снимает ограничения на их размеры.

Как уже отмечалось, системы FAT и NTFS по-разному тракту­ ют понятие *каталога.* С позиций FАТ-систем МFТ-файл выполняет роль *дерева каталогов,* так как содержит регистрационные записи обо всех файлах со ссьmками на номера занятых ими кластеров или на другие МFТ-записи (выполняющие роль подчиненных ка­ талогов), в которых размещены такие ссылки. Процедура последо­ вательного сканирования МFТ-файла позволит найти любой файл по значению его атрибута *FN (File Name)* и по ссылкам определить расположение этого файла в рабочей зоне тома.

* + 1. Схемы хранения каталогов

Однако есть проблема, практически исключающая возможность прямого использования МFТ-файла в качестве такого «суперката-

**154**

лога», и эта проблема - в производительности процедуры полного сканирования. Как известно, файловая система NTFS создавалась для обслуживания накопителей очень большой емкости, и МFТ­ файл может достигать гигантских размеров (порядка 8 Тбайт при длине МFТ-записи в 2 Кбайт и 4-байтовой длине поля номера такой записи), что потребует множества обращений к диску для его пол­ ного сканирования.

Для ускорения поиска файлов NTFS использует файлы специаль­ ной структуры, называемые *каталогами* или *индексными списками (Index List).* В отличие от каталогов FАТ-систем, NТFS-каталоги со­ держат только адресную информацию в форме ссылок на дочерние объекты - подчиненные им каталоги, файлы, или фрагменты кла­ стеров тома.

Один такой индексный список - корневой каталог - является метафайлом, имеет фиксированное имя$ (очевидно, разработчи­ ки NTFS решили подчеркнуть таким именем большую ценность информации, содержащейся в корневом каталоге) и хранится в од­ ной из начальных записей МFТ-файла. Остальные (подчиненные) каталоги хранятся как обычные файлы, и могут создаваться и име­ новаться пользователями стандартными средствами, например, ко­ мандой MD в случае использования интерфейса командной строки. Каждый каталог имеет одну запись в МFТ-файле, которая содер­ жит стандартный набор атрибутов (Н, *FN, SI, SD)* и дополнитель­ но - специальный атрибут *Ш (Root Index- корневой индекс),* ко­ торый, собственно, и является *индексным списком,* каждый элемент которого содержит пару: имя дочернего объекта (файла или подчи­ ненного каталога) и адресную ссьmку, в качестве которой исполь­ зуется номер начальной МFТ-записи этого объекта. Завершается индексный список строкой«####», используемой в качестве кода

конца списка.

Длина списка *Ш* каталога зависит от количества зарегистриро­ ванных в этом каталоге дочерних объектов и фактически опреде­ ляет и размер всего каталога. Каталоги, как и файлы NTFS, клас­ сифицируются по их размерам, и для каждой размерной категории каталогов применяется соответствующая схема их хранения, подоб­ но тому, как это сделано для файлов.

**Схема хранения *небольших (small)* каталогов.**

Если количество дочерних объектов каталога невелико, то его атрибут *Ш* получает статус *резидентного* и полностью размещается в начальной МFТ-записи каталога (рис. 6.6).

***Н* SI *FN*** Rl:<aaa;232> <abcd;48>

<xyz;1234> ... <абвгд; 21> ####

***SD***

*Рис.* 6.6. **Схема хранения небо11ьших ката11оrов**

**155**

Элементы списка *RI* упорядочены по именам объектов в алфа­ витном порядке, что позволяет применить к этому списку *алгоритм бинарного поиска* для определения ссылки на файл по его имени. Бинарный поиск (известный в математике как *метод дихотомии* или *метод деления пополам)* широко применяется в поисковых ал­ горитмах, так как позволяет отказаться от полного перебора всех *п* элементов списка и сокращает время поиска с *k* • *п* до *k* • 1og2n.

Особенно заметным выигрыш в скорости поиска становится при больших длинах списков: если при длине списка *п* =16 (24) бинарный поиск дает только 4-кратное преимущество по сравнению с методом полного сканирования, то при *п* = 1024 (210) - уже 100-кратное1.

Справедливости ради следует отметить, что поддержка упорядо­ ченных списков требует дополнительных временных затрат: напри­ мер, при каждом включении в каталог нового дочернего объекта потребуется повторно выполнять процедуру сортировки элементов списка *RI* с учетом алфавитной позиции имени добавленного дочер­ него объекта.

**Схема хранения *больших (large)* каталогов.** С увеличением количества дочерних объектов каталога соответственно увеличи­ вается и длина его атрибута *RI,* и в определенный момент он пе­ рестанет помещаться в начальную МFТ-запись каталога. В такой ситуации атрибут *RI* перестает быть *линейным списком* (как это было у небольшого каталога) и получает *иерархическую (древовид­ ную)* структуру, в которой одни элементы списка подчинены другим элементам. При этом часть списка *RI* («корень дерева») останется в начальной МFТ-записи каталога, а для хранения других его ча­ стей («листов дерева», подчиненных корню) будут зарезервированы другие МFТ-записи, как это показано на рис. 6.7.

Для преобразования длинного *линейного списка RI* в *дерево:*

* этот список делится на несколько групп элементов с сохране­ нием алфавитного порядка следования элементов;
* каждая такая группа помещается в дополнительную МFТ­ запись каталога и становится отдельным *листовым* списком, под­ чиненным корневому списку;
* в *корневом* списке остается по одному представителю каждой группы (например, только крайние левые, содержащие «минималь­ ные» по алфавиту имена объектов), а в качестве ссылок использу­ ются номера дополнительных МFТ-записей, в которых хранятся со­ ответствующие листовые группы.

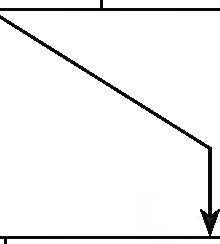
Поиск подчиненного каталогу объекта по его имени всегда начи­ нается с начальной МFТ-записи каталога, в которой хранится кор­ невая часть списка *RI.*

1 Изучение линейных и иерархических структур данных, а также алгоритмов сортировки и поиска данных, реализуемых на таких структурах, выходит за рамки нашего курса.

**156**

Бинарным поиском в этом списке определяется ближайшее (по алфавиту) имя группового элемента и далее осуществляется переход по ссылке к соответствующему групповому списку листо­ вого уровня. Далее производится бинарный поиск имени искомого объекта в групповом списке, и если объект найден, осуществляется переход по соответствующей ссылке на его начальную МFТ-запись. Возможны и другие иерархические схемы хранения каталогов, более сложные по сравнению с рассмотренной двухуровневой схемой. Если количество ссылок в групповом списке превыша­ ет допустимый предел, возможно его нерезидентное (в том числе и фрагментарное) хранение в кластерах рабочей зоны тома. При этом в МFТ-запись добавляется новый атрибут *IA (Index Allocation),* который содержит указатели на фрагменты кластеров *(LN, VN, CL)* подобно тому, как это сделано в одной из схем хранения больших

файлов.



**н**

**SI FN Корневая часть RI SD**

**Н**

**Листовая часть RI**

**Н**

**Листовая часть RI**

**Н Листовая часть RI**

*Рис. 6.7.* **Схема хранения бопьwих катапоrов**

* + 1. Атрибуты файлов и каталогов

Любой файл NTFS трактуется как набор его атрибутов, в число которых входит и атрибут *DATA,* который может содержать соб­ ственно данные файла. Каждый атрибут файла NTFS состоит из по­ лей: *тип (код)* атрибута, *длина* атрибута, *значение* атрибута и, воз­ можно, *имя* атрибута.

Существует два способа хранения атрибутов: резидентное хра­ нение в записях таблицы MFT и нерезидентное хранение - в кла­ стерах рабочей зоны тома. Сортировка файлов (в каталогах) может осуществляться только по их резидентным атрибутам.

Стандартные атрибуты имеют фиксированные коды, имена и форматы хранения, для пользовательских атрибутов все эти па­ раметры не стандартизованы. Обязательный (минимальный) набор стандартных атрибутов приведен в табл. 6.7, их полный перечень - в табл. 6.9.

**157**

*Таблицаб.9*

**Перечень стандартных атрибутов NTFS-фaйnoв**

|  |  |
| --- | --- |
| **Имя атрибута** | **Назначение и использование атрибута** |
| *Attribute Li.st* | Список атрибутов, допустимых для данного файла |
| *FileName* | Длинное имя файла, а также номер начальной МFТ-записи родительского каталога; если файл содержится в несколь- ких каталогах (что допустимо в NТFS), то у него будет несколько атрибутов типа *File Nате;* этот атрибут всегда должен быть резидентным. |
| *MS-DOSName* | Короткое DОS-имя файла в формате 8.3 |
| *Version* | Номер последней версии файла |
| *Security Descriptor* | Информация о защите файла: список прав доступа и поле аудита, которое определяет, какого рода операции над этим файлом нужно регистрировать в журнале транзакций |
| *Volume Version* | Версия тома |
| *VolumeName* | Метка тома |
| *Volume Information* | Номер версии NТFS |
| *Data* | Данные резидентного файла (или ссылки на фрагменты кластеров) |
| *MFГbltmap* | Карта использования секторов на томе |
| *Root Index* | Корневая часть индексного списка |
| *Index Allocation* | Указатели на нерезидентные части индексного списка |
| *External Attribute* | Номер первого кластера **и** количество кластеров нерези- дентного атрибута |
| *Standard Information* | Вся остальная стандартная информация о файле (напри- мер, время создания файла и время последнего обновле- ния файла) |

###### Контрольные вопросы

1. В каком отношении находятся понятия «сектор», «дорожка» и «ци­ линдр», используемые для описания технических характеристик дисковых запоминающих устройств ?
2. Какие функции выполняет контроллер дискового устройства?
3. Каково назначение главной загрузочной записи (Master Boot Record, MBR)? Как используется MBR в процессе начальной загрузки ОС?
4. Каковы структура и назначение таблицы разделов (Partition ТаЫе)? Как используется Partition ТаЫе в процессе начальной загрузки ОС и при выполнении файловых операций?
5. Как трактуется понятие «кластер» в файловых системах, и в каком соотношении находятся понятия «сектор» и «кластер»?

**158**

1. Какие из внутренних и внешних DОS-команд реализуют типовые файловые операции?
2. Какие структуры данных формируются в системной области FАТ-тома и как они используются в процессе выполнения файловых операций?
3. Как используется информация загрузочного сектора (Boot Record) FАТ-тома в процессе начальной загрузки ОС?
4. Какие из атрибутов могут использоваться для обеспечения надежного хранения файлов?
5. Какие из атрибутов могут использоваться для ограничения доступа к файлам?
6. Какой DОS-командой можно просматривать/изменять значения атрибутов файла?
7. Где и как хранятся метки FАТ-тома?
8. Может ли FАТ-том иметь несколько меток или не иметь метки?
9. Какими DОS-командами можно просматривать/изменять значения метки тома?
10. В какой области тома хранится FАТ-таблица?
11. Какую информацию содержат элементы FАТ-таблицы?
12. От чего (и как) зависит размер FАТ-таблицы?
13. Какие Какие из стандартных атрибутов NТFS-файлов являются обя­ зательными?
14. Какую роль в файловой системе NTFS выполняет МFТ-файл? Как ограничен размер этого файла?
15. Какие NТFS-файлы называют резидентными? Сколько кластеров занимает резидентный файл в рабочей зоне NTFS-тoмa? Как организован доступ к резидентным файлам?
16. Как организовано хранение больших по размеру и сильно фрагмен­ тированных NТFS-файлов?

## Тема7 ОБМЕН ДАННЫМИ

**С ПЕРИФЕРИЙНЫМИ УСТРОЙСТВАМИ**

###### Система обработки прерываний

Понятием «прерывание» обозначают механизм, обеспечиваю­ щий оперативную реакцию центрального процессора на опреде­ ленное событие, требующее прерывания выполнения исполняемой процессором программы и перехода к выполнению другой специ­ альной программы, называемой *обработчиком прерывания.* После завершения процесса обработки прерывания автоматически проис­ ходит возврат к выполнению прерванной программы.

Система обработки прерываний способна обрабатывать до 256 раз­ личных событий, все они пронумерованы в диапазоне от INT ООь до INT FFь, и с каждым *номером прерывания* связана определенная программа1. Часть номеров прерываний из этого диапазона зарезер­ вирована системой, все остальные могут использоваться прикладны­ ми программами, в том числе и для обработки событий, связанных с обслуживанием нестандартного периферийного оборудования, под­ ключаемого к компьютеру.

Система обработки прерываний (как, впрочем, и любая другая программная система) состоит из трех взаимодействующих компо­ нентов:

* аппаратного комплекса, основу которого составляет *контрол­ лер прерываний,* обеспечивающий взаимодействие периферийных устройств с центральным процессором;
* программного обеспечения, представленного множеством

*программ обработки прерываний;*

* структур данных, главная из которых - это *таблица векто­ ров прерываний,* организуемая в ОЗУ и выполняющая роль адрес­ ного справочника, используемого для хранения указателей на про­ граммы обработки прерываний.

1 Не все номера из этого диапазона являются номерами *прерываний* - ряд но­ меров (например, INT lFh и INT 43h) используется системой для хранения адресов служебных структур данных, размещенных в памяти ПК.

**160**

По типу источников событий, требующих оперативной реакции процессора в режиме прерываний, различают *аппаратные* и *про­ граммные* прерывания, аппаратные прерывания, в свою очередь, могут быть *внутренними* или *внешними.*

* + 1. Аппаратные прерывания

Инициатором *внутренних аппаратных прерываний* может быть сам центральный процессор, если он обнаружит нештатную ситу­ ацию при выполнении очередной машинной команды. Например, событие «деление на ноль» вызовет прерывание INT ООь, а преры­ вание INT 04ь зарезервировано за событием «переполнение» при выполнении арифметической операции.

Запросы на обработку внутренних аппаратных прерываний гене­ рируются внутренними блоками процессора, а индикаторами собы­ тий, вызывающих такие прерывания, могут быть соответствующие биты *регистра флагов* центрального процессора (см. п. 4.3), содер­ жащего, в частности, коды результатов выполненных команд.

*Внешние аппаратные прерывания* обеспечивают асинхронное взаимодействие центрального процессора с периферийным обору­ дованием, подключаемым к системной магистрали через соответ­ ствующие адаптеры (рис. 4.2), и являются следствием наступления некоторого внешнего события, например: прочитан очередной сек­ тор диска, нажата (или отпущена) клавиша клавиатуры, принтер послал сигнал готовности, сработал датчик внешнего оборудова­ ния, управляемого компьютером.

Адаптеры периферийных устройств передают запросы *контрол­ леру прерываний* (п. 7.1.3), основная задача которого- это иденти­ фикация устройства, инициировавшего такой запрос, и формирова­ ние соответствующего *номера прерывания.* Контроллер прерываний выставляет этот номер на системную шину данных, а по шине управления передает центральному процессору специальное сооб­ щение - запрос на прерывание.

Центральный процессор, получив запрос на прерывание, читает с шины данных номер прерывания и запускает стандартную *про­ цедуру обработки прерывания,* алгоритм реализации которой будет рассмотрен в п. 7.1.5.

* + 1. Программные прерывания

Источниками программных прерываний, как это следует из их названия, являются программы - как системные, так и при­ кладные. Процессор, обнаружив в машинном коде команду *INT п,* запускает стандартную процедуру обработки прерывания (см. п. 7.1.5), которая вызывает программу обработки прерывания № *п,* которая к этому моменту должна быть загружена в память ПК.

**161**

Программные прерывания эффективно используются в MS DOS для обращения к множеству системных функций: например, пре­ рывание INTlOь обеспечивает доступ к системным функциям видео­ сервиса, прерывание INT lбь - к функциям, обслуживающим ввод данных с клавиатуры, а прерывание INT 21ь - к функциям управ­ ления памятью и к функциям, реализующим файловые операции, некоторые из которых рассматривались в теме 6 данного курса.

Программное прерывание не является «неожиданным» для цен­ трального процессора и синхронизировано с процессом выполне­ ния команд машинной программы - в этом смысле оно является

«искусственным», использующим специфический способ запуска программ, созданный для обработки аппаратных прерываний. Для запуска такой программы достаточно связать с ней номер некото­ рого прерывания и записать соответствующий указатель в *таблицу векторов прерываний* (п. 7.1.4).

* + 1. Контроллер прерываний

Контроллер прерываний - специализированная микросхема (или несколько каскадно-соединенных микросхем), к входным лини­ ям которой подключаются контроллеры периферийных устройств - источников прерываний. В качестве источников прерываний могут выступать контроллеры клавиатуры, системного таймера или дат­ чиков различных технических устройств, требующих реакции цен­ трального процессора на поступающие от них сигналы.

Существенным здесь является то, что центральный процессор не может прогнозировать время поступления этих сигналов - со­ гласитесь, что трудно предугадать момент времени, когда задумчи­ вый пользователь ПК найдет на клавиатуре нужную ему клавишу и, наконец, нажмет на нее, и еще труднее прогнозировать момен­ ты срабатывания подключенных к компьютеру датчика задымлен­ ности воздуха в помещении или датчика температуры тормозной колодки автомобиля, управляемого бортовым компьютером. При этом процессор должен обеспечить оперативную (и правильную) реакцию на каждый из поступивших сигналов путем запуска со­ ответствующих программ, временно прерывая выполнение других программ, некоторые команды которых к этому моменту уже были загружены в регистр очереди команд (см. рис. 4.3).

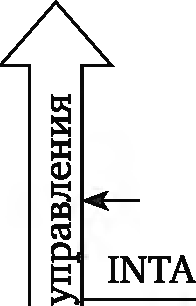
На рис. 7.1 приведена упрощенная функциональная схема контрол­ лера прерываний Intel 8259А, используемого в IВМ-совместимых ПК. Этот контроллер представлен в адресном пространстве ввода-вывода двумя портами (20ь и 21ь), через которые производится обмен дан­ ными и управляющими сигналами с центральным процессором ПК.

Intel 8259А- это программируемый контроллер, который может находиться в двух состояниях: в рабочем состоянии обслуживания запросов и в состоянии настройки. Основная задача, решаемая кон-

**162**

троллером в состоянии обслуживания запросов, - идентификация внешних событий, требующих прерывания программы, выполняе­ мой центральным процессором.

=QJ



INT

=:s:s:: RD

WR

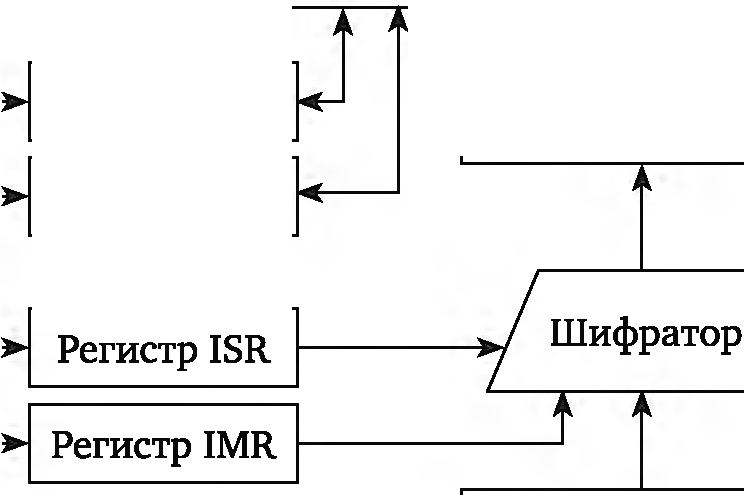
C\j

=):

QJ

u

u::s::



Системная шина данных

Буферный регистр: DO - D7

Регистр OCW

Регистр ICW

Регистр номера прерывания

C\j

*Q..*

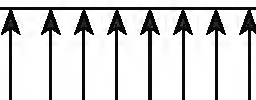
о

u

,::s::

о

Регистр IRR

IRQO  IRQ7

Внешние устройства

*Рис.7.1.* **Функциона11ьная структура контро1111ера прерываний**

Обобщенный алгоритм решения этой задачи можно представить следующим набором шагов:

прием запроса на прерывание от внешнего устройства;

* идентификация источника запроса;
* формирование номера прерывания, соответствующего ис- точнику;

передача номера прерывания центральному процессору;

* передача центральному процессору сообщения о прерывании;
* прием от центрального процессора подтверждения приема переданного сообщения.

Приведем более детальное описание этого алгоритма и проком­ ментируем назначение основных функциональных блоков контрол­ лера, показанных на рис. 7.1.

Контроллер ожидает приема запросов на прерывания - сигна­ лов IRQO - IRQ71 от контроллеров внешних устройств, и временно

1 Предусмотрена возможность увеличения числа линий запросов пуrем каскад­ ного подключения еще одного контроллера прерываний. В этом случае линия IRQ2 не используется для подключения внешнего устройства, она обеспечивает связь между основным и «ведомым» контроллерами прерываний.

**163**

сохраняет принятые запросы в В-разрядном регистре *IRR (InterRupt Register).* Единичное значение какого-либо бита этого регистра оз­ начает поступление запроса от устройства, подключенного к соот­ ветствующей линии IRQ.

Номер входной линии IRQ определяет *уровень прерывания,* уста­ навливающий приоритет запроса: как правило (если задан режим фиксированных приоритетов), наивысший приоритет имеет линия IRQO, и далее в порядке убывания - до линии IRQ7, имеющей низ­ ший приоритет.

Восьмиразрядный *регистр маскирования прерываний IMR (Inter­ rupt Mask Register)* используется для запрета (маскирования) об­ работки запросов, поступающих на соответствующие входы IRQ: единичное значение i-го бита регистра IMR маскирует запросы пре­ рываний, поступающие на i-й IRQ-вxoд регистра IRR. Программиро­ вание регистра IMR производится через порт 21ь,

Единичное значение i-го бита восьмиразрядного *регистра об­ служиваемых прерываний ISR (Interrupt Service Register)* показывает, прерывания каких уровней обрабатываются в микропроцессоре.

Устройство управления контроллера вырабатывает выходной сигнал INT, связанный по системной шине управления с входом INTR микропроцессора (см. рис. 4.3), и ожидает поступления от ми­ кропроцессора сигнала INTA, подтверждающего факт принятия им прерывания на обслуживание.

Микропроцессор, получив по шине управления сигнал INТR, ана­ лизирует состояние флага IF (см. рис. 4.3), единичное состояние ко­ торого разрешает, а нулевое - запрещает аппаратные прерывания. Если прерывания запрещены (IF = О), прерывание не выполняет­

ся до момента установки IF в единицу. Иначе (IF = 1):

* флаг IF сбрасывается в ноль;
* запускается процедура обработки прерывания, по заверше- нию которой флаг IF устанавливается в единицу;
* формируется выходной сигнал подтверждения прерывания INTA, связанный по системной шине управления с одноименным входным сигналом контроллера прерывания.

Контроллер прерываний, получив по системной шине управле­ ния сигнал подтверждения прерывания INTA:

* сбрасывает в ноль бит регистра IRR, соответствующий уров­ ню IRQ обработанного процессором прерывания;
* устанавливает в единицу соответствующий бит регистра ISR, что фиксирует факт принятия прерывания к обработке процессо­ ром;
* в соответствии с состоянием регистров IRR, IMR и ISR фор­ мирует *номер прерывания* - однобайтовое число, соответствующее уровню прерывания, т. е. номеру активной линии IRQ. Номер пре­ рывания - это, по существу, числовое «имя» внешнего события,

**164**

требующего прерывания работы процессора. Например, контрол­ лер системного таймера соединен с линией высшего приоритета IRQO, и ему стандартно присвоен 8-й номер, а контроллер клавиа­ туры связан с линией IRQl, и этой линии соответствует прерывание

№9.

Шифратор выполняет также функцию *арбитра приоритетов,* разрешающего конфликты при одновременном поступлении IRQ­ запросов разных уровней. Если поступившее прерывание уровня *i* не маскировано (IMR[i] = О), и приоритет прерывания, обрабаты­ ваемого процессором в данный момент, ниже приоритета поступив­ шего прерывания (ISR > *i),* шифратор вычисляет *номер прерывания,* который записывается в регистр номера прерывания и далее, через *буферный регистр,* передается на системную шину данных, доступ­

ную центральному процессору. В микропроцессоре этот номер ис­ пользуется для вычисления адреса программы обработки прерыва­ **ния.**

Если до завершения обработки выполняемого прерывания по­ ступит очередной IRQ-зaпpoc, то его приоритет определит дальней­ шие действия контроллера.

Если приоритет вновь поступившего IRQ-зaпpoca ниже или ра­ вен приоритету выполняемого прерывания:

* информация о новом IRQ-зaпpoce будет сохранена в регистре IRR (установкой в единицу соответствующего бита);
* обслуживание этого прерывания будет отложено до момента завершения программы обработки выполняемого прерывания.

Если приоритет вновь поступившего IRQ-зaпpoca выше приори­ тета выполняемого прерывания:

* прерывается программа обработки выполняемого прерыва­ ния (с сохранением в стеке всех оперативных данных этой програм­ мы, необходимых для продолжения ее работы);
* вызывается программа обработки нового прерывания.

В состоянии настройки контроллер принимает и сохраняет в од­ ноименных регистрах команды инициализации *ICW (Initialization Command Words),* задающие режим его работы, и операционные ко­ манды *OCW (Operation Control Words),* позволяющие перепрограм­ мировать контроллер, например, маскировать или размаскировать прерывания, менять их уровни (приоритеты).

Соответствующим программированием контроллера можно устанавливать специальные режимы завершения прерываний или, например, переводить контроллер в режим опроса *(Polling Mode),* в котором аппаратные прерывания не происходят автоматически, а центральный процессор считывает содержимое регистра IRR, по­ лучает от контроллера информацию о наличии и уровнях поступив­ ших запросов на прерывания и сам принимает решение о запуске процедуры прерывания.

**165**

* + 1. Таблица векторов прерываний

*Вектором прерывания* называется начальный адрес программы обработки этого прерывания (или адрес иной структуры данных, связанной с этим прерыванием). Вектор прерывания хранится в сегментной форме (см. п. 4.6.1) и занимает в памяти два двухбай­ товых машинных слова: слово со старшим адресом содержит номер сегмента, а слово с младшим адресом - смещение от начала этого сегмента.

Хранение векторов прерываний организовано в специальной структуре данных, называемой *таблицей векторов прерываний,* которая позволяет однозначно связать *номера прерываний* с соот­ ветствующими им *векторами прерываний.* Таблица векторов пре­ рываний формируется в ОЗУ и занимает первый килобайт нулевого сегмента адресного пространства (от [0000:ОООО]ь до [0000:ЗFFF] ь). Все векторы прерываний расположены в этой «таблице» линейно в порядке возрастания соответствующих им номеров прерываний (от ООь до FFь), и каждый вектор занимает в таблице 4 байта, что позволяет использовать простую формулу 4 • *п* для определения сме­ щения вектора *п-ного* прерывания относительно начала таблицы.

Таким образом, *номер* прерывания *п,* умноженный на 4, опреде­ ляет сегментный адрес 4-байтовой ячейки памяти [ОООО:4п]ь, в ко­ торой расположен *вектор п-го* прерывания, который, сам являясь сегментным адресом, выполняет роль указателя на программу об­ работки этого *(п-го)* прерывания.

На рис. 7.2 показан начальный фрагмент таблицы векторов прерываний. По адресу [0000:ОООО]ь хранится сегментный адрес [ООА7:1068]ь - это вектор О-го прерывания, представленный сег­ ментом № [ООА7]ь и смещением [1068]ь от начала этого сегмен­ та. Следующие 4 байта представляют вектор 1-го прерывания [0070:018В]ь, затем - 2-го, и т. д. до последнего вектора прерыва­ ния с самым большим номером FFь.

**m*(j)*m*ffi(j)*0m*ffi*** (j)(j)(j)I!). **68 1Ш** А? (j)(j) **8В** Ш1 ?Ш ***U)(j)*** 16 **шю** 9ш6с шз **8В** Ш1 ?шШс (j)(j) h.з.Jl.p...Ц.Jl.p.

В9

шз

шз

(j)(j)i(j) **8В** Ш1 ?Ш **mm**

1!)6 шс Ш2 41!) Ш?

Ш2 FF шз

Ш2 Jl.p.]I...@... ...

(j)(j)(j)(j) ШшШш2зШш

ШЕ

83 Ш1 ЕС ШF ЕС 1!)6 ***11***

ЗА (j)(j) 96

54 (j)(j) 9ш6с

Г.ь.ь Ц.Т.Ц.

**mmm0m**

**6Е** (j)(j) 9ш6с шз **88 mm** 9ш6с шз А2 (j)(j) 9ш6с шз FF шз шс Ш2 **n.Ц.И.Ц.в.Ц.**

(j)(j)(j)(j) ШШ4Ш **А**в**9**ш 1!)8 шс Ш2 **А4**

1!)9

1!)9

Ш2 АА

Ш2 5D Ш4

Ш2 **и ...д...к ... ] ...**

**шmmm** 1!)1!)51!)

1!)9

1!)9

шс Ш2 ШD Ш2 DшBс Ш2 **С4** шс Ш2 **8В** 1!)5 шс Ш2 \*····.1. Jl•••

1!)1!)61!) ШЕ

ШшСс Ш2 14 ШС Ш2 1F шс шс Ш2 AD 1!)6 шс Ш2 *. . - - - .... . .. н. ..*

**mшrum** ШШ?Ш **AD** 1!)6 Ш2 **А4** FШ (j)(j) FШ З? 1!)5 шс Ш2 F1 8F (j)(j) сш н ...11.Е.Е?...ёП.

***(j)(j)(j)(j)*** 1!)1!)81!) ?2 **1Ш** А? (j)(j) ?С **1Ш** А? (j)(j) **?F** Ш1 15 Ш4 **4В** Ш1 15 Ш4 ***1" .З.: .З.6 . .. J<. ..***

***(j)(j)(j)(j)*** 1!)1!)91!) 56 Ш1 15 Ш4 **86 1Ш** А? (j)(j)

91!)

9А

**1Ш** А? (j)(IJ **9А 1Ш** А? (j)(j) U...Ж.з.Р.з.Ь.з.

(j)(j)(j)(j) ШШАШ

54

С5

Ш1 ЕС ШF

Ш2 ?Ш (j)(j) F2 Ш4

DЗ В8 **1Ш** А? (j)(j) t.ь.Т.р.С.Ь"J.з.

***(j)*ш*(j)*ш*(j)*ru*(j)*ш** шшвш **В8 1Ш** А? (j)(j) В8 **1Ш** А? (j)(j) 41!) Ш1 15 Ш4 **1Ш** ШЕ ***11*** ШЕ 'J *.З.'j .З* .@ *11*

шшсш ЕА АЕ 11!) А? **шю** 38 **mm** FШ **В8 1Ш** А? **mm** С4 23 Ш2 DШ ьо .з.8.E'J .з.-11.

(j)(j)(j)(j) ШШDШ **В8 1Ш** А? (j)(j) **В8 1Ш** А? (j)(j) В8 **1Ш** А? (j)(j) **В8 1Ш** А? (j)(j) *'j .З.'j .З.'j .З.'j .З.*

***(j)ffi(j)(j)*** ШШЕШ **В8 1Ш** А? (j)(j) В8 **1Ш** А? ***(iJ(j)* В8 1Ш** А? (j)(j) В8 **1Ш** А? ***(iJ(j) =J .З.=J .З.=J .З.=J .З.***

**mm0m mm** ш *(!)(!)* ш **mm** ш *(!)(!)*

(j)(j)p(j) **В8 1Ш** А? В8 А? **В8** А? В8 А? *'j .З.'j .З •'1 .З.'j .З.*

шз

(j)(j)(j)(j) l!J1(j)(j) **8А** Ш4 шс Ш2 FF шс Ш2 ?с6ш шс Ш2 F1 А1 (j)(j) сш **к ... ...u ...ёб.**

1!)9

***ffi(i)(j)U)*** 1!)111!) FЗ **38 mm** FШ FЗ 38 **шш** FШ **шю** 41!) **шш** FЗ 38 **шш** FШ е8.Ее8.Е L.@.e8.Е

**mm0m** Ш12Ш FЗ **38 mm** FШ F3 38 **mm** FШ F3 **38 mm** FШ F3 38 **mm** FШ е8.Ее8.Ее8.Ее8.Е

(j)(j)(j)(j) 1!)131!) F3 **38** (j)(j) FШ F3 38 (j)(j) FШ F3 **38** (j)(j) FШ F3 38 (j)(j) FШ е8.Ее8.Ее8.Ее8.Е

**mmmm mm mm mm mm**

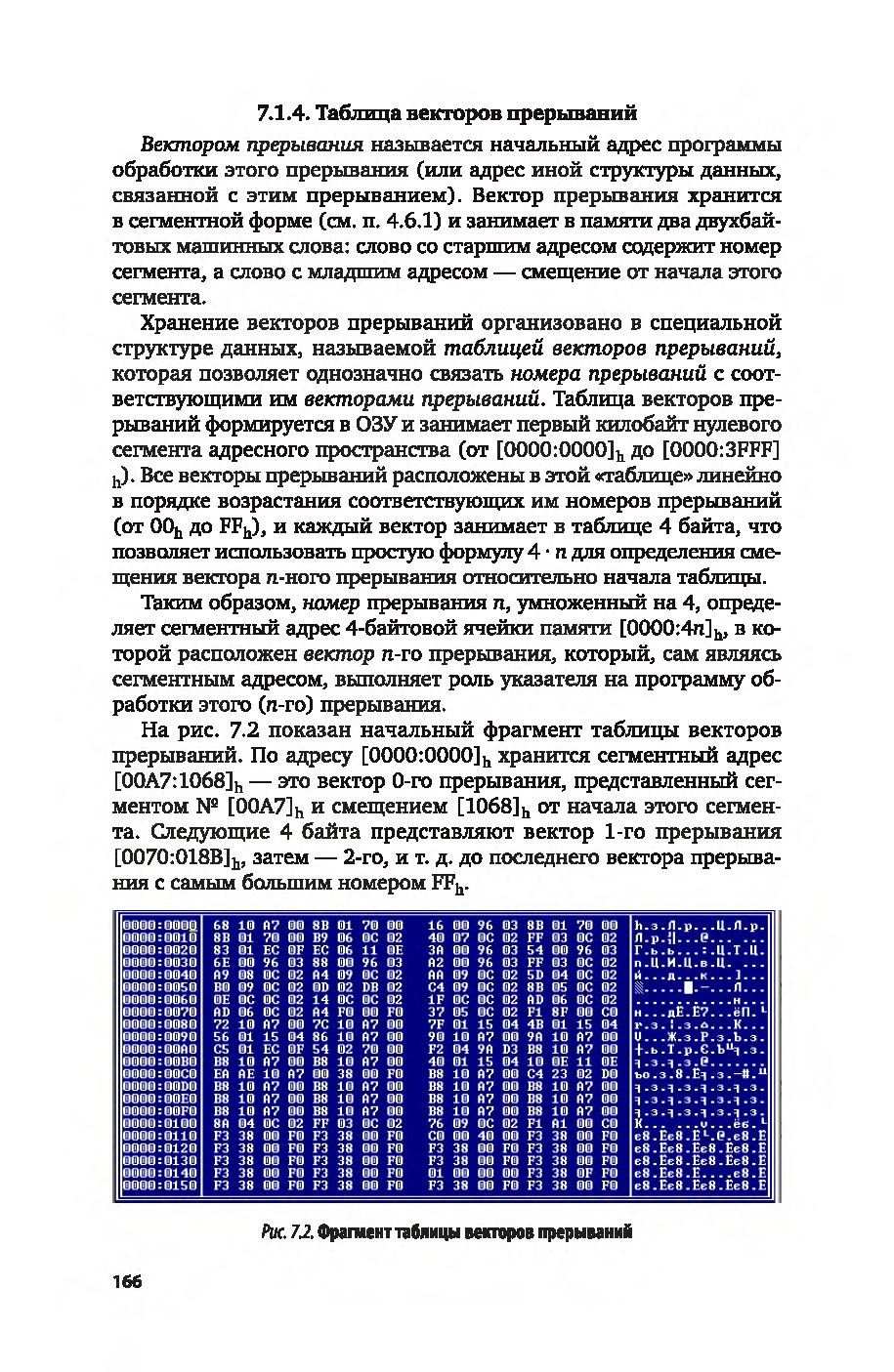
(j)i4(j) FЗ **38** FШ F3 38 FШ Ш1 (j)(j) F3 38 ШF FШ е8.Ее8.Е е8.Е

**mm0m**

1!)151!)

F3 **38 mm** FШ F3 38 **mm** FШ F3 **38 mm** FШ F3 38 **mm** FШ е8.Ее8.Ее8.Ее8.Е

*Рис. 7.2.* **Фрагмент таблицы векторов прерываний**



166

https://urait.ru

Таблица векторов прерываний частично инициализируется BIOS перед началом загрузки DOS, частично - при загрузке DOS. Поль­ зовательские программы также могут модифицировать эту таблицу, используя номера свободных прерываний или переустанавливая (в том числе и «на себя») некоторые из уже используемых векторов прерываний.

* + 1. Процедура обработки прерывания

Если запрос прерывания принят на обслуживание центральным процессором, выполняется следующая типовая процедура:

1. перед тем, как исполняемая процессором программа будет прервана, все ее оперативные данные (промежуточные результа­ ты, временно сохраненные в регистрах общего назначения, содер­ жимое регистра флагов и буфера очереди команд) сохраняются в области оперативной памяти, отведенной под стек. Положение стека в ОЗУ определяется состоянием соответствующих адресных регистров процессора (см. рис. 4.3): номер сегмента стековой па­ мяти определяет сегментный регистр SS *(Stack Segment),* а вершину стека - регистр-указатель SP *(Stack Pointer).* Вершина стека - это единственная его точка, через которую происходит как запись дан­ ных в стек, так и извлечение из стека записанных в него данных, что позволяет реализовать протокол LIFO *(Last Input First Output)* обслуживания стека;
2. с системной шины данных считывается *номер прерывания п,*

переданный на эту шину контроллером прерываний;

1. формируется сегментный адрес [ОООО:4п]ь точки входа в та­ блицу векторов прерываний, и считывается *вектор прерывания,* расположенный по этому адресу;
2. выполняется программа обработки прерывания, на которую

«указывает» вектор прерывания:

* + компоненты вектора прерывания (номер сегмента и смеще­ ние) помещаются в адресные регистры процессора CS и IP,
  + данные этих регистров обрабатываются сумматором адреса, формирующим начальный (линейный) адрес программы обработки прерывания,
  + по адресной шине запрашивается и затем выполняется пер­ вая команда этой программы и т. д., до тех пор пока не будет вы­ полнена команда «конец программы»;

1. по завершении программы обработки прерывания из стека восстанавливаются данные прерванной программы, и продолжает­ ся выполнение ее очередной команды.

Сохранение в стековой памяти данных прерванной программы обеспечивает возможность иерархической обработки прерываний. Это означает, что если в процессе обработки одного прерывания про­ цессору поступает запрос на выполнение прерывания более высоко-

**167**

го приоритета, первый обработчик прерывания будет прерван (т. е. теперь уже к нему будет применена рассмотренная выше процедура) и продолжит свое выполнение только после завершения работы про­ граммы обработки второго (более приоритетного) прерывания.

Управление приоритетами внешних аппаратных прерываний осуществляет контроллер прерываний (см. рис. 7.1), принимающий запросы от адаптеров периферийных устройств на свои входные ли­ нии IRQ: с каждой линией связан определенный уровень приори­ тета. Если до окончания обработки высокоприоритетного прерыва­ ния поступает запрос низкого приоритета, он становится в очередь (в соответствии со своим приоритетом), в противном случае запрос передается центральному процессору для исполнения.

##### Клавиатура персонаnьноrо компьютера

Клавиатура ПК-одно из важнейших его периферийных устройств, предназначенное для ввода текстовой (буквенно-цифровой) и управ­ ляющей информации.

Клавиатура является программно-управляемым устройством и имеет в своем составе специализированный контроллер, основ­ ной функцией которого является отслеживание фактов нажатия и отпускания клавиш путем циклического сканирования наборно­ го поля клавиатуры, в результате которого формируются *скан-коды* нажатых (или отпущенных) клавиш.

Скан-код - это однобайтовый номер, присваиваемый каждой клавише. Семь младших битов скан-кода - это собственно уни­ кальный код клавиши, а старший бит используется для кодирования факта ее нажатия (О) или отпускания (1). При такой системе ко­ дирования с каждой клавишей связано два ее скан-кода: код нажа­ той клавиши всегда на 128 (27) меньше, чем код этой же клавиши, формируемый при ее отпускании, а общее количество кодируемых клавиш ограничивается числом 128: коды в диапазоне [О + 7Fь] со­ ответствуют нажатым клавишам, а коды в диапазоне [80ь + FFь] - отпущенным. Заметим, что такое ограничение не противоречит требованиям стандартных форматов клавиатур, предусматриваю­ щих 101, 104, 108 или 109 клавиш.

Следует понимать, что скан-коды клавиш определяются «схе­

мой распайки» наборного поля клавиатуры и напрямую не связаны с обозначениями символов, нанесенными на их поверхность. Раз­ мещение символов на наборном поле клавиатуры (так называемая *раскладка клавиатуры)* определено соответствующими стандарта­ ми, а соответствие между скан-кодом клавиши и АSСП-кодом од­ ного из связанных с ней символов определяется программно (см. п. 7.2.2) в соответствии с используемой кодовой таблицей символов.

**168**

* + 1. Контроллер клавиатуры

Процесс ввода данных с клавиатуры реализуется тремя аппарат­ ными устройствами компьютера - контршmером клавиатуры, кон­ тршmером прерываний и центральным процессором.

Контроллер клавиатуры взаимодействует с центральным процес­ сором через систему прерываний (см. подтему 7.1) и связан с ли­ нией IRQ-1 контроллера прерываний, что обеспечивает клавиатуре высокий уровень приоритета (второй после приоритета системного таймера, связанного с линией IRQ-0).

Контроллер клавиатуры - это однокристалльная микро-ЭВМ со своим «центральным» процессором, ПЗУ, устройствами ввода и вывода данных.

В качестве устройства ввода данных используется *наборное поле*

клавиатуры - электронно-механическое (или иное) устройство, схематично показанное на рис. 7.3. Электрические линии строк и столбцов образуют прямоугольную матрицу, в узлах которой рас­ положены нормально разомкнутые контакты, управляемые соот­ ветствующими клавишами. Таким образом, каждая клавиша кла­ виатуры компьютера однозначно идентифицируется номером [i, *j]* связанного с ней узла матрицы наборного поля.

1 t·!··J·...-·1

*i*- Выходной *i*

...... ..................... ... *i* порт *i*

:

Наборное

: .• --

,::ts:::::

о

1

Операционное , , • IRQl

нн=::J--..;...-++-----:п=о=--'л=е=------,i-+---:· .;..;- устройство

i'::;;,::·

: *:* :

,, +5 В• ·····@/' •• •••

+ iScan

-:мб-у;-\_фu\_:- -:1-=\_С\_о\_d--'е► Port 60h

*i* виатуры данных

*Рис.7.3.* **Схема наборного попя кпавиатуры**

Устройство ввода данных контроллера клавиатуры представлено двумя внутренними портами: входной порт, связанный с линиями строк матрицы наборного поля, подключенными через резисторы к источнику питания, уровень напряжения которого +5 В, соответ­ ствует логической единице, и выходной порт, подающий сигналы уровня логического нуля на линии столбцов.

При отпущенных клавишах контакты наборного поля разомкну­ ты, и сигналы всех линий строк, поступающие на входной порт кон­ троллера, соответствуют логической единице, т. е. каждая [ij]-я кла­ виша находится в состоянии [О;1].

**169**

При нажатии клавиши соответствующий [ij]-й контакт наборно­ го поля замыкается, и в очередном цикле сканирования в момент подачи нулевого сигнала на i-й столбец входной сигнал *сj-й* стро­ ки получит нулевое значение. В результате [ij]-я клавиша сме­ нит состояние с [0;1] на [О;О], что является индикатором события

«[ij]-я клавиша *нажата».*

При отпускании нажатой клавиши ее состояние изменится с [О;О] на [0;1], этот факт будет обнаружен операционным устрой­ ством контроллера в следующем цикле сканирования наборного поля и использован для регистрации противоположного события

«[ij]-я клавиша *отпущена».*

Программа, хранимая в ПЗУ контроллера и выполняемая в его операционном устройстве, реализует следующий алгоритм в каж­ дом цикле сканирования наборного поля (с периодичностью поряд­ ка 10 раз в секунду):

* на каждую *i-ю* линию столбцов матрицы через выходной порт последовательно посылается электрический сигнал, уровень напря­ жения которого соответствует логическому нулю;
* считываются сигналы, поступающие на входной порт от каж­ дой *j-й* линии строк.
* контролируется изменение состояния каждой [ij]-й клавиши;
* клавише, сменившей свое состояние, программно присваи- вается уникальный для нее 7-битный код и к этому коду добавляет­ ся старший бит (О- при нажатии клавиши, 1- при отпускании), в результате чего формируется скан-код клавиши;
* скан-код клавиши помещается во внутренний буфер данных, выполняющий роль устройства вывода контроллера клавиатуры;
* из внутреннего буфера данных скан-код нажатой (или отпу­ щенной) клавиши передается в порт бОь адресного пространства ввода-вывода, доступный программам, выполняемым центральным процессором;
* на линию IRQl контроллера прерываний посылается сигнал запроса прерывания.

При удержании клавиши в нажатом состоянии контроллер кла­ виатуры переходит в режим автоповтора, в котором периодически генерируется скан-код нажатой клавиши.

Далее, уже без участия контроллера клавиатуры, выполняется процедура обработки прерывания, детально описанная выше (см. подтему 4.5 и пп. 7.1.4, 7.1.5):

* контроллер прерывания принимает запрос по линии IRQl, формирует соответствующий этой линии номер прерывания «9», выставляет этот номер на шину данных и посьтает центральному процессору запрос на прерывание по шине управления;
* центральный процессор прерывает выполняемую програм­ му, определяет вектор прерывания, соответствующий полученному

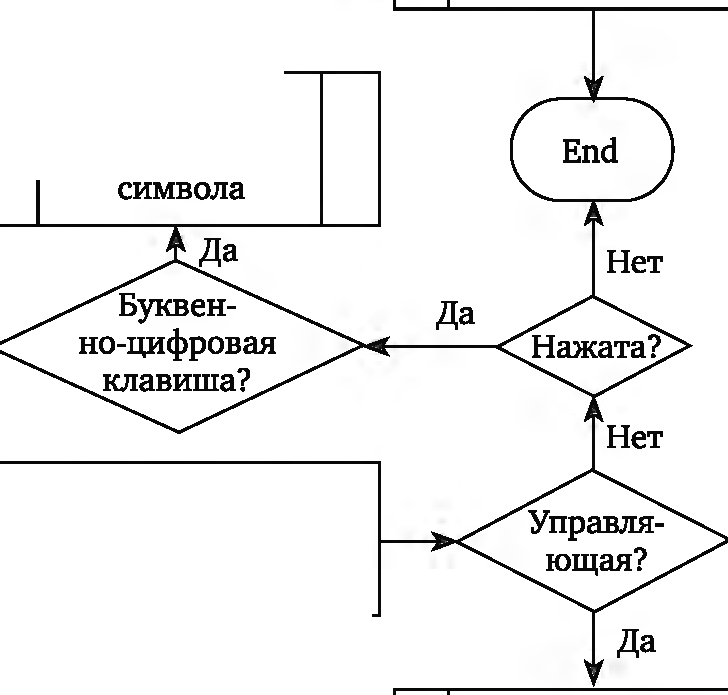
**170**

по шине данных номеру прерывания, и запускает программу обра­ ботки 9-го прерывания, соответствующую этому вектору.

* + 1. Алгоритм обработки прерывания клавиатуры

Программист, написавший в исходном коде прикладной про­ граммы input(), cin, getchar(), readln() или readkey, надеется на транслятор, который должен успешно справиться с задачей полу­ чения символьной информации от стандартного устройства ввода. У разработчика транслятора таюке есть надежда на то, что вызов со­ ответствующей системной функции избавит его от необходимости беспокоиться о деталях низкоуровневой реализации процесса обмена данными с клавиатурой. На чем же основаны эти надежды приклад­ ных и системных программистов? В том числе и на доверии к обра­ ботчику прерывания № 9, обеспечивающемупрограммный интерфейс между контроллером клавиатуры и прикладными программами, ожи­ дающими получения вводимых пользователями символьных данных. На рис. 7.4 приведена общая блок-схема алгоритма, реализуемо­

го программой обработки 9-го прерывания. Основное назначение этой программы - преобразование принятого скан-кода клавиши в код символа и передача этого кода прикладной программе для по­ следующей обработки.



Запись кодов в буфер клавиатуры

Определение АSСП-кода

Нет

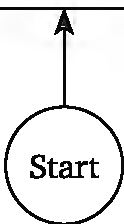
Определение типа клавиши

по ее 7-битному коду

Запись битовых флагов клавиатуры

Определение расширенного скан-кода

*Рис.7.4.* **Алгоритм обработки 9-го прерывания**



Чтение скан-кода клавиши

из порта 60h

Кроме этого, обработчик 9-го прерывания генерирует и сохра­ няет так называемые *расширенные скан-коды* для вспомогательных

**171**

клавиш или основных буквенно-цифровых клавиш при их нажа­ тии в комбинациях с управляющими клавишами *Ctrl* и (или) *Alt,* а также классифицирует принятые скан-коды управляющих клавиш и клавиш-переключателей, сохраняя информацию о текущем стату­ се клавиатуры для последующего использования как прикладными программами, так и самим обработчиком, например, для коррект­ ного преобразования скан-кодов клавиш в АSСП-коды символов.

**Структуры данных, используемые обработчиком 9-го прерывания**

Программа обработки 9-го прерывания использует три структу­ ры данных, зарезервированных для нее в области переменных ВIOS (табл. 7.1):

* два байта битовых «флагов» - индикаторов состояния управ­ ляющих клавиш и клавиш-переключателей;
* один байт для накопления АSСП-кода символа при его пря­ мом вводе (Alt + цифра);
* буфер клавиатуры, используемый для передачи прикладной программе АSСП-кода введенного пользователем символа или рас­ ширенного скан-кода вспомогательной клавиши или комбинации клавиш.

*Таблицаl.1*

**Бnок данных BIOS, обсnуживающий ввод скnавиатуры**

|  |  |  |  |
| --- | --- | --- | --- |
| **Начальный адрес** | **Длина, байтов** | **Назначение** | |
| [0040:0017]ь | 2 | Битовые флаги (статусы) клавиатуры - индика- торы состояния управляющих клавиш (рис. 7.4) | |
| [0040:0019]ь | 1 | Текущее (накопленное) значение вводимого АSСП-кода символа (Alt + цифра) | |
| [0040:ООlА]ь | 2 | **::s::**  l:Q C\j  §  о.  QJ | Указатель на «хвост» буфера - адрес (сме- щение по 40-му сегменту) ячейки, в которую обработчик 9-го прерывания должен записать код последнего символа, введенного с клави- атуры |
| [0040:ООlС]ь | 2 | Указатель на «голову» буфера - адрес ячейки, из которой прикладная программа должна прочитать код очередного введенно- го, но еще не прочитанного символа |
| [0040:ООlЕ]ь | Обыч- но32 | Собственно буфер (16 двухбайтовых элемен- тов) |
| [0040:ООВО]ь | 2 | Начальный адрес буфера (обычно - 001Еь) |
| [0040:0082]ь | 2 | Конечный адрес буфера+1 (обычно - ООЗЕь) |

**172**

**Формирование «флагов» клавиатуры**

Как показано на рис. 7.4, обработчик 9-го прерывания начина­ ет работу с чтения содержимого порта 60h и анализа прочитанного скан-кода последней нажатой (или отпущенной) клавиши.

Если прочитанный скан-код соответствует одной из управля­ ющих клавиш *Ctrl, Alt, Shift* или одной из клавиш-переключате­ лей *CapsLock, NumLock, ScrollLock, SysReq, Pause* или *Insert,* про­ грамма устанавливает соответствующие биты флагов клавиатуры (табл. 7.2) в «1» или «О» в зависимости от значения старшего бита скан-кода клавиши (нажата или отпущена), а для клавиш-переклю­ чателей - дополнительно изменяет значения соответствующих би­ тов на противоположное.

Например, при удержании любой из двух клавиш *Ctrl* второй бит первого байта флагов будет установлен в «1», а при нажатой правой клавише *Ctrl* дополнительно устанавливается в «1» первый бит вто­ рого байта флагов.

Еще один пример: при нажатии клавиши *CapsLock* 6-й бит перво­ го и второго байтов флагов устанавливается в «1», что соответствует ситуации: «установлен верхний регистр ввода символов, и клавиша *CapsLock* удерживается нажатой».

При отпускании этой клавиши 6-й бит второго байта будет сбро­ шен в «О», а первый байт останется без изменений, что соответству­ ет ситуации «установлен верхний регистр ввода символов, и клави­ ша *CapsLock* отпущена». После повторного нажатия и отпускания клавиши *CapsLock* оба этих бита будут сброшены в «О», что соот­ ветствует ситуации «установлен нижний регистр ввода символов, и клавиша *CapsLock* отпущена».

*Таблицаl.2*

**Структура байтов флагов клавиатуры**

|  |  |  |  |
| --- | --- | --- | --- |
| **1-й байт флагов [0040:0017]ь** | | **2-й байт флагов [0040:ООlВ]ь** | |
| **Бит** | **Статус клавиатуры при единичном значении**  **бита** | **Бит** | **Статус клавиатуры при единичном значении**  **бита** |
| о | Нажата правая клавиша *Shift* | о | Нажата правая клавиша *Ctrl* |
| 1 | Нажата левая клавиша *Shift* | 1 | Нажата левая клавиша *Alt* |
| 2 | Нажата (любая) клавиша *Ctrl* | 2 | Нажата клавиша *SysReq* |
| 3 | Нажата (любая) клавиша *Alt* | 3 | Установлен режим «Pause» |
| 4 | Установлен режим «ScrollLock» | 4 | Нажата клавиша *ScrollLock* |
| 5 | Установлен режим «NumLock» | 5 | Нажата клавиша *NumLock* |
| 6 | Установлен режим «CapsLock» | 6 | Нажата клавиша *CapsLock* |
| 7 | Установлен режим «Insert» | 7 | Нажата клавиша *Insert* |

**173**

Анализ структуры байтов флагов клавиатуры (см. табл. 7.2) по­ зволяет сделать следующие выводы:

1. парные управляющие клавиши *Shift, Ctrl* и *Alt* (отличающие­ ся значением своих скан-кодов) имеют отдельные позиции в байтах флагов, что позволяет идентифицировать ситуации с удержанием как левых или правых одноименных клавиш, так и обеих клавиш одновременно;
2. информация о клавишах-переключателях позволяет не толь­ ко идентифицировать установленный режим работы клавиатуры, но и определить текущее состояние клавиши-переключателя, уста­ новившей этот режим.

Алгоритм вычисления кодов символов

Все клавиши, не отнесенные к категориям управляющих и кла­ виш-переключателей (см. табл. 7.2), формально относятся к катего­ рии символьных клавиш. В эту категорию попадают все буквенно­ цифровые клавиши, а также вспомогательные клавиши, например, функциональные Fl - F12, клавиши табуляции и управления курсо­ ром, клавиши *Delete* и *BackSpace,* которые фактически символьными не являются и не требуют трансляции своих скан-кодов в АSСП-коды каких-либо символов. Объединяет все эти «формально символьные» клавиши только то, что информация об их нажатии хранится в бу­ фере клавиатуры в соответствующем двухбайтовом коде.

Если принятый из порта 60h скан-код соответствует «формаль­ но символьной» клавише, и при этом старший бит скан-кода этой клавиши имеет значение «О» (клавиша нажата), выполняется блок алгоритма (рис. 7.4), обеспечивающий трансляцию скан-кода либо в АSСП-код символа (для буквенно-цифровой клавиши), либо в рас­ ширенный скан-код (для вспомогательной клавиши или комбина­ ции нескольких «одновременно» нажатых клавиш).

*Основные буквенно-цифровые клавиши* - *АSСП-код символа.* Транслятор скан-кода буквенно-цифровой клавиши в АSСП-код со­ ответствующего ей символа может быть построен на базе бинарной таблицы, каждая строка которой представляет одну из таких клавиш в соответствии со стандартной раскладкой клавиатуры и используе­ мой кодовой таблицей символов: в левом столбце трансляционной таблицы записан скан-код клавиши, а в правом - АSСП-код сим­ вола. Такая структура таблицы позволит найти код символа по за­ данному значению скан-кода, но при этом потребуется решить про­ блему многозначности: одной клавише может соответствовать как минимум два кода символа в зависимости от установленного реги­ стра ввода.

Универсальное решение этой проблемы - сделать бинарную трансляционную таблицу тернарной, добавив третий столбец с АSСП-кодами заглавных букв и с кодами «верхних» символов циф-

**174**

ровых клавиш. Выбор одного из двух АSСП-столбцов в процессе трансляции зависит от состояния соответствующих флаговых битов (см. табл. 7.2), связанных с клавишами *Shift* и *CapsLock.* Возможна и оптимизация структуры трансляционной таблицы с учетом фак­ тического расположения строчных и прописных букв в кодовых та­ блицах символов. После завершения процедуры трансляции скан­ кода в АSСП-код оба эти кода записываются в буфер клавиатуры.

*Вспомогательные и виртуальные клавиши* - *расширенный скан­ код.* Прикладные программы, предназначенные для обработки сим­ вольной информации, например текстовые редакторы или специ­ ализированные программы верстки и дизайна книжной продукции, предлагают пользователям разнообразные «клавиатурные» приемы управления процессом редактирования, требующие использования не только буквенно-цифровых, но и вспомогательных клавиш.

Трудно представить себе текстовый редактор, не использующий клавиши навигации по тексту, клавиши *ТаЬ, Enter, Back Space, Delete, Ноте* или *End,* но количество таких клавиш ограничено, и они да­ леко не покрывают всех потребностей профессионального редакти­ рования текстов.

В этих условиях используют *виртуальные клавиши,* каждая из ко­ торых представлена комбинацией из нескольких клавиш, одновре­ менно удерживаемых в нажатом состоянии. Чаще всего использу­ ют комбинации из двух клавиш, одна из которых - управляющая *(Shift, Ctrl* или *Alt),* а другая может быть как буквенно-цифровой, так и любой из вспомогательных, что позволяет в четыре раза уве­ личить количество «клавиш», физически не расширяя наборное поле клавиатуры. Каждая «виртуальная клавиша» может быть за­ программирована на выполнение определенной операции.

Информация о нажатых вспомогательных клавишах и комби­ нациях нажатых клавиш кодируется двухбайтовым двоичным чис­ лом - так называемым «расширенным скан-кодом», в котором младший байт содержит скан-код основной клавиши, а значение старшего байта определяется тем, с какой из управляющих клавиш она скомбинирована (в некоторых случаях- «О»).

Значения расширенных скан-кодов стандартизированы, проце­ дура их формирования использует соответствующие трансляцион­ ные таблицы и учитывает состояние битовых флагов, связанных с клавишами *Shift, Alt* и *Ctrl.* Примеры расширенных скан-кодов приведены в табл. 7.3.

Расширенный скан-код хранится в буфере клавиатуры точно так же, как и пара «скан-код -АSСП-код» нажатой буквенно-цифровой клавиши. Такая система хранения предоставляет прикладным про­ граммам унифицированный доступ к символьным клавишам - как к основным буквенно-цифровым, так и к вспомогательным и вирту­ альным.

**175**

*Таблицаl.З*

**Примеры кодирования клавиш**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Группа клавиш** | **Клавиша или комби- нация клавиш** | **Port**  **бОь** | **Буфер клавиатуры** | |
| **Скан-**  **КОД** | **ASCII** | **Расширенный скан-код** |
| Управ- ляющие клавиши | Left Shift | 2Аь | - | - |
| Right Shift | Збь | - | - |
| Left Ctrl | lDь | - | - |
| Right Ctrl | llDь | - | - |
| Alt | 38ь | - | - |
| Caps Lock | ЗАь | - | - |
| NumLock | 145ь | - | - |
| Буквенно- цифровые клавиши | 1 / ! | 02ь | Зlь / 21ь | - |
| 2/@ | ОЗь | 32ь / 40ь | - |
| 3/# | 04ь | ЗЗь / 2Зь | - |
| q/Q | lОь | 71ь/ Slь | - |
| w/W | llь | 77ь/ 57ь | - |
| е/Е | 12ь | 65ь/ 45ь | - |
| r /R | 1Зь | 72ь/ 52ь | - |
| z/Z | 2Сь | 7Аь/ SАь | - |
| Вспомога- тельные клавиши | END | 4Fь | - | 4FЕОь |
| НОМЕ | 47ь | - | 47ЕОь |
| DEL | SЗь | - | SЗЕОь |
| Back Space | ОЕь | - | ОЕО8ь |
| Enter | 1Сь | - | lC0Dь |
| Fll | 57ь | - | 8500ь |
| F12 | 58ь | - | 8600ь |
| Виртуаль- ные клави- ши (ком- бинации клавиш) | Alt+z | - | - | 2СООь |
| Ctrl+z | - | - | 2С1Аь |
| Ctrl+Enter | - | - | lСОАь |
| Ctrl+Fll | - | - | 8900ь |
| Ctrl+F12 | - | - | 8АООь |
| Alt+Fll | - | - | 8В00ь |
| Alt+F12 | - | - | 8СООь |
| Shift+Fll | - | - | 8700ь |
| Shift+F12 | - | - | 8800ь |

**176**

Алгоритм заполнения и чтения буфера клавиатуры

Буфер клавиатуры - это область ОЗУ, в которую обработчик 9-го прерывания записывает данные о нажатых символьных клави­ шах и из которой прикладные программы считывают эти данные в порядке их записи. Для прикладной программы буфер клавиату­ ры - это устройство ввода, поставляющее программе (разумеется, по ее запросу) коды вводимых пользователем символов для их про­ граммной обработки.

Обмен данными между программами через буферную область па­ мяти требует выполнения, как минимум, двух условий: во-первых, каждой из программ, участвующих в обмене, должен быть известен адрес расположения буферной области и, во-вторых, размер этой области должен быть достаточным для временного хранения блока передаваемых данных.

Обработчик 9-го прерывания и прикладная программа обмени­ ваются блоками размером в 2 байта (скан-код клавиши+ АSСП-код символа для основных буквенно-цифровых клавиш или расширен­ ный скан-код для вспомогательных), поэтому двухбайтовой ячейки памяти было бы вполне достаточно для организации буфера обмена. Однако в условиях, когда скорости процессов ручного ввода данных и их программной обработки могут существенно отличаться (как в ту, так и в другую стороны), использование такого «короткого» буфера неизбежно приводило бы к потерям времени на ожидание моментов освобождения или заполнения буфера.

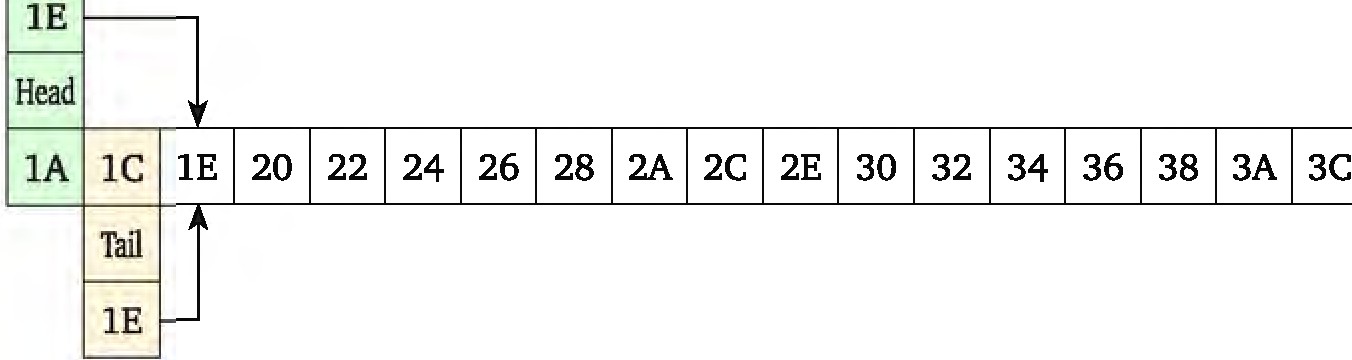
В первых моделях ПК поддерживался буфер клавиатуры фикси­ рованной длины в 32 байта, рассчитанный на регистрацию 16 по­ следовательных нажатий клавиш и занимавший в области данных BIOS стандартный диапазон адресов ячеек памяти. В дальнейшем появилась возможность управления длиной и местом расположения буфера: его начальный и конечный адреса также указываются в со­ ответствующих ячейках области данных BIOS (см. табл. 7.1).

Буфер клавиатуры как структура данных - это циклический спи­ сок типа «очередь», обслуживаемый по протоколу FIFO *(First Input First Output)* - т. е. чтение данных из буфера производится последо­ вательно в порядке их записи, а по достижении конца очереди (как при чтении, так и при записи) осуществляется переход к ее началу. Процессы записи и чтения буфера управляются двумя указателя­ ми: указатель «хвоста» очереди *(Tail)* определяет адрес очередной ячейки буфера, в которую обработчик прерывания должен записать блок данных о последней нажатой клавише, а указатель «головы» буфера *(Head)* - адрес ячейки буфера, из которой прикладная про­ грамма должна прочитать первый из еще не прочитанных блоков данных. Оба указателя представляют адреса соответствующих ячеек буфера клавиатуры (используется только часть сегментного адре­ са - смещение по сегменту 40h), для хранения каждого из них за-

**177**

резервирована двухбайтовая ячейка в области данных BIOS (см. табл. 7.1).

До того, как будет введен первый символ, буфер клавиатуры на­ ходится в исходном состоянии, как показано на рис. 7.51. В этом со­ стоянии значения обоих указателей одинаковы и совпадают с на­ чальным адресом буфера.



*Рис.7.5.* **Начальное состояние буфера клавиатуры**

После завершения каждой операции записи данных в буфер про­ грамма-обработчик 9-го прерывания инкрементирует (увеличивает на два) значение «хвоста», а программа, читающая буфер, соответ­ ственно инкрементирует значение «головы» - таким образом, при вводе данных с клавиатуры хвост буфера постоянно «убегает впе­ ред», а голова «догоняет».

На рис. 7.6 показана ситуация, когда в буфер записаны данные о восьми нажатых символьных клавишах, а прочитаны пока только пять (очевидно, пользователь вводил данные быстрее, чем приклад­ ная программа могла их обрабатывать).

281

Head

­

IA IC IE 20 22 24 26 28 2А 2С 2Е 30 32 34 36 38 ЗА ЗС

Tail

2Е 1 '

*Рис.7.6.* **Состояние «есть непрочитанные данные» буфера клавиатуры**

Буфер частично заполнен, в нем есть место для записи новых блоков данных, также имеются данные, еще не прочитанные при­ кладной программой. Запись очередного блока данных в буфер бу-

1 Все числа на рис. 7.5-7.8 заданы в шестнадцатеричной системе счисления и представляют адреса (смещения по сегменту 40h) соответствующих ячеек буфера клавиатуры. Курсивом показаны адреса ячеек буфера, а жирным шрифтом - значе­ ния указателей головы и хвоста буфера, таюке являющиеся адресами.

**178**

дет производиться по адресу «хвоста» (2Eh), а чтение - по адресу

«головы» (28h).

Если в буфер ничего не записывается (пользователь устал писать и решил немного отдохнуть), то чтение буфера будет продолжено, и после каждого прочтения указатель головы будет инкременти­ рован на 2 - и так до тех пор, пока значение указателя головы не сравняется со значением хвоста буфера. В этот момент *(Tail* =

= *Head)* в буфере не останется непрочитанных данных (рис. 7.7), бу­ дет идентифицирована ситуация «буфер логически пуст», и чтение должно быть прекращено.

2Е>

Head

lA lC lE 20 22 24 26 28 2А 2С 2Е 30 32 34 36 38 ЗА ЗС

Tail

2Е 1

*Рис.7.7.* **Состояние «яоrически пуст» буфера клавиатуры**

Буфер клавиатуры организован циклически: при достижении со­ ответствующим указателем конца буфера (т. е. когда он станет рав­ ным ООЗЕh после очередного «+2») этот указатель устанавливается в начало буфера (00lEh).

Ситуация «буфер переполнен» иллюстрируется рис. 7.8: указа­ тель «головы» буфера остался в состоянии 2Eh, так как теперь уже прикладная программа «взяла паузу» перед прочтением очередного блока данных (разумеется, эта пауза нужна программе не для отды­ ха, а для обработки предыдущего блока введенных пользователем данных), а пользователь продолжает вводить данные и ввел очеред­ ные 16 символов.

Формальным идентификатором ситуации «буфер переполнен» яв­ ляется равенство ITail - Head **1** = [Длина буфера] с учетом циклической

организации буфера. Попытка ввода данных об очередной нажатой клавише в переполненный буфер должна бьrrь (и будет) заблокирова­ на, о чем пользователь будет уведомлен звуковым сигналом.

2Е1

Head

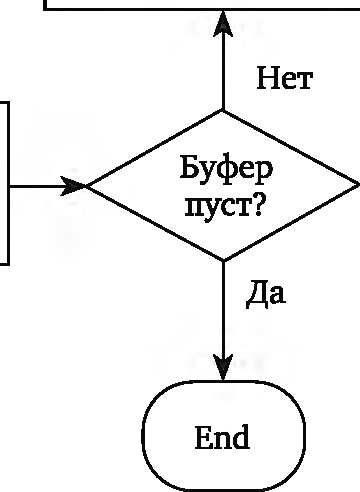
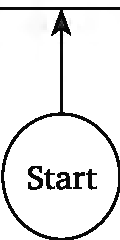
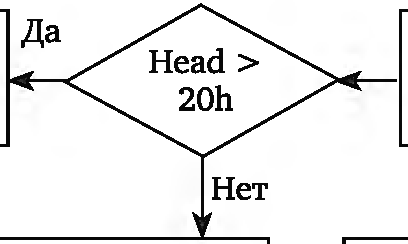
IA IC IE 20 22 24 26 28 2А 2С 2Е 30 32 34 36 38 ЗА ЗС

Tail 2С1

*Рис.7.8.* **Состояние «переполнения» буфера клавиатуры**

**179**

Блок-схема алгоритма чтения данных из буфера соответствую­ щей системной функцией (например, функцией № ООь программ­ ного прерывания INTlбь) приведена на рис. 7.9, а блок-схема алго­ ритма реализации модуля «Запись кодов в буфер клавиатуры» (см. рис. 7.4) программы обработки 9-го прерывания- на рис. 7.10.



Head --20h

Чтение указателя головы буфера Head

Чтение указателя хвоста буфера Tail

Чтение дан­ ных из буфера по адресу Head

Head ++ 2

*Рис.* 7.9. **Апrоритм чтения данных из буфера клавиатуры**

Рассмотрим пример короткой программы (листинг 7.1), которая предлагает пользователю ввести с клавиатуры произвольный длин­ ный текст и сохраняет результат ввода в переменной *kb\_inp* строко­ вого типа:

**Листинг 7.1. Пример программы однократного ввода символьной строки**

print('Enter а long text, as you want:') kb\_inp = input()

print(kb\_inp)

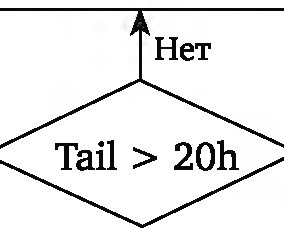
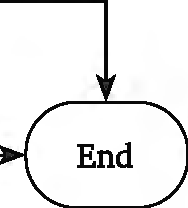
Функция *input()* циклически обращается к буферу клавиатуры (разумеется, используя для этого вызовы соответствующих систем­ ных функций). Пока пользователь не нажимает символьных клави­ шей, функция идентифицирует ситуацию «буфер логически пуст» *(Tail* = *Head* на рис. 7.6) и *не считывает* из буфера данные, располо­ женные по адресу *Head* (см. рис. 7.8).

В момент нажатия очередной символьной клавиши центральный процессор прерывает работу прикладной программы и передает управление программе обработки 9-го прерывания, которая запи­ сывает в буфер код введенного символа, инкрементирует значение указателя хвоста буфера (см. рис. 7.9) и заканчивает работу, возвра­ щая управление прерванной прикладной программе.

В новой ситуации буфер уже «не пуст», и прикладная программа читает код введенного символа по адресу головы буфера *Head,* со­ храняет его в ячейке памяти, выделенной транслятором для объ­ екта, связанного с переменной *kb\_inp,* затем инкрементирует зна-

**180**

чение указателя головы буфера (Head ++2 на рис. 7.8). В результате буфер клавиатуры снова оказывается в состоянии «логически пуст», и программа ожидает ввода очередного символа.



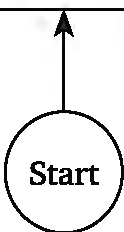
Tail -- 20h

Нет

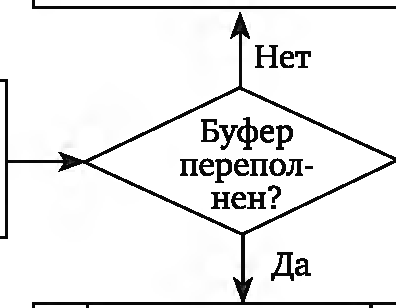
Tail ++ 2

ВЕЕР

*Рис.7.10.* **Аnrоритм записи данных в буфер кnавиатуры**



Чтение указателя хвоста буфера Tail



Чтение указателя головы буфера Head

Запись данных в буфер по адресу

Tail

Процесс чтения буфера клавиатуры будет продолжаться до тех пор, пока пользователь не нажмет на клавишу *Enter:* прикладная программа, прочитав из буфера расширенный скан-код этой клави­ ши, завершит выполнение оператора *input()* и продолжит работу.

Заметим, что в такой ситуации пользователь может продолжать нажимать символьные клавиши - при каждом нажатии клавиши обработчик 9-го прерывания будет записывать в буфер клавиатуры очередной блок данных, но эти данные (введенные после нажатия клавиши *Enter)* уже не будут восприняты прикладной программой, приведенной в листинге 7.1.

Для подтверждения правильности последнего утверждения мож­ но провести эксперимент с другой программой (листинг 7.2), в ко­ торой запрашивается ввод второй строки с 10-секундной задержкой после вода первой.

**листинг 7.2. Пример программы ввода двух символьных строк**

import time

print('Enter а two long texts, as you want:') first\_kb\_inp = input()

print(first\_kb\_inp) time.sleep(10) second\_kb\_inp = input() print(second\_kb\_inp)

**181**

Если пользователь, не дожидаясь истечения 10 секунд, введет вторую строку, программа прочтет ее после истечения паузы - точ­ но так же, как если бы эта строка была введена «вовремя».

Алгоритм прямого ввода кода символа

Любой символ В-битовой кодовой таблицы, в том числе и не свя­ занный ни с одной из клавиш клавиатуры, можно ввести, используя следующий прием: удерживая нажатой левую клавишу *Alt,* после­ довательно набрать АSСП-код символа в десятичной системе счис­ ления, используя дополнительные цифровые клавиши (разумеется, при их наличии на клавиатуре ПК), и отпустить клавишу *Alt* после завершения ввода кода символа.

Для временного хранения вводимого АSСП-кода символа в об­ ласти данных BIOS зарезервирована однобайтовая ячейка с фик­ сированным адресом [0040:0019]ь (см. табл. 7.1). Обработчик 9-го прерывания, получив информацию о скан-коде нажатой до­ полнительной цифровой клавиши, анализирует состояние флагов клавиатуры и, если битовый флаг, соответствующий левой клавише Alt, установлен в единицу, определяет числовой двоичный код, со­ ответствующий введенной десятичной цифре, затем суммирует его с числовым кодом, записанном в ячейке-накопителе АSСП-кода, за­ писывает полученную сумму в эту же ячейку и заканчивает работу. Под «суммированием» здесь понимается поразрядное суммиро­ вание со сдвигом: например, если в ячейке записан двоичный код числа 2, и нажата клавиша с цифрой 9, то после такого «суммирова­ ния» в ячейку будет записан двоичный код числа 29 (2 • 10 + 9). При суммировании учитывается ограничение длины ячейки (1 байт), и все биты переполнения будут отброшены: например, если продол­ жить ввод и после цифры 9 ввести цифру 7, то после суммирования получим 297 (29 • 10 + 7), что превышает максимально допустимое 255, и после отбрасывания битов переполнения в ячейке-накопите­

ле кода останется двоичный эквивалент числа 42.

Таким образом, пока нажата левая клавиша *Alt,* в указанной ячейке будет «накапливаться» числовой код, который после отпу­ скания клавиши *Alt* будет записан в очередную ячейку буфера кла­ виатуры в качестве АSСП-кода символа, при этом байт скан-кода в этой ячейке буфера получит нулевое значение.

##### Видеосистема персонаяьноrо компьютера

Если клавиатура ПК - это основное устройство ручного ввода данных, то главным устройством вывода информации, представлен­ ной в визуальной форме, удобной для восприятия органами зрения человека, безусловно, является *дисплей,* часто называемый *видеомо­ нитором.*

**182**

Экран дисплея является исполнительным устройством *видеоси­ стемы* ПК, включающей комплекс программно-аппаратных средств и структур данных, обеспечивающих визуализацию информации, хранящейся в памяти компьютера в кодированной форме.

Основными техническими характеристиками видеосистемы ПК, определяющими качество экранного изображения, являются *раз­ решающая способность,* определяемая числом строк и столбцов точек (пикселей), выводимых на экран дисплея (для текстовых ви­ деорежимов разрешающая способность оценивается также числом строк и столбцов символов на экране и размерами матрицы пиксе­ лей, используемой для изображения одного символа), и *цветность,* определяемая количеством цветов (или оттенков серого цвета) изображения, выводимого на экран. В текстовых режимах работы видеосистемы программируются цвет символа и цвет фона для каж­ дого знакоместа, в графических - цвет каждого пикселя.

* + 1. **Аппаратный комплекс**

*Аппаратный комплекс* видеосистемы включает устройство ото­ бражения информации (дисплей) и видеоадаптер- программи­ руемое устройство, управляющее дисплеем и обеспечивающее ин­ терфейс с центральным процессором компьютера. С точки зрения программиста существенным является тип установленного видео­ адаптера, режимы его работы, состав и назначение программируе­ мых и информационных регистров.

Первые ПК серии IВМ РС комплектовались монохромным дис­ плеем и видеоадаптером *MDA (Monochrome Display Adapter),* кото­ рый работал только в текстовом режиме и имел разрешающую спо­ собность 640 х 350 пикселей. Позднее фирмой Hercules Computer Technology был разработан видеоадаптер *Hercules Graphic Card,* который, так же, как и MDA, являлся монохромным (двухцвет­ ным), однако имел более высокую разрешающую способность 720 х 350 пикселей и, что более существенно, обладал возможно­ стью отображения графической информации.

История применения цветного экранного изображения в IВМ­ совместимых ПК начинается с видеоадаптера *CGA* (Color Graphic Array), который обеспечивал работу в текстовом и графическом режимах с 16-цветным изображением. При этом разрешающая способность CGA была весьма низкой (экран дисплея - 320 х 200, знакоместо символа - 8 х 8 пикселей) даже по сравнению с MDA и Hercules, что объясняется недостаточным для цветного изображе­ ния объемом видеопамяти (64 Кбайт), используемой этим видео­ адаптером.

Позднее фирмой IВМ были выпущены более совершенные видеоадаптеры *EGA* (Enhanced Graphic Array), *VGA* (Video Graphic Array) и *XGA* (eXtended Graphic Array) с улучшенными технически-

183

ми характеристиками (видеопамять - до 1 Мбайт, разрешение - до 1024 х 768, отображение до 65 536 цветов).

К настоящему времени различными фирмами - производите­ лями выпущено большое количество типов видеоадаптеров, су­ щественно превосходящих VGA по своим техническим характе­ ристикам и получивших статус самостоятельных вычислителей, соизмеримых по мощности с центральными процессорами ПК. Эти видеоадаптеры принято объединять общим названием *SVGA* (Super *VGA),* и режимы их работы могут существенно отличаться от стан­ дартных видеорежимов (табл. 7.4), разработанных в свое время фирмой IВМ.

*Таблицаl.4*

**Характеристики стандартных режимов работы видеоадаптеров**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **№**  **режима** | **Тип** | **Разрешение** | **Число цветов** | **Видео- адаптеры** | **Начальный адрес** |
| О, О\*, О+ | Текстовый | 40 х 25 зн/м | 16 (п/т) | CGA, EGA, VGA | [В800:ОООО]ь |
| 1, 1\*, 1+ | Текстовый | 40 х 25 зн/м | 16 |
| 2, 2\*, 2+ | Текстовый | 80 х 25 зн/м | 16 (п/т) |
| 3, 3\*, 3+ | Текстовый | 80 х 25 зн/м | 16 |
| 4 | Графиче- ский | 320 х 200 пкс. | 4 |
| 5 | Графиче- ский | 320 х 200 пкс. | 4(п/т) |
| 6 | Графиче- ский | 640 х 200 пкс. | 2 |
| 7 | Текстовый | 80 х 25 зн/м | 2 | MDA, EGA,VGA | [В000:ОООО]ь |
| 8-0Ch | Резерв | | | | |
| 0Dh | Графиче- ский | 320 х 200 пкс. | 16 | EGA, VGA | [АООО:ОООО]ь |
| 0Eh | Графиче- ский | 640 х 200 пкс. | 16 |
| 0Fh | Графиче- ский | 640 х 350 пкс. | 4 |
| 10h | Графиче- ский | 640 х 350 пкс. | 16 |
| llh | Графиче- ский | 640 х 480 пкс. | 2 | VGA |
| 12h | Графиче- ский | 640 х 480 пкс. | 16 |
| 13h | Графиче- ский | 320 х 200 пкс. | 256 |

**184**

*Примечания.*

1. Режимы О, 2 и 5 являются режимами с подавлением цвета. В этих режи­ мах вместо цветного выводится полутоновое изображение с заменой множе­ ства цветов на такое же количество оттенков серого цвета.
2. Разрешающая способность видеосистемы в текстовых режимах оценива­ ется количеством столбцов и строк знакомест на экране и размерами матрицы пикселей, описывающей одно знакоместо: для CGA - 8 х 8, для EGA - 8 х 14, дляVGА-9 х 16.
3. Режимы О\*, 1 \*, 2\* и 3\* (EGA) - это аналоги текстовых режимов 0--3 CGA, отличающиеся размерами матрицы описания символов (8 х 14).
4. Режимы О+, 1+, 2+ и 3+ (VGA) - это аналоги текстовых режимов 0--3 CGA, отличающиеся размерами матрицы описания символов (9 х 16).
   * 1. Программное обеспечение

Базовое *программное обеспечение* видеосистемы составляет на­ бор функций прерывания BIOS INTlOь, обеспечивающих управле­ ние видеоадаптерами в стандартных режимах. Перечень некоторых функций с краткими комментариями по их применению приведен в табл. 7.5.

*Таблицаl.5*

**Видеофункции прерывания INТ10ь**

|  |  |  |  |
| --- | --- | --- | --- |
| **Функ-**  **ции** | **Назначе- ние** | **Вход** | **Комментарии** |
| ООь | Выбор режима работы видео- адаптера | АН= ООь  AL = номер видеоре- жима | - |
| Оlь | Изме- нение размеров курсора | АН= Оlь  СН и СН - верхняя и нижняя границы курсора | СН: биты 0--3 задают положе- ние верхней границы курсора (О - 15); биты 4-5 задают тип курсора: 00 - обычный, 01 - невидимый, 10 - мигаю- щий, 11 - быстро мигающий |
| 02ь | Измене- ние поло- жения курсора | АН= 02ь  ВН = номер страницы DH = номер строки DL = номер столбца | При активизации страницы курсор устанавливается в за- данную ПОЗИЦИЮ |
| 09ь | Запись символов | АН= 09ь  AL = АSСП-код сим- вола  ВН = номер страницы BL = байт атрибутов СХ = количество сим- волов | Записываются одинаковые сим- волы с одинаковыми атрибута- ми в заданную страницу видео- памяти. Запись производится  с позиции, заданной текущим положением курсора. После за- вершения операции положение курсора не изменяется |

**185**

В листинге 7.3 приведен фрагмент ассемблерной программы, вы­ зывающей функцию № ООь 10-го прерывания для выбора 4-го ре­ жима работы видеоадаптера (номер функции записывается в стар­ ший байт АН регистра А процессора, номер режима записывается в младший байт AL этого же регистра и затем вызывается програм­ ма обработки 10-го прерывания):

**листинг 7.3. Пример вызова видеофункции**

mov АН,0; mov AL,4; int 10h

* + 1. **Структуры данных**

Основной структурой данных, обслуживающей процесс выво­ да информации на экран дисплея, является так называемая *видео­ память,* выполняющая роль буфера обмена между программами, выполняемыми центральным процессором, и видеоадаптером, управляющим дисплеем. Функционирование видеосистемы обеспе­ чивается также дополнительными структурами данных (табл. 7.6), организованными в ОЗУ и ПЗУ.

*Таблицаl.6*

**Структуры данных, обслуживающих видеосистему ПК**

|  |  |  |
| --- | --- | --- |
| **Начальный адрес** | **Длина, байт** | **Назначение области памяти** |
| Таблица векторов прерываний | | |
| [ОООО:0040]ь | 4 | INT 10ь - указатель на видеофункции BIOS |
| [ОООО:007С]ь | 4 | INT lFь - указатель на таблицу знакогенератора для СИМВОЛОВ с кодами 128-255 |
| [0000:ОlОС]ь | 4 | INT 4Зь - указатель на таблицы знакогенератора для символов с кодами 0---255 (EGA/VGA) |
| Область данных BIOS | | |
| [0040:ООlО]ь | 1 | Флаги конфигурации. 5-й и 4-й биты определяют тип видеоадаптера: 00 - EGA; 01 - CGA 40х25; 10 - CGA 80х25; 11 - MDA |
| [0040:0049]ь | 1 | Номер текущего видеорежима |
| [0040:004А]ь | 2 | Число символов в строке |
| [0040:004С]h | 2 | Размер видеостраницы (в байтах) |
| [0040:004Е]ь | 2 | Начальный адрес активной видеостраницы (сме- щение в видеосегменте) |
| [0040:0050]ь | 8х2 | Координаты курсора в каждой из 8 страниц: млад- ший байт - номер калонки, старший номер строки |
| [0040:ООбО]ь | 2 | Размер (форма) курсора: младший байт - номер последней линии, старший байт - номер первой линии |

**186**

|  |  |  |
| --- | --- | --- |
| **Начальный адрес** | **Длина, байт** | **Назначение области** *пaWJl№HUe табл. l.t* |
| [0040:0062]ь | 1 | Номер активной страницы видеопамяти |
| [0040:0065]ь | 1 | Данные регистра режима CGA |
| [0040:0066]ь | 1 | Данные регистра цветовой палитры CGA |
| [0040:0084]ь | 1 | Число текстовых строк экрана минус единица |
| [0040:ООВS]ь | 2 | Высота символа в пикселях |
| [0040:0087]ь | 1 | 1-й байт данных о EGA. 6-й и 5-й биты опреде- ляют объем установленной видеопамяти: 00 - 64 Кбайт, 01 - 128 Кбайт, 10 - 192 Кбайт, 11 -  256 Кбайт |
| [0040:0088]ь | 1 | 2-й байт данных о EGA |
| [0040:ООАВ]ь | 2 | Адрес таблицы окружения минус единица |
| Видеопамять | | |
| [АООО:ОООО]ь | 64 Кбайт | Видеопамять в графических режимах EGA, VGA |
| [ВООО:ОООО]ь | 32 Кбайт | Видеопамять в монохромном текстовом режиме MDA |
| [В800:ОООО]ь | 32 Кбайт | Видеопамять в цветовых текстовых режимах и в графическом режиме CGA |
| ПЗУ | | |
| [СООО:ОООО]ь | 16 Кбайт | ROM BIOS EGA / VGA |

*Видеопамять* - основная структура данных, выполняющая функ­ ции буфера обмена между прикладными программами и видеоадап­ тером. Видеопамять логически расположена в основном адресном пространстве в диапазоне адресов с [A000:0000]h по [B000:FFFF]h (128 Кбайт). Физически видеопамять расположена на плате видео­ адаптера и может иметь объем, многократно превышающий 128 Кбайт. Видеопамять логически разделена на множество так называемых видеостраниц - последовательно расположенных об­ ластей, объем каждой из которых достаточен для представления информации, отображаемой на одном полном экране. Структура видеопамяти, ее объем и расположение определяются типом виде­ оадаптера и режимом его работы.

*Переменные ВIOS.* Для обслуживания видеосистемы используют­ ся часть *области данных ВIOS,* формируемой в процессе инициали­ зации (загрузки DOS). Знание адресов этих переменных позволя­ ет определять количество и типы установленных видеоадаптеров, режимы их работы, объем видеопамяти и ряд других параметров видеосистемы.

**187**

*Таблица окружения.* Содержит восемь адресов - указателей на различные таблицы и буфера данных, используемые функциями BIOS, обслуживающих видеосистему: таблицу параметров, область сохранения, вспомогательные таблицы символов для текстовых и графических видеорежимов. Если элемент таблицы окружения равен нулю, то соответствующий блок данных не используется. Указатель на таблицу окружения хранится в области данных ВIOS по адресу [0040:00A8]h.

*Таблицы знакогенераторов* - специальные структуры данных, располагающиеся в ОЗУ и ПЗУ и используемые видеоадаптерами в процессе формирования «точечных» образов символов на экране. Указатели на таблицы знакогенераторов - *векторы прерываний* INТ lFh и INT 43h. Различные программы (например, программы-ру­ сификаторы) могут загружать собственные знакогенераторы и пе­ реустанавливать на них соответствующие векторы прерываний.

* + 1. Кодирование данных в видеопамяти

В соответствии с установленным режимом работы видеосистемы в видеопамяти ПК организуется множество *страниц,* каждая из ко­ торых может использоваться для хранения образа полного экрана дисплея. Одна из страниц объявляется *активной,* и ее номер запи­ сывается в области данных BIOS по адресу [0040:0062]h.

Прикладная программа формирует образ экрана и, используя справочную информацию области данных BIOS (табл. 7.6), записы­ вает кодированные данные в активную страницу видеопамяти. Ви­ деоадаптер циклически считает данные активной видеостраницы, преобразует (декодирует) их и формирует соответствующие сигна­ лы управления дисплеем.

Размер видеостраницы определяется типом видеорежима, разре­ шающей способностью и числом отображаемых цветов, а количе­ ство страниц ограничивается объемом установленной видеопамя­ ти. Все видеостраницы пронумерованы, начиная с нулевой.

В текстовых и графических режимах используются различные системы кодирования элементов экрана, так как в первом случае в качестве элемента экрана используется *знакоместо,* а во вто­ ром - *пикселъ.*

Кодирование данных в текстовых видеорежимах

Знакоместо - это прямоугольная область экрана (8 х 8, 8 х 14 или 8 х 16 пикселей в зависимости от типа видеоадаптера), внутри которой точечно отображается символ, АSСП-код которого записан в видеопамять по соответствующему адресу.

Каждое знакоместо экрана кодируется двухбайтовым машинным словом, в котором младший байт - это *байт символа,* а старший - *байт атрибута.* Байт символа содержит АSСП-код символа, выво-

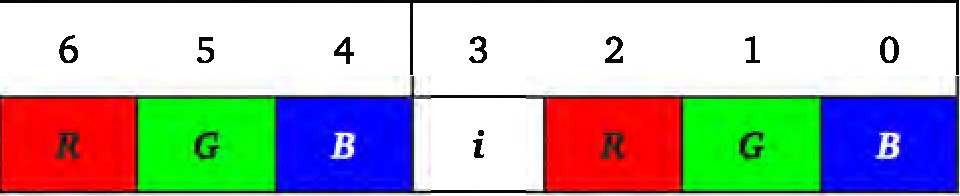
**188**

димого на экран, а байт атрибута задает цвет символа и цвет фона. Однобайтовая система кодирования цвета позволяет описать 256

(28) различных цветовых образов кодируемого знакоместа: млад­ шие 4 бита кодируют цвет символа, а старшие 4 бита - цвет фона. Система кодирования цвета знакоместа (рис. 7.11) поддержива­

ет модель RGB (Red - *Green* - *Blue),* в которой для кодирования цвета отводится по 4 бита: 3 младших бита - цветовые, а стар­ ший бит - это бит интенсивности выбранного цвета. Такая модель позволяет закодировать 16 различных цветов символа и столько же цветов фона: 8 различных комбинаций красного, зеленого и си­ него цветов, смешанных в равных пропорциях пониженной яркости (при нулевом значении бита интенсивности), и столько же соответ­ ствующих ярких цветов (при единичном значении бита интенсив­ ности). 7-й бит байта атрибута может «отвечать» либо за интенсив­ ность фона *(i),* либо за его мигание *(Ь),* в зависимости от установки параметров видеоадаптера.

Цвет фона Цвет символа



7

*i/Ь*

*Рис.7.11.* **Формат байта атрибута симвопа**

Такая система кодирования позволяет «пронумеровать» цвета символа и цвета фона целыми числами в диапазоне от О до 15, что используется во многих языках программирования при выводе сим­ волов на экран.

Пусть, например, переменные *Color* и *Background* (типа «целое без знака») получили значения *Color* = 12 и *Background* = 9, тогда переменная *Attrib* (того же типа), представляющая атрибут симво­ ла, записываемый в видеопамять, получит значение *Attrib* = *Color* +

+ 16 • *Background* = 156. В результате в соответствующем знакоме­ сте экрана будет отображен символ ярко-красного цвета *(Color* = 2 =

= 111002) на синем мигающем фоне *(Background* = 9 = 10012).

Размер видеостраницы определяется общим количеством зна­ комест на экране: например, для стандартных видеорежимов № 2 и № 3 (см. табл. 7.4) потребуется видеостраница размером 4000 бай­ тов (25 строк и 80 столбцов знакомест, по 2 байта на знакоместо). Фактически под видеостраницу отводится 4 Кбайт (4096 байтов, или lОООь в шестнадцатеричной системе счисления), из которых последние 96 байтов не используются.

Для текстовых видеорежимов (см. табл. 7.4) выделено всего 32 Кбайт видеопамяти (в сегменте В800ь для СGА-совместимых ви-

189

деорежимов и в сегменте ВОООь - для MDA), что позволяет орга­ низовать до 8 видеостраниц (для стандартных видеорежимов № 2,

№ 3 и № 7, и до 16 страниц - для режимов № О и № 1).

Так, для видеорежимов № 2 и № 3 нулевая страница будет за­ нимать диапазон адресов с [В800:ОООО]ь по [B800:0FFF]ь, и каждая последующая страница будет смещена на 4096 байтов относительно предыдущей: 1-я страница- с [В800:1000]ь по [B800:1FFF]ь, 2-я­ с [В800:2000]ь по [B800:2FFF]ь и т. д.

Зная размер видеостраниц в байтах *L* и номер видеостраницы *N,* можно определить ее начальный адрес (смещение от начала соот­ ветствующего сегмента видеопамяти): [N • L]ь.

При двухбайтовой системе кодирования адрес (смещение от на­ чала видеостраницы) машинного слова, представляющего в видео­ памяти некоторое знакоместо, вычисляется по формуле

2(у· *т+х),* (7.1)

где *х* иу- координаты знакоместа на экране (номер столбца и стро­ ки, которые отсчитываются слева направо и сверху вниз и нумеру­ ются с нуля), а *m* - количество символов в строке, определяемое выбранным видеорежимом (см. табл. 7.4).

Все эти данные ***(N, L, т,*** координаты курсорах иу, а также номер видеорежима, определяющий номер сегмента памяти, выделенный видеоадаптеру) хранятся в соответствующих *переменных BIOS* (см. табл. 7.6).

Процесс вывода символа на экран реализуется по следующей схеме:

1. программа (например, текстовый редактор или интерпре­ татор командной строки) обращается к области данных BIOS (см. табл. 7.6) и определяет номер активной страницы *N,* ее размер в байтах *L,* координаты курсорах и *у* и число символов в строке *т;*
2. по этим данным вычисляется адрес машинного слова, описы­ вающего в видеопамяти знакоместо, на которое указывает курсор. Например, для видеорежима № 2 и нулевой видеостраницы левое верхнее знакоместо экрана с координатами [0,0] отображается на ячейку видеопамяти с адресом [В800:ОООО]ь, а правое нижнее знакоместо с координатами [79,24] - [В800: ОF9Е]ь;

З) по вычисленному адресу программа записывает в видеопа­ мять байт символа и байт атрибутов (байт символа, содержащий его ASCII-код, до этого мог быть прочитан программой в буфере клавиатуры, а байт атрибута, описывающий цвета символа и фона, сгенерирован программой).

Байт символа (АSСП-код) используется далее видеоадаптером для вычисления адреса расположения соответствующей битовой матрицы в таблице знакогенератора (см. п. 7.3.5), а байт атрибу­ тов - для формирования сигнала управления цветом изображения символа.

**190**

**Кодирование данных в графических режимах**

В графических режимах (см. табл. 7.4) минимальной «единицей» экранного пространства является один пиксель (в отличие от зна­ коместа, описываемого матрицей пикселей в текстовых режимах), и единственным кодируемым параметром пикселя является его цвет. Размер блока видеопамяти, описывающего один пиксель, определяется количеством отображаемых цветов (или градаций серого цвета). Например, в монохромных (двухцветных) режимах

№ 6 и № llh для представления пикселя достаточно одного бита, в 4-цветных режимах № 4, № 5 и № 0Fh - двух битов, в 16-цветных режимах № 0D, № ОЕ, № 10h и № 12h - четырех битов, а в 256-цвет­ ном режиме № 13h для описания одного пикселя потребуется один байт.

Таким образом, одним байтом видеопамяти может быть описано от 1 до 8 пикселей, причем старшие биты байта отвечают за «ле­ вые» пиксели, а младшие - за «правые». Например, цвет левого верхнего пикселя экрана (с координатами [О,О]) кодируется следу­ ющими битами нулевого байта видеостраницы:

* в режимах № 6 и llh - старшим битом нулевого байта;
* в режимах № 4 и № 5 - двумя старшими битами нулевого байта;
* в режиме № 13h - всеми восемью битами нулевого байта. Двоичное число, описывающее пиксель в графическом режи­

ме - это числовой код, определяющий «номер» цвета пикселя в со­ ответствии с загруженной в видеоадаптер цветовой палитрой.

Размер видеостраницы в графических видеорежимах определя­ ется разрешающей способностью (количеством пикселей на экра­ не) и размером блока, описывающего один пиксель. Например, в режимах № 4 и № 5 (4 цвета, разрешение 320 х 200) и в режиме

№ 6 (2 цвета, разрешение 640 х 200) для одной видеостраницы по­ требуется 16 ООО байт (фактически - 16 Кбайт). В этих режимах объема видеопамяти достаточно для размещения двух страниц, од­ нако адаптеры CGA, EGA и VGA поддерживают одну страницу с на­ чальным адресом В800:0000. В режиме № 13h (256 цветов, разреше­ ние 320 х 200) одна страница занимает 64 Кбайт, т. е. целый банк видеопамяти, располагающийся с адреса АООО:0000.

* + 1. Знакогенераторы

Знакогенератор (или таблица знакогенератора) - это специаль­ ная структура данных, описывающая растровое (точечное) изобра­ жение символа и используемая видеоадаптером для преобразования АSСП-кода символа в последовательность сигналов, управляющих состоянием пикселей. Графический образ каждого символа опи­ сывается битовой матрицей, размерность которой определяется разрешающей способностью видеосистемы и различна для разных

**191**

видеоадаптеров (CGA - 8 х 8, EGA - 8 х 14, VGA - 8 х 16). Каж­ дый узел матрицы описывается одним битом, при этом единичное значение бита соответствует активному (светлому) пикселю, а ну­ левое - пассивному (темному).

Иллюстрация системы числового двоичного кодирования графиче­ ских образов символа для видеоадаптера CGA приведена на рис. 7.12. Как видно из рисунка, для кодирования образа одного символа ма­ трицей 8 х 8 требуется блок памяти размером в 8 байтов - по одно­ му байту на каждую строку матрицы. При использовании матрицы 8 х 14 (EGA) потребуется 14 последовательных байтов, а при исполь­ зовании матрицы 8 х 16 (VGA) - 16 байтов (по одному байту на каж­ дую строку матрицы). Количество строк пикселей *k,* используемых для представления символа в знакоместе, хранится в соответствую­ щей ячейке области данных BIOS ([0040:00SS]ь, см. табл. 7.6).

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Образ символа | | | | | | | | Коды строк матрицы | |
|  |  |  |  |  |  |  |  | 0000 0000 | ООь |
|  |  |  |  |  |  |  |  | 00001110 | ОЕь |
|  |  |  |  |  |  |  |  | 00010010 | 12ь |
|  |  |  |  |  |  |  |  | 0010 0010 | 22ь |
|  |  |  |  |  |  |  |  | 00111110 | ЗЕь |
|  |  |  |  |  |  |  |  | 0010 0010 | 22ь |
|  |  |  |  |  |  |  |  | 0010 0010 | 22ь |
|  |  |  |  |  |  |  |  | 0010 0010 | 22ь |

о

1

2

3

4

5

6

7

7 6 5 4 3 2 1 о

*Рис.7.12.* **Представление симвоnа «А» битовой матрицей 8 х 8**

В таблице знакогенератора блоки описаний символов располо­ жены линейно в порядке возрастания их АSСП-кодов, что позволяет вычислять начальные адреса блоков (смещение относительно нача­ ла таблицы знакогенератора) по простой формуле: *k* х АSСП-код, где *k* - количество байтов, отводимых в таблице знакогенератора для описания одного символа.

**192**

Начальный адрес таблицы знакогенератора EGA/VGA для симво­ лов с кодами от О до 255 хранится в ROM BIOS и доступен по век­ тору прерывания 43ь, EGA поддерживает 4 различных таблицы, а VGA - 8 таблиц, при этом одновременно могут быть активны две любые таблицы, что позволяет отображать на экране до 512 различ­ ных символов. Номера активных таблиц определяются содержимым регистра выбора знакогенератора видеоадаптера.

Начальный адрес таблицы знакогенератора для 128 символов

«национальных алфавитов» с АSСП-кодами от 128 до 255 досту­ пен по вектору прерывания lFь, что используется программами­

«русификаторами», которые формируют битовые матрицы для каждого символа соответствующего алфавита, записывают их в опре­ деленную область ОЗУ и соответственно переустанавливают вектор прерывания lFь,

BIOS EGA/VGA содержит специальную функцию, управляющую загрузкой шрифтов (функция llь прерывания INT lОь),

##### Контрольные вопросы

Система обработки прерываний

1. Что обозначают и вкаких отношениях находятся следующие понятия: а) прерывание, программное прерывание, аппаратные прерывания? б) IRQ-линия прерывания, номер прерывания, вектор прерывания?

в) сигнал запроса прерывания, приоритет прерывания, маскирование прерывания?

1. Какие основные функции выполняет контроллер прерываний?
2. Какую функцию выполняет контроллер прерываний?
3. Каково назначение регистров IRR, IMR и ISR контроллера прерываний Intel 8259А (см. рис. 7.1)?
4. Каково назначение линий INT и INTA микропроцессора Intel 8086 и линий INТR и INТA контроллера прерываний (см. рис. 4.3 и 7.1)?
5. Какие из перечисленных прерываний являются программными, ка­ кие - аппаратными, а какие - фактически не являются «прерываниями»: INТ ООь, INТ 08ь, INT 09ь, INT 10ь, INT lбь, INT lFь, INT 21ь, INT 43ь?
6. Какую информацию содержит таблица векторов прерываний и какова структура этой таблицы?

Клавиатура персонального компьютера

1. Какие функции выполняет контроллер клавиатуры?
2. С какими компонентами ПК взаимодействует контроллер клавиатуры и с какими целями?
3. Какая информация закодирована скан-кодом клавиши?
4. Какие манипуляции с клавишами не изменяют состояния флагов клавиатуры, а какие - состояния буфера клавиатуры?
5. Как программно идентифицировать ситуации «буфер клавиатуры переполнен» и «буфер клавиатуры пуст»?
6. Всегда ли пуст «пустой» буфер клавиатуры?

**193**

1. В каком формате хранятся указатели «головы» и «хвоста» буфера клавиатуры? Какие программы должны считывать и (или) модифицировать значения этих указателей и с какими целями?

Видеосистема персональноrо компьютера

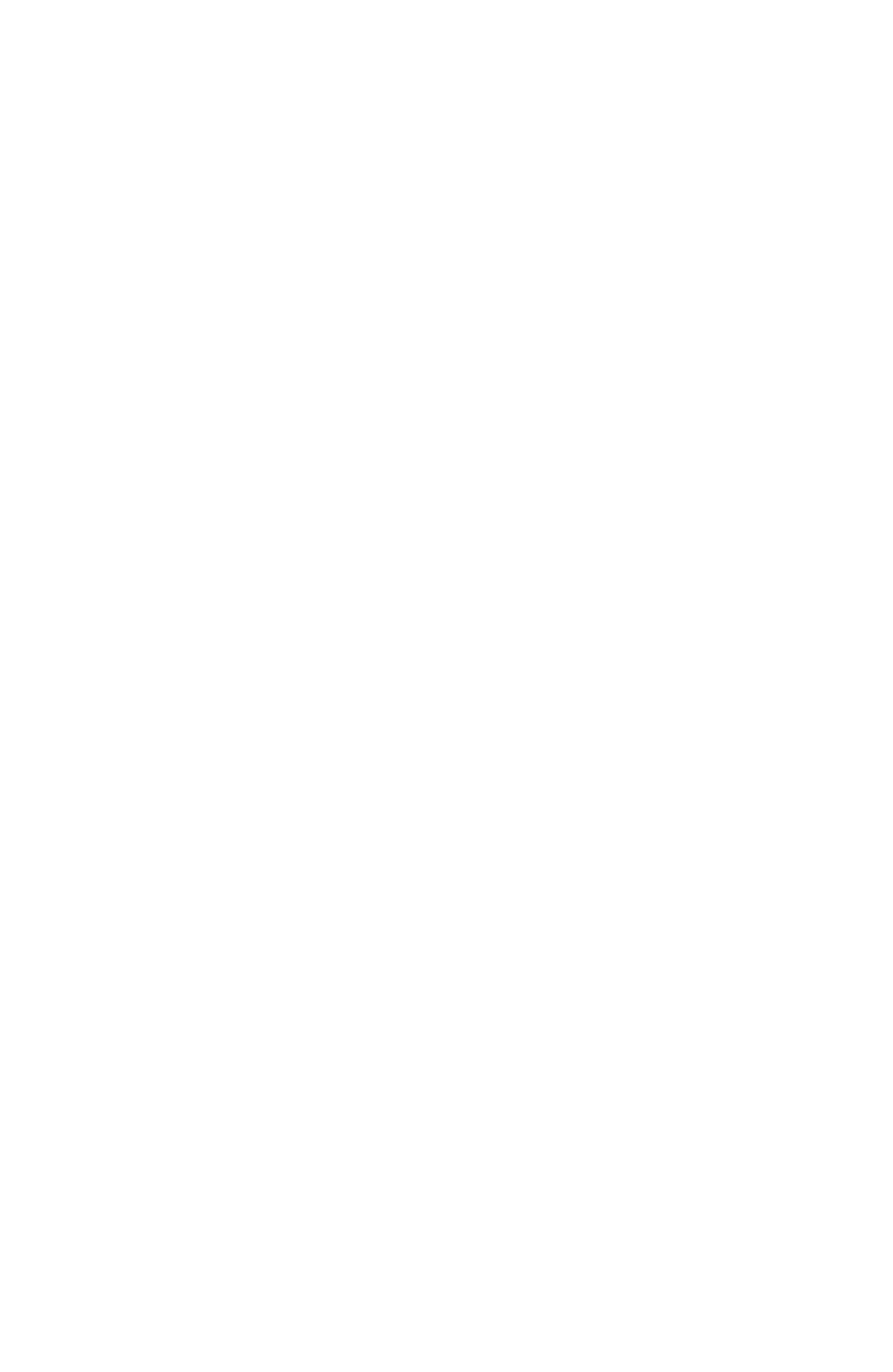
1. Какие аппаратные устройства обеспечивают работу видеосистемы ПК?
2. Какие программные компоненты обеспечивают работу видеосистемы ПК?
3. Какие структуры данных обеспечивают работу видеосистемы ПК?
4. В какой обрасти памяти ПК хранится информация о параметрах ис­ пользуемого видеорежима? Перечислите эти параметры.
5. В какой области базового адресного пространства ПК расположена видеопамять и каков ее размер?
6. Чем принципиально отличаются текстовые режимы работы видео­ системы от графических?
7. Каким десятичным числом должен быть представлен байт атрибута символа для его отображения синим цветом на красном фоне (см. рис. 7.11)?
8. Сколько байтов видеопамяти потребуется для кодирования одного знакоместа в 16-цветных текстовых видеорежимах?
9. Сколько байтов видеопамяти потребуется для кодирования одного пикселя в стандартном графическом видеорежиме № 13ь?
10. Что такое видеостраница и какой размер видеостраницы будет до­ статочным для стандартного текстового видеорежима № 3 и стандартного графического видеорежима № 13ь?
11. Что называют таблицей знакогенератора и для чего используется эта структура данных?
12. Какова структура одной ячейки таблицы знакогенератора?
13. Сколько байтов потребуется для хранения «пиксельного образа» одного символа в знакоместе размером 8 х 8 и 8 х 16?
14. Сколько байтов потребуется для хранения таблицы знакогенератора, описывающей все множество символов основной и дополнительной кодовых таблиц ASCII (знакоместо 8 х 16 пикселей)?
15. Сколько байтов потребуется для хранения таблицы знакогенератора, описывающей множество символов дополнительной кодовой таблицы ASCII (знакоместо 8 х 8 пикселей)?
16. Как определить адреса расположения таблиц знакогенераторов в па­ мяти ПК?

ПРАКТИКУМ

Сборник задач

с комментариями

и примерами решенииV



https://urait.ru

##### Общие методические указания

Программа изучения дисциплины предусматривает выполнение практических заданий разных уровней сложности по трем темати­ ческим разделам дисциплины:

* «Базовые понятия информатики и свойства информации» (тема 2);
* «Представление информации в компьютерных системах» (тема З);
* «Программно-аппаратное обеспечение ПК» (темы 4, 6, 7).

Учебный материал, минимально-необходимый для подготовки к выполнению практических заданий, изложен в соответствующих темах курса, в конце каждой темы приведен список контрольных вопросов, ответы на которые должны быть получены студентами до начала выполнения практических заданий.

Задания по первым двум тематическим разделам носят описа­ тельный и вычислительный характер, их выполнение требует зна­ ния определений базовых понятий информатики, ее арифметиче­ ских основ (двоичная и шестнадцатеричная системы счисления), а также методов двоичного кодирования и стандартных форматов представления текстовых и числовых данных в памяти компьютера. Задания третьего раздела носят в основном лабораторно-иссле­ довательский характер, для их выполнения потребуется установить на рабочий компьютер специальное программное обеспечение - виртуальную DОS-машину, программы-анализаторы содержимо­ го памяти ПК, электронный справочник по MS DOS (инструкция

по установке ПО приведена в приложении).

Для выполнения заданий повышенной сложности (помечены символом«\*»), как правило, потребуется самостоятельная разра­ ботка соответствующих компьютерных программ с использовани­ ем изучаемых студентами (или освоенных ими ранее) языков про­ граммирования.

Задания могут выполняться студентами как на аудиторных груп­ повых занятиях, так и в рамках самостоятельной работы - в по­ следнем случае задания выполняются в соответствии с индивиду­ альными вариантами, формируемыми преподавателем по каждой теме.

**197**

##### Задания по теме 2 «Информация»

**.1.** В базе данных информационной системы хранится 100 ООО до­ кументов. В ответ на свой запрос пользователь получил список из 100 документов, из которых 50 оказались релевантными запросу. Оцените релевантность системы при условии, что в ее базе данных содержится 500 документов, релевантных данному запросу.

* 1. В базе данных информационной системы хранится 100 ООО до­ кументов. В ответ на свой запрос пользователь получил список из 100 документов, из которых 50 оказались релевантными запросу. Оцените полноту системы при условии, что в ее базе данных содер­ жится 500 документов, релевантных данному запросу.
  2. В базе данных информационной системы хранится 500 ООО до­ кументов. В ответ на свой запрос пользователь получил список из 500 документов, из которых 50 оказались релевантными запросу. Оцените релевантность системы при условии, что в ее базе данных содержится 1000 документов, релевантных данному запросу.
  3. В базе данных информационной системы хранится 500 ООО до­ кументов. В ответ на свой запрос пользователь получил список из 500 документов, из которых 50 оказались релевантными запросу. Оцените полноту системы при условии, что в ее базе данных содер­ жится 1000 документов, релевантных данному запросу.
  4. В базе данных информационной системы хранится 50 ООО до­ кументов. В ответ на свой запрос пользователь получил список из 1000 документов, из которых 50 оказались релевантными за­ просу. Оцените релевантность системы при условии, что в ее базе данных содержится 500 документов, релевантных данному запросу.
  5. В базе данных информационной системы хранится 50 ООО до­ кументов. В ответ на свой запрос пользователь получил список из 1000 документов, из которых 50 оказались релевантными запро­ су. Оцените полноту системы при условии, что в ее базе данных со­ держится 500 документов, релевантных данному запросу.
  6. Приведите формулу для определения количества информа­ ции *I,* содержащейся в информационном сообщении, если известно количество *N* равновероятных ожидаемых вариантов сообщения. Рассчитайте по этой формуле количество битов информации, содер­ жащейся в сообщении об игральной карте, выбранной из колоды, содержащей 32 карты.
  7. Сколько вопросов достаточно задать собеседнику для вы­ яснения даты его рождения, при условии, что год рождения собе­ седника вам известен, все его ответы достоверны и представлены в бинарной форме: «да» или «нет»?
  8. Определите содержание информационной пирамиды DIКW для рассмотренных задач учета первичной информации о поездках

**198**

пассажиров, подготовки аналитических отчетов и принятия реше­ ний в автоматизированной системе анализа пассажиропотоков.

*Описание ситуации.* Несколько конкурирующих компаний пре­ доставляют услуги общественного транспорта жителям мегаполиса. Сформированы маршруты движения городских автобусов, в каждом автобусе установлены терминалы по продаже билетов (или другие устройства, регистрирующие факты оплаты поездок пассажира­ ми), все эти терминалы связаны компьютерной сетью и сохраняют в базе данных информацию о каждой поездке (маршрут, координа­ ты точки начала поездки, номер автобуса, дата и время начала по­ ездки, ее стоимость).

Транспортные компании используют информацию о выполнен­ ных поездках для решения оперативных задач финансовой отчет­ ности, при этом информация за прошлые периоды оказывается не­ востребованной и хранится в архивах «на всякий случай».

За многие годы работы в базах данных транспортных компаний накопилось огромное количество исторической информации, и, наконец, «всякий случай» наступил - Департамент транспорта го­ родской администрации принял решение о покупке у транспортных компаний этих баз данных для их использования при решении за­ дач оптимизации пассажиропотоков.

Были подготовлены следующие аналитические отчеты на основе полученной информации:

* средняя загрузка каждого маршрута (поездок за сутки);
* сравнительный анализ средней загруженности маршрутов по временам года;
* сравнительный анализ среднесуточной загруженности марш­ рутов по дням недели;
* сравнительный анализ среднечасовой загруженности марш­ рутов по времени суток.

По результатам проведенного анализа Департаментом транспор­ та были приняты следующие решения:

* допустить еще одну транспортную компанию для увеличения количества автобусов на следующих маршрутах (список прилага­ ется);
* увеличить количество автобусов на маршрутах (список при­ лагается) в дневное время по выходным и праздничным дням;
* проложить ветку метрополитена для связи центра города с районами (список районов прилагается), так как дальнейшее уве­ личение интенсивности движения наземного транспорта по этим маршрутам требует расширения улиц, а снос жилых домов обойдет­ ся городскому бюджету существенно дороже строительства метро.
  1. Определите содержание информационной пирамиды DIКW для задач оперативного учета продаж, маркетингового анализа

**199**

и принятия решений в автоматизированной системе маркетинго­ вого анализа в торгоме.

*Описание ситуации.* В торговом зале супермаркета устаномены кассовые аппараты, объединенные в локальную сеть и обеспечи­ вающие оперативную регистрацию в базе данных всех операций по продаже товаров покупателям супермаркета: категория товара, его наименование, цена и количество, дата и время покупки. Для постоянных клиентов (мадельцев дисконтных карт) дополнитель­ но указываются код клиента и текущий размер предоставленной ему торговой скидки.

В офисе супермаркета работают специалисты-маркетологи, фор­ мирующие следующие аналитические отчеты по результатам выпол­ нения соответствующих запросов к базе данных торгового учета:

* текущие складские запасы (по категориям товаров);
* объемы продаж за период времени (сутки, неделя, месяц, квартал, год) в денежном исчислении;

сравнительный анализ объемов продаж по категориям това-

ров;

почасовой анализ динамики продаж в течение суток; средняя доля продаж постоянным покупателям.

Управляющий супермаркетом анализирует содержание пред­ ставленных ему отчетов и после дополнительных консультаций с маркетологами и экономистами принимает следующие решения: пополнить складские запасы интенсивно продаваемых това-

ров;

установить 50%-ную скидку на цены товаров, срок реализа­

ции которых приближается к критическому;

* объявить акцию «У нас - всегда низкие цены!» (постоянная скидка 0,5 о/о на все товары ограниченного перечня категорий);
* объявить акцию: «Кто ходит в гости по утрам, тот экономит умно!» (скидка 10 о/о на все товары, купленные до 8:00);
* отменить дисконтные карты постоянным покупателям.
  1. Определите:

1. содержание элементов информационной пирамиды DIКW, необходимых для решения задач оперативного учета и анализа в ав­ томатизированной системе анализа успеваемости студентов, а так­ же для принятия решений № 1, 2 и З;
2. состав дополнительных данных, информации и знаний, не­ обходимых для реализации решения № 4.

*Описание ситуации.* Десять групп студентов-первокурсников Ин­ ститута компьютерных наук сдали экзамены по шести дисциплинам: математике, информатике, программированию, астрологии, хиро­ мантии и основам научного оккультизма. Преподаватели зафикси­ ровали в базе данных результаты сдачи экзаменов каждым из сту­ дентов по каждой дисциплине (оценки по 100-балльной шкале).

**200**

По окончании экзаменационной сессии для директора института были подготовлены печатные аналитические отчеты трех видов:

1. групповые экзаменационные ведомости по каждой из дисци­ плин;
2. рейтинговые списки студентов, ранжированные по среднему баллу студента по всем дисциплинам;
3. рейтинговые списки студенческих групп, ранжированные по среднему баллу всех студентов группы по всем дисциплинам.

По результатам анализа содержимого отчетов директор инсти­ тута принял следующие решения, оформленные соответствующими приказами.

1. Исключить экзаменационные ведомости из перечня предо­ ставляемых директору отчетов по причине их низкой информатив­ ности и большого объема неструктурированной информации, что затрудняет процесс принятия решений.
2. Назначить следующим студентам (список прилагается) по­ вышенный размер стипендии на следующий семестр за отличную успеваемость по всем дисциплинам.
3. Поощрить кураторов следующих групп (списки номеров групп и фамилии кураторов прилагаются) за высокие показатели академической успеваемости групп.
4. Провести дополнительное исследование причин плохой успе­ ваемости студентов по информатике, математике и программи­ рованию по сравнению с их высокими достижениями в освоении астрологии, хиромантии и основ научного оккультизма.

###### Задания по теме 3

**«Представление информации в компьютерных системах»**

Позиционные системы счисления

* 1. Даны равенства: *100k* х *100k* = *10000k* и *100k* + *100k* = *1000k.* В какой из k-ичных позиционных систем счисления оба ответа бу­ дут правильными?
  2. В какой из k-ичных позиционных систем счисления десятич­ ное число 1810 будет записано как 24k? Приведите формальное до­ казательство правильности своего ответа.
  3. В какой из k-ичных позиционных систем счисления десятич­ ное число 4410 будет записано как 24k? Приведите формальное до­ казательство правильности своего ответа.
  4. В какой из k-ичных позиционных систем счисления десятич­ ное число 2410 будет записано как 44k? Приведите формальное до­ казательство правильности своего ответа.

**201**

* 1. В какой из k-ичных позиционных систем счисления десятич­ ное число 14710 будет записано как *777k?* Приведите формальное доказательство правильности своего ответа.
  2. В какой из k-ичных позиционных систем счисления десятич­ ное число 24810 будет записано, как 888k? Приведите формальное доказательство правильности своего ответа.
  3. Какое из двоичных чисел соответствует (равно) шестнадца­ теричному числу АВСDь?
  4. Какое из шестнадцатеричных чисел соответствует (равно) двоичному числу 10000,ООlь?
  5. Представьте десятичное число 253 в двоичной системе счис­ ления.
  6. Представьте двоичное число 1010 1010ь в десятичной систе­ ме счисления.
  7. Представьте двоичное число 1011 lll0ь в шестнадцатерич­ ной системе счисления.
  8. Представьте десятичное число 4098 в шестнадцатеричной системе счисления.
  9. Представьте в двоичной системе счисления десятичное чис­ ло: а) 0,5; 6) 0,25; в) 7,125; г) 8,25.
  10. Представьте в шестнадцатеричной системе счисления деся­ тичное число: а) 4096,25; 6) 1024,5; в) 256,125; г) 16,25.
  11. Представьте шестнадцатеричное число FF,2ь в десятичной системе счисления.

Двоичное кодирование текстовых (символьных) данных

* 1. **\*.** Экспериментально исследуйте структуру двух кодовых та­ блиц (DOS СР866, Windows СР1251) и двух кодовых таблиц стан­ дарта Unicode (UТF-8 и UTF-16). Для проведения эксперимента ис­ пользуйте приложение Windows Far Manager (при его установке проконтролируйте наличие плагина FarТrans, при его отсутствии - установите).
* Используя встроенный текстовый редактор, создайте четыре небольших текстовых файла одинакового содержания (несколько коротких строк текста на русском и английском языках). При соз­ дании *(Shift* + F4)каждого файла выбирайте из меню соответствую­ щую символьную кодировку.
* Переведите редактируемый файл в режим просмотра (Fб - *view),* установите шестнадцатеричный формат просмотра (F4 - *Нех),* сравните коды одинаковых символов (в русских и английских строках текста) в разных кодировках.
* Определите управляющие коды, используемые для перевода строки.
* Откройте каждый из файлов стандартным Windows-peдaк­ тopoм (например, Блокнот или WordPad). Прокомментируйте ре-

**202**

зультаты. Какие кодировки символов поддерживаются этими редак­ торами?

* 1. **\*.** Напишите Руthоn-программу для определения кода тексто­ вого символа. Используя эту программу, экспериментально опреде­ лите диапазоны кодов, выделенные арабским цифрам, знакам пре­ пинания, строчным и прописным буквам английского и русского алфавитов. Эксперимент проведите для любых двух кодовых та­ блиц, поддерживаемых языком Python.
  2. **\*.** Предложите алгоритм преобразования строчных букв в прописные и обратно. Напишите программу (функцию), прини­ мающую символ и печатающую этот символ и его же в противопо­ ложном регистре.
  3. **\*.** Напишите Руthоn-программы для шифрования и дешиф­ рования текстовых сообщений.
     + Глубоко законспирированный агент Юстас, нелегально ра­ ботающий на территории стратегического противника, регулярно отправляет в Центр текстовые сообщения (на русском и (или) ан­ глийском языках) и в ответ получает из Центра от своего куратора благодарности и инструкции.
     + Все сообщения передаются в зашифрованном виде, причем ключ шифрования регулярно меняется, но всегда имеется и у Юста­ са, и у Центра.
     + В сообщении могут быть как строчные, так и прописные сим­ волы как русского, так и английского алфавитов.
     + И Юстас, и его куратор из Центра - высокообразованные люди, регулярно (и правильно) использующие в своих сообщениях как строчные, так и прописные буквы, поэтому в процессе шифро­ вания/дешифрования буквенных символов должны быть сохране­ ны алфавит (русская или английская буквы) и регистр (строчная или прописная буквы).
     + В качестве ключа используется целое число, которое при шифровании сообщения прибавляется к коду каждого буквенного символа.

*Вариант* № *1* - *простой ключ:* в качестве ключа Кеу использует­ ся целое число (в диапазоне от -128 до +128), одинаковое для всех символов шифруемого текста

*Вариант* № *2* - *сложный ключ:* для каждого *i-го* символа шифру­ емого текста используется индивидуальный ключ Key[i], значение которого совпадает с кодом Code *[i]* шифруемого символа

Двоичное кодирование целых десятичных чисел

* 1. Переменная, представляющая 1-байтовые целые числа без знака (натуральный ряд), получила значение некоторого десятич­ ного числа. Укажите, каким двоичным числом представлено это

**203**

значение в памяти ЭВМ, если десятичное число следующее: а) «1»;

6) «127»; в) «128»; г) «255»; д) «256».

* 1. Переменная, представляющая 1-байтовые целые числа со знаком, получила значение некоторого десятичного числа. Ука­ жите, каким двоичным числом представлено это значение в памяти ЭВМ, если десятичное число следующее: а) «+1»; 6) «-1»; в) «+127»;

г) «+128»; д) «-127»; е) «-128».

* 1. Однобайтовая ячейка памяти содержит двоичное число, эквивалентное некоторому шестнадцатеричному числу. Определи­ те значение записанного в эту ячейку десятичного числа при усло­ вии, что на эту ячейку отображается переменная, представляющая целые числа со знаком, если шестнадцатеричное число следующее: а) 10ь; 6) АОь; в) FFь; г) 81ь,
  2. Переменная *z* получает значения следующим оператором присваивания: *z* = *х* - *у,* где все три переменные имеют тип данных

«двухбайтовые целые со знаком». Как будут представлены в памяти компьютера значения всех операндов этого выражения (слагаемые **и** алгебраическая сумма), если:

1. *х* = 256, *у=* 1; 5) *х* = 512, *у=* 256;
2. *х=1,* у=256; 6)х=256, у=512;
3. *х* = 128, *у* = 127; 7) *х* = -256, *у* = -1;
4. *х* = 127, *у=* 128; 8) *х* = -256, *у=* -512.

**3.24\*.** В языке программирования Python определена операция побитной инверсии (~) двоичного представления целого десятич­ ного числа. Используя эту операцию, экспериментально подтвер­ дите (уточните или опровергните) следующее правило машинного представления целых чисел:

* если число положительно или равно нулю, старший (знако­ вый) бит содержит ноль, а младшие биты содержат значение модуля этого числа, представленное в прямом двоичном коде;
* если число отрицательно, старший (знаковый) бит содержит единицу, а младшие биты содержат значение модуля этого числа, представленное в дополнительном двоичном коде.

Результаты эксперимента и выводы приведите в отчете.

Двоичное кодирование вещественных десятичных чисел (стандарт IEEE 754)

* 1. Переменная типа float (32-битный формат стандарта IEEE- 754) получила значение: а) 32,5; 6) 0,0625 (1/16). Как будет представ­ лена в памяти компьютера нормализованная мантисса этого числа?
  2. Переменная типа float (32-битный формат стандарта IEEE- 754) получила значение: а) 32,5; 6) 0,0625 (1/16). Как будет пред­ ставлен в памяти компьютера порядок этого числа?
  3. Переменная типа float (32-битный формат стандарта IEEE- 754) получила значение: а) 32,5; 6) 0,0625 (1/16). Как будет пред­ ставлено в памяти компьютера это число?

**204**

* 1. Переменная типа float (32-битный формат стандарта IEEE- 754) получила значение отрицательного числа -32,5. Как будет представлено это число в памяти компьютера?
  2. Выполните задание 3.25: а) для 16-битного формата представления чисел; б) для 64-битного формата представления чисел.
  3. Выполните задание 3.26: а) для 16-битного формата представления чисел; б) для 64-битного формата представления чисел.
  4. Выполните задание 3.27: а) для 16-битного формата представления чисел; б) для 64-битного формата представления чисел.
  5. Выполните задание 3.28: а) для 16-битного формата представления чисел; б) для 64-битного формата представления чисел.
  6. Какое максимальное (по модулю) десятичное число можно пред­ ставить в следующих форматах стандарта IEEE-754: а) 16-бит; б) 32 бита; в) 64 бита?
  7. Какое минимальное (по модулю) десятичное число можно пред­ ставить в следующих форматах стандарта IEEE-754: а) 16-бит; б) 32 бита; в) 64 бита?
  8. С какой максимальной точностью (число знаков после запятой) можно представить десятичное число в следующих форматах стандарта IEEE-754: а) 16-бит; б) 32 бита; в) 64 бита?

###### Задания по теме 4

**«Адресное пространство персональноrо компьютера»**

Расчетные задания

* 1. Определите максимально допустимый объем прямо-адресу- емой памяти ПК, имеющего:

а) 16-разрядную адресную шину;

6) 24-х разрядную адресную шину; в) 32-х разрядную адресную шину.

* 1. Определите базовый адрес:

а) О-го сегмента памяти;

6) 1-го сегмента памяти; в) 2-го сегмента памяти.

* 1. Определите базовый адрес:

а) А-го сегмента памяти;

6) В-го сегмента памяти; в) Е-го сегмента памяти.

* 1. Определите базовый адрес:

а) F-го сегмента памяти;

6) 10ь-го сегмента памяти. в) llь-го сегмента памяти.

* 1. Определите, какой линейный адрес соответствует указанно­ му сегментному адресу:

а) (1234:1234)ь;

1. (0040:ООlЕ)ь;

**205**

в) (В800:4002)ь;

г) (F00F:F00F)ь.

* 1. Определите, какой сегментный адрес соответствует указан- ному линейному адресу (приведите три правильных ответа):

а) 0041Еь;

6) 0041Аь; в) 0041Вь; г) 0041Сь.

Лабораторно-исследователъские задания

Задания данного раздела выполняются на виртуальной DОS­ машине.

* 1. Исследование структуры области данных BIOS.

1. Используя справочник HELP, определите начальный и конеч­ ный адреса области данных BIOS и ответьте на вопросы:
   * в какой области памяти (RАМ или ROM) размещена эта струк­ тура данных;
   * сколько байтов она занимает в памяти ПК;
   * какая информация хранится в ячейках памяти с этими адре- сами.
2. Определите состав параллельных и последовательных адап­ теров (LPT\* и СОМ\*), установленных в вашем «виртуальном» ПК, и базовые адреса этих адаптеров. Запишите (и сохраните в отчете) базовые адреса в двоичной и шестнадцатеричной системах счисле­ **ния.**
3. Определите содержимое старшего и младшего байтов двухбай­ тового машинного слова, расположенного по адресу [0040:ООlС]ь. За­ пишите это машинное слово в шестнадцатеричной и двоичной систе­ мах счисления.
4. Повторите предыдущее задание для машинного слова, распо­ ложенного по адресу [ОООО:041С]ь- Сравните и прокомментируйте результаты.
5. Нажимая (не менее 16 раз) на любую из символьных клавиш клавиатуры, проследите за изменениями машинного слова по адре­ су [0040:ООlС]ь. Какая информация содержится в этом машинном слове?
   1. Исследование структуры таблицы векторов прерываний.
6. Используя справочник HELP, определите начальный и конеч­ ный адреса таблицы векторов прерываний и ответьте на вопросы:
   * в какой области памяти (RAM или ROM) размещена эта структура данных;
   * сколько байтов она занимает в памяти ПК;
   * какая информация хранится в ячейках памяти с этими адре- сами и в каком формате представлены соответствующие данные;
   * сколько байтов занимает в этой структуре данных один «век­ тор прерывания».

**206**

1. Определите сегментные адреса ячеек памяти, в которых хра­ нятся векторы О-го, 1-го, 8-го и 9-го прерываний.
2. Определите сегментные адреса ячеек памяти, в которых хра­ нятся векторы прерываний № lFь и № 4Зь.

##### Задания по теме 6

**«Файловые системы персонаnьного компьютера»**

Командный пользовательский интерфейс

*Методические указания.* Язык командной строки базируется на пользовательском представлении модели дискового простран­ ства ПК, в основе которого - множество иерархических (древо­ видных) структур вида «том - каталоги - файлы». Выполнение заданий 6.1 - 6.12 имеет целью изучение языка командной стро­ ки и приобретение практических навыков использования команд для выполнения типовых файловых операций. Выполнение зада­ ний 6.13 - 6.16 имеет целью изучение расширенного набора ко­ манд и освоение техники программирования пакетных (.bat) фай­ лов.

Все эти задания могут выполняться как в среде Windows (прило­ жение CMD.exe), так и в среде виртуальной DОS-машины.

* 1. Используя команду Help (и материал подтемы 6.2 курса), из­ учите назначение и синтаксис команд PROMPT, РАТН, DIR, MD, CD, REN, DEL, СОРУ и ТУРЕ.
  2. Используя команду PROMPT, установите вид приглашения: текущие дата и время;

версия Windows;

собственная фамилия и номер группы;

произвольный текст, заключенный в угловые скобки < >; восстановите стандартный вид приглашения (активный том

и путь к текущему каталогу).

* 1. Используя команду РАТН:
     + просмотрите установленный (по умолчанию) список путей к каталогам, содержащим исполнимые файлы;
     + «обнулите» установленный список путей к таким каталогам;
     + установите новый список путей к таким каталогам и прокон- тролируйте результаты проведенных изменений.
  2. Используя команду MD, создайте свой личный каталог с про­ извольным именем.
  3. Используя команду СОРУ, создайте в личном каталоге два коротких текстовых файла, содержащих по одной строке текста - ваши фамилия, имя и отчество на русском и английском языках.
  4. Используя команду REN, переименуйте созданные файлы.

**207**

* 1. Создайте в личном каталоге трехуровневую систему подчи­ ненных каталогов.
  2. Скопируйте в каждый из созданных каталогов под различны­ ми именами файлы, созданные при выполнении задания 6.5.
  3. Скопируйте одной командой файл из одного подчиненного каталога в другой подчиненный каталог того же уровня.
  4. Создайте в личном каталоге новый файл путем объедине­ ния двух файлов, созданных при выполнении задания 6.5.
  5. Используя команды СОРУ и ТУРЕ, просмотрите на экране содержимое всех файлов, созданных при выполнении предыдущих заданий.
  6. Установите текущим один из созданных каталогов и со­ храните его оглавление в файле Direct.txt, расположенном в этом каталоге.
  7. Сохраните оглавления личного каталога и всех подчинен­ ных ему каталогов в файле Му\_Dir.txt, расположенном в личном каталоге. Предложите несколько вариантов формирования такого файла.
  8. Используя программную оболочку Norton Commander (или любой ее аналог, например Dos Navigator, если вы работаете в вир­ туальной DОS-машине, или приложение Far, если вы работаете в ко­ мандной строке Windows):
     + отредактируйте файлы, созданные при выполнении зада­ ния 6.5 (например, дополните текст вашим домашним адресом);
     + выполните заданий 6.4 - 6.8 без прямого использования ко­ мандной строки.
  9. Подготовьте пакетные файлы, примеры которых приведены на листингах 6.1 - 6.7 (см. п. 6.2.6). Протестируйте их на различ­ ных наборах входных параметров, оцените работоспособность этих пакетных файлов.
  10. Используя команды Help и IF/?, изучите синтаксис коман­ ды IF... ELSE и подготовьте несколько примеров использования этой команды при программировании пакетных файлов. Прокомменти­ руйте 4-ю строку пакетного файла, представленного на листинге

6.5 (см. п. 6.2.6). Можно ли заменить эту строку на IF +%1 = = + GOTO Exit или на IF %1 == GOTO Exit без потери работоспособности

программы? Ответ подтвердите результатами эксперимента.

* 1. Подготовьте пакетный файл self\_cloning.bat, при выполне­ нии которого в текущем каталоге создается его копия под новым именем (имя указывается параметром команды). При этом соз­ данная копия файла должна оставаться способной к самоклониро­ ва- нию - т. е. к созданию своих собственных копий под разными именами.
  2. Оцените возможности пакетного файла Universal.bat, текст которого содержит единственную команду: %1 %2 %3. Как при-

**208**

менить такой пакетный файл для копирования и переименования файла, а также для удаления группы файлов заданного каталога? Попытайтесь найти и другие применения такому универсальному пакетному файлу.

Исследование алгоритмов выполнения типовых файловых операций

*Методические указания.* Исполнение команд управления фай­ лами, вводимых пользователем в командной строке (см. подте­ му 6.2 курса), осуществляется соответствующими системными функциями. При выполнении заданий 6.19 - 6.27 эксперименталь­ но исследуются алгоритмы, реализуемые такими функциями, а так­ же структуры данных, используемые этими алгоритмами Основная задача исследования - изучение модели данных, поддерживаемой файловыми FАТ-системами (см. подтему 6.3 курса). По результатам проведенных исследований требуется подтвердить, опровергнуть или уточнить описания алгоритмов выполнения основных файло­ вых операций, приведенные в п. 6.3.2 курса. Задания выполняются в среде виртуальной DОS-машины с использованием программы - анализатора дискового пространства DiskEdit.exe.

Задания 6.28 - 6.41 являются контрольными, их выполнение не требует проведения экспериментов, но требует понимания структур данных файловой системы и алгоритмов реализации ос­ новных файловых операций.

* 1. Исследование структуры таблицы разделов диска. Прямым доступом к таблице разделов (см. табл. 6.1) определите и сохраните в отчете следующие параметры для каждого из разделов:

аппаратные адреса начального и конечного секторов;

* + - количество секторов;
    - тип ОС и разрядность FAT.
  1. Исследование структуры загрузочного сектора тома. Про­ смотрите содержимое Ьооt-сектора (см. табл. 6.2) одного из томов виртуального ПК. Определите и сохраните в отчете:
     + параметры физического форматирования тома (количество рабочих поверхностей, цилиндров, секторов на дорожке, размер кластера);

количество секторов, выделенных FAT и корневому каталогу;

* + - размер (в байтах) таблицы параметров форматирования диска;
    - машинный код программы загрузки ОС.
  1. Исследование структуры корневого каталога. Просмотрите (командой DIR и программой Diskedit) оглавление корневого ката­ лога одного из томов. Определите и сохраните в отчете следующие параметры этого каталога:

размер каталога (в секторах и байтах);

физическое расположение (номера занятых каталогом секта-

ров);

**209**

* размер регистрационной записи о дочернем объекте каталога;
* количество дочерних объектов корневого каталога, в том числе файлов, подкаталогов, системных и скрытых файлов, меток томов;
* количество удаленных файлов.
  1. Исследование структуры байта атрибутов. Определите:
     + адрес (смещение в регистрационной записи о дочернем объ­ екте) байта атрибутов файла и порядковые номера битов для каж­ дого атрибута;
     + системные, скрытые и защищенные от модификации файлы корневого каталога, а также файлы, требующие архивирования при создании очередной резервной копии тома.

Установите для одного из текстовых файлов атрибут «только чтение» и экспериментально подтвердите повышение уровня защи­ щенности файла.

Установите для одного из текстовых файлов атрибут «является каталогом» и оцените последствия такого изменения.

Измените метку тома стандартными средствами (командой Label). Измените метку тома с помощью программы Diskedit и оце­ ните результат с помощью команды Vol. Предложите способ со­ крытия от (неквалифицированного) пользователя факта наличия на диске большой группы файлов.

* 1. Исследование структуры таблицы FAT. Определите и сохра­ ните в отчете:
     + номера секторов, занятых каждой копией FAT;
     + количество объектов (файлов и подчиненных каталогов) в томе;
     + общее количество кластеров на томе, в том числе свободных, занятых и «сбойных» кластеров;
     + объем (в байтах) свободного пространства на томе (подтвер­ дите ответ командой *Ctrl* + *L* Norton Commander).
  2. Исследование алгоритма создания подчиненных каталогов. Создайте (командой MD) подчиненный каталог и каталог, подчи­ ненный этому подчиненному каталогу; просмотрите оглавления созданных каталогов командой DIR и программой Diskedit. Опре­ делите и сохраните в отчете следующие параметры для каждого из двух созданных каталогов:

размер и начальные адреса (номера начальных кластеров);

* + - начальные адреса дочерних объектов и родительских катало­

гов.

Скопируйте несколько файлов в любой из подчиненных катало­ гов, созданных при выполнении этого задания, и ответьте на вопро­ сы (ответы подтвердите экспериментально):

* + - изменился ли размер (в кластерах) этого каталога?
    - сколько всего файлов можно зарегистрировать в этом катало- ге, чтобы сохранился его начальный (минимальный) размер?

**210**

* 1. Исследование алгоритмов удаления и восстановления фай­ лов. Просмотрите содержимое кластеров, занятых одним из файлов одного из каталогов; удалите (командой DEL) этот файл; повторно просмотрите содержимое этих же кластеров.

Прокомментируйте результат: что изменилось в таблице FAT и в родительском каталоге удаленного файла?

Оцените возможность восстановления удаленного файла и вос­ становите его с помощью программы Diskedit.

Удалите (командой DEL) группу из нескольких файлов одного каталога; оцените возможность восстановления удаленной группы файлов, попытайтесь восстановить эти файлы с помощью програм­ мы Diskedit.

* 1. Исследование алгоритмов копирования файлов. Выбери­ те по своему усмотрению два каталога одного тома, просмотрите их оглавления командой DIR и программой Diskedit;

Используя команду СОРУ (или FS в Norton Commander), скопи­ руйте файл одного из этих каталогов в другой. Что изменилось в ка­ талогах и таблице FAT этого тома?

Выполните операцию копирования файла между каталогами разных томов и сравните результаты этой операции с предыдущей.

Опишите алгоритмы копирования файлов между каталогами: а) каталоги размещены в одном томе;

б) каталоги размещены в разных томах.

* 1. Исследование алгоритмов перемещения файлов. Выбери­ те по своему усмотрению два каталога одного тома, просмотрите их оглавления командой DIR и программой Diskedit;

Используя команду F6 Norton Commander, переместите файл из одного из этих каталогов в другой. Что изменилось в каталогах и таблице FAT этого тома?

Выполните операцию перемещения файла между каталогами разных томов и сравните результаты этой операции с предыдущей.

Опишите алгоритмы перемещения файла между каталогами: а) каталоги размещены в одном томе;

б) каталоги размещены в разных томах.

* 1. Рассчитайте емкость (в секторах и мегабайтах) диска со следующими параметрами форматирования:

а) 16 рабочих поверхностей (головок), 1015 цилиндров, 63 сек­ тора на дорожке;

б) 2 рабочих поверхности (головки), 80 цилиндров, 36 сектора на дорожке.

* 1. Рассчитайте максимально-допустимое количество класте­ ров на томе, отформатированном в системе: а) FAT-12; б) FAT-16; в) FAT-32.
  2. Рассчитайте минимальный размер кластера (в байтах и секторах) на томе емкостью 4 Гб, отформатированном в системе: а) FAT-12; б) FAT-16; в) FAT-32.

**211**

* 1. Сколько секторов потребуется для хранения одной копии таблицы FAT в системной области тома со следующими параметра­ **ми:**

а) емкость ~ 2,88 Мб, FAT12, 1 кластер = 2 сектора;

6) емкость ~ 500 Мб, FAT16, 1 кластер = 16 секторов.

* 1. Рассчитайте размер системной области FАТ-тома со следу­ ющими параметрами:

а) емкость ~ 2,88 Мб, FAT12, 1 кластер = 2 сектора, корневой

каталог - 240 дочерних объектов;

6) емкость ~ 500 Мб, FATlб, 1 кластер = 16 секторов, корневой каталог- 512 дочерних объектов.

* 1. Рассчитайте максимально возможное количество файлов (с короткими DОS-именами) на FАТ-томе со следующими параме­ трами:

а) емкость ~ 2,88 Мб, FAT12, 1 кластер = 2 сектора, корневой

каталог - 240 дочерних объектов;

6) емкость ~ 500 Мб, FATlб, 1 кластер = 16 секторов, корневой каталог - 512 дочерних объектов.

* 1. Опишите пошаговый алгоритм выполнения файловой опе­ рации, используемой для определения количества «сбойных» кла­ стеров в рабочей области FАТ-тома.
  2. Опишите пошаговый алгоритм выполнения файловой операции, используемой для определения объема свободного про­ странства в рабочей области FАТ-тома.
  3. Опишите пошаговый алгоритм выполнения файловой опе­ рации, используемой для поиска файла (по его имени), зарегистри­ рованного в корневом каталоге FАТ-тома.
  4. Опишите пошаговый алгоритм выполнения файловой опе­ рации, используемой для поиска свободных кластеров, необходи­ мых для записи на FАТ-том файла заданного размера.
  5. Опишите пошаговый алгоритм выполнения файловой опе­ рации, используемой для создания на FАТ-томе подчиненного ката­ лога (команда MD).
  6. **\*.** Разработайте программную модель FАТ-таблицы, реализу­ ющую следующие алгоритмы:

а) определение количества «сбойных» кластеров в рабочей об­ ласти FАТ-тома;

6) определение объема свободного пространства в рабочей об­ ласти FАТ-тома.

в) определение общего количества файлов всех типов не томе; г) поиск свободных кластеров, необходимых для записи на FАТ­

том файла заданного размера.

* 1. **\*.** Разработайте программную модель каталога, реализую­ щую следующие алгоритмы:

а) поиск в каталоге файла по его имени;

6) переименование файла;

**212**

в) поиск в каталоге файла по номеру его начального кластера; г) поиск в каталоге записей обо всех удаленных файлах;

д) поиск в каталоге всех подчиненных ему каталогов.

* 1. **\*.** На основе программных моделей каталога и FАТ-таблицы (см. задания 6.39 и 6.40) разработайте программы, реализующие следующие алгоритмы:

а) создание подчиненного каталога;

6) получение списка номеров кластеров, занятых файлом ката­ лога;

в) получение списка имен файлов каталога, размещенных не­ фрагментарно;

г) удаление файла.

##### Задания по теме 7

**«Обмен данными с периферийными устройствами»**

Система обработки прерываний

*Методические указания.* Задания 7.1 - 7.6 выполняются в сре­ де виртуальной DОS-машины с использованием инструментальных программ-анализаторов памяти и электронного справочника HELP.

* 1. Используя электронный справочник HELP, определите назначе­ ние и использование следующих прерываний: INТ 00, INТ 08, INТ 09, INT lОь, INТ lбь, INТ lFь, INТ 21ь, INТ 43ь. Какие из перечисленных прерываний являются программными, какие - аппаратными?
  2. Используя одну из программ - анализаторов памяти (на­ пример, peek или fx\_show), определите начальные адреса программ обработки следующих прерываний: INT 00, INT 08, INT 09, INT lОь, INT lбь, INT 21ь. Определите программы, «перехватившие» указан­ ные выше прерывания.
  3. Определите состав программ, загруженных в память ПК, и номера прерываний, «перехваченных» этими программами.
  4. Определите начальный адрес программы РЕЕК.сот, просмотрите

машинный код этой программы, ответьте на следующие вопросы.

* + - Что происходит в момент загрузки этой программы в память ПК?
    - Что происходит в момент трехкратного нажатия на клавишу *Alt?*
    - Какие прерывания «перехвачены» программой Peek?
    - Измените первый байт первой машинной команды программы РЕЕК. com, прокомментируйте полученный результат.
  1. Определите начальный адрес программы RКEGA.com. просмотрите машинный код этой программы. Ответьте на следующие вопросы:
     + Какая роль отведена этой программе в процессе ввода данных с кла­ виатуры?
     + Как эта программа реагирует на одновременное нажатие двух клавиш

*Shift?*

* + - Какие прерывания «перехвачены» этой программой?

**213**

* 1. Определите назначение прерываний № lFь и № 43ь и началь­ ные адреса областей памяти, связанных с этими прерываниями.

Клавиатура персональвоrо компьютера

*Методические указания.* Задания 7.7 - 7.15 выполняются в сре­ де виртуальной DОS-машины с использованием инструменталь­ ных программ-анализаторов памяти. В процессе их выполнения продолжается экспериментальное исследование области данных ВIOS (в той ее части, которая используется обработчиком 9-го пре­ рывания и прикладными программами в процессе ввода данных с клавиатуры) и алгоритмов обмена данными между клавиату­ рой и прикладными программами, исполняемыми центральным процессором ПК. Исследуемые структуры данных представлены в табл. 7.1 и 7.3 темы 7 курса, иллюстрации соответствующих алго­ ритмов приведены на рис. 7.3 - 7.10.

Задания 7.16 и 7.17 являются контрольными, их выполнение не требует проведения экспериментов, но требует понимания алго­ ритмов реализации типовых процедур ввода данных с клавиатуры и соответствующих структур данных.

* 1. Исследуйте процесс изменения состояния флаговых байтов в процессе манипулирования управляющими клавишами *Shift, Ctrl, CapsLock, NumLock.* По результатам эксперимента определите соот­ ветствие битов двух флаговых байтов и управляющих клавиш. Ре­ зультаты представьте в шестнадцатеричном и двоичном форматах и сохраните в отчете.
  2. Установите режим «верхний регистр», не нажимая клавиши *CapsLock.* Попытайтесь отменить установленный режим с помощью этой клавиши. Проведите аналогичный эксперимент с клавишей *NumLock.* Прокомментируйте результаты эксперимента.
  3. Запишите адреса ячеек буфера клавиатуры, в которые будут записаны и из которых будут прочитаны скан-код и АSСП-код оче­ редной символьной клавиши. Экспериментально подтвердите пра­ вильность своих предположений.
  4. Введите символы «G», «g», «П» и «п», связанные с одной кла­ вишей. Определите скан-код этой клавиши и АSСП-коды введенных символов. Результаты представьте в шестнадцатеричной, десятич­ ной и двоичной системах счисления. Выполните аналогичные дей­ ствия для другой символьной клавиши. Результаты сохраните в от­ чете.
  5. Объясните, почему обработчик 9-го прерывания сгенери­ ровал четыре различных АSСП-кода символа по одному значению скан-кода клавиши.
  6. Введите символы «G», «g», «П» и «п», связанные с одной кла­ вишей, прямым набором их АSСП-кодов на цифровой клавиатуре

**214**

(при нажатой клавише *Alt).* Зафиксируйте промежуточные состоя­ ния ячейки с адресом [0040:0019]ь после ввода каждой очередной цифры АSСП-кода символа, а также состояние этой ячейки и соот­ ветствующих ячеек буфера клавиатуры после отпускания клавиши *Alt.* Опишите алгоритм ввода символа прямым набором его АSСП­ кода.

* 1. Введите прямым набором символ с АSСП-кодом «1234». Прокомментируйте результат ввода.
  2. Введите прямым набором управляющие символы (АSСП­ коды от О до 31). Используя электронный справочник HELP, опреде­ лите комбинации клавиш *(Ctrl* + клавиша), используемых для ввода управляющих символов.
  3. Используя электронный справочник HELP и технологию прямого набора кодов символов, составьте таблицу АSСП-кодов символов псевдографики.
  4. **\*.** Используя примеры, приведенные в подтеме 7.2 курса (см. рис. 7.5 - 7.10, листинги 7.1 и 7.2), проведите экспериментальное исследование алгоритма чтения-записи буфера клавиатуры. В отче­ те приведите исходный код написанной для этого программы и ком­ ментарии по результатам проведенного эксперимента.
  5. **\*.** Разработайте упрощенную программную модель буфера клавиатуры (хранение только АSСП-кодов) и реализуйте на этой модели следующие методы:

1. buf\_tail() - для определения указателя «хвоста» буфера кла­ виатуры;
2. buf\_head() - для определения указателя «головы» буфера клавиатуры;
3. buf\_full() - для определения состояния «буфер клавиатуры переполнен»;
4. buf\_empty() - для определения состояния «буфер клавиату­ ры пуст»;
5. buf\_write(code) - запись в буфер клавиатуры кода очередно­ го символа;
6. buf\_read() - чтение из буфера клавиатуры кода последнего введенного символа.

Видеосистема персонального компьютера

*Методические указания.* Задания 7.18 - 7.25 и 7.32 - 7.34 явля­ ются контрольными, остальные задания имеют экспериментальный характер, выполняются в среде виртуальной DОS-машины и имеют целью исследование структур данных, организуемых в видеопамяти ПК для реализации текстовых видеорежимов.

1. **18.** Определите цвет символа и цвет фона для следующих (деся­ тичных) значений байта атрибута (стандартный видеорежим № 3): а) 7; 6) 128; в) 156; г) 137; д) 255.

**215**

* 1. Определите значение байта атрибута (стандартный тексто­ вый видеорежим № 3) для отображения символа ярко-красным цве­ том на зеленом мигающем фоне и бледно-зеленым цветом на крас­ ном фоне.
  2. Определите понятие «видеостраница». Какой объем видео­ страницы необходим для реализации видеорежима № 3?
  3. Определите диапазон адресов ячеек видеопамяти для виде­ остраницы № 4 (стандартный видеорежим № 3).
  4. Определите адреса ячеек 2-й страницы видеопамяти, выде­ ленных для хранения символа в знакоместе экрана с координатами [20ь;10ь] (стандартный видеорежим № 3).
  5. Как программно изменить форму курсора и определить те­ кущее положение курсора на видеостранице с заданным номером?
  6. Для каких целей зарезервированы прерывания INТ\_lF и INT\_43h?
  7. Определите объем памяти, необходимый для хранения зна­ когенератора - дополнительная кодовая таблица (128 символов) в формате 8 х 8 пикселей.
  8. Определите объем памяти, необходимый для хранения знакогенератора - основная и дополнительная кодовые таблицы (256 символов) в формате 8 х 16 пикселей.
  9. Используя информацию области данных BIOS, определите и сохраните в отчете следующие параметры видеосистемы вашего виртуального ПК:

тип видеоадаптера; объем видеопамяти;

номер текущего видеорежима; число символов в строке;

размер видеостраницы (в байтах); начальный адрес активной видеостраницы;

координаты курсора для каждой из видеостраниц текущего текстового видеорежима.

* 1. Отобразите на экране область данных BIOS, установите ре­

жим редактирования памяти (FS в программе Peek.com). Как изме­ нилась форма курсора? Что (и по какой причине) изменилось в об­ ласти данных BIOS?

Перемещая курсор клавишами-стрелками, наблюдайте за изме­ нением координат курсора в каждой из видеостраниц. Какая про­ грамма вносит изменения в эти ячейки памяти? Какая программа

«рисует» на экране курсор «правильной» формы и в «правильном» месте экрана?

* 1. Выведите на экран образ активной видеостраницы. Про­

комментируйте результаты отображения и объясните систему ко­ дирования видеоданных в текстовом режиме.

**216**

* 1. Определите адрес машинного слова (байт символа и байт атрибутов), описывающего знакоместо экрана с заданными коор­ динатами *(х,* у) на активной видеостранице. Прочитайте (и сохра­ ните в отчете) машинное слово по этому адресу, убедитесь в том, что видеоадаптер отображает на экране символ, соответствующий прочитанному АSСП-коду и значению атрибута (цвета) символа.
  2. Используя векторы прерываний № lFь и № 43ь, определите расположение соответствующих таблиц знакогенератора.
  3. Исследуйте структуру одной из таблиц знакогенератора. По данным таблицы составьте битовую матрицу, описывающую символ с заданным АSСП-кодом (пример см. на рис. 7.12).
  4. **\*.** Создайте программную модель байта атрибута (стандарт­ ный видеорежим № 3) и реализуйте на базе этой модели следующие методы:

а) color(attrib) определение цвета символа по его атрибуту;

6) background(attrib) определение цвета фона по его атрибуту; в) attrib(color, background) для определения значения атрибута

по заданным цветам символа и фона.

* 1. **\*.** Создайте упрощенную программную модель таблицы зна­ когенератора (16 любых символов вашего персонального алфавита в формате 8 х 8 пикселей) и реализуйте на базе этой модели следу­ ющие методы:

а) write\_sign(code, matrix) для заполнения фрагмента знакоге­ нератора, соответствующего символу с кодом code, данными бито­ вой матрицы, представленными параметром matrix;

6) print\_sign(code) для отображения графического образа сим­ вола по его коду (подобно тому как это показано на рис. 7.12).

* 1. **\*.** Напишите программу для отображения графического обра­ за символа по его коду (см. рис. 7.12), использующую реальную та­ блицу знакогенератора, адресуемую вектором прерывания INТ 43ь.

##### Примеры решений

В данном разделе приведены образцы решений нескольких из вышеприведенных задач.

**2.6.** В какой из k-ичных позиционных систем счисления десятич­ ное число 24810 будет записано, как *888k?*

*Решение.* Согласно формуле (3.1), значение числа, заданного

в k-ичной позиционной системе счисления, определяется как сумма произведений базисных чисел *bi,* записанных в *i-x* позициях, на со­ ответствующие позиционные веса *ki.*

Значение k-ичного числа *888k* = 8 • *k2* + 8 • *k1* + 8 • *k0,* и для пере­ вода этого числа в десятичную систему счисления надо записать все операнды этого выражения в десятичной системе счисления и вы­

полнить в этой же системе все вычисления. В результате получим

**217**

следующее уравнение: 8 • *k2* + 8 • *k1* + 8 • *kO* = 24810, решением кото­ рого яRЛЯется *k* = 5.

*Ответ:* в пятеричной системе счисления (24810 = 8885).

**З.21в.** Переменная, представляющая 1-байтовые целые числа

со знаком, получила значение десятичного числа «+127». Каким двоичным числом представлено это значение в памяти ЭВМ?

*Решение.* Переменная получила положительное значение, следо­ вательно, старший бит двоичного кода, записанного в ячейку па­ мяти, равен «О», а младшие семь битов предстаRЛЯют модуль этого числа в прямом двоичном коде. Для получения прямого двоичного кода десятичного числа 127 воспользуемся правилом перевода деся­ тичных чисел в двоичную систему счисления и получим 1111111.

*Ответ:* в ячейке памяти записано двоичное число «0111 1111».

**З.21д.** Переменная, представляющая 1-байтовые целые числа со знаком, получила значение десятичного числа «-127». Каким двоичным числом представлено это значение в памяти ЭВМ?

*Решение.* Переменная получила отрицательное значение, следо­ вательно, старший бит двоичного кода, записанного в ячейку па­ мяти, равен «1», а младшие семь битов предстаRЛЯют модуль этого числа в дополнительном двоичном коде. Для получения дополни­ тельного двоичного кода десятичного числа 127 сначала получим его прямой код (переводом из десятичной в двоичную систему счис­ ления), затем из прямого кода получим обратный (заменой нулей на единицы, а единиц - на нули), а затем из обратного кода по­ лучим дополнительный (прибавлением единицы к обратному коду. Прямой двоичный код десятичного числа 127 - 1111111, обратный код - ООО 0000, дополнительный код модуля числа - ООО 0001.

*Ответ:* в ячейке памяти записано двоичное число «1000 0001».

**З.22в.** Однобайтовая ячейка памяти содержит двоичное число, эквивалентное шестнадцатеричному FFь. Определите значение за­ писанного в эту ячейку десятичного числа при условии, что на эту ячейку отображается переменная, представляющая целые числа со знаком.

*Решение.* FFь = 1111 llllь. Старший бит двоичного кода, запи­

санного в ячейку памяти, равен «1», следовательно, в этой ячейке закодировано отрицательное число, модуль которого (младшие семь битов) представлен в дополнительном двоичном коде. Для вос­ становления прямого кода модуля закодированного числа сначала получим из дополнительного кода обратный (вычитанием единицы из дополнительного), а затем из обратного кода получим прямой (заменой нулей на единицы, а единиц - на нули). Дополнительный двоичный код модуля числа - 1111111, обратный код- 1111110, прямой код - ООО 0001, что соответствует десятичному числу 1.

*Ответ:* в ячейке памяти закодировано десятичное число «-1».

**218**

**З.27а.** Переменная типа float (32-битный формат стандарта IEEE- 754) получила значение 32,5. Как будет представлено это число в памяти компьютера?

*Решение.* В соответствии со стандартом IEEE-754 (см. п. 3.4.3), для хранения двоичного кода десятичного числа в памяти будет выделено машинное слово длиной в 4 байта (32 бита): в старшем (31-м) бите хранится код знака числа (1 - для отрицательных чи­ сел, О - для положительных и нуля); 8 битов (с 23-го по 30-й) со­ держат значение порядка числа в так называемом смещенном фор­ мате (значение порядка +127); остальные 23 бита (с О-го по 22-й) содержат дробную часть нормализованной мантиссы.

Двоичное представление числа 32,510 = 100000,lь, Нормали­

зованная мантисса - 1,ОООООlь, ее дробная часть - ОООООlь, Порядок - 5 (101), смещенный порядок - 5 + 127 = 132 (101ь +

+ 0lllllllь = 10000100ь), Так как переменная получила положи­

тельное значение, 31-й бит будет равен О. Omвem:01000010000000100000000000000000.

* 1. **в.** Определите базовый адрес llь-го сегмента памяти.

*Решение.* Базовый адрес сегмента - это линейный (20-разряд- ный) адрес его начальной ячейки, вычисляется (см. формулу (4.2)) умножением 16-разрядного номера сегмента *CS* на двоичное число lООООь (операция сдвига на 4 разряда): llь • lОь = 110ь,

*Ответ:* базовый (20-разрядный) адрес llь-го сегмента 0000

0000 0001 0001 ООООь,

1. **Sв.** Определите, какой линейный адрес соответствует сегмент­ ному адресу (В800:4002)ь,

*Решение.* В соответствии с формулой (4.2) линейный адрес ра­ вен: В800ь • 10ь + 4002ь = ВСОО2ь,

*Ответ:* ВСОО2ь = 1011 1110 0000 0000 0010ь,

**4.бв.** Определите, какой сегментный адрес соответствует линей­ ному адресу 004lВь (приведите три правильных ответа).

*Решение.* Адресное пространство основной памяти (см. рис. 4.6) представлено множеством последовательно пронумерованных сег­ ментов размером по 64 Кб, расположенных со смещением в 16 байт относительно соседних сегментов. Учитывая тот факт, что размер сегмента 64 Кб, а смещение всего 16 байтов, одна и та же ячейка памяти (с заданным линейным адресом) может принадлежать до­ статочно большому количеству разных сегментов памяти. Формаль­ но это означает, что линейное уравнение вида (4.2) с двумя неиз­ вестными *CS* и *IP* имеет бесконечно много решений при заданном *LA* (с учетом ограничений размера сегмента «бесконечно много» превращается в 4096).

*Ответ:* 1) 0041Вь = 0000:041В; 2) 0041Вь = 0040:001В; 3) 0041Вь

= 0041:000В.

**219**

**6.29а.** Рассчитайте максимально-допустимое количество класте­ ров на томе, отформатированном в системе FAT-12.

*Решение.* Основное назначение таблицы FAT - числовое коди­ рование информации о кластерах рабочей области тома, при этом каждый элемент FAТ[i] описывает соответствующий (i-й) кластер, например содержит номер кластера, в котором продолжается файл, записанный в i-м кластере. 12-битным двоичным числом можно

представить одно из 212 = 4096 десятичных чисел, следовательно,

количество кластеров рабочей области тома не может превышать 4096. Однако не все эти числа используются для хранения номеров кластеров - имеется 18 служебных числовых кодов специального назначения (их состав и назначение уточните по табл. 6.4).

*Ответ:* 4096 - 18 = 4078 кластеров.

**6.306.** Рассчитайте минимальный размер кластера (в байтах и секторах) на томе емкостью 4 Гб, отформатированном в системе FAT-16.

*Решение.* Максимальное количество кластеров примем равным 216 = 65 536 (фактически чуть меньше, но для ориентировочного расчета примем это «круглое» число). Емкость тома примем рав­ ной 4 Гб = 232 байтов (реальная емкость рабочей области тома чуть меньше, но это мало скажется на результате нашего расчета). Оце­ ним минимальный размер кластера в байтах, разделив емкость тома на максимальное количество кластеров: 232: 216 = 216 = 65 536 бай­ тов. Размер сектора - 512 байтов, следовательно, размер кластера в секторах: 216: 29 = 27 = 128 секторов.

*Ответ:* При форматировании тома емкостью 4 Гб в систе­ ме FAT-16 размер кластера не может быть меньше 128 секторов (65 536 байтов).

**6.Зlа.** Сколько секторов потребуется для хранения одной копии таблицы FAT в системной области тома со следующими параметра­ ми: емкость ~ 2,88 Мб, FAT12, 1 кластер = 2 сектора.

*Решение.* Каждый элемент таблицы FAT представляет один кла­ стер рабочей области тома, следовательно, количество элементов FAT равно количеству кластеров. Оценим (ориентировочно) коли­ чество двухсекторных кластеров на томе размером 2880 Кб: раз­ мер кластера- 2 • 512 б = 1 Кб; количество кластеров на томе -

~ 2880 : 1 = 2880. Для хранения 2880 12-битных элементов FAT

потребуется 2880 • 1,5 = 4320 байтов = 8,44 сектора.

*Ответ:* Для хранения таблицы FAT в системной области тома по­ требуется 9 секторов.

**7.18г.** Определите цвет символа и цвет фона для следующих (десятичных) значений байта атрибута (стандартный видеорежим

№ 3): 137.

*Решение.* Стандартный текстовый видеорежим № 3 (см. табл. 7.4) является 16-цветным и поддерживает RGВ-модель двоичного коди-

**220**

рования цвета символа и цвета фона (см. рис. 7.11). Значение байта атрибута символа 137 = 1000 1001ь, Младшие 4 бита кодируют цвет

символа (голубого, повышенная яркость), старшие 4 бита кодируют цвет фона (ярко-черный, или бледно-черный мигающий в зависи­ мости от настройки видеоадаптера).

*Ответ:* символ ярко-синего цвета на ярко-черном (или бледно­ черном мигающем) фоне.

**7.21.** Определите диапазон адресов ячеек видеопамяти для виде­

остраницы № 4 (стандартный видеорежим № 3).

*Решение.* Видеостраница - это область видеопамяти, в которой хранится закодированный образ одного полного экрана. Для тек­ стовых видеорежимов размер видеостраницы определяется количе­ ством знакомест на экране и размером ячейки, в которой хранится образ одного знакоместа. Для стандартного видеорежима № 3 (см. табл. 7.4) на экране отображается 200 знакомест (25 • 80), каждое из которых кодируется двухбайтовым машинным словом (байт кода символа и байт кода цвета), что потребует 4000 байтов для хране­ ния всей видеостраницы. Фактически под каждую видеостраницу отводится по 4 Кб (4096 байтов, или 1000h в шестнадцатеричной системе счисления), из которых последние 96 байтов не использу­ ются.

Для текстового видеорежима № 3 в видеопамяти отведено 32 Кб, начиная с адреса [B800:0000]h (см. табл. 7.4), все видеостраницы последовательно пронумерованы (с нуля), 4-й видеостранице пред­ шествуют четыре страницы, вместе занимающие 16 Кб, следова­ тельно, начальный адрес 4-й видеостраницы будет [B800:4000]h. Так как фактический размер видеостраницы 4000 байтов (или FA0h в шестнадцатеричной системе счисления), конечный адрес 4-й ви­ деостраницы будет [B800:4FA0].

*Ответ:* 4-я страница видеопамяти (3-й стандартный виде­ орежим) занимает диапазон ячеек видеопамяти с [В800:4000] по [B800:4FA0].

**7.22.** Определите адреса ячеек 2-й страницы видеопамяти, выде­ ленных для хранения символа в знакоместе экрана с координатами [20h; 10h] (стандартный видеорежим № 3).

*Решение.* Начальный адрес 2-й видеостраницы для стандартного видеорежима № 3 - [В800:2000], каждое знакоместо кодируется двухбайтовым машинным словом (см. решение к заданию 7.21), а количество знакомест в строке экрана- 80 (S0h, см. табл. 7.4).

Смещение (в байтах) от начала этой страницы до двухбайтовой ячейки, описывающей знакоместо с координатами [20h; 10h], опре­ делим ПО формуле (7.1): 2 • (10h • 50h + 20h) = A40h.

*Ответ:* для хранения символа в знакоместе экрана с координа­

тами [20h; 10h] будут использованы две однобайтовые ячейки виде­ опамяти с адресами [В800:2А40] и [В800:2А41].

### *Приложение* Инструкция по установке виртуальной DОS-машины

1. Скопируйте файлы образов жесткого и гибкого дисков **dos. ova и floppy.img** на рабочий ПК.
2. Скачайте с официального ресурса современную версию уста­ новочного файла **Oracle VМ VirtualBox.**
3. Запустите установочный файл и дождитесь завершения уста­ новки виртуальной машины Oracle VМ VirtualBox на Ваш рабочий ПК (в процессе установки все опции принимайте по умолчанию, на все вопросы установщика отвечайте утвердительно: *Next, Yes).*
4. После завершения установки создайте на рабочем сто­ ле (или в другом удобном Вам месте) ярлык приложения Oracle VМ VirtualВox и запустите это приложение.
5. В меню *«Файл»* выберите *«Импорт»,* укажите путь к фай­ лу **dos.ova** на Вашем ПК и выберите этот файл. В результате будет импортирован образ жесткого диска (С:\), на котором уже будут установлены инструментальные средства, необходимые для выпол­ нения лабораторных работ.
6. В меню *«Настройки»* выберите пункт *«Носители»,* укажите путь к файлу **floppy.img** на Вашем ПК и выполните. В результате до­ полнительно будет подключен образ гибкого диска (А:\).

***DОS-машина на Вашем ПК готова к работе.***

Возможно, на Вашем ПК не установлен режим аппаратной под­ держки виртуализации. Для его установки (в случае использования ОС Windows 10) выполните следующие действия.

1. При загруженной ОС в главном меню «Пуск» найдите в раз- деле *«Параметры»* параметр *«Восстановление».*
2. Выберите *«Особые варианты загрузки».*
3. Далее *<<Перезапустить сейчас».*
4. После перезагрузки ОС:
   * выбор действия *«Диагностика»* (может быть названо *<<Поиск и устранение неисправностей»);*
   * *«Дополнительные параметры»* - *<<Параметры встроенного ПО UEFI»;*
   * *<<Перезапустить»;*
   * FlO - *System Configuration;*

**222**

* + выбрать из списка *«Virtualisation Technology»* и установить (клавишами FS/Fб) режим *ЕпаЫеd;*
  + выйти из режима настроек **с *сохранением изменений. Выполните эти настройки и затем повторно установите***

***Virtual Вох.***