МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

КУРГАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

В.К.Волк

ИССЛЕДОВАНИЕ ФУНКЦИОНАЛЬНОЙ СТРУКТУРЫ ПАМЯТИ ПЕРСОНАЛЬНОГО КОМПЬЮТЕРА

ЛАБОРАТОРНЫЙ ПРАКТИКУМ

Курган 2004

УДК 681.3.067(07)
В67

Рецензенты

Кафедра информатики и вычислительной техники Курганской государственной сельскохозяйственной академии (зав. кафедрой канд. экон. наук, доцент Голованова А.Х.) ; генеральный директор ОАО «Курганская информационная корпорация «Экспресс-Информ» Рымар В.С.

Печатается по решению методического совета Курганского государственного университета.

Научный редактор – канд.физ.-мат.наук, профессор Симахин В.А.

В67 Волк В.К. Исследование функциональной структуры памяти персонального компьютера. Лабораторный практикум.: Учебное пособие. – Курган: изд-во Курганского гос. ун-та, 2004. - 71 с.

Предлагаемое учебное пособие обеспечивает проведение цикла лабораторных работ, направленных на изучение логической организации и типовой информационной структуры запоминающих устройств персонального компьютера. В первом разделе пособия изучается файловая система ПК: рассматриваются внешнее и внутреннее представления о структуре дискового устройства внешней памяти, изучается "командный" пользовательский интерфейс и алгоритмы выполнения основных файловых операций. Второй раздел посвящен исследованию структуры областей оперативной памяти, обслуживающих механизм прерываний и процедуры обмена данными с периферийными устройствами (на примере клавиатуры ПК). В третьем разделе исследуются структуры данных, обеспечивающие работу видеосистемы ПК.

Каждая из лабораторных работ содержит краткое теоретическое введение по рассматриваемой теме, методические рекомендации по выполнению работы и описание используемых программных инструментальных средств, набор учебных заданий и список контрольных вопросов, ответы на которые должны быть получены студентом экспериментально в процессе выполнения работы.

Каждый из разделов пособия завершается перечнем контрольных задания повышенной сложности, которые могут быть использованы для промежуточного контроля знаний.

Пособие предназначено для студентов младших курсов технических специальностей, предусматривающих углубленную подготовку в области информатики и компьютерной техники.

Рис. - 7, табл. – 13, библиогр. - 6 назв.

ISBN 5-86328-617-2

С О Д Е Р Ж А Н И Е

стр.

1 ФАЙЛОВАЯ СИСТЕМА *ПК* 5

1.1 Лабораторная работа №1 5

Командный интерфейс пользователя *5*

1.1.1 Классификация и формат команд 5

1.1.2 Примеры использования команд 8

1.1.3 Шаблоны и групповые имена 11

1.1.4 Перенаправление ввода-вывода 11

1.1.5 Учебные задания 12

1.1.6 Контрольные вопросы 13

1.2 Лабораторная работа №2

Программирование пакетных файлов *14*

1.2.1 Типовая структура bat-файла 14

1.2.2 Переменные и параметры пакетных файлов 14

1.2.3 Специальные команды bat- файлов 15

Команда ECHO 16

Команда CALL 17

Команда GOTO 18

Команда IF 18

Команда Shift 20

Команда FOR 20

Команда CHOICE 21

1.2.4 Учебные задания 22

1.2.5 Контрольные вопросы 22

1.3. Лабораторная работа №3

Исследование алгоритмов реализации файловых операций *23*

1.3.1 Структура дискового пространства 23

1.3.1.1 Информационная структура 23

1.3.1.2 Физическая структура 24

1.3.1.3 Логическая структура 25

1.3.1.4 Особенности структуры жесткого диска 29

1.3.1.5 Особенности хранения длинных имен файлов 31

1.3.2 Алгоритмы выполнения типовых файловых операций 32

1.3.3 Учебные задания 34

1.3.4 Контрольные вопросы 35

1.4 Контрольная работа №1 36

2 АДРЕСНОЕ ПРОСТРАНСТВО *ПК* 39

2.1 Лабораторная работа №4

Исследование служебных областей ОЗУ *39*
2.1.1 Линейные адреса 39
2.1.2 Сегментная организация памяти 40
2.1.3 Информационная структура адресного пространства 41
2.1.4 Таблица векторов прерываний 43
2.1.5 Область данных BIOS 46
2.1.6 Адресное пространство ввода-вывода 47
2.1.7 Учебные задания 48
2.1.8 Контрольные вопросы 49

2.2 Лабораторная работа № 5

Клавиатура ПК *50*
2.2.1 Алгоритм ввода данных с клавиатуры 50
2.2.2 Флаги клавиатуры 51
2.2.3 Буфер клавиатуры 52
2.2.4 Учебные задания 54
2.2.5 Контрольные вопросы 55

2.3 Контрольная работа №2 55

3 ВИДЕОСИСТЕМА ПК 56

3.1 Структура и основные характеристики видеосистемы ПК *56*

3.2 Структуры данных, обслуживающие видеосистему *57*

3.3 Кодирование данных в видеопамяти *58*
3.3.1 Кодирование данных в текстовых режимах 59
3.3.2 Кодирование данных в графических режимах 60

3.4 Знакогенераторы *60*

3.5 Лабораторная работа №6

Исследование видеопамяти в текстовых режимах *62*
3.5.1 Учебные задания 62
3.5.2 Контрольные вопросы 63

3.6 Лабораторная работа №7

Исследование структуры таблиц знакогенераторов *63*
3.6.1 Учебные задания 63
3.6.2 Контрольные вопросы 63

3.7 Контрольная работа №3 ..... *63*

3.8 Справочные материалы *64*

СПИСОК ЛИТЕРАТУРЫ 70

1 ФАЙЛОВАЯ СИСТЕМА *ПК*

1.1 Лабораторная работа №1
КОМАНДНЫЙ ИНТЕРФЕЙС ПОЛЬЗОВАТЕЛЯ

Цель лабораторной работы – изучение языка команд и приобретение практических навыков их использования для выполнения типовых файловых операций.

1.1.1 Классификация и формат команд

Команда - это средство общения пользователя с операционной системой компьютера. Команда вводится с клавиатуры и отображается в **командной строке** экрана (сразу после "приглашения" - информационной текстовой строки, содержимое которой пользователь может изменить специальной командой PROMPT). В процессе записи команды ее можно редактировать, удаляя или заменяя введенные символы.

Ввод команды завершается нажатием клавиши ***Enter***, после чего команда записывается в специальный буфер ОЗУ[1] и начинается процесс ее обработки интерпретатором командной строки - резидентной (т.е. постоянно находящейся в ОЗУ) системной программой **Command.com** (для MS DOS и Windows 9x) или **Cmd.exe** (для Windows NT*).

Далее введенная команда интерпретируется, проверяется ее соответствие требуемому формату и запускается на выполнение соответствующая программа (или выдается сообщение о некорректности введенной команды).

Различают **внутренние** и **внешние** команды.

Процедуры обработки всех **внутренних команд** интегрированы в программу-интерпретатор командной строки, автоматически загружаемую в память компьютера. **Имя** внутренней команды – это зарезервированное слово (или сокращение) на английском языке, обозначающее определенное действие, например, **DIR (Directory)**, **COPY, DEL (Delete), REN (Rename)**.

---

[1] В буфере командной строки хранятся несколько введенных ранее команд в порядке их исполнения. Для извлечения команд из буфера в командную строку используются клавиши–стрелки: ↑ и ↓ для выбора очередной команды из списка исполненных команд; → для посимвольного выбора предыдущей команды. Если команды исполняются при активной программной оболочке Norton Commander (или любом из ее функциональных аналогов: DOS Navigator, Windows Commander, FAR manager и др.), извлечение очередной команды из буфера осуществляется комбинацией клавиш "Ctrl - E".

Внешняя команда - это имя файла[2], содержащего исполнимую программу (т.е. имя файла, имеющего расширение COM, EXE, BAT или CMD). При выполнении внешней команды производится поиск на диске требуемого файла, загрузка его в память компьютера и запуск на выполнение. **Имя** команды включает *спецификацию* исполнимого файла в соответствии с принятым стандартом: **<имя диска>:\<путь>\<имя файла>.<расширение>.** При этом <расширение> (если это **.COM**, **.EXE** или **.BAT**) может быть опущено во всех случаях, <имя диска> и <путь> могут быть опущены в случае, если файл находится в текущем каталоге активного диска или если путь к файлу был предварительно задан командой **PATH**.

По функциональному назначению (видам выполняемых операций) команды подразделяются на ряд групп, состав которых и примеры входящих в группы команд иллюстрируется приведенной ниже таблицей.

Таблица 1.1 Классификация команд

Группа команд	Имя команды	Тип Команды	Выполняемая функция
1	2	3	4
Дисковые операции	<Диск>**:**	Внутренняя	Активизация диска (<Диск> - имя диска – одна из букв от **A** до **Z**).
	LABEL	Внешняя	Редактирование метки диска (тома)
	VOL	Внутренняя	Отображение метки диска (тома)
	CHKDSK	Внешняя	Проверка состояния структуры диска (файлы, каталоги, FAT)
	DISKCOPY	Внешняя	Физическое (посекторное) копирование дискеты
	FDISK	Внешняя	Разбиение жесткого диска на логические разделы (тома)
	FORMAT	Внешняя	Форматирование диска

[2] В комплект поставки ОС входит определенный набор программ, реализующих внешние команды. Соответствующие файлы обычно находятся в каталоге ..\Windows\Command\ (для Windows9x) или в каталоге ..\WinNT\system32\ (для WindowsNT). При создании собственных прикладных программ следует помнить, что использование любого из служебных слов, зарезервированных для внутренних команд, в качестве имени файла, содержащего прикладную программу, приведет к невозможности выполнения этой программы из командной строки – вместо нее всегда будет выполняться одноименная внутренняя команда.

Окончание Таблицы 1.1

1	2	3	4
Операции с каталогами	DIR	Внутренняя	Вывод оглавления каталога
	CHDIR (CD)	Внутренняя	Изменение текущего каталога
	MKDIR (MD)	Внутренняя	Создание нового каталога
	RMDIR (RD)	Внутренняя	Удаление каталога
	TREE	Внешняя	Вывод "дерева каталогов"
Операции с файлами	COPY	Внутренняя	Копирование файлов
	RENAME	Внутренняя	Переименование файлов
	TYPE	Внутренняя	Просмотр текстового файла
	MORE	Внешняя	Постраничный просмотр файлов
	EDIT	Внешняя	Редактирование текстовых файлов
	DEL	Внутренняя	Удаление файлов
	ERASE	Внутренняя	Удаление файлов
	PRINT	Внешняя	Печать файлов
	XCOPY	Внешняя	Копирование групп файлов вместе со структурой (деревом) каталогов
	REPLACE	Внешняя	Замена файлов в одном каталоге одноименными файлами из другого каталога
Служебные команды	DATE	Внутренняя	Установка системной даты
	TIME	Внутренняя	Установка системного времени
	PROMPT	Внутренняя	Установка формы "приглашения DOS"
	PATH	Внутренняя	Установка пути поиска программных файлов

Вводимая команда должна соответствовать определенному формату, содержащему три компонента, разделенных символом "пробел" (из них только первый компонент - имя команды - является обязательным для всех команд):

< **имя команды** > < *параметры команды*>/< *модификаторы команды*>

Параметры команды - это, как правило, имена объектов, над которыми выполняется действие (имена файлов, каталогов, дисков и пр.). Параметры отделяются друг от друга символом "пробел". Состав и порядок записи параметров уникальны для каждой команды[3].

Модификаторы (ключи) команды используются для указания конкретных условий ее применения. Модификаторы могут записываться как перед, так и после параметров команды. В качестве разделителя используется символ "/".

1.1.2 Примеры использования команд

*Команда **DIR*** - просмотр оглавлений каталогов**.** Единственным параметром этой команды является путь[4] к просматриваемому каталогу.

- **DIR (без параметров)** - вывод на экран оглавления текущего каталога активного диска.
- **DIR** - просмотр оглавления корневого каталога активного диска
- **DIR ..** - просмотр оглавления родительского каталога
- **DIR D:** - просмотр оглавления корневого каталога диска D
- **DIR D:\DOC** - просмотр оглавления подкаталога **DOC** диска D
- **DIR C:\ /p** - постраничный просмотр оглавления корневого каталога диска **C**
- **DIR C:\Windows /p/s** - постраничный просмотр оглавлений каталога \Windows диска **C** и всех подчиненных ему каталогов.
- **DIR C:\Windows\Command*.exe /p** – постраничный просмотр части оглавления каталога Command, содержащей только объекты с расширением "exe".

---

[3] Формат команды и полную инструкцию по ее применению можно вывести на экран путем выполнения этой команды без параметров с ключом **/?**

[4] **Путь** к файлу или каталогу – это список имен каталогов в порядке их подчиненности (от родительских каталогов к дочерним). Путь **к текущему каталогу** в командах **не указывается** – он известен операционной системе. Символы "\" (обратная косая черта) и "**..**" (две точки), заданные в начале строки, описывающей путь, указывают на точку отсчета его начала: символ "\" задает путь **от корневого** каталога активного диска, символы "**..**" задают путь **от родительского** каталога.

*Команда **MD*** (**M**ake **D**irectory) - создание новых подчиненных каталогов.

- **MD ddd** - создание каталога **ddd** в текущем каталоге активного диска;
- **MD \first_d** - создание каталога **first_d** в корневом каталоге активного диска;
- **MD D:\first_d\next_dir** - создание каталога **next_dir**, подчиненного каталогу **first_d**.

*Команда **CD*** (**C**hange **D**irectory) - 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).

Примечание: использование имени диска в параметре команды CD ***не допускается.***

*Команда **COPY*** - копирование файлов. В этой команде необходимы 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.

- Полноформатная команда копирования файлов, в результате выполнения которой в каталоге **TEXT** диска **C** будет создан файл **File_2.txt** - копия файла **File1.txt,** расположенного в каталоге **TXT** диска E:

COPY E:\TXT\File_1.txt C:\TEXT\File_2.txt

- Копирование файла под новым именем из текущего каталога активного диска в подкаталог **TXT** диска E : **COPY File_1.txt E:\TEXT\File_2.txt**
- Копирование файла из текущего каталога активного диска в корневой каталог диска D под старым именем: **COPY File1.txt D:**
- Объединение (слияние) трех файлов в один файл **File.All** (все четыре файла в текущем каталоге): **COPY File1+File2+File3 File.All**
- Печать (копирование на принтер – системное имя устройства PRN) файла **File.txt** из текущего каталога: **COPY File.txt PRN**
- Просмотр файла **File.txt**: **COPY File.txt CON.** Здесь в качестве параметра команды используется имя **CON** (сокращение от **Con**sole), зарезервированное операционной системой для стандартных устройств ввода-вывода. При *вводе* данных это устройство - клавиатура, при *выводе* - экран видеомонитора. Аналогичного результата можно достичь и командой **TYPE File.txt.**

- Создание (копирование с клавиатуры) нового текстового файла **File.txt** в текущем каталоге.: **COPY CON A:\File.txt.** После выполнения такой команды можно вводить с клавиатуры произвольный текст. Для завершения процесса создания файла следует ввести специальный управляющий символ "конец файла" (CTRL-Z) и нажать клавишу Enter.

*Команда **PATH*** задает пути поиска файлов с исполнимыми программами, если в команде запуска программы путь к файлу не указан, а в текущем каталоге активного диска файл отсутствует:

PATH C:\;C:\Windows;C:\Windows\Command;C:\tools;D:

В этой команде используется всего один параметр - *список путей поиска,* разделенных символом "**;**" (точка с запятой). После выполнения такой команды список путей помещается в специальный буфер ОЗУ, и поиск файлов будет осуществляться в соответствии с этим списком (слева направо). Например, после ввода команды **Format** (запуск программы форматирования диска) DOS попытается найти файл **Format.com (Format.exe** или **Format.bat)** *в текущем каталоге*. Если ни одного из этих файлов в текущем каталоге нет, их поиск будет производиться в соответствии со списком путей, введенным командой **PATH:** сначала в корневом каталоге диска **C**, затем в подкаталоге **DOS** диска **C** , затем в подкаталоге **Tools** диска **C** и т.д. Если ни в одном из каталогов заданного списка указанный файл не найден, DOS выведет соответствующее сообщение.

Команда **PATH** с параметром "**;**" очищает буфер, то есть **отменяет** все ранее установленные **пути поиска** исполнимых файлов.

Команда **PATH**, введенная **без параметра**, выводит на экран текущее состояние буфера.

*Команда **FORMAT*** используется для форматирования дисков. Команда имеет один обязательный параметр - имя форматируемого диска - и множество модификаторов :

- **FORMAT A:** - стандартное (полное) форматирование дискеты, при котором производится разметка дорожек и секторов с контролем качества рабочих поверхностей.
- **FORMAT A: /q** - быстрое (**q**uick) форматирование дискеты, при котором производится только "обнуление" корневого каталога и таблицы расположения файлов (FAT).
- **FORMAT A: /S –** форматирование дискеты с последующим копированием файлов операционной системы.

*Команда **LABEL*** позволяет просматривать и редактировать **метки дисков.**

- **LABEL A:** - после ввода команды на экран будет выведена метка диска **A** (или сообщение об отсутствии метки на этом диске) и будет предложено ввести новую метку (не более 11 символов).

1.1.3 Шаблоны и групповые имена

В приведенных выше примерах файловых команд предполагалось, что одной командой выполняется соответствующая операция над одним файлом, имя которого указано в параметре команды. Использование в имени файла специальных шаблонов (подстановочных символов) позволяет одной командой произвести операцию над целой группой файлов.

Допускается использование в именах файлов два типа шаблонов: символ "**?**" обозначает любой один символ, а символ "*****" (звездочка) обозначает любое число любых символов в имени или расширении файла (каталога).

Например, команда **COPY *.txt A:** произведет копирование всех файлов, имеющих расширение **.txt**, из текущего каталога активного диска в корневой каталог диска **A** под старыми именами, а команда **DEL ??DUM.*** удалит из текущего каталога все файлы с именами из пяти символов, оканчивающимися на "DUM", и любыми расширениями.

1.1.4 Перенаправление ввода-вывода

Многие команды выводят на экран результаты своей работы (например, команды **DIR** и **TYPE**) или диагностические сообщения (например, команды **COPY** и **FORMAT**). С помощью специального символа ">", записываемого в конце командной строки, можно перенаправить вывод на другие устройства или записать всю выводимую информацию в файл.

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.lst** отсутствует, он будет создан в результате выполнения команды. Если файл уже существует, он будет замещен новым файлом с этим же именем.

Для того, чтобы добавлять очередные выводимые сообщения в существующий файл, надо использовать два символа ">>" вместо одного ">". Например, команда **DIR C:\ >>dir.lst**, выполненная после приведенной выше команды, "допишет" в файл **dir.lst** оглавление корневого каталога диска **C** после уже записанного в этом файле оглавления каталога диска **D**.

Фиктивное (реально не существующее) внешнее устройство с системным именем "**NUL**" используется для подавления вывода сообщений на экран. Например, при выполнении команды **COPY File1.txt File2.txt>NUL** будет заблокирован вывод стандартного сообщения этой команды "Один файл скопирован".

1.1.5 Учебные задания

- Создайте личную папку (каталог) в соответствии с указаниями преподавателя. Все объекты (файлы и каталоги), создаваемые при выполнении учебных заданий, следует размещать в этом каталоге или в подчиненных ему каталогах.
- Установите командный режим работы Вашего ПК - выберите "*Сеанс MS DOS*" или "*Командная строка*" в пункте "*Программы*" или выполните команду "Command" или "Cmd" в пункте "*Выполнить*" главного меню (кнопка "Пуск"). Все последующие учебные задания следует выполнять в командном режиме.

1 Используя команду PROMPT, установите "стандартный" вид приглашения DOS – активный диск и путь к текущему каталогу. Измените приглашение DOS:
   - текущая дата и время;
   - версия Windows;
   - собственная фамилия и номер группы;
   - восстановите стандартный вид приглашения.

2 Используя команду **COPY** создайте в личном каталоге два коротких текстовых файла, содержащих по одной строке текста - Ваши фамилия, имя и отчество на русском (в файле **xxx_rus.txt**) и английском (в файле **xxx_engl.txt**) языках. В именах файлов "xxx" - Ваши инициалы на английском языке.

3 Отредактируйте файлы, созданные при выполнении предыдущего задания, с помощью команды **EDIT** (например, дополните текст Вашим домашним адресом).

4 Создайте в Вашем личном каталоге трехуровневую систему подчиненных каталогов и скопируйте в каждый из них под различными именами файлы, созданные при выполнении предыдущего задания.

5 Создайте в личном каталоге новый файл путем объединения двух файлов, созданных при выполнении 3-го задания.

6 Используя команды COPY и TYPE, просмотрите на экране содержимое всех файлов, созданных при выполнении выполненных заданий.

7 Установите текущим один из созданных каталогов и сохраните его оглавление в файле **Direct.txt**, расположенном в этом каталоге.

8 Сохраните оглавления личного каталога и всех подчиненных ему каталогов в файле **My_Dir.txt**, расположенном в личном каталоге. Предложите несколько вариантов формирования такого файла.

9 Активизируйте программную оболочку Norton Commander (или любой ее функциональный аналог – например, DOS Navigator или FAR manager) и выполните ***задания 2 –6*** без прямого применения команд.

1.1.6 Контрольные вопросы

1 Объясните термин ***приглашение DOS***. Какая команда управляет формой *приглашения* ? Как установить в *приглашении* вывод системной даты *и* пути к текущему каталогу ?

2 Объясните термин ***команда DOS***. Каковы действия DOS при обработке внутренних и внешних команд? Каковы правила записи параметров и модификаторов команд ?

3 Объясните термины ***имя файла***, ***путь к файлу***, ***спецификация файла***. Найдите неправильные обозначения спецификаций файлов в приведенных ниже командах, исправьте ошибки и прокомментируйте результаты выполнения команд:

MD D:\QQ

CD D:\QQ

CD ..QQ

COPY A:/ddd.pp/fff.txt B/rrr.xtx

RENAME A:\ddd\rrr.123\fff.txt rrr.xtx

DEL A:\ddd.pppp\fff.txt

TYPE D:ABCD_EFGH.pas

TYPE C:\ABCD-EFG.bas1

4 Как реализуются операции над группами файлов ? Что произойдет в результате выполнения приведенных ниже команд?

COPY B:\TASK??.exe A:\TASK
COPY B:\TASK?.* A:
COPY B:\T*.prg A:
COPY B:*.* A:\DIR*.new

5 Охарактеризуйте группу команд, обеспечивающих работу с дисками.

6 Охарактеризуйте группу команд, обеспечивающих работу с каталогами.

7 Охарактеризуйте группу команд, обеспечивающих работу с файлами.

8 Объясните назначение и правила использования команды PATH.

9 Какие команды позволяют объединять несколько файлов в один ?

10. Чем отличаются команды **Format** и **Fdisk** ?

11. Чем отличаются команды **Copy, XCopy** и **DiskCopy** ?

12. Для чего и как используется перенаправление ввода-вывода ?

1.2 Лабораторная работа №2

ПРОГРАММИРОВАНИЕ ПАКЕТНЫХ ФАЙЛОВ

Цель лабораторной работы – изучение расширенного набора командного языка и получение навыков программирования пакетных файлов.

1.2.1 Типовая структура bat-файла

Несколько команд могут быть построчно записаны в текстовый файл, который должен иметь стандартное расширение **.bat** (от англ. **batch** - пакет, пачка). Такой файл является программой, которую можно "выполнить", используя имя файла, как внешнюю команду - при этом содержащиеся в bat-файле команды будут выполняться построчно в порядке их записи.

Пакетные файлы полезны тогда, когда есть необходимость в выполнении часто повторяющихся последовательностей команд. Например, если в текущем каталоге создан текстовый файл **111.bat**, содержащий четыре строки с командами :

```
MD  NewDir
COPY  qqq.txt  NewDir\ppp.txt
CD  NewDir                                    (1)
TYPE  ppp.txt
```

то при выполнении команды **111.bat** (или просто **111**) в текущем каталоге будет создан новый каталог **NewDir**, затем в этот каталог будет скопирован из текущего каталога файл **qqq.txt** под именем **ppp.txt**, затем каталог **NewDir** будет установлен текущим каталогом, и на экран будет выведен текст, содержащийся в файле **ppp.txt**.

1.2.2 Переменные и параметры пакетных файлов

В отличие от команд, вводимых непосредственно из командной строки, команды bat-файла могут содержать вместо своих фактических параметров идентификаторы (имена) переменных. Допускается использовать не более 10 переменных, обозначаемых (в тексте bat-файла) символом "%" и порядковым номером от 0 до 9 (%0, %1, ..., %9).

В языке пакетных файлов *отсутствует оператор присваивания* - единственным способом присвоения переменным их фактических значений является передача этих значений через список параметров, которые записываются после имени bat-файла при его запуске из командной строки. Каждый параметр списка отделяется от соседних символом "пробел" и автоматически подставляется в текст bat-файла вместо переменной с соответствующим номером. Например, переменная %2 получит при выполнении файла значение второго по порядку параметра из списка. Заметим,

что **переменная %0** не соответствует никакому параметру - она **получает значение имени bat-файла**, указанного в командной строке.

Соответствие имен переменных номерам параметров может быть изменено командой **SHIFT**. Эта же команда позволяет использовать количество параметров bat-файла, превышающее число используемых переменных.

Для иллюстрации правил использования переменных и параметров в пакетных файлах заменим в рассмотренном выше примере значения параметров команд соответствующими переменными :

```
MD  %1
COPY  %2  %1\%3
CD  %1                                    (2)
TYPE  %3
```

Если такой файл создан в текущем каталоге и имеет имя **222.bat**, то при выполнении внешней команды **222.bat NewDir qqq.txt ppp.txt** переменная **%1** получит значение **NewDir**, переменная **%2 -** значение **qqq.txt**, а переменная **%3** - значение **ppp.txt**. Таким образом, файл с такими параметрами полностью идентичен (по результатам его выполнения) файлу, приведенному в первом примере.

Если изменить значения параметров команды **222** при ее запуске, соответственно изменятся и параметры команд, помещенных в текст этого bat-файла.

1.2.3 Специальные команды bat- файлов

Специальные команды существенно повышают эффективность применения пакетных файлов, делая их полноценными программами. Ниже дан краткий обзор таких команд и рассмотрены примеры их применения.

Таблица 1.2 - Специальные команды пакетных файлов

Группа команд	Имя команды	Тип команды	Выполняемая функция
Команды пакетных файлов	ECHO	Внутренняя	Вывод сообщений, блокировка отображения команд
	GOTO	Внутренняя	Переход на метку
	IF	Внутренняя	Условное выполнение команды
	FOR	Внутренняя	Циклическое выполнение команды
	CALL	Внутренняя	Вызов подчиненного bat-файла
	SHIFT	Внутренняя	Сдвиг списка фактических параметров bat-файла относительно списка используемых переменных
	CHOICE	Внешняя	Выбор из списка альтернатив – используется для организации "меню".

*Команда **REM***, помещенная в начале строки, блокирует выполнение записанной в ней команды. REM обычно используется для записи комментариев в тексте командного файла или для временного блокирования команд.

Команда **:** (двоеточие), помещенная в начале текстовой строки (не более 8 символов), присваивает этой строке статус **метки**, которая может использоваться в командах GOTO.

Команда **@**, помещенная в начале строки, подавляет отображение этой строки на экране при выполнении bat-файла.

*Команда **ECHO*** используется в трех модификациях:

- **ECHO ON/OFF** - разрешение / запрет вывода на экран сообщений и командных строк при их выполнении. Действует на все последующие строки до отмены аналогичной командой **ECHO OFF/ON.** По умолчанию действует параметр **ON**.
- **ECHO** (без параметров) - вывод на экран текущего состояния команды (ON или OFF).
- **ECHO <сообщение>** - вывод на экран текста сообщения, заданного параметром команды.

Команда **ECHO** (совместно с символами перенаправления ввода-вывода) часто используется при выполнении команд из bat-файлов для замены стандартных (как правило, на английском языке) сообщений команд другими сообщениями, задаваемыми пользователем.

Следующий пример иллюстрирует возможности команды **ECHO**

```
@ECHO OFF
MD TEXT
DIR  TEXT >Dir.lst
ECHO  Оглавления всех каталогов - в файле Dir.lst
ECHO  Копирование текстовых файлов из текущего каталога
COPY  *.TXT   TEXT\*.* > NUL
ECHO  Копирование завершено
DIR  TEXT >>Dir.lst                                    (3)
ECHO  Удаление текстовых файлов из текущего каталога
DEL  TEXT\*.TXT > NUL
ECHO  Удаление завершено
DIR  TEXT >>Dir.lst
ECHO  Просмотр оглавлений каталогов
TYPE   Dir.lst
```

При выполнении этого файла оглавления каталога **TEXT** во всех его трех состояниях будут последовательно записаны в файл **Dir.lst**, расположенный в текущем каталоге активного диска. Содержимое этого файла затем будет выведено на экран командой **TYPE**. Стандартные сообщения команд **COPY** и **DEL** выводиться не будут - вместо них на экран будет выводиться текст, указанный в параметрах команд **ECHO**.

Команда CALL <имя bat-файла> <параметры> вызывает другой (вложенный) пакетный файл, по завершению работы которого управление передается следующей строке родительского пакетного файла.

Пусть в текущем каталоге зарегистрированы два bat-файла : **QQQ.bat** и **PPP.bat**

Файл **QQQ.bat**

```
@ ECHO OFF
COPY  %2   %1\%3
CALL PPP.bat    File_1  %3  %1
TYPE  %3                                    (4)
```

Файл PPP.bat

```
@ ECHO OFF
ECHO Исходные файлы %1 и  %2
CD %3
COPY   %1+%2   %2 > Nul
ECHO Слияние файлов завершено
```

При запуске файла QQQ командой "**QQQ.bat Dir1 Name1 Name2**" его переменные %1, %2 и %3 получат значения соответственно **Dir1, Name1 и Name2.** При выполнении второй команды этого файла в подкаталог **Dir1** будет скопирован файл **Name1** под именем **Name2**. Команда **CALL** запустит на выполнение файл **PPP.bat**, передав ему в качестве первого параметра текстовую константу **File_1,** в качестве второго параметра – значение переменной %3 файла QQQ.bat (то есть **Name2**), а в качестве третьего параметра – значение первого параметра файла QQQ.bat (то есть **Dir1**). В результате переменные %1, %2 и %3 файла **PPP.bat** получат значения соответственно **File_1, Name2 и Dir1**, и два соответствующих файла будут соединены и сохранены в файле **Name2**, зарегистрированном в каталоге **Dir1**. После завершения работы файла **PPP.bat** будет выполнена команда **TYPE** файла **QQQ.bat**, которая выведет на экран содержимое файла **Name2**.

Можно запустить bat-файл из другого bat-файла и без команды **CALL** (например, вместо строки **CALL PPP.bat File_1 %3 %1** записать строку **PPP.bat File_1 %3 %1**). Однако, в этом случае после завершения работы вложенного bat-файла **PPP.bat** не произойдет возврата к следующей строке исходного bat-файла **QQQ.bat**.

Команда GOTO <метка> - безусловный переход к строке с указанной меткой, то есть строке, которая начинается с двоеточия, после которого следует текст <метка> (не более 8 символов).

Команда IF<условие><команда>
или **IF NOT<условие><команда>**

Заданная **<команда>** будет выполнена в том случае, если **<условие>** истинно (или ложно). После завершения выполнения **<команды>** выполняется следующая строка командного файла. Если заданное **<условие>** ложно (или истинно), заданная **<команда>** не выполняется, и сразу выполняется следующая строка командного файла.

Параметр **<условие>** может иметь одну из следующих форм:

- **EXIST <спецификация файла>** - истинно, если файл существует;
- **NOT EXIST <спецификация файла>** - истинно, если такого файла нет;
- **<строка1>==<строка2>** - истинно при полном совпадении заданных символьных строк. Если в этом условии вместо строк используются переменные (%0 ... %9), то при выполнении команды на их место будут подставлены значения соответствующих параметров. Например : **%n==<строка>** - проверка совпадения **n**-го параметра с заданной текстовой строкой; **%n==%m** - проверка совпадения значений двух параметров. При сравнении параметров следует учитывать, что переменные, соответствующие **отсутствующим** параметрам, будут представлены символом "пробел", что может привести к ошибкам при выполнении bat-файлов, так как "пустые" параметры интерпретатором не обрабатываются. Для исключения таких ошибок при выполнении операций сравнения значений переменных с "пустыми" параметрами можно использовать при записи условий два одинаковых дополнительных символа (например, знак "минус") в левой и правой частях условия сравнения: например, условие **-%n==-** будет истинным, если **n** -й параметр пуст.

Errorlevel <значение>. Процедуры выполнения некоторых внутренних и внешних команд, завершая свою работу, формируют определенное числовое значение специальной системной переменной ERRORLEVEL (в переводе - "уровень ошибки"), называемой *кодом завершения программы*. Значение переменной ERRORLEVEL, сформированное последней из выполняемых программ, хранится в памяти ПК и может быть использовано другими программами, а также командой **IF** <условие> <команда> в составе параметра <условие>. Например, условие "ERRORLEVEL 5" будет "истинным", если переменная ERRORLEVEL получила значение, **равное или большее 5**.

Примеры кодов завершения некоторых программ, реализующих внешние команды, приведены в таблице 1.3.

Таблица 1.3 - Коды завершения программ

Команда	Errorlevel	Условие завершения программы
FORMAT	0 1, 2 3, 5 4	Успешное форматирование диска Неопределенная ошибка Форматирование прервано пользователем Фатальная (неисправимая) ошибка
XCOPY	0 1 2 4	Успешное копирование Не найдено файлов Копирование прервано пользователем Ошибка инициализации (не хватает памяти, не найден путь и др.)
DISKCOPY	0 1 2 3 4	Успешное копирование Ошибка ввода-вывода Копирование прервано пользователем Фатальная ошибка Ошибка инициализации (не хватает памяти, неправильный синтаксис)
REPLACE	0 2 3 5 8 11 15 22	Успешное завершение Не найдены входные файлы Не найден входной или выходной каталог Доступ к файлу или каталогу запрещен Нехватка памяти Неправильный формат команды Неправильный диск Неправильная версия DOS
CHOICE	1 2 ... n	Пользователем выбран 1-й пункт меню Пользователем выбран 2-й пункт меню ... Пользователем выбран n-й пункт меню

В приведенном ниже фрагменте текста bat-файла команда IF анализирует числовое значение кода завершения программы форматирования диска и выдает соответствующее сообщение.

```
Format A:
If Errorlevel 4 Echo Можете смело выбрасывать свою дискету !
GoTo Exit
If Errorlevel 3 Echo Форматирование прервано
GoTo Exit
If Errorlevel 1 Echo Попытайтесь отформатировать диск еще раз
GoTo Exit
If Errorlevel 0 Echo Все в порядке - Ваш диск отформатирован
:Exit
```

Команда ***Shift*** сдвигает список параметров пакетного файла на одну позицию влево относительно списка переменных. После однократного выполнения команды SHIFT переменная %1 получит значение второго параметра, %2 - третьего и т.д. Применение этой команды позволяет использовать практически неограниченное число параметров при ограниченном числе переменных.

```
:Loop
 ECHO %1
SHIFT
IF -%1==- GOTO Exit                                    (5)
GOTO Loop
:Exit
ECHO Список параметров исчерпан
```

В этом примере 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").

Команда **FOR** <*параметр цикла*> **IN (**<*список*>**) DO** <*команд*а> обеспечивает циклическое выполнение (**DO**) указанной <команды> для (**FOR**) всех значений ее параметра, помещенных в (**IN**) заданный список.

<*Параметр цикла*> (не путать с параметрами bat-файла !) обозначается одним символом (буквой), которому предшествуют два символа "%", например: **%%A, %%W.**

<*Список*> - это набор символьных строк, разделенных пробелами. Чаще всего - это спецификации файлов или имена подкаталогов – в этом случае допускается использование групповых имен с подстановочными символами "*" и "?".

```
FOR %%d IN (Dir1 Dir2 Dir3) DO MD %%d
FOR %%d IN (Dir1 Dir2 Dir3) DO COPY File1 %%d\         (6)
```

Выполнение этих двух команд, включенных в текст bat-файла, приведет к созданию в текущем каталоге трех подкаталогов - Dir1, Dir2 и Dir3 и копированию в каждый из них из текущего каталога файла File1 под своим именем.

```
FOR %%k IN (*.%1) DO ECHO %%k                          (7)
```

Пример иллюстрирует использование в <списке> переменных bat-файла и групповых имен файлов. Приведенная выше команда выводит на экран имена всех файлов текущего каталога, имеющих расширение, задаваемое первым параметром bat-файла.

```
FOR %%L IN (txt  doc  xls) DO IF exist %1.%%L COPY %1.%%L DIR1\  (8)
```

Пример иллюстрирует использование команды **IF**, вложенной в команду **FOR:** производится последовательное копирование в каталог

DIR1 из текущего каталога всех файлов с заданным именем (%1) и расширениями .txt, .doc, .xls, если такие файлы созданы в текущем каталоге.

Использование команд **IF, ECHO и CHOICE** для создания меню

Команда CHOICE (в переводе - выбор) выводит на экран **вопрос** и предлагает пользователю выбрать (то есть ввести с клавиатуры) один из нескольких **вариантов ответа** из предлагаемого списка возможных ответов. Текст вопроса должен быть задан, как параметр команды. Список возможных ответов задается после ключа **/C:**.

Например, команда **CHOICE /C:ДН Удалить файлы ... ?** выведет на экран вопрос **Удалить файлы ... ?** и предоставит для выбора список из двух вариантов ответа : "**Д**" или "**Н**". Пользователь может выбрать один из двух ответов путем ввода с клавиатуры символа "Д" или символа "Н". После ввода ответа команда **CHOICE** завершает свою работу и присваивает числовое значение переменой **ERRORLEVEL**, равное порядковому номеру ответа в предложенном списке. Это значение может быть использовано командами, записанными ниже в тексте bat-файла, например, командой **IF**.

При запуске следующего bat-файла команда удаления файлов будет выполняться в том случае, если пользователь выбрал утвердительный ответ "Д" (номер 1 в списке):

```
CHOICE /C:ДН Удалить все файлы из каталога \TEXT ?
IF ERRORLEVEL 2 GOTO Exit
ECHO Удаляю файлы                                    (9)
DEL \TEXT\*.* > Nul
ECHO Файлы удалены
: Exit
```

Замечания

1. Параметр команды **CHOICE**, содержащий текст вопроса, может быть опущен - в этом случае команда выведет только список ответов.
2. Ключ **/C:** с вариантами ответов тоже может быть опущен - в этом случае будет выведен стандартный список из двух ответов "**YN**".
3. Команда использует и другие ключи, полный список которых выводится при ее запуске с ключом **/?**.

Следующий пример иллюстрирует возможности создания меню из нескольких пунктов с помощью команд **CHOICE, ECHO, IF**. В тексте bat-файла используются переменные, значения которых должны быть заданы соответствующими параметрами при запуске файла на выполнение :
1-й параметр - имя редактируемого текстового файла (переменная %1);
2-й параметр - имя каталога на диске D (%2);
3-й параметр - расширение файлов, подлежащих копированию (%3).

```
:Start
ECHO  Выберите  нужную Вам операцию :
ECHO  ( F ) - Форматирование диска A
ECHO  ( C ) - Копирование на диск A файлов с жесткого диска
ECHO  ( D ) - Просмотр оглавления каталога
ECHO  ( E ) - Редактирование текстового файла
ECHO  ( 0 ) - Выход
CHOICE /C: FCDE0
IF ERRORLEVEL  5 GOTO  EXIT
IF ERRORLEVEL  4 GOTO  EDITOR
IF ERRORLEVEL  3 GOTO  DIRECTORY
IF ERRORLEVEL  2 GOTO  COPYFILE
FORMAT A:
GOTO  Start                                                    (10)
: EDITOR
EDIT %1.TXT
GOTO  Start
: DIRECTORY
DIR D:\%2
GOTO  Start
: COPYFILE
COPY D:\ %2\*.%3   A:\*.*
GOTO  Start
: EXIT
ECHO  Завершение работы
```

1.2.4 Учебные задания

Задание 1. Подготовить bat-файлы, иллюстрирующие приведенные выше примеры (1 ... 10). Выполнить эти файлы с различными вариантами входных параметров.

Задание 2. Подготовить bat-файл, при выполнении которого в текущем каталоге создается его копия под новым именем (имя указывается при запуске bat-файла). Созданная копия файла должна оставаться работоспособной – то есть способной создавать свои копии.

1.2.5 Контрольные вопросы

1. Каковы количественные ограничения на использование переменных и параметров пакетных файлов ?
2. Предложите несколько способов организации bat-файла, в котором команды выполняются не в том порядке, в котором они записаны.
3. Предложите несколько способов организации циклического выполнения группы команд.
4. Охарактеризуйте группу команд, используемых для создания пользовательских меню.

1.3. Лабораторная работа №3
Исследование алгоритмов реализации файловых операций

Файловая система - это часть операционной системы, обеспечивающая работу прикладных программ с внешними запоминающими устройствами. Термином "файловая система" охватывается функционально полный набор программных компонентов различных уровней, реализующих алгоритмы обработки и доступа к файлам (функции DOS и BIOS, драйверы устройств), а также используемые этими программами структуры данных, формируемые на диске и в памяти ПК.

Цель лабораторной работы - изучение логической структуры дисковых накопителей в FAT-ориентированных файловых системах и исследование алгоритмов исполнения команд, обеспечивающих выполнение типовых операций доступа к файлам и каталогам (DIR, MD, CD, COPY, REN, DEL и др.).

1.3.1 Структура дискового пространства

Файловая система использует несколько уровней представлений о структуре дискового пространства, каждый из которых ориентирован на определенную группу потребителей и позволяет решать определенный круг задач доступа к файлам. С этой точки зрения различают информационную, логическую и физическую структуры, для которых существует определенный набор структурных элементов и используется соответствующая терминология.

1.3.1.1 Информационная структура

Информационная структура дискового пространства – это его внешнее представление, ориентированное на пользователя и определяемое такими элементами, как ***том (***иначе называемый *логическим диском)*, ***каталог*** и ***файл.*** Элементы информационной структуры используются при общении пользователя с системой посредством команд, реализующих операции доступа к файлам и каталогам.

Дисковое пространство представляется пользователю, как множество *логических дисков*, для именования которых используются буквы латинского алфавита от **A** до **Z**. При этом имена **A** и **B** зарезервированы для гибких дисков.

С каждым из логических дисков связано *дерево каталогов,* включающее (в обязательном порядке) один *корневой каталог* (root directory) и (необязательно) множество иерархически подчиненных каталогов. Корневой и подчиненные каталоги могут "содержать" *файлы* и (или) другие (дочерние) подкаталоги. Как будет показано ниже, каталог содержит не сами файлы (подкаталоги), в только регистрационные записи, включающие имена, адреса и другую служебную информацию о дочерних каталогах и файлах.

Корневой каталог формируется автоматически при форматировании диска и имеет фиксированный размер, ограничивающий число регистрируемых в нем дочерних объектов (файлов и подкаталогов). Корневой каталог не имеет имени - можно считать, что имя корневого каталога совпадает с именем соответствующего логического диска.

Подчиненные каталоги – это (по существу) файлы определенной структуры, аналогичной структуре корневого каталога. Размер подчиненного каталога не фиксирован – он динамически изменяется при добавлении и удалении регистрируемых в нем объектов (файлов или дочерних подкаталогов). Размер подчиненного каталога ограничивается только размером соответствующего логического диска.

Подчиненные каталоги создаются пользователем (с помощью команды **MD**) и обязательно имеют имена, формируемые в соответствии с *соглашением об именах*, принятом в операционной системе[5].

1.3.1.2 Физическая структура

Технические средства чтения-записи информации имеют дело с физической структурой дискового пространства, которая описывается такими терминами, как ***диск*** (или ***пакет дисков)***, понимаемый как физическое устройство с автономным приводом, ***рабочая поверхность*** диска (Side), связанная с магнитной головкой чтения-записи (Head), магнитная ***дорожка*** (Track) и ***сектор***. Каждая дорожка поделена на секторы, для стандартных форматов диска количество секторов на каждой из дорожек одинаково. Множество дорожек одинакового радиуса, расположенных на всех рабочих поверхностях пакета дисков, образуют так называемый ***цилиндр***. Доступ ко всем дорожкам одного цилиндра может быть обеспечен при однократном радиальном позиционировании блока магнитных головок.

Сектор - это минимальная единица дискового пространства, к которой можно обратиться с целью чтения или записи информации. Информационная емкость сектора обычно составляет 512 байтов.

Все рабочие поверхности, дорожки и секторы последовательно пронумерованы, так что номер поверхности, номер дорожки на этой поверхности и номер сектора на этой дорожке однозначно определяют иерархический адрес сектора. Используют также и *абсолютную* нумерацию секторов – начиная с нулевого сектора нулевой дорожки нулевой поверхности диска. Для однозначного определения места расположения некоторого файла на диске необходимо связать с этим файлом упорядоченную последовательность пронумерованных секторов.

---

[5] Файлы и подчиненные каталоги в MS DOS должны иметь имена, содержащие не более 8 символов (английских букв, цифр и некоторых других символов), и могут иметь расширение, содержащее не более 3-х символов. ОС Windows 9x допускает использование "длинных" имен - до 255 любых ASCII-символов, в том числе и букв русского алфавита. В WindowsNT* возможности именования еще более расширены за счет использования 16-битовой системы кодирования символов (Unicode).

Следует отметить, что сектор - это весьма малая единица емкости дискового пространства, и использование номера сектора в качестве логического адреса файла на диске может создавать проблемы технического характера и снижает эффективность использования дисков большой емкости. Для этой цели используется другая, более крупная единица, называемая *кластером* и являющаяся основным элементом логической структуры диска.

1.3.1.3 Логическая структура

Логическая структура реализует линейную ("ленточную") модель дискового пространства тома. Согласно этой модели том разделен на две расположенные последовательно области - системную и рабочую.

Рабочая область, расположенная непосредственно после системной, разделена на последовательно пронумерованные *кластеры* и предназначена для хранения файлов и подкаталогов. Кластер используется в качестве минимальной единицы, выделяемой операционной системой одному файлу (или подкаталогу). Каждый кластер имеет уникальный номер (1, 2,...) и содержит несколько расположенных подряд секторов (1 или 2 сектора в кластере - для гибких дисков, 4 и более - для жестких). Между номером кластера и списком абсолютных номеров входящих в него секторов существует взаимно-однозначное соответствие.

Системная область занимает несколько начальных (в абсолютной нумерации) секторов, начиная с нулевого, и содержит блоки служебной информации, используемые для организации доступа к файлам и загрузки операционной системы: *блок загрузки* (Boot-Sector), *таблица распределения файлов* (FAT - File Allocation Table) и *корневой каталог* (Root Directory).

Блок загрузки - всегда занимает нулевой сектор и содержит таблицу параметров формата диска и короткую программу загрузки DOS. Структура блока загрузки поясняется таблицей 1.4.

Таблица 1.4 - Структура блока загрузки

Смещение	Длина, байт	Содержимое
1	*2*	*3*
+00	3	Команда перехода на программу загрузки (JMP)
+03	8	Служебная информация программы форматирования
+0bH	2	Размер сектора в байтах
+0dH	1	Размер кластера в секторах
+0eH	2	Количество секторов перед первой FAT
+10H	1	Количество FAT

Окончание таблицы 1.4

1	*2*	*3*
+11H	2	Максимальное число элементов корневого каталога
+13H	2	Общее число секторов на логическом диске
+15H	1	Дескриптор носителя (то же, что 1-й байт FAT)
+16H	2	Количество секторов в одной FAT
+18H	2	Количество секторов на дорожке
+1aH	2	Количество головок чтения/записи (поверхностей)
+1cH	2	Количество спрятанных секторов
+1eH		Размер форматированной порции корневого сектора
		Начало кода и данных загрузки

Корневой каталог состоит из набора регистрационных записей (32-байтных элементов)[6], каждая из которых содержит информацию об одном файле или подкаталоге и имеет следующую структуру.

Таблица 1.5 - Структура корневого каталога

Смещение	Длина, байт	Содержимое
+00	8	Имя файла или каталога
+08	3	Расширение
+0bH	1	Атрибуты файла
+0cH	0aH	Резерв
+16H	2	Время создания/модификации (в специальном формате)
+18H	2	Дата создания/модификации (в специальном формате)
+1aH	2	Номер начального кластера
+1cH	4	Размер файла в байтах
+20H		Размер элемента оглавления

Шесть младших битов *байта атрибутов* используются как битовые флаги, единичное значение каждого из которых задает определенное свойство соответствующего объекта:

0-й бит = 1 Только чтение (***R/O*** - Read Only)
1-й бит = 1 Спрятанный (***Hid*** - Hidden)
2-й бит = 1 Системный (***Sys*** - System)

[6] Указанная в таблице 1.5 структура записи каталога поддерживается MS DOS. Особенности структуры каталогов, поддерживаемой системой Windows рассмотрены ниже в п.1.3.1.5.

3-й бит = 1 Метка тома (***Vol*** - Volume)
4-й бит = 1 Элемент подкаталога (***Dir*** - Directory)
5-й бит = 1 Архивная копия файла НЕ создавалась (***Arc***)

Информация, хранимая в атрибутах, используется операционной системой при выполнении файловых операций. Например, значение атрибута ***Dir*** позволяет отличить файл от подчиненного каталога, а по значению атрибута ***Arc*** отбираются файлы для резервного копирования. Атрибут ***R/O*** блокирует возможность изменения и удаления файла, а атрибут ***Hid*** делает файл "невидимым" (команда DIR не выводит имя этого файла в списке). Если в записи корневого каталога установлено единичное значение атрибута ***Vol***, то поля "Имя" и "Расширение" этой записи (всего 11 байтов) будут трактоваться как метка тома, а остальные данные этой записи будут игнорироваться.

Время и дата создания файла (подкаталога) описываются двухбайтовыми блоками данных в следующих форматах :

Рисунок 1.1 Форматы полей "Дата" и "Время" в записи каталога

Подчиненные каталоги по своей структуре подобны корневому каталогу – то есть содержат множество 32-байтовых регистрационных записей.. В отличие от корневого, каждый подчиненный каталог имеет две служебных записи: первая запись подкаталога в поле "имя" содержит символ "**.**" (точка), а в поле "номер начального кластера" - ссылку "на самого себя"; вторая запись подкаталога с символами "**..**" (две точки) в поле "имя" – это ссылка на родительский каталог (если родительским является корневой каталог, то в качестве номера кластера указывается "0").

Все подчиненные каталоги хранятся в рабочей области диска подобно файлам. При создании подкаталога ему выделяется один кластер (из числа свободных), в котором создаются две служебных записи. После заполнения этого кластера регистрационными записями этому подкаталогу выделяется еще один свободный кластер и т. д.

Таблица распределения файлов (FAT) - это связанный список, который обеспечивает возможность фрагментарного расположения файлов на диске и используется файловой системой для определения последовательности кластеров, выделенных файлу или подкаталогу, а также для поиска свободного пространства, необходимого для записи новых файлов и подкаталогов. Число элементов этого списка равно числу кластеров рабочей области диска, так что каждому кластеру соответствует один "одноименный", то есть имеющий такой же номер, элемент FAT.

Каждому файлу или подкаталогу, записанному в рабочей области диска, соответствует *цепочка элементов FAT*. Номер начального элемента этой цепочки (совпадающий с номером начального кластера) указывается в соответствующей регистрационной записи каталога. Первый элемент цепочки указывает (содержит номер) на следующий элемент (кластер) и т.д. до конечного элемента, в котором указывается специальный код - признак конца файла.

Для гибких дисков используются 12-битовые элементы FAT, а для жестких – 16-ти и 32-битовые. Длина элемента FAT определяет разрядность хранимого в нем двоичного числа и, следовательно, ограничивает максимальное количество кластеров, которые могут быть сформированы в рабочей области тома (для FAT-12: $2^{12} = 4096$; для FAT-16: $2^{16} = 65536$; для FAT-32: $2^{32} = 4294967296$).

Каждый из *элементов FAT* представляет один кластер и может содержать следующие коды:

(0)000H	доступный (свободный) кластер
(0)002H до (f)fefH	номер следующего кластера
(f)ff0H до (f)ff7H	зарезервированный кластер
(f)ff7H	плохой кластер (BAD)
(f)ff8H до (f)fffH	конец цепочки (EOF)

Следует отметить, что первый байт FAT имеет особый статус и содержит так называемый "дескриптор носителя" (FAT ID-байт), в котором закодирована информация о типе и формате диска. Следующие 5 байтов (12-битовые FAT) или 7 байтов (16-битовые FAT) содержат код 0ffH. Далее следуют собственно элементы FAT.

Для иллюстрации основных концепций FAT рассмотрим фрагмент FAT, приведенный на рисунке 1.2 (*курсивом* обозначены номера элементов FAT, а жирным шрифтом – числовые значения, записанные в этих элементах). Все числа записаны в шестнадцатеричной системе счисления.

Пусть элемент некоторого каталога содержит регистрационную запись о файле, в которой содержится ссылка на начальный кластер (*№18*) из цепочки кластеров, выделенных этому файлу:

10	*11*	***12***	*13*	*14*	*15*	*16*	*17*	***18***	*19*	*1a*	*1b*	*1c*	*1d*	*1e*	*1f*
00	**00**	**13**	**14**	**15**	**Ff8**	**00**	**00**	**19**	**1a**	**1b**	**25**	**00**	**00**	**00**	**00**

00	**00**	**00**	**00**	**00**	**26**	**27**	**29**	**ff7**	**2a**	**2b**	**ff8**	**00**	**00**	**00**	**00**
20	*21*	*22*	*23*	*24*	*25*	*26*	*27*	*28*	*29*	*2a*	*2b*	*2c*	*2d*	*2e*	*2f*

Рисунок 1.2 Фрагмент FAT

Из приведенного на рисунке фрагмента FAT, в частности, следует:

- Файл занимает цепочку из десяти кластеров: *18 – 19 - 1a - 1b – 25 – 26 – 27 – 29 - 2a - 2b*. Каждый элемент указывает на следующий элемент цепочки, а последний элемент содержит специальный код EOF (**ff8**).

- Еще одна цепочка начинается с кластера №*12* и кончается кластером №*15*. Чтобы узнать, какому файлу (или подкаталогу) распределены эти кластеры, нужно отыскать в каком-либо каталоге тома регистрационную запись, содержащую ссылку на начальный кластер №12.

- Кластер №*28* помечен, как BAD (код **ff7**) и не входит ни в одну из цепочек. При поиске свободных кластеров для записи нового файла этот кластер будет игнорироваться.

- Кластеры №№ *10, 11, 16, 17, 1c…24 и 2c…2f* пусты (точнее – объявлены таковыми) и доступны для распределения под вновь записываемые файлы.

- Каждая цепочка кластеров, выделенных системой для одного файла (подкаталога), упорядочена в порядке возрастания их номеров.

1.3.1.4 Особенности структуры жесткого диска

Как уже отмечалось, жесткий диск ПК может быть разделен на несколько логических дисков (разделов, томов), каждый из них рассматривается как автономный диск. Каждый логический диск имеет собственную системную и рабочую области, структура которых описана выше, и может выступать в качестве загрузочного (системного) диска.

Для обеспечения процесса начальной загрузки операционной системы, а также для хранения данных о физическом расположении логических дисков, в первом секторе жесткого диска (0-й цилиндр, 0-я сторона, 1-й сектор) создается специальная информационная структура – *главная загрузочная запись* (*Master Boot Record,* MBR), содержащая *код программы начальной загрузки* и *таблицу разделов диска* (*Partition Table*).

Каждый раздел в таблице представлен одним 16-байтовым элементом, содержимое которого формируется программой форматирования жесткого диска (Fdisk или другой аналогичной системной утилитой). Для просмотра

таблицы разделов можно использовать утилиту DiskEdit. Редактирование таблицы разделов диска – крайне опасная процедура, в результате некорректного выполнения которой логические диски могут оказаться недоступными, что потребует полного переформатирования жесткого диска с потерей всех записанных на нем данных.

Таблица 1.6 - Структура таблицы разделов диска

Смещение	Длина, байт	Содержимое
1-й элемент (для первого раздела диска)		
+00	1	Флаг загрузки : 0 – не загружаемый, 80h – загружаемый (Bootable)
+01	1	Начало раздела : **HdS** (№ головки)
+02	2	Начало раздела :**Sec** (№ сектора - 6 младших битов) **Cyl** (№ цилиндра - 10 старших битов)
+04	1	Код системы: 0 – неизвестна; 1 – DOS (12 bit FAT); 4 - DOS (16 bit FAT)
+05	1	Конец раздела **:** **HdE** (№ головки)
+06	2	Конец раздела : **Sec** (№ сектора - 6 младших битов) **Cyl** (№цилиндра - 10 старших битов)
+08	4	Абсолютный номер начального сектора раздела (соответствует номерам сектора, головки и цилиндра начала раздела) : **Cyl** * сект./дор. * дор./цил. + **Hds** * сект./дор. + (**Sec** –1)
+0Ch	4	Число секторов раздела
2-й элемент (для второго раздела диска)		
+10h	1	Флаг загрузки
…	…	…
…	…	…
Последний элемент (после описания последнего раздела диска)		
		0AA55h

Процесс загрузки системы с жесткого диска начинается со считывания MBR в ОЗУ и передачи управления на код находящейся в MBR программы. Эта программа читает таблицу разделов диска и определяет первый из разделов, помеченный как *Bootable*. Затем в память считывается boot-сектор этого раздела и ему передается управление. Далее работает программа начальной загрузки, находящаяся в boot-секторе раздела, которая загружает все необходимые системные файлы.

1.3.1.5 Особенности хранения длинных имен файлов

В отличие от MS DOS, операционные системы семейства Windows поддерживают "длинные русские имена" файлов – то есть имена длиной до 255 символов, которые могут содержать весь набор 8-битовых символов ASCII (для Windows9x с файловой системой FAT-16 или FAT-32) или набор 16-битовых символов UNICODE (для Windows NT/XP с файловой системой NTFS). Для обеспечения совместимости Windows автоматически генерирует поддерживаемое MS-DOS короткое имя (имя 8 символов + расширение 3 символа) для каждого файла, преобразуя его настоящее длинное имя.

Правила формирования коротких имен файлов в стиле MS-DOS различны для FAT и NTFS.

В системе **FAT** короткое DOS-имя файла формируется по следующему алгоритму:

1. Из исходного длинного имени удаляются все запрещенные символы, точки (кроме одной) и пробелы.
2. Если в длинном имени присутствуют строчные русские буквы, они заменяются соответствующими прописными (заглавными).
3. Далее имя файла усекается справа до 6 символов, добавляется седьмой символ - тильда (~) и восьмой символ - цифра "1" (если отсутствует дубликат усеченного имени файла).
4. Если в одном каталоге имеется несколько файлов, в длинных именах которых первые 6 символов одинаковы, то у соответствующих им коротких имен будет изменяться последний символ (цифра 2, 3 и т.д.). Если таких "похожих" файлов в одном каталоге больше четырех, то первые 6 символов короткого имени определяются по специальному алгоритму, а окончание имени всегда одинаково: "~5".
5. Расширение имени файла усекается до 3 символов по такой же схеме.

Например, если файл имеет имя "*длинное имя файла.русское*", то его короткое имя "*ДЛИННО~1.РУС*". Если у какого-либо файла в этом каталоге первые 6 символов имени и три символа расширения окажутся такими же, например, "*длинное письмо.руслана*", короткий вариант будет выглядеть так: "*ДЛИННО~2.РУС*".

В системе **NTFS**, поддерживающей 16-битовое кодирование символов, короткое имя образуется по несколько иному алгоритму: за основу берутся не первые 6 первых символов длинного имени, а их эквивалент в UNICODE.

Короткое имя файла (11 байтов) хранится в полях "имя" и "расширение" основной регистрационной записи каталога, а длинное имя – в дополнительной записи (или нескольких дополнительных записях) этого же каталога.

1.3.2 Алгоритмы выполнения типовых файловых операций

Чтение файла с диска (например, при выполнении команды TYPE D:\TEXT\File.txt) реализуется следующей процедурой:

- В корневом каталоге логического диска D находится регистрационная запись, у которой: поле "имя" = TEXT, поле "расширение" = пусто, атрибут Dir = 1. В этой записи читается значение поля "номер начального кластера" (N).
- Читается первая копия FAT логического диска D и определяется цепочка номеров кластеров, (начинающаяся с N и оканчивающаяся EOF), выделенных каталогу TEXT.
- Читается каталог TEXT: последовательно считываются все кластеры этой "цепочки", начиная с N-ного; находится регистрационная запись, у которой: поле "имя" = File, поле "расширение" = txt, атрибут Dir = 0. В этой записи читается значение поля "номер начального кластера" (N1).
- Читается первая копия FAT диска D и определяется цепочка номеров кластеров, (начинающаяся с N1 и оканчивающаяся EOF), выделенных файлу File.
- Читается файл File: последовательно считываются все кластеры этой "цепочки", начиная с N1-го.

При записи на диск нового файла выполняется следующая процедура (приводится с упрощениями):

- По результатам анализа FAT определяется первый свободный ее элемент, номер которого записывается в очередную запись соответствующего каталога в качестве номера начального кластера.
- По данным таблицы формата диска, расположенной в его Boot-секторе, и известному размеру файла в байтах вычисляется количество кластеров, необходимых для размещения файла.
- Начиная с начального элемента FAT, определяется цепочка свободных элементов (содержащих "0"), расположенных в порядке возрастания их номеров. При этом в каждый элемент записывается номер следующего элемента цепочки, а в последний элемент записывается код (f)ff8H (EOF).
- Файл последовательно записывается в кластеры рабочей области диска, номера которых соответствуют номерам элементов FAT, включенным в цепочку.

Например (рисунок 1.2), при записи файла размером 1234 байта на дискету стандартного формата (512 байтов в секторе, 1 сектор в кластере), этому файлу будет выделено три кластера с номерами 10, 11 и 16. При этом последний (16-й) кластер будет занят лишь частично.

При удалении файла стандартными средствами (командой DEL):

- Рабочая область диска не модифицируется, то есть файл физически сохраняется в кластерах, выделенных ему при записи.

- Сохраняется также и регистрационная запись об этом файле в соответствующем каталоге, в том числе длина файла и ссылка на номер его начального кластера.

- Первый символ имени удаляемого файла в его регистрационной записи заменяется на код **E5**h (этот код соответствует строчной русской букве "х", недопустимой с точки зрения соглашения об именах MS DOS) – наличие такого символа в начале имени файла является признаком того, что файл логически удален.

- Обнуляется вся цепочка элементов FAT удаляемого файла, что делает соответствующие кластеры доступными для последующей записи.

Анализ рассмотренных выше процедур записи и удаления файлов показывает, что в ряде случаев сохраняется возможность восстановления удаленного файла (до тех пор, пока на его место физически не записан другой файл). Для восстановления логически удаленного файла достаточно изменить в регистрационной записи соответствующего каталога первый символ имени файла на любой из допустимых и восстановить в FAT цепочку номеров кластеров, занятых этим файлом.

Для повышения вероятности возможности восстановления удаленных файлов рекомендуется периодически дефрагментировать диск с помощью специальных программных утилит, которые перераспределяют дисковое пространство, закрепляя за файлами цепочки из непрерывно расположенных кластеров.

При переименовании файла изменяются только значения полей "имя" и "расширение" соответствующей регистрационной записи каталога.

Процедура ***копирования*** файла реализуется последовательным выполнением рассмотренных выше процедур чтения, записи и переименования.

Процедура ***перемещения*** файла между двумя каталогами одного логического диска сводится к перемещению соответствующей регистрационной записи. Если же файл перемещается на другой логический диск, то он "удаляется" с одного диска и "копируется" на другой.

1.3.3 Учебные задания

Методические указания

1. В качестве экспериментальной установки используется ПК, работающий в "командном режиме", объектом исследования является рабочая дискета (или логический диск), а основным инструментом - программная утилита Diskedit.exe, позволяющая просматривать в удобном для пользователя текстовом или шестнадцатеричном форматах и редактировать содержимое элементов рабочей и системной областей диска (секторы, кластеры, каталоги, Boot-record, FAT, MBR, PartitionTable).
2. Все рабочие файлы и каталоги, используемые при выполнении заданий, должны находиться либо на дискете, либо "внутри" личного каталога, созданного каждым студентом на рабочем логическом диске.

Задание 1. Выведите на экран содержимое Boot-сектора рабочего диска в текстовом формате и определите основные параметры форматирования диска. Просмотрите этот же boot-сектор в шестнадцатеричном формате. Определите:

- длину (в байтах) таблицы параметров форматирования диска;
- основные параметры формата диска, приведенные в таблице 1.4.

Задание 2. Просмотрите содержимое корневого каталога командой DIR и программой Diskedit. Определите для этого каталога:

- размер (в секторах и байтах);
- физическое расположение (номера занятых каталогом секторов);
- количество зарегистрированных в каталоге объектов, в том числе файлов, подкаталогов, скрытых файлов, меток томов
- количество удаленных файлов, файлов с длинными именами Windows.

Задание 3. Измените метку рабочего тома стандартными средствами (командой Label). Определите место (места) на диске, в котором записана метка. Измените метку тома "вручную" с помощью программы Diskedit. Оцените результат с помощью команды Vol. Предложите эффективный способ сокрытия от (неквалифицированного) пользователя факта наличия на диске большой группы файлов.

Задание 4. Просмотрите область FAT.

- Сколько копий FAT создано в системной области логического диска?
- Каков размер (в байтах и секторах) каждой копии FAT?
- Сколько "плохих", свободных и занятых кластеров на логическом диске?
- Определите несколько "цепочек" кластеров, выделенных объектам, записанным на логический диск.

Задание 5. Создайте (командой MD) рабочий каталог, подчиненный вашему личному каталогу. Просмотрите его содержимое командой DIR и программой Diskedit. Определите его размер (в кластерах) и место расположения на диске. Определите адрес (номер начального кластера) родительского каталога.

Задание 6. Скопируйте несколько файлов (с короткими английскими именами) в рабочий каталог, созданный при выполнении предыдущего задания. Изменился ли размер (в кластерах) этого каталога ? Сколько файлов можно зарегистрировать в этом каталоге, чтобы сохранить его минимальный размер ?

Задание 7. Скопируйте в рабочий каталог несколько файлов с длинными русскими именами. Просмотрите содержимое каталога командой DIR и программой Diskedit, определите места хранения длинного и короткого имен файла. Прокомментируйте результаты преобразования длинных имен.

Задание 8. Удалите (командой **Del**) один из файлов, зарегистрированных в рабочем каталоге. Прокомментируйте результат. Оцените возможность восстановления удаленного файла. Восстановите удаленный файл "вручную" (с помощью программы Diskedit).

Задание 9. Создайте еще один (пустой) рабочий каталог и удалите его командой **RD**. Прокомментируйте результат. Повторите ту же процедуру с непустым каталогом.

Задание 10. Исследуйте алгоритмы выполнения операций копирования, перемещения и переименования файлов. Подтвердите результатами экспериментов свои ответы на 7-й контрольный вопрос.

Задание 11. Просмотрите таблицу разделов диска (Partition Table). Определите:

- количество, размеры и адреса расположения логических дисков (разделов);
- количество системных (bootable) логических дисков и типы установленных на них операционных систем.

1.3.4 Контрольные вопросы

1 Поясните термины "*Сектор*" и "*Кластер*". С чем связаны ограничения на размер кластера ?

2 Как влияет разрядность элементов FAT на эффективность использования дискового пространства ?

3 Каково назначение второй копии FAT ?

4 Что изменится в системной и рабочей областях диска после завершения операции перемещения файла из одного каталога в другой (рассмотреть 2 случая: перемещение между каталогами одного логического диска и перемещение с одного логического диска на другой) ?

1.4 Контрольная работа №1

Задача 1.1. Определите термин "путь к файлу". Как реализуется ссылка на родительский каталог, если его имя явно не указано в команде ?

Задача 1.2. Подготовьте bat-файл в соответствии с вариантом, указанным преподавателем. Варианты заданий приведены в таблице 1.7.

Задача 1.3. Рассчитайте максимально-возможное количество кластеров рабочей области логического диска (FAT-12, FAT-16, FAT-32). Определите количество кластеров на дискете стандартного формата.

Задача 1.4. Определите максимально-возможное количество файлов, размещенных на дискете стандартного формата.

Задача 1.5. Опишите алгоритм работы процедуры восстановления файлов, удаленных стандартными средствами MS DOS.

Таблица 1.7 - Варианты контрольных заданий к задаче 1.2

Вариант	Список операций, выполняемых при запуске bat-файла	Модификации варианта	
		№	Описание
1	2	3	4
1	• Выбор из меню одной из трех операций : - просмотр группы файлов - редактирование файла - копирование группы файлов с изменением их расширений • Вывод сообщений о выполняемой операции • Запуск операции после того, как пользователем нажата любая клавиша	**1**	Каждая из трех операций выполняется отдельным bat-файлом, вызываемым из основного bat-файла
		2	Группа файлов задается их общим расширением TXT
		3	Группа файлов задается их общим именем MY_FILES
		4	Расширение группы файлов задается параметром
		5	Имя группы файлов задается параметром

Продолжение таблицы 1.7

1	2	3	4
2	• Создание в рабочем каталоге трехступенчатой системы подкаталогов • Копирование в каждый из них из рабочего каталога всех файлов, имеющих заданные расширения • Вывод оглавлений рабочего и созданных каталогов • Удаление из рабочего каталога тех файлов, копии которых созданы в других каталогах	**1**	Копирование файлов производится с использованием команды FOR.. IN.. DO
		2	Копирование файлов производится без использования команды FOR..IN..DO
		3	Каждая из трех операций выполняется отдельным bat-файлом, вызываемым из основного bat-файла
3	• Выбор из меню одной из четырех операций : - просмотр группы файлов - редактирование группы файлов - переименование группы файлов с заменой их расширений - переименование группы файлов с заменой их имен • Вывод сообщений о выполняемой операции • Запуск операции после того, как нажата любая клавиша	**1**	Каждая из трех операций выполняется отдельным bat-файлом, вызываемым из основного bat-файла
		2	Группа файлов задается их общим расширением (номер Вашей учебной гр.)
		3	Группа файлов задается их общим именем (Ваше имя по-английски)
		4	Расширение группы файлов задается параметром
		5	Имя группы файлов задается параметром

Окончание таблицы 1.7

1	2	3	4
4	• Выбор из меню одной из четырех операций : - просмотр группы файлов - редактирование группы файлов - переименование группы файлов с заменой их расширений - переименование группы файлов с заменой их имен • Проверка существования хотя бы одного файла заданной группы • Вывод сообщения о выполняемой операции или об отсутствии файлов в группе	**1**	Каждая из трех операций выполняется отдельным bat-файлом, вызываемым из основного bat-файла
		2	Группа файлов задается их общим расширением (параметр)
		3	Группа файлов задается их общим именем (параметр)
5	• Проверка введенного пользователем "пароля" (1-й параметр bat-файла) на соответствие одному из трех возможных значений • Если пароль введен верно - выбор из меню и запуск одной из трех программ • В противном случае – вывод сообщения о неправильно введенном пароле и завершение работы	**1**	"Пути" к каждой из трех программ задаются в тексте bat-файла
		2	"Пути" к каждой из трех программ задаются параметрами
		3	Перед запуском программы производится проверка наличия соответствующего файла
		4	При отсутствии программы в заданном каталоге – повторный выбор из меню

Примечание. При запуске bat-файла на экран должна выводиться **информационная заставка**, содержащая данные разработчика (например, ф.и.о. и номер группы), а также номер задания и номер варианта. Заставка должна храниться в отдельном текстовом файле и выводиться на экран соответствующей командой.

2 АДРЕСНОЕ ПРОСТРАНСТВО *ПК*

2.1 Лабораторная работа №4
ИССЛЕДОВАНИЕ СЛУЖЕБНЫХ ОБЛАСТЕЙ ОЗУ

Цель лабораторной работы - исследование информационной структуры областей оперативной памяти ПК, используемой операционной системой для хранения служебной информации (таблица векторов прерываний и область данных BIOS).

2.1.1 Линейные адреса

Минимальной адресуемой "порцией" памяти является ***байт*** – ячейка, содержащая 8-разрядный двоичный код. Все байты памяти последовательно пронумерованы: *порядковый номер байта* в адресном пространстве ПК называется его *линейным* (или физическим) *адресом*.

Диапазон линейных адресов, доступных процессору "напрямую", определяется шириной (разрядностью) адресной шины. В ранних IBM-совместимых ПК использовались 16-разрядные микропроцессоры фирмы Intel (i8086, i8088, i80186, i80188), оснащенные 20-разрядной адресной шиной. Такая шина способна передавать по своим линиям A0 – A19 адреса в диапазоне от **0000 0000 0000 0000 0000** до **1111 1111 1111 1111 1111** (или, в шестнадцатеричной системе счисления, от **00000**h до **FFFFF**h), что обеспечивает доступ к адресному пространству размером в 1048576 байтов (2^{20}, или 1Мб).

За один цикл обмена с памятью центральный процессор передает (принимает) по шине данных одно ***машинное слово***, разрядность которого определяется разрядностью регистров процессора и шириной шины данных. В 16-разрядных процессорах используются двухбайтовые машинные слова, каждое из которых содержит 16-разрядное двоичное число, сформированное из содержимого пары "соседних" байтов, линейные адреса которых отличаются на 1. При этом в Intel-совместимых компьютерах 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 – D15) машинного слова.

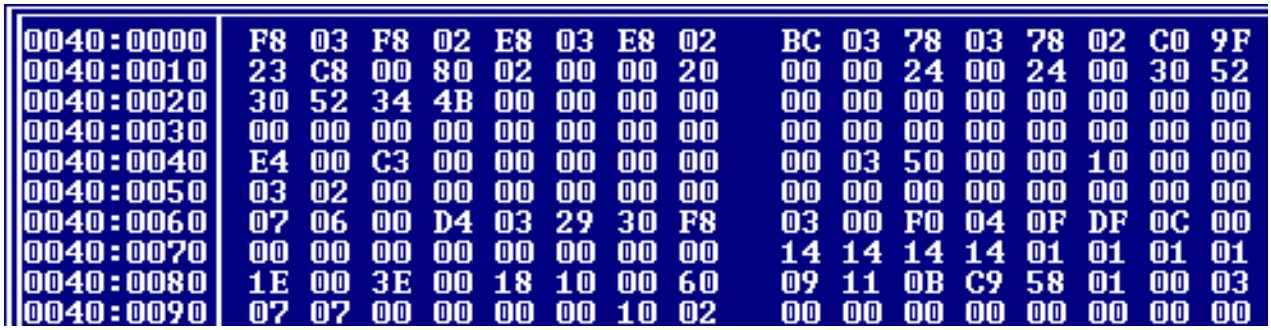

Рисунок 2.1 Фрагмент адресного пространства

На рисунке 2.1 показан экранный образ фрагмента памяти (в 16-ричном формате). Байт с линейным адресом **00400**h содержит число **F8,** байт **00401**h - число **03**, байт **00402**h - число **F8**, байт **00403**h - число **02.** При этом первая пара байтов образует машинное слово **03F8**, а вторая – **02F8**. Линейные адреса соседних машинных слов отличаются на 2.

2.1.2 Сегментная организация памяти

Для представления 20-разрядного линейного адреса в процессоре Intel 8086 используется пара 16-разрядных адресных регистров: один из них (называемый *базовым* или *сегментным* регистром) хранит **номер сегмента** памяти, а второй – *регистр адреса команды* **IP** (**I**nstruction **P**ointer) – хранит **смещение** относительно начала этого сегмента. Для хранения номеров сегментов могут использоваться различные специализированные *сегментные регистры*, например, регистр **CS** (**C**ode **S**egment) содержит номер сегмента программного кода, регистр **SS** (**S**tack **S**egment) – номер сегмента стека, а регистр **DS** (**D**ata **S**egment) – номер сегмента данных.

20-разрядный линейный адрес *вычисляется* в *сумматоре адреса* - специализированном аппаратном устройстве, входящем в состав центрального процессора. Сумматор адреса выполняет операцию сложения со сдвигом над двумя операндами, содержащимися в соответствующем сегментном регистре и регистре смещения, по следующей формуле:

$$\text{<Линейный адрес>} = \mathbf{CS} * \left\{ \begin{array}{c} 10000_2 \\ 10_{16} \\ 16_{10} \end{array} \right\} + \mathbf{IP}$$

При выполнении первой операции содержимое сегментного регистра (номер сегмента) сдвигается на 4 двоичных разряда влево (т.е. умножается на 16_{10}) – в результате получается 20-битовый ***базовый адрес*** **- линейный адрес начала сегмента**, номер которого указан в **CS**. После сложения базового адреса сегмента с содержимым регистра смещения **IP** на выходе сумматора формируется полный 20-битовый линейный адрес, который и передается на адресную шину[7].

Любой линейный адрес может быть записан двумя 16-битовыми числами в так называемой сегментной форме: *<номер сегмента>*:*<смещение>*. На рисунке 2.1 левый столбец содержит сегментные адреса в 16-ричном формате.

[7] Все сказанное здесь о сегментной организации памяти справедливо лишь для микропроцессоров i8086. В более поздних процессорах фирмы Intel увеличена разрядность адресной шины и используется другая (более эффективная) система сегментации адресного пространства и вычисления физического адреса. Микропроцессоры фирмы Motorola, используемые в персональных компьютерах Apple Macintosh, вообще не используют механизм сегментации памяти.

Учитывая приведенную выше формулу, а также тот факт, что все адресные регистры рассматриваемого центрального процессора – 16-разрядные, можно сделать следующие выводы:

1 Процессор может поддерживать не более 65536 различных сегментов памяти.

2 Размер каждого сегмента не может превышать 65536 байтов (64 Кб).

3 Сегменты выровнены по 16-байтовым границам, то есть их базовые адреса кратны 16 (это, в частности, означает, что если номера сегментов отличаются на "1", то их базовые адреса отличаются на "16").

4 Сегменты (если их длина превышает 16 байтов) могут пересекаться в физическом адресном пространстве, то есть один и тот же физический адрес может быть представлен различными сегментными адресами: например, **0000:0408**h эквивалентно **0040:0008**h, так как оба этих сегментных адреса определяют один и тот же физический адрес **00408**h.

5 В процессорах с 20-разрядной адресной шиной максимальное значение физического адреса составляет **FFFFF**h, что обеспечивает размер адресного пространства в 1 Мб. Однако, используемый в этих процессорах способ формирования физического адреса путем суммирования базового адреса сегмента со смещением позволяет определить линейные адреса, находящиеся за пределами 1-мегабайтной границы. При максимальных значениях номера сегмента и смещения сегментный адрес **FFFF:FFFF** соответствует линейному адресу **10FFEF**h, что "расширяет" адресное пространство почти на 64 килобайта и требует увеличения на "1" разрядности адресной шины. В процессоре i8086 такая ситуация обрабатывается путем игнорирования старшего бита вычисленного линейного адреса, который для рассмотренного выше примера будет иметь значение **0FFEF**h. В процессоре i80286 (из-за ошибки разработчиков!) программам оказалась доступной 21-я линия адресной шины (A20), что позволило реально расширить линейно-адресуемую область памяти до 1062 Мб и использовать так называемую "верхнюю" память (HMA - High Memory Area). В процессорах i80386 и выше возможна аппаратная блокировка линии A20 адресной шины.

2.1.3 Информационная структура адресного пространства

Базовая память (convertational memory)

Занимает младшие ***640 Кбайт*** адресного пространства (с адресами от **0000:0000** до **9000:FFFF**h). Это оперативная память (ОЗУ, RAM – Random Access Memory – "память с произвольным доступом"), доступная центральному процессору как для чтения, так и для записи. В базовую память загружаются системные и пользовательские программы и обрабатываемые ими данные.

Ряд фиксированных адресов в начале базовой памяти зарезервированы для хранения служебной информации, используемой операционной си-

стемой (таблица векторов прерываний, область данных BIOS, данные и программные компоненты DOS, драйверы внешних устройств, резидентные программы), а вся остальная базовая память доступна программам MS DOS.

Более детальные сведения об использовании базовой памяти операционной системой MS DOS приведены в таблице 2.1.

Таблица 2.1 - Распределение базовой памяти

<table>
<tr><th>Начальный адрес</th><th>Длина</th><th>Назначение области памяти</th></tr>
<tr><td>0000:0000</td><td>1 Кб</td><td>Таблица векторов прерываний</td></tr>
<tr><td>0000:0400</td><td>256 б</td><td>Область данных ROM-BIOS (таблица 3.3)</td></tr>
<tr><td>0000:0500</td><td>~ 128 Кб</td><td>Область DOS :</td></tr>
<tr><td colspan="3">

Надстройка над ROM BIOS: программы обработки прерываний, обслуживающие системный ввод/вывод - считываются из файла IO.sys при загрузке MS DOS.
Обработчики прерываний DOS (INT 21h) - считываются из файла MSDOS.sys при загрузке MS DOS (в системах Windows файл MSDOS.sys не используется, обработчики прерываний DOS хранятся в файле IO.sys).
Стеки MS DOS – используются программами обработки прерываний MS DOS. Число стеков и их размеры задаются параметрами команды STACK в конфигурационном файле CONFIG.sys.
Переменные окружения операционной системы – задаются командами SET, PATH, PROMPT и др. Размер области системного окружения задается командой SHELL в файле CONFIG.sys.
Буфера ввода – вывода дисковых накопителей. Число буферов задается командой BUFFERS в файле CONFIG.sys, на каждый буфер выделяется 532 байта.
Дескрипторы открытых файлов. Каждый дескриптор содержит таблицу доступа к файлу, используемую файловой системой для определения аппаратного адреса (цилиндр, головка, сектор) дисковой страницы (кластера) по номеру байта, запрашиваемого прикладной программой. На каждый дескриптор отводится по 64 байта, число дескрипторов задается командой FILES в файле CONFIG.sys.
Программы – драйверы дополнительных устройств, устанавливаемые командой DEVICE в файле CONFIG.sys.
Резидентная часть командного процессора COMMAND.COM (около 4Кбайт).
Резидентные программы, загружаемые командой INSTALL в файле CONFIG.sys.

</td></tr>
<tr><td></td><td>~512 Кбайт</td><td>Выполняемая прикладная программа</td></tr>
</table>

Видеопамять (Video RAM)

Следующие ***128 Кб*** (от **A000:0000**h до **B000:FFFF**h) занимает специализированная видеопамять, доступная как центральному процессору (чтение и запись), так и видеоадаптеру (только чтение), и используемая для временного хранения выводимой на экран информации. Видеоадаптер периодически сканирует эту область памяти и вырабатывает соответствующий её содержимому набор сигналов управления дисплеем, который, в свою очередь, формирует "экранный образ" фрагмента видеопамяти[8].

Постоянная память

Следующие ***256 Кбайт*** (от **C000:0000**h до **F000:FFFF**h) зарезервированы для ***постоянной памяти*** (ПЗУ, ROM – Read Only Memory), используемой для хранения неизменяемых программ и данных. В ПЗУ хранится программа самотестирования компьютера POST (Power_On_Self_Testing), автоматически запускаемая при включении питания, программа начальной загрузки MS DOS, компоненты базовой системы ввода-вывода ROM BIOS, другие программы, таблицы и идентификаторы. Распределение адресов ПЗУ в большой степени зависит от фирмы-производителя и модели ПК.

Расширенная память

Ячейки памяти, расположенные за пределами одно-мегабайтового адресного пространства, образуют так называемую "Расширенную (Extended) память". В MS DOS управление этой памятью осуществляет специальный программный драйвер (например, Himem.sys или Quemm386.sys). В расширенной памяти выделяется зона обмена данными с внешним микропроцессором и зона ***теневой BIOS*** (Shadow RAM), в которую копируются программы из системного ПЗУ и плат расширения (так как доступ к ПЗУ осуществляется существенно медленнее по сравнению с ОЗУ). В остальной части расширенной памяти могут размещаться прикладные и системные программы.

2.1.4 Таблица векторов прерываний

Понятием "прерывание" обозначают механизм, обеспечивающий быструю реакцию компьютера на определенное внутреннее или внешнее событие, требующее прерывания выполнения исполняемой программы и перехода к выполнению другой специальной программы, называемой *про-*

---

[8] Современные видеоадаптеры SVGA, обеспечивающие высокое разрешение изображения, требуют существенно больших объемов видеопамяти (превышающих 4Мбайт). При этом видеопамять занимает в адресном пространства ПК те же 128 Кбайт, а остальная ее часть размещена на плате видеоадаптера и организована постранично – по 128 Кбайт на каждую страницу. Управляет постраничным выделением этой памяти непосредственно видеоадаптер.

граммой обработки прерывания. После завершения обработки прерывания происходит возврат к выполнению прерванной программы.

Процессор способен обработать до 256 различных прерываний. Все возможные события, требующие описанной выше реакции, пронумерованы от **0** до **100**h (*номер прерывания*), и с каждым из них связана определенная программа обработки прерывания. Часть номеров прерываний зарезервированы системой, остальные предоставлены разработчику.

Для запуска программ - обработчиков прерываний используется специальная система указателей на их начальные адреса, называемая *таблицей векторов прерываний*. Таблица векторов прерываний формируется в ОЗУ и занимает первый килобайт адресного пространства (с **0000:0000** до **0000:3FF**Fh). Таблица состоит из 256 четырехбайтовых элементов – *векторов прерываний*. Каждый из элементов описывает один вектор (адрес) в формате <сегмент>:<смещение>, причем в младшем двухбайтовом слове вектора записывается смещение, а в старшем - номер сегмента.

Вектора в таблице расположены в порядке возрастания номеров прерываний: нулевому прерыванию соответствует вектор, записанный по адресу 0000:0000, первому - 0000:0004 и т.д. Таким образом, номер прерывания **n**, умноженный на 4, однозначно указывает на точку входа в таблицу векторов прерывания (**0000:4*n** – адрес расположения вектора), в которой, в свою очередь, располагается указатель (адрес) на программу обработки этого прерывания[9].

Таблица векторов прерываний частично инициализируется BIOS перед началом загрузки DOS, частично - при загрузке DOS. Пользовательские программы также могут модифицировать эту таблицу, переназначая некоторые из векторов прерываний (например, программы русификации клавиатуры переключают "на себя" 9-е прерывание).

По типу запроса различают *аппаратные* и *программные* прерывания. Аппаратные прерывания, в свою очередь, могут быть *внешними* или *внутренними*.

Внешнее аппаратное прерывание является следствием наступления некоторого события, асинхронного относительно тактовой частоты системы, например: прочитан очередной сектор диска, нажата (или отпущена) клавиша клавиатуры или кнопка мыши, принтер готов к вводу очередного символа, сработал датчик внешнего оборудования, управляемого компьютером и т.д.

Запросы центральному процессору на обработку внешних аппаратных прерываний генерируются *контроллером прерываний*, который принимает от внешнего источника сигнал прерывания, формирует на шине управле-

[9] Вектор не обязательно указывает на программу обработки прерывания, он может служить указателем на определенную информационную структуру, размещенную в памяти ПК. Например, вектор прерывания **1E**h указывает на начальный адрес области памяти, в которой расположена таблица параметров дискеты, а вектор прерывания **1F**h – на графическую таблицу (знакогенератор) для символов с кодами ASCII 128 – 255.

ния *запрос на прерывание* и на шине данных – *номер прерывания* (число в диапазоне от 0 до 256), соответствующий номеру линии контроллера, на которую поступил сигнал от внешнего источника прерывания.

Система приоритетов прерываний регулирует ситуации, когда от внешнего оборудования очередной сигнал прерывания поступает в процессе обработки предыдущего прерывания. С некоторыми упрощениями эта система работает следующим образом:

- Из двух одновременно поступивших прерываний первым будет обрабатываться прерывание с высшим приоритетом, и только по завершению его обработки – второе прерывание.
- Если очередное прерывание поступает в процессе обработки прерывания с высшим приоритетом, оно ставится в очередь и будет обработано только после завершения обработки всех поступивших прерываний, имеющих высший приоритет.
- Если в процессе обработки прерывания поступает очередное прерывание с высшим приоритетом, то будет приостановлен процесс обработки первого прерывания (с сохранением в стеке всех необходимых параметров), будет запущен процесс обработки поступившего прерывания, и после его завершения продолжится процесс обработки приостановленного прерывания.

Приоритет прерывания связан с *номером линии* (*уровня*) *прерывания* контроллера, к которой подключен внешний источник прерывания (не путать с *номером прерывания* !). Все уровни прерываний последовательно пронумерованы и сокращенно обозначаются **IRQ<m>**, где **m** – номер уровня.

Внутреннее аппаратное прерывание является следствием каких-либо проблем, возникших при выполнении программы (например, произошло переполнение или деление на ноль при выполнении процессором арифметической операции). Запросы на обработку таких прерываний генерируются внутренними блоками процессора.

Программное прерывание является "искусственным", оно инициируется специальной командой процессора **INT n**, где **n** - номер прерывания. По сути дела, программное прерывание реализует специфический способ запуска программ с использованием механизма, созданного для обработки аппаратных прерываний.

Программные прерывания используются для организации доступа из прикладной программы к программным модулям общего пользования – например, к функциям DOS (**INT21**h), или к функциям BIOS для работы с клавиатурой (**INT16**h) или видеосистемой (**INT10**h). Для запуска такого модуля нет необходимости знать адрес его расположения в памяти ПК, достаточно знать номер прерывания, связанного с этим модулем, и номер соответствующей функции.

Процедура обработки прерывания. Если запрос прерывания принят процессором на обслуживание, выполняется следующая процедура:

- текущий адрес исполняемой программы и содержимое флаговых регистров процессора сохраняются в стеке, что обеспечивает возможность возврата к прерванной программе;
- в процессор вводится номер прерывания, указывающий на точку входа в таблицу векторов прерываний;
- вектор прерывания из таблицы вводится в соответствующие адресные регистры процессора, и выполняется программа обработки прерывания;
- по завершению обработки прерывания восстанавливается (из стека) старое состояние процессора, и прерванная программа продолжает выполняться с соответствующей команды.

2.1.5 Область данных BIOS

256-байтовая область ОЗУ (0000:0400h ... 0000:04FFh) выполняет функции "адресного справочника" и рабочего поля для процедур ввода-вывода ROM BIOS.

В таблице 2.2 приведена схема распределения этой области данных.

Таблица 2.2 - Схема распределения области данных ROM_BIOS

Начальный адрес	Длина, байт	Назначение
1	*2*	*3*
0000:0400	2	Базовый адрес порта первого адаптера последовательного канала RS-232 (COM1)
0000:0402	2	То же для COM2
0000:0404	2	То же для COM3
0000:0406	2	То же для COM4
0000:0408	2	Базовый адрес порта для 1-го адаптера параллельного канала Centronix (LPT1)
0000:040A	2	То же для LPT2
0000:040C	2	То же для LPT3
0000:040E	2	То же для LPT4
0000:0410	2	Список установленного оборудования
0000:0413	2	Общая память (в килобайтах)
0000:0417	2	Флаги клавиатуры
0000:0419	1	Текущее (накопленное) значение ввода кода символа (**Alt** + цифра)
0000:041A	2	Адрес "головы" буфера клавиатуры
0000:041C	2	Адрес "хвоста" буфера клавиатуры
0000:041E	20h	Буфер клавиатуры
0000:0449	1	Номер текущего видеорежима
0000:044A	2	Число символов в строке

Окончание таблицы 2.2

1	*2*	*3*
0000:044C	2	Размер видеостраницы (в байтах)
0000:044E	2	Начальный адрес (смещение в видео-сегменте) активной видеостраницы
0000:0450	8 * 2	Координаты курсора в каждой из 8 страниц: младший байт – номер колонки, старший байт – номер строки.
0000:0460	2	Размер курсора : младший байт - последняя строка; старший байт – начальная строка.
0000:0462	1	Номер текущей активной видеостраницы
0000:046C	4	Счетчик "тиков" таймера (текущее время в 55-миллисекундных единицах)

2.1.6 Адресное пространство ввода-вывода

Особое место занимает адресное пространство ввода-вывода, используемое для связи центрального процессора с периферийными устройствами. Подключение периферийного оборудования (клавиатуры, дисплея, дисковых накопителей, принтера, "мыши" и т.д.) и обмен данными с центральным процессором осуществляется через специальные устройства, называемые *адаптерами* или *контроллерами*. В состав компьютерной системы входят как специализированные адаптеры (например, видеоадаптер, контроллер прерываний, контроллер дисковода и др.), так и универсальные адаптеры последовательного (COM1...COM4) и параллельного (LPT1...LPT4) каналов, к которым может подключаться различное программно-управляемое оборудование.

Адаптер содержит несколько информационных регистров (как правило, однобайтовых), через которые и происходит обмен данными между процессором и внешним устройством. Множество регистров всех подключенных адаптеров образуют адресное пространство ввода-вывода, доступ к которому со стороны центрального процессора осуществляется по 16-разрядной шине.

Множество всех регистров одного адаптера объединены понятием *порт ввода-вывода*, первый регистр порта называется *базовым регистром*, а его адрес в адресном пространстве ввода-вывода называется *базовым адресом порта*. Базовые адреса установленных портов определяются в процессе начального тестирования системы и записываются в ОЗУ (в область данных BIOS). Обращение к остальным регистрам порта производится по смещению относительно его базового адреса.

2.1.7 Учебные задания

Методические указания

1 Для исследования информационной структуры адресного пространства ПК необходимо использовать одну из программ – анализаторов памяти - рекомендуется использовать программу *Peek Poke resident* (файл PEEK.com) или функционально подобную ей программу *FX Show* (файл FXShow.com). Указанные программы позволяют просматривать на экране содержимое памяти в шестнадцатеричном или символьном форматах, изменять содержимое памяти "вручную" с помощью простого встроенного редактора, а также копировать в дисковый файл экранный образ фрагмента памяти.

2 Полезно использовать также резидентный калькулятор (CALC.com), позволяющий выполнять арифметические действия над целыми числами в десятичной и шестнадцатеричной системах счисления. После запуска калькулятора его можно активизировать нажатием комбинации клавиш Alt+ScrollLock или непосредственно из программы PEEK нажатием функциональной клавиши F9.

3 Более подробная информация о структуре служебных областей памяти ПК содержится в электронном справочнике HELP (файлы Help.exe и Help.dat).

Задание 1. Определите линейные адреса базовой памяти, соответствующие следующим адресам, заданным в сегментной форме: **0000:041E**h; **0040:001E**h; **0000:0408**h; **0000:0408**h. Результаты представьте в шестнадцатеричной и двоичной системах счисления. Поясните алгоритм работы сумматора адреса.

Задание 2. Оцените количество вариантов записи одного линейного адреса в сегментной форме.

Задание 3. Определите содержимое старшего и младшего байтов двухбайтового машинного слова, расположенного по адресу **0000:041E**h. Запишите это машинное слово в шестнадцатеричной и двоичной системах счисления. Нажимая (не менее 16 раз) на любую из символьных клавишей клавиатуры, проследите за изменениями этого машинного слова. Какая информация содержится в машинном слове по указанному адресу?

Задание 4. Определите состав параллельных и последовательных адаптеров (LPT* и COM*), установленных в Вашем компьютере, и их базовые адреса. Поясните термины: *адресное пространство ввода-вывода*, *адаптер*, *порт ввода-вывода*, *базовый адрес*.

Задание 5. Поясните термины: *"аппаратное прерывание", "программное прерывание", "внешнее прерывание", "внутреннее прерывание", "номер прерывания", "вектор прерывания", "обработчик прерывания".*

Задание 6. Опишите структуру *таблицы векторов прерываний* и алгоритм определения адреса программы обработки **N**-го прерывания.

Задание 7. Определите начальные адреса программ обработки прерываний №8, №9, №10h, №16h и №21h. Используя электронный справочник *HELP*, определите назначение этих программ.

Задание 8. Определите начальный адрес программы обработки прерывания клавиатуры (№9). Выведите на экран *карту памяти*. Определите, какие из программ загружены в память вашего ПК и какие из прерываний "перехвачены" этими программами.

Задание 9. Определите вектора прерываний №1Fh и № 43h. Используя электронный справочник *HELP*, определите назначение этих векторов.

2.1.8 Контрольные вопросы

1 Как влияет разрядность процессора и адресной шины на размер адресного пространства ? Определите диапазон линейных адресов для процессоров с 16-, 24- и 32- разрядной адресными шинами.

2 Какую функцию выполняют сегментные регистры процессора ?

3 Поясните назначение и алгоритм работы сумматора адреса.

4 Каковы действия DOS при поступлении сигнала аппаратного прерывания в процессе обработки предыдущего прерывания ?

5 Каковы возможные последствия программной модификации байтов данных, расположенных по следующим адресам :

0000:0014*h*, 0000:0408*h*, 0040:0008*h*, 0000:0024*h*, C000:FFFF*h*.

2.2 Лабораторная работа № 5
КЛАВИАТУРА ПК

Цель лабораторной работы – исследование алгоритма реализации процесса ввода данных с клавиатуры и информационной структуры области данных BIOS, обеспечивающей этот процесс.

2.2.1 Алгоритм ввода данных с клавиатуры

Клавиатура персонального компьютера - одно из важнейших его внешних устройств, предназначенное для кодирования и ввода буквенно-цифровой и управляющей информации.

Клавиатура является программно-управляемым устройством и имеет в своем составе специализированный контроллер (однокристальную микроЭВМ), основной функцией которого является отслеживание фактов нажатия и отпускания клавиш путем циклического сканирования наборного поля. Сканирование производится с периодичностью 10 раз в секунду, в каждом цикле формируется *скан-код* нажатой (или отпущенной) клавиши, который временно сохраняется во внутреннем буфере контроллера и затем передается в соответствующий порт ввода-вывода, доступный прикладным программам.

Скан-код - это уникальный однобайтовый номер, присваиваемый каждой клавише[10]. Семь младших битов скан-кода – это собственно код клавиши, а старший бит используется для кодирования факта её нажатия (0) или отпускания (1). Такая система позволяет закодировать 128 различных клавиш (коды с "0" по "127" при нажатии и с "128" по "255" - при отпускании), что превышает потребности стандартной 101-клавишной клавиатуры.

Клавиатура подключена к линии IRQ1 контроллера прерываний, которой соответствует прерывание №9, и к порту ввода-вывода с адресом 60h, в который помещается скан-код последней нажатой (или отпущенной) клавиши. В процессе обмена данными с клавиатурой участвует также порт 61h, доступный как для чтения, так и для записи. Запись "1" в старший бит этого порта блокирует обмен данными с клавиатурой.

Дальнейшая реализация процесса ввода данных осуществляется программным обработчиком клавиатурного прерывания, который читает принятый в порт 60h скан-код клавиши, анализирует его значение и формирует соответствующие информационные структуры в области данных BIOS (таблица 2.3).

[10] Следует отметить, что скан-коды клавиш определяются схемой распайки матрицы наборного поля клавиатуры и напрямую не связаны с обозначениями, нанесенными на поверхность клавиш. Соответствие между скан-кодом клавиши и кодом связанного с ней символа определяется программно обработчиком клавиатурного прерывания с учетом состояния управляющих клавиш.

Таблица 2.3 - Блок данных BIOS, обслуживающий ввод с клавиатуры

Начальный адрес (hex)	Длина, байт	Назначение
0040:0017	2	Флаги клавиатуры
0040:0019	1	Текущее (накопленное) значение вводимого ASCII-кода символа (**Alt** + цифра)
0040:001A	2	Адрес "хвоста" буфера клавиатуры
0040:001C	2	Адрес "головы " буфера клавиатуры
0040:001E	20h	Буфер клавиатуры (16 двухбайтовых элементов)

2.2.2 Флаги клавиатуры

Эти два байта в области данных BIOS формируются обработчиком прерывания №9 при нажатии и отпускании управляющих клавиш и клавиш-переключателей. Определенные биты этих флаговых байтов устанавливаются в "1" или "0" в зависимости от текущего состояния соответствующих клавиш или же от того, установлен или нет соответствующий режим.

Например, при нажатии клавиши <CapsLock> 6-й бит первого и второго флаговых байтов будет установлен в "1". При отпускании этой клавиши 6-й бит второго байта будет обнулен, а первый байт останется без изменений, что соответствует ситуации "установлен верхний регистр". После повторного нажатия и отпускания этой клавиши оба флаговых бита вернутся в исходное (нулевое) состояние.

В приведенных ниже диаграммах показана информационная структура флагов клавиатуры.

Рисунок 2.2 Структура 1-го "флагового" байта (0040:0017)

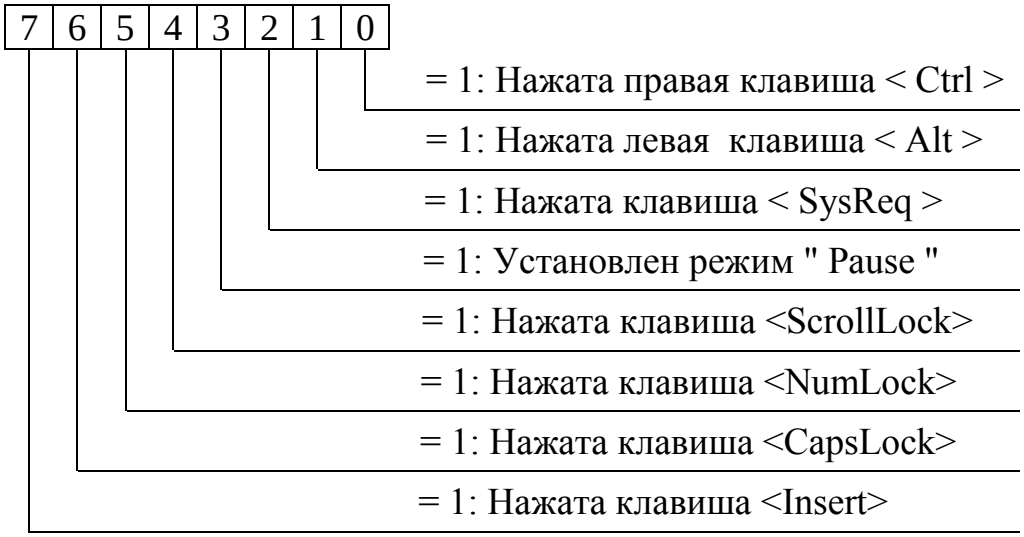

Рисунок 2.3 Структура 2-го "флагового" байта (0040:0018)

2.2.3 Буфер клавиатуры

Буфер клавиатуры - это линейная область ОЗУ, в которую программа-обработчик прерывания №9 записывает данные об очередной нажатой символьной клавише, а прикладные программы считывают эти данные для дальнейшего использования. Для каждого нажатия клавиши в буфере зарезервировано по два байта: один (младший) байт для скан-кода клавиши, другой (старший байт) - для ASCII-кода символа, соответствующего этой клавише.

В первых PC/XT поддерживался буфер клавиатуры фиксированной длины 32 байта, который располагался в диапазоне адресов с 0040:001Eh по 0040:003Eh. Такой буфер рассчитан на регистрацию 16 последовательных нажатий клавиш.

В PC/AT адрес начала буфера (смещение относительно сегмента 0040h) указан в ячейке 0000:0480h, а адрес конца буфера - в ячейке 0000:0482h. Обычно эти ячейки содержат значения соответственно 001Eh и 003Eh, при этом размер и расположение буфера клавиатуры PC/AT аналогичны PC/XT.

При нажатии символьной клавиши обработчик прерывания вычисляет значение кода ASCII по ее скан-коду (с учетом текущего состояния флагов клавиатуры) и записывает эти коды в два соседних байта буфера.

Процесс записи данных в буфер клавиатуры и чтения данных из буфера регулируется двумя указателями. Двухбайтовое слово по адресу 0040:001Ch содержит адрес (смещение относительно сегмента 0040h) "головы" буфера, то есть указывает обработчику прерывания, куда следует *записать* пару кодов очередной нажатой клавиши, а двухбайтовое слово по адресу 0040:001Ah содержит адрес "хвоста" буфера, то есть указывает прикладной программе, откуда следует *прочитать* пару кодов первой из еще не прочитанных клавиш.

В исходном состоянии значения этих указателей одинаковы и равны 001Eh. После завершения каждой операции записи данных в буфер программа-обработчик 9-го прерывания инкрементирует (увеличивает на два) значение "головы", а программа, читающая буфер, соответственно инкрементирует значение "хвоста", – таким образом, при вводе данных с клавиатуры хвост буфера постоянно "догоняет" голову. Буфер клавиатуры организован циклически, то есть при достижении соответствующим указателем конца буфера этот указатель программно устанавливается в начало буфера.

Если прочитаны все записанные в буфер данные – указатели головы и хвоста совпадают, что является признаком "пустого" буфера. Буфер клавиатуры может оказаться "переполненным" – такая ситуация создается в случае, если прикладная программа не прочитала данные из буфера о 16 последних нажатых клавишах, и, следовательно, хвост "отстает" от головы на длину буфера (с учетом его циклической организации). Попытка ввода данных об очередной нажатой клавише в переполненный буфер будет заблокирована.

При ***прямом вводе*** кода символа пользователь, удерживая нажатой клавишу <Alt>, набирает ASCII-код символа (в десятичной системе счисления) на дополнительной цифровой клавиатуре. При этом обработчик 9-го прерывания последовательно суммирует вводимый пользователем код (с учетом разряда числа) и сохраняет промежуточные результаты суммирования в ячейке 0040:0019h, пока клавиша <Alt> остается нажатой. После отпускания клавиши <Alt> результат суммирования переписывается из ячейки 0040:0019h в буфер клавиатуры, а сама эта ячейка обнуляется. Естественно, в этом случае соответствующий байт скан-кода в буфере клавиатуры остается нулевым.

Факты нажатия управляющих клавиш <Shift>, <Ctrl>, <Alt>, <ScrollLock>, <NumLock>, CapsLock>, <SysReq> или <Pause>, а также факты отпускания любой клавиши никак не отражаются на состоянии буфера клавиатуры. При нажатии других "несимвольных" клавиш в буфер попадает только скан-код, а байт кода символа остается нулевым.

2.2.4 Учебные задания

Методические указания

- Установите режим "**командная строка**" (сеанс MS DOS).
- Загрузите программы Calc.com, Peek.com и Help.exe (в указанном порядке).
- Найдите в электронном справочнике Help разделы, описывающие ASCII-коды символов и Scan-коды клавиш. Ознакомьтесь с содержимым указанных разделов.
- Не "закрывая" программы Help, активизируйте программу Peek (тройным нажатием *левой* клавиши <**Alt**>) и приступайте к выполнению учебных заданий.
- В любой момент Вы сможете переключиться на работу с программой Help, нажав клавишу <Esc>, и затем вновь активизировать программу Peek тройным нажатием клавиши <Alt>.

Задание 1. Исследуйте процесс изменения состояния "флаговых" байтов в процессе манипулирования управляющими клавишами Shift, Ctrl, Alt, CapsLock, NumLock. По результатам эксперимента составьте таблицы, аналогичные приведенным на рисунках 1 и 2. Результаты представьте в шестнадцатеричном и двоичном форматах.

Задание 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.

Задание 3. Определите расположение буфера клавиатуры и указателей на его "голову" и "хвост" в адресном пространстве Вашего компьютера. Запишите текущее состояние этих указателей в шестнадцатеричной системе счисления. Запишите адреса (в сегментной форме) "ячеек" буфера клавиатуры, в которые будут записаны и из которых будут прочитаны Scan-код и ASCII-код очередной символьной клавиши.

Задание 4. Введите символы "G", "g", "П" и "п". Определите состояние указателей головы и хвоста буфера клавиатуры после каждого нажатия на клавишу. Определите ASCII-коды введенных символов и Scan-коды соответствующих клавиш. Результаты представьте в шестнадцатеричной, десятичной и двоичной системах счисления. Выполните аналогичные действия для другой символьной клавиши.

Задание 5. Введите символы, указанные в предыдущем задании, прямым набором их ASCII-кодов при нажатой клавише <Alt>. Зафиксируйте промежуточные состояния ячейки с адресом 0040:0019h после ввода каждой очередной цифры ASCII-кода символа, а также состояние этой ячейки и соответствующих ячеек буфера клавиатуры после отпускания клавиши <Alt>. Прокомментируйте результаты.

Задание 6. Введите прямым набором символ с ASCII-кодом "1234". Прокомментируйте результат ввода.

Задание 7. Введите прямым набором "управляющие символы" (ASCII-коды от 0 до 31). Используя электронный справочник HELP, определите комбинации клавиш (Ctrl+клавиша), используемых для ввода управляющих символов.

Задание 8. Используя электронный справочник HELP и технологию прямого набора кодов символов, составьте таблицу ASCII-кодов символов русского алфавита и символов псевдографики.

2.2.5 Контрольные вопросы

1. Каковы основные функции контроллера клавиатуры ?
2. Какие манипуляции с клавишами не приводят к изменению состояния буфера клавиатуры ?
3. В каких случаях значения указателей "головы" и "хвоста" буфера клавиатуры совпадают ?
4. Как может возникнуть ситуация "переполнение буфера клавиатуры"? Какова реакция компьютера на такую ситуацию ?

2.3 Контрольная работа №2

Задача 2.1. Объясните понятия "линейный адрес" и "сегментный адрес". Опишите алгоритм работы сумматора адреса.

Задача 2.2. Определите линейные и сегментные адреса по данным приведенной ниже таблицы. Результаты представьте в шестнадцатеричной и двоичной системах счисления. Определите функциональное назначение соответствующих областей памяти ПК.

Линейный адрес	Сегментный адрес	Область памяти
	0000:010Ch	
00462h		
	0040:0002h	
0041Ah		
	B800:00CDh	

Задача 2.3. Опишите алгоритм работы программы обработки прерывания №9 при нажатии на любую из символьных клавиш.

Задача 2.4. Опишите алгоритм работы программы обработки прерывания №9 при нажатии на любую из управляющих клавиш.

Задача 2.5.(*) Напишите программу для определения SCAN-кодов символьных клавиш и ASCII-кодов соответствующих им символов (используйте средства прямого доступа к порту №60h и буферу клавиатуры).

3 ВИДЕОСИСТЕМА ПК

3.1 Структура и основные характеристики видеосистемы ПК

Понятием *видеосистема* охватывается комплекс программно-технических средств и структур данных, обеспечивающих визуализацию информации, хранящейся в памяти компьютера в кодированной форме.

Аппаратный комплекс видеосистемы включает устройство отображения информации (дисплей) и видеоадаптер - программируемое устройство, выполняющее функции управления дисплеем и интерфейса с центральным процессором. С точки зрения программиста существенным является тип установленного видеоадаптера, режимы его работы, состав и назначение программируемых и информационных регистров.

Первые ПК серии IBM PC комплектовались монохромным дисплеем и видеоадаптером ***MDA*** (**M**onochrome **D**isplay **A**dapter), который работал только в текстовом режиме и имел невысокую разрешающую способность (640 x 350). Позднее фирмой Hercules Computer Technology был разработан видеоадаптер ***Hercules***, который, так же, как и MDA, являлся монохромным (двухцветным), однако имел более высокую разрешающую способность (720 x 350) и возможность отображения графической информации.

История применения цветного экранного изображения в IBM-совместимых ПК начинается с видеоадаптера ***CGA*** (**C**olor **G**raphic **A**rray), который обеспечивал работу в текстовом и графическом режимах с 16-цветным изображением. При этом разрешающая способность ***CGA*** была весьма низкой (экран -320 x 200, символ – 8 x 8 пикселов) даже по сравнению с ***MDA*** и ***Hercules***, что объясняется недостаточным (для цветного изображения) объемом используемой видеопамяти (64 Kb).

Позднее фирмой IBM были выпущены более совершенные видеоадаптеры ***EGA*** (**E**nhanced **G**raphic **A**rray), ***VGA*** (**V**ideo **G**raphic **A**rray) и ***XGA*** (e**X**tended **G**raphic **A**rray) с более высокими техническими характеристиками (видеопамять – до 1 Mb, разрешение – до 1024 x 768 пикселов, отображение до 65536 цветов). Основные характеристики стандартных видеорежимов приведены в таблице 3.1.

К настоящему времени различными фирмами – производителями выпущено большое количество типов видеоадаптеров, превосходящих ***VGA*** по своим техническим характеристикам. Эти видеоадаптеры принято объединять общим названием ***SVGA*** (**S**uper **VGA**), и режимы их работы могут существенно отличаться от стандартных режимов IBM.

Базовое *программное обеспечение* видеосистемы составляет набор функций прерывания BIOS INT10h, обеспечивающих управление видеоадаптерами в стандартных режимах. Перечень основных функций с краткими комментариями по их применению приведен в таблице 3.3.

Основными техническими характеристиками видеосистемы ПК, определяющими качество экранного изображения, являются :

- *Разрешающая способность,* определяемая числом строк и столбцов точек (пикселов), выводимых на экран дисплея. Для текстовых видеорежимов разрешающая способность оценивается также числом строк и столбцов символов на экране и размерами матрицы пикселов, используемой для изображения одного символа (знакоместа).
- *Цветность*, определяемая количеством цветов (или оттенков серого цвета) изображения, выводимого на экран. Монохромные дисплеи позволяют выводить двухцветное (не обязательно черно-белое) изображение. В текстовых режимах работы видеосистемы программируются цвет символа и цвет фона для каждого знакоместа, в графических – цвет каждого пиксела.
- *Динамические характеристики* видеосистемы определяются скоростью смены «картинки» на экране дисплея. Для улучшения динамических характеристик в видеоадаптерах применяются специальные архитектурные решения, такие, например, как двойное сканирование или использование нескольких банков видеопамяти в едином адресном пространстве.

3.2 Структуры данных, обслуживающие видеосистему

- *Видеопамять* - основная структура данных, выполняющая функции буфера обмена между прикладными программами и видеоадаптером. Видеопамять логически расположена в основном адресном пространстве в диапазоне адресов с A000:0000 по B000:FFFF (128 Кбайт). Физически видеопамять расположена на плате видеоадаптера и может иметь объем, многократно превышающий 128 Кбайт. Структура видеопамяти, ее объем и расположение в адресном пространстве определяются типом видеоадаптера и режимом его работы.
- *Переменные BIOS.* Для обслуживания видеосистемы используются также часть *области данных BIOS*, формируемой в процессе инициализации (загрузки DOS). Знание адресов этих переменных позволяет определять количество и типы установленных видеоадаптеров, режимы их работы, объем видеопамяти и ряд других параметров видеосистемы.
- *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 BIOS по адресу 0000:04A8h.
- *Таблица параметров*. Содержит описания для каждого поддерживаемого видеоадаптером режима. Используется BIOS для установки регистров видеоадаптера.

- *Область сохранения.* В этой области BIOS EGA/VGA хранит значения регистров цветовой палитры видеоадаптера. Прикладные программы могут косвенно определять значения этих регистров, обращаясь к данной области.
- *Вспомогательная таблица символов текстового режима.* Определяет параметры знакогенератора для текстовых режимов и содержит список видеорежимов, использующих эту таблицу.
- *Вспомогательная таблица символов графического режима.* Определяет параметры знакогенератора для графических режимов и содержит список видеорежимов, использующих эту таблицу.
- *Таблицы знакогенераторов* – специальные структуры данных, располагающиеся в ОЗУ и ПЗУ и используемые в процессе формирования символов на экране.
- Указатели на таблицы знакогенераторов - *вектора прерываний* INT 1Fh и INT 43h. Различные программы (например, программы-русификаторы) могут загружать в видеопамять собственные знакогенераторы и переустанавливать соответствующие вектора прерываний.

3.3 Кодирование данных в видеопамяти

Процесс вывода информации на экран дисплея реализуется по следующей схеме.

- В соответствии с установленным режимом работы видеосистемы в видеопамяти ПК организуется множество *страниц*, каждая из которых может использоваться для хранения кодированной информации, предназначенной для отображения одного (полного) экрана дисплея.
- Одна из страниц объявляется *активной*, и ее номер записывается в области данных BIOS по адресу 0040:0062.
- Центральный процессор, работая под управлением прикладной программы и используя справочную информацию области данных BIOS, формирует образ экрана, записывая кодированные данные в активную страницу видеопамяти.
- Видеоадаптер циклически читает данные активной видеостраницы, преобразует (декодирует) их и формирует соответствующие сигналы управления дисплеем[11].

Размер страницы определяется типом видеорежима, разрешающей способностью и числом отображаемых цветов, а количество страниц ограничивается объемом установленной видеопамяти. Все страницы пронумерованы, начиная с нулевой.

Каждый элемент экрана описывается (кодируется) блоком данных, расположение которого в видеопамяти определяется координатами соответствующего элемента – номером строки и столбца, которые отсчитыва-

[11] Вопросы программирования видеоадаптеров в данном пособии не рассматриваются. Более подробная информация по этому вопросу содержится в [8].

ются сверху вниз и слева направо и нумеруются с нуля. Адрес блока данных задается смещением относительно начала видеостраницы.

В текстовых и графических режимах используются различные системы кодирования элементов экрана, так как в первом случае в качестве элемента экрана используется *знакоместо*, а во втором – *пиксел*.

3.3.1 Кодирование данных в текстовых режимах

Каждое знакоместо экрана кодируется двумя соседними байтами – *байтом символа* и *байтом атрибутов*. Байт символа (четный) хранит код ASCII символа, выводимого на экран, а байт атрибутов (нечетный) задает цвет символа, цвет фона и некоторые другие параметры, определяемые настройками видеоадаптера.

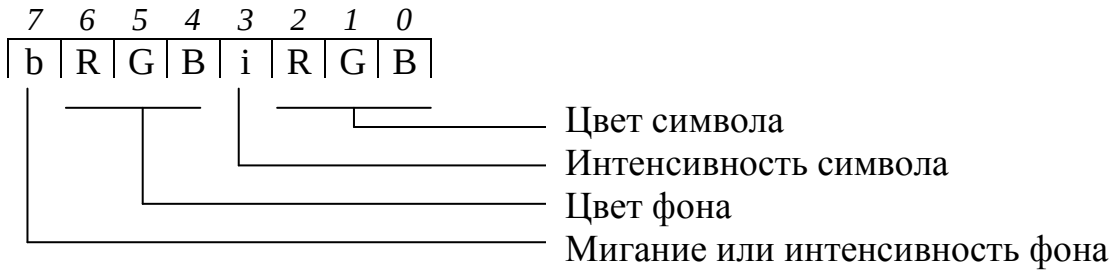

Рисунок 3.1 - Формат байта атрибутов

Процесс вывода символа на экран дисплея реализуется по следующей схеме:

- Видеоадаптер обращается к области данных BIOS (таблица 3.2) и определяет номер активной страницы и координаты курсора (**x,y**).
- По этим данным вычисляется адрес машинного слова, описывающего в видеопамяти знакоместо, на которое указывает курсор. Адрес (смещение относительно начала видеостраницы) рассчитывается по простой формуле: **2*(x + y * m)**, где **x** – номер столбца символов (от 0 до 39 или до 79), **y** – номер строки (от 0 до 24), **m** – число символов в строке (40 или 80), определяемое выбранным видеорежимом. Например, левое верхнее знакоместо экрана (0,0) отображается на видеопамять с адресом (смещением) **0000**, а правое нижнее знакоместо (79,24) – со смещением **0F9E**h.
- По этому адресу видеоадаптер считывают из видеопамяти байт символа и байт атрибутов.
- Байт символа (ASCII-код) используется далее для вычисления адреса расположения соответствующей битовой матрицы в таблице знакогенератора (п.3.4), а байт атрибутов – для формирования сигнала управления цветом изображения символа.

В соответствии с рассмотренной системой кодирования символов для текстовых режимов (2 байта на 1 знакоместо) каждая страница видеопамяти для режимов №0 и №1 потребует 2000 байтов (2 * 40 * 25), а для режимов №2 и №3 - 4000 байтов (2 * 80 * 25). Фактически размер видеостраниц устанавливается кратным целой степени числа 2 – соответ-

ственно 2 Кб и 4 Кб, причем последние 48 (96) байтов в каждой странице не используются.

Видеоадаптеры CGA, EGA и VGA поддерживают 8 страниц для 0-го, 1-го, 2-го и 3-го режимов. При этом нулевая страница располагается с адреса B800:000, а последующие – со смещением в 2 или 4 Кб относительно предыдущей страницы.

3.3.2 Кодирование данных в графических режимах

В графических режимах каждый блок видеопамяти описывает один пиксел, причем размер блока определяется количеством отображаемых цветов (или градаций серого цвета). Например, в монохромных (двухцветных) режимах №6 и 11h для описания пиксела достаточно одного бита, в 4-цветных режимах №4, №5 и 0Fh – двух битов, в 16-цветных режимах №0D, №0E, №10h и №12h – четырех битов, а в 256-цветном режиме №13h для описания пиксела потребуется один байт.

Таким образом, одним байтом видеопамяти может быть описано от 1 до 8 пикселов, причем старшие биты байта отвечают за «левые» пикселы, а младшие – за «правые». Например, цвет левого верхнего пиксела экрана кодируется: в режимах №6 и 11h - старшим битом, в режимах №4 и №5 - двумя старшими битами, а в режиме №13h – всеми восемью битами нулевого байта видеостраницы.

Размер видеостраницы определяется разрешающей способностью (количеством пикселов на экране) и размером блока, описывающего один пиксел. Например, в режимах №4 и №5[12] (4 цвета, разрешение 320 x 200) и в режиме №6 (2 цвета, разрешение 640 x 200) для одной видеостраницы потребуется 16000 байт (фактически – 16 Кб). В этих режимах объема видеопамяти достаточно для размещения двух страниц, однако адаптеры CGA, EGA и VGA поддерживают в этих режимах одну страницу с начальным адресом B800:0000. В режиме №13h (256 цветов, разрешение 320 x 200) одна страница занимает 64 Кбайт, то есть целый банк видеопамяти, располагающийся с адреса A000:0000.

3.4 Знакогенераторы

Знакогенератор (или таблица знакогенератора) – это специальная структура данных, описывающая растровое (точечное) изображение символа и используемая видеоадаптером для преобразования ASCII-кода символа в последовательность сигналов, управляющих состоянием пикселов.

[12] В режимах №4 и №5, первоначально разработанных для CGA, отображение видеопамяти на экран не является непрерывным. Первая половина видеостраницы (B800:0000 – B800:1FFF) содержит данные для всех четных линий экрана, а вторая половина (начиная с адреса B800:2000) – для всех нечетных. Такая же схема использования видеопамяти в этих режимах применяется и адаптерами EGA/VGA (для обеспечения совместимости с CGA).

Графический образ каждого символа описывается битовой матрицей, размерность которой определяется разрешающей способностью видеосистемы и различна для разных видеоадаптеров (CGA – 8x8, EGA – 8x8 и 8x14, VGA – 9x16). Каждый узел матрицы описывается одним битом, при этом единичное значение бита соответствует активному (светлому) пикселу, а нулевое – пассивному (темному).

Следующий рисунок иллюстрирует описание графического образа символа «A» для видеоадаптера CGA.

Образ символа								Коды строки матрицы	
								0000 0000	00h
				■	■	■		0000 1110	0Eh
			■			■		0001 0010	12h
		■				■		0010 0010	22h
		■	■	■	■	■		0011 1110	3Eh
		■				■		0010 0010	22h
		■				■		0010 0010	22h
		■				■		0010 0010	22h

Рисунок 3.2 - Описание символа битовой матрицей 8x8

Как видно из рисунка, для описания одного символа матрицей 8x8 требуется блок памяти размером в 8 байтов – по одному байту на каждую строку матрицы. Так, символ «A» будет описан следующей последовательностью байтов: 00h, 0Eh, 12h, 22h, 3Eh, 22h, 22, 22h.

При использовании матрицы 8x14 (EGA) для описания одного символа достаточно 14 последовательных байтов, а при использовании матрицы 9x16 (VGA) – 16 байтов (по одному байту на строку матрицы)[13].

В таблице знакогенератора блоки описаний символов расположены в порядке возрастания их ASCII-кодов. При этом адрес блока (смещение относительно начала таблицы знакогенератора) может быть вычислен по формуле : **m** * <Код символа>, где **m** – число байтов, отводимых для описания одного символа.

Видеоадаптер CGA хранит таблицу знакогенератора, используемую в текстовых режимах, во внутреннем ПЗУ, расположенном вне адресного пространства центрального процессора. Эта таблица описывает стандартные символы с кодами ASCII от 0 до 255, не содержит символов кириллицы и недоступна прикладным программам ни для чтения, ни для записи. Единственной возможностью изменения образа символов в текстовых режимах CGA является перепрограммирование ПЗУ видеоадаптера. В графических режимах CGA доступна таблица знакогенератора для символов с кодами от 127 до 255, расположенная в области ROM BIOS. Указателем на начало этой таблицы является вектор прерывания 1Fh, что используется

[13] Таблицы знакогенераторов для текстовых режимов EGA и VGA, загружаемые в видеопамять из области ROM BIOS, отводят для описания каждого символа по 32 байта, из которых используются только первые 14 (EGA) и 16 (VGA) байтов.

программами – русификаторами, которые формируют битовые матрицы для каждого символа, записывают их в определенную область ОЗУ и соответственно переустанавливают вектор прерывания 1Fh.

Таблицы знакогенераторов EGA / VGA для символов с кодами от 0 до 255 хранятся в ROM BIOS и доступны по вектору прерывания 43h. 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 различных символов[14].

Таблицы знакогенератора автоматически загружаются в видеопамять (во второй «цветовой слой») при выборе любого из текстовых режимов, что позволяет программно заменить стандартную таблицу знакогенератора своей собственной. BIOS EGA/VGA содержит специальную функцию, управляющую загрузкой шрифтов (функция 11h прерывания 10h).

3.5 Лабораторная работа №6

ИССЛЕДОВАНИЕ ВИДЕОПАМЯТИ В ТЕКСТОВЫХ РЕЖИМАХ

3.5.1 Учебные задания

Задание 1. Используя информацию области данных BIOS, определите:

- тип установленного в Вашем компьютере видеоадаптера;
- объем установленной видеопамяти;
- номер текущего видеорежима;
- число символов в строке;
- размер видеостраницы (в байтах);
- начальный адрес активной страницы видеопамяти;
- координаты курсора для каждой из видеостраниц текущего текстового видеорежима.

Задание 2. Определите адрес машинного слова (байт символа и байт атрибутов), описывающего знакоместо экрана с заданными координатами.

Задание 3. Используя программу PEEK, выведите на экран образ активной видеостраницы (в шестнадцатеричном и текстовом форматах). Прокомментируйте результаты отображения и объясните систему кодирования видеоданных в текстовом режиме.

[14] Номера активных таблиц определяются содержимым регистра выбора знакогенератора видеоадаптера, а выбор одной из двух активных таблиц для отображения символа кодируется значением 3-го бита байта атрибутов этого символа.

3.5.2 Контрольные вопросы

1. Чем ограничивается разрешающая способность видеосистемы в текстовых и графических видеорежимах ?
2. Объясните понятия: "страница видеопамяти", "активная страница", "текущая страница". Чем ограничивается количество видеостраниц ?
3. Опишите алгоритм вычисления адреса ячейки видеопамяти, описывающей знакоместо с заданными координатами для стандартного видеорежима № 3.

3.6 Лабораторная работа №7

ИССЛЕДОВАНИЕ СТРУКТУРЫ ТАБЛИЦ ЗНАКОГЕНЕРАТОРОВ

3.6.1 Учебные задания

Задание 1. Используя вектора прерываний №1Fh и №43h, определите расположение соответствующих таблиц знакогенератора.

Задание 2. Исследуйте структуру одной из таблиц знакогенератора. По данным таблицы составьте битовую матрицу, описывающую символ с заданным ASCII-кодом.

3.6.2 Контрольные вопросы

1. Каковы размерности матриц знакогенераторов для MDA, CGA, EGA, VGA ?

2. Опишите алгоритм вычисления адреса ячейки видеопамяти, описывающей пиксел с заданными координатами для стандартного видеорежима № 4.

3.7 Контрольная работа №3

Задача 3.1. **(*)** Напишите программу, изменяющую размеры курсора.

Задача 3.2. **(*)** Напишите программу вывода в заданную "точку" экрана символа, соответствующего нажатой клавише (задаются координаты точки, цвет символа и цвет фона).

Задача 3.3. **(*)** Напишите программу вывода на экран изображения символа в масштабе "один пиксел → одно знакоместо" в соответствии с одной из установленных таблиц знакогенератора.

3.8 Справочные материалы

Таблица 3.1 - Характеристики стандартных режимов работы видеоадаптеров

№ режима	Тип	Разреше-ние	Число цветов	Видеоадаптеры	Нач. адрес
0, 0*, 0+	Текстовый	40 x 25	16 (п/т)	CGA,EGA,VGA	B800
1, 1*, 1+	Текстовый	40 x 25	16	CGA,EGA,VGA	B800
2, 2*, 2+	Текстовый	80 x 25	16 (п/т)	CGA,EGA,VGA	B800
3, 3*, 3+	Текстовый	80 x 25	16	CGA,EGA,VGA	B800
4	Графический	320 x 200	4	CGA,EGA,VGA	B800
5	Графический	320 x 200	4(п/т)	CGA,EGA,VGA	B800
6	Графический	640 x 200	2	CGA,EGA,VGA	B800
7	Текстовый	80 x 25	2	MDA,EGA,VGA	B000
8 – 0Ch	Резерв				
0Dh	Графический	320x200	16	EGA,VGA	A000
0Eh	Графический	640x200	16	EGA,VGA	A000
0Fh	Графический	640x350	4	EGA,VGA	A000
10h	Графический	640x350	16	EGA,VGA	A000
11h	Графический	640x480	2	VGA	A000
12h	Графический	640x480	16	VGA	A000
13h	Графический	320x200	256	VGA	A000

Примечания

1. Режимы 0, 2 и 5 являются режимами с подавлением цвета. В этих режимах вместо цветного выводится полутоновое изображение с заменой множества цветов на оттенки серого цвета.
2. Разрешающая способность видеосистемы в текстовых режимах оценивается размерами матрицы пикселов, которой описывается один символ: для CGA – 8 x 8, для EGA – 8 x 14, для VGA – 9 x 16.
3. Режимы 0*, 1*, 2* и 3* (EGA) - это аналоги текстовых режимов 0,1,2 и 3 CGA, отличающиеся размерами матрицы описания символов (8x14).
4. Режимы 0+, 1+, 2+ и 3+ (VGA) - это аналоги текстовых режимов 0,1,2 и 3 CGA, отличающиеся размерами матрицы описания символов (9x16).

Таблица 3.2 - Расположение структур данных видеосистемы ПК

Начальный адрес	Длина, байт	Назначение области памяти
Таблица векторов прерываний		
0000:0040	4	INT 10h – указатель на видеофункции BIOS
0000:007C	4	INT 1Fh – указатель на таблицу знакогенератора для символов с кодами 127 –255
0000:010C	4	INT 43h – указатель на таблицы знакогенератора для символов с кодами 0 – 255 (EGA/VGA)
Область данных BIOS		
0000:0410	1	Флаги конфигурации. 5-й и 4-й биты определяют тип видеоадаптера: 00 – EGA; 01 – CGA 40 x 25; 10 – CGA 80 x 25; 11 – MDA
0000:0449	1	Номер текущего видеорежима
0000:044A	2	Число символов в строке
0000:044C	2	Размер видеостраницы (в байтах)
0000:044E	2	Начальный адрес активной видеостраницы (смещение в видеосегменте)
0000:0450	8 x 2	Координаты курсора в каждой из 8 страниц: Младший байт - № колонки, старший - № строки
0000:0460	2	Размер (форма) курсора: младший байт – № последней линии, старший байт – № первой линии
0000:0462	1	Номер активной страницы видеопамяти
0000:0463	2	Адрес порта контроллера ЭЛТ (3B4 или 3D4)
0000:0465	1	Данные регистра режима CGA
0000:0466	1	Данные регистра цветовой палитры CGA
0000:0484	1	Число текстовых строк экрана минус единица
0000:0485	2	Высота символа в пикселах
0000:0487	1	1-й байт данных о EGA. 6-й и 5-й биты определяют объем установленной видеопамяти: 00 - 64К, 01 - 128К, 10 – 192К, 11 – 256К
0000:0488	1	2-й байт данных о EGA
0000:04A8	2	Адрес таблицы окружения минус единица
0000:0500	1	Состояние печати экрана : 00h – печать окончена; 01h – экран печатается, FFh – ошибка печати
Видеопамять		
A000:0000	64Кб	Видеопамять в графических режимах EGA,VGA
B000:0000	32Кб	Видеопамять в монохромном текстовом режиме
B800:0000	32Кб	Видеопамять в цветовых текстовых режимах и в графическом режиме CGA
П З У		
C000:0000	16Кб	ROM BIOS EGA / VGA

Приведенные в таблице 3.3 функции BIOS EGA/VGA доступны по прерыванию №10h. Полный список видеофункций BIOS с комментариями и примерами использования приведен в электронном справочнике "Help".

Например, следующая программа вызывает функцию №00h для выбора 4-го режима работы видеоадаптера: **mov ah, 0; mov al, 4; int 10h** .

Таблица 3.3 - Функции BIOS, обслуживающие видеосистему

Функция	Назначение	Вход	Выход	Комментарии
1	2	3	4	5
00h	Выбор режима работы видеоадаптера	AH = 00h AL = № режима	-	
01h	Изменение размеров курсора	AH = 01h CH = верхняя граница курсора CL = нижняя граница курсора	-	CH: биты D0-D3 задают положение верхней границы курсора (0 – 15); биты D4-D5 задают тип курсора – 00 – обычный, 01 – невидимый, 10 – мигающий, 11 – быстро мигающий; биты D6, D7 – не используются
02h	Изменение положения курсора	AH = 02h BH = № видеостраницы DH = № строки DL = № столбца	-	Для каждой видеостраницы может быть задано свое положение курсора. При активизации страницы курсор устанавливается в заданную позицию, предназначенную для вывода очередного символа.
03h	Определение положения и формы курсора	AH = 03h BH = № видеостраницы	CH = верхняя граница курсора CL = нижняя гр. DH = № строки DL = № столбца	
05h	Изменение активной страницы	AH = 05h AL = № страницы	-	

Продолжение таблицы 3.3

1	2	3	4	5
08h	Чтение символа	AH = 08h BH = № страницы	AL = ASCII-код символа AH = байт атрибутов символа	Символ читается из позиции, определенной положением курсора на заданной видеостранице
09h	Запись символов	AH = 09h AL = ASCII-код символа BH = № страницы BL =байт атрибутов CX = число записываемых символов	-	Записываются один или несколько одинаковых символов с одинаковыми атрибутами в заданную (активную или пассивную) страницу видеопамяти. Запись производится начиная с позиции, заданной текущим положением курсора на этой странице. После завершения операции положение курсора не изменяется.
0Ch	Запись пиксела	AH = 0Ch AL = № цвета BH = № страницы CX = X DX = Y	-	
0Dh	Чтение пиксела	AH = 0Dh BH = № страницы CX = X DX = Y	AL = номер цвета пиксела	
0Fh	Определение текущего режима работы видеоадаптера	AH = 0Fh	AH = число символов в строке AL = № текущего режима BH = № активной страницы	

Продолжение таблицы 3.3

1	2	3	4	5
11h	Загрузка таблиц знакогенератора. Данная функция содержит 11 подфункций, основные из которых описаны ниже			
П/ф 00h	Загрузка пользовательского набора символов	AH = 11h AL = 00h, 10h ES:BP = начальный адрес набора символов CX = число символов в наборе DX = № 1-го заменяемого символа в таблице ЗГ BL = № таблицы ЗГ BH = число байтов на символ в таблице (1 - 32)	-	Функция заменяет текущий набор символов или его часть набором, определяемым пользователем. Новый набор символов должен быть загружен в память до вызова функции
П/ф 01h	Загрузка набора символов EGA (8x14)	AH = 11h AL = 01h, 11h BL = № загружаемой таблицы ЗГ	-	Функция загружает стандартный набор символов из ПЗУ BIOS во второй цветовой слой видеопамяти
П/ф 02h	То же для стандартного набора символов CGA 8 x 8 (AL = 02h,12h)			
П/ф 04h	То же для стандартного набора символов VGA 9 x 16 (AL = 04h,14h)			

Окончание таблицы 3.3

1	2	3	4	5
П/ф 20h	Установка вектора прерывания 1Fh	AH = 11h AL = 20h ES:BP = адрес таблицы символов (с кодами от 127 до 255)	-	Используется в графических режимах №4,5 и 6, совместимых с CGA, если необходимо отображать одновременно более 128 различных символов. Размер символов в таблице должен быть 8x8
П/ф21 h	Установка вектора прерывания 43h	AH = 11h AL = 21h ES:BP = адрес таблицы символов пользователя CX = байтов на символ BL = число строк на экр.	-	BL = 0 – число строк содержится в регистре DL 1 – 14 строк 2 – 25 строк 3 – 43 строки
П/ф 30h	Получение ин-ин-формации об исполь пользуемом наборе символов	AH = 11h AL = 30h BH – код вида запрашиваемой информации (0-7)	CL = высота символа DL = число текстовых строк на экране минус 1 ES:BP = указатель (см. BH=)	Виды запрашиваемой информации (BH): **0** – вектор INT 1Fh; **1** - вектор INT 43h; **2** – указатель на основной набор символов 8x8; **3** – то же для 8x14; **4** - то же для 9x16; **5**, **6** и **7** – то же, что 2, 3 и 4, только для альтернативных наборов символов

СПИСОК ЛИТЕРАТУРЫ

1 Брябрин В.М. Программное обеспечение персональных ЭВМ.- М.:Наука.1988.–272 с.

2 Журден Р. Справочник программиста персональных компьютеров типа IBM PC, XT и AT / Пер. с англ. – М.: Финансы и статистика, 1992.

3 Нортон П. Персональные компьютеры фирмы IBM и операционная система MS DOS. – М.: Радио и связь, 1991. – 416 с.

4 Фигурнов В.Э. IBM PC для пользователя. – М.: Финансы и статистика, 1998. – 326 с.

5 Фролов А.В., Фролов Г.В. Аппаратное обеспечение IBM PC. В 2 ч. – М.: Диалог - МИФИ, 1992. – 208 с. (Библиотека системного программиста. Т2. Ч.1,2).

6 Фролов А.В., Фролов Г.В. Программирование видеоадаптеров CGA, EGA и VGA – М.: Диалог - МИФИ, 1992. – 288 с..(Библиотека системного программиста. Т3).

Учебное издание

Волк Владимир Константинович

ИССЛЕДОВАНИЕ ФУНКЦИОНАЛЬНОЙ СТРУКТУРЫ ПАМЯТИ ПЕРСОНАЛЬНОГО КОМПЬЮТЕРА

ЛАБОРАТОРНЫЙ ПРАКТИКУМ

УЧЕБНОЕ ПОСОБИЕ

Редактор Н.М.Кокина

Подписано в печать	22.09.2004	Формат 60x84 1/16	Бумага тип. №1
Плоская печать		Усл. печ. л. 4,4	Уч. изд. л. 4,4
Заказ № 252		Тираж 300 экз.	Цена свободная

Издательство Курганского государственного университета
640669, г. Курган, ул. Гоголя, 25.
Курганский государственный университет, ризограф.