Некоммерческое акционерное общество

АЛМАТИНСКИЙ УНИВЕРСИТЕТ ЭНЕРГЕТИКИ И СВЯЗИ

Кафедра информационных систем

СИСТЕМНОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ

Методические указания к лабораторным работам
для студентов специальности
5В060200 – Информатика

Алматы 2014

СОСТАВИТЕЛЬ: Н.А. Водолазкина Системное программное обеспечение. Методические указания к выполнению лабораторных работ для студентов специальности 5В060200 – «Информатика – Алматы: АУЭС, 2014. – 37 стр.

Методические уаказания содержат необходимые материалы для выполнения лабораторных работ , контрольные вопросы и перечень рекомендуемой литературы по дисциплине «Системное программное обеспечение».

Методические указания предназначены для бакалавров специальности 5В060200 – «Информатика.

Ил.- 33, библиогр.- 9 назв.

Рецензент: к.т.н., доцент Куликов А.А.

Печатается по плану издания некоммерческого акционерного общества «Алматинский университет энергетики и связи» на 2013 г.

Содержание

Введение

Целью дисциплины «Системное программное обеспечение» является изучение основных возможностей доступа к системным ресурсам ПЭВМ дискам, оперативной памяти, дисплею, клавиатуре, возможности изменять некоторые стандартные характеристики ресурсов, используя различные уровни доступа к ресурсам.

Следующий этап изучения курса - написание программ с использованием элементов системного программирования. На этом этапе рассматриваются способы организации информации и управление ею при вводе в буфер ЭВМ и выводе на экран и печать, управление возможностями программных оболочек, работа с файлами, директориями и так далее.

Методические указания к проведению лабораторных работ по курсу «Системное программное обеспечение» содержат процедуры, которые можно использовать при написании курсовых проектов по тематике «Системное программирование», «Системное программное обеспечение».

Материал курса построен таким образом, что лекционный материал закрепляется во время выполнения лабораторных работ.

1 Варианты заданий к лабораторным работам

1.1 Лабораторная работа №1. Использование функций прерывания 21h. Определение системных ресурсов ПЭВМ

Цель работы: использование функций прерывания 21h, называемых системными вызовами, определение системных ресурсов ПЭВМ, используя BIOS и CMOS-память.

Общие сведения.

Функции прерывания 21h (DOS) .

Прерывание 21h (DOS) предназначено для предоставления программисту различных услуг со стороны DOS. Этими услугами является набор функций.

00h - завершение работы программы

01h - ввод символа с ожиданием и эхосопровождением.

На выходе в AL - ACSII-код символа.

02h - вывод символа.

На входе в DL - ACSII-код символа

05h - вывод символа на принтер.

На входе в DL - ACSII-код символа

07h ввод символа с ожиданием и эхосопровождением.

08h ввод с консоли без эхопечати.

На выходе в AL - ACSII-код символа. 08h при вводе проверяет, не нажато ли CTRL+BREAK

09h - вывод строки на экран.

На входе - ds:dx =адресу строки с символом $ на конце

0ah - ввод строки с клавиатуры.

На входе ds:dx =адрес буфера с форматом:

1 байт - размер буфера для ввода (формирует пользователь)

2 байт - число фактически введенных символов (заполняет с-ма по окончанию ввода-нажатию Enter (0dh), этот символ не считает)

3 байт и далее - введенная строка с символом 0dh на конце

На выходе - введенная строка в буфере

0bh - проверка состояния буфера клавиатуры

На выходе AL=0 - буфер пуст,

AL=offh - в буфере есть символы

Последовательность действий при использовании этих функций:

1) Поместить номер функции в регистр AH.

2) Поместить передаваемые функции параметры в определенные регистры.

3) Вызвать прерывание командой int 21h.

4) Извлечь результаты работы функции из определенных регистров.

Общие понятия о BIOS и SMOS.

BIOS — базовая программа, самостоятельно запускающаяся при включении компьютера и отвечающая за ввод и вывод данных.

Ее основной задачей является подготовка машины к загрузке операционной системы и установления ее контроля над компьютером. Программа выполняет самотестирование устройств и осуществляет поиск загрузчика операционной системы на доступных носителях — жестком диске, флеш-карте, CD, дискете (в настройках можно указывать тип носителя, с которого необходимо произвести загрузку). Если загрузчик не обнаружен, то программа выдает сообщение об ошибке.

«Базовая система ввода-вывода» — так расшифровывается аббревиатура BIOS, звучащая в оригинале как Basic Input-Output System. При возникновении проблем BIOS может помочь произвести ремонт компьютера, не прибегая к посторонней помощи — выявить неисправности и настроить процесс загрузки операционной системы. Чаще всего в стационарных машинах роль ключа, открывающего доступ, играет нажатие клавиши Delete, во многих ноутбуках — F2; также могут использоваться клавиши Esc, F1, Tab. О том, какие клавиши следует нажимать, можно узнать из:

- строк, отображающихся при загрузке системы (если они исчезают слишком быстро, вам может понадобиться помощь: нажмите клавишу паузы — Pause, она еще может называться Break или Tab);
- руководства пользователя (информация о материнской плате).

Было упомянуто, что при помощи программы BIOS Setup можно производить настройки BIOS. С одной стороны нельзя ничего менять, с другой - можно производить настройки? Все дело в том, что сама BIOS действительно содержит набор программ, который остается неизменным, а вот "настройки БИОС" - это не что иное, как данные для этих самых программ, которые хранятся не в микросхеме BIOS и их-то мы и можем менять во время работы с BIOS Setup.

Вот эти самые значения, изменять которые можно в BIOS Setup, хранятся в специальной микросхеме динамической памяти, которая называется *CMOS* (название технологии, по которой производится микросхема: *Complementary Metal-Oxide-Semiconductor* - комплиментарный металлооксидный полупроводник или КМОП). Кроме настроек BIOS в CMOS хранятся параметры конфигурации компьютера. Суммарный объем памяти CMOS составляет всего 256 байт и потребляет она очень мало энергии. Стандартная батарейка, расположенная на материнской плате, питает CMOS в течение 5-6 лет, после чего необходимо производить ее замену.

При включении компьютера происходит тестирование оборудования, в процессе которого сравнивается его текущая конфигурация с данными в CMOS-памяти. Если обнаруживаются отличия, то происходит автоматическое обновление CMOS-памяти, либо вызывается BIOS Setup.

Если срок батарейки, питающей CMOS, подошел к концу, то при включении компьютера на экран будет выведено сообщение, например,

«*CMOS-checksum error»*. Для возобновления работы компьютера необходимо будет установить новую батарейку взамен вышедшей из строя. После замены батарейки при первом включении компьютера заводские настройки, хранящиеся в BIOS, будут «сброшены» в CMOS-память. Это, кстати, один из способов устранить неисправность. Для этого надо выключить компьютер, вынуть на 30 секунд батарейку из материнской платы, установить ее назад, и заводские настройки BIOS будут восстановлены, а компьютер снова заработает.

Батарейку обнаружить на материнской плате не составит большого труда, т.к. она довольно крупная. Как правило, рядом с ней располагается и микросхема CMOS. А вот BIOS-микросхема может находиться совершенно в другом месте платы - надо будет посмотреть паспорт на материнскую плату, где эта информация указана.

Задание А.

Выбрав вариант согласно последней цифре учебного шифра, составить программу на языке Си для решения приведённой в нём задачи. Для удобства после условий вариантов приводятся функции прерывания 21h, используемые в соответствующих вариантах.

Методические указания.

Ниже приведены функции прерывания 21h.

Десятичный номер функции устанавливается в регистр AH. Некоторые функции требуют установки дополнительных регистров на входе. Затем указываются значения регистров, содержащих требуемые результаты после вызова *прерывания 21h.*

1. AH=42. На выходе : CX - год после 1980, DH - номер месяца (1 - январь и т.д.), DL - день месяца.
2. AH=44. На выходе : CH - часы (от 0 до 23), CL - минуты (от 0 до 59),DH - секунды (от 0 до 59), DL - сотые доли секунд (от 0 до 99).
3. AH=51, AL=05. На выходе : DL - номер дисковода загрузки операционной системы (1 - A:, и т.д.).
4. AH=51, AL=06. На выходе : BL - номер основной версии DOS, BH номер подверсии.
5. AH=1. На выходе : AL - символ, принятый со стандартного входа (с клавиатуры).
6. AH=6. Если на входе в регистр DL записан код ff, то символ вводится со стандартного входа и помещается на выходе в регистр AL. Если на входе в регистр DL записан другой код, то соответствующий ему символ выводится на экран.
7. AH=9. На входе : DS:DX - сегментный адрес и смещение для выводимой строки, которая должна заканчиваться символом $.
8. AH=67. На входе : DS:DX установлен на ASCIIZ строку имени файла. Если на входе AL=0, то атрибуты файла возвращаются на выходе в

CX, если на входе AL=1, то на выходе файлу назначаются атрибуты, заданные в CX. Причем, в случае удачного завершения операции флаг CF будет сброшен, а в случае ошибки - установлен и в регистр AX возвращается код ошибки. Биты байта атрибутов при наличии в них единицы означают следующее : 0-й бит - файл только для чтения, 1 - скрытый файл, 2 - системный, 3 - метка тома, 4 - каталог, 5 - архивный файл, 6,7 биты не используются.

9. AH=25. На выходе : AL - номер текущего диска.

10. AH=14. На входе : DL - номер диска, который должен стать текущим. На выходе : AL - число логических дисковых устройств.

11. AH=71. На входе : DL - номер диска (0 - текущий диск, 1 - A: и т.д.), DS:SI - указатель на область памяти длиной 64 байт. В эту область на выходе помещается строка, содержащая составное имя текущего каталога, которая начинается с первой буквы подкаталога в корневом каталоге и заканчивается нулевым байтом.

12. AH=57. На входе : DS:DX - полная спецификация подкаталога в виде ASCIIZ-строки. На выходе : создание подкаталога с указанным именем. Причем, флаг CF сброшен в случае удачного завершения операции, в противном случае флаг CF установлен и в AX помещён код ошибки.

13. AH=59. Отличается от описания функции 57-й лишь тем, что на выходе указанный каталог устанавливается текущим.

14. AH=60. На входе : DS:DX - имя файла в виде ASCIIZ-строки, CX - байт атрибутов (назначение его битов объяснено при описании 67-й функции). На выходе : создание файла с указанным именем. Причем, флаг CF сброшен, если операция завершена успешно, и установлен в противном случае с кодом ошибки в AX.

15. AH=61. На входе : DS:DX - имя файла в виде ASCIIZ-строки, AL - код доступа (0 - для чтения, 1 - для записи, 2 - для чтения и записи). На выходе : открыт существующий файл с заданным именем, номер (дескриптор) которого помещён в AX. Флаг CF описывается так же, как и для 60-й функции.

16. AH=62. На входе : BX - номер (дескриптор) файла. На выходе : закрытие файла с указанным номером. Флаг CF описывается так же, как и для 60-й функции.

17. AH=63. На входе : BX - номер (дескриптор) файла, CX - число байтов для чтения, DS:DX - адрес буфера для считанных данных. На выходе : AX - число фактически прочитанных байтов. Чтение осуществляется, начиная с текущей позиции.

18. AH=64. Подобна функции 63, с той лишь разницей, что в CX - число байтов для записи в файл из буфера.

19. AH=65. На входе : имя файла в виде ASCIIZ-строки. На выходе : в случае успешного удаления файла флаг CF сброшен, в противном случае флаг CF установлен, а код ошибки помещён в AX.

20. AH=58. На входе : Полная спецификация удаляемого каталога в виде ASCIIZ-строки, причем, каталог должен быть пустым. На выходе : Флаг CF описывается так же, как и для функции 65.

Пример.

*Задание***:** определить, является ли данный файл системным, и если нет, то сделать его архивным.

Решение.

Для решения задачи следует определить байт атрибутов заданного файла. В нашем случае это файл PROBA текущего каталога.

Наличие единицы в соответствующем бите байта атрибутов означает следующее : 0 бит - файл только для чтения, 1 - скрытый, 2 - системный, 3 - метка тома, 4 - каталог, 5 - архивированный, 6,7 - не используются. Функция 67 (43h) прерывания 21h возвращает атрибуты файла в регистр CX, если на входе, т.е. перед вызовом прерывания 21h в регистре AL нуль, если же AL=1, то файлу назначаются атрибуты, заданные в CX. Кроме того, на входе в DS:DX устанавливается указатель на строку с именем файла. В Турбо-Си значения в регистрах устанавливаются с использованием соответствующих псевдопеременных. Так, в тексте программы оператор _AX=0x4300 означает, что в регистр AH заносится номер функции прерывания 43h, в AL - нуль. левая часть оператора - имя соответствующей псевдопеременной. Далее, операторы _DS=FP_SEG(s) и _DX=FP_OFF(s) устанавливают соответственно значение сегмента и смещения указателя на строку с именем файла. Указанное макро описано в файле dos.h.

После необходимой установки регистров оператор geninterrupt(0x21) вызывает подфункцию 00h функции 43h прерывания 21h. Прототип функции geninterrupt() находится в dos.h.

Листинг программы в кодах приведен в Приложении А.

Варианты задания А.

1. Определить текущий день недели, год, месяц, число, а также текущее время - часы, минуты, секунды, сотые доли секунды.

2. Определить дисковод загрузки, а также номер версии DOS и ее подверсии.

3. Ввести с клавиатуры символ с выводом эха на консоль, затем вывести на консоль символ 0 и символьную строку.

4. Установить предназначен ли заданный файл только для чтения, а, если нет, то сделать его таковым.

5. Определить номер текущего диска и его имя, затем сделать текущим диск D:, если он таковым не является.

6. Установить имя текущего каталога и диска, после чего указать полное имя текущего каталога.

7. Создать в текущем каталоге подкаталог KATAL, и сделать его текущим. При этом определить имя текущего каталога как до вхождения в подкаталог, так и после. При этом, используя функцию system(), получить оглавление соответствующих директорий

8. Создать файл fil.f в текущем каталоге, открыть его, записать в него слово KATAL, после чего закрыть его. Затем, используя функцию system(), получить как оглавление текущего каталога, так и содержимое файла fil.f

9. Пусть имеется подкаталог KATAL с единственным файлом fil.f в нём. Нужно, сделав этот подкаталог текущим, удалить в нём файл. выйти из этого подкаталога в родительский каталог и затем удалить подкаталог KATAL. При этом, используя функцию system(), получить оглавление соответствующих директорий.

10. Пусть в текущем каталоге имеется файл fil.f, содержащий единственное слово KATAL. Нужно, открыв этот файл, прочесть его содержимое, затем закрыть его.

11. Создать в текущем каталоге подкаталог KATAL 1, и сделать его текущим. При этом определить имя текущего каталога .

12. Установить предназначен ли заданный файл только для записи, а, если нет, то сделать его таковым.

13. Определить номер текущего диска и его имя, затем сделать текущим диск A:, если он таковым не является.

14. Ввести с клавиатуры символ без эха на консоль, затем вывести на консоль символьную строку.

15. Войти в подкаталог, в родительский каталог и затем удалить подкаталог. При этом, используя функцию system(), получить оглавление соответствующих директорий.

16. Войти в в родительский каталог, подкаталог. При этом, используя функцию system(), получить оглавление соответствующих директорий.

17. Ввести с клавиатуры символ без эха и с эхом на консоль, затем вывести на консоль 2 символьных строки.

18. Определить номер текущего диска и его имя, затем сделать текущим диск F:, если он таковым не является.

19. Пусть имеется подкаталог DIRECTORI с двумя файлами в нём. Нужно, сделав этот подкаталог текущим, удалить один из файлов. выйти из этого подкаталога в родительский каталог. При этом, используя функцию system(), получить оглавление соответствующих директорий.

20. Открыть файл a2.txt, прочесть его содержимое, затем закрыть его.

Задание Б.

Выбрав вариант согласно последней цифре учебного шифра, составить программу на языке Си для решения приведённой в нём задачи.

Методические указания.

1. Байт оперативной памяти по адресу FE00:1FFF интерпретируется следующим образом : FF - IBM PC; FE - IBM PC/XT; FC - IBM PC/AT.

2. По адресу F000:FFF5 записывается номер версии BIOS.

3. Объём оперативной памяти читается по адресу 0:0413.

4. Установленное в ПЭВМ оборудование можно определить, прочитав значение переменной BIOS по адресу 0:0410. Значения её битов следующие: если бит 0 в единице, то присутствует НГМД; если бит 1 в единице, то имеется сопроцессор (XT,AT); биты 2,3 задают базовую память (в AT не используются); биты 4,5 задают активный видеоадаптер (11 - монохромный, 10 - цветной с разрешением 80 строк по 25 знаков, 01 - цветной с разрешением 40 строк по 25 знаков); биты 6,7 задают число дисководов (00 - 1, 01 - 2, 10 - 3, 11 - 4); бит 8 не используется для PC,XT,AT; биты 9...11 задают число коммуникационных адаптеров от 0(000) до 7(111); если бит 12 в единице, то имеется игровой адаптер; бит 13 не используется для PC,XT,AT; биты 14,15 задают число установленных принтеров от 0(00) до 3(11).

5. Ниже указываются десятичные номера некоторых регистров CMOS-памяти и данные, которые в них записаны : 0 - секунды; 2 - минуты; 4 - часы; 6 - день недели; 7 - день месяца; 8 - месяц; 9 - год; 18 тип фиксированного диска (биты 7...4 - первый диск, 3...0 - второй); 20 - установленное оборудование (биты 7...6 : 00 один НГМД, 01 - два НГМД; биты 5...4 : 01 - цветной дисплей с разрешением 40x25, 10 - цветной дисплей с разрешением 80x25, 11 - монохромный дисплей; биты 3,2 не используются; бит 1 : 1 - имеется сопроцессор, 0 - отсутствует; бит 0 : 1 - имеется НГМД, 0 - отсутствует); 21 - объём памяти в килобайтах на системной плате (младший байт); 22 - объём памяти в килобайтах на системной плате (старший байт); 23 - объём общей памяти в килобайтах (младший байт); 24 - объём общей памяти в килобайтах (старший байт); 48 - память сверх 1 Мб (младший байт); 49 - память сверх 1 Мб (старший байт); 50 - текущее столетие.

Пример.

Задание: определить наличие сопроцессора, используя переменную BIOS по адресу 0:0410. Затем, используя CMOS-память, определить тип накопителя на гибких магнитных дисках.

Решение.

Для решения задачи следует определить бит 1 по адресу 0:0410. Наличие сопроцессора соответствует тому, что в этом бите установлена единица. Функция peek, прототип которой int peek(int segment, int offset), возвращает в переменную h слово, расположенное по адресу segment:offset. Далее, достаточно заметить, что выражение h&2 обнуляет все биты слова h, кроме первого.

Как известно, персональная ЭВМ типа AT хранит информацию о конфигурации в так называемой CMOS-памяти, включающей 64 регистра с

номерами от 0 до 63. Для того, чтобы прочитать значение из регистра с номером N, необходимо сначала послать этот номер в порт 70h посредством функции outportb, прототип которой void outportb(int portid, unsigned char value), а затем прочитать значение из порта 71h, используя функцию inportb, прототип которой unsigned char inportb(int port).

В нашей задаче для того, чтобы определить тип НГМД, следует воспользоваться информацией регистра 10h, в битах 7-4 для первого накопителя и в битах 3-0 для второго. Тогда значение 0 соответствует отсутствию накопителя, значение 1 - накопителю на 360 Кб, значение 2 - на 1.2 Мб, значение 3 - на 720 Кб, значение 4 - на 1.44 Мб.

В приведённом листинге байт из порта 71h заносится в переменную c. Выражения c>>4 и c&0x0f позволяют проанализировать значения этого байта в битах 7-4 и 3-0 соответственно, что и даёт возможность определить тип накопителя.

Листинг программы в кодах приведен в Приложении Б.

Варианты задания Б.

1. Определить тип используемой ПЭВМ, прочитав байт оперативной памяти по адресу FE00:1FFE.
2. Определить и указать дату изготовления версии BIOS, записанную по адресу F000:FFF5.
3. Определить объём оперативной памяти, прочитав его по адресу 0:0413.
4. Определить монохромен ли активный видеоадаптер, используя переменную BIOS по адресу 0:0410.
5. Используя переменную BIOS по адресу 0:0410, определить наличие НГМД.
6. Используя переменную BIOS по адресу 0:0410, определить активный видеоадаптер.
7. Используя переменную BIOS по адресу 0:0410, определить число дисководов.
8. Используя переменную BIOS по адресу 0:0410, определить число установленных принтеров.
9. Используя переменную BIOS по адресу 0:0410, определить наличие игрового адаптера.
10. Используя переменную BIOS по адресу 0:0410, определить число коммуникационных адаптеров.
11. Используя CMOS-память, указать текущее время в час и мин.
12. Определить объём общей памяти в килобайтах, используя CMOS-память.
13. Используя CMOS-память, определить объём памяти в килобайтах на системной плате.

14. Используя CMOS-память, определить тип накопителя на гибких магнитных дисках.

15. Используя CMOS-память, указать наличие сопроцессора.

16. Используя CMOS-память, определить тип дисплея.

17. Используя CMOS-память, определить текущее столетие

18. Указать текущую дату (число, месяц, год), используя CMOS-память.

19. Определить наличие НГМД и их количество, используя CMOS-память.

20. Используя CMOS-память, определить объём памяти в килобайтах сверх 1 Мб.

Основная литература: [1] – 64-82 с., 83-96 с.

Дополнительная литература: [3] – 23-38 с.

Контрольные вопросы.

1. Обзор системных функций DOS ввода данных с клавиатуры.
2. Дать сравнительную характеристику функций DOS ввода с .
3. В какой регистр устанавливается десятичный номер функции 21 прерывания?
4. Использование группы функций Int 21h (01h, 06h, 07h, 08h, 0Ah, 0Ch) дать назначение каждой.
5. Работа с функцией 02h. Вывод одиночного символа.
6. Работа с функцией 09h. Вывод строки.
7. Работа с функцией 40h. Вывод данных в файл или в устройство.
8. Общие понятия о BIOS и SMOS.
9. Как используя CMOS-память, указать текущее время в час и мин?
10. Используя CMOS-память, определить тип накопителя на гибких магнитных дисках.
11. Определить монохромен ли активный видеоадаптер, используя переменную BIOS?

1.2 Лабораторная работа №2. Использование средств BIOS для работы с клавиатурой. Использование средств BIOS для работы с видеоадаптерами

Цель работы: проверить буфер клавиатуры на наличие в нём символов, а также получить состояние переключающих клавиш, изменить начертания заданных символов, установив соответствующий шрифт, вывести их начертания.

Общие сведения.

Функции BIOS - INT 16H: сервис клавиатуры.

Это - интерфейс прикладного уровня с клавиатурой. Нажатия клавиш на самом деле обрабатываются асинхронно на заднем плане, когда клавиша получена от клавиатуры, она обрабатывается прерыванием INT 09H и помещается в циклическую очередь.

AH сервис.

--- ----------

00H читать (ожидать) следующую нажатую клавишу
выход: AL = ASCII символ (если AL=0, AH содержит расширенный код ASCII). AH = скан-код или расширенный код ASCII

--- ----------

01H проверить готовность символа (и показать его, если так)
выход: ZF = 1 если символ не готов.
ZF = 0 если символ готов. AX = как для подфункции 00H (но символ здесь не удаляется из очереди).

--- ----------

02H читать состояние shift-клавиш. определить, какие shift-клавиши нажаты в данный момент, находится ли клавиатура в состоянии NumLock.
выход: AL = статус клавиатуры – смотреть флаги клавиатуры.

Использование прерывания int 10.

Функция 00H устанавливает видеорежим. Работает на следующих видеоадаптерах: MDA, CGA, EGA, MCGA, VGA. Выбирает текущий видеорежим и активный видеоконтроллер. Для вызова нужно установить значение регистра и затем использовать процедуру вызова прерывания Intr ($10,R), где R - регистр.

Вызов:
AH := 00H
AL := <режим>

Возвращаемое значение - 0.

Режим 02H = 80 символов x 25 строк, 16 цветов - текстовый режим;

Режимы 4H = 320x200x4, 6H: 640x200x2, 0FH: 640x350x21, 10H: 640x350x16Ю, 12H: 640x480x16, 13H: 320x200x256 –графические.

Функция 01H устанавливает размер курсора. Выбирает начальные и конечные строки развёртки для аппаратного мигающего курсора в текстовом режиме.

Вызов:
AH=01H
CH биты 0-4 - номер начальной строки развёртки для курсора
CL биты 0-4 - номер конечной строки развёртки для курсора
Возвращаемое значение отсутствует.

Функция 02H - перемещение курсора. Позиционирует курсор, использует тестовые координаты.
AH = 02H
BH = страница
DH = строка (y - координата)
DL = столбец (x - координата)
Возвращаемое значение отсутствует.

Функция 03H - получить текущую позицию курсора.
Вызов:
AH = 03H
BH = страница
Возвращаемое значение:
DH = строка (x - координата)
DL = столбец (y - координата)
CH = начало строки развёртки курсора
CL = конец строки развёртки курсора

Функция 05H - установить страницу дисплея.
AH = 05H
AL = страница {может быть от 0 до 7}

Функция 0AH - вывести символ на экран в текущую позицию курсора. Символ наследует атрибуты того символа, который ранее находился в этой позиции.
Вызов:
AH = 0AH
AL = символ
BH = страница
BL = цвет
CX = число символов
Возвращаемое значение отсутствует.

Функция 0CH - вывести графический пиксель. Рисует на дисплее точку в указанных графических координатах.
Вызов:
AH = 0CH
AL = значение пикселя
BH = страница
CX = столбец (x - координата)
DX = строка (y - координата)
Работает только на графических картах.
Возвращаемое значение отсутствует.

Функция 0DH - прочитать графический пиксель.
Вызов:
AH = 0DH
BH = страница
CX = столбец (x - координата)

DX = строка (y - координата)
Возвращаемое значение:
AL = цвет пикселя

Функция 0FH - получить текущий текстовый режим. Возвращает установленный в данный момент режим активного видеоадаптера.
Вызов:
AH = 0FH
Возвращаемые значения:
AH = количество столбцов символов на экране
AL = видеорежим
BH = страница

Функция 10H - подфункция 01H - установить цвет границы экрана.
Вызов:
AH = 10H
AL = 01H
BH = значение цвета
Работает на EGA и VGA. Возвращаемое значение отсутствует.

Задание А.

Составить программу на языке Си, выводящую на экран в цикле какой-либо символ. Нужно, чтобы работа программы завершалась при нажатии клавиши Esc, если *необходимая клавиша* находится *во включенном или выключенном состоянии.*

Методические указания.

1. Байт оперативной памяти по адресу FE00:1FFF интерпретируется следующим образом : FF - IBM PC; FE - IBM PC/XT; FC - IBM PC/AT.

2. По адресу F000:FFF5 записывается номер версии BIOS.

3. Объём оперативной памяти читается по адресу 0:0413.

4. Установленное в ПЭВМ оборудование можно определить, прочитав значение переменной BIOS по адресу 0:0410. Значения её битов следующие : если бит 0 в единице, то присутствует НГМД; если бит 1 в единице, то имеется сопроцессор (XT,AT); биты 2,3 задают базовую память (в AT не используются); биты 4,5 задают активный видеоадаптер (11 - монохромный, 10 - цветной с разрешением 80 строк по 25 знаков, 01 - цветной с разрешением 40 строк по 25 знаков); биты 6,7 задают число дисководов (00 - 1, 01 - 2, 10 - 3, 11 - 4); бит 8 не используется для PC,XT,AT; биты 9...11 задают число коммуникационных адаптеров от 0(000) до 7(111); если бит 12 в единице, то имеется игровой адаптер; бит 13 не используется для PC,XT,AT; биты 14,15 задают число установленных принтеров от 0(00) до 3(11).

5. Ниже указываются десятичные номера некоторых регистров CMOS-памяти и данные, которые в них записаны : 0 - секунды; 2 - минуты; 4 - часы; 6 - день недели; 7 - день месяца; 8 - месяц; 9 - год; 18 тип фиксированного диска (биты 7...4 - первый диск, 3...0 - второй); 20 - установленное

оборудование (биты 7...6 : 00 один НГМД, 01 - два НГМД; биты 5...4 : 01 - цветной дисплей с разрешением 40x25, 10 - цветной дисплей с разрешением 80x25, 11 - монохромный дисплей; биты 3,2 не используются; бит 1 : 1 - имеется сопроцессор, 0 - отсутствует; бит 0 : 1 - имеется НГМД, 0 - отсутствует); 21 - объём памяти в килобайтах на системной плате (младший байт); 22 - объём памяти в килобайтах на системной плате (старший байт); 23 - объём общей памяти в килобайтах (младший байт); 24 - объём общей памяти в килобайтах (старший байт); 48 - память сверх 1 Мб (младший байт); 49 - память сверх 1 Мб (старший байт); 50 - текущее столетие.

Пример.

Задание: составить программу на языке Си, выводящую на экран в цикле символ '*'. Нужно, чтобы работа программы завершалась при нажатии клавиши Esc, если левый Shift не нажат.

Решение.

В задаче в бесконечном цикле for(;;) выводится символ '*'. Нужно остановить этот вывод, нажав на клавишу Esc, причем, должно быть учтено состояние переключающей клавиши (левый Shift). Для проверки того, есть ли в буфере клавиатуры коды нажатых клавиш, используется функция 01h прерывания 16h. Для этого на входе в регистр AH заносится 01h. Тогда на выходе регистр AL содержит ASCIIZ-код символа или 0, если AH содержит расширенный ASCIIZ-код символа; а регистр AH - скэн-код или расширенный ASCIIZ-код символа.

При этом флаг ZF в регистре флагов микропроцессора устанавливается в нуль, если в буфере имеется код нажатой на клавиатуре клавиши. Но программа не переводится в состояние ожидания даже если буфер клавиатуры пуст. В этом случае ZF устанавливается в единицу и управление возвращается программе.

Вызывается прерывание 16h посредством функции int86, прототип которой int int86(int intno, union REGS inregs, union REGS outregs) описан в заголовочном файле dos.h. Эта функция загружает внутренние регистры микропроцессора значениями, записанными в объединении по шаблону union REGS, на начало которого указывает inregs, и выполняет прерывание с номером intno.

Значения внутренних регистров на выходе из прерывания записываются в объединение по шаблону union REGS, на начало которого указывает outregs.

Шаблон union REGS описан в заголовочном файле dos.h и представляет собой объединение двух структур : union REGS

```
    {
      struct WORDREGS x;
      struct BYTEREGS h;
    }
  struct WORDREGS
```

```
    {
     unsigned int ax,bx,cx,dx,si,di,cflag,flags;
    }
 struct BYTEREGS
    {
     unsigned char al,ah,bl,bh,cl,ch,dl,dh;
    }
```

Так как флаг ZF - это 6-й бит регистра флагов микропроцессора, то он выделяется посредством выражения rg.x.flags&0x40.

Скэн-код клавиши Esc равен 1, функция 02h прерывания 16h позволяет получить состояние переключающих клавиш. Для этого на входе в регистр AH заносится 02h, а на выходе AL содержит байт состояния переключающих клавиш.

Состояние переключающей клавиши левый Shift определяется выражением rg.h.al&0x02. Это следует из того, что формат байта состояния соответствует формату байта, находящгося в области данных BIOS по адресу 0000h:0417h. Ниже приведена информация при установке соответствующего бита этого байта в 1.

Бит 0 - нажата и не отпущена клавиша правый Shift,
1 - нажата и не отпущена клавиша левый Shift,
2 - нажата и не отпущена клавиша Ctrl,
3 - нажата и не отпущена клавиша левый Alt,
4 - включён ScrollLock,
5 - включён NumLock,
6 - включён CapsLock,
7 - включён режим вставки Ins.

Дальнейшие пояснения содержатся в комментариях, указанных в листинге задачи.

Пример программы в кодах приведен в Приложении В.

Варианты задания А.

1. Выключённое состояние клавиши CapsLock.
2. Включённое состояние клавиши CapsLock.
3. Выключенное состояние клавиши ScrollLock.
нное состояние клавиши ScrollLock.
5. Выключенное состояние клавиши Insert.
6. Включённое состояние клавиши Insert.
7. Выключенное состояние клавиши NumLock.
8. Включённое состояние клавиши NumLock.
9. Нажата правая клавиша Shift.
10. Нажата клавиша ESC.
11.Нажата правая клавиша Ctrl.
12. Нажата левая клавиша Ctrl.
13. Нажата левая клавиша Ctrl.

14. Нажата клавиша PrintScreen.
15. Нажата клавиша Tab.
16. Нажата клавиша Enter.
17. Выключенное состояние клавиши NumLock.
18. Включенное состояние клавиши NumLock.
19. Нажата клавиша Alt.
20. Включённое состояние клавиши Ins.

Задание Б.

Выбрав вариант, изменить начертания двух заданных символов, ASCII-коды которых разнятся на 1, а также установить соответствующий шрифт, как указано в таблице 1. До и после изменения вышеупомянутых символов вывести их начертания.

Методические указания.

В текстовом режиме символы выводятся на основании информации об их начертании в определенной битовой матрице. Такие матрицы, например, для адаптеров EGA/VGA могут иметь разные размеры :
8x8,8x14, 8x16, 9x14,9x16 точек. Функция 11h прерывания 10h имеет различные подфункции, позволяющие устанавливать те или иные режимы работы со шрифтами.

Например, для загрузки шрифта 8x14 перед вызовом прерывания 10h в регистр AH заносим 11h (номер функции), в AL - 1 (номер подфункции), в BL - 0 (нулевой блок знакогенератора). Загрузка шрифтов 8x16 и 8x8 производится аналогично, лишь с той разницей, что здесь в регистр AL помещаются функции 14h и 12h соответственно.

Для загрузки оригинального шрифта пользователя в регистр AL заносится 0. При этом нужно создать в оперативной памяти область данных, содержащих новые начертания символов.

Например, 8x16 матрица

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

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

соответствует знаку "крест" (типа +). При этом светящимся пикселам соответствуют единицы, а темным - нули. Приведенная матрица может быть записана в виде символьной строки

"\x18\x18\x18\x18\x18\x18\x18\xff\xff\x18\x18\x18\x18\x18\x18\x18"

Далее, в регистры ES:BP заносится адрес символьной строки, задающей указанным выше образом новые начертания для нескольких подряд идущих ASCII-кодов, а в регистр BH заносится число строк пиксельной матрицы (8, 14 или 16).

Число подряд идущих символов, подлежащих перекодировке, заносится в регистр CX, а в регистр DX - код первого из этих символов.

Пример.

Задание: изменить начертания двух символов X(ASCII-код 88) и Y(ASCII-код 89) соответственно на символы типа прямой и перевернутой буквы T. Показать изображения символов до и после изменения.

Решение.

В рассматриваемом примере перед вызовом прерывания 10h имеем в регистре BH 16, BL - 0, CX - 2, DX - 88, AL - 0, AH - 11.

После вызова рассматриваемого прерывания символы указанных кодов изменяют своё начертание в соответствии с указанным в битовых матрицах.

Пример программы в кодах приведен в Приложении Г.

Варианты задания Б.

Т а б л и ц а 1

Номер варианта	Исходные символы	ASCII-коды	Шрифт	Новые Символы
1	? @	63 64	8x16	M З
2	^ -	94 95	8x8	C T
3	% &	37 38	8x14	O П
4	? @	63 64	8x16	I A
5	^ -	94 95	8x8	Ж K
6	% &	37 38	8x14	Ф X
7	? @	63 64	8x16	K Ч
8	^ -	94 95	8x8	Г Д
9	% &	37 38	8x14	Z X
1 0	% &	37 38	8x14	A L
11	? @	63 64	8x16	П Д
12	% &	37 38	8x14	P Л
13	^ -	94 95	8x8	O Ж
14	% &	37 38	8x14	J Y
15	? @	63 64	8x16	E G

Основная литература: [1] - 32-52 с., [2] - 43-62с.
Дополнительная литература:[3] - 46-59с., [4] - 26-45с.

Контрольные вопросы.

1. Обзор функций BIOS int 16h.
2. Описать шаблон union REGS.
3. Что называется расширенным ASCIIZ-кодом символа?
4. Чем определяется состояние переключающей клавиши левый Shift ?

5. Как работает 01h прерывания 16h?
6. Как работает функция int86?
7. Как в программе осуществляется проверка буфера клавиатуры?
8.Чему равен scan-код клавиши ESC?
9. Работа с функцией 00H.
10. Назначение функций 01H, 02H, 03H .
11. Назначение функций 05H, 0CH, 0DH .

1.3 Лабораторная работа №3. Системные возможности по работе с дисками и буфером клавиатуры

Цель работы: обрести навыки работы по работе с дисками и буфером клавиатуры.

*Задание***:** требуется составить программу на языке Си по вариантам.

Общие сведения.

Функции DOS - INT 25H/26H: Прямая дисковая операция чтения/записи.

Вход.

AL = номер диска (0=A, 1=B, и т.д.). CX = счетчик считываемых или записываемых секторов. DX = начальный сектор (логический номер сектора DOS). DS:BX = адрес данных (исходный или целевой буфер).

Выход.

AX = код ошибки, если взведен флаг переноса (CF=1) (см. ниже). Значения всех регистров меняются, за исключением DS,ES,SS,SP. Эти функции оставляют одно лишнее слово в стеке.

Этот сервис DOS предоставляет прямой доступ к любому сектору диска, доступному через DOS-BIOS или устанавливаемые драйверы устройств. INT 25H- читает секторы, INT 26H - записывает секторы.

Логические секторы DOS начинаются с корневого сектора . первый сектор в разделе DOS - это логический сектор 0. номера логических секторов возрастают сначала для каждого сектора на цилиндре, затем по каждой головке, и наконец по каждому цилиндру на устройстве.

Главная корневая запись и любые другие секторы вне раздела DOS недоступны через этот сервис. Прерывание INT 13H - это единственный сервис, который предоставляет полный доступ к любой части твердого диска.

Ошибки.

Ошибки, возникающие в процессе INT 25H/26H, не обрабатываются обработчиком критических ошибок INT 24H . При выходе из INT 25H/26H, DOS-BIOS взводит флаг переноса (CF=1), когда встречается ошибка. Если установлен флаг CF:

AL = ошибка устройства (0-0cH) - аналогично битам 0-7 в DI для INT 24H - это

AL = один из кодов ошибок 13H-1fH (без 13H) в списке кодов ошибок DOS, AH = 40H = ошибка операции поиска на диске, AH = 08H = неверная контрольная сумма (CRC) при чтении дискеты, AH = 04H = запрошенный сектор не найден, AH = 03H = попытка записи на защищенную от записи дискету, AH = 02H = ошибка, отличная от перечисленных выше

Пример.

Участок программы, читающий корневой сектор дискеты:

```
mov     al,0          ;выбрать диск A
mov     dx,0          ;выбрать корневой сектор DOS
mov     cx,1          ;читать один сектор ...
lea     bx,my_buf  ; ... в буфер по адресу DS:BX
int     25H
pop     dx            ;удалить лишнее слово из стека
jnc     no_err
..и т.д.              ;обработать любую ошибку диска (код в AX)
no_err:  ...и т.д.    ;продолжить (ошибок нет)
```

Понятие о скан-кодах клавиш.

При нажатии клавиши на клавиатуре генерируется её скан-код. Скан-код можно считать «номером клавиши» на клавиатуре. Однако это не совсем так. Например, если скан-код клавиши «F10» равен «68», то тот же код клавиши с нажатой клавишей «Shift» будет равен «93», с клавишей «Ctrl» – «103», а с клавишей «Alt» – «113».

Существуют так называемые «алфавитно-цифровые» («белые») и «управляющие» («серые») клавиши. Первые генерируют скан-код и ASCII символы. Вторые генерируют только расширенный скан-код, а в поле «ASCII символа» стоит ноль. Благодаря этому программист легко может понять, как обрабатывать код клавиши: выводить ли символ на экран или запускать управляющую последовательность действий.

Более подробно о скан-кодах клавиш смотри [1, 8, 33].

«Эхопечать» символов.

При вводе с клавиатуры пара значений «Скан-код» – «ASCII символ» заносится в клавиатурный буфер для её обработки на компьютере. При этом на старых терминалах все введённые символы отображались на экране дисплея. Но, с появлением персональных ЭВМ, выяснилось, что такой вывод («Эхопечать») не всегда удобный. Что делать, если Вы вводите пароль, и Вы не хотите, чтобы кто-либо его прочитал с экрана? А что делать, если Вы используете алфавитно-цифровую клавишу как управляющую (например, при обработке меню)? В этом случае Вам помогут функции ввода с клавиатуры без эхопечати.

Клавиатурный буфер.

Полученный в результате преобразований в контроллере двухбайтовый код посылается в кольцевой буфер ввода, который служит для синхронизации ввода данных с клавиатуры и приёма их выполняемой программой. Объём кольцевого буфера составляет 15 слов (30 байт). При этом буфер организован по принципу: «первым записан – первым считан» (английская аббревиатура «FIFO»). При переполнении буфера новые коды в него не поступают, а нажатие на клавиши вызывает предупреждающие сигналы.

Консольный ввод в Си.

В языке Си и его разновидностях существуют следующие функции для ввода данных с клавиатуры:

1) cgets – чтение строки с консоли;
2) getch – чтение символа с консоли;
3) getche – чтение символа с эхопечатью;
4) kbhit – проверка нажатия клавиши на консоли;
5) cscanf – чтение данных с консоли по формату;
6) getpass – ввод с терминала пароля без эхопечати.

Рассмотрим некоторые из них поподробнее.

Функция getch.

Описана в: <conio.h> /* используется только для описания функции */

Синтаксис: int getch();

Назначение: ввод одиночного символа с консольного терминала без эхопечати (без вывода на дисплей).

Функция возвращает: прочитанный символ.

Примечание: функция getch использует входной поток stdin.

Функция getche.

Описана в: <conio.h> /* используется только для описания функции */

Синтаксис: int getche();

Назначение: функция читает одиночный символ с консольного терминала, с эхопечатью введённого символа на экране с текущей позиции курсора.

Возвращаемое значение: прочитанный символ.

Примечание: 1) Нет ошибочных кодов возврата.
2) Функция корректно обрабатывает «ctrl+break».

Функция cgets.

Описана в: <conio.h> /* используется только для описания функции */

Синтаксис: char *cgets(str);
char *str;

Назначение: функция читает строку символов непосредственно с консоли, и помещает строку и её длину по указателю str. Аргумент str должен быть указателем на массив символов;

Замечание: первый элемент массива str[0] должен содержать максимально допустимую длину считываемой строки. Массив должен иметь достаточную длину, чтобы поместить строку, символ ASCII 0 плюс два байта дополнительно для дескриптора; cgets читает символы до тех пор, пока не прочитаны символы <CR> и <LF>, или пока не будет прочитано установленное число символов. Прочитанная строка начинается со str[2]. Если прочитана комбинация <CR>/<LF>, эти символы будут располагаться за символом конца строки (ASCII 0). Действительная длина строки помещается в str[1];

Возвращаемое значение: указатель на начало строки, то есть на адрес элемента str[2];

Примечание: 1) Нет кодов ошибок.
2) Длина строки ограничена ~ 250 символами.

Задание А.

Составить две программы на языке Си.

Первая из которых читает загрузочную запись диска (BOOT-сектор) в специальную структуру и затем получает из нее значение поля, указанного в варианте.

Вторая же программа помещает загрузочную запись в некоторую область памяти длины 512 байт и, увеличив адрес её начала на нужную величину, непосредственно получает требуемый результат.

Методические указания.

Самый первый сектор жесткого диска (сектор 1, дорожка 0, головка 0) содержит так называемую главную загрузочную запись (Master Boot Record). Эта запись занимает не весь сектор, а только его начальную часть. Сама по себе главная загрузочная запись является программой. Эта программа во время начальной загрузки операционной системы с жесткого диска помещается по адресу 7C00:0000, после чего ей передается управление. Загрузочная запись продолжает процесс загрузки операционной системы. Загрузочная запись считывается в область памяти длины 512 байт. При этом используется прерывание DOS INT 25h, которое читает сектор по его логическому номеру.

На входе этого прерывания в регистр AL заносится адрес НГМД или НМД (0 - A:, 1 - B:, ...); в CX - количество секторов, которые нужно прочитать; в DX - логический номер начального сектора; в DS:BX - адрес буфера для чтения. На выходе прерывания INT 25h в регистр AH помещён код ошибки при неуспешном завершении операции, а флаг CF содержит 1, если произошла ошибка и 0, если ошибки нет. Поскольку в рассматриваемом

прерывании значения задаются и в сегментном регистре, то для реализации прерывания используется функция int86x.

Пример.

Задание: найти количество секторов на одну копию FAT.

Решение.

Для реализации используется функция int86x, прототип которой описан в dos.h и имеет вид:

```
int int86x(int intno, union REGS *inregs, union REGS *outregs,
struct SREGS *segregs)
```

Описание структуры REGS приводится в лабораторной работе 3.

Шаблон структуры SREGS содержится в файле dos.h и имеет вид

```
struct SREGS {
      unsigned int es;
      unsigned int cs;
      unsigned int ss;
      unsigned int ds;
    };
```

По выходу из прерывания в переменную segreg копируются значения всех сегментных регистров. В функции getbootc() указано, как заполняются регистровые структуры для вызова прерывания 25h. В результате вызова функции getbootc() в буфер, выделенный функцией malloc(), считывается загрузочная запись указанного НМД.

Адрес буфера присваивается far-указателю boot_rc.

Далее, чтобы ответить на вопрос задачи, т.е. определить количество секторов на одну копию FAT, следует воспользоваться форматом загрузочной записи. Искомая величина находится в блоке параметров BIOS (BPB) со смещением 11 байт от его начала, а сам блок параметров BIOS располагается со смещением 11 байт от начала загрузочного сектора.

Поэтому количество секторов на одну копию FAT определяется величиной *(unsigned *)(boot_rc+22). Здесь операция приведения типа (unsigned *) сообщает компилятору размер объекта.

Пример программы в кодах приведен в Приложении Д.

Варианты задания А.

1. Найти количество элементов FAT-таблиц.
2. Найти количество байтов в одном секторе.
3. Найти максимальное количество дескрипторов файлов в корневом каталоге диска.
4. Найти количество секторов в одном кластере.
5. Найти общее количество секторов на носителе данных в разделе DOS.
6. Найти физический номер дисковода.
7. Найти серийный номер диска.

8. Найти метку диска.
9. Найти количество магнитных головок.
10. Найти количество секторов на дорожке.
11. Найти сколько кластеров содержится на данном диске.
12. Найти номер дисковода для функций BIOS.
13. Определить тип устройства.
14. Определить размер одной FAT в секторах.
15. Вывести на консоль аббревиатуру операционной системы.
16. Вывести на консоль аббревиатуру файловой системы.
17. Номер главной таблицы FAT.
18. Определить версию FAT32.
19. Найти размер диска в секторах.
20. Определить дату/время создания диска.

Задание Б.

Записать в буфер клавиатуры последовательность символов согласно выбранному варианту: с использованием функции 5 прерывания 16h, осуществляя непосредственный доступ к ячейкам памяти, выделенным под буфер клавиатуры, по текущему значению указателя «головы» буфера или по текущему указателю "хвоста" буфера.

Методические указания.

Буфер клавиатуры организуется, как кольцевая очередь, доступ к которой осуществляется с помощью указателя «головы» : (head pointer), адрес которого 40h:1Ah, и указателя «хвоста» с адресом 40h:1Ch. Указатель «хвоста» задаёт смещение, где будет записан обработчиком прерывания 9 двухбайтовый код буферизуемой клавиши. Указатель «головы» задаёт смещение возвращаемого слова по запросу операционной системы или BIOSa. Буфер клавиатуры занимает 32 байта памяти с адреса 40h:1Eh до 40h:3Eh. Если значения указателей «головы» и «хвоста» равны между собой, то буфер пуст.

Пример.

Задание: составить программу на языке Си, которая заносит в буфер клавиатуры фамилию «Ivanov», осуществляя непосредственный доступ к ячейкам памяти, выделенным под буфер клавиатуры по текущему значению указателя «хвоста» буфера.

Решение.

В приведенном листинге функция int enter_kbt(unsigned key_code) вводит с «хвоста» в буфер клавиатуры символ key_code.

Используются переменные tail, head, tmp типа register unsigned _es * . tail - это адрес, по которому хранится адрес «хвоста»; head - адрес, содержащий адрес «головы»; tmp - значение * tail указателя «хвоста»,

преобразованное к виду (unsigned _es *). В случае, когда tmp указывает на конец буфера, то ему присваивается значение на начало буфера. Поэтому значение указателя «хвоста» может быть и меньше значения указателя «головы».

Если tmp+1=(unsigned _es *)*head, то буфер полон и происходит выход из функции.

Указанные комментарии поясняют текст программы. Пример программы в кодах приведен в Приложении Е.

Варианты задания Б.

1. Создать файл по имени «f1» и записать в него слово «name», поместив в буфер клавиатуры указанную последовательность кодов символов и кодов клавиш:

<Shift/F4>,f1,<Enter>,name,<Esc>,<Enter>

2. Вызвать Турбо-Си, создать в его среде файл по имени «f1» и записать в этот файл слово «name», поместив в буфер клавиатуры указанную последовательность кодов символов и кодов клавиш:

tc,<Enter>,<Alt/F>,<Enter>,f1,<Enter>,name,<F2>,<Alt/x>

3. Поместить в буфер клавиатуры имя выполняемой программы и код <Enter>, что позволит после окончания работы данной программы запустить её вновь и так до тех пор, пока не произойдёт перезагрузка машины.

4. Поместить в буфер клавиатуры последовательность кодов символов и код <Enter>, позволяющих после окончания работы программы удалить её исполняемый файл, предполагая, что он находится в текущем каталоге.

5. Сменить активность окон программной оболочки FAR, поместив в буфер клавиатуры код клавиши <Tab>.

6. Выйти из программной оболочки Windows Commander, поместив в буфер клавиатуры коды клавиш <F10>,<Enter>.

7. Получить оглавление каталога текущего диска, поместив в буфер клавиатуры коды символов и клавиш : dir,<Enter>.

8. Подсветить все файлы и подкаталоги текущего каталога, поместив в буфер клавиатуры коды клавиш : <gray +>,<Enter>.

9. Убрать левое окно программной оболочки Windows Commander, поместив в буфер клавиатуры коды клавиш : <Ctrl>,<F1>.

10. Получить справку о распределении памяти на текущем диске, поместив в буфер клавиатуры коды клавиш : <Ctrl>,<L>.

11. Создать файл по имени «cc» и записать в него слово «kode», поместив в буфер клавиатуры указанную последовательность кодов символов и кодов клавиш :

<Shift/F4>,cc,<Enter>, kode,<Esc>,<Enter>

12. Вызвать Турбо-Си, создать в его среде файл по имени «cc» и записать в этот файл слово «kode», поместив в буфер клавиатуры указанную последовательность кодов символов и кодов клавиш:

tc,<Enter>,<Alt/F>,<Enter>,f1,<Enter>,name,<F2>,<Alt/x>

13. Получить оглавление каталога KATAL, поместив в буфер клавиатуры коды символов и клавиш: dir,<Enter>.

14. Поместить в буфер клавиатуры последовательность кодов символов строки «KODE» и код <Enter>, позволяющих после окончания работы программы удалить её исполняемый файл.

15. Подсветить все файлы каталога K1, поместив в буфер клавиатуры коды клавиш : <gray +>,<Enter>.

16. Убрать правое окно программной оболочки Windows Commander, поместив в буфер клавиатуры коды клавиш : <Ctrl>,<F2>.

17. Вызвать панель устройств в правом окне программной оболочки Windows Commander, поместив в буфер клавиатуры коды клавиш : <Alt>,<F2>.

18. Вызвать панель устройств в левом окне программной оболочки Windows Commander, поместив в буфер клавиатуры коды клавиш : <Alt>,<F1>.

19. Войти в меню программной оболочки Norton Commander, поместив в буфер клавиатуры коды клавиш <F9>.

20. Создать файл по имени «fgg.txt» и записать в него слово «save», по местив в буфер клавиатуры указанную последовательность кодов символов и кодов клавиш:

<Shift/F4>, fgg.txt,<Enter>, save ,<Esc>,<Enter>

Основная литература: [2] – 2- 37 с**.**
Дополнительная литература:[3] – 73-86 с.

Контрольные вопросы:

1. Функции DOS – int 25H/26H.
2. Назвать скан-коды основных клавиш.
3. Как найти серийный номер диска?
4. Консольный ввод в Си.
5. Назначение клавиатурного буфера.
6. Вид функции int86x.
7. Функции чтения строк и символов в Си.
8. Как выполнить размещения в буфер клавиатуры имя выполняемой программы?
9. Как получить справку о распределении памяти на текущем диске?

Заключение

В представленной разработке основной акцент сделан на самостоятельную работу студентов при выполнении ими лабораторных работ. Теоретический материал, подробные примеры и листинги программ позволяют самостоятельно изучить материал и успешно выполнить задания.

В результате выполнения лабораторных работ студентами приобретаются знания системного программного обеспечения и навыки системного программирования на языке Си.

Приложение А

```
                /*      Текст программы и выходные результаты      */
#include <stdio.h>
#include <dos.h>
main()
{
  unsigned cx;
  char s[]="PROBA";   /*  Это имя файла  */
  _DS=FP_SEG(s); _DX=FP_OFF(s);
  _AX=0x4300;
  geninterrupt(0x21);
  cx=_CX;
  if((cx&4)==0)  /*  Если не системный  */
  /*  Скобки выше существенны для порядка операций  */
  {
     printf("Файл proba не был системным, сделали его архивным\n");
     cx=cx|32; _CX=cx; /*  Занести 1 в 5-й бит
                       байта состояния  */
     _AX=0x4301; geninterrupt(0x21);
  }
  else printf("Файл proba системный\n");
}
Файл proba не был системным, стал  архивным
Файл proba системный
```

Приложение Б

```
                /*      Текст программы  и выходные результаты        */
#include <stdio.h>
#include <dos.h>
main()
{
  int h,i; unsigned char c,cr[2];
  char dsk[2]={'A','B'};
  /*  Определить наличие сопроцессора  */
  h=peek(0,0x410);
  if(h&2) puts("Есть сопроцессор\n");
  else puts("Нет сопроцессора\n");
  /*  Определить тип накопителя на гибких дисках  */
  /*  Читаем нужную ячейку КМОП-памяти  */
  outportb(0x70,0x10); c=inportb(0x71);
  cr[0]=(c>>4); cr[1]=(c&0x0f);
```

```
 for(i=0;i<2;i++)
 {
  switch(cr[i])
  {
   case 0 : printf("Дисковод %c: не установлен\n",
   dsk[i]); break;
   case 1 : printf("Дисковод %c: на 360 Кб\n",
   dsk[i]); break;
   case 2 : printf("Дисковод %c: на 1.2 Мб\n",
   dsk[i]); break;
   case 3 : printf("Дисковод %c: на 720 Кб\n",
   dsk[i]); break;
   case 4 : printf("Дисковод %c: на 1.44 Мб\n",
   dsk[i]); break;
  }
 }
}
```

```
Нет сопроцессора
Дисковод A: на 1.44 Мб
Дисковод B: не установлен
```

Приложение в

```
                /*      Текст программы  и выходные результаты          */
#include <stdio.h>
#include <dos.h>
void main(void)
{
 union REGS rg;
 int i,zflag;
 for(;;)
 {
  /*  Выводим в цикле символ  '*'  */
  putchar('*');
  /*  Небольшая задержка во времени  */
  for(i=0;i<1000;i++) ;
  /*  Вызываем прерывание INT 16h для проверки
                        буфера клавиатуры  */
  /*  Устанавливаем флаг, который будет сброшен
              при нажатии на любую клавишу  */
  zflag=1;
  rg.x.ax=0x0100;
```

```
   /*  01x - номер функции (проверка буфера клавиатуры)
                         для прерывания INT 16h  */
   int86(0x16,&rg,&rg);
   zflag=rg.x.flags&0x40; /*  Получаем в zflag значение
                  разряда ZF из регистра флагов
                  микропроцессора    */
   if(zflag==0)
   {
    /*  Если флаг сброшен т.е. нажали к.л. клавишу,
        то читаем код нажатой клавиши из буфера при
        помощи функции 00h прерывания INT 16h   */
    rg.h.ah=0; int86(0x16,&rg,&rg);
    /*  Если была нажата клавиша <ESC>,
        то завершаем работу программы при условии,
        что переключатель Shift (левый) не нажат        */
    if(rg.h.ah==1)
    {
      /*  Дополнительно проверяем состояние клавиши
          Shift (левый), этой клавише отвечает бит 0x02
          в слове состояния                        */
      rg.h.ah=2; int86(0x16,&rg,&rg);
      if((rg.h.al&0x02)==0)
      /*  Если Shift (левый) не нажат  */  break;
      else printf("\nДля завершения нажмите <ESC> "
                  "при неприжатой клавише Shift (левый)\n");
    }
    else printf("\nДля завершения нажмите <ESC> "
                "при неприжатой клавише Shift (левый)\n");
   }
  }
}
```

Приложение Г

```
                /*      Текст программы  и выходные результаты        */
#include <stdio.h>
#include <dos.h>
void main()
{
  int i;
  unsigned char *d;  /*  d - Указатель на новую таблицу  */
  /*  новая таблица для замены символов X с кодом ASCII 88
      и Y с кодом ASCII 89 на другие символы           */
  d="\xff\xff\x18\x18\x18\x18\x18\x18"
```

```
   "\x18\x18\x18\x18\x18\x18\x18\x18"
   "\x18\x18\x18\x18\x18\x18\x18\x18"
   "\x18\x18\x18\x18\x18\x18\xff\xff";
   clrscr(); /*  Очистка экрана  */
   puts("Old symbols  XXXXX  and  YYYYY ");
   printf("Для продолжения нажмите любую клавишу\n");
   getch();
   clrscr(); /*  Очистка экрана. */
   _ES=_DS;  /*  ES устанавливается на область данных */
   _BL=0;    /*  Нулевой блок для загрузки  */
   _AH=0x11; /*  номер функции 11  */
   _AL=0;    /*  номер подфункции 0, означает -
               загрузить шрифт пол-ля, а при _AL=1, например,
               была бы загрузка монохромного шрифта из ПЗУ  */
   _BH=16;   /*  16 байт на символ (если б шрифт исходный
               был не обычный Font 8X16, а, скажем, Font 8X14,
               то писали бы _BH=14; но по умолчанию машина делает
               Font 8X16  )  */
   _CX=2;    /*  заменяются только 2 символа  */
   _DX=88;   /*  заменяются символы - % и &, с кодами 88, 88+1  */
   _BP=(unsigned)d;  /*  ES:BP должен указывать на заданную
                         таблицу с именем block  */
   geninterrupt(0x10);  /*  Вызов прерывания 0x10  */
   puts("New symbols  XXXXX  and  YYYYY");
}
```

Приложение Д

(Вариант из лабораторной работы №5)

```
                /*      Текст программы и выходные результаты       */
#include <stdio.h>
#include <alloc.h>
#include <dos.h>
/*     Функция getbootc() считывает загрузочную запись
   указанного  НМД непосредственно в область из 512 байтов,
   возвращает 0 в случае успешного считывания, а иначе - 1 */
int getbootc(char *boot,int drive)
{ union REGS reg;
  struct SREGS segreg;
  /*  Заполняем регистровые структуры для вызова
      прерывания DOS int 25h     */
   reg.x.ax=drive;   reg.x.bx=FP_OFF(boot);
   segreg.ds=FP_SEG(boot); reg.x.cx=1;  reg.x.dx=0;
```

```
  int86x(0x25,&reg,&reg,&segreg);   return(reg.x.cflag);
}
void main(void)
{
 char far * boot_rc;
 int statusc;  char drive;
 /*  Заказываем буфер для чтения BOOT-записи  */
 /*  Адрес буфера присваиваем far-указателю  */
 boot_rc=malloc(512);  /* Такова длина сектора загрузочного */
 /*    Запрашиваем диск, для которого необходимо выполнить
    чтение загрузочной записи                          */
 printf("\nВвед.обозначен.диска (A, B, ... ) : ");
 drive=getche();
 /*  Вычисляем номер дисковода  */
 drive=toupper(drive)-'A';
 /*  Читаем загрузочную запись в буфер  */
 statusc=getbootc((char far *)boot_rc,drive);
 /*  Если произошла ошибка, то завершим работу  */
 if(statusc)

 {  printf("\nОшибка при чтении BOOT-сектора\n"); exit(-1);  }
 printf("\nКоличество секторов на одну копию FAT равно %d\n",
       *(unsigned *)(boot_rc+22)); /* 11 байт до начала
              подстр-ры EBPB, а там ещё сдвиг 11 байт  */
 /*  Освобождаем буфер  */     free(boot_rc);
}
Введ.обозначен.диска (A, B, ... ) : c

Количество секторов на одну копию FAT равно 8
```

Приложение E

```
                /*      Текст программы   и выходные результаты    */
#include <stdio.h>
#include <dos.h>
#include <process.h>
#include <conio.h>
#define KB_BUFFER_START 0x1e /* Смещение начала буфера клавиатуры */
#define KB_BUFFER_END   0x3e /* Смещение конца буфера клавиатуры */
#define HEAD_PTR        0x1a /* Смещение указателя "головы" */
#define TAIL_PTR        0x1c /* Смещение указателя "хвоста" */
#define OK       0   /* Код нажатия записан в буфер */
#define FULL     1   /* Буфер клавиатуры заполнен до предела */
/*  Записать один символ в буфер клав-ры с "хвоста"  */
```

```
int enter_kbt(unsigned key_code)
{
 unsigned register _es *tail=(unsigned _es *) TAIL_PTR,
                   _es *head=(unsigned _es *) HEAD_PTR,
                   _es *tmp,ret;
 _ES=0x40;
 disable(); /*  Начало критической секции кода */
   tmp=(unsigned _es *) *tail;
   if(tmp==(unsigned _es *)KB_BUFFER_END)
     /*  "Перескок" указателя на начало буфера */
     tmp=(unsigned _es *)KB_BUFFER_START;
   if(tmp+1==(unsigned _es *)*head) /*  Буфер полон */
     ret=FULL;
   else
     { *(unsigned _es *)tmp=key_code;
       tmp++;
       *tail=(unsigned)tmp;
        ret=OK;   }
 enable();  return ret;
}
main()
{  int i;
  unsigned key_codes[]={0x1749,0x2f76,0x1e61,0x316e,
  0x186f,0x2f76};
  /*  В массиве key_codes[] <I>,<v>,<a>,<n>,<o>,<v>       */
  while(kbhit()) getch(); /* Очистка буфера клавиатуры  */
  /* Запись в буфер клавиатуры массива key_codes с "хвоста".  */
  for(i=0;i<6;i++) enter_kbt(key_codes[i]);
}
```

Ivanov

Список литературы

1. Побегайло А.П.; Системное программирование в Windows. Наиболее полное руководство.СПб.: БХВ-Петербург, 2006.

2. Харт Джонсон. Системное программирование в среде Windows.М.: [и др.], Вильямс, 2007.

3. Вильямс Ал. Системное программирование в Windows 2000, Изд-во Питер, 2010.

4. Фельдман С. К.*; Системное программирование. Полный курс лекций с теоретическими материалами и лабораторными работами, 2005.

Сводный план 2013 г. поз. 197

Наталья Александровна Водолазкина

СИСТЕМНОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ

Методические указания к лабораторным работам
для студентов специальности
5В060200 – Информатика

Редактор Н.М. Голева
Специалист по стандартизации Н.К. Молдабекова

Подписано в печать ___.___._____. Формат 60х84 1/16
Тираж 30 экз. Бумага типографская №1
Объем __ уч.- изд. л. Заказ _____ Цена ____тн.