ФЕДЕРАЛЬНОЕ АГЕНТСТВО СВЯЗИ
Северо-Кавказский филиал
ордена Трудового Красного Знамени федерального государственного бюджетного образовательного учреждения высшего образования
"Московский технический университет связи и информатики"

Методические указания
для проведения лабораторной работы №1

по дисциплине

«СИСТЕМНОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ»

по теме

«Исследование механизма программных прерываний»

Направление подготовки:
09.03.01 Информатика и вычислительная техника

Профили **Программное обеспечение и интеллектуальные системы**
Вычислительные машины, комплексы, системы и сети

Ростов-на-Дону
2019

УДК 681.3.06 (076)
ББК 32.07

Чикалов А.Н. Системное программное обеспечение. Исследование механизма программных прерываний. Методические указания для проведения лабораторной работы №1. Ростов-на-Дону: Северо-Кавказский филиал МТУСИ, 2019.- 19 с.

В пособии изложены методические рекомендации и содержательные материалы для проведения практических и лабораторных занятий по изучению механизмов межпроцессорных коммуникаций в ЭВМ. Рассматриваются программные и аппаратные прерывания, технология использования интерфейса прикладного программирования.

Пособие содержит необходимые справочные материалы.

Методические указания предназначены для студентов, обучающихся по направлению подготовки 09.03.01 Информатика и вычислительная техника, профилей Вычислительные машины, комплексы, системы и сети, Программное обеспечение и интеллектуальные системы.

Пособие предназначено для использования при изучении дисциплин Системное программное обеспечение, а также может быть использовано преподавателями и студентами при изучении родственных дисциплин и в процессе самостоятельной работы.

Учебное пособие обсуждено и одобрено на заседании кафедры ИВТ
Протокол №1 от 26.08.2019 г.

Рецензент Зав. кафедрой ИВТ д.т.н. профессор Соколов С.В.

Лабораторная работа№1. Исследование механизма программных прерываний.

Цель

1. Выработать практические умения и приобрести навыки организации программных прерываний, оформления процедур обработки прерываний и их использования в вычислительном процессе.
2. Приобрести навыки применения интерфейса прикладного программирования (API). Углубить знания о механизмах реализации основных операций обработки данных на базе аппаратно-программного комплекса ПЭВМ.
3. Приобрести навыки анализа, обобщения и систематизации полученных результатов, навыки составления и оформления отчетных результатов.

Учебные вопросы

1. Исследование приемов использования прерываний, передачи параметров системным функциям и способов получения результатов.
2. Разработка и исследование программ, использующих прерывания, на основе справочной информации о функциях обслуживания.

Литература для подготовки к занятию

1. Гордеев А.В., Молчанов А.Ю. Системное программное обеспечение.- СПб.: Питер, 2001.
2. Олифер В.Г., Олифер Н.А. Сетевые операционные системы. – СПб.: Питер, 2001.
3. Финогенов К.Г. Самоучитель по системным функциям MS-DOS. - М.: Радио и связь, Энтроп, 1995.
4. Джордейн Р. Справочник программиста персональных компьютеров типа IBM PC, XT и AT. – М.: Финансы и статистика, 1992.
5. Данкан Р. Профессиональная работа в MS-DOS. – М.: Мир, 1993.

Содержание отчета

1. Название лабораторной работы и учебные вопросы.

2. Последовательно для каждого из разделов исследования:

- задания и разработанные программы;

- схемы и рисунки, поясняющие механизм реализации программных прерываний;

- пояснения к фрагментам рабочих программ.

3. Выводы о целесообразности реализации программных прерываний с использованием нестандартных процедур обработки прерываний.

4. Краткие ответы на те контрольные вопросы, которые ещё не нашли своего отражения в отчете.

Вопросы для самопроверки

1. Приведите определение операционной системы.
2. Каким образом осуществляется обращение к функциям DOS и BIOS?
3. Перечислите регистры процессора, поясните их назначение.
4. Поясните процедуру прерывания, проиллюстрируйте ее.
5. Приведите классификацию прерываний.
6. В чем различие между программными и аппаратными прерываниями?
7. Поясните понятие вектора прерываний.
8. Каким образом осуществляется обращение из прикладной программы к системным функциям?
9. Поясните порядок использования и программирования процедур, использующих программные прерывания.
10. Как вычислить адрес размещения вектора прерывания INT 10h, INT 21h?

Актуальность занятия

1. Механизм прерываний является одним из способов организации мультипрограммирования и основой работы любой ОС. Он позволяет реагировать на внешние события, происходящие асинхронно вычислительному процессу. Это важно, т.к. подобным образом организованы все системы реального времени, в том числе и системы военного назначения

2. Это эффективный способ вызова процедур ОС, использования разработчиком имеющихся ресурсов.

3. Это возможность получить доступ к средствам ОС в том числе из среды Pascal, Delphi и др.

Назначение и механизм прерываний

Прерывания – это принудительная передача управления программе обработки (процедуре) с последующим возвратом к исходному коду. По сути, это изменение порядка выполнения команд, которое может произойти в любой момент времени. Механизм обеспечивает приостановку текущей работы ПЭВМ для обработки ситуации, требующей немедленного вмешательства. Используется (цель):

- - для координации функционирования параллельных и асинхронно работающих устройств;
- - как реакция на особые состояния процессора.

Механизм обработки прерывания предполагает выполнение в общем случае следующих этапов:

1. Установку факта прерывания и идентификацию прерывания.

2. Сохранение контекста прерванного процесса. В частности, сохраняются в стеке регистры флагов, значения счетчика команд (регистры CS:IP), осуществляется запрет на обработку последующих прерываний;

3. Передачу управления подпрограмме обработки прерывания. Этапы 1-3 выполняются аппаратно;

4. Сохранение информации о прерванном процессе. В частности, для корректной работы прерванной программы часто в той или иной степени сохраняют регистры общего назначения процессора;

5. Непосредственная обработка прерывания;

6. Восстановление информации прерванного процесса. Этапы 4-6 выполняются программно и представляют собой непосредственно тело обработчика прерывания;

7. Возврат в прерванную программу (извлечение из стека данных, сохраненных на этапе 2. Этот этап выполняется аппаратно.

Поступление прерывания не мешает завершению начатой команды программы, при которой прерывание поступило.

Классы прерываний

По источнику, вызвавшему прерывание, прерывания делятся на три класса:

1. **Внешние прерывания** (аппаратные, асинхронные). Источник таких прерываний находится вне прерываемого процесса. Ими являются системный таймер, устройства ввода-вывода, аппаратура контроля нарушения питания, пульт оператора, сетевые карты и т.д. Обработчиками таких прерываний являются драйверы внешних устройств;
2. **Внутренние прерывания** (исключения). Они наступают синхронно с работой процессора (выполнением программы) при появлении аварийной ситуации. Такими ситуациями могут быть нарушения защиты памяти, попытка деления на ноль, переполнение разрядной сетки, нарушение четности, попытки нарушить режим привилегий и т.д. Обработчиками внутренних прерываний являются специальные модули ядра операционной системы;
3. **Программные прерывания.** Это выполнение особой команды процессора (**INT**), имитирующей прерывание. "Истинным" прерыванием оно как бы не является, но механизм обработки тот же. Команда размещается программистом в кодах программы и поэтому появление программного прерывания предсказуемо. Обработчиками программных прерываний являются так называемые процедуры обслуживания системных вызовов, размещаемые в ОС. Как правило, координирует их работу диспетчер прерываний.

Применение программных прерываний позволяет создать компактный код программы по сравнению со стандартными процедурами. Подпрограмм в ОС немного (не более 256) и адрес в команде занимает всего 1 байт. Его называют индексом или номером прерывания. Кроме того, появляется воз-

можность смены пользовательского режима на привилегированный в момент выполнения команды INT.

Все прерывания в направлении понижения **приоритета** упорядочены:
- средства контроля процессора (внутренние),
- системный таймер,
- внешние устройства (диски, сети, терминалы),
- программные прерывания.

Диспетчер прерываний – координирует работу отдельных обработчиков, но вход имеет для всех обработчиков один. Выбор вида обработчика осуществляется внутри самого обработчика по номеру заданной функции.

Технология использования программных прерываний

Система прерываний организована на базе **таблицы векторов прерываний**. Это позволяет увеличить быстродействие и обеспечить гибкость настройки системы.

Таблица векторов прерываний (рис.1.1) размещена в самом начале памяти (адреса от 0000H до 03FFH) и занимает 256 четырехбайтовых областей (256 x 4 байта = 1024 байта).

Адрес ОЗУ	Данные в байте	Номер вектора	Назначение данных в байтах
0000h	Мл. IP0	0	4 байта хранят адрес (вектор) обработчика №0, который загружается в CS и IP ЦП при наступлении прерывания №0
0001h	Ст. IP0		
0002h	Мл. CS0		
0003h	Ст. CS0		
0004h	Мл. IP1	1	4 байта хранят адрес (вектор) обработчика №1, который загружается в CS и IP ЦП при наступлении прерывания №1
0005h	Ст. IP1		
0006h	Мл. CS1		
0007h	Ст. CS1		
…….	…….	…….	
03FFh	Ст. CS255	255	Последний байт таблицы
0400h			

Рис.1.1. Таблица векторов прерываний

Каждая область хранит два байта сегментного адреса (это старшие байты области, они используется сегментным регистром программ CS) и два байта смещения (это младшие адреса области, они используются указателем

команд IP в качестве смещения в сегменте). При выполнении программы пара регистров CS и IP указывают адрес очередной выполняемой команды (образуют счетчик команд). Адрес, сохраняемый в каждой четырехбайтовой области таблицы векторов, есть адрес начала соответствующего обработчика прерывания. По этой причине адрес, хранимый в конкретной области, и называют **вектором** прерывания, т.к. он указывает на начало обработчика.

Векторы, как и соответствующие им прерывания, имеют **номера**. Вектор с номером 0 располагается, начиная с адреса 0, вектор с номером 1 – с адреса 4, с номером 2 – с адреса 8 и т.д. Получив сигнал на выполнение процедуры обработки прерывания с номером N, процессор заканчивает выполнение очередной (уже начатой) команды, сохраняет в стеке в соответствии с механизмом обработки прерывания слово флагов FLAGS (16 разрядов), сегментный и относительный адрес очередной команды (регистры CS и IP) и загружает CS и IP адреса начала обработчика, извлекая его из соответствующего вектора прерывания, по адресу, начиная с (N*4) до (N*4+3). Тем самым осуществляется переход на программу обработки прерывания с номером N.

Программа обработки прерывания обычно заканчивается командой возврата из прерывания IRET, выполняющей обратные действия – загрузку IP, CS и регистра флагов из стека, что приводит к возврату в основную программу в точку, где она была прервана (по адресу команды, которая является следующей за последней выполненной командой прерванного процесса).

Большая часть векторов прерывания зарезервирована для выполнения конкретных функций:

00H – деление на ноль;
01H - пошаговое выполнение;
02H – немаскируемое прерывание;
03H – команда INT без числового параметра;
04H – прерывание по переполнению;
05H – прерывание по нажатию клавиши PrintScreen;
08H – таймер;
09H – клавиатура;
0AH – зарезервировано для подключения нестандартного устройства;
0BH – последовательный порт COM2;
0CH – последовательный порт COM2;
0DH – жесткий диск (PC, XT), параллельный порт LPT2 (AT);
0FH - параллельный порт LPT1;
10H – видеоадаптер BIOS;
13H – драйвер BIOS диска;
14H - драйвер последовательного порта;
16H – драйвер BIOS клавиатуры;
17H – драйвер BIOS принтера;
19H – начальный загрузчик BIOS;
1AH – календарь-часы BIOS;
1BH – обработчик прерываний по Ctrl/Break;

1CH – программа-заглушка BIOS для обработки прерываний от системного таймера (18,2 прерывания/с);

1DH – адрес таблицы видеопараметров, BIOS;

1EH – адрес таблицы параметров дискеты, BIOS;

1FH – адрес второй половины таблицы шрифтов графических режимов 4…6, BIOS;

21H – диспетчер функций DOS;

22H – адрес перехода при завершении процесса, DOS;

23H – обработчик прерываний по Ctrl/C;

24H – обработчик прерываний по критической ошибке;

25H – абсолютное чтение диска;

26H – абсолютная запись на диск;

28H – программа-заглушка DOS для активизации резидентных программ командами с клавиатуры;

2FH - мультиплексное прерывание DOS;

33H – драйвер мыши;

43H – адрес таблицы шрифтов графических режимов, BIOS;

60H – 66H – прерывания пользователя;

67H – драйвер дополнительной памяти LIM EMS;

68H – 6FH – свободные векторы;

70H – КМОП-часы реального времени;

71H – программа BIOS, возбуждающая прерывания INT 0AH для совместимости машин XT и AT в части обслуживания нестандартных внешних устройств;

72H – 73H зарезервированы;

74H – мышь PS/2;

76H – жесткий диск AT, PS/2;

77H – резерв;

78H – 7FH – свободные векторы;

F1H – FFH – не используются.

Все векторы разделены на **группы**:

- векторы внутренних прерываний (00H – 07H);
- векторы аппаратных прерываний (08H – 0FH и 70H – 77H);
- драйверы BIOS (10H, 13H, 16H и т.д.);
- программы DOS (21H, 22H, 23H и т.д.);
- адреса системных таблиц BIOS (1DH, 1EH, 43H и т.д.).

Системные программы, адреса которых хранятся в векторах прерываний, в большинстве своем являются всего лишь **диспетчерами**, открывающими доступ к большим группам программ, реализующим системные функции. Так, например, видеодрайвер BIOS (вектор 10H) включает программы:

- установка видеорежима (функция 00H);
- установки позиции курсора (функция 02H);
- получения позиции и размера курсора (функция 03H);
- чтения символа и атрибута в позиции курсора (функция 08H);
- задания цветовой палитры (функция 10H подфункция 02H);

- загрузки шрифтов пользователя (функция 11H подфункция 10H) и т.д.

Через вектор 21H вызываются функции DOS:

- ввод символа с клавиатуры с эхом (функция 01H);
- вывод символа на экран (функция 02H);
- установка вектора прерывания (функция 25H);
- установка системной даты (функция 2BH);
- создание каталога (функция 39H);
- создание файла (функция 3CH);
- чтение из файла (функция 3FH) и т.д.

Программирование программных прерываний

Для обращения к системным функциям необходимо в регистре AH задать номер функции, а в других регистрах указать необходимые данные (параметры) для конкретной системной программы. Сформировать команду программного прерывания с указанием его номера. Результаты работы программы также возвращаются в соответствующих регистрах процессора.

Для вызова системных функций удобно пользоваться языком ассемблера. Паскаль допускает использование встроенных ассемблерных кодов. Для того чтобы поместить в программу на языке Паскаль ассемблерный код, достаточно заключить его в следующие операторные скобки:

asm
<ассемблерный код>
end;

Вызов системных функций с помощью встроенного ассемблера имеет дополнительное преимущество, поскольку при этом автоматически сохраняются значения всех регистров, которые восстанавливаются после выхода из ассемблерного блока. В противном случае, это пришлось бы делать вручную.

Для вызова системных функций, в основном, используются два оператора ассемблера.

1. **mov** dest, source - выполняет пересылку значения source в dest.

Например:

mov ax, bx - переслать содержимое регистра bx в регистр ax;

mov ax, 3 - переслать значение 3 в регистр ax;

mov ax, x - переслать значение переменной x в регистр ax. Естественно, переменная X должна быть объявлена средствами Паскаля, причем по своему типу должна занимать 2 байта (т.к. AX – адрес двухбайтного регистра);

mov x, ax - переслать содержимое регистра ax в переменную x;

2. **int** <номер вектора прерываний> - вызывает прерывание с указанным номером. Например, int 21h.

Большинство системных функций возвращают в флаге переноса CF код завершения. Если функция выполнилась успешно, CF=0. В случае любой

ошибки CF=1, а в одном из регистров (чаще всего в AX) возвращается еще и код ошибки. Предполагается, что при корректном программировании будет осуществлен анализ этого кода. Поэтому типовой вариант обращения к процедуре имеет структуру:

```
        Mov     AH,funk     ;funk – номер функции
        Mov     CX,0        ;Задание параметров
        ….
        INT     21H         ;Переход к функции DOS
        Jc      error       ;Переход к анализу возможных ошибок
        ….
error:  Cmp         AX,1
        Je      error1
        Cmp     AX,2
        Je      error2
        …
```

Пример справочной информации для работы с системными вызовами выглядит следующим образом:

INT 21h, функция 39h. Создание каталога.

Создает каталог в конце указанного пути.

При вызове: AH=39h ; задание номера функции

DS:DX= адрес пути в виде строки ASCIIZ

При ошибке: CF=1

AX= код ошибки.

Вопрос 1.1. Исследование приемов использования прерываний, передачи параметров системным функциям и способов получения результатов

Пример №1. Изменить цвет края экрана дисплея

Пусть необходимо изменить цвет края экрана дисплея, например, для того, чтобы информировать пользователя о переходе в другой режим (например, смене алфавита в среде Pascal). Это области, называемые выбегом развертки, не используются для представления данных в пределах экрана. Для изменения цвета можно воспользоваться обслуживанием BIOS, вызываемым прерыванием 10h. Номер функции, выполняющей эти действия - 1001h (это номер функции и подфункции одновременно). Параметром этой функции является цвет, который необходимо установить. Этот параметр должен быть передан через регистр bh процессора. Цвет задается в формате кзсКЗС (красный-зеленый-синий): маленькими буквами обозначается 33% интенсивности цвета, большими - 66%. Такое управление возможно в буквенно-цифровом (символьном) режиме работы дисплея. Такой режим предполагает, что символ формируется знакоместом размером, например, 8X8 точек, которые создают и фон и силуэт символа. Такая матрица точек размещается по

месту установки курсора и совместно с соседними матрицами образует выводимое изображение.

Например:

Значение разряда	0	0	0	0	1	1	0	1
Аддитивные цвета	-	-	к	з	с	К	З	С

00001101 = 0Dh задает фиолетовый цвет.

Программа, выполняющая установление фиолетового края экрана, будет выглядеть следующим образом.

```
Program int1;
begin
asm
      mov ax, 1001h
      mov bh,0dh
      int 10h
end;
end.
```

Задание. Выполните установку фиолетового, красного, желтого, черного цветов. Справочные данные, программу и таблицу формирования кода цвета занесите в конспект.

Пример №2. Установка позиции курсора

Задает положение курсора в текстовых координатах (0-24 для строк, 0-79 для столбцов) на указанной странице (установка активной видеостраницы из двух возможных осуществляется функцией AH=05: номер страницы - в AL). Изображение на странице в неактивном режиме не пропадает, потому что для каждой страницы имеются отдельные блоки памяти.

Справочная информация. Прерывание 10h, функция 02h, параметры: номер страницы (0 или 1) – в регистре bh, номер строки – в регистре dh, номер столбца – в регистре dl.

Соответствующий программный код будет выглядеть следующим образом:

```
Program int2;
Begin
asm
      mov ah,02h
      mov bh,0
      mov dh,12
      mov dl,40
      int 10h
end;
readln
end.
```

Задание.

1. Определить количество страниц видеоадаптера.
2. Установить курсор на невидимый экран и там же написать слово (средствами Pascal). Затем переключить видеостраницы.
3. Запишите текст задания и справочные данные.

Пример №3. Изменить палитру цветов

Адаптер поддерживает 64 цвета, из которых на экране можно отобразить только 16. Этот набор называется палитрой цветов. Палитра цветов представляет собой массив байтов с номерами от 0 до 16. Каждый байт представляет собой цвет в формате кзсКЗС. Шестнадцатый элемент – цвет границы экрана. Обращение к каждому цвету палитры производится путем указания его номера. Стандартная палитра соответствует следующему набору цветов: 1 – синий; 2 – зеленый; 3 – голубой; 4 – красный и т.д. Полностью изменить палитру можно функцией 1002h прерывания 10h. В качестве параметров ей необходимо передать адрес массива с новой палитрой в регистрах ES:DX.

Логический адрес любой ячейки или группы ячеек состоит из двух частей: сегмента и смещения, то есть адрес представляется, например, следующим образом: 3fd9:ff01. Для того, чтобы извлечь из адреса сегмент, нужно воспользоваться директивой **seg**, чтобы извлечь смещение – директивой **offset**.

Регистр ES является сегментным регистром, куда значение можно переслать только из другого регистра. Поэтому пересылку сегмента осуществляем через посредника – регистр AX. Знак "$" обозначает шестнадцатеричную систему представления числа.

```
program int3;
uses graph;
const newpalette:array[0..16]of byte=
($3c,$24,$54,$55,$3d,0,0,0,0,0,0,0,0,0,0,0,$55);
var gd,gm,i:integer;
begin
{Переключимся в графический режим}
gd:=detect;
initgraph(gd,gm,'');
{Нарисуем четыре разноцветных прямоугольника в стандартной палитре цветов}
for i:=1 to 4 do
  begin
   setfillstyle(1,i); {Установка типа закраски 1 и ее цвет i}
   bar(1,i*50,200,50+i*50);  {Рисование прямоугольника в стиле "левый верхний, правый нижний"}
  end;
 readln; {сделаем задержку до нажатия клавиши ENTER}
```

```
{Изменим палитру}
asm
  mov ax,seg newpalette
  mov es,ax
  mov dx,offset newpalette
  mov ax,1002h
  int 10h
end;
readln; {сделаем задержку до нажатия клавиши ENTER}
closegraph;
end.
```

Задание.

1. Определите как происходит "перерисовывание" прямоугольников.
2. Скройте третий сверху прямоугольник (на его месте должен быть пропуск).
3. Занесите программу в конспект с необходимыми пояснениями.
4. Выделите команды, позволяющие указывать для системных функций области оперативной памяти в качестве параметров.
5. Запишите справочные данные.
6. В чем отличие номера цвета от кода цвета?
7. Почему в 17 цветов при возможности отображения только 16?
8. Как используется 17-й цвет, какой у него номер и какие программные системные средства его используют?
9. Для чего используется цвет с номером 0?

Пример №4. Загрузка шрифта пользователя

Каждый символ на экране отображается своей комбинацией точек. Эти данные, объединенные в кодовые страницы, обозначаются условными номерами, например, США – 437, РФ – 866. Стандартные страницы представляют собой наборы из трех таблиц для каждого символа, точечные изображения которых можно представить матрицами 8x8, 8x14, 8x16, состоящими из нулей и единиц, причем единице соответствует наличие изображения в клетке, а ноль – его отсутствию. Такие совокупности матриц обычно называются знакогенераторами. Изображение формируется тем цветом, который в настоящее время установлен активным. Создадим, например, символ 8x8.

1	2	3	4	5	6	7	8	Hex - представление
0	**0**	**0**	**1**	**1**	**1**	**0**	**0**	**1C**
0	**0**	**1**	**0**	**0**	**1**	**0**	**0**	**24**
0	**1**	**1**	**1**	**0**	**1**	**0**	**0**	**74**
0	**1**	**0**	**1**	**0**	**1**	**0**	**0**	**54**
0	**1**	**1**	**0**	**1**	**1**	**0**	**0**	**6C**
0	**0**	**1**	**1**	**0**	**1**	**0**	**0**	**34**

0	1	0	0	0	1	0	0	44
0	1	1	1	1	0	1	1	7B

Сначала воспользуемся функцией 1112 прерывания 10h для установления шрифта 8x8, передадим ей номер страницы в регистре bl. Затем воспользуемся функцией 1110h прерывания 10h для замены одного символа. Параметры функции: в регистрах ES:BP – адрес таблицы символов; в BH – высота символа; в BL – номер блока знакогенератора; в CX – количество заменяемых символов в таблице; в DX – код символа, начиная с которого необходимо произвести замену.

Код 49 (31h) принадлежит символу «1» в стандартной таблице кодировки. Выведем на экран строку, содержащую этот символ, и убедимся, что замена произошла.

```
program int4;
uses crt;
const
  newchar:array[1..8]of byte=($1c,$24,$74,$54,$6c,$34,$44,$7b);
begin
clrscr;
asm
 ; установим шрифт 8x8
 mov ax,1112h
 mov bl,0
 int 10h
; заменим символ
 mov ax, seg newchar
 mov es,ax
 mov bp, offset newchar
 mov ax,1110h
 mov bh,8
 mov bl,0
 mov cx,1
 mov dx,49
 int 10h
end;
gotoxy(40,12);
writeln('222111333');
readln;
end.
```

Задание.

1. Записать справочные данные для функций 1112h и 1110h.
2. Сколько матриц хранится в ПЭВМ?
3. Нарисуйте свой символ вместо символа "1".

4. В чем отличие графического режима монитора от буквенно-цифрового при работе?
5. Подготовьте символ "11", который должен отображаться при нажатии клавиши "7" на клавиатуре.

Пример №5. Получение флагов клавиатуры

Прерывание INT 16h работает с клавиатурой на уровне BIOS, считывая двухбайтные коды из кольцевого буфера ввода (код ASCII + SCAN-код). Функция 02h возвращает байт флагов клавиатуры, описывающий состояние управляющих клавиш клавиатуры (байт в области данных BIOS по адресу 0000h:0417h).

Слово флагов клавиатуры	40h:17h	Ins	Caps Lock	Num Lock	Scroll Lock	Alt	Ctrl	Shift лев	Shift прав
		7	6	5	4	3	2	1	0

Функция используется программами, работающими на уровне SCAN-кодов.

При вызове: AH=02h

При возврате: AL= флаги. Биты байта имеют следующие значения:

0 – нажата правая клавиша Shift
1 - нажата левая клавиша Shift
2 – нажата клавиша Ctrl
3 - нажата клавиша Alt
4 – Включен режим Scroll Lock
5 - Включен режим Num Lock
6 - Включен режим Caps Lock
7 - Включен режим Insert

```
Program int5;
Var keys: byte;
Asm
  Mov ah, 02h
  Int 16h
  Mov keys, al
End;
Writeln ( ‘keys=’, keys);
Readln;
End.
```

Задание. Переведите полученное десятичное число в двоичный код и определите состояние флагов.

Вопрос 1.2. Разработка и исследование программ, использующих прерывания на основе справочной информации о функциях обслуживания

Используя следующую справочную информацию, воспользуйтесь прерываниями для вызова соответствующих функций. Убедитесь, что результат достигнут.

INT 21h, функция 09h . Вывод строки.

Выводит строку символов на устройство стандартного вывода. Строка должна заканчиваться символом $. Допустимо перенаправление вывода. Допустимо использование Esc-последовательностей. Коды ASCII : 07h - звонок, 08h – шаг назад , 0Dh – возврат каретки, 0Ah – перевод строки, рассматриваются как управляющие и выполняются соответствующие им действия. Выполняет обработку <Ctrl>/C при вводе этой комбинации с клавиатуры перед выводом каждого 64-го символа.

При вызове: **AH**=09h

DS:DX=адрес строки

INT 21h, функция 0Eh. Выбор диска.

Назначает текущий диск и возвращает число логических дисководов в системе.

При вызове: **AH**= 0Eh

AL= код дисковода (0=A, 1=B и т.д.)

При возврате: **AL**= число логических дисководов в системе

INT 21h, функция 19h . Получение текущего диска.

Возвращает код текущего диска.

При вызове: **AH**=19h

При возврате: **AL**= код текущего диска (0=A, 1=B и т.д.)

INT 21h, функция 1Bh. Получение информации о текущем диске.

Возвращает характеристики текущего диска.

При вызове: **AH**=1Bh

При возврате: **AL**=количество секторов в кластере

CX=количество байтов в секторе

DX=общее количество кластеров на диске

DS:BX->байт описания носителя:

FFh - дискета 320 Кбайт

FEh - дискета 160 Кбайт

FDh - дискета 360 Кбайт

FCh - дискета 180 Кбайт

F9h - дискета 1,2 Мбайт

F8h - жесткий диск

F0h - другие

INT 21h, функция 2Ah . Получение системной даты.

Позволяет получить значение текущей даты.

При вызове: **AH**=2Ah

При возврате: **CX=**год (от 1980 до 2099)
DH=месяц (от 1 до 12)
DL=день (от 1 до 31)
AL=день недели (0-воскресенье и т.д.)

INT 21h, функция 2Ch. Получение времени.

Позволяет получить значение текущего времени.

При вызове: **AH**=2Ch

При возврате: **CH=**часы (от 0 до 23)
CL=минуты (от 0 до 59)
DH=секунды (от 0 до 59)

INT 10h, функция 06h. Инициализация или прокрутка окна вверх.

Инициализирует окно с указанными координатами пробелами ASCII с заданным атрибутом или прокручивает содержимое окна вверх на заданное число строк. Действует только для активной страницы. При прокрутке появляющиеся внизу строки заполняются пробелами ASCII с заданным атрибутом. Функцию удобно использовать для быстрой очистки всего экрана или любой прямоугольной области на экране.

При вызове: **AH**=06h
AL=число строк прокрутки; если AL=0 , все окно очищается
BH=атрибут символов в окне
CH=Y - координата верхнего левого угла окна
CL=X - координата верхнего левого угла окна
DH=Y - координата нижнего правого угла окна
DL=X - координата нижнего правого угла окна

INT 10h, функция 07h. Инициализация или прокрутка окна вниз.

Инициализирует окно с указанными координатами пробелами ASCII с заданным атрибутом или прокручивает содержимое окна вниз на заданное число строк. Действует только для активной страницы. При прокрутке появляющиеся внизу строки заполняются пробелами ASCII с заданным атрибутом. Функцию удобно использовать для быстрой очистки всего экрана или любой прямоугольной области на экране.

При вызове: **AH**=06h
AL=число строк прокрутки; если AL=0 , все окно очищается
BH=атрибут символов в окне
CH=Y - координата верхнего левого угла окна
CL=X - координата верхнего левого угла окна

DH=Y - координата нижнего правого угла окна
DL=X - координата нижнего правого угла окна

INT 10h, функция 09h. Запись символа и атрибута в позицию курсора.

При вызове: AH = 09h
AL = символ
BH = страница
BL = атрибут (текстовый режим) или цвет (графический)
CX = коэффициент повторения.

Атрибут символа: Назначение битов в атрибуте.

7	4-6	5	0-2
мерцание	Цвет фона	яркость	Цвет символа

INT 10h, функция 10h, подфункция 03h. Переключение бита "мерцание/яркость".

Определяет назначение старшего бита (7) атрибута символа: мерцание символа или повышенная яркость фона.

При вызове: **AX**=1003h
BL=назначение старшего бита атрибута :
0 - повышенная яркость фона
1 - мерцание символа

INT 21h, функция 36h. Получение объема свободного пространства на диске.

При вызове: AH = 36h;
DL = код дисковода (0 – текущий, 1 – A; 2 – B и т.д.
При возврате:
AX = число секторов в кластере
BX = число свободных кластеров
CX = размер сектора в байтах
DX = полное число кластеров на диске
При ошибке - AX = ffffh

Контрольные вопросы (ПК-11):

1. Приведите определение операционной системы.
2. Каким образом осуществляется обращение к функциям DOS и BIOS?
3. Поясните механизм реализации процедуры прерывания, проиллюстрируйте ее.
4. Приведите классификацию прерываний.
5. В чем различие между программными и аппаратными прерываниями?
6. Поясните понятие вектора прерываний.

7. Каким образом осуществляется обращение из прикладной программы к системным функциям?
8. Поясните порядок использования и программирования процедур, использующих программные прерывания.
9. Как вычислить адрес размещения вектора прерывания INT 10h, INT 21h?
10. Поясните технологию вызова системных обработчиков.
11. Каким образом в обработчик передаются параметры?
12. Каким образом обработчик возвращает результаты прерывания?