МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНА МЕТАЛУРГІЙНА АКАДЕМІЯ УКРАЇНИ

О.І. Михальов, Ю.О. Каліберда

М Е Т О Д И Ч Н І В К А З І В К И

до виконання лабораторних робіт з дисципліни

«Методи та засоби комп'ютерних інформаційних технологій»

для студентів напряму – «Комп’ютерні науки»

Дніпропетровськ - 2019

Задание на лабараторную работу

Изучить теоретический материал по предмету.

Освоить практические навыки решения задач по данному предмету.

Порядок выполнения работы

1. Изучить теоретический материал по предмету.
2. Освоить практические навыки решения задач по данному предмету.
3. Решить задачи по предмету.
4. Вариант по списку журнала старосты группы.
5. Выполнить индивидуальное задание.

Лабораторная работа №1

Вывести на консоль ASCII, SCAN код нажатых клавиш на клавиатуре, используя прерывания.

Примечание:

```
asm
    {
        mov ah,0
        int 0x16
        mov sc_code, ah
        mov ascii_code, al
        mov ah,0x12
        int 0x16
        mov sd_code,ah
        mov sd2_code,al
        mov aa,ax
    }
```

Краткие теоретические сведения:

Клавиатура выполнена, как правило, в виде отдельного устройства, подключаемого к компьютеру тонким кабелем. Если рассмотреть сильно упрощенную принципиальную схему клавиатуры, представленную на рисунке, можно заметить, что все клавиши находятся в узлах

матрицы:

Рис.1. Упрощенная схема клавиатуры

Все горизонтальные линии матрицы подключены через резисторы к источнику питания +5 В. Клавиатурный компьютер имеет два порта - выходной и входной. Входной порт подключен к горизонтальным линиям матрицы (X0-X4), а выходной - к вертикальным (Y0-Y5). Устанавливая по очереди на каждой из вертикальных линий уровень напряжения, соответствующий логическому 0, клавиатурный компьютер опрашивает состояние горизонтальных линий. Если ни одна клавиша не нажата, уровень напряжения на всех горизонтальных линиях соответствует логической 1 (т.к. все эти линии подключены к источнику питания +5 В через резисторы). Если оператор нажмет на какую-либо клавишу, то соответствующая вертикальная и горизонтальная линии окажутся замкнутыми. Когда на этой вертикальной линии процессор установит значение логического 0, то уровень напряжения на горизонтальной линии также будет соответствовать логическому 0. Как только на одной из горизонтальных линий появится уровень логического 0, клавиатурный процессор фиксирует нажатие на клавишу. Он посылает в центральный компьютер запрос на прерывание и номер клавиши в матрице. Аналогичные действия выполняются и тогда, когда оператор отпускает нажатую ранее клавишу.

Номер клавиши, посылаемый клавиатурным процессором, однозначно связан с распайкой клавиатурной матрицы и не зависит напрямую от обозначений, нанесенных на поверхность клавиш. Этот номер называется скан-кодом (Scan Code).

Слово scan ("сканирование"), подчеркивает тот факт, что клавиатурный компьютер сканирует клавиатуру для поиска нажатой клавиши.

Но программе нужен не порядковый номер нажатой клавиши, а соответствующий обозначению на этой клавише ASCII-код. Этот код не зависит однозначно от скан-кода, т.к. одной и той же клавише могут соответствовать несколько значений ASCII-кода. Это зависит от состояния других клавиш. Например, клавиша с обозначением '1' используется еще и для ввода символа '!' (если она нажата вместе с клавишей SHIFT).

Поэтому все преобразования скан-кода в ASCII-код выполняются программным

обеспечением. Как правило, эти преобразования выполняют модули BIOS. Для использования символов кириллицы эти модули расширяются клавиатурными драйверами.

Если нажать на клавишу и не отпускать ее, клавиатура перейдет в режим автоповтора. В этом режиме в центральный компьютер автоматически через некоторый период времени, называемый периодом автоповтора, посылается код нажатой клавиши. Режим автоповтора облегчает ввод с клавиатуры большого количества одинаковых символов.

Следует отметить, что клавиатура содержит внутренний 16-байтовый буфер, через который она осуществляет обмен данными с компьютером.

Лабораторная работа №2

По нажатию на комбинацию клавиш заблокировать работу клавиши. При повторном нажатии, восстановить работу клавиши.

Примечание:

```
void interrupt (* oldHand)();
void interrupt myHand();
oldHand = getvect(0x9);
setvect(0x9, myHand);
```

Краткие теоретические сведения:

Этот обработчик выполняет следующие действия:

- читает из порта 60h скан-код нажатой клавиши;
- записывает вычисленное по скан-коду значение ASCII-кода нажатой клавиши в специальный буфер клавиатуры, расположенный в области данных BIOS;
- устанавливает в 1 бит 7 порта 61h, разрешая дальнейшую работу клавиатуры;
- возвращает этот бит в исходное состояние;
- записывает в порт 20h значение 20h для правильного завершения обработки аппаратного прерывания.

Обработчик прерывания INT09h не просто записывает значение ASCII-кода в буфер клавиатуры. Дополнительно отслеживаются нажатия таких комбинаций клавиш, как

Ctrl-Alt-Del, обрабатываются специальные клавиши PrtSc и SysReq. При вычислении кода ASCII нажатой клавиши учитывается состояние клавиш Shift и CapsLock.

Буфер клавиатуры имеет длину 32 байта и расположен по адресу 0000h:041Eh для машин IBM PC/XT.

В IBM AT и PS/2 расположение клавиатурного буфера задается содержимым двух слов памяти с адресами 0000h:0480h (компонента смещения адреса начала буфера) и 0000h:0482h (смещение конца буфера). Обычно в IBM AT эти ячейки памяти содержат значения, соответственно, 001Eh и 003Eh. Так как смещения заданы относительно сегментного адреса 0040h, то видно, что обычное расположение клавиатурного буфера в IBM AT и PS/2 соответствует его расположению в IBM PC/XT.

Клавиатурный буфер организован циклически. Это означает, что при его переполнении самые старые значения будут потеряны. Две ячейки памяти, находящиеся в области данных BIOS с адресами 0000h:041Ah и 0000h:041Ch содержат, соответственно, указатели на начало и конец буфера. Если значения этих указателей равны друг другу, буфер пуст. (Можно удалить все символы из буфера клавиатуры, установив оба указателя на начало буфера. Однако есть более предпочтительный способ с использованием прерывания BIOS INT 16h).

Указателями на начало и конец клавиатурного буфера обычно управляют обработчики прерываний INT09h и INT16h.

Программа извлекает из буфера коды нажатых клавиш, используя различные функции прерывания INT16h.

Таблица варинатов:

№	Сочетание клавиш	Заблокировать клавишу
1	Ctrl+Alt+a	F1
2	Shift+Alt+b	F2
3	Ctrl+Alt+c	F3
4	Shift+Alt+d	F4

5	Ctrl+Alt+e	F5
6	Shift+Alt+f	F6
7	Ctrl+Alt+g	F7
8	Shift+Alt+h	F8
9	Ctrl+Alt+j	F9
10	Shift+Alt+k	F10
11	Ctrl+Alt+l	F11
12	Shift+Alt+m	F12
13	Ctrl+Alt+n	a
14	Shift+Alt+o	s
15	Ctrl+Alt+p	d
16	Shift+Alt+r	f
17	Ctrl+Alt+s	g
18	Shift+Alt+t	h
19	Ctrl+Alt+y	j
20	Shift+Alt+q	k
21	Ctrl+Alt+r	l
22	Shift+Alt+v	x
23	Ctrl+Alt+w	c
24	Shift+Alt+x	v

Лабораторная работа №3

Используя прерывание мыши, выполнит задание для своего варианта.

Примечание:

Определить положение курсора

На входе: AX = 0003h.

На выходе: BX = состояние клавиш мыши:

бит 0=1 - нажата левая клавиша;
бит 1=1 - нажата правая клавиша;
бит 2=1 - нажата средняя клавиша

(для мыши системы Mouse Systems);
CX = координата X (по горизонтали);
DX = координата Y (по вертикали).

Краткие теоретические сведения:

Драйвер мыши, независимо от того, реализован он через устанавливаемый драйвер или резидентную программу, определяет обработчик прерывания INT 33h. Этот обработчик выполняет все операции, связанные с обслуживанием мыши:

- сброс мыши и установка драйвера в исходное состояние;
- включение/выключение курсора мыши;
- установка курсора в определенное место экрана;
- определение текущих координат курсора и текущего состояния клавиш;
- определение координат курсора и состояния клавиш в момент нажатия на клавишу и в момент отпускания клавиши;
- определение области на экране, в пределах которой может перемещаться курсор;
- определение области на экране, в пределах которой курсор не будет виден;
- определение формы графического и текстового курсоров;
- определение величины перемещения мыши в сотых долях дюйма;
- подключение к драйверу пользовательской процедуры, получающей управление при нажатии на заданную клавишу или при перемещении мыши;
- запоминание и восстановление состояния драйвера;
- управление эмуляцией светового пера;
- управление скоростью движения курсора;
- задание/определение используемой видеостраницы;
- управление драйвером мыши.

Таблица варинатов:

№	Условие	Действие
1	Положение оси X кратное 3	Вывести в левом верхнем углу координаты мыши

2	Положение оси X кратное 5	Вывести в левом верхнем углу координаты мыши
3	Положение оси X кратное 10	Вывести в левом верхнем углу координаты мыши
4	Положение оси У кратное 3	Вывести в левом верхнем углу координаты мыши
5	Положение оси У кратное 5	Вывести в левом верхнем углу координаты мыши
6	Положение оси У кратное 10	Вывести в левом верхнем углу координаты мыши
7	Положение оси X и У кратное 3	Вывести в левом верхнем углу координаты мыши
8	Положение оси X и У кратное 5	Вывести в левом верхнем углу координаты мыши
9	Положение оси X и У кратное 10	Вывести в левом верхнем углу координаты мыши
10	Нажата ЛКМ	Воспроизвести звуковой сигнал
11	Нажата ПКМ	Воспроизвести звуковой сигнал
12	Нажата ЛКМ и ПКМ	Воспроизвести звуковой сигнал

Лабораторная работа №4

Установка видео режима

Примечание:

Функция 00h прерывания 10h позволяет задать любой режим работы видеоадаптера:

```
пример выбора режима видеоадаптера:
mov ah,0 ;функция выбора режима работы видеоадаптера
mov al,mode ;выбираем режим mode
int 10h ;
```

Краткие теоретические сведения:

Существуют несколько стандартных режимов работы видеоадаптеров, определенных фирмой IBM. Список стандартных режимов работы видеоадаптеров представлен в таблице.. Стандартные раежимы работы не включают все режимы, в которых могут работать видеоадаптеры. Многие фирмы - производители видеоадаптеров выпускают адаптеры, поддерживающие нестандартные режимы, имеющие улучшенные характеристики.

Характеристики нестандартных режимов отличаются для видеоадаптеров разных фирм. В приложении приведены параметры нестандартных режимов для наиболее распространенных видеоадаптеров.

Режимы работы видеоадаптеров характеризуются типом информации, которую они отображают (текстовая или графическая), количеством используемых цветов, разрешающей способностью и размерами символов.

Режим работы	Тип информации	Число цветов	Разрешение	Поддерживаемые дисплеи
0Eh	графический цветной	16	640 X 200	CD, ECD, VGA, многочастотный
0, 1	текстовый цветной	16 (8x8)	40 X 25	CD, ECD, VGA, многочастотный
0*, 1*	текстовый цветной	16 (8*14)	40 X 25	ECD, VGA, многочастотный
0+, 1+	текстовый цветной	16 (9*16)	40 X 25	VGA, многочастотный
2, 3	текстовый цветной	16 (8*8)	80 X 25	CD, ECD, VGA, многочастотный
2*, 3*	текстовый цветной	16 (8*14)	80 X 25	ECD, VGA, многочастотный
2+, 3+	текстовый цветной	16 (9*16)	80 X 25	VGA, многочастотный
4, 5	графический	4	320 X 200	CD, ECD, VGA,

цветной многочастотный

6 графический 2 640 X 200 CD, ECD, VGA,
цветной многочастотный

7 текстовый 2 80 X 25 монохромный,
монохромный (9*14) VGA

7+ текстовый 2 80 X 25 VGA
монохромный (9*16)

8,9, используются только видеоадаптерами компьютера PC jr
0Ah

0Bh,0Ch зарезервировано

0Dh графический 16 320 X 200 CD, ECD, VGA,
цветной многочастотный

0Eh графический 16 640 X 200 CD, ECD, VGA,
цветной многочастотный

0Fh графический 2 640 X 350 монохромный,
монохромный VGA

10h графический 16 640 X 350 ECD, VGA,
цветной многочастотный

11h графический 2 640 X 480 VGA,
цветной многочастотный

12h графический 16 640 X 480 VGA,
цветной многочастотный

13h графический 256 320 X 200 VGA,
цветной многочастотный

Лабораторная работа №5

Установка видео режима VESA

Выполните свой вариант.

Примечание:

Для рисования линий и окружностей, используйте Алгоритм Брезенхэма

```
void drawLine(int x1, int y1, int x2, int y2) {
const int deltaX = abs(x2 - x1);
const int deltaY = abs(y2 - y1);
const int signX = x1 < x2 ? 1 : -1;
const int signY = y1 < y2 ? 1 : -1;
//
```

```
    int error = deltaX - deltaY;
    //
    setPixel(x2, y2);
    while(x1 != x2 || y1 != y2) {
      setPixel(x1, y1);
      const int error2 = error * 2;
      //
      if(error2 > -deltaY) {
        error -= deltaY;
        x1 += signX;
      }
      if(error2 < deltaX) {
        error += deltaX;
        y1 += signY;
      }
    }

void drawCircle(int x0, int y0, int radius) {
      int x = 0;
      int y = radius;
      int delta = 1 - 2 * radius;
      int error = 0;
      while(y >= 0) {
            setPixel(x0 + x, y0 + y);
            setPixel(x0 + x, y0 - y);
            setPixel(x0 - x, y0 + y);
            setPixel(x0 - x, y0 - y);
            error = 2 * (delta + y) - 1;
            if(delta < 0 && error <= 0) {
                  ++x;
                  delta += 2 * x + 1;
                  continue;
            }
            error = 2 * (delta - x) - 1;
            if(delta > 0 && error > 0) {
                  --y;
                  delta += 1 - 2 * y;
                  continue;
            }
            ++x;
            delta += 2 * (x - y);
            --y;
      }
}
```

Таблица варинатов:

№	Нарисовать
1	Произвольный треугольник
2	Прямоугольник
3	Равнобедренный треугольник
4	Равносторонний треугольник
5	Квадрат
6	Ромб
7	Овал
8	Окружность
9	Трапецию
10	Линию
11	Закрашенный Произвольный треугольник
12	Закрашенный Прямоугольник
13	Закрашенный Квадрат
14	Закрашенный Ромб
15	Закрашенный Овал
16	Закрашенная Трапеция