МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по лабораторной работе №5
по дисциплине «Организация ЭВМ и систем»
ТЕМА: Разработка собственного прерывания.

Студент гр. 9382 _____ Бочаров Г.С.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург
2020

Цель работы.

Изучить команды для работы с прерываниями в ассемблере, написать собственное прерывание.

Теоретические сведения:

Прерывание - это процесс вызова процедур для выполнения некоторой задачи, обычно связанной с обслуживанием некоторых устройств (обработка сигнала таймера, нажатия клавиши и т.д.). Когда возникает прерывание, процессор прекращает выполнение текущей программы (если ее приоритет ниже) и запоминает в стеке вместе с регистром флагов адрес возврата(CS:IP) - места, с которого будет продолжена прерванная программа.

Затем в CS:IP загружается адрес программы обработки прерывания и ей передается управление. Адреса 256 программ обработки прерываний, так называемые векторы прерывания, имеют длину по 4 байта (в первых двух хранится значение IP , во вторых - CS) и хранятся в младших 1024 байтах памяти. Программа обработки прерывания должна заканчиваться инструкцией IRET (возврат из прерывания), по которой из стека восстанавливается адрес возврата и регистр флагов.

Программа обработки прерывания - это отдельная процедура, имеющая структуру:

```
SUBR_INT PROC FAR

 PUSH AX ; сохранение изменяемых регистров

 <действия по обработке прерывания>
```

```
POP AX ; восстановление регистров

...

MOV AL, 20H

OUT 20H,AL

IRET

SUBR_INT ENDP
```

Две последние строки обработчика прерывания, указанные перед командой IRET выхода из прерывания, необходимы для разрешения обработки прерываний с более низкими уровнями, чем только что обработанное.

Замечание: в лабораторной работе действиями по обработке прерывания может быть вывод на экран некоторого текста, вставка цикла задержки в вывод сообщения или включение звукового сигнала.

Программа, использующая новые программы обработки прерываний при своем завершении должна восстанавливать оригинальные векторы прерываний. Функция 35 прерывания 21H возвращает текущее значение вектора прерывания, помещая значение сегмента в ES, а смещение в BX. В соответствии с этим, программа должна содержать следующие инструкции:

```
; -- в сегменте данных
```

```
KEEP_CS DW 0 ; для хранения сегмента

KEEP_IP DW 0 ; и смещения вектора прерывания

; -- в начале программы

MOV AH, 35H ; функция получения вектора

MOV AL, 1CH ; номер вектора

INT 21H

MOV KEEP_IP, BX ; запоминание смещения

MOV KEEP_CS, ES ; и сегмента вектора прерывания
```

Для установки адреса нового обработчика прерывания в поле векторов прерываний используется функция 25H прерывания 21H, которая помещает заданные адреса сегмента и смещения обработчика в вектор прерывания с заданным номером.

```
PUSH DS

MOV DX, OFFSET ROUT ; смещение для процедуры в DX

MOV AX, SEG ROUT ; сегмент процедуры
```

```
MOV DS, AX ; помещаем в DS

MOV AH, 25H ; функция установки вектора

MOV AL, 60H ; номер вектора

INT 21H ; меняем прерывание

POP DS
```

Далее может выполняться вызов нового обработчика прерывания. В конце программы восстанавливается старый вектор прерывания

```
CLI

PUSH DS

MOV DX, KEEP_IP

MOV AX, KEEP_CS

MOV DS, AX

MOV AH, 25H

MOV AL, 1CH

INT 21H ; восстанавливаем старый вектор прерывания
```

POP DS

STI

Задание:

1B

1 - 1Ch - прерывание от часов - генерируется автоматически операционной системой 18 раз в сек;

B - Выдача звукового сигнала;

Ход работы:

При разработке программы были использованы следующие команды:

Инструкция OUT выводит данные из регистра AL или AX (ИСТОЧНИК) в порт ввода-вывода. Номер порта должен быть указан в ПРИЁМНИКЕ.

Выводы.

В результате выполнения лабораторной работы был разработан код, определяющий собственное прерывание. Освоена работа с динамиком и таймером.

Приложение А.
Исходный код программы

Текст файла 2.ASM

```
stack segment stack
     dw 6 dup(?)
stack ends
data segment
     keep_seg dw 0
     keep_offset dw 0
data ends

code segment
     assume ds:data, cs:code, ss:stack

     interrupt proc far
 push ax
 push cx


  IN   AL,  61h
  PUSH AX
  OR   AL,  00000011b
  OUT  61h, AL
  MOV  AL,  90
  OUT  42h, AL

  MOV  CX,  1000


Zvuk:
  PUSH CX
  MOV  CX,  150
```

```
  Cicle:
     LOOP Cicle
  POP  CX
  LOOP Zvuk

  POP  AX             ;

  AND  AL,  11111100b ;
  OUT  61h, AL
pop cx
pop ax
mov al, 20h
out 20h, al

iret
     interrupt endp


     main proc far
          push ds
          sub ax, ax
          push ax

          mov ax, data
          mov ds, ax


          mov ax, 351ch
          int 21h

          mov keep_offset, bx
          mov keep_seg, es
          cli
          push ds
          mov dx, offset interrupt
          mov ax, seg interrupt
          mov ds, ax

          mov ax, 251ch
          int 21h

          pop ds
          sti



           looper:

          mov ah, 1h
          int 21h
          cmp al, '1'
```

```
            je next
            jmp looper

            next:


            cli
            push ds

            mov dx, keep_offset
            mov ax, keep_seg
            mov ds, ax

            mov ah, 25h
            mov al, 1ch
            int 21h

            pop ds
            sti


            ret
      main endp
code ends
end main
```