**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. 0383 _____ Живаев М.А.

Преподаватели _____ Ефремов М.А.

Санкт-Петербург
2021

**Цель работы.**

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

**Задание.**

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

<действие 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;

**Выполнение 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 lab5.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 dx; save reg
  push bx; save reg


    mov     bx,270    ; Hz
    mov     ax,34DDh
    mov     dx,12h    ;(dx,ax)=1193181 , 12h -> 18Hz
    cmp     dx,bx     ;
    jnb     Done      ;
    div     bx        ;ax=(dx,ax)/bx (word)
    mov     bx,ax     ;
    in      al,61h    ;
    or      al,3      ; 0 -> ch2, 1 -> out
    out     61h,al
    mov     al,00000110b   ; 3-1-> imp, 0->format
```

```
   mov     dx,43h    ; timer
   out     dx,al     ;
   dec     dx    ;     channel 2 42h
   mov     al,bl     ;
   out     dx,al     ;
   mov     al,bh     ;
   out     dx,al     ;
Done:

pop bx
pop dx
pop ax

iret
interrupt endp


        main proc far
                push ds
                sub ax, ax
                push ax

                mov ax, data
                mov ds, ax


                mov ax, 351ch ; 35 - get vec(bx = offset, es = seg), 1ch - ?vec
                int 21h ;

                mov keep_offset, bx ; save vec
                mov keep_seg, es

                cli
                push ds
                mov dx, offset interrupt ;
```

```
                mov ax, seg interrupt    ;
                mov ds, ax

                mov ax, 251ch ; 25 - set(offset = dx, seg = ds), 1ch - ?vec
                int 21h

                pop ds
                sti

looper:

                mov ah, 1h ; 1h - get char
                int 21h
                cmp al, '1'
                je next
                jmp looper
next:

   push    ax  ;
   in  al,61h  ;
   and al,not 3; turn off 0,1 bit
   out 61h,al  ;
   pop ax  ;




                cli
                push ds

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

                mov ah, 25h ; 25h - set(25 - set(offset = dx, seg = ds))
```

```
                mov al, 1ch; 1ch - ?vec
                int 21h ;

                pop ds
                sti

                ret
            main endp
        code ends
        end main
```

**Текст файла lab5.lst**

```
Microsoft (R) Macro Assembler Version 5.10          12/16/21 11:02:1
                                    Page     1-1


0000                        stack segment stack
0000  0006[                         dw 6 dup(?)
        ????
                            ]

000C                        stack ends
0000                        data segment
0000  0000                          keep_seg dw 0
0002  0000                          keep_offset dw 0
0004                        data ends

0000                        code segment
                                    assume ds:data, cs:code, ss:stack



0000                        interrupt proc far
0000  50                      push ax;
```

```
0001  52                  push dx; save reg
0002  53                  push bx; save reg


0003  BB 010E                    mov     bx,270    ; Hz
0006  B8 34DD                    mov     ax,34DDh
0009  BA 0012                    mov     dx,12h    ;(dx,ax)=1193181 , 12h -
                        > 18Hz
000C  3B D3               cmp     dx,bx     ;
000E  73 17               jnb     Done      ;
0010  F7 F3               div     bx        ;ax=(dx,ax)/bx (word)
0012  8B D8               mov     bx,ax     ;
0014  E4 61               in      al,61h    ;
0016  0C 03               or      al,3      ; 0 -> ch2, 1 -> out
0018  E6 61               out     61h,al
001A  B0 06               mov     al,00000110b   ; 3-1-> imp, 0->for
                        mat

001C  BA 0043                     mov     dx,43h    ; timer
001F  EE                  out     dx,al     ;
0020  4A                  dec     dx    ;   channel 2 42h
0021  8A C3               mov     al,bl     ;
0023  EE                  out     dx,al     ;
0024  8A C7               mov     al,bh     ;
0026  EE                  out     dx,al     ;
0027                      Done:

0027  5B                  pop bx
0028  5A                  pop dx
0029  58                  pop ax

002A  CF                  iret
002B                      interrupt endp
```

```
002B  1E                        push ds
002C  2B C0                     sub ax, ax
002E  50                        push ax

002F  B8 ---- R                 mov ax, data
0032  8E D8                     mov ds, ax


0034  B8 351C                           mov ax, 351ch ; 35 - get vec(bx
                        = offset, es = seg), 1ch - ?vec
0037  CD 21                     int 21h ;

0039  89 1E 0002 R              mov keep_offset, bx ; save vec
003D  8C 06 0000 R              mov keep_seg, es

0041  FA                        cli
0042  1E                       push ds
0043  BA 0000 R                 mov dx, offset interrupt ;
0046  B8 ---- R                 mov ax, seg interrupt    ;
0049  8E D8                     mov ds, ax

004B  B8 251C                           mov ax, 251ch ; 25 - set(offset
                        = dx, seg = ds), 1ch - ?vec
004E  CD 21                     int 21h

0050  1F                        pop ds
0051  FB                        sti
```

```
0052                    looper:

0052  B4 01                     mov ah, 1h ; 1h - get char
0054  CD 21                          int 21h
0056  3C 31                          cmp al, '1'
0058  74 02                          je next
005A  EB F6                          jmp looper
005C                    next:

005C  50                  push   ax  ;
005D  E4 61               in  al,61h  ;
005F  24 FC               and al,not 3; turn off 0,1 bit
0061  E6 61               out 61h,al  ;
0063  58                  pop ax  ;




0064  FA                          cli
0065  1E                          push ds
```

```
0066  8B 16 0002 R                    mov dx, keep_offset
006A  A1 0000 R                       mov ax, keep_seg
006D  8E D8                           mov ds, ax

006F  B4 25                           mov ah, 25h ; 25h - set(25 - se
                                    t(offset = dx, seg = ds))
0071  B0 1C                           mov al, 1ch; 1ch - ?vec
0073  CD 21                           int 21h ;

0075  1F                              pop ds
0076  FB                              sti



0077  CB                              ret
0078                                main endp
0078                          code ends
                              end main
```



Segments and Groups:

| N a m e | Length | Align | Combine | Class |
|---|---|---|---|---|
| CODE . . . . . | 0078 | PARA | NONE | |
| DATA . . . . . | 0004 | PARA | NONE | |
| STACK . . . . . | 000C | PARA | STACK | |

Symbols:

```
            N a m e         Type     Value  Attr

DONE . . . . . .            L NEAR         0027   CODE

INTERRUPT . . . . . .       F PROC         0000   CODE  Length = 002B

KEEP_OFFSET . . . . . .     L WORD         0002   DATA
KEEP_SEG . . . . . .        L WORD         0000   DATA

LOOPER . . . . . .          L NEAR         0052   CODE

MAIN . . . . . .  F PROC          002B   CODE  Length = 004D

NEXT . . . . . .  L NEAR          005C   CODE

@CPU . . . . . .            TEXT  0101h
@FILENAME . . . . . .       TEXT  lab5
@VERSION . . . . . .        TEXT  510


   118 Source  Lines
   118 Total   Lines
    15 Symbols

 48018 + 461289 Bytes symbol space free

     0 Warning Errors
     0 Severe  Errors
```