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»

Тема: Разработка собственного прерывания.

Студент гр. 9383 _____ Чумак М.А.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург

2020

Цель работы.

Изучить основные принципы работы с прерываниями. Применить на практике полученные знания о прерываниях, разработать своё собственное прерывание.

Краткие 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.

Вариант 4B.

Разработать собственное прерывание.

4 - 08h - прерывание от системного таймера - генерируется автоматически операционной системой 18 раз в сек.

B - Выдача звукового сигнала;

Выполнение работы.

Была реализована программа, которая с помощью прерывания 60h производит звуковой сигнал. Для этого было необходимо сохранить сегмент

вектора прерывания и смещение изначального прерывания 60h. После этого прерывание 60h было переопределено под вызов функции SUBR_INT, в которой выполняется воспроизведение звукового сигнала через динамик. После воспроизведения звукового сигнала, старый вектор прерывания восстанавливается обратно. Функция MAIN завершается при помощи инструкции RET.

Исходный код программы представлен в приложении А.

Тестирование.

При запуске программы пользователь слышит непрерывный звуковой сигнал заданной программой частоты, пока не нажмёт ESC.

Выводы.

Были изучены основные принципы работы с прерываниями. Были применены на практике полученные знания о прерываниях и было разработано своё собственное прерывание.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММ

Файл lab5.asm

```
AStack SEGMENT STACK
	DB 1024 DUP(?)
AStack ENDS

DATA SEGMENT
	keep_cs DW 0		;для хранения сегмента
	keep_ip DW 0		;и смещения прерывания
DATA ENDS

CODE SEGMENT
   ASSUME SS:AStack, DS:DATA, CS:CODE

SUBR_INT PROC FAR
	jmp begin
	int_keep_ss DW 0	;для хранения начальных значений сегмента стэка,
	int_keep_sp DW 0	;указателя на стэк,
	int_keep_ax DW 0	;регистра промежуточных операций
	IntStack DW 16 DUP(?)	;внутренний стэк
begin:
	mov int_keep_sp, sp	;запоминаем
	mov int_keep_ax, ax	;нужные нам
	mov ax, ss		;начальные
	mov int_keep_ss, ax	;регистры

	mov ax, int_keep_ax
	mov sp, OFFSET begin
	mov ax, seg IntStack
```

```
        mov ss, ax

        push ax                 ;сохранение
        push dx                 ;изменяемых регистров

        mov al, 10110110b               ;устанавливаем режим для
        out 43h, al             ;для 2-го канала

        mov ax, 300                     ;определяем звук с заданной частотой

        out 42h, al             ;устанавливаем звук
        mov al, ah              ;в порт
        out 42h, al             ;динамика

        in al, 61h              ;выбор режима
        mov ah, al              ;управления
        or al, 3                ;динамиком
        out 61h, al             ;включение звука
        sub cx, cx
kill_time:
        loop kill_time
        mov al, ah
        out 61h, al             ;выключение звука
        pop dx                          ;восстановление регистра dx
        pop ax                          ;восстановление регистра ax
        mov int_keep_ax, ax
        mov sp, int_keep_sp
        mov ax, int_keep_ss
        mov ss, ax
        mov ax, int_keep_ax
```

```
        mov al, 20h             ;разрешаем обработку прерываний
        out 20h, al             ;с более низкими уровнями
        iret                    ;конец прерывания

SUBR_INT ENDP

MAIN PROC FAR
        push ds
        sub ax, ax
        push ax
        mov ax, DATA
        mov ds, ax

        mov ah, 35h             ;получаем вектор прерывания
        mov al, 08h
        int 21h

        mov keep_ip, bx
        mov keep_cs, es
        push ds
        mov dx, offset SUBR_INT
        mov ax, seg SUBR_INT
        mov ds, ax

        mov ah, 25h             ;устанавливаем вектор прерывания
        mov al, 08h
        int 21h

        pop ds
check_end:
```

```
        mov ah, 01h                 ;получаем символ
        int 21h
        cmp al, 1bh                 ;программа работает, пока не будет нажат
ESC
        je func_end
        jmp check_end
    func_end:
        cli                         ;запрещаем прерывания от внешних
устройств
        push ds
        mov dx, keep_ip
        mov ax, keep_cs
        mov ds, ax

        mov ah, 25h
        mov al, 08h
        int 21h

        pop ds
        sti                         ;разрешаем прерывания от внешних
устройств
        ret
    MAIN ENDP

    CODE ENDS

    END MAIN
```

Файл lab5.lst

```
#Microsoft (R) Macro Assembler Version 5.10          12/9/20 23:41:39
                                                     Page     1-1
```

```
0000                    AStack SEGMENT STACK
0000  0400[                        DB 1024 DUP(?)
        ??
                        ]

0400                    AStack ENDS

0000                    DATA SEGMENT
0000  0000                   keep_cs DW 0                  ;PrP»CЏ
                       C…CЂP°PSPµPSPëCЏ CЃPµPiPjPµPSC,P°
0002  0000                   keep_ip DW 0                  ;Pë CЃP
                       jPµC‰PµPSPëCЏ PїCЂPµCЂC‹PIP°PSPëCЏ
0004                    DATA ENDS

0000                    CODE SEGMENT
                          ASSUME SS:AStack, DS:DATA, CS:CODE

0000                    SUBR_INT PROC FAR
0000  EB 27 90                 jmp begin
0003  0000                   int_keep_ss DW 0       ;PrP»CЏ
                        C…CЂP°PSPµPSPëCЏ  PSP°C‡P°P»CЊPSC‹C…
P·PSP°C‡Pµ
                       PSPëP№ CЃPµPiPjPµPSC,P° CЃC,CЌPєP°,
0005  0000                   int_keep_sp DW 0       ;CrPєP°
                       P·P°C,PµP»CЏ PSP° CЃC,CЌPє,
0007  0000                   int_keep_ax DW 0       ;CЂPµPi
                       PëCЃC,CЂP°      PїCЂPsPjPµP¶CrC,PsC‡PSC‹C…
PsPїPµCЂP
```

```
                        °C†PëP№
0009  0010[                      IntStack DW 16 DUP(?)       ;PIPSCŕ
                        C,CЂPµPSPSPëP№ CЃC,CЌPє
        ????
                    ]

0029                    begin:
0029  2E: 89 26 0005 R           mov int_keep_sp, sp         ;P·P°Pï
                        PsPjPëPSP°PµPj
002E  2E: A3 0007 R              mov int_keep_ax, ax         ;PSCŕP¶
                        PSC‹Pµ PSP°Pj
0032  8C D0                      mov ax, ss             ;PSP°C‡
                        P°P»CЊPSC‹Pµ
0034  2E: A3 0003 R              mov int_keep_ss, ax         ;CЂPµPi
                        PëCЃC,CЂC‹

0038  2E: A1 0007 R              mov ax, int_keep_ax
003C  BC 0029 R             mov sp, OFFSET begin
003F  B8 ---- R             mov ax, seg IntStack
0042  8E D0                      mov ss, ax

0044  50                         push ax                ;CЃPsC…CЂP°PSPµ
                        PSPëPµ
0045  52                         push dx                ;PëP·PjPµPSCЏPµ
                        PjC‹C… CЂPµPiPëCЃC,CЂPsPI

0046  B0 B6                      mov al, 10110110b           ;CŕCЃC,
                        P°PSP°PIP»PëPIP°PµPj CЂPµP¶PëPj PrP»CЏ
#Microsoft (R) Macro Assembler Version 5.10          12/9/20 23:41:39
                                 Page     1-2
```

```
0048  E6 43                 out 43h, al             ;PrP»CЏI
                            2-PiPs PєP°PSP°P»P°

004A  B8 012C               mov ax, 300             ;PsPїCЋ
                            PµPrPµP»CЏIPµPj P·PICґPє CЃ
P·P°PrP°PSPSPsP№ C‡P
                            °CЃC,PsC,PsP№

004D  E6 42                 out 42h, al             ;CґCЃC,
                            P°PSP°PIP»PёPIP°PµPj P·PICґPє
004F  8A C4                 mov al, ah              ;PI PїP
                            sCЋC,
0051  E6 42                 out 42h, al             ;PrPёPS
                            P°PjPёPєP°

0053  E4 61                 in al, 61h              ;PIC‹P±
                            PsCЋ CЋPµP¶PёPjP°
0055  8A E0                 mov ah, al              ;CґPїCЋ
                            P°PIP»PµPSPёCЏI
0057  0C 03                 or al, 3                ;PrPёPS
                            P°PjPёPєPsPj
0059  E6 61                 out 61h, al             ;PIPєP»
                            CЋC‡PµPSPёPµ P·PICґPєP°
005B  2B C9                 sub cx, cx
005D                        kill_time:
005D  E2 FE                 loop kill_time
005F  8A C4                 mov al, ah
0061  E6 61                 out 61h, al             ;PIC‹Pє
```

```
                                P»CЋC‡PµPSPëPµ P·PICґPєP°
        0063  5A                        pop dx                  ;PIPsCЃ
                                CЃC,P°PSPsPIP»PµPSPëPµ  CЂPµPiPëCЃC,CЂP°
dx
        0064  58                        pop ax                  ;PIPsCЃ
                                CЃC,P°PSPsPIP»PµPSPëPµ  CЂPµPiPëCЃC,CЂP°
ax
        0065  2E: A3 0007 R             mov int_keep_ax, ax
        0069  2E: 8B 26 0005 R          mov sp, int_keep_sp
        006E  2E: A1 0003 R             mov ax, int_keep_ss
        0072  8E D0                     mov ss, ax
        0074  2E: A1 0007 R             mov ax, int_keep_ax
        0078  B0 20                     mov al, 20h             ;CЂP°P·
                                CЂPµC€P°PµPj            PsP±CЂP°P±PsC,PєCґ
PїCЂPµCЂC‹PIP°P
                                SPëP№
        007A  E6 20                     out 20h, al             ;CЃ P±P
                                sP»PµPµ PSPëP·PєPëPjPë CґCЂPsPIPSCЏPjPë
        007C  CF                        iret                    ;PєPsPS
                                PµC† PїCЂPµCЂC‹PIP°PSPëCЏ

        007D                    SUBR_INT ENDP

        007D                    MAIN PROC FAR
        007D  1E                        push ds
        007E  2B C0                     sub ax, ax
        0080  50                        push ax
        0081  B8 ---- R                 mov ax, DATA
        0084  8E D8                     mov ds, ax
      #Microsoft (R) Macro Assembler Version 5.10        12/9/20 23:41:39
```

```
0086  B4 35                    mov ah, 35h             ;PïPsP»
                        CŕC‡P°PµPj                    PIPµPєC,PsCЂ
PïCЂPµCЂC‹PIP°PSPëCЏI
0088  B0 08                    mov al, 08h
008A  CD 21                    int 21h

008C  89 1E 0002 R             mov keep_ip, bx
0090  8C 06 0000 R             mov keep_cs, es
0094  1E                  push ds
0095  BA 0000 R           mov dx, offset SUBR_INT
0098  B8 ---- R           mov ax, seg SUBR_INT
009B  8E D8                    mov ds, ax

009D  B4 25                    mov ah, 25h             ;CŕCЃC,
                        P°PSP°PIP»PëPIP°PµPj          PIPµPєC,PsCЂ
PïCЂPµCЂC‹PIP
                        °PSPëCЏI
009F  B0 08                    mov al, 08h
00A1  CD 21                    int 21h

00A3  1F                  pop ds
00A4                    check_end:
00A4  B4 01                    mov ah, 01h                 ;PïPsP»
                        CŕC‡P°PµPj CЃPëPjPIPsP»
00A6  CD 21                    int 21h
00A8  3C 1B                    cmp al, 1bh             ;PïCЂPs
```

```
                        PiCЂP°PjPjP° CЂP°P±PsC,P°PµC,, PїPsPєP° PSPµ
P±
                        CґPrPµC, PSP°P¶P°C, ESC
        00AA  74 02             je func_end
        00AC  EB F6             jmp check_end
        00AE            func_end:
        00AE  FA                cli             ;P·P°Pї
                        CЂPµC‰P°PµPj    PїCЂPµCЂC‹PIP°PSPёCЏI
PsC, PIPSPµC€
                        PSPёC… CґCЃC,CЂPsP№CЃC,PI
        00AF  1E                push ds
        00B0  8B 16 0002 R      mov dx, keep_ip
        00B4  A1 0000 R         mov ax, keep_cs
        00B7  8E D8             mov ds, ax

        00B9  B4 25             mov ah, 25h
        00BB  B0 08             mov al, 08h
        00BD  CD 21             int 21h

        00BF  1F                pop ds
        00C0  FB                sti             ;CЂP°P·
                        CЂPµC€P°PµPj    PїCЂPµCЂC‹PIP°PSPёCЏI
PsC, PIPSPµC€
                        PSPёC… CґCЃC,CЂPsP№CЃC,PI
        00C1  CB                ret
        00C2            MAIN ENDP

        00C2            CODE ENDS

                        END MAIN
```

Segments and Groups:

N a m e	Length	Align	Combine	Class
ASTACK	0400	PARA	STACK	
CODE	00C2	PARA	NONE	
DATA	0004	PARA	NONE	

Symbols:

N a m e	Type	Value	Attr	
BEGIN	L NEAR	0029	CODE	
CHECK_END	L NEAR	00A4	CODE	
FUNC_END	L NEAR	00AE	CODE	
INTSTACK	L WORD	0009	CODE	Length = 0010
INT_KEEP_AX	L WORD	0007	CODE	
INT_KEEP_SP	L WORD	0005	CODE	
INT_KEEP_SS	L WORD	0003	CODE	
KEEP_CS	L WORD	0000	DATA	
KEEP_IP	L WORD	0002	DATA	
KILL_TIME	L NEAR	005D	CODE	

```
MAIN . . . . . .          F PROC      007D CODE       Length = 0045

SUBR_INT . . . . . .      F PROC      0000  CODE      Length = 007D

@CPU . . . . . .          TEXT  0101h
@FILENAME  . . . . . .    TEXT  LAB5
@VERSION . . . . . .      TEXT  510


   111 Source  Lines
   111 Total   Lines
    20 Symbols

 48004 + 457206 Bytes symbol space free

     0 Warning Errors
     0 Severe  Errors
```