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: Написание собственного прерывания
Вариант 3 (1С)

Студентка гр. 0382 _____ Деткова А.С.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург
2021

Цель работы.

Изучить понятие прерывания. Познакомиться с основными системными прерываниями. Написать обработчик собственного прерывания.

Задание.

Вариант 3. Шифр задания — 1С.

Написать процедуру обработки прерывания.

1 — 08h: прерывание от системного таймера, генерируется автоматически операционной системой 18 раз в сек;

С — Выдача звукового сигнала с заданной длительностью звучания.

Основные 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.

Выполнение работы.

Сегмент данных: *KEEP_CS* — сохранение сегмента, *KEEP_IP* — сохранение смещение прерывания, так как функция *35H* прерывания *21H* возвращает текущее значение вектора прерывания, помещая значение сегмента в *ES*, а смещение в *BX*.

В функции *MAIN*:

В начале работы программы, вызываем функцию *35H* прерывания *21H* и сохраняем текущее значение сегмента и смещения для обрабатываемого прерывания.

Далее задается адрес собственного прерывания, вызывается функция 25*H* прерывания *21H,* меняем прерывание.

Чтобы обрабатываемое прерывание работало, в регистр bx помещается частота генерируемого звука. Звук генерируется, пока пользователь не нажмет на клавишу *esc*.

В конце работы восстанавливаем старый вектор прерывания.

В функции *SUBR_INT*:

Создается внутренний стек для безопасности, записываются новые значения регистров *SS* и *SP* в соответствии с новым стеком.

Сохраняются на стек изменяемые внутри обработчика прерываний регистров. Включается динамик, генерируется звук на заданной частоте, выключается динамик. Восстанавливаем изменяемые регистры со стека.

Восстанавливаем значения регистров *SS* и *SP.*

Тестирование.

После запуска программа включает динамик, он генерирует звук, пока пользователь не нажмет клавишу *esc*.

Выводы.

Было изучено понятие прерывания.

В результате лабораторной работы была разработана программа, которая перехватывает прерывание 08H и выдает звуковой сигнал, пока не будет нажат клавиша esc.

ПРИЛОЖЕНИЕ А

КОД ПРОГРАММЫ

Название файла: Lab5.asm

```
AStack    SEGMENT  STACK
          DW 20 DUP('!')
AStack    ENDS


DATA      SEGMENT

    KEEP_CS DW 0 ; для хранения сегмента
    KEEP_IP DW 0 ; и смещения прерывания

DATA      ENDS


CODE      SEGMENT

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


SUBR_INT  PROC FAR

          jmp handle
          SS_int dw 0
          SP_int dw 0
          int_Stack DW 512 DUP('0')

          handle:

          mov SS_int, SS
          mov SP_int, SP

          mov SP, seg int_Stack
          mov SS, SP
          mov SP, offset handle

          push ax ; сохранение изменяемых регистров
          push cx


          mov     al, 10110110b
          out     43h, al
          mov     ax, bx
          out     42h, al
          mov     al, ah
          out     42h, al

          in      al,61h
          or      al,03h
```

```
           out     61h,al

           sub cx, cx
           TIME: LOOP TIME

         in      al,61h
         and     al,0fch
         out     61h,al


           pop cx ; восстановление изменяемых регистров
           pop ax

           mov SS, SS_int
           mov SP, SP_int

           MOV AL, 20H
           OUT 20H,AL

           IRET
SUBR_INT   ENDP

MAIN    PROC  FAR

         push  DS
         sub   AX,AX
         push  AX
         mov   AX,DATA
         mov   DS,AX

           mov AH, 35h ; функция получения вектора
           mov AL, 08h ; номер вектора
           int 21h
           mov KEEP_IP, bx ; сохраняем смещение
           mov KEEP_CS, es ; сохраняем сегмент

           push ds
           mov dx, offset SUBR_INT
           mov ax, seg SUBR_INT
           mov ds, ax
           mov ah, 25h
           mov al, 08h
           int 21h
           pop ds

         mov bx, 4000 ; частота звука записывается в регистр bx

           get_key:

              mov ah, 0h
              int 16h
              cmp al, 27

           jne get_key
```

```
            cli
            push DS
            mov  DX, KEEP_IP
            mov  AX, KEEP_CS
            mov  DS, AX
            mov ah, 25h
            mov al, 08h
            int  21H          ; восстанавливаем вектор
            pop  DS
            sti

            ret

Main        ENDP
CODE        ENDS
            END MAIN
```

ПРИЛОЖЕНИЕ Б

ЛИСТИНГ ПРОГРАММЫ

Название файла: Lab5.LST

```
Microsoft (R) Macro Assembler Version 5.10                  12/15/21
22:06:1
                                                              Page
1-1


 0000                   AStack    SEGMENT  STACK
 0000  0014[                            DW 20 DUP('!')
         0021
                    ]

 0028                   AStack    ENDS


 0000                   DATA      SEGMENT

 0000  0000                      KEEP_CS DW 0
 0002  0000                      KEEP_IP DW 0

 0004                   DATA      ENDS


 0000                   CODE      SEGMENT

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


 0000                   SUBR_INT  PROC FAR

 0000  E9 0407 R                 jmp handle
 0003  0000                            SS_int dw 0
 0005  0000                            SP_int dw 0
 0007  0200[                           int_Stack DW 512 DUP('0')
         0030
                    ]


 0407                             handle:

 0407  2E: 8C 16 0003 R                       mov SS_int, SS
 040C  2E: 89 26 0005 R                       mov SP_int, SP

 0411  BC ---- R                  mov SP, seg int_Stack
 0414  8E D4                            mov SS, SP
 0416  BC 0407 R                  mov SP, offset handle

 0419  50                         push ax
```

```
 041A  51                          push cx


 041B  B0 B6                            mov     al, 10110110b
 041D  E6 43                            out     43h, al
 041F  8B C3                            mov     ax, bx
 0421  E6 42                            out     42h, al
 0423  8A C4                            mov     al, ah
 0425  E6 42                            out     42h, al

Microsoft (R) Macro Assembler Version 5.10                      12/15/21
22:06:1
                                                                    Page
1-2


 0427  E4 61                            in      al,61h
 0429  0C 03                            or      al,03h
 042B  E6 61                            out     61h,al

 042D  2B C9                            sub cx, cx
 042F  E2 FE                            TIME: LOOP TIME

 0431  E4 61                         in      al,61h
 0433  24 FC                         and     al,0fch
 0435  E6 61                         out     61h,al


 0437  59                          pop cx
 0438  58                          pop ax

 0439  2E: 8E 16 0003 R                      mov SS, SS_int
 043E  2E: 8B 26 0005 R                      mov SP, SP_int

 0443  B0 20                            MOV AL, 20H
 0445  E6 20                            OUT 20H,AL

 0447  CF                          IRET
 0448                  SUBR_INT  ENDP

 0448                  MAIN    PROC  FAR

 0448  1E                          push  DS
 0449  2B C0                          sub   AX,AX
 044B  50                          push  AX
 044C  B8 ---- R                   mov   AX,DATA
 044F  8E D8                          mov   DS,AX

 0451  B4 35                            mov AH, 35h
 0453  B0 08                            mov AL, 08h
 0455  CD 21                            int 21h
 0457  89 1E 0002 R                     mov KEEP_IP, bx
 045B  8C 06 0000 R                     mov KEEP_CS, es

 045F  1E                          push ds
```

```
 0460  BA 0000 R               mov dx, offset SUBR_INT
 0463  B8 ---- R               mov ax, seg SUBR_INT
 0466  8E D8                        mov ds, ax
 0468  B4 25                        mov ah, 25h
 046A  B0 08                        mov al, 08h
 046C  CD 21                        int 21h
 046E  1F                    pop ds

 046F  BB 0FA0                  mov bx, 4000
Microsoft (R) Macro Assembler Version 5.10                  12/15/21
22:06:1
                                                              Page
1-3

 0472                        get_key:

 0472  B4 00                           mov ah, 0h
 0474  CD 16                           int 16h
 0476  3C 1B                           cmp al, 27

 0478  75 F8                        jne get_key

 047A  FA                    cli
 047B  1E                    push DS
 047C  8B 16 0002 R                 mov  DX, KEEP_IP
 0480  A1 0000 R             mov  AX, KEEP_CS
 0483  8E D8                        mov  DS, AX
 0485  B4 25                        mov ah, 25h
 0487  B0 08                        mov al, 08h
 0489  CD 21                        int  21H
 048B  1F                    pop  DS
 048C  FB                    sti

 048D  CB                    ret

 048E                Main    ENDP
 048E                CODE    ENDS
                             END MAIN
Microsoft (R) Macro Assembler Version 5.10                  12/15/21
22:06:1
                                                           Symbols-1


Segments and Groups:

                N a m e         Length      Align     Combine Class

ASTACK . . . . . .              0028  PARA  STACK
CODE . . . . . .                048E  PARA  NONE
DATA . . . . . .                0004  PARA  NONE

Symbols:

                N a m e         Type   Value      Attr
```

```
GET_KEY  . . . . . .      L NEAR     0472  CODE

HANDLE . . . . . .        L NEAR     0407  CODE

INT_STACK  . . . . . .    L WORD     0007  CODE  Length    =
0200

KEEP_CS  . . . . . .      L WORD     0000  DATA
KEEP_IP  . . . . . .      L WORD     0002  DATA

MAIN . . . . . .          F PROC     0448  CODE  Length    =
0046

SP_INT . . . . . .        L WORD     0005  CODE
SS_INT . . . . . .        L WORD     0003  CODE
SUBR_INT . . . . . .      F PROC     0000  CODE  Length    =
0448

TIME . . . . . .          L NEAR     042F  CODE

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


    118 Source  Lines
    118 Total   Lines
     18 Symbols

  48000 + 459257 Bytes symbol space free

      0 Warning Errors
      0 Severe  Errors
```