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 _____ Тарасов К.О.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург

2021

Цель работы

Написать программу обработки прерывания

Задание

Вариант 2а:

2 - 60h - прерывание пользователя - должно генерироваться в программе;

A - Выполнить вывод сообщения на экран заданное число раз,после чего вставить фиксированную задержку и вывести сообщение о завершении обработчика.

Ход работы

Прерывание реализовано в процедуре MY_INT. В процедуре Main с помощью функции 35h/int 21h запоминается текущий вектор прерывания под номером 60h в переменные KEEP_CS, KEEP_IP. С помощью функции 25h/int 21h устанавливается новый вектор прерывания (реализованная процедура прерывания). Далее это прерывание вызывается в программе, предварительно в CX командой MOV заносится некоторое положительное число, соответствующее количеству вывода строки на экран. В конце программы вектор прерывания под номером 60h восстанавливается с помощью переменных KEEP_CS и KEEP_IP

Тестирование

Табл. 1. Результат тестирования.

Вызванные команды	Результат	Комментарий
mov cx, 3	Sample text Sample text Sample text End	Верно
mov cx, 5	Sample text Sample text Sample text Sample text Sample text End	Верно
mov cx, 7	Sample text Sample text Sample text	Верно

	Sample text Sample text Sample text Sample text End	

Выводы:

В ходе выполнения работы было разработано собственное прерывание на языке Ассемблер.

ПРИЛОЖЕНИЕ А

Текст компонентов программы lr5

lr5.asm

```
DATA SEGMENT
    KEEP_CS DW 0 ; для хранения сегмента
    KEEP_IP DW 0 ; и смещения вектора прерывания
            TMP1 DW 0
            TMP2 DW 0
            TMP3 DW 0
    mes   DB 'Sample text',10,13,'$'
            mes_end   DB 'End',10,13,'$'

DATA ENDS


Stack    SEGMENT  STACK
        DW 12 DUP(?)
Stack    ENDS



CODE     SEGMENT
        ASSUME CS:Code, DS:DATA, SS:Stack

WriteMsg PROC NEAR
         MOV AH,9
         int 21h
         ret
WriteMsg ENDP

MY_INT PROC FAR
    jmp start
```

```
        ST_SS DW 0000
        ST_AX DW 0000
        ST_SP DW 0000
        IStack DW 30 DUP(?)
start:

        mov ST_SP, SP
          mov ST_AX, AX
          mov AX, SS
          mov ST_SS, AX
          mov AX, Stack
          mov SS, AX
          mov AX, ST_AX
          push ax
          push ds

          MOV   DX, OFFSET mes
          metka:
          call WriteMsg  ; Вывод сообщения
          loop metka ; Заданное число раз

          mov di,32
          mov ah,0
          int 1Ah
          mov bx,dx; счетчик с момента сброса
Delay:
          mov ah,0
          int 1Ah
          sub dx,bx
          cmp di,dx
          ja Delay;переход,если больше

          MOV DX, OFFSET mes_end ;Вывод сообщения о завершении обработчика
          MOV AH,9
          int 21h
```

```
        pop dx
      pop ax
        mov ST_AX,AX
        mov AX,ST_SS
        mov SS,AX
        mov SP,ST_SP
        mov AX,ST_AX
        mov al,20h
        out 20h,al

        iret
MY_INT ENDP


Main    PROC  FAR

      push  DS      ;\  Сохранение адреса начала PSP в стеке
      sub   AX,AX   ; > для последующего восстановления по
      push  AX      ;/  команде ret, завершающей процедуру.
      mov   AX,DATA           ; Загрузка сегментного
      mov   DS,AX             ; регистра данных.

      MOV AH, 35H ; функция получения вектора
      MOV AL, 60H ; номер вектора
      INT 21H ; возвращает текущее значение вектора прерывания
      MOV KEEP_IP, BX ; запоминание смещения
      MOV KEEP_CS, ES ; и сегмента вектора прерывания

      PUSH DS
      MOV DX, OFFSET MY_INT ; смещение для процедуры в DX
      MOV AX, SEG MY_INT ; 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

          mov cx, 3
        int 60H; вызов измененного прерывания

        CLI
        PUSH DS
        MOV DX, KEEP_IP
        MOV AX, KEEP_CS
        MOV DS, AX
        MOV AH, 25H
        MOV AL, 60H
        INT 21H ; восстанавливаем старый вектор прерывания
        POP DS
        STI

        RET
Main    ENDP
CODE    ENDS
        END Main
```

lr5.lst

```
0000                    DATA SEGMENT
0000  0000                    KEEP_CS DW 0 ; РґР»СЏ С…СЂР°РЅРµРЅРёСЏ
                        СЃРµРіРјРµРЅС‚Р°
0002  0000                    KEEP_IP DW 0 ; Рё СЃРјРµС‰РµРЅРёСЏ РІРµ
                        СЂС,РsСЂР° РїСЂРµСЂС‹РІР°РЅРёСЏ
0004  0000                          TMP1 DW 0
0006  0000                          TMP2 DW 0
0008  0000                          TMP3 DW 0
000A  53 61 6D 70 6C 65       mes   DB 'Sample text',10,13,'$'
    20 74 65 78 74 0A
    0D 24
0018  45 6E 64 0A 0D 24              mes_end   DB 'End',10,13,'$'

001E                    DATA ENDS


0000                    Stack    SEGMENT  STACK
0000  000C[                    DW 12 DUP(?)
        ????
                      ]

0018                    Stack    ENDS


0000                    CODE     SEGMENT
                              ASSUME CS:Code, DS:DATA, SS:Stack

0000                    WriteMsg PROC NEAR
0000  B4 09                    MOV AH,9
0002  CD 21                    int 21h
```

```
0004  C3                          ret
0005                        WriteMsg ENDP

0005                        MY_INT PROC FAR
0005  EB 43 90                      jmp start

0008  0000                    ST_SS DW 0000
000A  0000                    ST_AX DW 0000
000C  0000                    ST_SP DW 0000
000E  001E[                   IStack DW 30 DUP(?)
          ????
                        ]

004A                        start:

004A  2E: 89 26 000C R              mov ST_SP, SP
004F  2E: A3 000A R                   mov ST_AX, AX
0053  8C D0                   mov AX, SS
0055  2E: A3 0008 R           mov ST_SS, AX
0059  B8 ---- R               mov AX, Stack
005C  8E D0                   mov SS, AX
005E  2E: A1 000A R                 mov AX, ST_AX
0062  50                      push ax
```

```
0063  1E                        push ds

0064  BA 000A R                 MOV   DX, OFFSET mes
0067                            metka:
0067  E8 0000 R                 call WriteMsg  ; P'C‹PIPsPr CЃPsPsP±C‰P
                                µPSPëCЏ
006A  E2 FB                     loop metka ; P—P°PrP°PSPSPsPµ C‡PëCЃP»P
                                s CЂP°P·

006C  BF 0020                       mov di,32
006F  B4 00                     mov ah,0
0071  CD 1A                     int 1Ah
0073  8B DA                     mov bx,dx; CЃC‡PµC,C‡PëPє CЃ PjPsPjPµPS
                                C,P° CЃP±CЂPsCЃP°
0075                            Delay:
0075  B4 00                     mov ah,0
0077  CD 1A                     int 1Ah
0079  2B D3                     sub dx,bx
007B  3B FA                     cmp di,dx
007D  77 F6                     ja Delay;PїPµCЂPµC…PsPr,PµCЃP»Pë P±PsP»
                                CЊCЄPµ

007F  BA 0018 R                 MOV DX, OFFSET mes_end ;P'C‹PIPsPr CЃPs
                                PsP±C‰PµPSPëCЏ Ps P·P°PIPµCЂCЄPµPSPëPë PsP±CЂP°
                                P±PsC,C‡PëPєP°
0082  B4 09                     MOV AH,9
0084  CD 21                     int 21h
0086  5A                       pop dx
0087  58                      pop ax
0088  2E: A3 000A R                 mov ST_AX,AX
008C  2E: A1 0008 R             mov AX,ST_SS
```

```
0090  8E D0                     mov SS,AX
0092  2E: 8B 26 000C R               mov SP,ST_SP
0097  2E: A1 000A R               mov AX,ST_AX
009B  B0 20                     mov al,20h
009D  E6 20                     out 20h,al

009F  CF                        iret
00A0                      MY_INT ENDP


00A0                      Main      PROC  FAR

00A0  1E                        push  DS      ;\ PЎPsC…CЂP°PSPµPSPёPµ
                          P°PrCЂPµCЃP° PSP°C‡P°P»P° PSP PI CЃC,PµPєPµ
00A1  2B C0                     sub   AX,AX    ; > PrP»CЏ PїPsCЃP»PµPrC
                          ŕCЂC‰PµPiPs PIPsCЃCЃC,P°PSPsPIP»PµPSPёCЏ PїPs
00A3  50                        push  AX      ;/  PєPsPjP°PSPrPµ ret,
                          P·P°PIPµCЂC€P°CЂC‰PµP№ PїCЂPsC†PµPrCŕCЂCŕ.
00A4  B8 ---- R                 mov   AX,DATA           ; P—P°PiCЂCŕP
                          ·PєP° CЃPµPiPjPµPSC,PSPsPiPs
00A7  8E D8                     mov   DS,AX             ; CЂPµPiPёCЃC
                          ,CЂP° PrP°PSPSC‹C….
```

```
00A9  B4 35                   MOV AH, 35H ; C„CґPSPєC†PëCЏ PїPsP»CґC‡
                              PµPSPëCЏ PIPµPєC,PsCЂP°
00AB  B0 60                   MOV AL, 60H ; PSPsPjPµCЂ PIPµPєC,PsCЂP°
00AD  CD 21                   INT 21H ; PIPsP·PICЂP°C‰P°PµC, C,PµPєCґ
                              C‰PµPµ P·PSP°C‡PµPSPëPµ PIPµPєC,PsCЂP° PїCЂPµCЂ
                              C‹PIP°PSPëCЏ
00AF  89 1E 0002 R            MOV KEEP_IP, BX ; P·P°PїPsPjPëPSP°PSPëP
                              µ CЃPjPµC‰PµPSPëCЏ
00B3  8C 06 0000 R            MOV KEEP_CS, ES ; Pë CЃPµPiPjPµPSC,P° P
                              IPµPєC,PsCЂP° PїCЂPµCЂC‹PIP°PSPëCЏ

00B7  1E                      PUSH DS
00B8  BA 0005 R               MOV DX, OFFSET MY_INT ; CЃPjPµC‰PµPSPëP
                              µ PrP»CЏ PїCЂPsC†PµPrCґCЂC‹ PI DX
00BB  B8 ---- R               MOV AX, SEG MY_INT ; CЃPµPiPjPµPSC, PїC
                              ЂPsC†PµPrCґCЂC‹
00BE  8E D8                   MOV DS, AX ; PїPsPjPµC‰P°PµPj PI DS
00C0  B4 25                   MOV AH, 25H ; C„CґPSPєC†PëCЏ CґCЃC,P°PS
                              PsPIPєPë PIPµPєC,PsCЂP°
00C2  B0 60                   MOV AL, 60H ; PSPsPjPµCЂ PIPµPєC,PsCЂP°
00C4  CD 21                   INT 21H ; PjPµPSCЏPµPj PїCЂPµCЂC‹PIP°PS
                              PëPµ
00C6  1F                      POP DS

00C7  B9 0003                       mov cx, 3
00CA  CD 60                   int 60H; PIC‹P·PsPI PëP·PjPµPSPµPSPSPsP
                              iPs PїCЂPµCЂC‹PIP°PSPëCЏ

00CC  FA                      CLI
00CD  1E                      PUSH DS
00CE  8B 16 0002 R            MOV DX, KEEP_IP
```

```
00D2  A1 0000 R          MOV AX, KEEP_CS
00D5  8E D8              MOV DS, AX
00D7  B4 25              MOV AH, 25H
00D9  B0 60              MOV AL, 60H
00DB  CD 21              INT 21H ; PIPsCЃCЃC,P°PSP°PIP»PëPIP°PµP
                         j CЃC,P°CЂC‹P№ PIPµPєC,PsCЂ PїCЂPµCЂC‹PIP°PSPëC
                         Ц
00DD  1F                 POP DS
00DE  FB                 STI

00DF  CB                 RET
00E0                     Main      ENDP
00E0                     CODE      ENDS
                             END Main
```

Segments and Groups:

N a m e	Length	Align	Combine	Class
CODE	00E0	PARA	NONE	
DATA	001E	PARA	NONE	
STACK	0018	PARA	STACK	

Symbols:

N a m e	Type	Value	Attr
DELAY	L NEAR	0075	CODE
ISTACK	L WORD	000E	CODE Length = 001E
KEEP_CS	L WORD	0000	DATA
KEEP_IP	L WORD	0002	DATA
MAIN	F PROC	00A0	CODE Length = 0040
MES	L BYTE	000A	DATA
MES_END	L BYTE	0018	DATA
METKA	L NEAR	0067	CODE
MY_INT	F PROC	0005	CODE Length = 009B
START	L NEAR	004A	CODE
ST_AX	L WORD	000A	CODE
ST_SP	L WORD	000C	CODE
ST_SS	L WORD	0008	CODE
TMP1	L WORD	0004	DATA

```
TMP2 . . . . . .        L WORD      0006    DATA
TMP3 . . . . . .        L WORD      0008    DATA

WRITEMSG . . . . . .    N PROC      0000    CODE Length = 0005

@CPU . . . . . .        TEXT  0101h
@FILENAME  . . . . . .          TEXT  lr5
@VERSION . . . . . .    TEXT  510


   120 Source  Lines
   120 Total   Lines
    25 Symbols

 48030 + 459230 Bytes symbol space free

     0 Warning Errors
     0 Severe  Errors
```