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

Студентка гр. 0383 _____ Петровская Е.С.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург
2021

Цель работы.

Изучение работы и принципа создания прерываний на языке Ассемблера.

Задание.

Вариант 26:

4е — по прерыванию от клавиатуры 16h выполнить чтение и ввод на экран отсчета часов реального времени из памяти CMOS (в формате BCD)

Выполнение работы.

Была написана процедура SUBR_INT для реализации прерывания, в котором перед началом обработки самого прерывания сохраняются изначальные регистры. Затем было использовано прерывание 1Ah с серивисом AH = 02h, позволяющим читать время из постоянных CMOS часов реального времени в формате BCD. После этого поочередно происходит вывод данных из регистров CX и DH, преобразованных из BCD формата в ASCII символы соответствующим им числам.

Преобразование происходит с помощью разделения искомого числа на разряды по регистрам AH и AL и дальнейшего их обращения в ASCII символ через сложение со значением 0 в таблице.

Вывод на экран происходит с использованием прерывания 21h.

В функции Main сохраняются исходные значения нынешнего вектора прерывания 60h(его номер и вектор) с помощью функции 25h/INT 21h. Далее вызывается само измененное прерывание и, по завершению его работы, восстанавливается его исходное значение.

Исходный код программы см. в Приложении А

Таблица 1 – Результаты работы программы lab5

Входные данные	Выходные данные	Комментарий
09:32 q	09:32:40	Верно

09:59 q	09:59:13	Верно
10:00 q	10:00:03	Верно

Выводы.

В ходе выполнения лабораторной работы были изучены принципы создания собственных прерываний на языке Ассемблера.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: lab5.asm

```
     ;Create your own interruption
     ;interrupt = proc with certain functions
     ;By the end of the program make sure to return original vectors of
interrupts
     ;VAR 26 - 4e: 16h - interrupt from keyboard(by pressing a key do E:
read and input to screen real-time clock counting from memory CMOS(in BCD
format)
     ;!should use more than 1Kb for stack
               ;PROG:

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

     STACK SEGMENT STACK
          DW 1024 DUP(?)       ;Declare  an  array  of  1024  unitialized
Words (2 bytes) for stack
     STACK ENDS

     DATA  SEGMENT
          KEEP_CS DW 0         ;to store segment
          KEEP_IP DW 0         ;to store interruption vector shift
     DATA  ENDS

     CODE  SEGMENT

     SUBR_INT PROC FAR         ;port 70h is for input(stores addr) use
it to get CMOS
                           ;registers, port 71h - to read from them - if
not using INT 1Ah
          JMP start
          INT_STACK DB 80 DUP(?)

          read_CMOS PROC
               PUSH DX
                    ;hours
               MOV AL, CH      ;in CX=HHMM,
               CALL print_bcd
               CALL colon
                    ;minutes
               MOV AL, CL
               CALL print_bcd
               CALL colon
                    ;seconds
               MOV AL, DH      ;in DH=SS
               CALL print_bcd
               POP DX
               RET
          read_CMOS ENDP

          colon PROC
               MOV DL, ':'
               MOV AH, 02h
```

```
            INT 21H
            RET
        colon ENDP

        print_bcd PROC
            PUSH DX         ;save initial registers
            PUSH CX
            MOV CL, 4
            MOV AH, AL      ;now al = 43 = ah
            AND AL, 00001111b    ;now al = 03 (as it is in BCD, each
digit is xxxxb)
            SHR AH, CL      ;now ah = 04 => ax = 0403
            ADD AL, '0'          ;get ASCII value of '0' + shift in
AL
            ADD AH, '0'
            MOV DL, AH      ;handle print (DL = to print)
            MOV DH, AL
            MOV AH, 02h
            INT 21h
            MOV DL, DH      ;for  some  reason  otherwise  DL  doesnt
change
            INT 21h
            POP CX               ;return initial values
            POP DX
            RET
        print_bcd ENDP

    start:
    ;-------------------------------<save original registers>
        PUSH BP
        MOV BP, SP ;set up the base pointer to the stack storing the
args for this proc
        PUSH AX
        PUSH CX
        PUSH DX
        MOV AX, CS ;set up DS to point to the segment with data items
        MOV DS, AX ;here DS is also CS

    ;-------------------------------<process the interrupt>
        MOV AH, 02H          ;read real time from CMOS
        INT 1Ah         ;returns CX:DX = clock count
        CALL read_CMOS

        POP DX               ;restore registers
        POP CX
        POP AX
        POP BP

        MOV AL, 20H          ;these  lines  allow  to  process  lower
level
        OUT 20H, AL          ;interrupts than those we worked with
        IRET            ;exit from iterrupt
    SUBR_INT ENDP


    Main  PROC FAR
        PUSH DS         ;write into stack
        SUB AX, AX      ;write a 0
```

```
        PUSH AX         ;write ax into stack => stack initialzation
        MOV AX, DATA          ;DataSegment initialization
        MOV DS, AX
    ;-----------<save current vector>
        MOV AH, 35H           ;get curr vector
        MOV AL, 60H           ;get curr vector number
        INT 21H
        MOV KEEP_IP, BX ;store the shift
        MOV KEEP_CS, ES ;store interruption vector segment

    ;----------<install new interrupt vector>
        PUSH DS
        MOV DX, OFFSET SUBR_INT    ;shift fot the proc into DX
        MOV AX, SEG SUBR_INT  ;procedure segment we save and
        MOV DS, AX      ;put into DS
        MOV AH, 25H           ;funtion  to  install  new  vector,  it
stores
                        ;segment and shift addresses into interrupt
                        ;vector with chosen number.
        MOV AL, 60H           ;new vector number
        INT 21H         ;change the interrupt
        POP DS

    ;----------<get key scan-code-(let`s it be 'Q')>
        readkey:
             MOV AH, 0        ;by pressing key in AH a BIOS scancode
is stored, and in AL - an ASCII symbol
             INT 16H          ;interrupt to get the key scancode
             CMP AH, 16       ;16 is a scancode of 'Q'
             JNE readkey           ;if it`s not 'Q' -> repeat reading,
else continue

             INT 60H          ;call changed interrupt

    ;-----------restore original interrupt vector-
        CLI                   ;disable interrupts
        PUSH DS               ;save ds
        MOV DX, KEEP_IP       ;restore original shift
        MOV AX, KEEP_CS       ;restore int vector segment
        MOV DS, AX
        MOV AH, 25H                ;to set int vector
        MOV AL, 60H                ;vector num
        INT 21H               ;restore vector
        POP DS
        STI                   ;enable interrupts
        MOV AH, 4CH
        INT 21H
    Main ENDP
    CODE ENDS
    END Main
```

Название файла: LAB5.LST

```
                        ;Create your own interruption
                        ;interrupt = proc with certain functions
                        ;By the end of the program make sure to return
                        original vectors of interrupts
                        ;VAR 26 - 4e: 16h - interrupt from keyboard(by
                        pressing a key do E: read and input to screen
r
                        eal-time clock counting from memory CMOS(in
BCD
                         format)
                        ;!should use more than 1Kb for stack
                                  ;PROG:

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

0000                    STACK SEGMENT STACK
0000  0400[                       DW 1024 DUP(?)       ;Declare    an
arr
                        ay of 1024 unitialized Words (2 bytes) for
stac
                        k
        ????
                  ]

0800                    STACK ENDS

0000                    DATA  SEGMENT
0000  0000                        KEEP_CS DW 0         ;to      store
segme
                        nt
0002  0000                        KEEP_IP DW 0         ;to      store
inter
                        ruption vector shift
0004                    DATA  ENDS

0000                    CODE  SEGMENT

0000                    SUBR_INT PROC FAR          ;port 70h is fo
                        r input(stores addr) use it to get CMOS
                                                ;registers, por
                        t 71h - to read from them - if not using INT
1A
                        h
0000  E9 0090 R               JMP start
0003  0050[                        INT_STACK DB 80 DUP(?)
       ??
                  ]


0053                              read_CMOS PROC
0053  52                         PUSH DX
                                       ;hours
```

```
0054  8A C5                         MOV AL, CH      ;in CX=
                    HHMM,
0056  E8 0072 R                 CALL print_bcd
0059  E8 006B R                 CALL colon
                                    ;minutes
005C  8A C1                         MOV AL, CL
005E  E8 0072 R                 CALL print_bcd
0061  E8 006B R                 CALL colon
                                    ;seconds

     #Microsoft      (R)      Macro      Assembler      Version      5.10
12/2/21 09:32:30
                                                                      Page
1-2


0064  8A C6                         MOV AL, DH      ;in DH=
                    SS
0066  E8 0072 R                 CALL print_bcd
0069  5A                        POP DX
006A  C3                        RET
006B                       read_CMOS ENDP

006B                       colon PROC
006B  B2 3A                         MOV DL, ':'
006D  B4 02                         MOV AH, 02h
006F  CD 21                         INT 21H
0071  C3                        RET
0072                       colon ENDP

0072                       print_bcd PROC
0072  52                         PUSH DX         ;save initial r
                    egisters
0073  51                        PUSH CX
0074  B1 04                         MOV CL, 4
0076  8A E0                         MOV AH, AL      ;now al
                     = 43 = ah
0078  24 0F                         AND AL, 00001111b     ;now al
                     = 03 (as it is in BCD, each digit is xxxxb)
007A  D2 EC                         SHR AH, CL      ;now ah
                     = 04 => ax = 0403
007C  04 30                         ADD AL, '0'           ;get AS
                    CII value of '0' + shift in AL
007E  80 C4 30                      ADD AH, '0'
0081  8A D4                         MOV DL, AH      ;handle
                     print (DL = to print)
0083  8A F0                         MOV DH, AL
0085  B4 02                         MOV AH, 02h
0087  CD 21                         INT 21h
0089  8A D6                         MOV DL, DH      ;for so
                    me reason otherwise DL doesnt change
008B  CD 21                         INT 21h
008D  59                        POP CX                ;return
                     initial values
008E  5A                        POP DX
008F  C3                        RET
0090                       print_bcd ENDP
```

```
      0090                  start:
                            ;------------------------------<save  original
r
                            egisters>
      0090  55                    PUSH BP
      0091  8B EC                      MOV BP, SP ;set up the base pointe
                            r to the stack storing the args for this proc
      0093  50                    PUSH AX
      0094  51                    PUSH CX
      0095  52                    PUSH DX
      0096  8C C8                      MOV AX, CS ;set up DS to point to
                            the segment with data items
      0098  8E D8                      MOV DS, AX ;here DS is also CS

     #Microsoft      (R)      Macro      Assembler      Version      5.10
12/2/21 09:32:30
                                                                      Page
1-3



                            ;------------------------------<process    the
int
                            errupt>
      009A  B4 02                      MOV AH, 02H          ;read     real
time
                             from CMOS
      009C  CD 1A                      INT 1Ah         ;returns   CX:DX   =
clock
                            count
      009E  E8 0053 R           CALL read_CMOS

      00A1  5A                    POP DX                ;restore regist
                            ers
      00A2  59                    POP CX
      00A3  58                    POP AX
      00A4  5D                    POP BP

      00A5  B0 20                      MOV AL, 20H          ;these   lines
al
                            low to process lower level
      00A7  E6 20                      OUT 20H, AL          ;interrupts
tha
                            n those we worked with
      00A9  CF                    IRET            ;exit from iter
                            rupt
      00AA                  SUBR_INT ENDP


      00AA                  Main  PROC FAR
      00AA  1E                    PUSH DS         ;write into stack
      00AB  2B C0                      SUB AX, AX      ;write a 0
      00AD  50                    PUSH AX         ;write ax into stack =>
                             stack initialzation
      00AE  B8 ---- R           MOV AX, DATA          ;DataSegment in
                            itialization
      00B1  8E D8                      MOV DS, AX
                            ;-----------<save current vector>
```

```
      00B3  B4 35                   MOV AH, 35H          ;get      curr
vecto
                        r
      00B5  B0 60                   MOV AL, 60H          ;get      curr
vecto
                        r number
      00B7  CD 21                   INT 21H
      00B9  89 1E 0002 R            MOV KEEP_IP, BX ;store the shift
      00BD  8C 06 0000 R            MOV KEEP_CS, ES ;store interrup
                        tion vector segment

                        ;----------<install new interrupt vector>
      00C1  1E                 PUSH DS
      00C2  BA 0000 R          MOV DX, OFFSET SUBR_INT    ;shift fot the
                        proc into DX
      00C5  B8 ---- R          MOV AX, SEG SUBR_INT ;procedure segm
                        ent we save and
      00C8  8E D8                   MOV DS, AX      ;put into DS
      00CA  B4 25                   MOV AH, 25H          ;funtion    to
ins
                        tall new vector, it stores
                                      ;segment and sh
                        ift addresses into interrupt
                                      ;vector with ch

     #Microsoft      (R)      Macro      Assembler      Version      5.10
12/2/21 09:32:30
                                                                Page
1-4


                        osen number.
      00CC  B0 60                   MOV AL, 60H          ;new    vector
num
                        ber
      00CE  CD 21                   INT 21H         ;change          the
interrupt
      00D0  1F                 POP DS

                        ;----------<get  key  scan-code-(let`s  it  be
'Q')
                        >
      00D1                     readkey:
      00D1  B4 00                        MOV AH, 0       ;by pre
                        ssing key in AH a BIOS scancode is stored, and
                        in AL - an ASCII symbol
      00D3  CD 16                        INT 16H         ;interrupt  to
g
                        et the key scancode
      00D5  80 FC 10                     CMP AH, 16      ;16 is
                        a scancode of 'Q'
      00D8  75 F7                        JNE readkey          ;if it`
                        s not 'Q' -> repeat reading, else continue

      00DA  CD 60                        INT 60H         ;call  changed
i
                        nterrupt
```

```
                        ;-----------restore original interrupt vector-
       00DC  FA                 CLI                    ;disabl
                        e interrupts
       00DD  1E                 PUSH DS                ;save ds
       00DE  8B 16 0002 R            MOV DX, KEEP_IP       ;restore
origin
                        al shift
       00E2  A1 0000 R          MOV AX, KEEP_CS        ;restore int ve
                        ctor segment
       00E5  8E D8                    MOV DS, AX
       00E7  B4 25                    MOV AH, 25H                ;to set
                         int vector
       00E9  B0 60                    MOV AL, 60H                ;vector
                         num
       00EB  CD 21                    INT 21H                ;restore
vector
       00ED  1F                 POP DS
       00EE  FB                 STI                    ;enable
                         interrupts
       00EF  B4 4C                    MOV AH, 4CH
       00F1  CD 21                    INT 21H
       00F3                     Main ENDP
       00F3                     CODE ENDS
                                END Main
```

```
     #Microsoft      (R)      Macro      Assembler      Version      5.10
12/2/21 09:32:30
                                                                Symbol
s-1


     Segments and Groups:

                N a m e         Length     Align     Combine Class

     CODE . . . . . .  .  .      00F3  PARA  NONE
     DATA . . . . . .  .  .      0004  PARA  NONE
     STACK  . . . . . .  .  .    0800  PARA  STACK

     Symbols:

                N a m e         Type  Value      Attr

     COLON  . . . . . .  .  .    N PROC     006B  CODE  Length  =
0007

     INT_STACK  . . . . . .  .   L BYTE     0003  CODE  Length  =
0050

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

     MAIN . . . . . .  .  .  .   F PROC     00AA  CODE  Length  =
0049

     PRINT_BCD  . . . . . .  .   N PROC     0072  CODE  Length  =
001E

     READKEY  . . . . . .  .  .  L NEAR     00D1  CODE
     READ_CMOS  . . . . . .  .   N PROC     0053  CODE  Length  =
0018

     START  . . . . . .  .  .  .  L NEAR     0090  CODE
     SUBR_INT . . . . . .  .  .  F PROC     0000  CODE  Length  =
00AA

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


         145 Source  Lines
         145 Total   Lines
          18 Symbols

       48004 + 455158 Bytes symbol space free

           0 Warning Errors
           0 Severe  Errors
```