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(?)
          INIT_SS DW 0000h
          INIT_SP DW 0000h

          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>
          MOV INIT_SP, SP
          PUSH AX
          MOV AX, SS
          MOV INIT_SS, AX
          POP AX
          MOV SP, OFFSET start
          MOV SS, AX
          PUSH AX
          PUSH CX
          PUSH DX

     ;-------------------------------<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
          MOV SP, INIT_SP
          PUSH AX
          MOV AX, INIT_SS
          MOV SS, AX
          POP AX
```

```
          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 0094 R               JMP start
0003  0050[                        INT_STACK DB 80 DUP(?)
       ??
                  ]

0053  0000                         INIT_SS DW 0000h
0055  0000                         INIT_SP DW 0000h

0057                          read_CMOS PROC
```

```
0057  52                        PUSH DX
                                      ;hours
0058  8A C5                          MOV AL, CH      ;in CX=
                        HHMM,
005A  E8 0076 R                   CALL print_bcd
005D  E8 006F R                   CALL colon
                                      ;minutes
0060  8A C1                          MOV AL, CL
0062  E8 0076 R                   CALL print_bcd
```

```
0065  E8 006F R                CALL colon
                                    ;seconds
0068  8A C6                       MOV AL, DH      ;in DH=
                    SS
006A  E8 0076 R                   CALL print_bcd
006D  5A                          POP DX
006E  C3                          RET
006F                         read_CMOS ENDP

006F                         colon PROC
006F  B2 3A                       MOV DL, ':'
0071  B4 02                       MOV AH, 02h
0073  CD 21                       INT 21H
0075  C3                         RET
0076                         colon ENDP

0076                         print_bcd PROC
0076  52                         PUSH DX         ;save initial r
                    egisters
0077  51                         PUSH CX
0078  B1 04                       MOV CL, 4
007A  8A E0                       MOV AH, AL      ;now al
                     = 43 = ah
007C  24 0F                       AND AL, 00001111b     ;now al
                     = 03 (as it is in BCD, each digit is xxxxb)
007E  D2 EC                       SHR AH, CL      ;now ah
                     = 04 => ax = 0403
0080  04 30                       ADD AL, '0'           ;get AS
                    CII value of '0' + shift in AL
0082  80 C4 30                    ADD AH, '0'
0085  8A D4                       MOV DL, AH      ;handle
                     print (DL = to print)
0087  8A F0                       MOV DH, AL
0089  B4 02                       MOV AH, 02h
008B  CD 21                       INT 21h
008D  8A D6                       MOV DL, DH      ;for so
                    me reason otherwise DL doesnt change
008F  CD 21                       INT 21h
0091  59                          POP CX               ;return
                     initial values
0092  5A                         POP DX
0093  C3                         RET
0094                         print_bcd ENDP

0094                         start:
                             ;------------------------------<save  original
r
                             egisters>
0094  2E: 89 26 0055 R            MOV INIT_SP, SP
0099  50                         PUSH AX
009A  8C D0                       MOV AX, SS
009C  2E: A3 0053 R               MOV INIT_SS, AX
```

```
00A0  58                  POP AX
00A1  BC 0094 R           MOV SP, OFFSET start
00A4  8E D0                    MOV SS, AX
```

```
00A6  50                PUSH AX
00A7  51                PUSH CX
00A8  52                PUSH DX

                        ;------------------------------<process the int
                        errupt>
00A9  B4 02                     MOV AH, 02H          ;read real time
                         from CMOS
00AB  CD 1A                     INT 1Ah         ;returns CX:DX = clock
                        count
00AD  E8 0057 R           CALL read_CMOS

00B0  5A                  POP DX                ;restore regist
                        ers
00B1  59                  POP CX
00B2  58                  POP AX
00B3  2E: 8B 26 0055 R              MOV SP, INIT_SP
00B8  50                  PUSH AX
00B9  2E: A1 0053 R       MOV AX, INIT_SS
00BD  8E D0               MOV SS, AX
00BF  58                  POP AX

00C0  B0 20                     MOV AL, 20H          ;these lines al
                        low to process lower level
00C2  E6 20                     OUT 20H, AL          ;interrupts tha
                        n those we worked with
00C4  CF                  IRET            ;exit from iter
                        rupt
00C5                    SUBR_INT ENDP


00C5                    Main  PROC FAR
00C5  1E                  PUSH DS         ;write into stack
00C6  2B C0                     SUB AX, AX      ;write a 0
00C8  50                  PUSH AX         ;write ax into stack =>
                         stack initialzation
00C9  B8 ---- R           MOV AX, DATA          ;DataSegment in
                        itialization
00CC  8E D8                     MOV DS, AX
                        ;-----------<save current vector>
00CE  B4 35                     MOV AH, 35H          ;get curr vecto
                        r
00D0  B0 60                     MOV AL, 60H          ;get curr vecto
                        r number
00D2  CD 21                     INT 21H
```

```
00D4  89 1E 0002 R            MOV KEEP_IP, BX ;store the shift
00D8  8C 06 0000 R            MOV KEEP_CS, ES ;store interrup
                    tion vector segment

                    ;----------<install new interrupt vector>
00DC  1E                 PUSH DS
00DD  BA 0000 R          MOV DX, OFFSET SUBR_INT    ;shift fot the
                    proc into DX
00E0  B8 ---- R          MOV AX, SEG SUBR_INT  ;procedure segm
```

```
                        ent we save and
00E3  8E D8                     MOV DS, AX      ;put into DS
00E5  B4 25                     MOV AH, 25H          ;funtion    to
ins
                        tall new vector, it stores
                                        ;segment and sh
                        ift addresses into interrupt
                                        ;vector with ch
                        osen number.
00E7  B0 60                     MOV AL, 60H          ;new    vector
num
                        ber
00E9  CD 21                     INT 21H         ;change         the
interrupt
00EB  1F                  POP DS

                        ;----------<get  key  scan-code-(let`s  it  be
'Q')
                        >
00EC                      readkey:
00EC  B4 00                             MOV AH, 0       ;by pre
                        ssing key in AH a BIOS scancode is stored, and
                        in AL - an ASCII symbol
00EE  CD 16                             INT 16H         ;interrupt  to
g
                        et the key scancode
00F0  80 FC 10                          CMP AH, 16      ;16 is
                        a scancode of 'Q'
00F3  75 F7                             JNE readkey          ;if it`
                        s not 'Q' -> repeat reading, else continue

00F5  CD 60                             INT 60H         ;call  changed
i
                        nterrupt

                        ;-----------restore original interrupt vector-
00F7  FA                  CLI                   ;disabl
                        e interrupts
00F8  1E                  PUSH DS               ;save ds
00F9  8B 16 0002 R              MOV DX, KEEP_IP       ;restore
origin
                        al shift
00FD  A1 0000 R           MOV AX, KEEP_CS       ;restore int ve
                        ctor segment
0100  8E D8                     MOV DS, AX
0102  B4 25                     MOV AH, 25H               ;to set
                         int vector
0104  B0 60                     MOV AL, 60H               ;vector
                         num
0106  CD 21                     INT 21H               ;restore
vector
0108  1F                  POP DS
```

```
0109  FB                    STI                     ;enable
                          interrupts
010A  B4 4C                       MOV AH, 4CH
010C  CD 21                       INT 21H
010E                    Main ENDP
010E                    CODE ENDS
                        END Main
```

```
     #Microsoft      (R)      Macro      Assembler      Version      5.10
12/2/21 15:00:27
                                                                Symbol
s-1


     Segments and Groups:

                N a m e         Length     Align    Combine Class

     CODE . . . . . .  .  .      010E  PARA  NONE
     DATA . . . . . .  .  .      0004  PARA  NONE
     STACK  . . . . . .  .  .    0800  PARA  STACK

     Symbols:

                N a m e         Type  Value     Attr

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

     INIT_SP  . . . . . .  .  .  L WORD    0055  CODE
     INIT_SS  . . . . . .  .  .  L WORD    0053  CODE
     INT_STACK  . . . . . .  .  .  L BYTE    0003  CODE  Length  =
0050

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

     MAIN . . . . . .  .  .  .   F PROC    00C5  CODE  Length  =
0049

     PRINT_BCD  . . . . . .  .  .  N PROC    0076  CODE  Length  =
001E

     READKEY  . . . . . .  .  .  L NEAR    00EC  CODE
     READ_CMOS  . . . . . .  .  .  N PROC    0057  CODE  Length  =
0018

     START  . . . . . .  .  .  .  L NEAR    0094  CODE
     SUBR_INT . . . . . .  .  .  F PROC    0000  CODE  Length  =
00C5

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


         154 Source  Lines
         154 Total   Lines
          20 Symbols

       48004 + 455158 Bytes symbol space free

           0 Warning Errors
           0 Severe  Errors
```