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 11

Студент гр.0382 _____ Литягин С.М.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург
2021

Цель работы.

Написать собственное прерывание, согласно заданию.

Задание.

Прерывание - это процесс вызова процедур для выполнения некоторой задачи, обычно связанной с обслуживанием некоторых устройств (обработка сигнала таймера, нажатия клавиши и т.д.). Написать собственное прерывание согласно варианту

Вариант №11:

2d

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

d – Выполнить чтение и вывод на экран отсчета системных часов (в тиках, где 1 тик = 55 мсек).

Выполнение работы:

В сегменте данных DATA содержаться переменные: KEEP_CS, KEEP_IP для хранения сегмента и смещения старого прерывания соответственно.

Процедура пользовательского прерывания OutInt. Здесь выделяем отдельный стек для прерывания, не забывая сохранить смещение на изначальный. Далее сохраняем все изменяемые регистры в стеке. Помещаем в регистр AH значение 00h (функция получения времени в тиках прерывания 1Ah), вызываем прерывание 1Ah. В регистры CX(записывается старшая часть), DX записывается время. Для вывода времени написана процедура IntToStr (в ней берутся остатки от деления числа, помещенного в регистр AX, на 10, они переводятся в символы и выводятся с помощью функции 02h прерывания 21h на дисплей. Перед ее вызовом помещаем в регистр AX значение, что должно быть выведено. Таким образом, выводим CX и DX. Восстанавливаем регистры из цикла. Выходим из прерывания.

В процедуре Main запоминаем смещение и сегмент текущего 60h прерывания в KEEP_IP, KEEP_CS с помощью функции 35h прерывания 21h.

Используя же функцию 25h прерывания 21, устанавливаем вектор прерывания 60h на наше прерывание OutInt. Затем происходит его вызов. Когда его работа будет завершена – восстанавливаем старый вектор прерывания.

Исходный код программы см. в приложении А.

Тестирование:

Для проверки работоспособности были проведены тесты, результаты представлены в таблице 1.

Таблица 1 – Тестирование и результаты

Номер теста	Входные данные	Выходные данные	Верность результата
1	нет		верно
2	нет		верно

Выводы.

В ходе работы были изучены прерывания. Также было написано собственное прерывание по чтению и выводу системного времени на экран.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: lb5.asm

```
AStack SEGMENT STACK
 DW 30 DUP(?)
AStack ENDS

DATA SEGMENT
        KEEP_CS DW 0 ; для хранения сегмента
        KEEP_IP DW 0 ; и смещения вектора прерывания
DATA ENDS


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


IntToStr PROC
    push AX ; сохраняем регистры, что будем использовать
    push DX
    push BX
    push CX

    xor CX, CX ; обнуляем CX для хранения кол-ва символов
    mov BX, 10 ; делитель для 10 с.с.
lp:
    xor DX,DX
    div BX ; деление AX = (DX, AX)/BX, остаток в DX
    add DL, '0' ; перевод цифры в символ
    push DX ; сохраяем остаток в стек
    inc CX ; увеличиваем счетчик
    test AX, AX ; проверка AX
    jnz lp ; если частное не 0, то повторяем
mov ah, 02h
lp2:
    pop DX ; достаем символ из стека
    int 21h
    loop lp2 ; пока cx не 0 выполняется переход

    pop CX ; возвращаем значения из стека
    pop BX
    pop DX
    pop AX
    ret
IntToStr endp


OutInt PROC FAR
        jmp handle
     KEEP_SS DW 0
     KEEP_SP DW 0
     IStack DB 50 dup(" ")
```

```
handle:
      mov KEEP_SP, SP
      mov KEEP_SS, SS
      mov SP, SEG IStack
      mov SS, SP
      mov SP, offset handle

      push AX    ; сохранение изменяемых регистров
      push CX
      push DX

      mov AH, 00h
      int 1Ah

      mov AX, CX
      call IntToStr
      mov AX, DX
      call IntToStr

      pop DX
      pop CX
      pop AX   ; восстановление регистров

      mov SS, KEEP_SS
      mov SP, KEEP_SP

      mov AL, 20H
      out  20H,AL
      iret
OutInt ENDP


Main  PROC  FAR
      push DS
      sub AX,AX
      push AX
      mov AX, DATA
      mov DS, AX

      mov AH,35h ; возвращение текущего значения вектора прерывания
      mov AL,60h ; номер вектора
      int 21h
      mov KEEP_IP, BX ; запоминание смещения
      mov KEEP_CS, ES ; запоминание сегмента

      push DS
      mov DX, offset OutInt; смещение для процедуры
      mov AX, seg OutInt ; сегмент процедуры
      mov DS, AX
      mov AH, 25h ; функция установки вектора
      mov AL, 60h ; номер вектора
      int 21h ; устанавливаем вектор прерывания на указанный адрес нового
обработчика
      pop DS

      int 60h ; вызываем прерывание пользователя
```

```
      CLI
      push DS
      mov DX, KEEP_IP
      mov AX, KEEP_CS
      mov DS, AX
      mov AH, 25h
      mov AL, 60h
      int 21h
      pop DS
      STI

      ret
Main ENDP
CODE ENDS
      END Main
```

Название файла: lb5.lst

```
Microsoft (R) Macro Assembler Version 5.10                  12/9/21
02:00:16
                                                             Page     1-1


       0000                    AStack SEGMENT STACK
       0000  001E[                    DW 30 DUP(?)
              ????
                          ]

       003C                    AStack ENDS

       0000                    DATA SEGMENT
       0000  0000                      KEEP_CS    DW    0    ;    PrP»CЦ
C…CЂP°PSPµPSPёCЦ
                               CЃPµPiPjPµPSC,P°
       0002  0000                      KEEP_IP     DW     0     ;     Pё
CЃPjPµC‰PµPSPёCЦ PIPµ
                               PєC,PsCЂP° PїCЂPµCЂC‹PIP°PSPёCЦ
       0004                    DATA ENDS


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


       0000                    IntToStr PROC
       0000  50                    push      AX      ;      CЃPsC…CЂP°PSCЦPµPj
CЂPµPiPёCЃC,CЂ
                               C‹, C‡C,Ps P±CґPrPµPj PёCЃPїPsP»CЊP·PsPIP°C,CЊ
       0001  52                    push DX
       0002  53                    push BX
       0003  51                    push CX

       0004  33 C9                       xor  CX,  CX  ;  PsP±PSCґP»CЦPµPj  CX
PrP»CЦ C…C
                               ЂP°PSPµPSPёCЦ PєPsP»-PIP° CЃPёPjPIPsP»PsPI
```

```
      0006  BB 000A                 mov BX, 10 ; PrPµP»PëC,PµP»CЊ PrP»CЏ
10 CЃ.
                        CЃ.
      0009                      lp:
      0009  33 D2                     xor DX,DX
      000B  F7 F3                     div  BX  ;  PrPµP»PµPSPëPµ  AX  =  (DX,
AX)/BX, P
                        sCЃC,P°C,PsPє PI DX
      000D  80 C2 30                  add   DL,   '0'   ;   PïPµCЂPµPIPsPr
C†PëC„CЂC‹ PI
                        CЃPëPjPIPsP»
      0010  52                      push DX ; CЃPsC…CЂP°CЏPµPj PsCЃC,P°C,PsPє
P
                        I CЃC,PµPє
      0011  41                      inc     CX     ;     CґPIPµP»PëC‡PëPIP°PµPj
CЃC‡PµC,C‡P
                        ëPє
      0012  85 C0                     test AX, AX ; PïCЂPsPIPµCЂPєP° AX
      0014  75 F3                     jnz  lp  ;  PµCЃP»Pë  C‡P°CЃC,PSPsPµ
PSPµ 0, C,
                        Ps PïPsPIC,PsCЂCЏPµPj
      0016  B4 02                    mov ah, 02h
      0018                      lp2:
      0018  5A                      pop   DX   ;   PrPsCЃC,P°PµPj   CЃPëPjPIPsP»
PëP· C
                        ЃC,PµPєP°
      0019  CD 21                     int 21h
      001B  E2 FB                     loop  lp2  ;  PïPsPєP°  cx  PSPµ  0
PIC‹ PïPsP»PSC
                        ЏPµC,CЃCЏ PïPµCЂPµC…PsPr

      001D  59                      pop     CX     ;     PIPsP·PICЂP°C‰P°PµPj
P·PSP°C‡PµPSP
                        ëCЏ PëP· CЃC,PµPєP°
      001E  5B                      pop BX
Microsoft (R) Macro Assembler Version 5.10                      12/9/21
02:00:16
                                                              Page     1-2


      001F  5A                      pop DX
      0020  58                      pop AX
      0021  C3                      ret
      0022                      IntToStr endp


      0022                      OutInt PROC FAR
      0022  EB 37 90                  jmp handle
      0025  0000                    KEEP_SS DW 0
      0027  0000                    KEEP_SP DW 0
      0029  0032[                   IStack DB 50 dup(" ")
             20
                    ]

      005B                      handle:
      005B  2E: 89 26 0027 R          mov KEEP_SP, SP
```

```
0060  2E: 8C 16 0025 R               mov KEEP_SS, SS
0065  BC ---- R           mov SP, SEG IStack
0068  8E D4                    mov SS, SP
006A  BC 005B R           mov SP, offset handle

006D  50                  push AX    ; CЃPsC…CЂP°PSPµPSPёPµ PёP·P
                       jPµPSCЏPµPjC‹C… CЂPµPiPёCЃC,CЂPsPI
006E  51                  push CX
006F  52                  push DX

0070  B4 00                    mov AH, 00h
0072  CD 1A                    int 1Ah

0074  8B C1                    mov AX, CX
0076  E8 0000 R           call IntToStr
0079  8B C2                    mov AX, DX
007B  E8 0000 R           call IntToStr

007E  5A                  pop DX
007F  59                  pop CX
0080  58                  pop AX   ; PIPsCЃCЃC,P°PSPsPIP»PµPSPёPµ
                        CЂPµPiPёCЃC,CЂPsPI

0081  2E: 8E 16 0025 R               mov SS, KEEP_SS
0086  2E: 8B 26 0027 R               mov SP, KEEP_SP

008B  B0 20                    mov AL, 20H
008D  E6 20                    out  20H,AL
008F  CF                  iret
0090                   OutInt ENDP


0090                   Main  PROC  FAR
0090  1E                  push DS
0091  2B C0                    sub AX,AX
0093  50                  push AX
0094  B8 ---- R           mov AX, DATA
0097  8E D8                    mov DS, AX
□Microsoft (R) Macro Assembler Version 5.10                    12/9/21
02:00:16
                                                            Page     1-3


0099  B4 35                    mov AH,35h ; PIPsP·PICЂP°C‰PµPSPёPµ
C,P
                       µPєCѓC‰PµPiPs  P·PSP°C‡PµPSPёCЏ  PIPµPєC,PsCЂP°
P
                       ïCЂPµCЂC‹PIP°PSPёCЏ
009B  B0 60                    mov     AL,60h     ;     PSPsPjPµCЂ
PIPµPєC,PsCЂP°
009D  CD 21                    int 21h
009F  89 1E 0002 R             mov       KEEP_IP,       BX       ;
P·P°PїPsPjPёPSP°PSPёP
                       µ CЃPjPµC‰PµPSPёCЏ
```

```
      00A3  8C 06 0000 R            mov       KEEP_CS,       ES       ;
P·P°PiPsPjPëPSP°PSPëP
                           µ CЃPµPiPjPµPSC,P°

      00A7  1E                      push DS
      00A8  BA 0022 R               mov DX, offset OutInt; CЃPjPµC‰PµPSPëPµ
                            PrP»CЦ PiCЂPsC†PµPrCѓCЂC‹
      00AB  B8 ---- R               mov AX, seg OutInt ; CЃPµPiPjPµPSC, PiC
                           ЂPsC†PµPrCѓCЂC‹
      00AE  8E D8                   mov DS, AX
      00B0  B4 25                   mov   AH,   25h   ;   C„CѓPSPєC†PëCЦ
CѓCЃC,P°PS
                           PsPIPєPë PIPµPєC,PsCЂP°
      00B2  B0 60                   mov    AL,    60h    ;    PSPsPjPµCЂ
PIPµPєC,PsCЂP°
      00B4  CD 21                   int              21h             ;
CѓCЃC,P°PSP°PIP»PëPIP°PµPj PI
                          PµPєC,PsCЂ PiCЂPµCЂC‹ PIP°PSPëCЦ PSP° CѓPєP°P·P°
                           PSPSC‹P№        P°PrCЂPµCЃ        PSPsPIPsPiPs
PsP±CЂP°P±PsC,
                           C‡PëPєP°
      00B6  1F                      pop DS

      00B7  CD 60                   int    60h    ;    PIC‹P·C‹PIP°PµPj
PiCЂPµCЂC‹PI
                           P°PSPëPµ PiPsP»CЊP·PsPIP°C,PµP»CЦ

      00B9  FA                      CLI
      00BA  1E                      push DS
      00BB  8B 16 0002 R            mov DX, KEEP_IP
      00BF  A1 0000 R               mov AX, KEEP_CS
      00C2  8E D8                   mov DS, AX
      00C4  B4 25                   mov AH, 25h
      00C6  B0 60                   mov AL, 60h
      00C8  CD 21                   int 21h
      00CA  1F                      pop DS
      00CB  FB                      STI

      00CC  CB                      ret
      00CD                  Main ENDP
      00CD                  CODE ENDS
                                    END Main
Microsoft (R) Macro Assembler Version 5.10                    12/9/21
02:00:16

Symbols-1


     Segments and Groups:

                N a m e         Length     Align     Combine Class

     ASTACK . . . . . .          003C  PARA  STACK
     CODE . . . . . .            00CD  PARA  NONE
     DATA . . . . . .            0004  PARA  NONE
```

```
Symbols:

                N a m e         Type  Value     Attr

HANDLE . . . . . .              L NEAR    005B  CODE

INTTOSTR . . . . . .            N PROC    0000  CODE  Length =
0022
ISTACK . . . . . .              L BYTE    0029  CODE  Length =
0032

KEEP_CS  . . . . . .            L WORD    0000  DATA
KEEP_IP  . . . . . .            L WORD    0002  DATA
KEEP_SP  . . . . . .            L WORD    0027  CODE
KEEP_SS  . . . . . .            L WORD    0025  CODE

LP . . . . . .                  L NEAR    0009  CODE
LP2  . . . . . .                L NEAR    0018  CODE

MAIN . . . . . .                F PROC    0090  CODE  Length =
003D

OUTINT . . . . . .              F PROC    0022  CODE  Length =
006E

@CPU . . . . . .                TEXT  0101h
@FILENAME  . . . . . .          TEXT  lb5
@VERSION . . . . . .            TEXT  510


    120 Source  Lines
    120 Total   Lines
     19 Symbols

  48030 + 459230 Bytes symbol space free

      0 Warning Errors
 0 Severe  Errors
```