**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 _____ Козлов Т.В.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург

2021

**Цель работы.**

Написать программу обработки прерывания.

**Ход работы.**

Вариант 1d:

1 - 08h - прерывание от системного таймера - генерируется автоматическиоперационной системой 18 раз в сек;

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

Для хранения сегмента и смещения прерывания были созданы переменные KEEP_CS и KEEP_IP (обе по 2 байта). Функция 35H прерывания 21H возвращает текущее значение вектора прерывания (в данном случае 08H), и его смещение и сегмент заносятся в вышеупомянутые переменные для их последующего восстановления.

После этого с помощью функции 25H прерывания 21H устанавливается свое прерывание (процедура SUBR_INT) посредством помещения смещения в DX, а сегмента в DS.

Далее с помощью функции 25H прерывания 21H восстанавливается старое прерывание.

Реализация собственного прерывания:

Для вывода десятичных чисел на экран была написана функция OutInt, которая выводит десятичные числа, находящиеся в AX (см. комментарии в коде).

Для получения отчета системных часов в тиках используется функция 00H прерывания 1AH, которая помещает в регистры CX и DX время в тиках (CX – старшее значение). Далее эти значения помещаются в AX, и для каждого из них вызывается функция вывода на экран.

Табл.1: Тестирование программы main.asm

| Входные данные | Результирующая строка | Комментарий |
|---|---|---|
| Отсутствуют | 66786 | Программа работает корректно |
| Отсутствуют (запуск спустя примерно секунду) | 66796 | Программа работает корректно |

Код программы см. в приложении А.

**Выводы.**

В ходе выполнения работы была изучена работа с прерываниями на языке Ассемблер. Написано собственное прерывание.

## ПРИЛОЖЕНИЕ А

**main.asm:**

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

STACK    SEGMENT  STACK
         DW 1024 DUP(?)
STACK    ENDS

DATA SEGMENT
        KEEP_CS DW 0 ; для хранения сегмента
        KEEP_IP DW 0 ; и смещения вектора прерывания
                NUM DW 0
                MESSAGE DB 2 DUP(?)

DATA ENDS


CODE      SEGMENT

OutInt PROC
         push DX
         push CX

    xor     cx, cx ; cx - количество цифр
    mov     bx, 10 ; основание сс. 10 для десятеричной и т.п.
oi2:
    xor     dx,dx
    div     bx ; делим число на основание сс и сохраняем остаток в стеке
    push    dx
    inc     cx; увеличиваем количесвто цифр в cx
```

```
    test    ax, ax ; проверка на 0
    jnz     oi2
; Вывод
    mov     ah, 02h
oi3:
    pop     dx
    add     dl, '0' ; перевод цифры в символ
    int     21h
; Повторим ровно столько раз, сколько цифр насчитали.
    loop    oi3 ; пока cx не 0 выполняется переход

        POP CX
        POP DX
    ret

OutInt endp


SUBR_INT PROC FAR
       JMP start_proc

              save_SP DW 0000h
              save_SS DW 0000h
              INT_STACK DB 40 DUP(0)
start_proc:

    MOV save_SP, SP
        MOV save_SS, SS
        MOV SP, SEG INT_STACK
        MOV SS, SP
        MOV SP, offset start_proc
```

```
        PUSH AX    ; сохранение изменяемых регистров
        PUSH CX
        PUSH DX

        mov AH, 00H
        int 1AH

        mov AX, CX
        call OutInt
        mov AX, DX
        call OutInt

        POP  DX
        POP  CX
        POP  AX   ; восстановление регистров
        MOV  SS, save_SS
        MOV  SP, save_SP
        MOV  AL, 20H

        OUT  20H,AL

        iret

SUBR_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


; Запоминание текущего вектора прерывания
MOV  AH, 35H   ; функция получения вектора
MOV  AL, 08H   ; номер вектора
INT  21H
MOV  KEEP_IP, BX  ; запоминание смещения
MOV  KEEP_CS, ES  ; и сегмента

; Установка вектора прерывания
PUSH DS
MOV  DX, OFFSET SUBR_INT ; смещение для процедуры в DX
MOV  AX, SEG SUBR_INT    ; сегмент процедуры
MOV  DS, AX          ; помещаем в DS
MOV  AH, 25H         ; функция установки вектора
MOV  AL, 08H         ; номер вектора
INT  21H             ; меняем прерывание
POP  DS

int 08H; на всякий вывод в консоль отдельно от отладчика

; Восстановление изначального вектора прерывания (можно
закомментить)
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

        MOV AH, 4Ch
        INT 21h
Main    ENDP
CODE ENDS
        END Main
```

**Main.lst**

```
Microsoft (R) Macro Assembler Version 5.10              12/2/21 12:28:07
                                                        Page     1-1


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

0000                    STACK   SEGMENT  STACK
0000  0400[                     DW 1024 DUP(?)
        ????
                    ]

0800                    STACK   ENDS

0000                    DATA SEGMENT
0000  0000                  KEEP_CS DW 0 ; для хранения
                        сегмента
0002  0000                  KEEP_IP DW 0 ; и смещения ве
                        ктора прерывания
0004  0000                      NUM DW 0
```

```
0006  0002[                         MESSAGE DB 2 DUP(?)
        ??
                    ]


0008                DATA ENDS


0000                CODE    SEGMENT

0000                OutInt PROC
0000  52                   push DX
0001  51                   push CX

0002  33 C9                   xor   cx, cx ; cx - количество
                    цифр
0004  BB 000A                 mov   bx, 10 ; основание сс. 1
                    0 для десятеричной и т.п.
0007                oi2:
0007  33 D2                   xor   dx,dx
0009  F7 F3                   div   bx ; делим число на оэ
                    Онование сс и сохраняем ос
                    таток в стеке
000B  52                    push  dx
000C  41                    inc   cx; увеличиваем коли
                    чесвто цифр в cx

000D  85 C0                   test  ax, ax ; проверка на 0
000F  75 F6                   jnz   oi2
                    ; Вывод
```

```
0011  B4 02                 mov     ah, 02h
0013                  oi3:
0013  5A                    pop     dx
0014  80 C2 30              add     dl, '0' ; перевод цифры
                            в символ
0017  CD 21                 int     21h
                      ; Повторим ровно столько р
                      аз, сколько цифр насчиталЙ
                      ,.
```

```
0019  E2 F8                    loop    oi3 ; пока cx не 0 выпол
                              няется переход

001B  59                      POP CX
001C  5A                      POP DX
001D  C3                    ret

001E                        OutInt endp


001E                        SUBR_INT PROC FAR
001E  EB 2D 90                      JMP start_proc

0021  0000                          save_SP DW 0000h
0023  0000                          save_SS DW 0000h
0025  0028[                             INT_STACK DB 40 DUP(0)
        00
                    ]

004D                        start_proc:

004D  2E: 89 26 0021 R          MOV save_SP, SP
0052  2E: 8C 16 0023 R              MOV save_SS, SS
0057  BC ---- R                 MOV SP, SEG INT_STACK
005A  8E D4                         MOV SS, SP
005C  BC 004D R                     MOV SP, offset start_proc
005F  50                        PUSH AX    ; сохранение изЙ
```

¼еняемых регистров

```
0060  51                  PUSH CX
0061  52                  PUSH DX

0062  B4 00                    mov AH, 00H
0064  CD 1A                    int 1AH

0066  8B C1                    mov AX, CX
0068  E8 0000 R           call OutInt
006B  8B C2                    mov AX, DX
006D  E8 0000 R           call OutInt

0070  5A                  POP  DX
0071  59                  POP  CX
0072  58                  POP  AX   ; восстановлениЙ
```

µ регистров

```
0073  2E: 8E 16 0023 R          MOV  SS, save_SS
0078  2E: 8B 26 0021 R          MOV  SP, save_SP
007D  B0 20                     MOV  AL, 20H

007F  E6 20                     OUT  20H,AL

0081  CF                  iret

0082                SUBR_INT ENDP
```

```
0082                    Main  PROC  FAR
0082  1E                      push  DS      ;\  Сохранение
                        адреса начала PSP в стеке
0083  2B C0                        sub   AX,AX    ; > для последэ
                        )ющего восстановления по
0085  50                      push  AX      ;/  команде ret,
                        завершающей процедуру.
0086  B8 ---- R               mov   AX,DATA          ; ЗагруЙ
                        ·ка сегментного
0089  8E D8                        mov   DS,AX


                              ; Запоминание текущег
                        о вектора прерывания
008B  B4 35                        MOV  AH, 35H   ; функция полэ
                        )чения вектора
008D  B0 08                        MOV  AL, 08H   ; номер вектоэ
                        a
008F  CD 21                        INT  21H
0091  89 1E 0002 R                 MOV  KEEP_IP, BX  ; запоминанЙ
                        ,е смещения
0095  8C 06 0000 R                 MOV  KEEP_CS, ES  ; и сегмента

                              ; Установка вектора пэ
                        ерывания
0099  1E                      PUSH DS
```

```
009A  BA 001E R                MOV  DX, OFFSET SUBR_INT ; смещен
                               ие для процедуры в DX
009D  B8 ---- R                MOV  AX, SEG SUBR_INT    ; сегмен
                               т процедуры
00A0  8E D8                    MOV  DS, AX         ; помещаем
                                в DS
00A2  B4 25                    MOV  AH, 25H        ; функция э
                               )становки вектора
00A4  B0 08                    MOV  AL, 08H        ; номер веЙ
                               °тора
00A6  CD 21                    INT  21H            ; меняем пэ
                               ерывание
00A8  1F                       POP  DS

00A9  CD 08                    int 08H; на всякий вывод Й
                               ² консоль отдельно от отла
                               дчика

                               ; Восстановление изна
                               чального вектора прерываЙ
                               ½ия (можно закомментить)
00AB  FA                       CLI
00AC  1E                       PUSH DS
00AD  8B 16 0002 R             MOV  DX, KEEP_IP
00B1  A1 0000 R                MOV  AX, KEEP_CS
00B4  8E D8                    MOV  DS, AX
00B6  B4 25                    MOV  AH, 25H
00B8  B0 08                    MOV  AL, 08H
```

```
00BA  CD 21                     INT  21H        ; восстанавЙ
                        »иваем вектор
00BC  1F                        POP  DS
00BD  FB                        STI

00BE  B4 4C                     MOV AH, 4Ch
00C0  CD 21                     INT 21h
00C2                    Main    ENDP
00C2                    CODE ENDS
                        END Main
```

Segments and Groups:

| N a m e | Length | Align | Combine | Class |
|---|---|---|---|---|
| CODE . . . . . | 00C2 | PARA | NONE | |
| DATA . . . . . | 0008 | PARA | NONE | |
| STACK . . . . . | 0800 | PARA | STACK | |

Symbols:

| N a m e | Type | Value | Attr | |
|---|---|---|---|---|
| INT_STACK . . . . . | L BYTE | 0025 | CODE | Length = 0028 |
| KEEP_CS . . . . . | L WORD | 0000 | DATA | |
| KEEP_IP . . . . . | L WORD | 0002 | DATA | |
| MAIN . . . . . | F PROC | 0082 | CODE | Length = 0040 |
| MESSAGE . . . . . | L BYTE | 0006 | DATA | Length = 0002 |
| NUM . . . . . | L WORD | 0004 | DATA | |
| OI2 . . . . . | L NEAR | 0007 | CODE | |
| OI3 . . . . . | L NEAR | 0013 | CODE | |
| OUTINT . . . . . | N PROC | 0000 | CODE | Length = 001E |
| SAVE_SP . . . . . | L WORD | 0021 | CODE | |

```
SAVE_SS . . . . . .      L WORD    0023  CODE
START_PROC . . . . . .   L NEAR    004D CODE
SUBR_INT . . . . . .     F PROC    001E  CODE       Length = 0064

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


   130 Source  Lines
   130 Total   Lines
    21 Symbols

 48020 + 457190 Bytes symbol space free

     0 Warning Errors
     0 Severe  Errors
```