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 _____ Пустовалова Е.М.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург
2021

Цель работы.

Написать собственное прерывание.

Задание.

Вариант 11.

Цифра задает номер и назначение заменяемого вектора прерывания:

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

Буква определяет действия, реализуемые программой обработки прерываний:

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

Выполнение работы.

Для хранения сегмента и смещения прерывания были созданы переменные KEEP_CS и KEEP_IP. Функция 35h прерывания 21h возвращает текущее значение вектора прерывания (в варианте лабораторной работы - 60h), и его смещение и сегмент заносятся в переменные KEEP_CS и KEEP_IP для дальнейшего восстановления. После этого с помощью функции 25h прерывания 21h устанавливается свое прерывание (процедура SUBR_INT) путем помещения смещения в DX, сегмента в DS. Потом с помощью функции 25h прерывания 21h восстанавливается старое прерывание.

Для вывода десятичных чисел на экран была написана функция GetInt, которая выводит десятичные числа, хранящиеся в AX. 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.

Таблица 1 – Проверка работы программы.

№	Входные данные	Результат	Комментарии
1	Отсутствуют	1327072	Верно

2	Отсутствуют (запуск примерно спустя 1 секунду после первого запуска)	1327251	Верно

Исходный код программы находится в приложении А.

Выводы.

В ходе выполнения данной лабораторной работы был изучен механизм написания собственного прерывания.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: lab5.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

GetInt PROC
    push DX
    push CX

    xor     cx, cx ; cx - количество цифр
    mov     bx, 10 ; основание системы счисления
gi2:
    xor     dx,dx
    div     bx ; деление числа на основание cc и сохранение остатка в
стеке
    push    dx
    inc     cx;
    test    ax, ax ; проверка на 0
    jnz     gi2
; Вывод
    mov     ah, 02h
gi3:
    pop     dx
    add     dl, '0' ; перевод цифры в символ
    int     21h
    loop    gi3 ; переход, пока cx != 0
```

```
    POP CX
    POP DX
    ret

GetInt 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 GetInt
    mov AX, DX
    call GetInt

    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               ; 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 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, 60H ; номер вектора
        INT 21H ; меняем прерывание
        POP DS

        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
```

Название файла: lab5.lst

```
_Microsoft (R) Macro Assembler Version 5.10                 12/16/21
13:28:3
                                                            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 ; -¥-ª—è —Ö—Ä-∞-Ω-µ-Ω-∏—è
                        —Å-µ-≥-°-µ-Ω—Ç-∞
0002  0000                      KEEP_IP DW 0 ; -∏ —Å-°-µ—â-µ-Ω-∏—è -≤-µ
                        -∫—Ç-æ—Ä-∞ -ø—Ä-µ—Ä-ã-≤-∞-Ω-∏—è
0004  0000                      NUM DW 0
0006  0002[                          MESSAGE DB 2 DUP(?)
        ??
                    ]


0008                    DATA ENDS


0000                    CODE      SEGMENT

0000                    GetInt PROC
0000  52                   push DX
0001  51                   push CX

0002  33 C9                        xor     cx, cx ; cx - -∫-æ-ª-∏—á-µ—Å-Ç-
≤-æ
                        —Ü-∏—Ñ—Ä
```

```
        0004  BB 000A                   mov     bx, 10 ; -æ—Å-Ω-æ-≤-∞-Ω-∏-µ —Å-
∏—Å—
                                  Ç-µ-°—ã —Å—á-∏—Å-ª-µ-Ω-∏—è
        0007                      gi2:
        0007  33 D2                       xor     dx,dx
        0009  F7 F3                       div     bx ; -¥-µ-ª-µ-Ω-∏-µ —á-∏—Å-ª-∞ -
Ω-∞
                                   -æ—Å-Ω-æ-≤-∞-Ω-∏-µ —Å—Å -∏ —Å-æ—Ö—Ä-∞-Ω-µ-Ω-∏-
                                  µ -æ—Å—Ç-∞—Ç-∫-∞ -≤ —Å—Ç-µ-∫-µ
        000B  52                      push    dx
        000C  41                      inc     cx;
        000D  85 C0                       test    ax, ax ; -ø—Ä-æ-≤-µ—Ä-∫-∞ -Ω-∞ 0
        000F  75 F6                       jnz     gi2
                                  ; -í—ã-≤-æ-¥
        0011  B4 02                       mov     ah, 02h
        0013                      gi3:
        0013  5A                      pop     dx
        0014  80 C2 30                    add     dl, '0' ; -ø-µ—Ä-µ-≤-æ-¥ —Ü-∏—Ñ—
Ä—ã
                                   -≤ —Å-∏-°-≤-æ-ª
        0017  CD 21                       int     21h
        0019  E2 F8                       loop    gi3 ; -ø-µ—Ä-µ—Ö-æ-¥, -ø-æ-∫-∞ —
Å—Ö
                                   != 0

        001B  59                      POP CX
        001C  5A                      POP DX
```

```
001D  C3                  ret

001E                  GetInt 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    ; ─Å-æ─Ö─Ä-∞-Ω-µ-Ω-∏-µ -∏-∑-°-µ-
                        Ω─è-µ-°─ã─Ö ─Ä-µ-≥-∏─Å─Ç─Ä-æ-≤
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 GetInt
006B  8B C2                    mov AX, DX
006D  E8 0000 R           call GetInt

0070  5A                  POP  DX
0071  59                  POP  CX
0072  58                  POP  AX   ; -≤-æ─Å─Å─Ç-∞-Ω-æ-≤-ª-µ-Ω-∏-µ ─Ä
```

```
                        -µ-≥-∏—Å—Ç—Ä-æ-≤
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        ;\  -°-æ—Ö—Ä-∞-Ω-µ-Ω-∏-µ
                       -∞-¥—Ä-µ—Å-∞ -Ω-∞—á-∞-ª-∞ PSP -≤ —Å—Ç-µ-∫-µ
0083  2B C0                        sub   AX,AX     ; > -¥-ª—è -ø-æ—Å-ª-
µ-¥—
```

É—é—â-µ-≥-æ -≤-æ—Å—Å—Ç-∞-Ω-æ-≤-ª-µ-Ω-∏—è -ø-æ

0085 50 push AX ;/ -∫-æ-°-∞-Ω-¥-µ ret, -∑-∞-≤-µ—Ä-à-∞—é—â-µ-π -ø—Ä-æ—Ü-µ-¥—É—Ä—É.

0086 B8 ---- R mov AX,DATA ; -ó-∞-≥—Ä—É-∑-∫-∞ —Å-µ-≥-°-µ-Ω—Ç-Ω-æ-≥-æ

0089 8E D8 mov DS,AX ; —Ä-µ-≥-∏—Å—

Ç—Ä-∞ -¥-∞-Ω-Ω—ã—Ö.

008B B4 35 MOV AH, 35H ; —Ñ—É-Ω-∫—Ü-∏—è -ø-æ-ª—É—á

-µ-Ω-∏—è -≤-µ-∫—Ç-æ—Ä-∞

008D B0 60 MOV AL, 60H ; -Ω-æ-°-µ—Ä -≤-µ-∫—Ç-æ—Ä-∞

008F CD 21 INT 21H ; -≤-æ-∑-≤—Ä-∞—â-∞-µ—Ç —Ç-µ-∫—É

—â-µ-µ -∑-Ω-∞—á-µ-Ω-∏-µ -≤-µ-∫—Ç-æ—Ä-∞ -ø—Ä-µ—Ä —ã-≤-∞-Ω-∏—è

0091 89 1E 0002 R MOV KEEP_IP, BX ; -∑-∞-ø-æ-°-∏-Ω-∞-Ω-∏-

µ —Å-°-µ—â-µ-Ω-∏—è

0095 8C 06 0000 R MOV KEEP_CS, ES ; -∏ —Å-µ-≥-°-µ-Ω—Ç-∞ -

≤-µ-∫—Ç-æ—Ä-∞ -ø—Ä-µ—Ä—ã-≤-∞-Ω-∏—è

0099 1E PUSH DS

009A BA 001E R MOV DX, OFFSET SUBR_INT ; —Å-°-µ—â-µ-Ω-∏-µ -¥-ª—è -ø—Ä-æ—Ü-µ-¥—É—Ä—ã -≤ DX

009D B8 ---- R MOV AX, SEG SUBR_INT ; —Å-µ-≥-°-µ-Ω—Ç -ø—Ä-æ—Ü-µ-¥—É—Ä—ã

00A0 8E D8 MOV DS, AX ; -ø-æ-°-µ—â-∞-µ-° -≤ DS

00A2 B4 25 MOV AH, 25H ; —Ñ—É-Ω-∫—Ü-∏—è —É—Å-∞-Ω

-æ-≤-∫-∏ -≤-µ-∫—Ç-æ—Ä-∞

00A4 B0 60 MOV AL, 60H ; -Ω-æ-°-µ—Ä -≤-µ-∫—Ç-æ—Ä-∞

```
00A6  CD 21                         INT 21H ; -°-µ-Ω-è-µ-° -ø-Ä-µ-Ä-ã-≤-
∞-Ω
                                    -∏-µ
00A8  1F                            POP DS

00A9  CD 60                         int 60H; -≤-ã-∑-æ-≤ -∏-∑-°-µ-Ω-µ-Ω-
Ω-æ-
                                    ≥-æ -ø-Ä-µ-Ä-ã-≤-∞-Ω-∏-è

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 60                         MOV AL, 60H
00BA  CD 21                         INT 21H ; -≤-æ-Å-Å-Ç-∞-Ω-∞-≤-ª-∏-≤-
∞-µ-
                                    ° -Å-Ç-∞-Ä-ã-π -≤-µ-∫-Ç-æ-Ä -ø-Ä-µ-Ä-ã-≤-∞-Ω-∏-
                                    è
00BC  1F                            POP DS
00BD  FB                            STI

00BE  CB                            RET
00BF                          Main      ENDP
00BF                          CODE      ENDS
                                  END Main
```

Segments and Groups:

N a m e	Length	Align	Combine Class
CODE	00BF	PARA	NONE
DATA	0008	PARA	NONE
STACK	0800	PARA	STACK

Symbols:

N a m e	Type	Value	Attr	
GETINT	N PROC	0000	CODE	Length = 001E
GI2	L NEAR	0007	CODE	
GI3	L NEAR	0013	CODE	
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 = 003D
MESSAGE	L BYTE	0006	DATA	Length = 0002
NUM	L WORD	0004	DATA	
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 lab5			
@VERSION	TEXT 510			

124 Source Lines

```
  124 Total   Lines
   21 Symbols

48018 + 457192 Bytes symbol space free

    0 Warning Errors
    0 Severe  Errors
```