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.

Студент гр. 0382 _____ Тихонов С.В.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург
2021

Цель работы.

Узнать как работает механизм прерываний в ассемблере. Научится писать собственные прерывания.

Задание.

Вариант 3D

Назначение заменяемого вектора прерывания:

3 - 23h - прерывание, генерируемое при нажатии клавиш Control+C;

Действие, реализуемые программой обработки прерываний:

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

Выполнение работы.

В сегменте данных выделяем память для запоминания адреса старого прерывания (чтобы потом вернуть все как было).

```
DATA    SEGMENT
    KEEP_CS DW 0
    KEEP_IP DW 0
DATA    ENDS
```

В Main записываем сегмент и смещение прерывания 23h, с помощью 23h и 21h. С помощью 25h int 21< устанавливаем вектор прерывания 23h на наше прерывание ABC и вызываем его. По завершению прерывания восстанавливаем старый вектор.

В пользовательской процедуре прерывания ABC сначала мы сохраняем все изменяемые регистры в стек. Далее происходит вызов функций для вывода времени в тиках. Он происходит два раза, так как нужные нам данные хранятся в двух сегментах, сначала мы вызываем для CX, а затем для BX.

Выводы.

Был изучен механизм прерываний в языке ассемблер.

В ходе данной лабораторной работы была разработана программа , которая создает свое собственное прерывание, которое после нажати ctr+c выполнить вывод на экран системное время в тиках.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: lb5.asm

```
AStack  SEGMENT STACK
    DB 1024 DUP(?)
AStack  ENDS

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

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

WriteMsg  PROC  NEAR
           push dx
      push cx
      xor cx,cx
      mov bx,10
f1:
      xor dx,dx
      div bx
      push dx
      inc cx
      test ax,ax
      jnz f1
      mov ah,02h
f2:
      pop dx
      add dl,'0'
      int 21h
      loop f2
      pop cx
      pop dx
      ret
WriteMsg  ENDP

ABC PROC FAR
             JMP start
           KEEP_SP DW 0000h
```

```
            KEEP_SS DW 0000h
            INT_STACK DB 40 DUP(0)

      start:
            mov KEEP_SS, SS
            mov KEEP_SP, SP
            mov ss, sp
            mov sp, OFFSET start
            push ax
            push cx
            push dx


            mov ah,0h
            int 1Ah
            mov ax,cx
            call WriteMsg
            mov ax,dx
            call WriteMsg


            pop ax
            pop cx
            pop dx
            mov ss, KEEP_SS
            mov sp, KEEP_SP
            mov al, 20h
            out 20h, al
            iret
ABC ENDP

MAIN PROC FAR
    push ds
    mov ax, DATA
    mov ds, ax

    mov ah, 35h ; функция получения вектора
    mov al, 23h ; номер вектора
    int 21h
    mov KEEP_IP, bx ; запоминание смещения
    mov KEEP_CS, es ; и сегмента вектора прерывания

    push ds
    mov dx, OFFSET ABC ; смещение для процедуры в DX
    mov ax, SEG ABC ; сегмент процедуры
    mov ds, ax ; помещаем в DS
    mov ah, 25h ; функция установки вектора
    mov al, 23h ; номер вектора
    int 21h ; меняем прерывание
    pop ds

       begin:
      mov ah, 0
            int 16h
            cmp al, 'q'
            je quit
```

```
            cmp al, 3
            jnz begin
            int 23h
            jmp begin
      quit:
      cli
      push ds
      mov dx, KEEP_IP
      mov ax, KEEP_CS
      mov ds, ax
      mov ah, 25h
      mov al, 23h
      int 21h ; восстанавливаем старый вектор прерывания
      pop ds
      sti
            mov ah, 4ch
            int 21h
MAIN ENDP
CODE ENDS
     END MAIN
```