**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. 1383 _____ Малых А.А.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург
2022

**Цель работы.**

Получить навыки реализации собственного обработчика прерываний.

**Задание.**

**(Вариант №13)**

Код — 2g.

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

g - Вывод на экран заданного количества (3-5) сообщений, задержка между которыми возрастает в 2 раза, начиная от 1 сек

**Выполнение работы.**

Разработана программа, вызывающая и обрабатывающая прерывание 60h. Обработчик прерываний выводит на экран 4 сообщения «Hello, World!», с возрастающей в 2 раза задержкой начиная от 1 секунды. В процедуре INT_HANDLER, обрабатывающей прерывание, в начале ds настраивается на начало сегмента DATA, а в dx кладётся смещение до строки, которая будет выводиться. В cx кладётся количество выводимых сообщений. Далее в цикле вызывается процедура WriteMsg, которая выводит на экран сообщение с помощью прерывания 21h. Затем в цикле от 0 до delay_time вызывается процедура Delay, которая осуществляет секундную задержку с помощью функции 86h прерывания 15h. Когда данный цикл закончен, значение delay_time умножается на 2, а в delay_count кладётся 0. Если cx == 1, то задержки не делается — просто выводится сообщение и происходит выход из цикла.

Исходный код программы см. в Приложении А

**Выводы.**

Получены навыки реализации собственного обработчика прерываний.

## ПРИЛОЖЕНИЕ А
## ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: lab5.asm

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

DATA    SEGMENT
    KEEP_CS DW 0
    KEEP_IP DW 0
    message DB 'Hello, World!', 13, 10, '$'
    delay_time DB 1
    delay_count DB 0
DATA    ENDS

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


WriteMsg  PROC  NEAR
          mov ah, 9
          int 21h
          ret
WriteMsg  ENDP

Delay Proc NEAR
      ; Секундная задержка
      push cx
      push dx
      mov cx, 0fh
      mov dx, 4240h
      mov ah, 86h
      int 15h
      pop dx
      pop cx
      ret
```

```
Delay ENDP

INT_HANDLER  PROC FAR
     push ax
        mov ax, seg DATA
        mov ds, ax
     mov dx, offset message

        mov cx, 4
     printing_msg:
          call WriteMsg
               cmp cx, 1
               je end_loop
               delay_cycle:
               call Delay
               inc delay_count
                       mov al, delay_count
                       cmp al, delay_time
               jl delay_cycle
          shl delay_time, 1
               and delay_count, 0
          loop printing_msg
        end_loop:
     pop ax
     mov al, 20H
     out 20H, al
     iret
INT_HANDLER ENDP

Main PROC FAR
   mov  ah, 35h   ; функция получения вектора
   mov  al, 60h   ; номер вектора
   int  21h
   mov KEEP_IP, bx  ; запоминание смещения
   mov KEEP_CS, es  ; и сегмента

   push ds
```

```
    mov dx, offset INT_HANDLER ; смещение для процедуры в DX
    mov ax, seg INT_HANDLER    ; 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

    cli
    push ds
    mov dx, KEEP_IP
    mov ax, KEEP_CS
    mov ds, ax
    mov ah, 25h
    mov al, 60h
    int 21h          ; восстанавливаем вектор
    pop ds
    sti
    mov ah, 4ch
    int 21h
    ret

Main ENDP
CODE ENDS
END Main
```