

**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. 1303

Попандопуло А. Г.

Преподаватель

Ефремов М.А.

Санкт-Петербург

2022

## **Цель работы.**

Практическое изучение прерываний на языке Ассемблера, написание собственного прерывания согласно условию.

## **Задание.**

Вариант 22 (шифр 4а):

Написать прерывание 16h - прерывание от клавиатуры, выполняющее вывод сообщения на экран заданное число раз, после чего вставить фиксированную задержку и вывести сообщение о завершении обработчика.

Замечание: для исключения возможного взаимного влияния системных и пользовательских прерываний рекомендуется отвести в программе под стек не менее 1К байт.

## **Выполнение работы**

Для хранения сегмента заменяемого прерывания и для хранения смещения заменяемого прерывания, в сегменте памяти выделяем место под слова `keep_cs` и `keep_ip` соответственно. Согласно условию, на стек отводим 1 Кб.

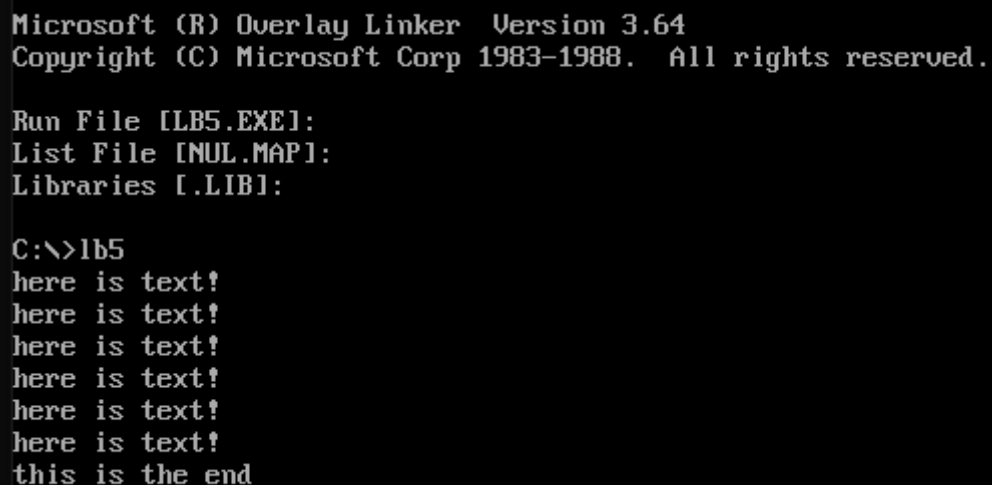
С помощью 35h сохраняем адрес прошлого прерывания, с помощью 25h – устанавливаем адрес нового. Инициализируем строки `MESSAGE` – повторяющееся некоторое количество раз (во время обработки прерывания) сообщение, и `END_MESSAGE` – сообщение о завершении обработчика.

Далее следует ожидание ввода символа от пользователя, в соответствии с условием, взят символ «а». На метке `check_key` происходит считывание из порта клавиатуры 60h с последующим сравнением на 1Eh – скан, соответствующий клавише «а». Непосредственно прерывание вызывается при нажатии нужной клавиши; в противном случае, вновь переходим на метку `check_key`, таким образом, «ожидая» нужного символа.

В сегменте кода, помимо процедуры самого прерывания, была определена процедура `WriteMsg` – для печати сообщения. В процедуре прерывания, сохраняем в стек изначальные значения регистров, после чего посредством `Ir` строка из `dx` выводится заданным в `sx` числом раз. Задержка после нужного

числа выводов строк происходит через прерывания 15h; после нее выводится  
END\_MESSAGE.

### Тестирование:



```
Microsoft (R) Overlay Linker  Version 3.64
Copyright (C) Microsoft Corp 1983-1988.  All rights reserved.

Run File [LB5.EXE]:
List File [NUL.MAP]:
Libraries [LIB1]:

C:\>lb5
here is text!
here is text!
here is text!
here is text!
here is text!
here is text!
this is the end
```

Рис. 1

**Вывод:** в ходе выполнения лабораторной работы, на практике были изучены способы работы с прерываниями на языке Ассемблера; удалось реализовать собственное прерывание, соответствующее заданному условию.

## Приложение А.

### Исходный код программы.

```
#include <iostream>
#include <stdio.h>
#include <cstring>
#include <fstream>
```

```
char input[81];
char output[300];
```

```
int main() {
    system("chcp 1251 > nul");
    setlocale(LC_CTYPE, "rus");
```

```
    std::cout << "Попандопуло Александр 1303\nПреобразование введенных во
входной строке шестнадцатиричных цифр в двоичную СС.\n";
```

```
    std::cout << "Введите строку..\n";
    std::cin.getline(input, 81);
```

```
    std::ofstream file;
    file.open("result.txt");
```

```
__asm {
    push ds
    pop es
    mov esi, offset input
    mov edi, offset output
```

```
checks :
lodsb
    cmp al, '2'
    jne symbol3
    mov ax, '01'
    stosw
    jmp next
```

```
symbol3 :
cmp al, '3'
jne symbol4
mov ax, '11'
stosw
```

jmp next

symbol4 :  
cmp al, '4'  
jne symbol5  
mov ax, '01'  
stosw  
mov al, '0'  
stosb  
jmp next

symbol5 :  
cmp al, '5'  
jne symbol6  
mov ax, '01'  
stosw  
mov al, '1'  
stosb  
jmp next

symbol6 :  
cmp al, '6'  
jne symbol7  
mov ax, '11'  
stosw  
mov al, '0'  
stosb  
jmp next

symbol7 :  
cmp al, '7'  
jne symbol8  
mov ax, '11'  
stosw  
mov al, '1'  
stosb  
jmp next

symbol8 :  
cmp al, '8'  
jne symbol9  
mov eax, '0001'  
stosd  
jmp next

```
symbol9 :  
cmp al, '9'  
jne symbolA  
mov eax, '1001'  
stosd  
jmp next
```

```
symbolA :  
cmp al, 'A'  
jne symbolB  
mov eax, '0101'  
stosd  
jmp next
```

```
symbolB :  
cmp al, 'B'  
jne symbolC  
mov eax, '1101'  
stosd  
jmp next
```

```
symbolC :  
cmp al, 'C'  
jne symbolD  
mov eax, '0011'  
stosd  
jmp next
```

```
symbolD :  
cmp al, 'D'  
jne symbolE  
mov eax, '1011'  
stosd  
jmp next
```

```
symbolE :  
cmp al, 'E'  
jne symbolF  
mov eax, '0111'  
stosd  
jmp next
```

```
symbolF:  
cmp al, 'F'  
jne letter
```

```
    mov eax, '1111'  
    stosd  
    jmp next
```

```
    letter :  
    stosb
```

```
    next :  
    mov ecx, '\0'  
    cmp ecx, [esi]  
    je end  
    jmp checks  
    end :  
};
```

```
    std::cout << "Строка с шестнадцатиричными цифрами, преобразованными в  
двоичную СС:\n";  
    std::cout << output;  
    file << output;  
    file.close();  
  
    return 0;  
}
```