

**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.**  
**Вариант 15.**

Студент гр. 0383

\_\_\_\_\_

Смирнов И.А.

Преподаватель

\_\_\_\_\_

Ефремов М.А.

Санкт-Петербург

2021

### **Цель работы.**

Изучить работу прерываний на языке Ассемблера и написать собственное.

### **Задание.**

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

A — Выполнить вывод сообщения на экран заданное число раз, после чего вставить фиксированную задержку и вывести сообщение о завершении обработчика.

### **Ход работы:**

В сегменте данных DATA хранятся следующие переменные: KEEP\_CS, KEEP\_IP — для хранения сегмента и смещения старого прерывания, COUNTER — для количества выводимых строк, MESSAGE — сообщение, которое надо вывести несколько раз, FINALLY — сообщение о завершении обработчика.

Процедура пользовательского прерывания называется FUNC. В начале данной процедуры мы сохраняем все изменяемые регистры в стеке с помощью push. Далее запускаем цикл по метке start для вывода сообщения MESSAGE на экран несколько раз. Как только COUNTER = 0, цикл прекращается. После этого мы кладем временной промежуток в cx и dx, в ah кладем 86h, следовательно вызываем прерывание паузы. После данной паузы печатается сообщение о завершении. Для вывода строк на экран написана процедура WriteMsg. В конце процедуры прерывания восстанавливаем регистры из стека и выходим из пользовательского прерывания.

В главной процедуре программы Main запоминаем смещение и сегмент прерывания 23h в KEEP\_IP, KEEP\_CS с помощью 35h и 21h. С помощью 25h прерывания 21, устанавливаем вектор прерывания 23h на пользовательское прерывание FUNC и производим его вызов. По завершении прерывания восстанавливаем его старый вектор.

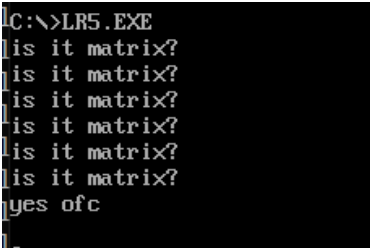
Исходный код программы см. в приложении А.

Файл листинга см. в приложении Б.

### Тестирование:

Для проверки работоспособности программы были проведены тесты, см. Таблицу 1.

Таблица 1 — Результаты тестирования.

№ теста	Входные данные	Выходные данные	Оценка результата
1	(нажато Ctrl + C)		Выводится фиксированное количество сообщений(6 штук), после с задержкой в 3с выводится сообщение о завершении обработчика

### Выводы.

В данной лабораторной работе были изучены прерывания языка Ассемблера. Написано собственное прерывание, которое выводит строки на экран и сообщение о завершении с задержкой.

## ПРИЛОЖЕНИЕ А

### ИСХОДНЫЙ КОД ПРОГРАММЫ

#### Название файла: lr5.asm

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

DATA    SEGMENT
    MESSAGE DB 'is it matrix?', 0dh, 0ah, '$'
    FINALLY DB 'yes ofc$'
    COUNTER DW 6
    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
    mov AH, 9
    int 21h
    ret
WriteMsg ENDP

FUNC PROC FAR
    mov dx, OFFSET MESSAGE
    push ax
    push bx
    push cx
    push dx
    push ds
    start:
        call WriteMsg
        sub COUNTER, 1
        cmp COUNTER, 0
        jnz start
    mov cx, 0033h
    mov dx, 00FFh
    mov ah, 86h
    int 15h
    mov dx, OFFSET FINALLY
    call WriteMsg
    pop ax
    pop bx
    pop cx
    pop dx
    pop ds
    mov al, 20h
    out 20h, al
    iret
```

FUNC 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 FUNC ; смещение для процедуры в DX
    mov ax, SEG FUNC ; 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, 3
        jnz begin
        int 23h

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

**ПРИЛОЖЕНИЕ Б**  
**ФАЙЛЫ ЛИСТИНГА ПРОГРАММЫ**

**Название файла: lr5.lst**

Microsoft (R) Macro Assembler Version 5.10

12/16/21 15:22:3

Page 1-1

```
0000                      AStack SEGMENT STACK
0000 0400[                DB 1024 DUP(?)
    ??
    ]

0400                      AStack ENDS

0000                      DATA  SEGMENT
0000 0000                KEEP_CS DW 0   ; для хранения сегмента век
    тора прерывания
0002 0000                KEEP_IP DW 0   ; для смещения вектора прер
    ывания
0004                      DATA  ENDS

0000                      CODE   SEGMENT
    ASSUME CS:CODE, DS:DATA, SS:AStack

0000                      WriteMsg PROC NEAR
0000 B4 09                mov AH, 9
0002 CD 21                int 21h
0004 C3                  ret
0005                      WriteMsg ENDP
```

0005	FUNC PROC FAR
0005 EB 71 90	jmp process
0008 0000	KEEP_SS DW 0
000A 0000	KEEP_SP DW 0
000C 69 73 20 69 74 20	MESSAGE DB 'is it matrix?', 0dh, 0ah, '
	\$'
6D 61 74 72 69 78	
3F 0D 0A 24	
001C 79 65 73 20 6F 66	FINALLY DB 'yes ofc', 0dh, 0ah, '\$'
63 0D 0A 24	
0026 0006	COUNTER DW 6
0028 0028[	func_stack DW 40 DUP (?)
????	
]	
0078	process:
0078 2E: 8C 16 0008 R	mov KEEP_SS, SS
007D 2E: 89 26 000A R	mov KEEP_SP, SP
0082 B8 ---- R	mov ax, SEG func_stack
0085 8E D0	mov ss, ax
0087 BC 0078 R	mov sp, OFFSET process
008A 50	push ax
008B 53	push bx
008C 51	push cx
008D 52	push dx
008E 1E	push ds
008F B8 ---- R	mov ax, SEG func_stack
0092 8E D8	mov ds, ax
0094 BA 000C R	mov dx, OFFSET MESSAGE

0097

start:

0097 E8 0000 R

call WriteMsg



```
009A 2E: 83 2E 0026 R 01          sub COUNTER, 1
00A0 2E: 83 3E 0026 R 00          cmp COUNTER, 0
00A6 75 EF                        jnz start
00A8 2E: C7 06 0026 R 0006        mov COUNTER, 6
00AF B8 ---- R                   mov ax, DATA
00B2 8B D0                        mov dx, ax
00B4 B9 0033                      mov cx, 0033h
00B7 BA 00FF                      mov dx, 00FFh
00BA B4 86                        mov ah, 86h
00BC CD 15                        int 15h
00BE BA 001C R                    mov dx, OFFSET FINALLY
00C1 E8 0000 R                    call WriteMsg
00C4 58                           pop ax
00C5 5B                           pop bx
00C6 59                           pop cx
00C7 5A                           pop dx
00C8 1F                           pop ds
00C9 2E: 8E 16 0008 R              mov ss, KEEP_SS
00CE 2E: 8B 26 000A R              mov sp, KEEP_SP
00D3 B0 20                        mov al, 20h
00D5 E6 20                        out 20h, al
00D7 CF                           iret
00D8                               FUNC ENDP

00D8                               MAIN PROC FAR
00D8 1E                           push ds
```

00D9 B8 ---- R	mov ax, DATA
00DC 8E D8	mov ds, ax
00DE B4 35	mov ah, 35h ; функция получения вектора
00E0 B0 23	mov al, 23h ; номер вектора
00E2 CD 21	int 21h
00E4 89 1E 0002 R	mov KEEP_IP, bx ; запоминание смещения
00E8 8C 06 0000 R	mov KEEP_CS, es ; и сегмента вектора прерыв
	ания
00EC 1E	push ds
00ED BA 0005 R	mov dx, OFFSET FUNC ; смещение для
процедур	
	ы в DX
00F0 B8 ---- R	mov ax, SEG FUNC ; сегмент процедуры
00F3 8E D8	mov ds, ax ; помещаем в DS
00F5 B4 25	mov ah, 25h ; функция установки вектора
00F7 B0 23	mov al, 23h ; номер вектора
00F9 CD 21	int 21h ; меняем прерывание
00FB 1F	pop ds
00FC	begin:
00FC B4 00	mov ah, 0
00FE CD 16	int 16h
0100 3C 71	cmp al, 'q'
0102 74 08	je quit
0104 3C 03	cmp al, 3
0106 75 F4	jnz begin
0108 CD 23	int 23h

```
010A EB F0                                jmp begin
010C                                quit:
010C FA                                cli
010D 1E                                push ds
010E 8B 16 0002 R                      mov dx, KEEP_IP
0112 A1 0000 R                        mov ax, KEEP_CS
0115 8E D8                            mov ds, ax
0117 B4 25                            mov ah, 25h
0119 B0 23                            mov al, 23h
011B CD 21                            int 21h ; восстанавливаем старый вектор
                                     прерывания
011D 1F                                pop ds
011E FB                                sti
011F B4 4C                            mov ah, 4ch
0121 CD 21                            int 21h
0123                                MAIN ENDP
0123                                CODE ENDS
                                END MAIN
```

## Symbols-1

## Segments and Groups:

N a m e	Length	Align	Combine	Class
ASTACK .....	0400	PARA		STACK
CODE .....	0123	PARA		NONE
DATA .....	0004	PARA		NONE

## Symbols:

N a m e	Type	Value	Attr	
BEGIN .....	L NEAR	00FC	CODE	
COUNTER .....	L WORD	0026	CODE	
FINALLY .....	L BYTE	001C	CODE	
FUNC .....	F PROC	0005	CODE	Length = 00D3
FUNC_STACK .....	L WORD	0028	CODE	Length = 0028
KEEP_CS .....	L WORD	0000	DATA	
KEEP_IP .....	L WORD	0002	DATA	
KEEP_SP .....	L WORD	000A	CODE	
KEEP_SS .....	L WORD	0008	CODE	
MAIN .....	F PROC	00D8	CODE	Length = 004B

MESSAGE .....	L BYTE	000C	CODE	
PROCESS .....	L NEAR	0078	CODE	
QUIT .....	L NEAR	010C	CODE	
START .....	L NEAR	0097	CODE	
WRITEMSG .....	N PROC	0000	CODE	Length = 0005
@CPU .....	TEXT	0101h		
@FILENAME .....	TEXT	LR5		
@VERSION .....	TEXT	510		

111 Source Lines

111 Total Lines

23 Symbols

48030 + 459230 Bytes symbol space free

0 Warning Errors

0 Severe Errors