

**МИНОБРНАУКИ РОССИИ**  
**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**  
**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**  
**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**  
**Кафедра МО ЭВМ**

**ОТЧЕТ**  
**по лабораторной работе №5**  
**по дисциплине «Операционные системы»**  
**Тема: Сопряжение стандартного и пользовательского обработчиков**  
**прерываний**

Студент гр. 8383

\_\_\_\_\_

Шишкин И.В.

Преподаватель

\_\_\_\_\_

Ефремов М.А.

Санкт-Петербург

2020

### **Цель работы.**

Исследование возможности встраивания пользовательского обработчика прерываний в стандартный обработчик от клавиатуры.

### **Ход работы.**

Был написан программный модуль типа .EXE, который выполняет следующие функции:

- 1) Проверяет, установлено ли пользовательское прерывание с вектором 09h.
- 2) Устанавливает резидентную функцию для обработки прерывания и настраивает вектор прерываний, если прерывание не установлено, и осуществляет выход по функции 4Ch прерывания int 21h.
- 3) Если прерывание установлено, то выводится соответствующее сообщение и осуществляется выход по функции 4Ch прерывания int 21h.
- 4) Выгрузка прерывания по соответствующему значению параметра в командной строке /un. Выгрузка прерывания состоит в восстановлении стандартного вектора прерываний и освобождении памяти, занимаемой резидентом. Затем осуществляется выход по функции 4Ch прерывания int 21h.

Обработчик прерываний обрабатывает клавиши цифр с 1 до 0 (скан-коды с 02h до 0Ah). Рисунок скан-кодов клавиш изображен на рис. 1. После нажатия на экран выводится сумма предыдущего числа и введенного. Например, если сначала нажать 2, а потом 3, то выведется 5. Если после сложения число оказывается двузначным, то от него отнимается 10. Так, если нажать 5, а потом 8, то выведется 3. Пример работы программы приведен на рис. 2 и рис. 3. Если же была нажата другая клавиша, то управление передается стандартному обработчику прерывания.

01	3B	3C	3D	3E	3F	40	41	42	43	44	45	46		
29	02	03	04	05	06	07	08	09	0A	0B	0C	0D	2B	0E
0F	10	11	12	13	14	15	16	17	18	19	1A	1B		
3A	1E	1F	20	21	22	23	24	25	26	27	28	1C		
2A	2C	2D	2E	2F	30	31	32	33	34	35	36			
1D	38	39								E0,38			E0,1D	

Рисунок 1 – Скан-коды клавиш

```
C:\>os5.exe
Interrupt not yet set
C:\>37
```

Рисунок 2 – После нажатия сначала 3, а потом 4

```
C:\>os5.exe
Interrupt not yet set
C:\>373_
```

Рисунок 3 – После нажатия сначала 3, затем 4, а потом 6

С помощью программы из ЛРЗ было выведено состояние памяти: на рис. 4 – до загрузки прерывания, на рис. 5 – после загрузки прерывания, на рис. 6 – после выгрузки прерывания.

```
C:\>os3_2.com

Amount of available memory: E6D0
Memory freed
Extended memory size: 3C00
-----
Number 1
Area belongs to MS DOS
Area size: 0010
-----
Number 2
Free area
Area size: 0040
-----
Number 3
0040
Area size: 0100
-----
Number 4
0192
Area size: 0090
-----
Number 5
0192
Area size: 0670      OS3_2
-----
Number 6
Free area
Area size: E050
```

Рисунок 4 – До загрузки прерывания

```

Interrupt not yet set
C:\>os3_2.com

Amount of available memory: E2B0
Memory freed
Extended memory size: 3C00
-----
Number 1
Area belongs to MS DOS
Area size: 0010
-----
Number 2
Free area
Area size: 0040
-----
Number 3
0040
Area size: 0100
-----
Number 4
0192
Area size: 0090
-----
Number 5
0192
Area size: 0370          OS5
-----
Number 6
01D4
Area size: 0090
-----
Number 7
01D4
Area size: 0670          OS3_2
-----
Number 8
Free area
Area size: DC30

```

Рисунок 5 – После загрузки прерывания

```

Interrupt already set, but the /un parameter is found
C:\>os3_2.com

Amount of available memory: E6D0
Memory freed
Extended memory size: 3C00
-----
Number 1
Area belongs to MS DOS
Area size: 0010
-----
Number 2
Free area
Area size: 0040
-----
Number 3
0040
Area size: 0100
-----
Number 4
0192
Area size: 0090
-----
Number 5
0192
Area size: 0670      OS3_2
-----
Number 6
Free area
Area size: E050      =      Δb0

```

Рисунок 6 – После выгрузки прерывания

Как видно из рисунков, после выгрузки прерывания, блоки, соответствующие программе ЛР5, удаляются.

### Контрольные вопросы.

1. Какого типа прерывания использовались в работе?

Сервис DOS int 21h.

Прерывание от клавиатуры int 09h. Была написана резидентная программа, которая перехватывает int 09h и проверяет на определенный ключ.

2. Чем отличается скан код от кода ASCII?

Скан-код – код, присвоенный каждой клавише, с помощью которого драйвер клавиатуры распознает, какая клавиша была нажата. ASCII код же, в свою очередь, является не кодом клавиши, а кодом символа.

### **Выводы.**

В ходе выполнения лабораторной работы была реализована программа, загружающая и выгружающая пользовательское прерывание от клавиатуры.

## ПРИЛОЖЕНИЕ

### КОД ПРОГРАММЫ

CODE SEGMENT

ASSUME CS:CODE, DS:DATA, SS:AStack, ES:NOTHING

ROUT PROC FAR

jmp INTERRUPT\_BEGIN

ADDITION\_RESULT dw 0

CHAR db 0

INTERRUPT\_ID dw 9888h

KEEP\_AX dw 0

KEEP\_SS dw 0

KEEP\_SP dw 0

KEEP\_IP dw 0

KEEP\_CS dw 0

KEEP\_PSP DW 0

INTERRUPTION\_STACK dw 128 dup(0)

INTERRUPT\_BEGIN:

mov KEEP\_SS, SS

mov KEEP\_SP, SP

mov KEEP\_AX, AX

mov AX, SEG INTERRUPTION\_STACK

mov SS, AX

mov AX, offset INTERRUPTION\_STACK

add AX, 256 ; на конец стека

mov SP, AX

push BX

push CX

push DX

push SI

push DS

push ES

mov AX, SEG CHAR

mov DS, AX

in AL, 60h ;читать ключ

cmp AL, 02h

jne CONTINUE\_FROM\_1

mov AX, ADDITION\_RESULT ;1

add AX, 1

cmp AX, 9



```

jg IF_ADD_RES_GREATER_9
jmp DO_REQ
CONTINUE_FROM_1:
    cmp AL, 03h
    jne CONTINUE_FROM_2
    mov AX, ADDITION_RESULT    ;2
    add AX, 2
    cmp AX, 9
    jg IF_ADD_RES_GREATER_9
    jmp DO_REQ
CONTINUE_FROM_2:
    cmp AL, 04h
    jne CONTINUE_FROM_3
    mov AX, ADDITION_RESULT    ;3
    add AX, 3
    cmp AX, 9
    jg IF_ADD_RES_GREATER_9
    jmp DO_REQ
CONTINUE_FROM_3:
    cmp AL, 05h
    jne CONTINUE_FROM_4
    mov AX, ADDITION_RESULT    ;4
    add AX, 4
    cmp AX, 9
    jg IF_ADD_RES_GREATER_9
    jmp DO_REQ
CONTINUE_FROM_4:
    cmp AL, 06h
    jne CONTINUE_FROM_5
    mov AX, ADDITION_RESULT    ;5
    add AX, 5
    cmp AX, 9
    jg IF_ADD_RES_GREATER_9
    jmp DO_REQ
CONTINUE_FROM_5:
    cmp AL, 07h
    jne CONTINUE_FROM_6
    mov AX, ADDITION_RESULT    ;6
    add AX, 6
    cmp AX, 9
    jg IF_ADD_RES_GREATER_9
    jmp DO_REQ
CONTINUE_FROM_6:
    cmp AL, 08h

```

```

jne CONTINUE_FROM_7
mov AX, ADDITION_RESULT      ;7
add AX, 7
cmp AX, 9
jg IF_ADD_RES_GREATER_9
jmp DO_REQ
IF_ADD_RES_GREATER_9: ;если в результате хранится
двузначное число
add AX, -10
jmp DO_REQ
CONTINUE_FROM_7:
cmp AL, 09h
jne CONTINUE_FROM_8
mov AX, ADDITION_RESULT      ;8
add AX, 8
cmp AX, 9
jg IF_ADD_RES_GREATER_9
jmp DO_REQ
CONTINUE_FROM_8:
cmp AL, 0Ah
jne CONTINUE_FROM_9
mov AX, ADDITION_RESULT
add AX, 9
cmp AX, 9
jg IF_ADD_RES_GREATER_9
jmp DO_REQ
CONTINUE_FROM_9:
cmp AL, 0Bh
jne IF_NO_CORRECT_SYM
mov AX, ADDITION_RESULT
jmp DO_REQ
IF_NO_CORRECT_SYM:
pushf ;уйти на исходный обработчик
call DWORD PTR CS:KEEP_IP
jmp END_OF_INT
DO_REQ: ;следующий код необходим для обработки аппаратного
прерывания
mov ADDITION_RESULT, AX
add AL, 30h;перевести число в символ
mov CHAR, AL
;int 29h ;вывести его на экран!
in AL, 61h ;взять значение порта управления клавиатурой
mov AH, AL ;сохранить его
or AL, 80h ;установить бит разрешения для клавиатуры

```

```

out 61h, AL ;и вывести его в управляющий порт
xchg AL, AL ;извлечь исходное значение порта
out 61h, AL ;и записать его обратно
mov AL, 20h ;послать сигнал "конец прерывания"
out 20h, AL ;контроллеру прерываний 8259

```

WRITE\_ANS:

```

mov AH, 05h
mov CL, CHAR
mov CH, 00h
int 16h
or AL, AL
jz END_OF_INT

mov AX, 0040h
mov ES, AX
mov AX, ES:[1Ah]
mov ES:[1Ch], AX
jmp WRITE_ANS

```

END\_OF\_INT:

```

pop ES
pop DS
pop SI
pop DX
pop CX
pop BX
mov SP, KEEP_SP
mov AX, KEEP_SS
mov SS, AX
mov AX, KEEP_AX
mov AL, 20h
OUT 20h, AL
IRET

```

ROUT ENDP

LAST\_BYTE:

;-----

PRINT PROC near

```

push AX
mov AH, 09h
int 21h
pop AX
ret

```

PRINT ENDP

```

;-----
SET_INTERRUPT PROC near
    push AX
    push BX
    push CX
    push DX
    push DS
    push ES

    mov AH, 35H ; функция получения вектора
    mov AL, 09H ; номер вектора
    int 21H
    mov KEEP_IP, BX ; запоминание смещения
    mov KEEP_CS, ES ; и сегмента

    CLI
    push DS
    mov DX, offset ROUT
    mov AX, seg ROUT
    mov DS, AX
    mov AH, 25H
    mov AL, 09H
    int 21H ; восстанавливаем вектор
    pop DS
    STI

    mov DX, offset LAST_BYTE
    add DX, 10Fh
    mov CL, 4h ; перевод в параграфы
    shr DX, CL
    inc DX ; размер в параграфах
    xor AX, AX
    mov AH, 31h
    int 21h

    pop ES
    pop DS
    pop DX
    pop CX
    pop BX
    pop AX
    ret
SET_INTERRUPT ENDP
;-----

```

INTERRUPT\_UPLOAD PROC near

push AX  
push BX  
push DX  
push DS  
push ES  
push SI

CLI

mov AH, 35h  
mov AL, 09h  
int 21h  
mov SI, offset KEEP\_IP  
sub SI, offset ROUT  
mov DX, ES:[BX+SI]  
mov AX, ES:[BX+SI+2]  
push DS  
mov DS, AX  
mov AH, 25h  
mov AL, 09h  
int 21h  
pop DS  
mov AX, ES:[BX+SI+4]  
mov ES, AX  
push ES  
mov AX, ES:[2Ch]  
mov ES, AX  
mov AH, 49h  
int 21h  
pop ES  
mov AH, 49h  
int 21h  
STI

pop SI  
pop ES  
pop DS  
pop DX  
pop BX  
pop AX  
ret

INTERRUPT\_UPLOAD ENDP

-----  
CHECK\_PARAMETER PROC near

```

push AX
push ES

mov AX, KEEP_PSP
mov ES, AX
cmp byte ptr ES:[81h+1], '/'
jne END_OF_PARAMETER
cmp byte ptr ES:[81h+2], 'u'
jne END_OF_PARAMETER
cmp byte ptr ES:[81h+3], 'n'
jne END_OF_PARAMETER
mov PARAMETER, 1

END_OF_PARAMETER:
    pop ES
    pop AX
    ret
CHECK_PARAMETER ENDP
;-----
CHECK_09H PROC near
    push AX
    push BX
    push SI

    mov AH, 35h
    mov AL, 09h
    int 21h
    mov SI, offset INTERRUPT_ID
    sub SI, offset ROUT
    mov AX, ES:[BX+SI]
    cmp AX, 9888h
    jne END_OF_CHECK
    mov IS_INTERRUPT_LOADED, 1

END_OF_CHECK:
    pop SI
    pop BX
    pop AX
    ret
CHECK_09H ENDP
;-----
BEGIN PROC FAR
    mov AX, DATA
    mov DS, AX

```

```

mov KEEP_PSP, ES

call CHECK_09H
call CHECK_PARAMETER
mov AL, PARAMETER
cmp AL, 1
je IF_UN

mov AL, IS_INTERRUPT_LOADED
cmp AL, 1
jne IF_NEED_TO_SET_INTERRUPT
mov DX, offset IF_INTERRUPT_SET
call PRINT
jmp ENDD

IF_NEED_TO_SET_INTERRUPT:
    mov DX, offset IF_INTERRUPT_NOTSET
    call PRINT
    call SET_INTERRUPT
    jmp ENDD

IF_UN:
    mov AL, IS_INTERRUPT_LOADED
    cmp AL, 1
    jne IF_1CH_NOT_SET
    mov DX, offset STR_UN
    call PRINT
    call INTERRUPT_UPLOAD
    jmp ENDD

IF_1CH_NOT_SET:
    mov DX, offset IF_INTERRUPT_NOTSET
    call PRINT

ENDD:
    xor AL, AL
    mov AH, 4Ch
    int 21h
BEGIN ENDP
CODE ENDS

AStack SEGMENT STACK
    dw 128 dup(0)
Astack ENDS

```

```
DATA SEGMENT
    IS_INTERRUPT_LOADED db 0
    PARAMETER db 0
    IF_INTERRUPT_SET db 'Interrupt already set $'
    IF_INTERRUPT_NOTSET db 'Interrupt not yet set $'
    STR_UN db 'Interrupt already set, but the /un parameter is found $'
DATA ENDS
    END BEGIN
```