

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

**ОТЧЕТ**  
**по лабораторной работе №4**  
**по дисциплине «Операционные системы»**  
**Тема: Обработка стандартных прерываний**

Студентка гр. 8382

\_\_\_\_\_

Рочева А.К.

Преподаватель

\_\_\_\_\_

Ефремов М.А.

Санкт-Петербург

2020

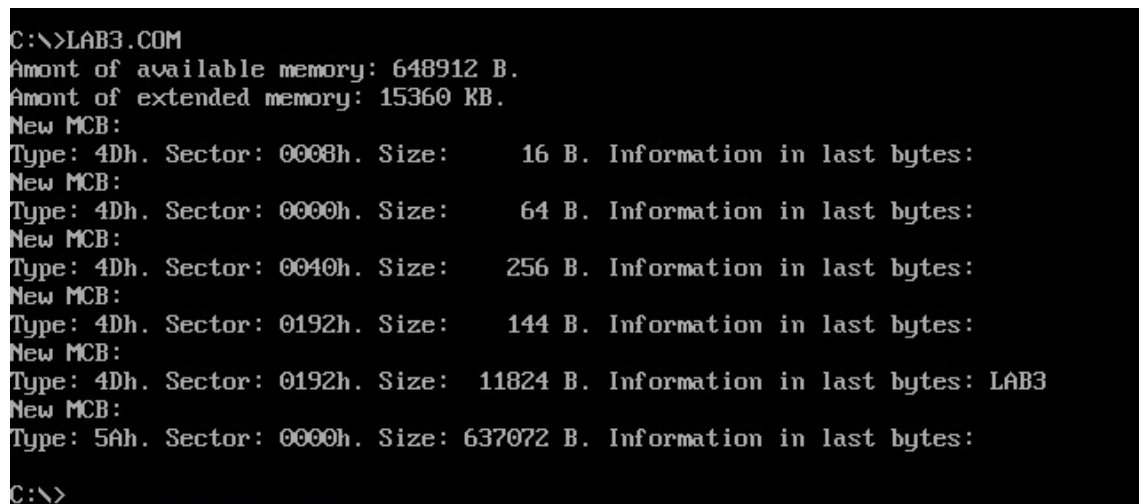
### **Цель работы.**

Построение обработчика прерываний сигналов таймера.

### **Ход выполнения.**

Для выполнения работы была написана программа (приложение А), проверяющая, установлено ли пользовательское прерывание с вектором 1Ch (в процедуре CHECK\_INT), устанавливающая резидентную функцию для обработки прерывания, настраивающая вектор прерывания, если прерывание не установлено (в процедуре LOAD\_INT). Выгрузка прерывания происходит в процедуре INT\_UNLOAD по значению параметра командной строки /un (происходит восстановление стандартного вектора прерываний и освобождение памяти, занимаемой резидентном).

На рис. 1 представлена карта памяти в виде блоков MCB до загрузки прерывания.



```
C:\>LAB3.COM
Amount of available memory: 648912 B.
Amount of extended memory: 15360 KB.
New MCB:
Type: 4Dh. Sector: 0008h. Size: 16 B. Information in last bytes:
New MCB:
Type: 4Dh. Sector: 0000h. Size: 64 B. Information in last bytes:
New MCB:
Type: 4Dh. Sector: 0040h. Size: 256 B. Information in last bytes:
New MCB:
Type: 4Dh. Sector: 0192h. Size: 144 B. Information in last bytes:
New MCB:
Type: 4Dh. Sector: 0192h. Size: 11824 B. Information in last bytes: LAB3
New MCB:
Type: 5Ah. Sector: 0000h. Size: 637072 B. Information in last bytes:
C:\>
```

Рис.1

На рис. 2 представлен запуск модуля LAB4.EXE (загрузка прерывания) и карта памяти (уже с размещенным прерыванием в памяти).

```

C:\>LAB4.EXE
Interruption loaded Interruption: 0072
C:\>LAB3.COM
Amount of available memory: 640784 B.
Amount of extended memory: 15360 KB.
New MCB:
Type: 4Dh. Sector: 0008h. Size:      16 B. Information in last bytes:
New MCB:
Type: 4Dh. Sector: 0000h. Size:      64 B. Information in last bytes:
New MCB:
Type: 4Dh. Sector: 0040h. Size:     256 B. Information in last bytes:
New MCB:
Type: 4Dh. Sector: 0192h. Size:     144 B. Information in last bytes:
New MCB:
Type: 4Dh. Sector: 0192h. Size:    7952 B. Information in last bytes: LAB4
New MCB:
Type: 4Dh. Sector: 038Eh. Size:    7144 B. Information in last bytes:
New MCB:
Type: 4Dh. Sector: 038Eh. Size:   11824 B. Information in last bytes: LAB3
New MCB:
Type: 5Ah. Sector: 0000h. Size: 628944 B. Information in last bytes:
Interruption: 0098
C:\>

```

Рис.2 — карта памяти с размещенным в ней прерыванием

На рис. 3. представлен повторный запуск программы, выводящей блоки памяти. Как видно, прерывание не поменяло своего расположения.

```

C:\>LAB3.COM
Amount of available memory: 640784 B.
Amount of extended memory: 15360 KB.
New MCB:
Type: 4Dh. Sector: 0008h. Size:      16 B. Information in last bytes:
New MCB:
Type: 4Dh. Sector: 0000h. Size:      64 B. Information in last bytes:
New MCB:
Type: 4Dh. Sector: 0040h. Size:     256 B. Information in last bytes:
New MCB:
Type: 4Dh. Sector: 0192h. Size:     144 B. Information in last bytes:
New MCB:
Type: 4Dh. Sector: 0192h. Size:    7952 B. Information in last bytes: LAB4
New MCB:
Type: 4Dh. Sector: 038Eh. Size:    7144 B. Information in last bytes:
New MCB:
Type: 4Dh. Sector: 038Eh. Size:   11824 B. Information in last bytes: LAB3
New MCB:
Type: 5Ah. Sector: 0000h. Size: 628944 B. Information in last bytes:
Interruption: 0280
C:\>

```

Рис. 3 — карта памяти с размещенным в ней прерывание (второй запуск)

На рис.4 представлен запуск программы с ключом /up и сразу после этого вывод блоков памяти.

```

C:\>LAB4.EXE /un
Interruption unloaded
C:\>LAB3.COM
Amount of available memory: 648912 B.
Amount of extended memory: 15360 KB.
New MCB:
Type: 4Dh. Sector: 0008h. Size:      16 B. Information in last bytes:
New MCB:
Type: 4Dh. Sector: 0000h. Size:      64 B. Information in last bytes:
New MCB:
Type: 4Dh. Sector: 0040h. Size:     256 B. Information in last bytes:
New MCB:
Type: 4Dh. Sector: 0192h. Size:     144 B. Information in last bytes:
New MCB:
Type: 4Dh. Sector: 0192h. Size:   11824 B. Information in last bytes: LAB3
New MCB:
Type: 5Ah. Sector: 0000h. Size: 637072 B. Information in last bytes:
C:\>

```

Рис. 4 - запуск программы с ключом /un и вывод блоков памяти

Как видно, произошла выгрузка прерывания, больше оно не занимает блоки памяти.

### Ответы на вопросы:

1. Как реализован механизм прерывания от часов?

При каждом вызове прерывания сохраняется состояние регистров, устанавливается флаг для запрета внешних прерываний и определяется источник прерывания (по его номеру определяется смещение в таблице векторов прерываний). Затем первые два байта помещаются в IP, вторые два байта в CS. Управление передается по адресу CS:IP, происходит обработка прерывания. После обработки происходит возврат управления прерванной программе.

2. Какого типа прерывания использовались в работе?

В работе использовались аппаратные прерывания (int 1Ch) и пользовательские прерывания (int 21h, int 10h).

## **Выводы**

В ходе выполнения работы был построен обработчик прерываний сигналов таймера.

## ПРИЛОЖЕНИЕ А

### ИСХОДНЫЙ КОД LAB4.ASM

```
AStack SEGMENT STACK
    dw 100h dup(?)
AStack ENDS

DATA SEGMENT
    INT_LOADED db 0
    MESSAGE_INT_LOADED db 'Interruption loaded$'
    MESSAGE_INT_NOT_LOADED db 'Interruption unloaded$'
DATA ENDS

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

ROUT PROC FAR
    jmp Start
    INT_COUNT db 'Interruption: 0000$'
    INT_ID dw 4040h
    KEEP_AX dw 0
    KEEP_SS dw 0
    KEEP_SP dw 0
    KEEP_IP dw 0
    KEEP_CS dw 0
    KEEP_PSP DW 0
    INT_STACK dw 100h dup(0)

Start:
    mov KEEP_SS, ss
    mov KEEP_SP, sp
    mov KEEP_AX, ax
    mov ax, seg INT_STACK
    mov ss, ax
    mov ax, offset INT_STACK
    add ax, 100h
    mov sp, ax

    push bx
    push cx
    push dx
    push si
    push ds
    push bp
    push es

    mov ah, 03h
    mov bh, 00h
    int 10h
    push dx

    mov ah, 09h
    mov bh, 0
    mov cx, 0
    int 10h

    mov ah, 02h
    mov bh, 0
    mov dh, 23
    mov dl, 20
    int 10h

    mov ax, seg INT_COUNT
    push ds
    push bp
    mov ds, ax
    mov si, offset INT_COUNT
    add si, 13
    mov cx, 4
```

```

Cycle:
    mov bp, cx
    mov ah, [si+bp]
    inc ah
    mov [si+bp], ah
    cmp ah, ':'
    jne CycleEnd
    mov ah, '0'
    mov [si+bp], ah
    loop CYCLE

CycleEnd:
    pop bp
    pop ds
    push es
    push bp
    mov ax, seg INT_COUNT
    mov es, ax
    mov bp, offset INT_COUNT
    call outputBP
    pop bp
    pop es

    pop dx
    mov ah, 02h
    mov bh, 0
    int 10h

    pop es
    pop bp
    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:

outputBP PROC near
    push ax
    push bx
    mov ah, 13h
    mov al, 1
    mov bl, 04h ;красненький цвет
    mov cx, 18
    mov bh, 0
    int 10h
    pop bx
    pop ax
    ret
outputBP ENDP

LOAD_INT PROC near
    push ax
    push bx
    push cx
    push dx
    push ds
    push es

    mov ah, 35h ; функция получения вектора

```

```

mov al, 1Ch ; номер вектора
int 21h
mov KEEP_IP, bx ; запоминание смещения
mov KEEP_CS, es ; и сегмента

push ds
mov dx, offset ROUT
mov ax, seg ROUT
mov ds, ax
mov ah, 25h
mov al, 1Ch
int 21h ; восстанавливаем вектор
pop ds

mov dx, offset LAST_BYTE
mov cl, 4h ; перевод в параграфы
shr dx, cl
add dx, CODE
inc dx
xor ax, ax
mov ah, 31h
int 21h

pop es
pop ds
pop dx
pop cx
pop bx
pop ax
ret
LOAD_INT ENDP

```

```

INT_UNLOAD PROC near
push ax
push bx
push dx
push ds
push es
push si

cli
mov ah, 35h
mov al, 1Ch
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, 1Ch
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
INT_UNLOAD ENDP

CHECK_INT PROC near
    push ax
    push bx
    push si

    mov ah, 35h
    mov al, 1Ch
    int 21h
    mov si, offset INT_ID
    sub si, offset ROUT
    mov ax, es:[bx+si]
    cmp ax, 4040h
    jne EndCheck
    mov INT_LOADED, 1

EndCheck:
    pop si
    pop bx
    pop ax
    ret
CHECK_INT ENDP

WRITE PROC near
    push ax
    mov ah, 09h
    int 21h
    pop ax
    ret
WRITE ENDP

MAIN PROC FAR
    mov ax, DATA
    mov ds, ax
    mov KEEP_PSP, es

    call CHECK_INT

    mov ax, KEEP_PSP
    mov es, ax
    cmp byte ptr es:[81h+1], '/'
    jne WithoutUN
    cmp byte ptr es:[81h+2], 'u'
    jne WithoutUN
    cmp byte ptr es:[81h+3], 'n'
    jne WithoutUN
    mov dx, offset MESSAGE_INT_NOT_LOADED
    call WRITE
    call INT_UNLOAD
    jmp EndInt

WithoutUN:
    mov al, INT_LOADED
    cmp al, 1
    je EndInt
    mov dx, offset MESSAGE_INT_LOADED
    call WRITE
    call LOAD_INT
    jmp EndInt

EndInt:
    xor al, al
    mov ah, 4Ch

```

```
        int 21h  
MAIN ENDP  
CODE ENDS  
END MAIN
```