

**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. 7381

\_\_\_\_\_

Кортев Ю.В.

Преподаватель

\_\_\_\_\_

Ефремов М.А.

Санкт-Петербург

2019

### **Цель работы**

Построить обработчик прерываний сигналов таймера. Таким образом, управление будет передано функции, чья точка входа записана в соответствующий вектор прерывания.

### **Ход работы**

Разработан набор функций для выполнения поставленной задачи, функции приведены в табл. 1.

Таблица 1 - Описание функций.

<b>Название функции</b>	<b>Назначение</b>
GetCurs	Читать позицию и размер курсора
SetCurs	Установка позиции курсора
Set_Int	Установка резидентного прерывания
MAIN	Основная функция

Примеры работы программы отображены на рис. 1-4

```

C:\>LAB3_1.COM

Available memory: 648912 B
Extended memory: 15360 KB

  MCB Address  MCB Type  Owner      Size      Name
  016F         4D      0008        16
  0171         4D      0000        64
  0176         4D      0040       256
  0187         4D      0192       144
  0191         5A      0192    648912    LAB3_1

```

Рис. 1 - Проверка состояния памяти до запуска утилиты, при помощи LAB3\_1.COM.

```

C:\>LAB4.EXE
The int has Loaded.

C:\>LAB3_1.COM

Available memory: 647296 B
Extended memory: 15360 KB

  MCB Address  MCB Type  Owner      Size      Name
  016F         4D      0008        16
  0171         4D      0000        64
  0176         4D      0040       256
  0187         4D      0192       144
  0191         4D      0192     1440    LAB4
  01EC         4D      01F7       1144
  01F6         5A      01F7     6472

```

Рис. 2 - Загрузка утилиты lab4 в память и очередной вывод состояния памяти

```

C:\>LAB4.EXE /un
The int has been unloaded.

C:\>LAB3_1.COM

Available memory: 647296 B
Extended memory: 15360 KB

  MCB Address  MCB Type  Owner      Size      Name
  016F         4D      0008        16
  0171         4D      0000        64
  0176         4D      0040       256
  0187         4D      0192       144
  0191         4D      0192     1440    LAB4
  01EC         4D      01F7       1144
  01F6         5A      01F7     6472

```

Рис. 4 - Запуск программы lab4.exe с ключом выгрузки /un и последующий вызов LAB3\_1.COM.

## **Выводы.**

В процессе выполнения данной лабораторной работы был построен обработчик прерываний сигналов таймера. Код программы представлен в приложении А.

## **Ответы на контрольные вопросы.**

Как реализован механизм прерывания от часов?

Сначала сохраняется содержимое регистров, потом определяется источник прерывания, по номеру которого определяется смещение в таблице векторов прерывания, сохраняется в CS:IP, передаётся управление по адресу CS:IP и происходит выполнение обработчика, и в конце происходит возврат управления прерванной программе. Прерывания генерируются системным таймером.

Какого типа прерывания использовались в работе?

В программе использовались пользовательское прерывание по таймеру 1Ch, вектор прерывания 03h (используется отладчиками, чтобы перехватывать управление, когда программа достигает указанного пользователем адреса), 02h (посылает символ из DL на стандартный вывод) и программные прерывания 21h и 10h.

## ПРИЛОЖЕНИЕ А

### lr4.asm

```
AStack SEGMENT STACK
    DW 100h DUP(?)
AStack ENDS

DATA SEGMENT
    alreadyLoaded db 'The int has already been loaded.',0DH,0AH,'$'
    UnLoaded db 'The int has been unloaded.',0DH,0AH,'$'
    Loaded db 'The int has Loaded.',0DH,0AH,'$'
DATA ENDS

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

Print PROC NEAR
    push ax
    mov ah, 09h
    int 21h
    pop ax
    ret
Print ENDP
;
; Установка позиции курсора
setCurs PROC
    push ax
    push bx
    push cx
    mov ah,02h
    mov bh,0
    int 10h
    pop cx
    pop bx
    pop ax
    ret
setCurs ENDP

getCurs PROC
    push ax
    push bx
    push cx

    mov ah,03h ; читать позицию и размер курсора
    mov bh,00h ; вход: BH = видео страница
    int 10h ; выполнение

    pop cx
    pop bx
    pop ax
    ret
getCurs ENDP

ROUT PROC FAR
    jmp case

    KEEP_CS DW 0 ; для хранения сегмента
    KEEP_IP DW 0 ; и смещения прерывания
    KEEP_PSP DW 0
    VALUE db 0
    IDENT db '0000'
    COUNTER db '  Number of calls: 00000  '$'
    STACK    dw    64 dup (?)
    KEEP_SS DW 0
    KEEP_AX  dw ?
    KEEP_SP DW 0

case:
    mov KEEP_SS,ss
    mov KEEP_AX,ax
    mov KEEP_SP,sp
    mov ax,seg STACK_
    mov ss,ax
    mov sp,0
```

```

mov ax,KEEP_AX

push ax ; сохранение изменяемых регистров
push dx
push ds
push es
cmp VALUE,1
je ROUT_RES

call getCurs

push dx
mov dh,22
mov dl,45

call setCurs

push si
push cx
push ds
push ax
mov ax,SEG COUNTER
mov ds,ax
mov bx,offset COUNTER
add bx,22
mov si,3

case_next:
mov ah,[bx+si]
inc ah
cmp ah,58
jne _NEXT

mov ah,48
mov [bx+si],ah
dec si
cmp si,0
jne case_next

_NEXT:
mov [bx+si],ah
pop ds
pop si
pop bx
pop ax
push es
push bp
mov ax,SEG COUNTER
mov es,ax
mov ax,offset COUNTER
mov bp,ax
mov ah,13h
mov al,0
mov cx,30
mov bh,0
int 10h
pop bp
pop es
pop dx
call setCurs
jmp ROUT_END

; при выгрузке обработчика прерываний
ROUT_RES:
cli
mov ax,KEEP_CS
mov dx,KEEP_IP
mov ds,ax
mov ah,25h ; функция установки вектора прерывания на указанный адрес
mov al,1Ch ; номер вектора
int 21h ; восстанавливаем вектор
mov es,KEEP_PSP
mov es,es:[2Ch]
mov ah,49h
int 21h
mov es,KEEP_PSP

```

```

mov ah,49h
int 21h
sti

```

ROUT\_END:

```

pop es
pop ds
pop dx
pop ax ; восстановление регистров
mov ax,KEEP_SS
mov ss,ax
mov sp,KEEP_SP
mov ax,KEEP_AX
iret

```

ROUT ENDP

Check PROC

```

mov ah,35h ; функция получения вектора
mov al,1Ch ; номер вектора
int 21h
mov si,offset IDENT
sub si,offset ROUT
mov ax,'00'
cmp ax,es:[bx+si]
jne UnLoad
cmp ax,es:[bx+si+2]
je Load

```

UnLoad:

```

call Set_Int
mov dx,offset LAST_BYTE ; размер в байтах от начала
mov cl,4 ; перевод в параграфы
shr dx,cl
inc dx ; размер в параграфах
add dx,CODE
sub dx,KEEP_PSP
xor al,al
mov ah,31h
int 21h

```

Load:

```

push es
push ax
mov ax,KEEP_PSP
mov es,ax
cmp byte ptr ES:[82h], '/'
jne BACK
cmp byte ptr ES:[83h], 'u'
jne BACK
cmp byte ptr ES:[84h], 'n'
je UnLoad_

```

BACK:

```

pop ax
pop es
mov dx,offset alreadyLoaded
call Print
ret

```

UnLoad\_ :

```

pop ax
pop es
mov byte ptr ES:[BX+SI+10],1
mov dx,offset UnLoaded
call Print
ret

```

Check ENDP

Set\_Int PROC

```

push dx
push ds
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
mov dx,offset ROUT ; смещение для процедуры в DX

```

```

mov ax,seg ROUT ; сегмент процедуры
mov ds,AX ; помещаем в DS
mov ah,25h ; функция установки вектора прерывания на указанный адрес
mov al,1Ch ; номер вектора
int 21h ; меняем прерывание
pop ds
mov dx,offset Loaded
call Print
pop dx
ret
Set_Int ENDP

MAIN:
mov ax,DATA
mov ds,ax
mov KEEP_PSP,es

call Check

xor al,al
mov ah,4Ch
int 21H

LAST_BYTE:

CODE ENDS
END MAIN

```