

**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**

Студент гр. 7381

\_\_\_\_\_

Габов Е. С.

Преподаватель

\_\_\_\_\_

Ефремов М. А.

Санкт-Петербург

2019

### Цель работы.

Построение обработчика прерываний сигналов таймера.

### Описание функций.

Имя	Описание
UNOAD_INT	Восстанавливает сохраненный вектор прерывания
CHECK_INT	Проверяет, установлено ли прерывание
IS_UNLOAD	Проверяет необходимость выгрузки прерывания из памяти
LOAD_INT	Загружает прерывание резидентно в память
INTER	Обработчик прерывания
setCurs	Устанавливает курсор в строку dh и столбец dl
getCurs	Возвращает положение курсора в dx
outputAL	Вывод символа в AL в текущее положение курсора

### Описание структур данных.

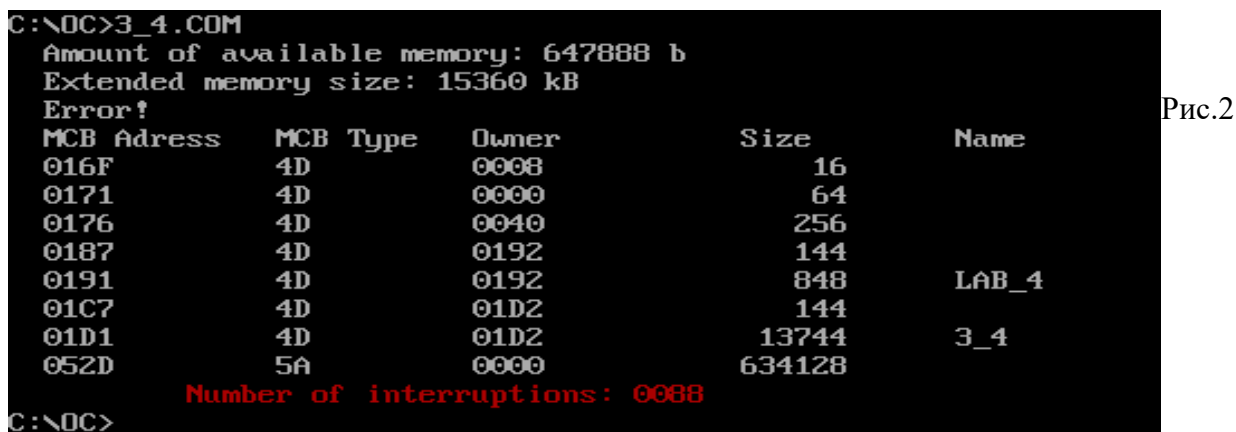
Имя	Тип	Назначение
INTER_ALREADY	db	Строка "Interrupt is already loaded"
INTER_UNLOADED	db	Строка "Interrupt is unloaded"
INTER_LOADED	db	Строка "Interrupt is loaded"
INTER_NOT_LOADED	db	Строка "Interrupt is not loaded"
KEY_WORD	db	Строка "MY_INT"
KEEP_CS	dw	Сохранённое значение CS
KEEP_IP	dw	Сохранённое значение IP
KEEP_PSP	dw	Сохранённое значение адреса PSP
COUNT	db	Счётчик

## Тестирование.

### 1. Загрузка прерывания.



Рис.1 Результат работы программы lab4.exe.



Результат работы программы lab3\_1.com.

### 2. Повторный запуск программы.

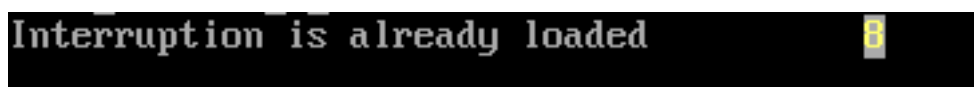


Рис.3 Результат работы программы lab4.exe.

### 3. Возврат стандартного обработчика прерывания.

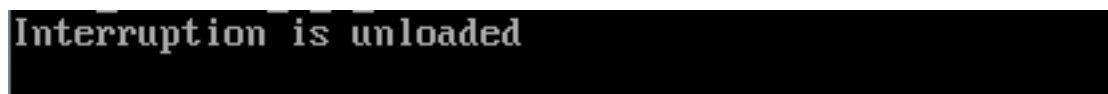


Рис.4 Результат работы программы lab4.exe с ключом "/un".

## Ответы на контрольные вопросы.

### 1. Как реализован механизм прерывания от часов?

Ответ: Это аппаратное прерывание, обработчик которого (1ch) вызывается 18 раз в секунду.

### 2. Какого типа прерывания использовались в работе?

Ответ: В работе использовались аппаратные (прерывание от часов) и программные (int 21h).

### **Вывод.**

В ходе выполнения лабораторной работы было изучено создание резидентных программ, а также построен обработчик прерывания от часов.

## **Приложение А.**

### **lab4.asm**

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

;=====

DATA SEGMENT
    INTER_ALREADY db 'Interrupt is already loaded',13,10,36
    INTER_UNLOADED db 'Interrupt is unloaded',13,10,36
    INTER_LOADED db 'Interrupt is loaded',13,10,36
    INTER_NOT_LOADED db 'Interrupt is not loaded',13,10,36

DATA ENDS

;=====

CODE SEGMENT
```

```
ASSUME CS:CODE, DS:DATA, SS:STACK
```

```
;-----
```

```
setCurs PROC
```

```
    push ax
    push bx
    push dx
    push cx
    mov ah,02h
    mov bh,0
```

```
    int 10h
    pop cx
    pop dx
    pop bx
    pop ax
    ret
```

```
setCurs ENDP
```

```
;-----
```

```
getCurs PROC
```

```
    push ax
    push bx
    push cx
    mov ah,03h
    mov bh,0
    int 10h
    pop cx
    pop bx
    pop ax
```

```
getCurs ENDP
```

```
;-----
```

```
outputAL PROC
```

```
    push ax
    push bx
    push cx
    mov ah,09h
```

```

        mov bh,0
        mov cx,1
        int 10h
        pop cx
        pop bx
        pop ax
        ret
outputAL ENDP
;-----
INTER PROC FAR
        jmp INT_CODE
        KEY_WORD db 'MY_INT'
        KEEP_CS DW 0
        KEEP_IP DW 0
        KEEP_PSP DW 0
        COUNT db 0
        INT_CODE:
        push ax
        push dx
        push ds
        push es

        call getCurs
        push dx

        mov dh,22
        mov dl,40
        call setCurs

        cmp count,10
        jle next_it
        mov count,0
next_it:

```

```

    mov al,count
    add al,30h
    call outputAL
    inc count
    pop dx
    call setCurs

    pop es
    pop ds
    pop dx
    pop ax
    mov al,20h
    out 20h,al
    iret
end_inter:
INTER ENDP
;-----
LOAD_INT PROC near
    push ax
    push cx
    push bx
    push dx
    push ds
    mov ah, 35h
    mov al, 1Ch
    int 21h

    mov KEEP_IP, bx
    mov KEEP_CS, es
    mov ax, SEG INTER
    mov dx, OFFSET INTER
    mov ds,ax
    mov ah, 25h

```

```

mov al, 1Ch
int 21h

mov dx, OFFSET end_inter
mov cl, 4
shr dx, cl
inc dx
add dx, CODE
sub dx, KEEP_PSP
mov ah, 31h
int 21h
pop ds
pop dx
pop bx
pop cx
pop ax
ret
LOAD_INT ENDP
;-----
IS_UNLOAD PROC near
    push di
    mov di, 81h
    cmp byte ptr [di+0], ' '
    jne bad_key
    cmp byte ptr [di+1], '/'
    jne bad_key
    cmp byte ptr [di+2], 'u'
    jne bad_key
    cmp byte ptr [di+3], 'n'
    jne bad_key
    cmp byte ptr [di+4], 0Dh
    jne bad_key
    cmp byte ptr [di+5], 0h
    jne bad_key
    pop di

```



```

        mov al,1
        ret
bad_key:
        pop di
        mov al,0
        ret
IS_UNLOAD ENDP
;-----
CHECK_INT PROC near
        push ax
        push bx
        push es
        mov ah, 35h
        mov al, 1ch
        int 21h
        mov ax, OFFSET KEY_WORD
        sub ax, OFFSET INTER
        add bx, ax
        mov si,bx
        push ds
        mov ax,es
        mov ds,ax
        cmp [si], 'YM'
        jne false
        add si,2
        cmp [si], 'I_'
        jne false
        add si,2
        cmp [si], 'TN'
        jne false
        pop ax
        mov ds,ax
        pop es
        pop bx
        pop ax

```

```

        mov al,1
        ret

false:
        pop ax
        mov ds,ax
        pop es
        pop bx
        pop ax
        mov al,0
        ret
CHECK_INT ENDP
;-----
UNLOAD_INT PROC near
        push ax
        push dx
        mov ah, 35h
        mov al, 1Ch
        int 21h
        cli
        push ds
        mov dx, es:KEEP_IP
        mov ax, es:KEEP_CS
        mov ds, ax
        mov ah, 25h
        mov al, 1Ch
        int 21h
        pop ds

        mov es, KEEP_PSP
        push es
        mov es, es:[2Ch]
        mov ah, 49h
        int 21h

```

```

    pop es
    mov ah, 49h
    int 21h

    sti
    pop dx
    pop ax
    ret
UNLOAD_INT ENDP
;-----
MAIN PROC FAR
    push ds
    sub ax,ax
    push ax
    mov KEEP_PSP, es

    call CHECK_INT
    cmp al, 1
    je int_loaded

    call IS_UNLOAD
    cmp al, 1
    je int_not_loaded

    mov dx, offset INTER_LOADED
    mov ax, DATA
    mov ds, ax
    mov ah, 9
    int 21h
    call LOAD_INT
    jmp end_prog

int_not_loaded:

```

```

    mov dx, offset INTER_NOT_LOADED
    mov ax, DATA
mov ds, ax
    mov ah, 9
    int 21h
    jmp end_prog

int_loaded:
    call IS_UNLOAD
    cmp al, 1
    je need_to_unload

    mov dx, offset INTER_ALREADY
    mov ax, DATA
mov ds, ax
    mov ah, 9
    int 21h
    jmp end_prog

need_to_unload:
    call UNLOAD_INT
    mov dx, offset INTER_UNLOADED
    mov ax, DATA
mov ds, ax
    mov ah, 9
    int 21h
    jmp end_prog

end_prog:
    xor al, al
    mov ah, 4ch
    int 21h

MAIN ENDP

```

CODE ENDS

END MAIN