

**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. 9382

Голубева В.П.

Преподаватель

Ефремов М.А.

Санкт-Петербург

2021

### **Цель работы.**

В архитектуре компьютера существуют стандартные прерывания, за которыми закреплены определенные вектора прерываний. Вектор прерываний хранит адрес подпрограммы обработчика прерываний. При возникновении прерывания, аппаратура компьютера передает управление по соответствующему адресу вектора прерывания. Обработчик прерываний получает управление и выполняет соответствующие действия.

В лабораторной работе № 4 предлагается построить обработчик прерываний сигналов таймера. Эти сигналы генерируются аппаратурой через определенные интервалы времени и, при возникновении такого сигнала, возникает прерывание с определенным значением вектора. Таким образом, управление будет передано функции, чья точка входа записана в соответствующий вектор прерывания.

### **Задание.**

Шаг 1. Для выполнения лабораторной работы необходимо написать и отладить программный модуль типа .EXE, который выполняет следующие функции:

1) Проверяет, установлено ли пользовательское прерывание с вектором lCh.

2) Устанавливает резидентную функцию для обработки прерывания и настраивает вектор прерываний, если прерывание не установлено, и осуществляется 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.

Для того, чтобы проверить установку прерывания, можно поступить следующим образом. Прочитать адрес, записанный в векторе прерывания. Предположим, что этот адрес указывает на точку входа в установленный резидент. На определенном, известном смещении в теле резидента располагается сигнатура, некоторый код, который идентифицирует резидент. Сравнив известное значение сигнатуры с реальным кодом, находящимся в резиденте, можно определить, установлен ли резидент. Если значения совпадают, то резидент установлен. Длину кода сигнатуры должна быть достаточной, чтобы сделать случайное совпадение маловероятным.

Программа должна содержать код устанавливаемого прерывания в виде удаленной процедуры. Этот код будет работать после установки при возникновении прерывания. Он должен выполнять следующие функции:

- 1) Сохранить значения регистров в стеке при входе и восстановить их при выходе.
- 2) При выполнении тела процедуры накапливать общее суммарное число прерываний и выводить на экран. Для вывода на экран следует использовать прерывание int 10h, которое позволяет непосредственно выводить информацию на экран.

Шаг 2. Запустите отлаженную программу и убедитесь, что резидентный обработчик прерывания lCh установлен. Работа прерывания должна отображаться на экране, а также необходимо проверить размещение прерывания в памяти. Для этого запустите программу ЛР 3, которая отображает карту памяти в виде списка блоков МСВ. Полученные результаты поместите в отчет.

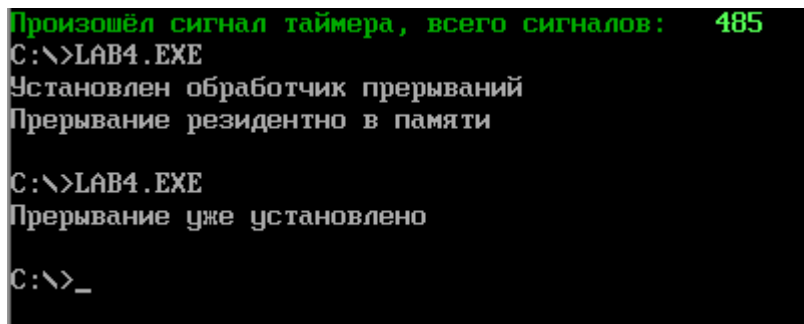
Шаг 3. Запустите отлаженную программу еще раз и убедитесь, что программа определяет установленный обработчик прерываний. Полученные результаты поместите в отчет.

Шаг 4. Запустите отлаженную программу с ключом выгрузки и убедитесь, что резидентный обработчик прерывания выгружен, то есть сообщения на экран не выводятся, а память, занятая резидентом освобождена. Для того также следует запустить программу ЛР 3. Полученные результаты поместите в отчет.

Шаг 5. Ответьте на контрольные вопросы.

### **Выполнение работы.**

Был создан exe модуль, который менял вектор прерывания у таймера на написанное. В ходе его работы выводилась информация о количестве сигналов таймера с момента запуска программы. Программа оставалась резидентной в памяти, при попытке повторной загрузки в память выводилось сообщение о том, что прерывание уже установлено(для проверки загрузки сравнивался сигнатура у обработчика прерывания пользовательской программы и той, что загружена в память). Результаты работы можно посмотреть в Рисунке 1.



```
Произошёл сигнал таймера, всего сигналов: 485
C:\>LAB4.EXE
Установлен обработчик прерываний
Прерывание резидентно в памяти

C:\>LAB4.EXE
Прерывание уже установлено

C:\>_
```

Рисунок 1. Результат загрузки программы в память

Для того, чтобы проверить, что программа находится в памяти, использовалась программа из лабораторной работы номер три, которая выводит цепочку mcb-блоков. Запустим эту программу и увидим, что обработчик прерывания находится в памяти. Результаты можно посмотреть в Рисунке 2.

```
Количество доступной памяти в байтах: 647344
Размер расширенной памяти Кб: 15360
```

```
0008h - участок принадлежит MS DOS
Размер участка в байтах: 16
Последовательность символов: [] [] [] [] [] [] [] []
-----
```

```
0000h - свободный участок
Размер участка в байтах: 64
Последовательность символов: [] [] [] [] [] [] [] []
-----
```

```
Пользовательский участок
Размер участка в байтах: 256
Последовательность символов: [] [] [] [] [] [] [] []
-----
```

```
Пользовательский участок
Размер участка в байтах: 144
Последовательность символов: [] [] [] [] [] [] [] []
-----
```

```
Пользовательский участок
Размер участка в байтах: 1392
Последовательность символов: LAB4[] [] [] []
-----
```

```
Пользовательский участок
Размер участка в байтах: 144
Последовательность символов: [] [] [] [] [] [] [] []
-----
```

```
Пользовательский участок
Размер участка в байтах: 647344
Последовательность символов: LAB3_1[] []
-----
```

Рисунок 2. Результат работы программы LAB\_1.COM из лабораторной работы номер 3

Теперь выгрузим обработчик из памяти и снова запустим программу лабораторной работы номер три. Результаты можно посмотреть в Рисунке 3 и Рисунке 4.

```

C:\>LAB4.EXE
Установлен обработчик прерываний
Прерывание резидентно в памяти

C:\>LAB4.EXE
Прерывание уже установлено

C:\>LAB4.EXE /un
Прерывание уже установлено
Требуется выгрузить обработчик прерывания из памяти
Обработчик прерываний выгружен из памяти

```

Рисунок 3. Выгрузка обработчика прерывания из памяти

---

```

Количество доступной памяти в байтах: 648912
Размер расширенной памяти Кб: 15360

```

```

0008h - участок принадлежит MS DOS
Размер участка в байтах: 16
Последовательность символов: [] [] [] [] [] [] []
-----

```

```

0000h - свободный участок
Размер участка в байтах: 64
Последовательность символов: [] [] [] [] [] [] []
-----

```

```

Пользовательский участок
Размер участка в байтах: 256
Последовательность символов: [] [] [] [] [] [] []
-----

```

```

Пользовательский участок
Размер участка в байтах: 144
Последовательность символов: [] [] [] [] [] [] []
-----

```

```

Пользовательский участок
Размер участка в байтах: 648912
Последовательность символов: LAB3_1[] []
-----

```

Рисунок 4. Демонстрация освобождения памяти после выгрузки программы

Таким образом можно убедиться, что обработчик прерывания был успешно выгружен из памяти.

## **Ответы на контрольные вопросы**

Контрольные вопросы по лабораторной работе №4

1) Как реализован механизм прерывания от часов?

Ответ: когда происходит сигнал часов, то сохраняется состояние регистров для того, чтобы вернуться в текущую программу, затем определяется источник прерывания, из вектора прерывания считываются CS и IP обработчика прерывания, прерывание обрабатывается, затем управление возвращается прерванной программе

2) Какого типа прерывания использовались в работе?

Ответ: аппаратные и пользовательские

## **Выводы.**

Был изучен механизм обработки прерываний в DOS, написана программа, которая заменяет текущий обработчик прерываний от таймера на пользовательский.

## ПРИЛОЖЕНИЕ А

### ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: lab4.asm

```
stacks segment  STACK
    db 256 dup(?)
stacks ends

data segment
; количество сигналов таймера
isBoot dw 0
isUnBoot dw 0
memAdrPsp dw 0
vect dw 0, 0

str_in_no db 'Прерывание не установлено', 0AH, 0DH, '$'
str_int_mem db 'Прерывание резидентно в памяти', 0AH, 0DH, '$'
str_in_yes db 'Установлен обработчик прерываний', 0AH, 0DH, '$'
str_unb db 'Обработчик прерываний выгружен из памяти', 0AH, 0DH,
'$'
str_need_unboot db 'Требуется выгрузить обработчик прерывания из
памяти', 0AH, 0DH, '$'
str_in_already db 'Прерывание уже установлено', 0DH, 0AH, '$'
str_ax db 'AX=      ', 0DH, 0AH, '$'
len equ $ - str_ax

data ends

code segment

    ASSUME CS:code, DS:data, SS:stacks

TETR_TO_HEX PROC near
    and AL, 0Fh
```



```

        cmp AL, 09
        jbe NEXT
        add AL, 07
NEXT:   add AL, 30h
        ret
TETR_TO_HEX ENDP
;-----
BYTE_TO_HEX PROC near
;byte AL translate in two symbols on 16cc numbers in AX
        push CX
        mov AH,AL
        call TETR_TO_HEX
        xchg AL,AH
        mov CL, 4
        shr AL,CL
        call TETR_TO_HEX
        pop CX
        ret
BYTE_TO_HEX ENDP
;-----
WRD_TO_HEX PROC near
;translate in 16cc a 16 discharge number
;in AL - number, DI - the address of the last symbol
        push BX
        mov BH,AH
        call BYTE_TO_HEX
        mov [DI],AH
        dec DI
        mov [DI],AL
        dec DI
        mov AL,BH
        call BYTE_TO_HEX
        mov [DI],AH
        dec DI
        mov [DI],AL
        pop BX
        ret

```

```
WRD_TO_HEX ENDP
```

```
interr proc far
```

```
jmp run
```

```
signature dw 0ff00h, 0ffffh
```

```
count dw 1
```

```
str_time db 'Произошёл сигнал таймера, всего сигналов: ',  
'$';42
```

```
run:
```

```
push ax
```

```
push bx
```

```
push cx
```

```
push dx
```

```
;mem a cursor position
```

```
mov ah, 03
```

```
mov bh, 00
```

```
int 10h
```

```
push dx;-----cursor position in dx
```

```
;print a string of information
```

```
push ds
```

```
push es
```

```
mov ax, seg interr
```

```
mov ds, ax
```

```
mov dx, offset str_time
```

```
mov bp, dx
```

```
;es - указывает на начало процедуры
```

```
mov ax, ds
```

```
mov es, ax
```

```

    ;print a string
    mov ah,13h
    mov al,1 ;sub function code
    ;1 = use attribute in BL; leave cursor an end of string
    mov bh,0
    mov bl, 2
    mov dh,0;
    mov dl,0
    mov cx, 42
    int 10h

    pop es
    pop ds

;-----
    inc count
    cmp count, 0ffffh
    jne ifn
    mov count, 1

ifn:

    mov cl, 43
    mov [di], cl

    mov ax, count
    ;----preparing ax
    mov bx,1h
    mul bx

    ;dx:ax - number
    mov bx,0Ah
    xor cx,cx
    divis:
    div bx
    push dx

```

```

inc cx
xor dx,dx
cmp ax,0
jne divis

print_simb:

pop dx

push cx
push dx

;-----change cursor position
mov cl, [di]
inc cl
mov [di], cl

;put symbol into al

add dl, 30h
;print a symbol in dl
mov al, dl

;set curs
mov ah, 02
mov bh, 00
mov dh, 0
mov dl, [di]
int 10h

pop dx
;in dl - digit for print

mov ah, 09h    ;писать символ в текущей позиции курсора
mov bh, 0      ;номер видео страницы
mov cx, 1      ;число экземпляров символа для записи
int 10h        ;выполнить функцию

```

```

    pop cx
    loop print_simb

;return curs postition

    pop dx
    mov ah, 02
    mov bh, 00
    int 10h

;-----end

    mov al, 20h
    out 20h, al

    pop dx
    pop cx
    pop bx
    pop ax

    iret
interr endp

;-----
resident proc
    ;оставляем процедуру прерывания резидентной
    ;AH - номер функции 31h;
    ;AL - код завершения программы;
    ;DX - размер памяти в параграфах, требуемый резидентной
программе.
    push ax
    push dx
    push cx
    push bx

```

```
mov dx, offset str_int_mem
mov ah, 09h
int 21h
```

```
mov AX, memAdrPsp
mov BX, seg code
sub BX, AX
mov DX, offset eeend; размер в байтах от начала сегмента
mov CL, 4 ; перевод в параграфы
shr DX, CL
inc DX ; размер в параграфах
add DX, BX
mov AH, 31h
mov al, 00h
int 21h
```

```
pop bx
pop cx
pop dx
pop ax
```

```
ret
resident endp
```

```
;-----
setInterr proc
```

```
push ax
push dx
push bx
```

```
mov dx, offset str_in_yes
mov ah, 09h
int 21h
```

```

; в программе при загрузке обработчика прерывания
MOV AH, 35H ; функция получения вектора
MOV AL, 1CH ; номер вектора
INT 21H

;сохраняем вектор исходного обработчика прерывания таймера
mov word ptr vect+2, es
mov word ptr vect, bx

;теперь устанавливаем на его место наш обработчик

PUSH DS
MOV DX, OFFSET interr ;смещение для процедуры в dx
;кладём в ds сегмент процедуры
MOV AX, SEG interr ;сегмент процедуры
MOV DS, AX ;помещаем в ds

MOV AH, 25H ;функция установки вектора прерывания
MOV AL, 1CH ;номер вектора

INT 21H ;меняем прерывание
POP DS

pop bx
pop dx
pop ax

ret
setInterr endp
;-----
isBootFunc proc
    push ax
    push bx
    push dx
    push es

```

```

    mov ax, 351Ch ;
    int 21h
    ;bx, es
    ;-----
    add bx, offset signature - offset interr
    mov dx, es:[bx];---signature

    mov ax, 0ff00h
    cmp dx, ax
    jne e_i_s
    mov dx, es:[bx+2];---signature

    mov ax, 0ffffh
    cmp dx, ax
    je ad
    jmp e_i_s

ad:

    pop es
    mov isBoot, 1
    mov dx, offset str_in_already
    mov ah, 09h
    int 21h
    jmp ex_
e_i_s:
    ;pop ds
    pop es
ex_:
    pop dx
    pop bx
    pop ax
    ret
isBootFunc endp

;-----

```



```

isUnBootFunc proc
    push es
    push ax
    push dx
    push cx

    mov ax, memAdrPsp
    mov es, ax
    mov cl, es:[80h]

    cmp cl, 4h
    jl non

    mov dl, es:[81h]
    cmp dl, ' '
    jne non

    mov dl, es:[81h+1]
    cmp dl, '/'
    jne non

    mov dl, es:[81h+2h]
    cmp dl, 'u'
    jne non

    mov dl, es:[81h+3h]
    cmp dl, 'n'
    jne non

    mov isUnBoot, 1h

    mov dx, offset str_need_unboot
    mov ah, 09h
    int 21h

non:
    pop cx

```

```

        pop dx
        pop ax
        pop es
        ret
isUnBootFunc endp
;-----

UnBootFunc proc

        push dx
        push ax
        push es
        push bx
        mov dx, offset str_unb
        mov ah, 09h
        int 21h

        mov ax, seg data
        mov bx, seg code

        sub bx, ax

        push es
        push bx

        mov ah, 35h
        mov al, 1Ch;
        int 21h
        ;es, bx in resident
        pop bx
        push es;---memAdrCode
        mov ax, es

        sub ax, bx
        mov es, ax

        mov bx, offset vect; oTH ds

```

```

; в программе при выгрузке обработчика прерываний
CLI
PUSH DS

mov dx, es:[bx];---bx
mov ax, es:[bx+2];--es
MOV DS, AX

MOV AH, 25H ;устанавливаем вектор прерывания
MOV AL, 1CH
INT 21H ; восстанавливаем вектор
;pop es
pop ds

;---free memory
pop es
mov bx, es;---code in resident
pop es
mov ax, es;---now es

mov dx, seg code;--seg adress now program
sub dx, ax ;---defference
sub bx, dx;--es in resident
mov es, bx

push es

mov ax, es:[2Ch]
mov es, ax

mov ah, 49h
int 21h

pop es

```

```

        mov ah, 49h
        int 21h

        pop bx
        pop es
        pop ax
        pop dx
        STI

        ret
UnBootFunc endp
;-----

BEGIN proc far

        mov AX, data
        mov DS, AX

        mov bx, es
        mov memAdrPsp, bx

        call isBootFunc
        call isUnBootFunc

        cmp isBoot, 1h
        je mayunboot
;-----
boot:
        call setInterr
        call resident
;-----
mayunboot:
        cmp isUnBoot, 1h
        jne end_pr
        call UnBootFunc

```

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

```
end_pr:
```

```
    xor  AL,AL
```

```
    mov  AH,4Ch
```

```
    int  21H
```

```
begin endp
```

```
eeend:
```

```
code    ENDS
```

```
        END begin
```