

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

ЛАБОРАТОРНАЯ РАБОТА № 4
по дисциплине «Операционные системы»
Тема: Обработка стандартных прерываний

Студент гр. 9381

Матвеев А. Н.

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 передаёт управление и выполняет соответствующие действия.

В данной лабораторной работе предлагается построить обработчик прерываний сигналов таймера. Эти сигналы генерируются аппаратурой через определённые интервалы времени и, при возникновении такого сигнала, возникает прерывание с определённым значением вектора. Таким образом, управление будет передано функции, чья точка входа записана в соответствующий вектор прерывания.

Ход работы.

1) Написан и отлажен программный модуль типа .EXE, который выполняет следующие функции:

- ❖ Проверяет, установлено ли пользовательское прерывание с вектором 1Ch.
- ❖ Устанавливает резидентную функцию для обработки прерывания и настраивает вектор прерываний, если прерывание не установлено, и осуществляется выход по функции 4Ch прерывания int 21h.
- ❖ Если прерывание установлено, то выводится соответствующее сообщение и осуществляется выход по функции 4Ch прерывания int 21h.
- ❖ 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.

Сначала был написан код на Ассемблере, затем при помощи компилятора MASM.EXE и компоновщика LINK.EXE был получен модуль lab4.exe (см. рис. 1). Далее была произведена отладка.

```
Z:\>F:

F:\>masm lab4.asm
Microsoft (R) Macro Assembler Version 5.10
Copyright (C) Microsoft Corp 1981, 1988. All rights reserved.

Object filename [lab4.OBJ]:
Source listing [NUL.LST]:
Cross-reference [NUL.CRF]:

    49892 + 449176 Bytes symbol space free

    0 Warning Errors
    0 Severe Errors

F:\>link lab4.obj

Microsoft (R) Overlay Linker Version 3.64
Copyright (C) Microsoft Corp 1983-1988. All rights reserved.

Run File [LAB4.EXE]:
List File [NUL.MAP]:
Libraries [.LIB]:

F:\>
```

Рис. 1. Получение lab4.exe.

2) Затем программа была запущена. (см. рис. 2). Работа прерывания отображается на экране. Нетрудно удостовериться, что резидентный обработчик прерывания 1Ch установлен. Чтобы проверить размещение прерывания в памяти, был запущен .com файл из предыдущей лабораторной работы, в котором отображается карта памяти в виде списка блоков МСВ. (см. рис. 3). Из последнего рисунка видно, что МСВ-блок программы располагается 5-м по порядку.

```

Count of interruptions: 0177
F:\>
F:\>
F:\>
F:\>
F:\>
F:\>
F:\>
F:\>
F:\>
F:\>lab4.exe
INTERRUPTION IS LOADING AT THIS MOMENT

```

Рис. 2. Запуск программы.

```

Count of interruptions: 1010
F:\>
F:\>
F:\>lab4.exe
INTERRUPTION IS LOADING AT THIS MOMENT
F:\>lab3.com
Accesible memory: 648176 bytes
Extended memory: 15360 kilobytes

MCB type: 4D    PSP address: 0008    Size: 16 bytes    SC/SD:
MCB type: 4D    PSP address: 0000    Size: 64 bytes    SC/SD:
MCB type: 4D    PSP address: 0040    Size: 256 bytes    SC/SD:
MCB type: 4D    PSP address: 0192    Size: 144 bytes    SC/SD:
MCB type: 4D    PSP address: 0192    Size: 560 bytes    SC/SD: LAB4
MCB type: 4D    PSP address: 01C0    Size: 144 bytes    SC/SD:
MCB type: 4D    PSP address: 01C0    Size: 768 bytes    SC/SD: LAB3
MCB type: 5A    PSP address: 0000    Size: 647392 bytes    SC/SD: < tmm
F:\>

```

Рис. 3. Запуск lab3.com.

3) Отлаженная программа была запущена повторно; Убедился, что программа определяет установленный обработчик прерываний. На экран вывелось сообщение о том, что прерывание уже загружено в память (см. рис. 4).

```

F:\>
Count of interruptions: 4111
F:\>
F:\>lab4.exe
INTERRUPTION IS LOADING AT THIS MOMENT
F:\>lab3.com
Accessible memory: 648176 bytes
Extended memory: 15360 kilobytes

MCB type: 4D PSP address: 0008 Size: 16 bytes SC/SD:
MCB type: 4D PSP address: 0000 Size: 64 bytes SC/SD:
MCB type: 4D PSP address: 0040 Size: 256 bytes SC/SD:
MCB type: 4D PSP address: 0192 Size: 144 bytes SC/SD:
MCB type: 4D PSP address: 0192 Size: 560 bytes SC/SD: LAB4
MCB type: 4D PSP address: 01C0 Size: 144 bytes SC/SD:
MCB type: 4D PSP address: 01C0 Size: 768 bytes SC/SD: LAB3
MCB type: 5A PSP address: 0000 Size: 647392 bytes SC/SD: < tmm

F:\>lab4.exe
INTERRUPTION IS ALREADY LOAD

```

Рис. 4. Повторный запуск программы.

4) Запустил программу с ключом выгрузки (параметр /un, смотреть рис. 5) и удостоверился, что резидентный обработчик прерывания выгружен, т.е. сообщения в консоль не выводятся, а память, занятая резидентом освобождена. Для проверки освобождения памяти, занятой резидентом, снова был запущен исполняемый .com файл ЛР3 (см. рис. 6). Из последнего рисунка видно, что MCB-блок программы lab4.exe уже отсутствует в списке.

```

F:\>lab4.exe
INTERRUPTION IS LOADING AT THIS MOMENT

F:\>lab3.com
Accesible memory: 648176 bytes
Extended memory: 15360 kilobytes

MCB type: 4D    PSP address: 0008    Size: 16 bytes    SC/SD:
MCB type: 4D    PSP address: 0000    Size: 64 bytes    SC/SD:
MCB type: 4D    PSP address: 0040    Size: 256 bytes    SC/SD:
MCB type: 4D    PSP address: 0192    Size: 144 bytes    SC/SD:
MCB type: 4D    PSP address: 0192    Size: 560 bytes    SC/SD: LAB4
MCB type: 4D    PSP address: 01C0    Size: 144 bytes    SC/SD:
MCB type: 4D    PSP address: 01C0    Size: 768 bytes    SC/SD: LAB3
MCB type: 5A    PSP address: 0000    Size: 647392 bytes SC/SD:  < t↑un

F:\>lab4.exe
INTERRUPTION IS ALREADY LOAD

F:\>lab4.exe /un
INTERRUPTION RESTORED

```

Рис. 5. Запуск lab4.exe с ключом выгрузки.

```

F:\>lab4.exe /un
INTERRUPTION RESTORED

F:\>lab3.com
Accesible memory: 648912 bytes
Extended memory: 15360 kilobytes

MCB type: 4D    PSP address: 0008    Size: 16 bytes    SC/SD:
MCB type: 4D    PSP address: 0000    Size: 64 bytes    SC/SD:
MCB type: 4D    PSP address: 0040    Size: 256 bytes    SC/SD:
MCB type: 4D    PSP address: 0192    Size: 144 bytes    SC/SD:
MCB type: 4D    PSP address: 0192    Size: 768 bytes    SC/SD: LAB3
MCB type: 5A    PSP address: 0000    Size: 648128 bytes SC/SD:  ||Eh

F:\>_

```

Рис. 6. Повторный запуск lab3.com для проверки освобождения памяти, занятой резидентом.

Сведения о функциях и структурах данных.

Программа использует процедуры и переменные, приведенные в таблицах ниже.

Названия процедур	Описание
INT_COUNT_FUNC	Выводит количество вызванных прерываний
CHECK_INTSET	Проверяет установку созданного вектора прерывания

CHECK_UN	Проверяет наличие параметра /un и возвращает, значение, по которому в функции MAIN происходит загрузка/выгрузка прерываний
REDEF_INT	Устанавливает новые обработчики прерывания (переопределяет) при помощи функции 25h прерывания int 21h
RESTORE_INT	восстанавливает сохраненные обработчики прерываний и выгружает резидентную функцию
WRITE	Выводит строку в консоль
MAIN	Главная процедура

Таблица 1. Процедуры

Название переменных и массивов символов	Назначение
PSP_1 (слово)	Хранит предыдущее значение ES до того, как программа была оставлена резидентной в памяти
KEEP_CS (слово)	Хранит сегмент прерывания
KEEP_IP (слово)	Хранит смещение прерывания
INT_COUNT_VAL (слово)	Хранит количество вызванных прерываний
COUNT_MES (массив байт)	Текст: 'COUNT OF INTERRUPTIONS: 0000'
RESTORED_MES (массив байт)	Текст: 'INTERRUPTION RESTORED'
ALREADY_LOAD_MES (массив байт)	Текст: 'INTERRUPTION IS ALREADY LOAD'
LOADING_MES (массив байт)	Текст: 'INTERRUPTION IS LOADING AT THIS MOMENT'

NOTLOAD_MES (массив байт)	Текст: 'INTERRUPTION NOT LOAD'
------------------------------	--------------------------------

Таблица 2. Переменные и массивы символов.

Результаты исследования проблем.

1. Как реализован механизм прерывания от часов?

Ответ:

- 1) поступает сигнал прерывания (периодично);
- 2) сохраняются значения регистров;
- 3) по номеру источника прерывания в таблице векторов устанавливается смещение;
- 4) сохраняются адреса: 2 байта в IP и 2 байта в CS;
- 5) по сохраненному адресу выполняется прерывание;
- 6) данные прерванного процесса восстанавливаются;
- 7) Прерванной программе возвращается управление.

2. Какого типа прерывания использовались в работе?

Ответ:

- 1) аппаратные прерывания (прерывание от часов - 1Ch);
- 2) программные прерывания (функций BIOS (10h) и функций DOS (21h)).

Заключение.

В ходе выполнения работы была написана программа обработчика прерывания от сигналов таймера и освоена техника установки резидентной программы в память и её выгрузка из памяти. Помимо этого была более подробно изучена обработка стандартных прерываний.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД

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

```
MY_STACK SEGMENT STACK
```

```
    DW 64 DUP(?)
```

```
MY_STACK ENDS
```

```
CODE SEGMENT
```

```
INT_COUNT_FUNC PROC FAR ; обработчик прерываний. печатает количество  
вызванных прерываний
```

```
    jmp PROC_CODE
```

```
    KEEP_CS dw 0 ;3 ; для хранения сегмента
```

```
    KEEP_IP dw 0 ;5 и смещения прерывания
```

```
    PSP0 dw 0 ;7
```

```
    PSP1 dw 0 ;9 хранит старое значение ES
```

```
до того, как программа оставлена резидентной в памяти
```

```
    INT_COUNT_VAL dw 0FEDCh ;11 хранит количество вызванных  
прерываний
```

```
    COUNT_MES db 'Count of interruptions: 0000 $' ;13
```

```
PROC_CODE:
```

```
    push ax
```

```
    push bx
```

```
    push cx
```

```
    push dx
```

```
    mov ah, 3h ; получить позицию и форму курсора
```

```
    mov bh, 0h ; страница
```

```
    int 10h
```

```
    push dx
```

```
    mov ah, 2h ; установить курсор
```

```
    mov bh, 0h
```

```
    mov bl, 3h
```

```
    mov dx, 220h
```

```

int 10h

push si
push cx
push ds
mov ax, SEG COUNT_MES
mov ds, ax
mov si, offset COUNT_MES
add si, 27

mov cx, 4
CYCLE:
mov ah, [si]
inc ah
mov [si], ah
cmp ah, 3Ah
jne END_INT
mov ah, 30h
mov [si], ah
dec si
loop CYCLE

END_INT:
    pop ds
    pop cx
    pop si

push es
push bp
mov ax, SEG COUNT_MES
mov es, ax
mov ax, offset COUNT_MES
mov bp, ax
mov ah, 13h
mov al, 00h
mov cx, 28
mov bh, 0
int 10h

```

```

pop bp
pop es

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

pop dx
pop cx
pop bx
pop ax

iret
INT_COUNT_FUNC ENDP
;-----
MORE_MEMORY PROC
MORE_MEMORY ENDP
;-----
CHECK_INTSET PROC NEAR;функция проверки установлен ли разработанный
вектор прерывания
push bx
push dx
push es

mov ah, 35h
mov al, 1Ch
int 21h

mov dx, es:[bx + 11] ;INT_COUNT_VAL
cmp dx, 0FEDCh
je INT_IS_SET
mov al, 00h
jmp END0

INT_IS_SET:
mov al, 01h
jmp END0

```

```

END0:
    pop es
    pop dx
    pop bx

    ret
CHECK_INTSET ENDP
;-----
CHECK_UN PROC NEAR; проверка параметра un. процедура загрузки/выгрузки
    push es

    mov ax, PSP0
    mov es, ax

    mov al, es:[81h+1]
    cmp al, '/'
    jne END_CHECK_UN

    mov al, es:[81h+2]
    cmp al, 'u'
    jne END_CHECK_UN

    mov al, es:[81h+3]
    cmp al, 'n'
    jne END_CHECK_UN
    mov al, 1h
END_CHECK_UN:
    pop es
    ret
CHECK_UN ENDP

REDEF_INT PROC NEAR; Устанавливает новые обработчики прерывания (25h
прерывания int 21h)
    push ax
    push bx
    push dx
    push es

```

```

; получаем адрес обработчика прерывания (старого) для того чтобы
сохранить
mov ah, 35h ;функция получения вектора
mov al, 1Ch ; номер вектора
int 21h
; на выходе в ES:BX = адрес обработчика прерывания
;возвращает значение вектора прерывания для INT (AL); то есть,
загружает в BX 0000:[AL*4], а в ES - 0000:[(AL*4)+2].
mov KEEP_IP, bx
mov KEEP_CS, es

push ds
; установка вектора прерывания: вход: ah = 25h, al = номер прерывания,
; ds:dx = вектор прерывания: адрес программы обработки прерывания
;на выходе ничего нет
mov dx, offset INT_COUNT_FUNC
mov ax, seg INT_COUNT_FUNC
mov ds, ax
mov ah, 25h ; функция установки вектора
mov al, 1Ch ; номер вектора
int 21h ; меняем прерывание
pop ds

mov dx, offset LOADING_MES
call WRITE

pop es
pop dx
pop bx
pop ax

ret
REDEF_INT ENDP
;-----
RESTORE_INT PROC NEAR
push ax
push bx
push dx

```

```
push es
```

```
mov ah, 35h
```

```
mov al, 1Ch
```

```
int 21h
```

```
cli; сбрасывает флаг прерывания в регистре флагов.
```

```
; Когда этот флаг сброшен, процессор игнорирует все прерывания (кроме  
NMI) от внешних устройств
```

```
push ds
```

```
mov dx, es:[bx + 5] ; загружаем в ds:dx адрес восстанавливаемой  
программы обработки прерывания ; KEEP_IP
```

```
mov ax, es:[bx + 3] ; KEEP_CS
```

```
mov ds, ax
```

```
mov ah, 25h
```

```
mov al, 1Ch
```

```
int 21h
```

```
pop ds
```

```
sti
```

```
mov dx, offset RESTORED_MES
```

```
call WRITE
```

```
push es
```

```
mov cx, es:[bx + 7] ; PSP0
```

```
mov es, cx
```

```
mov ah, 49h
```

```
int 21h
```

```
pop es
```

```
mov cx, es:[bx + 9] ; PSP1
```

```
mov es, cx
```

```
int 21h
```

```
pop es
```

```
pop dx
```

```
pop bx
```

```
pop ax
```

```

    ret
RESTORE_INT ENDP
;-----
WRITE PROC NEAR;печать строки
    push ax
    mov ah, 09h
    int 21h
    pop ax
    ret
WRITE ENDP
;-----
MAIN PROC FAR
    mov bx, 02Ch
    mov ax, [bx]
    mov PSP1, ax
    mov PSP0, ds
    xor ax, ax
    xor bx, bx

    mov ax, DATA
    mov ds, ax

    call CHECK_UN    ;Загрузка или выгрузка(проверка параметра)
    cmp al, 01h
    je UNLOAD_YET

    call CHECK_INTSET    ;Установлен ли разработанный вектор прерывания
    cmp al, 01h
    jne INTERRUPTION_IS_NOT_LOADED

    mov dx, offset ALREADY_LOAD_MES    ;Уже установлен(выход с сообщение)
    call WRITE
    jmp FINAL

    mov ah, 4Ch
    int 21h

```

INTERRUPTION_IS_NOT_LOADED:

call REDEF_INT

mov dx, offset MORE_MEMORY

mov cl, 04h

shr dx, cl

add dx, 1Bh

mov ax, 3100h

int 21h

UNLOAD_YET:

call CHECK_INTSET

cmp al, 00h

je NOT_SET

call RESTORE_INT

jmp FINAL

NOT_SET:

mov dx, offset NOTLOAD_MES

call WRITE

jmp FINAL

FINAL: ; ЗАБЕРШЕНИЕ ПРОГРАММЫ

mov ah, 4Ch

int 21h

MAIN ENDP

CODE ENDS

DATA SEGMENT

NOTLOAD_MES db "INTERRUPTION NOT LOAD", 13, 10, '\$'

RESTORED_MES db "INTERRUPTION RESTORED", 13, 10, '\$'

ALREADY_LOAD_MES db "INTERRUPTION IS ALREADY LOAD", 13, 10, '\$'

LOADING_MES db "INTERRUPTION IS LOADING AT THIS MOMENT", 13, 10, '\$'

DATA ENDS

END MAIN

