**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. 6381 |  | Герасимова Д.В. |
| Преподаватель |  | Губкин А.Ф. |

Санкт-Петербург

2018

**Цель работы:**

В архитектуре компьютера существуют стандартные прерывания, за которыми закреплены определённые вектора прерываний. Вектор прерываний хранит адрес подпрограммы обработчика прерываний. При возникновении прерывания, аппаратура компьютера передаёт 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 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.

**Постановка задачи:**

**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 1Ch.
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 о 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.

**Шаг 2.** Далее необходимо запустить отлаженную программу и убедиться, что резидентный обработчик прерывания 1Ch установлен. Работа прерывания должна отображаться на экране, а также необходимо проверить размещение прерывания в памяти. Для этого нужно запустить программу ЛР 3, которая отображает карту памяти в виде с писка блоков MCB.

**Шаг 3.** Затем необходимо запустить отлаженную программу еще раз и убедиться, что программа определяет установленный обработчик прерываний.

**Шаг 4.** Далее нужно запустить отлаженную программу с ключом выгрузки и убедиться, что резидентный обработчик прерывания выгружен, то есть сообщения на экран не выводятся, а память, занятая резидентом освобождена. Для этого также следует запустить программу ЛР 3.

**Необходимые сведения для составления программы:**

Резидентные обработчики прерываний - это программные модули, которые вызываются при возникновении прерываний определенного типа (сигнал таймера, нажатие клавиши и т.д.), которым соответствуют определенные вектора прерывания. Когда вызывается прерывание, процессор переключается на выполнение кода обработчика, а затем возвращается на выполнение прерванной программы. Адрес возврата в прерванную программу (CS:IP) запоминается в стеке вместе с регистром флагов. Затем в CS:IP загружается адрес точки входа программы обработки прерывания и начинает выполняться его код. Обработчик прерывания должен заканчиваться инструкцией IRET (возврат из прерывания).

Вектор прерывания имеет длину 4 байта. В первом хранится значение IP, во втором - CS. Младшие 1024 байта памяти содержат 256 векторов. Вектор для прерывания 0 начинается с ячейки 0000:0000, для прерывания 1 - с ячейки 0000:0004 и т.д.

Обработчик прерывания - это отдельная процедура, имеющая следующую структуру:

ROUT PROC FAR

PUSH AX ; сохранение изменяемых регистров

.....................................

<действия по обработке прерывания>

POP AX ; восстановление регистров

MOV AL, 20H

OUT 20H,AL

IRET

ROUT ENDP

Две последние строки необходимы для разрешения обработки прерываний с более низкими уровнями, чем только что обработанное. Для установки написанного прерывания в поле векторов прерываний используется функция 25H прерывания 21H, которая устанавливает вектор прерывания на указанный адрес.

PUSH DS

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

MOV AL, 1CH ; номер вектора

INT 21H ; меняем прерывание

POP DS

Программа, выгружающая обработчик прерываний должна восстанавливать оригинальные векторы прерываний. Функция 35 прерывания 21H позволяет восстановить значение вектора прерывания, помещая значение сегмента в ES, а смещение в BX. Программа должна содержать следующие инструкции:

; -- хранится в обработчике прерываний

KEEP\_CS DW 0 ; для хранения сегмента

KEEP\_IP DW 0 ; и смещения прерывания

; -- в программе при загрузке обработчика прерывания

MOV AH, 35H ; функция получения вектора

MOV AL, 1CH ; номер вектора ШТЕ 21Р

MOV KEEP\_IP, BX ; запоминание смещения

MOV KEEP\_CS, ES ; и сегмента

; -- в программе при выгрузке обработчика прерываний CLI

PUSH DS

MOV DX, KEEP\_IP

MOV AX, KEEP\_CS

MOV DS, AX

MOV AH, 25H

MOV AL, 1CH

INT 21H ; восстанавливаем вектор

POP DS

STI

Для того, чтобы оставить процедуру прерывания резидентной в памяти, следует воспользоваться функцией DOS 31h прерывания 21h. Эта функция оставляет память, размер которой указывается в качестве параметра, занятой, а остальную память освобождает и осуществляет выход в DOS.

Функция 31h int 21h использует следующие параметры:

AH - номер функции 31h;

AL - код завершения программы;

DX - размер памяти в параграфах, требуемый резидентной программе.

Пример обращения к функции:

MOV DX, OFFSET LAST\_BYTE ; размер в байтах от начала сегмента

MOV CL,4 ; перевод в параграфы

SHR DX,CL

INC DX ; размер в параграфах

MOV AH,31h

INT 21h

**В программе используются следующие процедуры:**

|  |  |
| --- | --- |
| **Название процедуры** | **Назначение** |
| INTERRUPT | Функция обработчика прерывания |
| ISLOADED | Проверка установки резидента |
| CHECK\_UNLOAD\_FLAG | Проверка команды '/un' |
| LOAD | Загрузка резидента |
| UNLOAD | Выгрузка резидента |

**Определение структуры данных:**

|  |  |  |
| --- | --- | --- |
| **Название поля данных** | **Тип** | **Назначение** |
| int\_not\_loaded | db | Резидент не загружен |
| int\_al\_loaded | db | Резидент уже загружен |
| int\_loaded | db | Резидент загружен |
| int\_unload | db | Резидент был выгружен |

**Результат работы программ**

1. Состояние памяти до загрузки резидента (используем модуль, разработанный в третьей лабораторной работе). (Рис. 1)

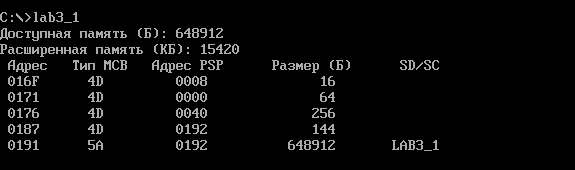


Рис.1

1. Загрузка резидента в память. (Рис. 2)

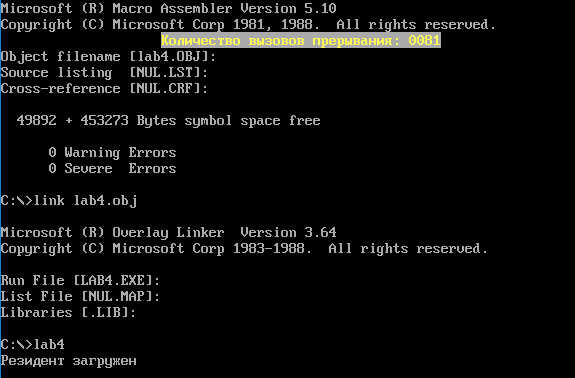


Рис. 2

1. Попытка повторной загрузки резидента в память. (Рис. 3)

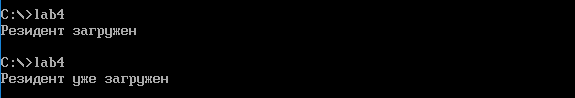


Рис. 3

1. Состояние памяти при загрузке в неё резидента. (Рис. 4)

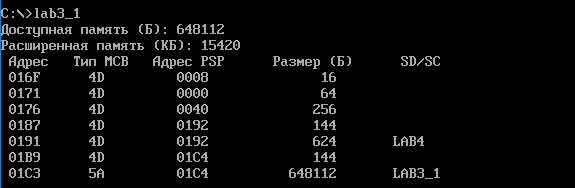


Рис. 4

1. Запускаем отложенную программу с ключом /un, тем самым выгружаем резидент и смотрим состояние памяти после выгрузки резидента. (Рис. 5)

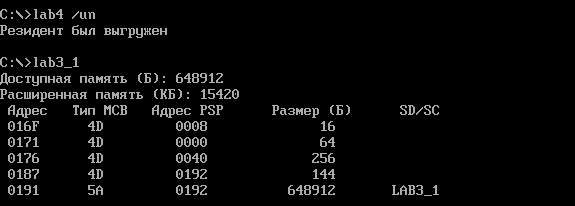


Рис. 5

**Ответы на контрольные вопросы:**

1. Как реализован механизм прерывания от часов?

**Ответ:** Сначала сохраняется содержимое регистров, потомопределяется 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. Аппаратное прерывание от таймера происходит каждые 55 мс.

1. Какого типа прерывания использовались в работе?

**Ответ:** аппаратные прерывания(1Сh), прерывания функций DOS(21h), прерывания функций BIOS(10h).

**Заключение:**

В ходе лабораторной работы был построен обработчик прерывания от сигналов таймера. Изучены дополнительные функции работы с памятью: установка программы-резидента и его выгрузка из памяти.

**Приложение**

**Код программы lab4.asm**

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

AStack SEGMENT STACK

DW 64 DUP(?)

AStack ENDS

CODE SEGMENT

INTERRUPT PROC FAR

jmp function

;DATA

AD\_PSP dw ?

SR\_PSP dw ?

keep\_cs dw ?

keep\_ip dw ?

is\_loaded dw 0FFDAh

counter db 'Количество вызовов прерывания: 0000 $'

function:

push ax

push bx

push cx

push dx

;Получение курсора

mov ah,3h

mov bh,0h

int 10h

push dx ;Сохраняем курсор в стеке

;Установка курсора

mov ah,02h

mov bh,0h

mov dx,0214h

int 10h

;Подсчет кол-ва прерываний

push si

push cx

push ds

mov ax,SEG counter

mov ds,ax

mov si,offset counter

add si,34

mov ah,[si]

inc ah

mov [si],ah

cmp ah,3Ah

jne \_not

mov ah,30h

mov [si],ah

mov bh,[si-1]

inc bh

mov [si-1],bh

cmp bh,3Ah

jne \_not

mov bh,30h

mov [si-1],bh

mov ch,[si-2]

inc ch

mov [si-2],ch

cmp ch,3Ah

jne \_not

mov ch,30h

mov [si-2],ch

mov dh,[si-3]

inc dh

mov [si-3],dh

cmp dh,3Ah

jne \_not

mov dh,30h

mov [si-3],dh

\_not:

pop ds

pop cx

pop si

;Печать строки

push es

push bp

mov ax,SEG counter

mov es,ax

mov ax,offset counter

mov bp,ax

mov ah,13h

mov al,00h

mov cx,35

mov bh,0

int 10h

pop bp

pop es

;восстановка курсора

pop dx

mov ah,02h

mov bh,0h

int 10h

pop dx

pop cx

pop bx

pop ax ;восстановление ax

iret

INTERRUPT ENDP

END\_INT PROC

END\_INT ENDP

ISLOADED PROC near

push dx

push es

push bx

mov ax,351Ch ;получение вектора прерываний

int 21h

mov dx,es:[bx+11]

cmp dx,0FFDAh ;проверка на совпадение кода

je int\_is\_loaded

mov al,0h

pop bx

pop es

pop dx

ret

int\_is\_loaded:

mov al,01h

pop bx

pop es

pop dx

ret

ISLOADED ENDP

CHECK\_UNLOAD\_FLAG PROC near

push es

mov ax,AD\_PSP

mov es,ax

xor bx,bx

inc bx

mov al,es:[81h+bx]

inc bx

cmp al,'/'

jne unload\_end

mov al,es:[81h+bx]

inc bx

cmp al,'u'

jne unload\_end

mov al,es:[81h+bx]

inc bx

cmp al,'n'

jne unload\_end

mov al,1h

unload\_end:

pop es

ret

CHECK\_UNLOAD\_FLAG ENDP

LOAD PROC near

push ax

push bx

push dx

push es

mov ax,351Ch

int 21h

mov keep\_ip,bx

mov keep\_cs,es

push ds

mov dx,offset INTERRUPT

mov ax,seg INTERRUPT

mov ds,ax

mov ax,251Ch

int 21h

pop ds

mov dx,offset int\_loaded

mov ah,09h

int 21h

pop es

pop dx

pop bx

pop ax

ret

LOAD ENDP

UNLOAD PROC near

push ax

push bx

push dx

push es

mov ax,351Ch

int 21h

cli

push ds

mov dx,es:[bx+9] ;IP стандартного

mov ax,es:[bx+7] ;CS стандартного

mov ds,ax

mov ax,251Ch

int 21h

pop ds

sti

mov dx,offset int\_unload ;сообщение о выгрузке

mov ah,09h

int 21h

;Удаление MCB

push es

mov cx,es:[bx+3]

mov es,cx

mov ah,49h

int 21h

pop es

mov cx,es:[bx+5]

mov es,cx

int 21h

pop es

pop dx

pop bx

pop ax

ret

UNLOAD ENDP

Main PROC far

mov bx,02Ch

mov ax,[bx]

mov SR\_PSP,ax

mov AD\_PSP,ds ;сохраняем PSP

sub ax,ax

xor bx,bx

mov ax,data

mov ds,ax

call CHECK\_UNLOAD\_FLAG ;Загрузка или выгрузка(проверка параметра)

cmp al,1h

je un\_load

call ISLOADED ;Установлен ли разработанный вектор прерывания

cmp al,01h

jne al\_loaded

mov dx,offset int\_al\_loaded ;Уже установлен(выход с сообщение)

mov ah,09h

int 21h

mov ah,4Ch

int 21h

al\_loaded:

;Загрузка

call LOAD

;Оставляем обработчик прерываний в памяти

mov dx,offset END\_INT

mov cl,4h

shr dx,cl

inc dx

add dx,1Ah

mov ax,3100h

int 21h

;Выгрузка

un\_load:

call ISLOADED

cmp al,0h

je not\_loaded

call UNLOAD

mov ax,4C00h

int 21h

not\_loaded:

mov dx,offset int\_not\_loaded ;Если резидент не установлен, то нежелательно выгружать стандартный ВП

mov ah,09h

int 21h

mov ax,4C00h

int 21h

Main ENDP

CODE ENDS

DATA SEGMENT

int\_not\_loaded db 'Резидент не загружен',13,10,'$'

int\_al\_loaded db 'Резидент уже загружен',13,10,'$'

int\_loaded db 'Резидент загружен',13,10,'$'

int\_unload db 'Резидент был выгружен',13,10,'$'

DATA ENDS

END Main