**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МОЭВМ**

**отчет**

по лабораторной работе №**4**

по дисциплине «Операционные системы»

на тему: Обработка стандартных прерываний

|  |  |  |
| --- | --- | --- |
| Студент гр. 6381 |  | Кухарев М.А. |
| 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 и выполняет соответствующие действия.

В лабораторной работе номер 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.

**Необходимые сведения для выполнения лабораторной работы:**

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 прерывание - это отдельная процедура, имеющая следующую структуру:

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 |

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 INT 21H

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

**Описание функций:**

|  |  |
| --- | --- |
| Название | Назначение |
| MY\_INTERRUPTION | Функция обработчика прерывания |
| IS\_INTERRUPTION\_SET | Проверка установки резидента |
| CHECK\_COMMAND\_PROMT | Проверка команды '/un' |
| LOAD\_INTERRUPTION | Загрузка резидента |
| UNLOAD\_INTERRUPTION | Выгрузка резидента |

**Описание данных:**

|  |  |  |
| --- | --- | --- |
| Название | Тип | Назначение |
| M\_INT\_RESTORED | Db | Резидент загружен |
| M\_INT\_ISLOADED | Db | Резидент уже загружен |
| M\_INT\_ISLOADED0 | Db | Резидент был выгружен |
| M\_INT\_NOT\_SET | Db | Резидент не загружен |

**Ход работы:**

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

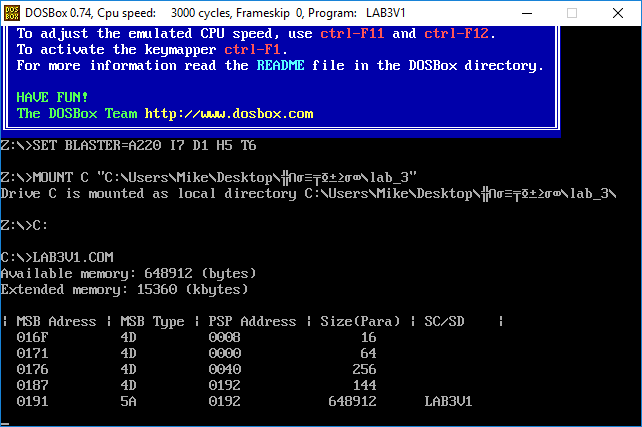


Рисунок 1:работа программы LAB3V1.COM

1. Загрузка резидента в память.

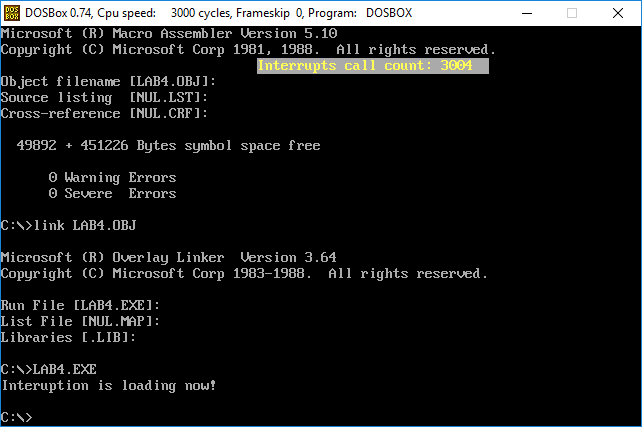


Рисунок 2: Первый запуск программы

1. Попытка повторной загрузки резидента в память.

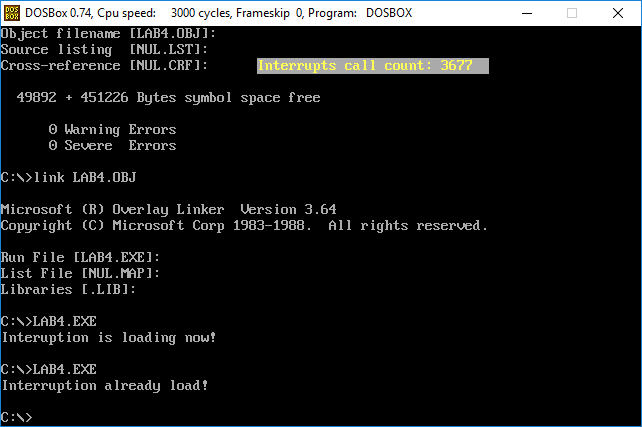


Рисунок 3:Повторный запуск программы

1. Состояние памяти при загрузке в неё резидента.

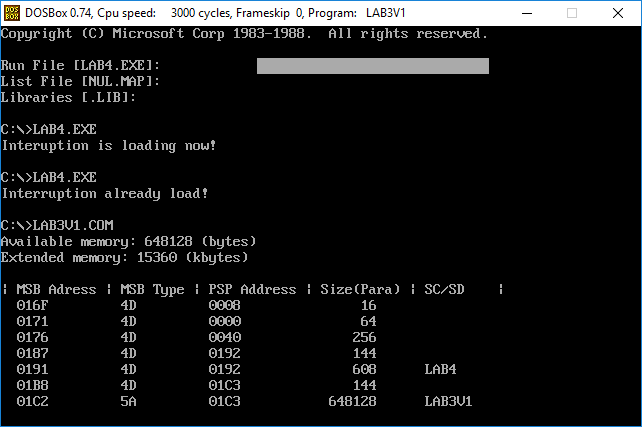


Рисунок 4: Повторный запуск программы LAB3V1.COM

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

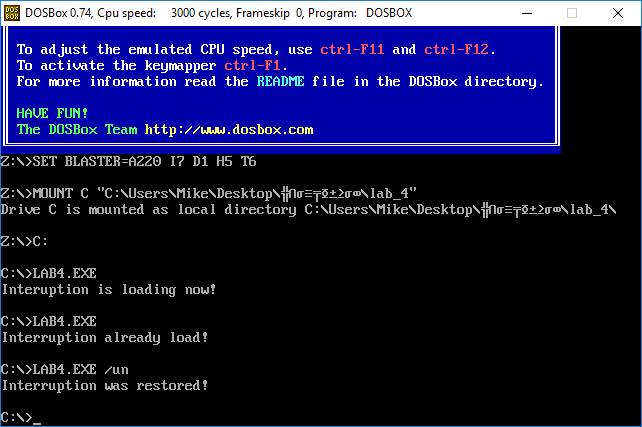


Рисунок 5

**Контрольные вопросы**

* Как реализован механизм прерывания от часов?

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.

* Какого типа прерывания использовались в работе?

1)Аппаратные прерывания

2)Прерывания функций DOS(21h)

3)Прерывания функций BIOS(10h)

**Приложение.** Код программы:

ASSUME CS:CODE, DS:DATA, SS:MY\_STACK

;------------------------------------

MY\_STACK SEGMENT STACK

DW 64 DUP(?)

MY\_STACK ENDS

;------------------------------------

CODE SEGMENT

;------------------------------------

MY\_INTERRUPTION PROC FAR

jmp START\_FUNC

;TMP DATA

PSP\_ADDRESS\_0 dw 0 ;offset - 3

PSP\_ADDRESS\_1 dw 0 ;offset - 5

KEEP\_CS dw 0 ;offset - 7

KEEP\_IP dw 0 ;offset - 9

MY\_INTERRUPTION\_SET dw 0FEDCh ;offset - 11

INT\_COUNT db 'Interrupts call count: 0000 $' ;offset - 13

START\_FUNC:

push ax

push bx

push cx

push dx

mov ah, 03h

mov bh, 00h

int 10h

push dx

mov ah, 02h

mov bh, 00h

mov dx, 0220h

int 10h

push si

push cx

push ds

mov ax, SEG INT\_COUNT

mov ds, ax

mov si, offset INT\_COUNT

add si, 1Ah

mov ah,[si]

inc ah

mov [si], ah

cmp ah, 3Ah

jne END\_CALC

mov ah, 30h

mov [si], ah

mov bh, [si - 1]

inc bh

mov [si - 1], bh

cmp bh, 3Ah

jne END\_CALC

mov bh, 30h

mov [si - 1], bh

mov ch, [si - 2]

inc ch

mov [si - 2], ch

cmp ch, 3Ah

jne END\_CALC

mov ch, 30h

mov [si - 2], ch

mov dh, [si - 3]

inc dh

mov [si - 3], dh

cmp dh, 3Ah

jne END\_CALC

mov dh, 30h

mov [si - 3],dh

END\_CALC:

pop ds

pop cx

pop si

push es

push bp

mov ax, SEG INT\_COUNT

mov es, ax

mov ax, offset INT\_COUNT

mov bp, ax

mov ah, 13h

mov al, 00h

mov cx, 1Dh

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

MY\_INTERRUPTION ENDP

;------------------------------------

NEED\_MEM\_AREA PROC

NEED\_MEM\_AREA ENDP

;------------------------------------

IS\_INTERRUPTION\_SET PROC NEAR

push bx

push dx

push es

mov ah, 35h

mov al, 1Ch

int 21h

mov dx, es:[bx + 11]

cmp dx, 0FEDCh

je INT\_IS\_SET

mov al, 00h

jmp POP\_REG

INT\_IS\_SET:

mov al, 01h

jmp POP\_REG

POP\_REG:

pop es

pop dx

pop bx

ret

IS\_INTERRUPTION\_SET ENDP

;------------------------------------

CHECK\_COMMAND\_PROMT PROC NEAR

push es

mov ax, PSP\_ADDRESS\_0

mov es, ax

mov bx, 0082h

mov al, es:[bx]

inc bx

cmp al, '/'

jne NULL\_CMD

mov al, es:[bx]

inc bx

cmp al, 'u'

jne NULL\_CMD

mov al, es:[bx]

inc bx

cmp al, 'n'

jne NULL\_CMD

mov al, 0001h

NULL\_CMD:

pop es

ret

CHECK\_COMMAND\_PROMT ENDP

;------------------------------------

LOAD\_INTERRUPTION PROC NEAR

push ax

push bx

push dx

push es

mov ah, 35h

mov al, 1Ch

int 21h

mov KEEP\_IP, bx

mov KEEP\_CS, es

push ds

mov dx, offset MY\_INTERRUPTION

mov ax, seg MY\_INTERRUPTION

mov ds, ax

mov ah, 25h

mov al, 1Ch

int 21h

pop ds

mov dx, offset M\_INT\_ISLOADED0

call PRINT\_STRING

pop es

pop dx

pop bx

pop ax

ret

LOAD\_INTERRUPTION ENDP

;------------------------------------

UNLOAD\_INTERRUPTION PROC NEAR

push ax

push bx

push dx

push es

mov ah, 35h

mov al, 1Ch

int 21h

cli

push ds

mov dx, es:[bx + 9]

mov ax, es:[bx + 7]

mov ds, ax

mov ah, 25h

mov al, 1Ch

int 21h

pop ds

sti

mov dx, offset M\_INT\_RESTORED

call PRINT\_STRING

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\_INTERRUPTION ENDP

;------------------------------------

PRINT\_STRING PROC NEAR

push ax

mov ah, 09h

int 21h

pop ax

ret

PRINT\_STRING ENDP

;------------------------------------

MAIN\_PROGRAM PROC FAR

mov bx, 02Ch

mov ax, [bx]

mov PSP\_ADDRESS\_1, ax

mov PSP\_ADDRESS\_0, ds

sub ax, ax

xor bx, bx

mov ax, DATA

mov ds, ax

call CHECK\_COMMAND\_PROMT ;Загрузка или выгрузка(проверка параметра)

cmp al, 01h

je UNLOAD\_START

call IS\_INTERRUPTION\_SET ;Установлен ли разработанный вектор прерывания

cmp al, 01h

jne INTERRUPTI0N\_IS\_NOT\_LOADED

mov dx, offset M\_INT\_ISLOADED ;Уже установлен(выход с сообщение)

call PRINT\_STRING

jmp EXIT\_PROGRAM

mov ah,4Ch

int 21h

INTERRUPTI0N\_IS\_NOT\_LOADED:

call LOAD\_INTERRUPTION

mov dx, offset NEED\_MEM\_AREA

mov cl, 04h

shr dx, cl

add dx, 1Bh

mov ax, 3100h

int 21h

UNLOAD\_START:

call IS\_INTERRUPTION\_SET

cmp al, 00h

je INT\_IS\_NOT\_SET

call UNLOAD\_INTERRUPTION

jmp EXIT\_PROGRAM

INT\_IS\_NOT\_SET:

mov dx, offset M\_INT\_NOT\_SET

call PRINT\_STRING

jmp EXIT\_PROGRAM

EXIT\_PROGRAM:

mov ah, 4Ch

int 21h

MAIN\_PROGRAM ENDP

;------------------------------------

CODE ENDS

;------------------------------------

DATA SEGMENT

;messages

M\_INT\_NOT\_SET db "Interruption didnt load!", 0dh, 0ah, '$'

M\_INT\_RESTORED db "Interruption was restored!", 0dh, 0ah, '$'

M\_INT\_ISLOADED db "Interruption already load!", 0dh, 0ah, '$'

M\_INT\_ISLOADED0 db "Interuption is loading now!", 0dh, 0ah, '$'

DATA ENDS

;------------------------------------

END MAIN\_PROGRAM