**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МО ЭВМ**

отчет

**по лабораторной работе №5**

**по дисциплине «Операционные системы»**

Тема: Сопряжение стандартного и пользовательского обработчиков прерываний

|  |  |  |
| --- | --- | --- |
| Студентка гр. 6381 |  | Лопатина А.С. |
| Преподаватель |  | Губкин А.Ф. |

Санкт-Петербург

2018

**Постановка задачи**

**Цель работы.**

Исследование возможности встраивания пользовательского обработчика прерываний в стандартный обработчик от клавиатуры. Пользовательский обработчик прерывания получает управление по прерыванию (int 09h) при нажатии клавиши на клавиатуре. Он обрабатывает скан-код и осуществляет определенные действия, если скан-код совпадает с определенными кодами, которые он должен обрабатывать. Если скан-код не совпадает с этими кодами, то управление передается стандартному прерыванию.

**Сведения о функциях и структурах данных управляющей программы.**

Сведения о функциях:

|  |  |
| --- | --- |
| Название | Назначение |
| OUTPUT\_PROC | Вывод на экран |
| INTERRUPTION | Обработчик прерывания для 09h |
| INSTALL\_CHECK | Проверка установки прерывания |
| UN\_CHECK | Проверка на введение /un |
| INSTALL\_INTER | Загрузка обработчика прерывания |
| UNLOAD\_INTER | Выгрузка обработчика прерывания |
| MAIN | Основная функция |

Сведения о структурах данных:

|  |  |  |
| --- | --- | --- |
| Название | Тип | Назначение |
| INSTALL | db | Обработчик прерывания установлен |
| NOT\_INSTALL | db | Обработчик прерывания не установлен |
| ALR\_INSTALL | db | Обработчик прерывания уже установлен |
| UNLOAD | db | Обработчик прерывания выгружен |
| ADDR\_PSP1 | dw | Переменная для хранения PSP |
| ADDR\_PSP2 | dw | Вторая переменная для хранения PSP |
| KEEP\_CS | dw | Переменная для хранения сегмента стандартного прерывания 09h |
| KEEP\_IP | dw | Переменная для хранения смещения стандартного прерывания 09h |
| INTER\_SET | dw | Адрес пользовательского прерывания 09h |
| REQ\_KEY\_6 | db | Скан код цифры 6 |
| REQ\_KEY\_7 | db | Скан код цифры 7 |
| REQ\_KEY\_8 | db | Скан код цифры 8 |
| REQ\_KEY\_9 | db | Скан код цифры 9 |
| REQ\_KEY\_0 | db | Скан код цифры 0 |

**Последовательность действий, выполняемых программой.**

Модуль типа .EXE выполняет следующие функции:

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

2) Устанавливает резидентную функцию для обработки прерывания настраивает вектор прерываний, если прерывание не установлено, и осуществляется выход о функции 4Ch прерывания int 21h.

3) Если прерывание установлено, то выводится соответствующее сообщение и осуществляется выход по функции 4Ch прерывания int 21h.

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.

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 анализируется скан-код.

3) Если этот код совпадает с одним из заданных, то требуемый код записывается в буфер клавиатуры.

4) Если этот код не совпадает ни с одним из заданных, то осуществляется передача управления стандартному обработчику прерывания.

**Результат работы программы.**

1. Написали и отладили программный модуль типа .ЕХЕ. Запускаем отлаженную программу и убеждаемся, что резидентный обработчик прерывания 09H установлен. Работа прерывания проверяется введением различных символов, обрабатываемых установленным обработчиком и стандартным обработчиком;

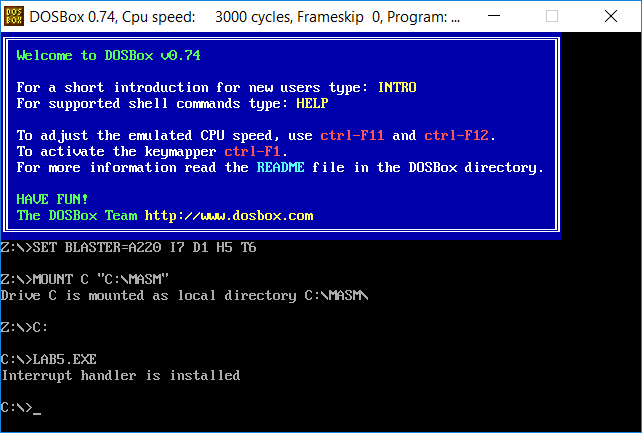


рис. 1

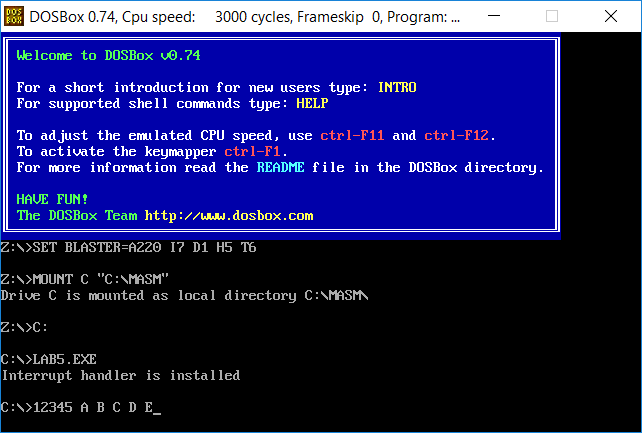


рис. 2 Введение символов от 1 до 0

1. Проверяем размещение прерывания в памяти. Для этого запускаем программу lab3, которая отображает карту памяти в виде списка блоков МСВ;

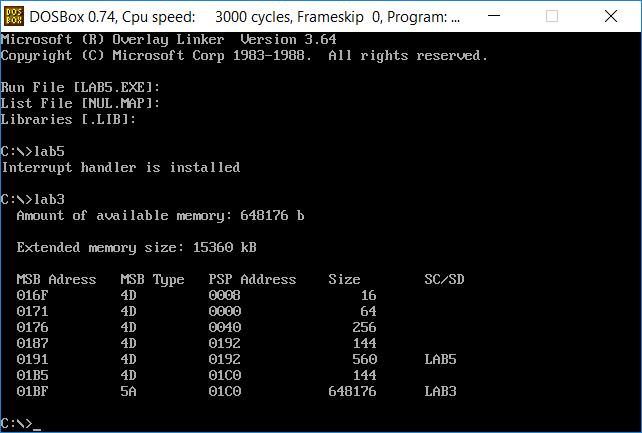


рис. 3

1. Запускаем отлаженную программу еще раз и убеждаемся, что программа

определяет установленный обработчик прерываний;

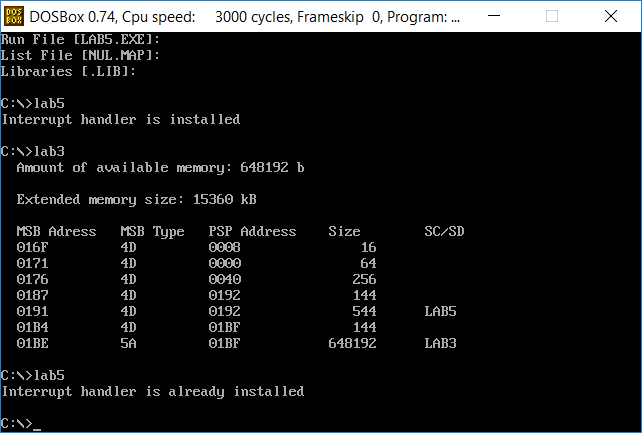


рис. 4

1. Запускаем отлаженную программу с ключом выгрузки и убеждаемся, что резидентный обработчик прерывания выгружен, то есть сообщения на экран не выводятся, а память, занятая резидентом освобождена. Для этого также запускаем программу lab3;

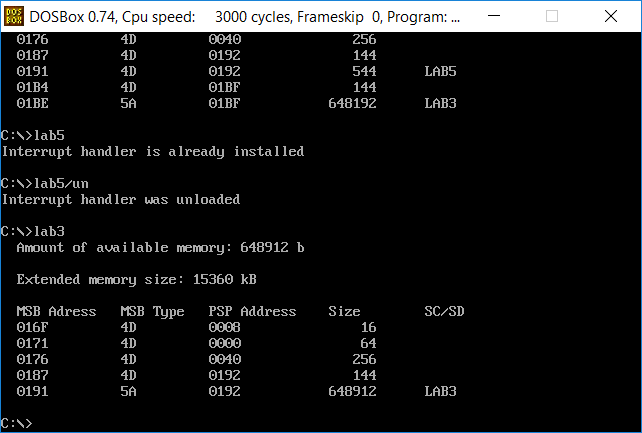


рис. 5

**Описание результатов исследования проблем, поставленных в лабораторной работе**

1. Какого типа прерывания использовались в работе?

Ответ: в работе использовались прерывания функций BIOS (int 09h, int 16h), а также пользовательское прерывание функций DOS (int 21h);

1. Чем отличается скан код от кода ASCII?

Ответ: скан код – это код, присвоенный каждой клавише, с помощью которого драйвер клавиатуры распознаёт, какая клавиша была нажата. А ASCII-код – это код, определяющий закреплённый за клавишей символ.

**Заключение**

В результате выполнения лабораторной работы были изучены возможности встраивания пользовательского обработчика прерываний в стандартный обработчик от клавиатуры и разработано пользовательское прерывание от клавиатуры, которое обрабатывает скан-коды, выполняет вывод сообщения результата нажатия и при несовпадении скан-кода передает управление стандартному обработчику.

Приложения

**Код программы** **LAB5**.**asm**

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

ASTACK SEGMENT STACK

DW 64 DUP(?)

ASTACK ENDS

CODE SEGMENT

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

OUTPUT\_PROC PROC NEAR ;Вывод на экран сообщения

push ax

mov ah, 09h

int 21h

pop ax

ret

OUTPUT\_PROC ENDP

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

INTERRUPTION PROC FAR

jmp begin

ADDR\_PSP1 dw ? ;offset 3

ADDR\_PSP2 dw ? ;offset 5

KEEP\_IP dw ? ;offset 7

KEEP\_CS dw ? ;offset 9

INTER\_SET dw 0ABCDh ;offset 11

REQ\_KEY\_6 db 07h

REQ\_KEY\_7 db 08h

REQ\_KEY\_8 db 09h

REQ\_KEY\_9 db 0Ah

REQ\_KEY\_0 db 0Bh

begin:

in al,60h ;Cчитать ключ

cmp al, REQ\_KEY\_6

je key6

cmp al, REQ\_KEY\_7

je key7

cmp al, REQ\_KEY\_8

je key8

cmp al, REQ\_KEY\_9

je key9

cmp al, REQ\_KEY\_0

je key0

jmp dword ptr cs:[KEEP\_IP] ;переходим на стандартный обработчик

key6:

mov cl, 'A'

jmp do\_req

key7:

mov cl, 'B'

jmp do\_req

key8:

mov cl, 'C'

jmp do\_req

key9:

mov cl, 'D'

jmp do\_req

key0:

mov cl, 'E'

do\_req:

in al,61h ;Взять значение порта управления клавиатурой

mov ah,al ;Сохранить его

or al,80h ;Установить бит разрешения для клавиатуры

out 61h,al ;И вывести его в управляющий порт

xchg ah, al ;Извлечь исходное значение порта

out 61h,al ;И записать его обратно

mov al,20h ;Послать сигнал конца прерывания контроллеру прерываний 8259

out 20h,al

push bx

push cx

push dx

mov ah, 05h ;функция, позволяющая записать символ в буфер клавиатуры

mov cl,al

mov ch, 00h ;символ в CL уже занесён ранее, осталось обнулить CH

int 16h

or al, al ;проверка переполнения буфера

jnz skip ;если переполнен - идём в skip

jmp return ;иначе выходим

skip: ;очищаем буфер

push es

push si

mov ax, 0040h

mov es, ax

mov si, 001ah

mov ax, es:[si]

mov si, 001ch

mov es:[si], ax

pop si

pop es

return:

pop dx

pop cx

pop bx

iret

INTERRUPTION ENDP

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

inter\_end:

INSTALL\_CHECK PROC NEAR ;Проверка установки прерывания

push bx

push dx

push es

mov ah, 35h ;Получение вектора прерываний

mov al, 09h ;Функция выдает значение сегмента в ES, смещение в BX

int 21h

mov dx, es:[bx + 11]

cmp dx, 0ABCDh ;Проверка на совпадение кода прерывания

je install\_

mov al, 00h

jmp end\_install

install\_:

mov al, 01h

jmp end\_install

end\_install:

pop es

pop dx

pop bx

ret

INSTALL\_CHECK ENDP

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

UN\_CHECK PROC NEAR ;Проверка на то, не ввёл ли пользователь /un

push es

mov ax, ADDR\_PSP1

mov es, ax

cmp byte ptr es:[82h], '/'

jne not\_enter

cmp byte ptr es:[83h], 'u'

jne not\_enter

cmp byte ptr es:[84h], 'n'

jne not\_enter

mov al, 1h

not\_enter:

pop es

ret

UN\_CHECK ENDP

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

INSTALL\_INTER PROC NEAR ;Cохранение стандартного обработчика прерываний и загрузка собственного

push ax

push bx

push dx

push es

mov ah, 35h

mov al, 09h

int 21h

mov KEEP\_IP, bx ;Запоминаем смещение и сегмент

mov KEEP\_CS, es

push ds

lea dx, INTERRUPTION

mov ax, seg INTERRUPTION

mov ds, ax

mov ah, 25h

mov al, 09h

int 21h

pop ds

lea dx, INSTALL

call OUTPUT\_PROC

pop es

pop dx

pop bx

pop ax

ret

INSTALL\_INTER ENDP

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

UNLOAD\_INTER PROC NEAR ;Выгрузка обработчика прерывания

push ax

push bx

push dx

push es

mov ah, 35h

mov al, 09h

int 21h

cli

push ds

mov dx, es:[bx + 7]

mov ax, es:[bx + 9]

mov ds, ax

mov ah, 25h

mov al, 09h

int 21h

pop ds

sti

lea dx, UNLOAD

call OUTPUT\_PROC

push es ;Удаление MCB

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

mov ah, 4Ch ;Выход из программы через функцию 4C

int 21h

ret

UNLOAD\_INTER ENDP

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

MAIN PROC FAR

mov bx,2Ch

mov ax,[bx]

mov ADDR\_PSP2,ax

mov ADDR\_PSP1,ds ;сохраняем PSP

mov dx, ds

sub ax,ax

xor bx,bx

mov ax,data

mov ds,ax

xor dx, dx

call UN\_CHECK ;Проверка на введение /un

cmp al, 01h

je unload\_

call INSTALL\_CHECK ;Проверка не является ли программа резидентной

cmp al, 01h

jne not\_resident

lea dx, ALR\_INSTALL ;Программа уже загружена

call OUTPUT\_PROC

jmp quit

;Загрузка резидента

not\_resident:

call INSTALL\_INTER

lea dx, inter\_end

mov cl, 04h

shr dx, cl

add dx, 1Bh

mov ax, 3100h

int 21h

;Выгрузка резидента

unload\_:

call INSTALL\_CHECK

cmp al, 0h

je not\_install\_

call UNLOAD\_INTER

jmp quit

;Прерывание выгружено

not\_install\_:

lea dx, UNLOAD

call OUTPUT\_PROC

quit:

mov ah, 4Ch

int 21h

MAIN ENDP

CODE ENDS

DATA SEGMENT

INSTALL db 'Interrupt handler is installed', 0dh, 0ah, '$'

NOT\_INSTALL db 'Interrupt handler is not installed', 0dh, 0ah, '$'

ALR\_INSTALL db 'Interrupt handler is already installed', 0dh, 0ah, '$'

UNLOAD db 'Interrupt handler was unloaded', 0dh, 0ah, '$'

DATA ENDS

END Main