**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

|  |  |  |
| --- | --- | --- |
| Студент гр. 6381 |  | Поляков Н.С. |
| Преподаватель |  | Губкин А.Ф. |

Санкт-Петербург

2018

**Цель работы.**

Встраивание пользовательского прерывания в стандартный обработчик прерывания от клавиатуры.

**Описание функций.**

|  |  |
| --- | --- |
| Имя | Описание |
| UNOAD\_INT | Восстанавливает сохраненный вектор прерывания |
| CHECK\_INT | Проверяет, установлено ли прерывание |
| IS\_UNLOAD | Проверяет необходимость выгрузки прерывания из памяти |
| LOAD\_INT | Загружает прерывание резидентно в память |
| INTER | Обработчик прерывания |

**Описание структур данных.**

|  |  |  |
| --- | --- | --- |
| Имя | Тип | Назначение |
| INTER\_ALREADY | db | Строка “Interruption is already loaded” |
| INTER\_UNLOADED | db | Строка “Interruption is unloaded” |
| INTER\_LOADED | db | Строка “Interruption is loaded” |
| INTER\_NOT\_LOADED | db | Строка “Interruption is not loaded” |
| KEY\_WORD | db | Строка “MY\_IN5” |
| KEEP\_CS | dw | Сохранённое значение CS |
| KEEP\_IP | dw | Сохранённое значение IP |
| KEEP\_PSP | dw | Сохранённое значение адреса PSP |
| KEEP\_AX | dw | Сохраненное значение регистра ax |
| KEEP\_SS | dw | Сохраненное значение регистра ss |
| KEEP\_SP | dw | Сохраненное значение регистра sp |

**Тестирование.**

1. Загрузка прерывания.

**../../../../../../../../../Users/nestor/Desktop/Screen%20Sho**

Рис.1 Результат работы программы lab5.exe.

1. При нажатии на клавишу "esc" выводится символ "X". При нажатии других клавиш, поведение остаётся неизменным.

**../../../../../../../../../Users/nestor/Desktop/Screen%20Sho**

Рис.2 Проверка работы программы lab5.exe.

1. Проверка определения загруженного обработчика загружаемой программой.

../../../../../../../../../Users/nestor/Desktop/Screen%20Sho

Рис.3 Повторный запуск программы lab5.com.

1. Проверка резидентности загруженного обработчика с помощью программы lab3.com.

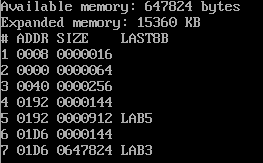
****

Рис.4 Результат работы программы lab3.com.

1. Проверка работы программы с ключом выгрузки.

../../../../../../../../../Users/nestor/Desktop/Screen%20Sho

Рис.5 Результат работы программы 5.exe с ключом "/un".

1. Проверка корректного освобождения памяти при выгрузки обработчика из памяти.

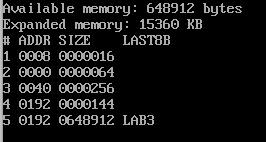


Рис.5 Результат работы программы lab3.exe после выгрузки прерывания из памяти.

**Ответы на контрольные вопросы.**

1. Какого типа прерывания использовались в работе?

Ответ: В работе использовались аппаратные (прерывание от от клавиатуры) и программные (int 21h).

1. Чем отличается скан-код от кода ASCII?

Ответ: Скан-код посылает клавиатура в порт 60h чтобы обрабатывать нажатия клавиш. ASCII код – цифровое представление символов.

**Вывод.**

В ходе выполнения лабораторной работы было изучено встраивание пользовательского обработчика прерывания в стандартный обработчик и построен такой обработчик.

**Приложение 1. Код программы.**

**lab5.asm**

INT\_STACK SEGMENT

DW 100h DUP(?)

INT\_STACK ENDS

;=======================================================

CODE SEGMENT

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

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

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

INTER PROC FAR

jmp int\_code

KEY\_WORD db 'MY\_IN5'

KEEP\_IP DW 0

KEEP\_CS DW 0

KEEP\_PSP DW 0

KEEP\_SS dw 0

KEEP\_SP dw 0

KEEP\_AX dw 0

int\_code:

mov KEEP\_AX,ax

mov KEEP\_SS,ss

mov KEEP\_SP,sp

mov ax, seg INT\_STACK

mov ss, ax

mov sp, 100h

mov ax,KEEP\_AX

push ax

push dx

push ds

push es

in al,60h

cmp al,01h

je my\_key

pushf

call dword ptr CS:KEEP\_IP

jmp int\_end

my\_key:

push ax

in AL,61h

mov ah,al

or al,80h

out 61h,al

xchg ah,al

out 61h,al

mov al,20h

out 20h,al

pop ax

add\_to\_buf:

mov ah,05h

mov cl,'X'

mov ch,00h

int 16h

or al,al

jz int\_end

CLI

mov ax,es:[1Ah]

mov es:[1Ch],ax

STI

jmp add\_to\_buf

int\_end:

pop es

pop ds

pop dx

pop ax

mov ss, KEEP\_SS

mov sp, KEEP\_SP

mov al,20h

out 20h,al

iret

end\_inter:

INTER ENDP

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

LOAD\_INT PROC near

push ax

push cx

push bx

push dx

push ds

mov ah, 35h

mov al, 09h

int 21h

mov KEEP\_IP, bx

mov KEEP\_CS, es

mov ax, SEG INTER

mov dx, OFFSET INTER

mov ds,ax

mov ah, 25h

mov al, 09h

int 21h

mov dx, OFFSET end\_inter

mov cl,4

shr dx,cl

inc dx

add dx, CODE

sub dx, KEEP\_PSP

mov ah, 31h

int 21h

pop ds

pop dx

pop bx

pop cx

pop ax

ret

LOAD\_INT ENDP

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

IS\_UNLOAD PROC near

push di

mov di, 81h

cmp byte ptr [di+0], ' '

jne bad\_key

cmp byte ptr [di+1], '/'

jne bad\_key

cmp byte ptr [di+2], 'u'

jne bad\_key

cmp byte ptr [di+3], 'n'

jne bad\_key

cmp byte ptr [di+4], 0Dh

jne bad\_key

cmp byte ptr [di+5], 0h

jne bad\_key

pop di

mov al,1

ret

bad\_key:

pop di

mov al,0

ret

IS\_UNLOAD ENDP

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

CHECK\_INT PROC near

push ax

push bx

push es

mov ah, 35h

mov al, 09h

int 21h

mov ax, OFFSET KEY\_WORD

sub ax, OFFSET INTER

add bx, ax

mov si,bx

push ds

mov ax,es

mov ds,ax

cmp [si], 'YM'

jne false

add si,2

cmp [si], 'I\_'

jne false

add si,2

cmp [si], '5N'

jne false

pop ax

mov ds,ax

pop es

pop bx

pop ax

mov al,1

ret

false:

pop ax

mov ds,ax

pop es

pop bx

pop ax

mov al,0

ret

CHECK\_INT ENDP

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

UNLOAD\_INT PROC near

push ax

push dx

mov ah, 35h

mov al, 09h

int 21h

cli

push ds

mov dx, es:KEEP\_IP

mov ax, es:KEEP\_CS

mov ds, ax

mov ah, 25h

mov al, 09h

int 21h

pop ds

mov es, es:KEEP\_PSP

push es

mov es, es:[2Ch]

mov ah, 49h

int 21h

pop es

mov ah, 49h

int 21h

sti

pop dx

pop ax

ret

UNLOAD\_INT ENDP

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

MAIN PROC FAR

push ds

sub ax,ax

push ax

mov cs:KEEP\_PSP, es

call CHECK\_INT

cmp al, 1

je int\_loaded

call IS\_UNLOAD

cmp al, 1

je int\_not\_loaded

mov dx, offset INTER\_LOADED

mov ax,DATA

mov ds,ax

mov ah, 9

int 21h

call LOAD\_INT

jmp end\_prog

int\_not\_loaded:

mov dx, offset INTER\_NOT\_LOADED

mov ax,DATA

mov ds,ax

mov ah, 9

int 21h

jmp end\_prog

int\_loaded:

call IS\_UNLOAD

cmp al, 1

je need\_to\_unload

mov dx, offset INTER\_ALREADY

mov ax,DATA

mov ds,ax

mov ah, 9

int 21h

jmp end\_prog

need\_to\_unload:

call UNLOAD\_INT

mov dx, offset INTER\_UNLOADED

mov ax,DATA

mov ds,ax

mov ah, 9

int 21h

jmp end\_prog

end\_prog:

xor al,al

mov ah,4ch

int 21h

MAIN ENDP

CODE ENDS

;=======================================================

STACK SEGMENT

DW 100h DUP(?)

STACK ENDS

;=======================================================

DATA SEGMENT

INTER\_ALREADY db 'Interruption is already loaded',13,10,36

INTER\_UNLOADED db 'Interruption is unloaded',13,10,36

INTER\_LOADED db 'Interruption is loaded',13,10,36

INTER\_NOT\_LOADED db 'Interruption is not loaded',13,10,36

DATA ENDS

;=======================================================

END MAIN