**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. 7381 |  | Минуллин М.А. |
| Преподаватель |  | Ефремов М.А. |

Санкт-Петербург

2019

**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.

**Порядок выполнения работы**

1. Проверить, установлено ли пользовательское прерывание с вектором 09h.
2. Если прерывание не установлено, то установить резидентную функцию для обработки прерывания и настроить вектор прерываний. Адрес точки входа в стандартный обработчик прерывания находится в теле пользовательского обработчика. Осуществить выход по функции 4Ch прерывания int 21h.
3. Если прерывание установлено, то выводится соответствующее сообщение и осуществляется выход по функции 4Ch прерывания int 21h.

**Ход работы**

Состояние памяти до запуска lab5.exe представлено на Рис.1 (использовалась программа lab3\_1.com):

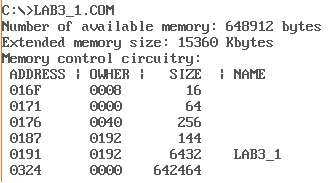


Рисунок 1 – Результат работы программы lab3\_1.com

Запуск программы lab5.exe представлен на Рис.2:



Рисунок 2 – Установка пользовательского обработчика прерывания

Проверим загрузку пользовательского обработчика и его работу. При нажатии клавиши пробела выводится стрелочка (см. Рис.3):



Рисунок 3 – Результат ввода различных символов

Исходный код программы представлен в приложении А.

Состояние памяти после установки пользовательского обработчика прерывания представлено на рис.4:

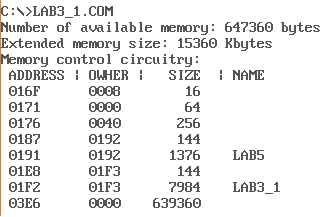


Рисунок 4 – Результат работы программы lab3\_1.com после запуска lab5.exe

Выгрузка пользовательского обработчика прерывания с ключом /un (см. Рис.5):



Рисунок 5 – Выгрузка пользовательского обработчика прерывания

Состояние памяти после выгрузки пользовательского обработчика прерывания представлено на Рис.6:

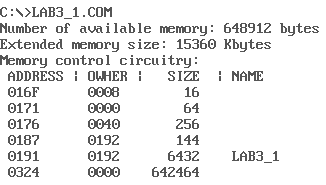


Рисунок 6 – Результат выполнения программы lab3\_1.com

**Выводы**

В процессе выполнения данной лабораторной работы была исследована возможность встраивания пользовательского обработчика прерываний в стандартный обработчик от клавиатуры.

**Контрольные вопросы.**

В: Какого типа прерывания использовались в работе?

О: в работе использовались программное (int 21h) и аппаратные (int 09h и 16h) прерывания.

В: Чем отличается скан код от кода ASCII?

О: скан код – в IBM-совместимых компьютерах код, присвоенный каждой клавише, с помощью которого драйвер клавиатуры распознает, какая клавиша была нажата. При нажатии любой клавиши контроллер клавиатуры распознаёт клавишу и посылает её скан-код в порт 60h.

**ПРИЛОЖЕНИЕ А**

**ТЕКСТ ИСХОДНОЙ ПРОГРАММЫ**

ASTACK SEGMENT STACK

    dw 100h dup (?)

ASTACK ENDS

CODE SEGMENT

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

; Ž–…„“›

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

ROUT PROC FAR

    jmp go\_

    SIGNATURA   DW 0ABCDh

    KEEP\_PSP    DW 0

    KEEP\_IP     DW 0

    KEEP\_CS     DW 0

    INT\_STACK   DW  100 dup (?)

    KEEP\_SS     DW 0

    KEEP\_AX     DW  ?

KEEP\_SP     DW 0

    go\_:

    mov KEEP\_SS, SS

    mov KEEP\_SP, SP

    mov KEEP\_AX, AX

    mov AX, seg INT\_STACK

    mov SS, AX

    mov SP, 0

    mov AX, KEEP\_AX

    push AX

    push ES

    push DS

    push DX

    push DI

    push CX

    mov AL, 0

    in AL, 60h

    cmp AL, 39h ; space

    je do\_req

    pushf

    call dword ptr CS:KEEP\_IP

    jmp skip

    do\_req:

    in AL, 61h

    mov AH, AL

    or AL, 80h

    out 61h, AL

    xchg AH, AL

    out 61h, AL

    mov AL, 20h

    out 20h, AL

    buf\_push:

    mov AL,0

    mov AH, 05h

    mov CL, 1Ah

    mov CH, 00h

    int 16h

    or AL, AL

    jz skip

    mov AX, 0040h

    mov ES, AX

    mov AX, ES:[1Ah]

    mov ES:[09h], AX

    jmp buf\_push

    skip:

    pop CX

    pop DI

    pop DX

    pop DS

    pop ES

    mov AL, 20h

    out 20h, AL

    pop AX

    mov AX, KEEP\_SS

   mov SS, AX

   mov AX, KEEP\_AX

   mov SP, KEEP\_SP

    iret

ROUT ENDP

LAST\_BYTE:

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

PRINT PROC

    push AX

    mov AH, 09h

    int 21h

    pop AX

    ret

PRINT ENDP

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

PROV\_ROUT PROC

    mov AH, 35h

    mov AL, 09h

    int 21h

    mov SI, offset SIGNATURA

    sub SI, offset ROUT

    mov AX, 0ABCDh

    cmp AX, ES:[BX+SI]

    je ROUT\_EST

    call SET\_ROUT

    jmp PROV\_KONEC

    ROUT\_EST:

    call DEL\_ROUT

    PROV\_KONEC:

    ret

PROV\_ROUT ENDP

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

SET\_ROUT PROC

    mov AX, KEEP\_PSP

    mov ES, AX

    cmp byte ptr ES:[80h],0

    je UST

    cmp byte ptr ES:[82h],'/'

    jne UST

    cmp byte ptr ES:[83h],'u'

    jne UST

    cmp byte ptr ES:[84h],'n'

    jne UST

    mov DX, offset INT\_NOT\_INST

    call PRINT

    ret

    UST:

    call SAVE\_STAND

    mov DX, offset INT\_INST

    call PRINT

    push DS

    mov DX, offset ROUT

    mov AX, seg ROUT

    mov DS, AX

    mov AH, 25h

    mov AL, 09h

    int 21h

    pop DS

    mov DX, offset LAST\_BYTE

    mov CL, 4

    shr DX, CL

    add DX, 1

    add DX, 40h

    mov AL, 0

    mov AH, 31h

    int 21h

    ret

SET\_ROUT ENDP

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

DEL\_ROUT PROC

    push DX

    push AX

    push DS

    push ES

    mov AX, KEEP\_PSP

    mov ES, AX

    cmp byte ptr ES:[82h],'/'

    jne UDAL\_KONEC

    cmp byte ptr ES:[83h],'u'

    jne UDAL\_KONEC

    cmp byte ptr ES:[84h],'n'

    jne UDAL\_KONEC

    mov DX, offset INT\_RM

    call PRINT

    CLI

    mov AH, 35h

    mov AL, 09h

    int 21h

    mov SI, offset KEEP\_IP

    sub SI, offset ROUT

    mov DX, ES:[BX+SI]

    mov AX, ES:[BX+SI+2]

    mov DS, AX

    mov AH, 25h

    mov AL, 09h

    int 21h

    mov AX, ES:[BX+SI-2]

    mov ES, AX

    mov AX, ES:[2ch]

    push ES

    mov ES, AX

    mov AH, 49h

    int 21h

    pop ES

    mov AH, 49h

    int 21h

    STI

    jmp UDAL\_KONEC2

    UDAL\_KONEC:

    mov DX, offset INT\_ALRD\_INST

    call PRINT

    UDAL\_KONEC2:

    pop ES

    pop DS

    pop AX

    pop DX

    ret

DEL\_ROUT ENDP

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

SAVE\_STAND PROC

    push AX

    push bx

    push ES

    mov AH,35h

    mov AL,09h

    int 21h

    mov KEEP\_CS, ES

    mov KEEP\_IP, BX

    pop ES

    pop BX

    pop AX

    ret

SAVE\_STAND ENDP

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

BEGIN:

    mov AX, DATA

    mov DS, AX

    mov KEEP\_PSP, ES

    call PROV\_ROUT

    xor AL, AL

    mov AH, 4Ch

    int 21H

CODE ENDS

DATA SEGMENT

    INT\_INST        db 'interrupt handler is installed',0DH,0AH,'$'

    INT\_RM          db 'interrupt handler is removed',0DH,0AH,'$'

    INT\_ALRD\_INST   db 'interrupt handler is already installed',0DH,0AH,'$'

    INT\_NOT\_INST    db 'interrupt handler is not installed',0DH,0AH,'$'

    STRENDL         db 0DH,0AH,'$'

DATA ENDS

END BEGIN