**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. 6383 |  | Никитин К.В. |
| Преподаватель |  | Губкин А.Ф. |

Санкт-Петербург

2018

**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.

**Описание функций и структур данных.**

|  |  |
| --- | --- |
| **Название** | **Назначение** |
| OUTPUT\_ALL | Вывод на экран |
| ROUTINE | Реализация пользовательского обработчика прерывания (печать D вместо a) |
| CHECK\_INT | Проверка на установку прерывания |
| Main | Головная функция |
| SET\_INT | Установка обработчика прерываний |
| UNLOAD\_INT | Выгрузка обработчика |
| CHECK\_UN | Проверка на /un для выгрузки |

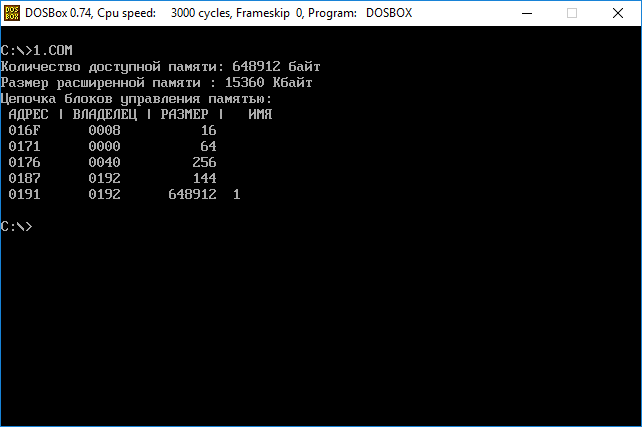
**Ход работы.**

1. Проверяет, установлено ли пользовательское прерывание.
2. Устанавливает резидентную функцию для обработки прерывания и настраивает вектор прерываний, если прерывание не установлено.
3. Выгрузка прерывания и восстановление стандартного вектора прерывания.

Пользовательское прерывание заменяет символ ‘a’ на символ ‘D’.

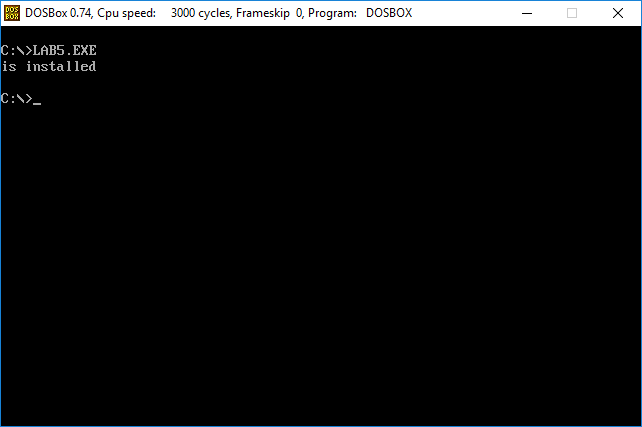
**Последовательность действий, выполняемых утилитой.**

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



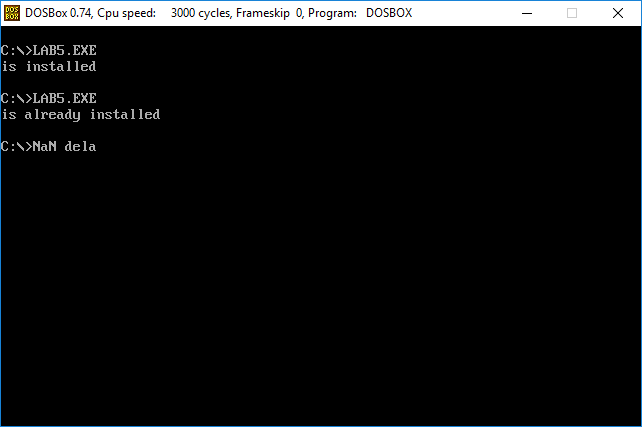
*Рисунок 1. Состояние памяти до загрузки резидента.*

1. Загрузка резидента в память показана на рис. 2.



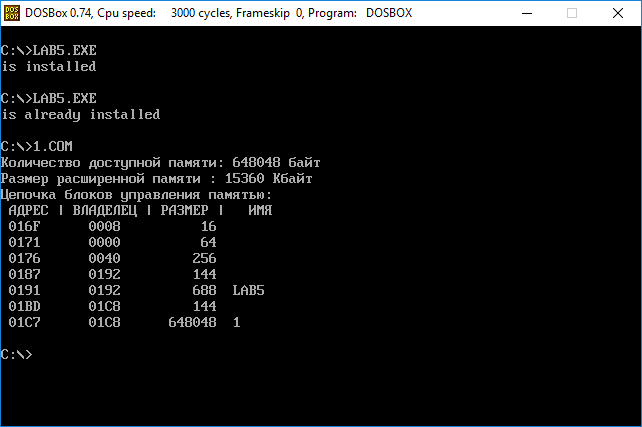
*Рисунок 2. Загрузка резидента в память.*

1. Попытка повторной загрузки резидента в память показана на рис. 3. Как и требовалось, вместо символа ‘k’ выводится символ ‘N’.



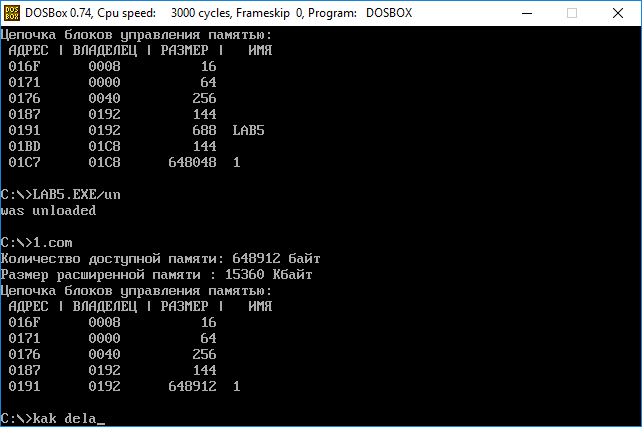
*Рисунок 3. Попытка повторной загрузки резидента.*

1. Состояние памяти при загрузке в неё резидента показано на рис. 4.



*Рисунок 4. Состояние памяти при загрузке в нее резидента*

1. Запуск отложенной программы с ключом /un показан на рис. 5.



*Рисунок 5. Запуск отложенной программы с ключом /un*

**Заключение.**

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

**Ответы на контрольные вопросы.**

1. *Какого типа прерывания использовались в работе?*

В работе использовались прерывания BIOS 9h,16h и пользовательское прерывание DOS 21h.

1. *Чем отличается скан код от кода ASCII?*

Скан код – это код нажатой клавиши, используемый для распознавания драйвером клавиатуры, а код ASCII – это код символа, закрепленного за клавишей

**Приложение А.**

**lr5.asm**

ASSUME CS:CODE, DS:DATA, SS:SSTACK, ES:NOTHING

SSTACK SEGMENT STACK

DW 64 DUP(?)

SSTACK ENDS

CODE SEGMENT

ROUTINE PROC FAR

jmp BEGIN

PSP1 DW ?

PSP2 DW ?

KEEP\_IP DW ?

KEEP\_CS DW ?

SIGN DW 01234h

INT\_STACK DW 64 dup (?)

KEEP\_SS DW ?

KEEP\_AX DW ?

KEEP\_SP DW ?

BEGIN:

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

in AL,60h

cmp AL,25h

jne STAND\_INT

jmp DO\_REQ

STAND\_INT:

pop ES

pop DS

pop DX

mov AX,CS:KEEP\_AX

mov SP,CS:KEEP\_SP

mov SS,CS:KEEP\_SS

jmp dword ptr CS:[KEEP\_IP]

DO\_REQ:

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

\_PUSH:

mov AH,05h

mov CL,'N'

mov CH,00h

int 16h

or AL,AL

jz \_QUIT

CLI

mov AX,ES:[1Ah]

mov ES:[1Ch],AX

STI

jmp \_PUSH

\_QUIT:

pop ES

pop DS

pop DX

mov AX,CS:KEEP\_AX

mov AL,20h

out 20h,AL

mov SP,CS:KEEP\_SP

mov SS,CS:KEEP\_SS

iret

ROUTINE ENDP

\_LASTBT:

CHECK\_INT PROC NEAR

push BX

push DX

push ES

mov AH,35h

mov AL,09h

int 21h

mov DX,ES:[BX + 11]

cmp DX,01234h

je \_INSTALL

mov AL,00h

jmp \_END

\_INSTALL:

mov AL,01h

jmp \_END

\_END:

pop ES

pop DX

pop BX

ret

CHECK\_INT ENDP

CHECK\_UN PROC NEAR

push ES

mov AX,PSP1

mov ES,AX

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

jne \_WITHOUT

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

jne \_WITHOUT

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

jne \_WITHOUT

cmp byte ptr ES:[85h], 0Dh

jne \_WITHOUT

cmp byte ptr ES:[86h], 0h

jne \_WITHOUT

mov AL,1h

\_WITHOUT:

pop ES

ret

CHECK\_UN ENDP

SET\_INT PROC NEAR

push AX

push BX

push DX

push ES

mov AH,35h

mov AL,09h

int 21h

mov KEEP\_IP,BX

mov KEEP\_CS,ES

push DS

lea DX,ROUTINE

mov AX,seg ROUTINE

mov DS,AX

mov AH,25h

mov AL,09h

int 21h

pop DS

lea DX,STR\_INSTALL

call OUTPUT\_ALL

pop ES

pop DX

pop BX

pop AX

ret

SET\_INT ENDP

UNLOAD\_INT PROC NEAR

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, STR\_UNLOAD

call OUTPUT\_ALL

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

mov AH,4Ch

int 21h

ret

UNLOAD\_INT ENDP

OUTPUT\_ALL PROC NEAR

push AX

mov AH,09h

int 21h

pop AX

ret

OUTPUT\_ALL ENDP

MAIN PROC FAR

mov BX,2Ch

mov AX,[BX]

mov PSP2,AX

mov PSP1,DS

mov DX,DS

sub AX,AX

xor BX,BX

mov AX,data

mov DS,AX

xor DX,DX

call CHECK\_UN

cmp AL,01h

je \_UNLOAD

call CHECK\_INT

cmp AL,01h

jne \_NOT

lea DX,STR\_IS\_ALR\_INSTALL

call OUTPUT\_ALL

jmp \_EXIT

\_NOT:

call SET\_INT

lea DX,\_LASTBT

mov CL,04h

shr DX,CL

add DX,1Bh

mov AX,3100h

int 21h

\_UNLOAD:

call CHECK\_INT

cmp AL,0h

je \_NOT2

call UNLOAD\_INT

jmp \_EXIT

\_NOT2:

lea DX, STR\_UNLOAD

call OUTPUT\_ALL

\_EXIT:

mov AH,4Ch

int 21h

MAIN ENDP

CODE ENDS

DATA SEGMENT

STR\_INSTALL DB 'is installed', 0dh, 0ah, '$'

STR\_NOT\_INSTALL DB 'is not installed', 0dh, 0ah, '$'

STR\_IS\_ALR\_INSTALL DB 'is already installed', 0dh, 0ah, '$'

STR\_UNLOAD DB 'was unloaded', 0dh, 0ah, '$'

DATA ENDS

END Main