

МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МОЭВМ

ОТЧЕТ
по практической работе № 5
по дисциплине «Операционные системы»
Тема: Сопряжение стандартного и пользовательского обработчика
прерываний

Студент гр. 8383

Степанов В.Д.

Преподаватель

Губкин А.Ф.

Санкт-Петербург

2020

Цель работы.

Исследование возможности встраивания пользовательского обработчика прерываний в стандартный обработчик от клавиатуры.

Выполнение работы.

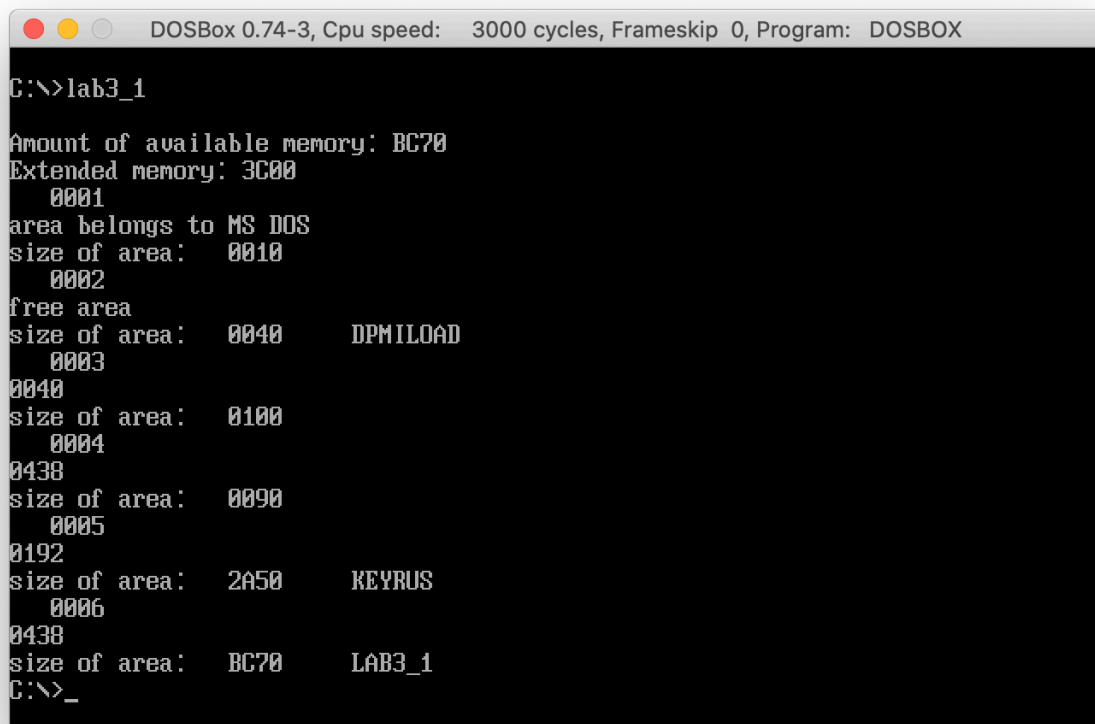
1. Был написан программный модуль типа .EXE, который выполняет следующий функции:

- 1) Проверяет, установлено ли пользовательское прерывание с вектором 1Ch.
- 2) Устанавливает резидентную функцию для обработки прерывания и настраивает вектор прерываний, если прерывание не установлено, и осуществляет выход по функции 4Ch прерывания int 21h.
- 3) Если прерывание установлено, то выводится соответствующие сообщение и осуществляется выход по функции 4Ch прерывания int 21h.
- 4) Выгрузка прерывания по соответствующему значению параметра командной строке /un.

Программа содержит код устанавливаемого прерывания в виде удаленной процедуры INTER. Он выполняет функции:

- 1) Сохраняет значения регистров в стек при входе и восстанавливает при выходе.
- 2) При выполнении тела процедуры анализирует 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.
- 5) При нажатии клавиши TAB печатается строка "DON'T TOUCH TAB".

2. С помощью лабораторной работы №3 посмотрим карту памяти до загрузки прерывания. Результат представлен на рисунке 1. Теперь загрузим пребывания, нажмем клавишу TAB, результат представлен на рисунке 2.

A screenshot of a DOSBox window. The title bar reads "DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: DOSBOX". The command prompt shows "C:\>lab3_1". The output displays memory information: "Amount of available memory: BC70", "Extended memory: 3C00", and a series of memory areas with their sizes and names. The last line shows "LAB3_1" at address BC70.

```
C:\>lab3_1

Amount of available memory: BC70
Extended memory: 3C00
0001
area belongs to MS DOS
size of area: 0010
0002
free area
size of area: 0040      DPMILOAD
0003
0040
size of area: 0100
0004
0438
size of area: 0090
0005
0192
size of area: 2A50      KEYRUS
0006
0438
size of area: BC70      LAB3_1
C:\>_
```

Рисунок 1 – Вывод карты памяти

После опять выведем карту памяти. Результат представлен на рисунке 3. В конце выгрузим прерывание и выведем карту памяти. Результат представлен на рисунке 4.

Контрольные вопросы.

1) Какого типа прерывания использовались в работе?

int 9h – прерывание от клавиатуры (аппаратное)

int 16h – сервис клавиатуры (программное)

int 21h – сервис DOS (программное)

2) Чем отличается скан код от кода ASCII?

Скан-код – это код, присвоенный каждой клавише, с помощью которого драйвер клавиатуры распознает, какая клавиша была нажата (в IBM-совместимых компьютерах). ASCII-код – это код печатных и непечатных символов.

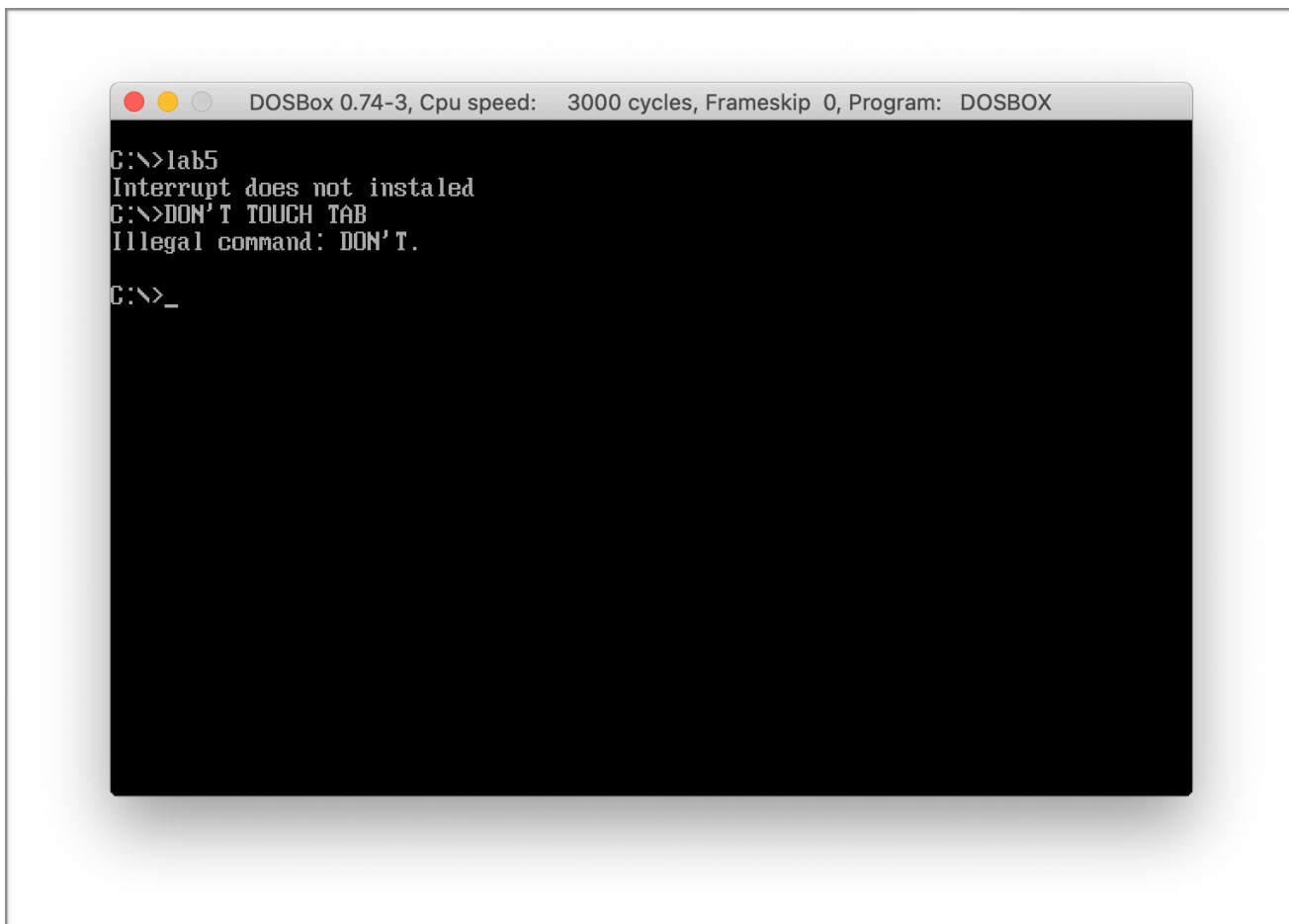


Рисунок 2 – Демонстрация работы прерывания

Выводы.

В ходе лабораторной работы была исследована возможности встраивания пользовательского обработчика прерываний в стандартный обработчик от клавиатуры.

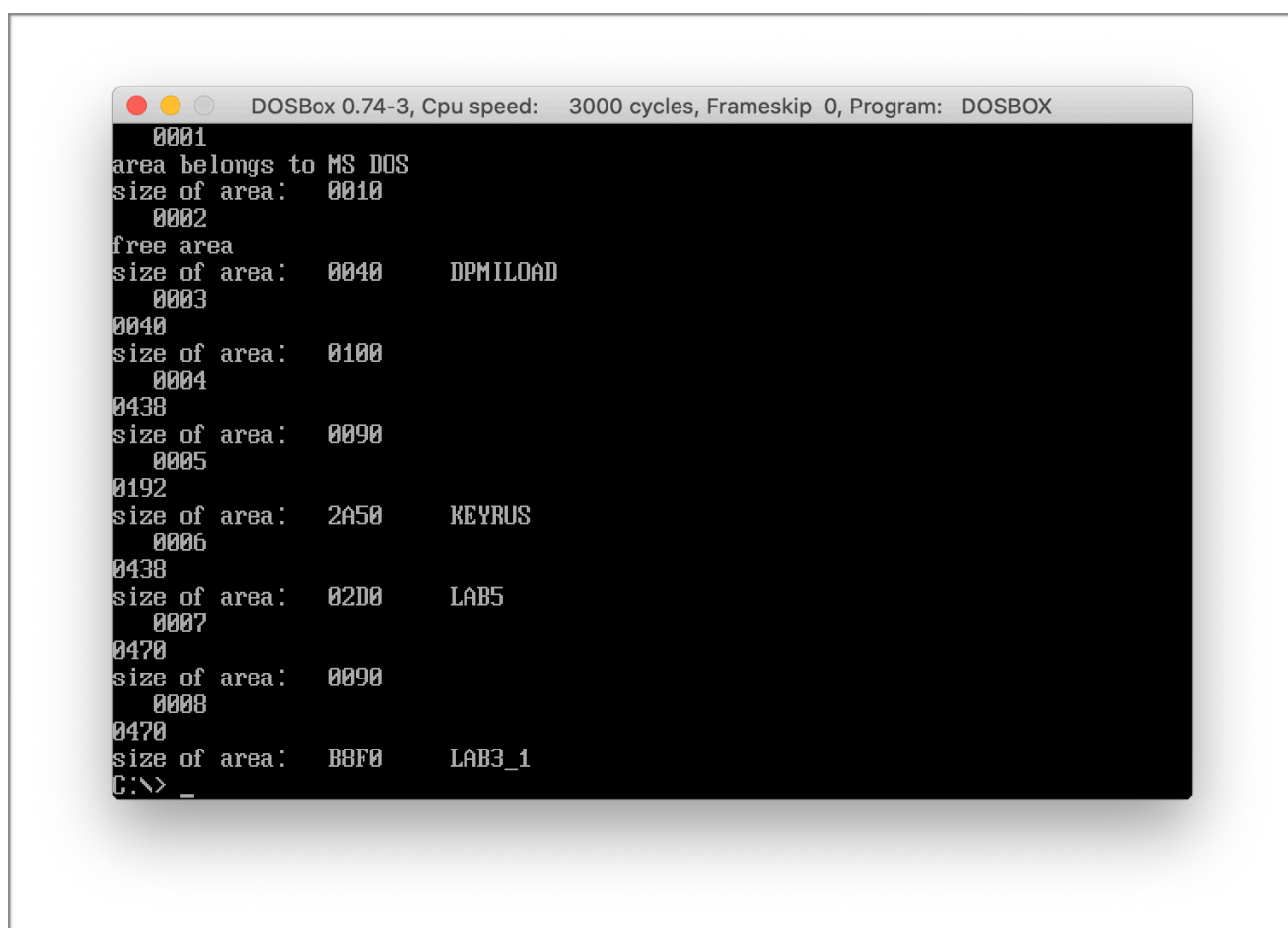


Рисунок 3 – Вывод карты памяти после загрузки прерывания

```
DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: DOSBOX
C:\> lab5/un
C:\>lab3_1
Amount of available memory: BC70
Extended memory: 3C00
0001
area belongs to MS DOS
size of area: 0010
0002
free area
size of area: 0040 DPMILOAD
0003
0040
size of area: 0100
0004
0438
size of area: 0090
0005
0192
size of area: 2A50 KEYRUS
0006
0438
size of area: BC70 LAB3_1
C:\>_
```

Рисунок 4 – Вывод карты памяти после выгрузки прерывания

ПРИЛОЖЕНИЕ

КОД ПРОГРАММЫ

```
LAB3CODE SEGMENT
    ASSUME CS: LAB3CODE, ds:DATA, es:NOTHING, SS:LAB3STACK

INTER PROC FAR

    jmp START
    STR_OUT db 'DON T TOUCH TAB!'
    INTER_ID dw 0804h
    KEEP_AX dw 0
    KEEP_SS dw 0
    KEEP_SP dw 0
    KEEP_IP dw 0
    KEEP_CS dw 0
    PSP_SEG dw 0
    INTER_STACK dw 128 dup(0)

START:
    mov KEEP_SS, ss
    mov KEEP_SP, sp
    mov KEEP_AX, ax
    mov ax, seg INTER_STACK
    mov ss, ax
    mov ax, offset INTER_STACK
    add ax, 256
    mov sp, ax
    push bx
    push cx
    push dx
    push si
    push ds
    push bp
    push es

    mov ax, seg STR_OUT
    mov ds, ax

    in al, 60h
    cmp al, 0Fh
    je TAB
```

```

pushf
call DWORD PTR cs:KEEP_IP
jmp FUN_END

TAB:
    in al, 61h
    mov ah, al
    or al, 80h
    out 61h, al
    xchg al, al
    out 61h, al
    mov al, 20h
    out 20h, al

WRITE_IN_BUFF:
    mov ah, 05h
    xor bx, bx
    mov [STR_OUT+3], 27h ; добавление ' в строку

CYCLE:
    mov cl, [STR_OUT+bx]
    mov CH, 00h
    int 16h
    inc bx
    cmp bx, 14
    jle CYCLE

    or AL, AL
    jz FUN_END

    mov ax, 0040h
    mov es, ax
    mov ax, es:[1Ah]
    mov es:[1Ch], ax
    jmp WRITE_IN_BUFF

FUN_END:

    pop es
    pop bp
    pop ds
    pop si
    pop DX

```



```

    pop cx
    pop bx
    mov sp, KEEP_SP
    mov ax, KEEP_SS
    mov ss, ax
    mov ax, KEEP_AX
    mov al, 20h
    OUT 20h, al
    IRET
    ret
INTER ENDP
INTER_END:
;-----

```

```

CHECK_INTER PROC NEAR

```

```

    push ax
    push bx
    push si

```

```

    MOV ah, 35H

```

```

;MOV al, 1CH
    mov al, 09H

```

```

    int 21h
    mov si, offset INTER_ID
    sub si, offset INTER
    mov ax, es:[bx + si]
    cmp ax, 0804h
    jne CHECK_END
    mov DOES_INTER_INSTAL, 1

```

```

CHECK_END:

```

```

    pop si

```

```

    pop bx
    pop ax

    ret
CHECK_INTER ENDP
;-----

PRINT PROC near

    push ax
    sub ax, ax
    mov ah, 9h
    int 21h
    pop ax

    ret
PRINT ENDP
;-----

CHECK_UN PROC NEAR

    push ax
    push es

    mov ax, PSP_SEG
    mov es, ax
    cmp byte ptr es:[82h], '/'
    jne CHECK_UN_END
    cmp byte ptr es:[83h], 'u'
    jne CHECK_UN_END
    cmp byte ptr es:[84h], 'n'
    jne CHECK_UN_END
    mov IS_UN, 1

CHECK_UN_END:
    pop es
    pop ax
    ret
CHECK_UN ENDP
;-----

UNSTE_INTER PROC NEAR
    CLI
    push ax
    push bx
    push dx

```

```

    push ds
    push es
    push si

    mov ah, 35h
    mov al, 09H
    int 21h

    mov si, offset KEEP_IP
    sub si, offset INTER

    mov DX, es:[bx + si]
    mov ax, es:[bx + si + 2]
    push ds
    mov ds, ax
    mov ah, 25h
    mov al, 09h
    int 21h
    pop ds
    mov ax, es:[bx + si + 4]
    mov es, ax
    push es
    mov ax, es:[2Ch]
    mov es, ax
    mov ah, 49h
    int 21h
    pop es
    mov ah, 49h
    int 21h

    pop si
    pop es
    pop ds
    pop dx
    pop bx
    pop ax
    STI

    ret
UNSTE_INTER ENDP
;-----
SET_INTER PROC NEAR

    push ax

```

```

push bx
push cx
push dx
push ds
push es

mov ah, 35h
mov al, 09H
int 21H
mov KEEP_IP, bx
mov KEEP_CS, es

push ds
mov DX, offset INTER
mov ax, seg INTER
mov ds, ax
mov ah, 25H
mov al, 09H
int 21H
pop ds

mov dx, offset INTER_END
add dx, 10Fh
mov cl, 4h
shr dx, cl
inc dx
xor ax, ax
mov ah, 31h
int 21h

pop es
pop ds
pop dx
pop cx
pop bx
pop ax
ret
SET_INTER ENDP
;-----
MAIN PROC

```

```

push ds
xor ax, ax
push ax
mov ax, DATA
mov ds, ax
mov PSP_SEG, es

call CHECK_INTER
call CHECK_UN

mov al, IS_UN
cmp al, 1
je UNLOAD_INTER

mov al, DOES_INTER_INSTAL
cmp al, 1
jne CHECK_NOT_OK
mov dx, offset INTER_INSTAL
call PRINT
jmp MAIN_END

CHECK_NOT_OK:
mov dx, offset INTER_DOES_NOT_INSTAL
call PRINT
call SET_INTER
jmp MAIN_END

UNLOAD_INTER:
mov al, DOES_INTER_INSTAL
cmp al, 1
jne FAIL_UNLOAD

call UNSTE_INTER

jmp MAIN_END

FAIL_UNLOAD:
mov dx, offset INTER_DOES_NOT_INSTAL
call PRINT

MAIN_END:
mov ah, 4Ch
int 21h

```

```

MAIN ENDP
LAB3CODE ENds
;-----
LAB3STACK SEGMENT STACK
    dw 128 dup(0)
LAB3STACK ENds
;-----
DATA SEGMENT
    IS_UN db 0
    DOES_INTER_INSTAL db 0
    INTER_INSTAL db 'Interrupt instaled$'
    INTER_DOES_NOT_INSTAL db 'Interrupt does not instaled$'
DATA ENds
;-----

END MAIN

```