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

ОТЧЕТ

по лабораторной работе №5
по дисциплине «Операционные системы»
Тема: Сопряжение стандартного и пользовательского обработчиков прерывания

Студент гр.8382 _____ Синельников М.Р

Преподаватель _____ Ефремов М.А

Санкт-Петербург
2020

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 определённые действия, если скан-код совпадает с определёнными кодами, которые он должен обрабатывать. Если скан-код не совпадает с этими кодами, то управление передаётся стандартному прерыванию.

Ход работы.

1) Пользовательский обработчик перехватывает нажатие клавиши «0» и помещает в буфер клавиатуры символ «D». Остальные нажатия передаются на стандартный обработчик.

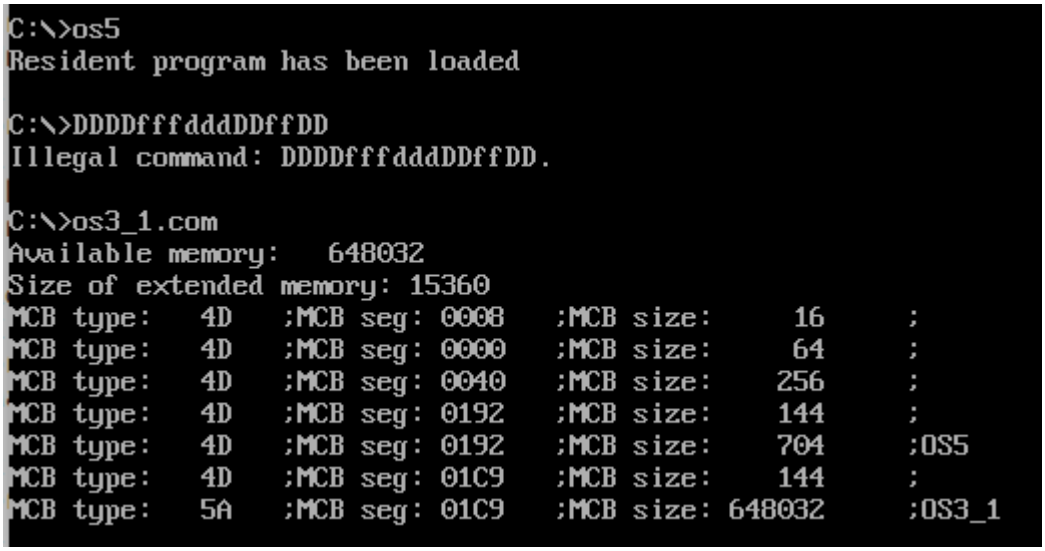

рисунок 1 — состояние памяти после загрузки пользовательского обработчика

2) Повторный запуск программы

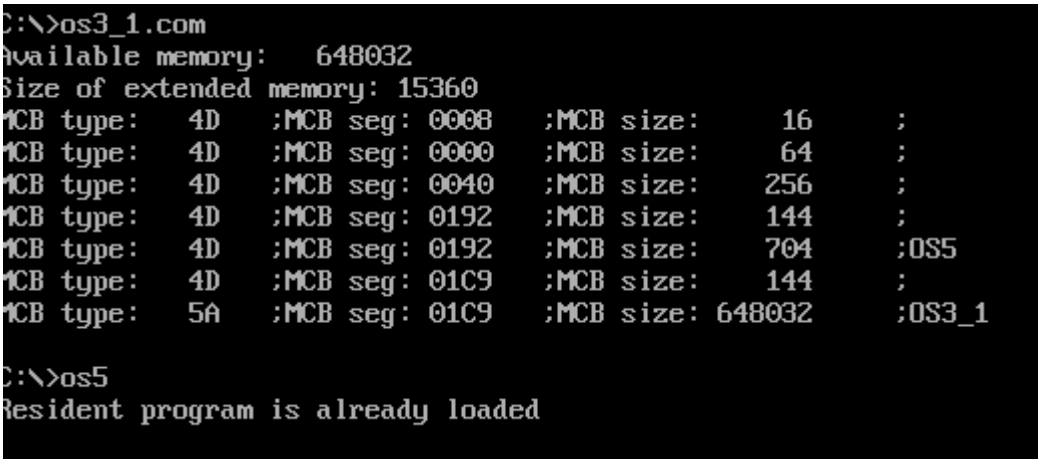

рисунок 2 - повторная загрузка обработчика прерывания

3) Выгрузка обработчика и состояние памяти

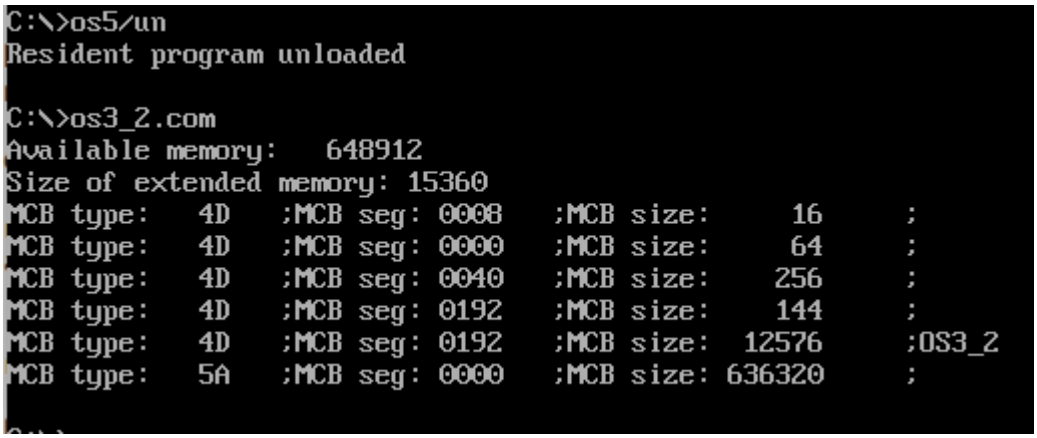

рисунок 3 — выгрузка обработчика и состояние памяти

Контрольные вопросы.

1) Какого типа прерывания использовались в программе?

Был реализован пользовательский обработчик для прерывания от клавиатуры 09h. Использовались также системные прерывания 21h и 10h.

2) Чем отличается скан-код от кода ASCII?

Скан-код — номер клавиши, ASCII — номер символа в таблице кодировки.

Вывод.

В ходе выполнения работы был реализован пользовательский обрабочик прерывания от клавиатуры.

Приложение А

Исходный код файла OS5.asm

```
CODE    SEGMENT

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

HANDLER PROC FAR
        jmp HANDLER_CODE

        HANDLER_DATA:
                HANDLER_SIGNATURE dw 6000h
                keep_ip dw 0
                KEEP_CS dw 0
                keep_psp dw 0
                KEEP_SS         dw 0
                KEEP_SP         dw 0
                KEEP_AX         dw 0
                REQ_KEY  db 0Bh
                HANDLER_STACK dw 100 DUP(0)
                Top:



        HANDLER_CODE:
        mov KEEP_SS, ss
        mov KEEP_SP, sp
        mov KEEP_AX, ax
        mov ax, seg HANDLER_STACK
        mov ss, ax
        mov sp, offset Top
        push cx
        push dx
        push si
        push es

        in al, 60h
        cmp al, REQ_KEY
        je do_req
```

```
pushf
call        dword ptr CS:KEEP_ip
jmp continue

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

mov ah,05h
mov cl,'D'
mov ch,00h
int 16h
or al,al
jnz skip

jmp continue

skip:

mov ax, 0040h
mov es, ax
mov si, 001ah
mov ax, es:[si]
mov si, 001ch
mov es:[si], ax

jmp do_req

continue:
pop es
pop si
pop dx
pop cx
```

```
                mov sp, KEEP_SP
                mov ax, KEEP_SS
                mov ss, ax
                mov ax, KEEP_AX
                mov al, 20h
                out 20h, al

                iret
HANDLER ENDP
HANDLER_END:

Un_check  PROC          FAR
                cmp byte ptr es:[82h], '/'
                jne FALSE
                cmp byte ptr es:[83h], 'u'
                jne FALSE
                cmp byte ptr es:[84h], 'n'
                jne FALSE

                jmp TRUE

                FALSE:
                mov ax, 0
                ret
                TRUE:
                mov ax, 1
                ret
Un_check  ENDP

check_on_09h PROC FAR
                push bx
                push si
                push es
                mov si, offset HANDLER_SIGNATURE
                sub si,offset HANDLER
                mov ah,35h
                mov al,09h
                int 21h
                mov ax,es:[bx+si]
                mov bx,HANDLER_SIGNATURE
```

```
                cmp ax,bx
                je CHECK_TRUE
                mov ax,0
                jmp finish_09h

                CHECK_TRUE:
                mov ax,1
                finish_09h:
                pop es
                pop si
                pop bx
                ret
check_on_09h endp

Keep_interr     PROC
                push ax
                push bx
                push es
                mov ah, 35h
                mov al, 09h
                int 21h
                mov keep_ip, bx
                mov keep_cs, es
                pop es
                pop bx
                pop ax
                ret

Keep_interr     ENDP

Load_handler    PROC

                push ax
                push bx
                push dx
                push es
                call keep_interr
                push ds
                mov dx,offset Handler
                mov ax,seg Handler
                mov ds,ax
```

```
                mov ah,25h
                mov al,09h
                int 21h
                pop ds
                pop es
                pop dx
                pop bx
                pop ax
                ret

Load_handler  ENDP

Unload_handler  PROC
                push ax
                push bx
                push dx
                push es
                push si
                mov si,offset keep_ip
                sub si,offset Handler
                mov ah, 35h
                mov al,09h
                int 21h
                cli
                push ds
                mov dx,es:[bx + si]
                mov ax,es:[bx + si + 2]
                mov ds,ax
                mov ah,25h
                mov al,09h
                int 21h
                pop ds
                sti

                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 dx
                pop bx
                pop ax
                ret

Unload_handler  ENDP

Make_resident  PROC
                mov dx, offset HANDLER_END
                mov cl, 4
                shr dx, cl

                add dx, 16h
                inc dx

                mov ax, 3100h
                int 21h
Make_resident  ENDP

print_message     PROC
                push ax
                mov ah, 09h
                int 21h
                pop ax
                ret
print_message           ENDP

Main     PROC
                push ds
                xor ax, ax
                push ax
                mov ax, data
                mov ds, ax
```

```
                mov Keep_PSP, es
                call check_on_09h
                cmp ax,1
                jne loading
                call Un_check
                cmp ax, 1
                jne alr_loaded
                call Unload_handler
                lea dx, Message2
                call print_message
                mov ax, 4c00h
                int 21h
                jmp finish
        loading:
                call Load_Handler
                lea DX, Message1
                call print_message
                call Make_resident
        alr_loaded:
                lea dx, Message3
                call print_message
                mov ax, 4C00h
                int 21h

                finish:

Main    ENDP
CODE                    ENDS

AStack          SEGMENT  STACK
    DW 64 DUP(0)
AStack  ENDS

DATA            SEGMENT
  Message1      db 'Resident program has been loaded', 0dh, 0ah, '$'
  Message2        db 'Resident program unloaded', 0dh, 0ah, '$'
  Message3              db 'Resident program is already loaded', 0dh, 0ah, '$'
DATA            ENDS
        END Main
```