

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. 9382

Павлов Р.В.

Преподаватель

Ефремов М.А.

Санкт-Петербург

2021

Постановка задачи.

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.

Функции и структуры данных:

| Название процедуры | Описание процедуры |
|--------------------|---|
| main | Вызов процедур загрузки и выгрузки, проверки на наличие пользовательского вектора |
| print_string | Вывод указанной строки |
| inter | Обработчик прерываний от клавиатуры |
| load | Загрузка обработчика прерываний в память |
| unload | Выгрузка обработчика прерываний из памяти |
| is_lodd | Проверка на наличие обработчика прерывания в памяти |
| is_tounld | Проверка на флаг выгрузки |
| loading_msg | Сообщение об успешной загрузке обработчика в память |
| loaded | Сообщение о наличии обработчика в памяти |
| unloading_msg | Сообщение о выгрузке обработчика из памяти |

| | |
|-------------|---|
| missing_msg | Сообщение об отсутствии обработчика в памяти |
| lodd | Флаг наличия обработчика в памяти |
| tounld | Флаг выгрузки |

Ход работы.

1) Написан обработчик прерываний. Процедура получает сигнал от клавиатуры, в ответ на который устанавливает разрешающий бит и переходит к обработке символа (те символы, которые не рассматриваются, передаются для обработки стандартному обработчику прерываний). После этого обработанный символ заносится в буфер и выводится на экран.

```
C:\>etu8
Illegal command: etu8.

C:\>lab5
Прерывание загружено в память

C:\>leti
Illegal command: leti.

C:\>1234567i90 qwlreytiop
Illegal command: 1234567i90.
```

2) Реализованы процедуры загрузки обработчика в память и выгрузки из неё. При загрузке программе(сегменту) выделяется память, а вектор продолжает указывать на неё даже после завершения работы. При выгрузке исходный вектор восстанавливается, а память, используемая программой ранее, освобождается.

3) Реализованы процедуры проверки пользовательского ввода на предмет выбора загрузки или выгрузки.

```
C:\>lab5
Прерывание уже в памяти

C:\>lab5 -d
Прерывание выгружено из памяти

C:\>lab5 -d
Прерывания нет в памяти
```

4) В главной процедуре на основании ввода и текущего состояния обработчика принимается решение о выводе того или иного сообщения и предприятии какого-либо действия либо бездействии.

```
C:\>lab5
Прерывание загружено в память

C:\>lab3
Доступный размер памяти: 643952 байт
Размер расширенной памяти: 246720 байт
Блок№ 001 Адрес: 016F; PSP: 0008; Размер: 16 байт; SC/CD:
Блок№ 002 Адрес: 0171; PSP: 0000; Размер: 64 байт; SC/CD:
Блок№ 003 Адрес: 0176; PSP: 0040; Размер: 256 байт; SC/CD:
Блок№ 004 Адрес: 0187; PSP: 0192; Размер: 144 байт; SC/CD:
Блок№ 005 Адрес: 0191; PSP: 0192; Размер: 4784 байт; SC/CD: LAB5
Блок№ 006 Адрес: 02BD; PSP: 02C8; Размер: 144 байт; SC/CD:
Блок№ 007 Адрес: 02C7; PSP: 02C8; Размер: 643952 байт; SC/CD: LAB3

C:\>lab5 -d
Прерывание выгружено из памяти

C:\>lab3
Доступный размер памяти: 648912 байт
Размер расширенной памяти: 246720 байт
Блок№ 001 Адрес: 016F; PSP: 0008; Размер: 16 байт; SC/CD:
Блок№ 002 Адрес: 0171; PSP: 0000; Размер: 64 байт; SC/CD:
Блок№ 003 Адрес: 0176; PSP: 0040; Размер: 256 байт; SC/CD:
Блок№ 004 Адрес: 0187; PSP: 0192; Размер: 144 байт; SC/CD:
Блок№ 005 Адрес: 0191; PSP: 0192; Размер: 648912 байт; SC/CD: LAB3
```

Ответы на контрольные вопросы.

Сегментный адрес недоступной памяти

1. Прерывания DOS – 21h и BIOS – 09h и 16h.
2. Коды ASCII используются для кодирования текста, а скан-коды – определённые идентификаторы клавиш на клавиатуре, которые распознаются драйвером.

Выводы.

В результате выполнения лабораторной работы реализована операция загрузки обработчика прерывания в память, при этом в случаях, когда обработать какой-либо сигнал нельзя, он передаётся стандартному обработчику прерываний.

ПРИЛОЖЕНИЕ А. ИСХОДНЫЙ КОД

- имя файла : lab5.asm

```
astack segment stack
    dw 64 dup(0)
astack ends

data segment
    loading_msg db 'Прерывание загружено в память', 10, 13, '$'
    loaded db 'Прерывание уже в памяти', 10, 13, '$'
    unloading_msg db 'Прерывание выгружено из памяти', 10, 13, '$'
    missing_msg db 'Прерывания нет в памяти', 10, 13, '$'
    lodd db 0
    tounld db 0
data ends

code segment
assume cs:code, ds:data, ss:astack

inter proc far
    jmp init
        substack dw 128 dup(0)           ; Вложенный стек, хранилище для cs
и ip
        ip1 dw 0
        cs1 dw 0
        psp1 dw 0
        ax1 dw 0
        ss1 dw 0
        sp1 dw 0
        id dw 0a492h
        k db 0

    init:
        mov ax1, ax
        mov sp1, sp
        mov ss1, ss
        mov ax, seg substack
        mov ss, ax
        mov ax, offset substack         ; настройка стека
        add ax, 256
        mov sp, ax

        push ax
        push bx
        push cx
        push dx
        push si
        push es
        push ds

        in al, 60h
        cmp al, 12h ; сравнение скан-кодов
        je k1
        cmp al, 14h
        je k2
        cmp al, 16h
```

```

je k3
cmp al, 9h
je k4

pushf
call dword ptr cs:ipl ; передача управления стандартному обработчику,
если ни один не подошёл
jmp exit1

k1:
    mov k, 'l'
    jmp recall
k2:
    mov k, 'e' ; запись соответствующего символа
    jmp recall
k3:
    mov k, 't'
    jmp recall
k4:
    mov k, 'i'

recall:
    in al, 61h
    mov ah, al
    or al, 80h
    out 61h, al ; установка бита разрешения
    xchg al, al
    out 61h, al
    mov al, 20h
    out 20h, al

to_buff:
    mov ah, 05h
    mov cl, k
    mov ch, 00h
    int 16h
    or al, al
    jz exit1
    mov ax, 0040h ; запись символа в буфер
    mov es, ax
    mov ax, es:[lah]
    mov es:[lch], ax
    jmp to_buff

exit1:
    pop ds
    pop es
    pop si
    pop dx
    pop cx
    pop bx
    pop ax

    mov sp, sp1
    mov ax, ssl ; восстановление исходных значений сег-
ментов программы
    mov ss, ax
    mov ax, ax1

    mov al, 20h
    out 20h, al

```



```

        iret
inter endp

seg_end:

load proc
    push ax
    push bx
    push cx
    push dx
    push es
    push ds

    mov ax, 3509h
    int 21h
    mov cs1, es
    mov ip1, bx
    mov ax, seg inter           ; получение вектор
    mov ds, ax
    mov dx, offset inter

    mov ax, 2509h
    int 21h
    pop ds
    mov dx, offset seg_end     ; запись в него своего обработчика
    mov cl, 4
    shr dx, cl
    add dx, 110h
    mov ax, 3100h              ; программа остаётся в памяти
    int 21h

    pop es
    pop dx
    pop cx
    pop bx
    pop ax
    ret
load endp

unload proc
    cli
    push ax
    push bx
    push dx
    push ds
    push es
    push si

    mov ah, 35h
    mov al, 09h
    int 21h
    mov si, offset ip1
    sub si, offset inter
    mov dx, es:[bx + si]       ; вернуть прежние значения cs и ip
вектора
    mov ax, es:[bx + si + 2]

    push ds
    mov ds, ax
    mov ah, 25h
    mov al, 09h                ; записать его заново

```

```

        int 21h
        pop ds
        mov ax, es:[bx + si + 4]
        mov es, ax
        push es
        mov ax, es:[2ch] ; очистить память, ранее занимаемую
программой
        mov es, ax
        mov ah, 49h
        int 21h
        pop es
        mov ah, 49h
        int 21h

        sti
        pop si
        pop es
        pop ds
        pop dx
        pop bx
        pop ax
        ret
unload endp

is_lodd proc near
        push es
        push ax
        push bx
        push si

        mov ah, 35h
        mov al, 09h
        int 21h
        mov si, offset id
        sub si, offset inter ; если id не соответствует заданному, значит,
стандартное прерывание
        mov ax, es:[bx + si]
        cmp ax, id
        jne exit2
        mov lodd, 1

        exit2:
                pop si
                pop bx
                pop ax
                pop es
        retn
is_lodd endp

is_tounld proc
        push ax
        push es

        mov ax, psp1
        mov es, ax
        cmp byte ptr es:[82h], '-'
        jne exit3
        cmp byte ptr es:[83h], 'd' ; если в хвост консоли тэг "-d",
программа выгружается из памяти
        jne exit3
        mov tounld, 1

```

