# 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		Кузьмин Д. И.
Преподаватель		Ефремов М. А.
	Санкт-Петербург	

2021

#### 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.

#### Задание.

Шаг 1. Для выполнения лабораторной работы необходимо написать и отладить программный модуль типа .ЕХЕ, который выполняет такие же функции, как в программе ЛР 4, а именно: 1) Проверяет, установлено ли пользовательское прерывание с вектором 09h. 2) Если прерывание не устанавливает резидентную функцию обработки установлено TO, ДЛЯ прерывания и настраивает вектор прерываний. Адрес точки входа в стандартный обработчик прерывания находится в теле пользовательского обработчика. Осуществляется выход по функции 4Ch прерывания int 21h. 3) Если прерывание установлено, то выводится соответствующее сообщение и осуществляется выход по функции 4Ch прерывания int 21h. Выгрузка прерывания по соответствующему значению параметра в командной строке /un. прерывания состоит в восстановлении стандартного вектора И освобождении памяти, занимаемой резидентом. осуществляется выход по функции 4Ch прерывания int 21h. Для того чтобы проверить установку прерывания, можно поступить следующим образом. Прочитать адрес, записанный в векторе прерывания. Предположим, что этот адрес указывает на точку входа в установленный резидент. На определенном, известном смещении в теле резидента располагается сигнатура, некоторый код, который идентифицирует резидент. Сравнив известное значение сигнатуры с реальным кодом, находящимся в резиденте, можно определить, установлен ли

резидент. Если значения совпадают, то резидент установлен. Длину кода сигнатуры должна быть достаточной, чтобы сделать случайное совпадение маловероятным. Программа должна содержать код устанавливаемого прерывания в виде удаленной процедуры. Этот код будет работать после установки при возникновении прерывания. Он должен выполнять следующие функции: 1) Сохранить значения регистров в стеке при входе и восстановить их при выходе. 2) При выполнении тела процедуры анализируется 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.

- Шаг 2. Запустите отлаженную программу и убедитесь, что резидентный обработчик прерывания 09h установлен. Работа прерывания проверяется введением различных символов, обрабатываемых установленным обработчиком и стандартным обработчиком.
- Шаг 3. Также необходимо проверить размещение прерывания в памяти. Для этого запустите программу ЛР 3, которая отображает карту памяти в виде списка блоков МСВ. Полученные результаты поместите в отчет.
- Шаг 4. Запустите отлаженную программу еще раз и убедитесь, что программа определяет установленный обработчик прерываний. Полученные результаты поместите в отчет.
- Шаг 5. Запустите отлаженную программу с ключом выгрузки и убедитесь, что резидентный обработчик прерывания выгружен, то есть сообщения на экран не выводятся, а память, занятая резидентом освобождена. Для этого также следует запустить программу ЛР 3. Полученные результаты поместите в отчет.

Шаг 6. Ответьте на контрольные вопросы.

## Выполнение работы.

1) Первым шагом было создание пользовательского обработчика

прерываний, который срабатывает на каждую нажатую клавишу и в зависимости от нее либо обрабатывает ее (если нажата клавиша 1, то выводится символ D), либо переход к стандартному обработчику прерываний.

- 2) Далее была создана функция, проверяющая, установлено ли пользовательское прерывание и если нет, то устанавливает его, загружая при этом резидентную программу.
- 3) Проверка прерываний осуществляется с помощью функции 31h прерывания int 21h, а установка с помощью функции 25h.
- 4) Далее была релизована проверка параметра /un, отвечающего за выгрузку прерывания. В случае, если параметр указан, при помощи функции 49h прерывания int 21h освобождается память, занимаемая резидентной программой и восстанавливается исходный вектор прерываний.

Исходный код см. в приложении А Результаты работы программы см. в приложении Б

#### Контрольные вопросы.

- Какого типа прерывания использовались в работе?
   int 21h функции DOS
   int 09h прерывание от клавиатуры
- 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. ASCII код – это уникальный код для каждого символа из таблицы ASCII.

#### Выводы.

Были получены навыки разработки программ, встраивающих пользовательскый обработчик прерываний в стандартный обработчик от клавиатуры

# приложение а. исходный код

#### Файл lab5.asm

```
ASTACK
           SEGMENT
                            STACK
          DW 512 DUP (?)
ASTACK
          ENDS
DATA SEGMENT
resident set db 'Interruption is already loaded' , 13, 10, '$'
resident not set db 'Interruption loaded succesfully', 13, 10, '$'
unload db 'Interruption unloaded', 13, 10, '$'
param db ' /un'
DATA ENDS
CODE SEGMENT
     .386
      ASSUME CS:CODE, DS:DATA, SS:ASTACK
; Процедуры
m1:
ROUT PROC FAR
                 jmp start
                 sum dw 0
                 keep ip dw 0
                 keep_cs dw 0
                 sign dw 0A35Fh
                 start adress dw 0
                 keep sp dw 0
                 keep ss dw 0
                 int9 vect dd 0
                 REQ KEY db 2h
                new stack dw 32 dup (?)
                 start:
                mov keep_sp, sp
                 mov keep ss, ss
                mov ss, new stack
                mov sp, 0
                push ax
                 in al, 60h
                 cmp al, REQ KEY ;обрабывается скан-код для символа '1'
                 је do req ;если совпадает, то переход к пользовательской
обработке прерывания
                pop ax
                mov ss, keep ss
                 mov sp, keep sp
                 jmp cs:[int9 vect];
           do req:
                 pop ax
                push ax
                 in al, 61h ;взять значение порта управления
клавиатурой
                mov ah, al
                               ; сохранить его
                mov an, aı ; сохранить его or al, 80h ;установить бит разрешения для клавиатуры
```

```
out 61h, al \,\,; и вывести его в управляющий порт
                   xchg ah, al ;извлечь исходное значение порта out 61h, al ;и записать его обратно mov al, 20h ;послать сигнал "конец прерывания" out 20h, al ; контроллеру прерываний 8259
                   out 20h, al
                                     ; контроллеру прерываний 8259
             ; символ 'D' записывается в буфер клавиатуры
                   pop ax
                   mov ah, 05h
                   mov cl, 'D'
                   mov ch,00h;
                   int 16h ;
                   or al, al ; проверка переполнения буфера
                   jnz skip; если переполнен идем skip
                   jmp exrout
            skip: ; очистить буфер и повторить
                   push es
                   CLI
                   xor ax, ax
                   MOV es, ax
                   MOV al, es:[41AH]
                   MOV es: [41CH], al
                   STI
                   pop es
            exrout:
                   mov ss, keep_ss
                   mov sp, keep sp
                   IRET
ROUT ENDP
m2:
WRITE MSG PROC near
                   push ax
                   mov AH, 09h
                   int 21h
                   pop ax
                   ret
WRITE MSG ENDP
SET INTERRUPT PROC NEAR
                   mov ah, 35h
                   mov al, 09h
                   int 21h
                   mov keep cs, es
                   mov keep ip, bx
                   mov int9 vect + 2, es
                   mov word ptr int9 vect, bx
                   PUSH DS
                   PUSH AX
                   PUSH DX
                   MOV DX, OFFSET ROUT
```

```
MOV AX, SEG ROUT
                MOV DS, AX
                MOV AH, 25H
                MOV AL, 09h
                INT 21H
                POP DX
                POP AX
                POP DS
                RET
SET INTERRUPT ENDP
LOAD TO RESIDENT PROC NEAR
                push ax
                push bx
                push dx
                push cx
                mov dx, 0A00h
                mov cl,4h
                shr dx,cl
                inc dx
                mov ah, 31h
                int 21h
                pop cx
                pop dx
                pop bx
                pop ax
                ret
LOAD_TO_RESIDENT ENDP
;-----
RESTORE VECTOR PROC NEAR
                CLI
                PUSH DS
                push dx
                push ax
                MOV DX, es:keep_ip
                MOV AX, es:keep_cs
                MOV DS, AX
                MOV AH, 25H
                MOV AL, 09h
                INT 21H
                pop ax
                pop dx
                POP DS
                STI
                RET
RESTORE VECTOR ENDP
```

```
;-----
CHECK_VECTOR PROC NEAR
                PUSH AX
                PUSH BX
                PUSH ES
                PUSH SI
                MOV AH, 35H
                MOV AL, 09h
                INT 21H
                mov ax, 0A35Fh ;уникальное значение
                cmp ax,es:sign
                jne setres
                call CHECK PARAM ;если установлено, переход к проверку
параметра /un
                jmp endthis
                setres: ;если прерывание не установлено
                mov dx, offset resident not set
                call WRITE MSG
                call SET INTERRUPT
                call LOAD TO RESIDENT
                endthis:
                POP SI
                POP ES
                POP BX
                POP AX
                RET
CHECK VECTOR ENDP
UNLOAD INTERRUPTION PROC NEAR
                call RESTORE VECTOR
                mov ax, es:start adress
                mov es,ax
                push es
                mov ax,es:[2ch] ;среда
                mov es,ax
                mov ah, 49h
                int 21h
                pop es
                mov ah, 49h ; резидентная часть
                int 21h
                ret
UNLOAD INTERRUPTION ENDP
CHECK PARAM PROC NEAR
```

```
push es
mov es, start_adress
mov cx, 4
mov di, 81h
mov si, offset param
repe cmpsb
jne notequal
pop es
call UNLOAD INTERRUPTION
mov dx, offset unload
call WRITE_MSG
jmp ex
notequal:
pop es
mov dx, offset resident set
call WRITE_MSG
ex:
ret
```

CHECK PARAM ENDP

MAIN PROC FAR

mov ax,DATA
mov ds,ax
mov start\_adress, es
call CHECK\_VECTOR
mov ah, 4ch
int 21h

MAIN ENDP CODE ENDS END MAIN

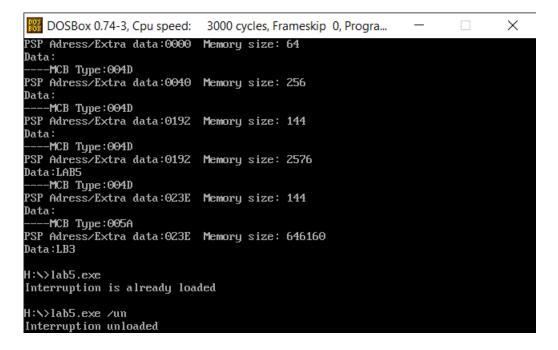
## ПРИЛОЖЕНИЕ Б. ДЕМОНСТРАЦИЯ РАБОТЫ ПРОГРАММЫ

```
BOSBox 0.74-3, Cpu speed:
                                3000 cycles, Frameskip 0, Progra...
                                                                                     X
Object filename [lab5.OBJ]:
Source listing [NUL.LST]:
Cross-reference [NUL.CRF]:
  49968 + 453197 Bytes symbol space free
      0 Warning Errors
      0 Severe Errors
H:N>link lab5.obj
Microsoft (R) Overlay Linker Version 3.64
Copyright (C) Microsoft Corp 1983–1988. All rights reserved.
Run File [LAB5.EXE]:
List File [NUL.MAP]:
Libraries [.LIB]:
H:\>lab5.exe
Interruption loaded succesfully
H:∖>DDDDDDD
Illegal command: DDDDDD.
```

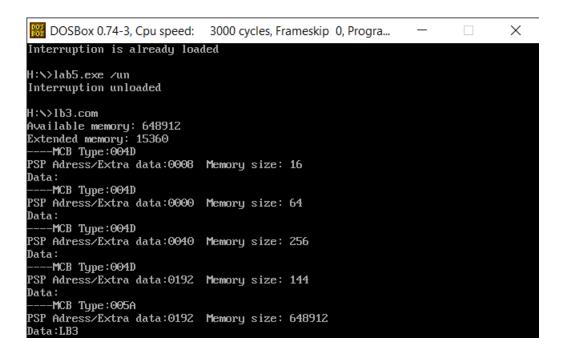
Установка резидентного обработчика прерывания и демонстрация того, что он обрабатывает нажатые клавишы

```
BB DOSBox 0.74-3, Cpu speed:
                                                           _ _
                                                                            X
                            3000 cycles, Frameskip 0, Progra...
Available memory: 646160
Extended memory: 15360
 ---MCB Type:004D
PSP Adress/Extra data:0008 Memory size: 16
Data:
 ---MCB Type:004D
PSP Adress/Extra data:0000 Memory size: 64
Data:
 ---MCB Type:004D
PSP Adress/Extra data:0040 Memory size: 256
 ---MCB Type:004D
PSP Adress/Extra data:0192 Memory size: 144
Data:
 ---MCB Type:004D
PSP Adress/Extra data:0192 Memory size: 2576
Data:LAB5
  --MCB Type:004D
PSP Adress/Extra data:023E Memory size: 144
Data:
 ---MCB Type:005A
PSP Adress/Extra data:023E Memory size: 646160
Data:LB3
```

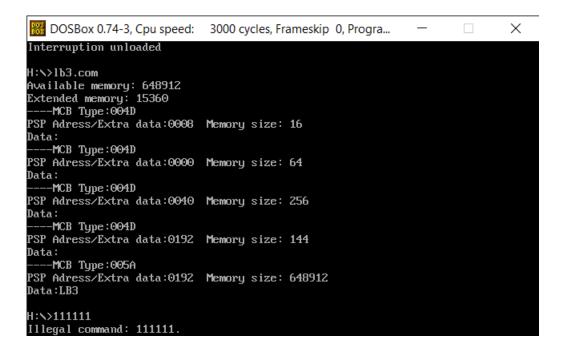
Состояние памяти после загрузки резидентной части



## Определение уже установленного прерывания



Выгрузка прерывания и состояние памяти после выгрузки.



Отсутствие обработки клавиши 1