

**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 обработчика**  
**прерываний**

Студент гр. 9381

\_\_\_\_\_

Гурин С.Н.

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 и 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.

## **Порядок выполнения работы.**

Шаг 1. Для выполнения лабораторной работы необходимо написать и отладить программный модуль типа .EXE, 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 установлено то, устанавливает резидентную функцию для обработки 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. Выгрузка прерывания состоит в восстановлении стандартного вектора прерываний и освобождении памяти, занимаемой резидентом. Затем 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.

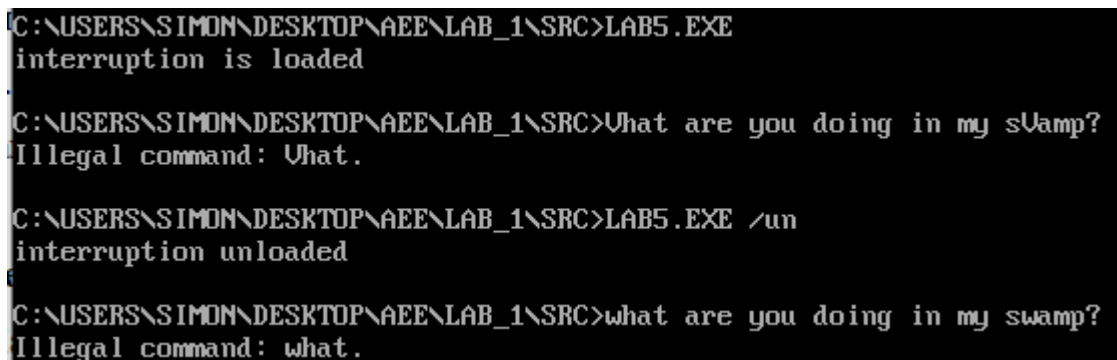
Шаг 4. Запустите отлаженную программу еще раз и убедитесь, что программа определяет установленный обработчик прерываний. Полученные результаты поместите в отчет.

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.**

В данном .EXE модуле было реализовано пользовательское прерывание, которое заменяет символ “w” на “V”. Далее было продемонстрирована работа модуля.



```
C:\USERS\SIMON\DESKTOP\AEE\LAB_1\SRC>LAB5.EXE
interruption is loaded

C:\USERS\SIMON\DESKTOP\AEE\LAB_1\SRC>What are you doing in my sUamp?
Illegal command: What.

C:\USERS\SIMON\DESKTOP\AEE\LAB_1\SRC>LAB5.EXE /un
interruption unloaded

C:\USERS\SIMON\DESKTOP\AEE\LAB_1\SRC>what are you doing in my swamp?
Illegal command: what.
```

Рис. 1

Далее происходит проверка памяти после загрузки прерывания, с помощью ранее реализованного модуля. Далее был продемонстрирован результат.

```

C:\USERS\SIMON\DESKTOP\AEE\LAB_1\SRC>LAB5.EXE
interruption is loaded

C:\USERS\SIMON\DESKTOP\AEE\LAB_1\SRC>LAB3_1.COM
available memory size: 643728 bytes
extended memory size: 246720 bytes
MCB #1: address: 016F PSP address: 0008 size: 16 SC/SD:
MCB #2: address: 0171 PSP address: 0000 size: 64 SC/SD:
MCB #3: address: 0176 PSP address: 0040 size: 256 SC/SD:
MCB #4: address: 0187 PSP address: 0192 size: 144 SC/SD:
MCB #5: address: 0191 PSP address: 0192 size: 5008 SC/SD: LAB5
MCB #6: address: 02CB PSP address: 02D6 size: 1448 SC/SD:
MCB #7: address: 02D5 PSP address: 02D6 size: 643728 SC/SD: LAB3_1

```

Рис. 2

Так же выгрузим это прерывание и проверим состояние памяти.

```

C:\USERS\SIMON\DESKTOP\AEE\LAB_1\SRC>LAB5.EXE /un
interruption unloaded

C:\USERS\SIMON\DESKTOP\AEE\LAB_1\SRC>LAB3_1.COM
available memory size: 648912 bytes
extended memory size: 246720 bytes
MCB #1: address: 016F PSP address: 0008 size: 16 SC/SD:
MCB #2: address: 0171 PSP address: 0000 size: 64 SC/SD:
MCB #3: address: 0176 PSP address: 0040 size: 256 SC/SD:
MCB #4: address: 0187 PSP address: 0192 size: 144 SC/SD:
MCB #5: address: 0191 PSP address: 0192 size: 648912 SC/SD: LAB3_1

```

Рис. 3

Прерывание корректно загружается и выгружается.

### Ответы на контрольные вопросы

1) Какого типа прерывания используются в работе?

1. 21h – прерывания ф-ций DOS
2. 16h, 09h – прерывания ф-ций BIOS

2) Чем отличается скан код от кода ASCII?

Код ASCII – это код символа из таблицы ASCII

Скан-код – код, присвоенный каждой клавише с помощью которого драйвер распознает, какая клавиша была нажата.

### Вывод

В ходе выполнения лабораторной работы была написана программа пользовательского обработчика прерывания от клавиатуры, размещающегося в резидентной памяти.

## ПРИЛОЖЕНИЕ А

### ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: lab5.asm

```
CODE                                SEGMENT
                                   ASSUME    CS:CODE, DS:DATA, SS:SSTACK

SSTACK                             SEGMENT STACK
    DW  256 DUP(0)
SSTACK  ENDS

DATA                                SEGMENT
STR_LOAD        DB  "INTERRUPTION IS LOADED",0DH,0AH,"$"
STR_LOADED      DB  "INTERRUPTION ALREADY LOADED",0DH,0AH,"$"
STR_UNLOAD      DB  "INTERRUPTION UNLOADED",0DH,0AH,"$"
STR_NOT_LOADED  DB  "INTERRUPTION ISN'T LOADED",0DH,0AH,"$"
IS_LOAD         DB  0
IS_UN           DB  0
DATA            ENDS

INTERRUPTION    PROC FAR
                JMP  START

                INTERRUPTIONDATA:
KEY_VALUE       DB  0
SIGNATURE       DW  6666H
KEEP_IP         DW  0
KEEP_CS         DW  0
KEEP_PSP        DW  0
KEEP_AX         DW  0
KEEP_SS         DW  0
KEEP_SP         DW  0
NEW_STACK       DW  256 DUP(0)

                START:
                MOV  KEEP_AX, AX
                MOV  KEEP_SP, SP
                MOV  KEEP_SS, SS
                MOV  AX, SEG NEW_STACK
```

```

MOV    SS, AX
MOV    AX, OFFSET NEW_STACK
ADD    AX, 256
MOV    SP, AX

```

```

PUSH   AX
PUSH   BX
PUSH   CX
PUSH   DX
PUSH   SI
PUSH   ES
PUSH   DS
MOV    AX, SEG KEY_VALUE
MOV    DS, AX

```

```

IN      AL, 60H
CMP     AL, 11H
JE      SWAP

```

```

PUSHF
CALL   DWORD PTR CS:KEEP_IP
JMP    END_INT

```

SWAP:

```

MOV    KEY_VALUE, 'V'
IN      AL, 61H
MOV    AH, AL
OR      AL, 80H
OUT     61H, AL
XCHG   AL, AL
OUT     61H, AL
MOV    AL, 20H
OUT     20H, AL

```

PRINT\_KEY:

```

MOV    AH, 05H
MOV    CL, KEY_VALUE
MOV    CH, 00H

```

```

        INT     16H
        OR      AL, AL
        JZ      END_INT
        MOV     AX, 0040H
        MOV     ES, AX
        MOV     AX, ES:[1AH]
        MOV     ES:[1CH], AX
        JMP     PRINT_KEY

END_INT:
        POP     DS
        POP     ES
        POP     SI
        POP     DX
        POP     CX
        POP     BX
        POP     AX

        MOV     SP, KEEP_SP
        MOV     AX, KEEP_SS
        MOV     SS, AX
        MOV     AX, KEEP_AX

        MOV     AL, 20H
        OUT     20H, AL
        IRET

INTERRUPTION     ENDP

_END:

IS_INT_LOAD      PROC
        PUSH    AX
        PUSH    BX
        PUSH    SI

        MOV     AH, 35H
        MOV     AL, 09H
        INT     21H

```

```

MOV     SI, OFFSET SIGNATURE
SUB     SI, OFFSET INTERRUPTION
MOV     AX, ES:[BX + SI]
CMP     AX, SIGNATURE
JNE     END_IS_L
MOV     IS_LOAD, 1

END_IS_L:
POP     SI
POP     BX
POP     AX
RET

IS_INT_LOAD      ENDP

INT_LOAD         PROC

PUSH    AX
PUSH    BX
PUSH    CX
PUSH    DX
PUSH    ES
PUSH    DS

MOV     AH, 35H
MOV     AL, 09H
INT     21H
MOV     KEEP_CS, ES
MOV     KEEP_IP, BX
MOV     AX, SEG INTERRUPTION
MOV     DX, OFFSET INTERRUPTION
MOV     DS, AX
MOV     AH, 25H
MOV     AL, 09H
INT     21H
POP     DS

MOV     DX, OFFSET _END
MOV     CL, 4H
SHR     DX, CL

```



```

                                ADD      DX, 10FH
                                INC      DX
                                XOR      AX, AX
                                MOV      AH, 31H
                                INT      21H

                                POP      ES
                                POP      DX
                                POP      CX
                                POP      BX
                                POP      AX
                                RET

INT_LOAD      ENDP

UNLOAD_INT    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 KEEP_IP
                SUB     SI, OFFSET INTERRUPTION
                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

STI

POP     SI
POP     ES
POP     DS
POP     DX
POP     BX
POP     AX

RET

UNLOAD_INT      ENDP

IS_UNLOAD_      PROC
    PUSH    AX
    PUSH    ES

    MOV     AX, KEEP_PSP
    MOV     ES, AX
    CMP     BYTE PTR ES:[82H], '/'
    JNE     END_UNL
    CMP     BYTE PTR ES:[83H], 'U'
    JNE     END_UNL
    CMP     BYTE PTR ES:[84H], 'N'
    JNE     END_UNL
    MOV     IS_UN, 1

```

```

END_UNL:

```

```

                                POP    ES
                                POP    AX
                                RET
IS_UNLOAD_                     ENDP

MAIN PROC

                                PUSH    DS
                                XOR     AX, AX
                                PUSH    AX
                                MOV     AX, DATA
                                MOV     DS, AX
                                MOV     KEEP_PSP, ES

                                CALL     IS_INT_LOAD
                                CALL     IS_UNLOAD_
                                CMP     IS_UN, 1
                                JE              UNLOAD
                                MOV     AL, IS_LOAD
                                CMP     AL, 1
                                JNE     LOAD
                                MOV     DX, OFFSET STR_LOADED

                                PUSH    AX
                                MOV     AH, 09H
                                INT     21H
                                POP     AX

                                JMP     _END_

LOAD:

                                MOV     DX, OFFSET STR_LOAD

                                PUSH    AX
                                MOV     AH, 09H
                                INT     21H
                                POP     AX

                                CALL     INT_LOAD
                                JMP     _END_

```

```

UNLOAD:
    CMP     IS_LOAD, 1
    JNE     NOT_LOADED
    MOV     DX, OFFSET STR_UNLOAD

    PUSH    AX
    MOV     AH, 09H
    INT     21H
    POP     AX

    CALL    UNLOAD_INT
    JMP     _END_

NOT_LOADED:
    MOV     DX, OFFSET STR_NOT_LOADED

    PUSH    AX
    MOV     AH, 09H
    INT     21H
    POP     AX

_END_:
    XOR     AL, AL
    MOV     AH, 4CH
    INT     21H

MAIN      ENDP
CODE      ENDS
          END    MAIN

```