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 прерываний

Студент гр. 9382 _____ Субботин М.О.

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 **.EXE,** который выполняет следующие функции:

1) Проверяет, установлено ли пользовательское прерывание с вектором 09h.

2) Устанавливает резидентную функцию для обработки прерывания и настраивает вектор прерываний, если прерывание не установлено, и 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 MCB. 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.

Ход выполнения:

Прерывание по нажатию кнопки f2 выводит последовательно символы из сообщения:

```
D:\LAB5>lab5.exe
Interruption just loaded

D:\LAB5>GOOD J_
```

Состояние памяти после загрузки прерывания:

```
PSP address:  0008
Size of peace in bytes: 16
Sequence of chars:
PSP address:  0000
Size of peace in bytes: 64
Sequence of chars:
PSP address:  0040
Size of peace in bytes: 256
Sequence of chars:
PSP address:  0192
Size of peace in bytes: 144
Sequence of chars:
PSP address:  0192
Size of peace in bytes: 816
Sequence of chars: LAB5
PSP address:  01D0
Size of peace in bytes: 144
Sequence of chars:
PSP address:  01D0
Size of peace in bytes: 880
Sequence of chars: LAB3_2
PSP address:  0000
Size of peace in bytes: 647024
Sequence of chars: 9W↕t3í\>
D:\LAB5>_
```

Выгрузка прерывания:

```
D:\LAB5>lab5.exe /un
Interruption just unloaded
```

Состояние памяти после выгрузки прерывания:

```
D:\LAB5>lab3_2.com

Size of accessible memory: 648912  byte
Size of extended memory: 245760 byte
PSP address:  0008
Size of peace in bytes: 16
Sequence of chars:
PSP address:  0000
Size of peace in bytes: 64
Sequence of chars:
PSP address:  0040
Size of peace in bytes: 256
Sequence of chars:
PSP address:  0192
Size of peace in bytes: 144
Sequence of chars:
PSP address:  0192
Size of peace in bytes: 880
Sequence of chars: LAB3_2
PSP address:  0000
Size of peace in bytes: 648016
Sequence of chars:   ⌸ D:\L
```

Как видно, прерывание корректно отрабатывает, загружается и выгружается из памяти.

Ответы на контрольные вопросы.

1. **Какого типа прерывания использовались в работе?**

   В работе использовались прерывания функции DOS (21h) и прерывания функции BIOS.

2. **Чем отличается скан код от кода ASCII?**

   Скан-код – это код, который присвоен конкретной клавише, с помощью этого кода драйвер клавиатуры распознает какая клавиша была нажата.

   ASCII код – это уникальный код для каждого символа.

   Т.е. скан код характеризует клавишу, а код ASCII – символ.

Выводы.

Была исследована возможность встраивания пользовательского обработчика прерываний в стандартный обработчик от клавиатуры.

ПРИЛОЖЕНИЕ А

КОД ПРОГРАММЫ

Lab5.asm:

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

CUSTOM_INTERRUPTION PROC FAR
        JMP START_CUSTOM_INTERRUPTION

        PSP DW ?
        KEEP_IP DW 0
        KEEP_CS DW 0

    INTERRUPTION_INDEX DW 1234H
    INTERRUPTION_STR DB 'GOOD JOB! $'

        KEEP_SS DW ?
          KEEP_SP DW ?
          KEEP_AX DW ?



    INTER_KEY DB 3CH
    INDEX_STR DB 0

          INTERRUPTION_STACK DW 200 DUP (?)
          END_STACK DW ?

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

        MOV AX,CS
        MOV SS,AX
        MOV SP, OFFSET END_STACK

        PUSH BX
        PUSH CX
        PUSH DX
```

```
    IN AL, 60H
    CMP AL, INTER_KEY
    JE DO_JOB
    CALL DWORD PTR CS:KEEP_IP
    JMP ITERATION_END

DO_JOB:
    IN AL, 61H
    MOV AH, AL

    OR AL, 80H
    OUT 61H, AL

    XCHG AH, AL
    OUT 61H, AL

    MOV AL, 20H
    OUT 20H, AL

    XOR BX, BX
    MOV BL, INDEX_STR

WRITE_INTERRUPTION:
    MOV AH, 05H
    MOV CL, INTERRUPTION_STR[BX]
    CMP CL, '$'
    JE STR_END
    MOV CH, 00H
    INT 16H
    OR AL, AL
    JNZ DO_SKIP
    INC BL
    MOV INDEX_STR, BL
    JMP ITERATION_END

DO_SKIP:
    MOV AX, 0C00H
    INT 21H
    JMP WRITE_INTERRUPTION

STR_END:
    MOV INDEX_STR, 0

ITERATION_END:
```

```
    POP DX
    POP CX
    POP BX
    MOV AX, KEEP_SS
    MOV SS, AX
    MOV AX, KEEP_AX
    MOV SP, KEEP_SP

    IRET

INTERRUPTION_ENDED:
CUSTOM_INTERRUPTION ENDP

WRITE_STRING PROC NEAR
    PUSH AX
    MOV AH,09H
    INT 21H
    POP AX
    RET
WRITE_STRING ENDP

LOAD_FLAG PROC NEAR
        PUSH AX

        MOV PSP,ES
        MOV AL,ES:[81H+1]
        CMP AL,'/'
        JNE LOAD_FLAG_END

        MOV AL,ES:[81H+2]
        CMP AL,'U'
        JNE LOAD_FLAG_END

        MOV AL,ES:[81H+3]
        CMP AL,'N'
        JNE LOAD_FLAG_END

        MOV FLAG,1H

LOAD_FLAG_END:
        POP AX
        RET
LOAD_FLAG ENDP
```

```
IS_LOADED PROC NEAR
        PUSH AX
        PUSH SI

    ;BY 35H GETTING INTERRUPTION'S ADDRESS
        MOV AH,35H
        ;1CH -- NUMBER OF INTERRUPTION
        MOV AL,1CH
        INT 21H

        MOV SI, OFFSET INTERRUPTION_INDEX
        SUB SI, OFFSET CUSTOM_INTERRUPTION
        MOV DX,ES:[BX+SI]
        CMP DX, 1234H
        JNE IS_LOADED_END
        MOV FLAG_LOAD,1H

IS_LOADED_END:
        POP SI
        POP AX
        RET
IS_LOADED ENDP


LOAD_INTERRUPTION PROC NEAR
        PUSH AX
        PUSH DX

        ;CHECKING IF INTERRUPTION IS ALREADY LOADED
        CALL IS_LOADED
        CMP FLAG_LOAD,1H
        JE CUSTOM_ALREADY_LOADED
        JMP STARTING_TO_LOAD

CUSTOM_ALREADY_LOADED:
        LEA DX, INTERRUPTION_ALREADY_LOADED_SEQ
        CALL WRITE_STRING
        JMP END_LOADED

STARTING_TO_LOAD:
        MOV AH,35H
          MOV AL,1CH
          INT 21H
```

```
        MOV KEEP_CS,ES
        MOV KEEP_IP,BX

    PUSH DS
    LEA DX, CUSTOM_INTERRUPTION
    MOV AX, SEG CUSTOM_INTERRUPTION
    MOV DS,AX
    MOV AH,25H
    MOV AL,1CH
    INT 21H
    POP DS
    LEA DX, INTERRUPTION_JUST_LOADED_SEQ
    CALL WRITE_STRING

    LEA DX, INTERRUPTION_ENDED
    MOV CL,4H
    SHR DX,CL
    INC DX
    MOV AX,CS
    SUB AX, PSP
    ADD DX,AX
    XOR AX,AX
    MOV AH,31H
    INT 21H

END_LOADED:
    POP DX
    POP AX
    RET
LOAD_INTERRUPTION ENDP


UNLOAD_INTERRUPTION PROC NEAR
    PUSH AX
    PUSH SI

    CALL IS_LOADED
    CMP FLAG_LOAD,1H
    JE START_UNLOAD

    LEA DX, INTERRUPTION_NOT_LOADED_SEQ
    CALL WRITE_STRING
    JMP UNLOAD_END
```

```
START_UNLOAD:
        CLI
        PUSH DS
        MOV AH,35H
          MOV AL,1CH
          INT 21H

        MOV SI, OFFSET KEEP_IP
          SUB SI, OFFSET CUSTOM_INTERRUPTION
          MOV DX,ES:[BX+SI]
          MOV AX,ES:[BX+SI+2]
        MOV DS,AX
        MOV AH,25H
        MOV AL,1CH
        INT 21H
        POP DS
      STI

      LEA DX, INTERRUPTION_UNLOADED_SEQ
      CALL WRITE_STRING

        MOV AX,ES:[BX+SI-2]
        MOV ES,AX
      MOV AX,ES:[2CH]
        PUSH ES


        MOV ES,AX
        MOV AH,49H
        INT 21H

        POP ES
        INT 21H

UNLOAD_END:
        POP SI
        POP AX
        RET
UNLOAD_INTERRUPTION ENDP


MAIN     PROC  FAR
        PUSH  DS
        XOR   AX,AX
```

```
        PUSH  AX
        MOV   AX,DATA
        MOV   DS,AX

        CALL LOAD_FLAG
        CMP FLAG, 1H
        JE IF_UNLOADED
        CALL LOAD_INTERRUPTION
        JMP THE_END

IF_UNLOADED:
        CALL UNLOAD_INTERRUPTION

THE_END:
        MOV AH,4CH
        INT 21H
MAIN      ENDP
CODE      ENDS

ASTACK    SEGMENT  STACK
  DW 200 DUP(?)
ASTACK    ENDS

DATA      SEGMENT

        FLAG_LOAD DB 0
        FLAG DB 0

        INTERRUPTION_JUST_LOADED_SEQ  DB 'INTERRUPTION JUST
LOADED', 0AH, 0DH,'$'
        INTERRUPTION_UNLOADED_SEQ  DB 'INTERRUPTION JUST
UNLOADED', 0AH, 0DH,'$'
        INTERRUPTION_NOT_LOADED_SEQ  DB 'INTERRUPTION ISNT
LOADED', 0AH, 0DH,'$'
        INTERRUPTION_ALREADY_LOADED_SEQ  DB 'INTERRUPTION IS
ALREADY LOADED', 0AH, 0DH,'$'

DATA      ENDS
        END MAIN
```