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 f1 выводит символ “+”:

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

D:\LAB5>+++asd+++
```

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: 944
Sequence of chars: LAB5
PSP address:  01D8
Size of peace in bytes: 144
Sequence of chars:
PSP address:  01D8
Size of peace in bytes: 880
Sequence of chars: LAB3_2
PSP address:  0000
Size of peace in bytes: 646896
Sequence of chars:
D:\LAB5>_
```

Выгрузка прерывания:

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

D:\LAB5>
```

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:  δΓ]▼^Z
```

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 SS:ASTACK, DS:DATA, CS:CODE

CUSTOM_INTERRUPTION PROC FAR
        JMP START_CUSTOM_INTERRUPTION
INTERRUPTION_DATA:
        KEEP_IP DW 0
        KEEP_CS DW 0
    PSP DW 0
    KEEP_AX DW 0
    KEEP_SS DW 0
    KEEP_SP DW 0
    INTERRUPTION_INDEX DW 1234H
    INTER_KEY DB 3BH
         INTERRUPTION_STACK DW 100H DUP (?)


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

        MOV AX, SEG INTERRUPTION_STACK
        MOV SS,AX
        MOV SP, OFFSET INTERRUPTION_STACK
    ADD SP, 200H

    PUSH AX
        PUSH BX
        PUSH CX
        PUSH DX
    PUSH SI
    PUSH DS
    PUSH BP
    PUSH ES


    MOV AX, SEG INTERRUPTION_DATA
```

```
    MOV DS, AX

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

DO_JOB:
    PUSH AX
    IN AL, 61H
    MOV AH, AL

    OR AL, 80H
    OUT 61H, AL

    XCHG AH, AL
    OUT 61H, AL

    MOV AL, 20H
    OUT 20H, AL

    POP AX

WRITE_INTERRUPTION:
    MOV AH, 05H
    MOV CL, '+'
    XOR CH, CH
    INT 16H
    OR AL,AL
    JNZ DO_SKIP
    JMP STR_END

DO_SKIP:
    MOV AX, 0040H
    MOV ES,AX
    MOV AX, ES:[1AH]
    MOV ES:[1CH], AX
    JMP WRITE_INTERRUPTION

STR_END:
    POP ES
    POP BP
```

```
    POP DS
    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_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
    PUSH ES

    MOV AX, PSP
    MOV ES, AX
    CMP BYTE PTR ES:[82H], '/'
    JNE LOAD_FLAG_END
    CMP BYTE PTR ES:[83H], 'U'
    JNE LOAD_FLAG_END
    CMP BYTE PTR ES:[84H], 'N'
    JNE LOAD_FLAG_END
    MOV FLAG, 1

LOAD_FLAG_END:
    POP ES
      POP AX
      RET
```

```
LOAD_FLAG ENDP


IS_LOADED PROC NEAR
        PUSH AX
        PUSH SI

     ;BY 35H GETTING INTERRUPTION'S ADDRESS
        MOV AH,35H
        ;09H -- NUMBER OF INTERRUPTION
        MOV AL,09H
        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, 1

IS_LOADED_END:
        POP SI
        POP AX
        RET
IS_LOADED ENDP


LOAD_INTERRUPTION PROC NEAR
        PUSH AX
     PUSH BX
     PUSH ES
        PUSH DX
     PUSH CX

        MOV AH,35H
          MOV AL,09H
          INT 21H
          MOV KEEP_CS,ES
          MOV KEEP_IP,BX

        PUSH DS
        MOV DX, OFFSET CUSTOM_INTERRUPTION
        MOV AX, SEG CUSTOM_INTERRUPTION
        MOV DS, AX
```

```
        MOV AH, 25H
        MOV AL, 09H
        INT 21H
        POP DS

        MOV DX, OFFSET INTERRUPTION_ENDED
      ADD DX, 10FH
        MOV CL, 4
        SHR DX, CL
        INC DX
      XOR AX, AX
      MOV AH, 31H
      INT 21H

      POP CX
        POP DX
      POP ES
      POP BX
        POP AX
        RET
LOAD_INTERRUPTION ENDP


UNLOAD_INTERRUPTION PROC
      CLI
        PUSH AX
      PUSH BX
      PUSH DX
      PUSH ES
        PUSH SI

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

        MOV SI, OFFSET KEEP_IP
          SUB SI, OFFSET CUSTOM_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 ES,ES:[BX+SI+4]
    PUSH ES
        MOV ES, ES:[2CH]
    MOV AH, 49H
    INT 21H
    POP ES
    MOV AH, 49H
    INT 21H

        POP SI
    POP ES
    POP DX
    POP BX
        POP AX
    STI
        RET
UNLOAD_INTERRUPTION ENDP


MAIN    PROC
    MOV AX, DATA
    MOV DS, AX
    MOV PSP, ES
    CALL IS_LOADED
    CALL LOAD_FLAG
    CMP FLAG, 1
    JE UNLOAD
    CMP FLAG_LOAD, 0
    JE LOAD
    LEA DX, INTERRUPTION_ALREADY_LOADED_SEQ
    CALL WRITE_STRING
    JMP END_MAIN
UNLOAD:
    CMP FLAG_LOAD, 0
    JE NOT_EXIST
    CALL UNLOAD_INTERRUPTION
    LEA DX, INTERRUPTION_UNLOADED_SEQ
    CALL WRITE_STRING
    JMP END_MAIN

NOT_EXIST:
```

```
    LEA DX, INTERRUPTION_NOT_LOADED_SEQ
    CALL WRITE_STRING
    CALL END_MAIN

LOAD:
    LEA DX, INTERRUPTION_JUST_LOADED_SEQ
    CALL WRITE_STRING
    CALL LOAD_INTERRUPTION
END_MAIN:
    XOR AL, AL
    MOV AH, 4CH
    INT 21H
MAIN    ENDP
CODE    ENDS

ASTACK   SEGMENT  STACK
  DW 100 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
```