МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МОЭВМ

ОТЧЕТ

по лабораторной работе №4
по дисциплине «Операционные системы»
Тема: «Сопряжение стандартного и пользовательского обработчиков прерываний»

Студент гр. 7381 _____ Вологдин М.Д.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург
2019

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.

Описание функций:

ROUT	Процедура пользовательского прерывания, которая при нажатии клавиш alt+w выводит D
PRINT	Процедура вызова прерывания, печатающего строку
CHECK_INT	Процедура, устанавливающая пользовательский обработчик прерывания, если сигнатуры не совпали, и печатающая сообщение о том, что прерывание уже установлено, если сигнатуры совпали. Помещает в SHOULD_BE_DELETED функции прерывания 1, если программа была запущена с ключом /un и сигнатуры функций прерываний совпадали
SET_INT	Процедура установки пользовательского прерывания
DELETE_INT	Процедура удаления пользовательского прерывания

Описание структур данных:

STR_INT_IS_ALR_LOADED	Строка, информирующая, что пользовательское прерывание уже установлено
STR_INT_IS_UNLOADED	Строка, информирующая, что пользовательское прерывание было успешно выгружено
STR_INT_IS_LOADED	Строка, информирующая, что пользовательское прерывание было успешно загружено
STRENDL	Строка, переводящая каретку на начало новой строки

Выполнение работы.

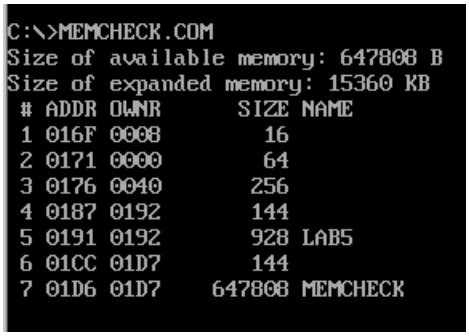
Был написан программный модуль .EXE, который выполняет следующие функции:

1) Проверяет, установлен ли пользовательский обработчик прерывания с вектором 09h.
2) Если не установлен, то устанавливается резидентная функция для обработки прерывания и осуществляется выход в DOS с помощью функции 31h прерывания 21h, оставляющей программу прерывания резидентной.
3) Если установлен, то выводится соответствующее сообщение и осуществляется выход в DOS.
4) Если установлен и программа запущена с ключом /un, то обработчик прерывания выгружается из памяти, восстанавливается стандартный вектор прерывания и осуществляется выход в DOS.

Тестирование

1) Загрузка прерывания.

2) Память после загрузки прерывания.

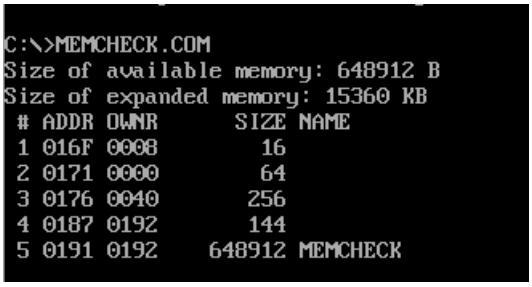


3) Попытка загрузки второго прерывания.

4) Проверка работы прерывания при нажатии различных клавиш. При нажатии Alt+w выводиться знак «#»

5) Выгрузка прерывания

6) Память после выгрузки прерывания



Выводы.

В результате выполнения данной лабораторной работы была исследована возможность встраивания пользовательского обработчика прерываний в стандартный обработчик от клавиатуры.

Ответы на контрольные вопросы

1. Какого типа прерывания использовались в работе?

Ответ: В работе использовались аппаратные (прерывания от контроллера клавиатуры при нажатии клавиш) и программные прерывания.

2. Чем отличается скан-код от кода ASCII?

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

ASCII код – код для представления символа в виде числа из стандартной кодировочной таблицы.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД

```
ISTACK SEGMENT STACK
     DW 100H DUP (?)
ISTACK ENDS

CODE SEGMENT
 ASSUME CS:CODE, DS:DATA, ES:DATA, SS:STACK
START: JMP BEGIN
;--------------------------------------
ROUT PROC FAR
     JMP INT_CODE
     SIGNATURE DB 'AAAB'
     KEEP_IP DW 0
     KEEP_CS DW 0
     KEEP_PSP DW 0
     KEEP_SS DW 0
     KEEP_SP DW 0
     KEEP_AX DW 0
     INT_CODE:

     MOV CS:KEEP_AX, AX
     MOV CS:KEEP_SS, SS
     MOV CS:KEEP_SP, SP
     MOV AX, ISTACK
     MOV SS, AX
     MOV SP, 100H
     PUSH DX
     PUSH DS
     PUSH ES

          IN AL,60H
          CMP AL,11H
          JNE ROUT_STNDRD
```

```
      MOV AX,0040H
      MOV ES,AX
      MOV AL,ES:[18H]
      AND AL,00000010B
      JZ ROUT_STNDRD
JMP ROUT_USER


ROUT_STNDRD:
      POP ES
      POP DS
      POP DX
      MOV AX, CS:KEEP_AX
      MOV SP, CS:KEEP_SP
      MOV SS, CS:KEEP_SS
      JMP DWORD PTR CS:KEEP_IP

ROUT_USER:

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

ROUT_PUSH_TO_BUFF:
      MOV AH,05H
      MOV CL,'#'
```

```
            MOV CH,00H
            INT 16H
            OR AL,AL
            JZ ROUT_END

                  CLI
                  MOV AX,ES:[1AH]
                  MOV ES:[1CH],AX
                  STI
                  JMP ROUT_PUSH_TO_BUFF

      ROUT_END:
      POP ES
      POP DS
      POP DX
      MOV AX, CS:KEEP_AX
      MOV AL,20H
      OUT 20H,AL
      MOV SP, CS:KEEP_SP
      MOV SS, CS:KEEP_SS
      IRET
ROUT ENDP
      LAST_BYTE:
;---------------------------------------
PRINT PROC NEAR
      PUSH AX
      MOV AH,09H
      INT 21H
      POP AX
      RET
PRINT ENDP
;---------------------------------------
;
CHECK_INT PROC
```

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

MOV SI, OFFSET SIGNATURE
SUB SI, OFFSET ROUT

      MOV AX,'AA'
      CMP AX,ES:[BX+SI]
      JNE LABEL_INT_IS_NOT_LOADED
      MOV AX,'BA'
      CMP AX,ES:[BX+SI+2]
      JNE LABEL_INT_IS_NOT_LOADED
      JMP LABEL_INT_IS_LOADED

LABEL_INT_IS_NOT_LOADED:
      MOV DX,OFFSET STR_INT_IS_LOADED
      CALL PRINT
      CALL SET_INT

            MOV DX,OFFSET LAST_BYTE
            MOV CL,4
            SHR DX,CL
            INC DX
            ADD DX,CODE
            SUB DX,CS:KEEP_PSP
      XOR AL,AL
      MOV AH,31H
      INT 21H

LABEL_INT_IS_LOADED:
      PUSH ES
      PUSH BX
      MOV BX,KEEP_PSP
```

```
            MOV ES,BX
            CMP BYTE PTR ES:[82H],'/'
            JNE CI_DONT_DELETE
            CMP BYTE PTR ES:[83H],'U'
            JNE CI_DONT_DELETE
            CMP BYTE PTR ES:[84H],'N'
            JE CI_DELETE
            CI_DONT_DELETE:
            POP BX
            POP ES

      MOV DX,OFFSET STR_INT_IS_ALR_LOADED
      CALL PRINT
      RET

            CI_DELETE:
            POP BX
            POP ES

            CALL DELETE_INT
            MOV DX,OFFSET STR_INT_IS_UNLOADED
            CALL PRINT
            RET
CHECK_INT ENDP
;---------------------------------------

SET_INT PROC
      PUSH DS
      MOV AH,35H
      MOV AL,09H
      INT 21H
      MOV CS:KEEP_IP,BX
      MOV CS:KEEP_CS,ES
```

```
      MOV DX,OFFSET ROUT
      MOV AX,SEG ROUT
      MOV DS,AX
      MOV AH,25H
      MOV AL,09H
      INT 21H
      POP DS
      RET
SET_INT ENDP
;---------------------------------------
DELETE_INT PROC
      PUSH DS
            CLI
            MOV DX,ES:[BX+SI+4] ; IP
            MOV AX,ES:[BX+SI+6] ; CS
            MOV DS,AX

            MOV AX,2509H
            INT 21H

            PUSH ES
            MOV AX,ES:[BX+SI+8]
            MOV ES,AX
            MOV ES,ES:[2CH]
            MOV AH,49H
            INT 21H
            POP ES
            MOV ES,ES:[BX+SI+8]
            MOV AH, 49H
            INT 21H
            STI
      POP DS
      RET
DELETE_INT ENDP
```

```
;---------------------------------------
BEGIN:
     MOV AX,DATA
     MOV DS,AX
     MOV CS:KEEP_PSP,ES

     CALL CHECK_INT

     XOR AL,AL
     MOV AH,4CH
     INT 21H
CODE ENDS

DATA SEGMENT
     STR_INT_IS_ALR_LOADED  DB  'USER  INTERRUPTION  IS  ALREADY
LOADED',0DH,0AH,'$'
     STR_INT_IS_UNLOADED DB 'USER INTERRUPTION IS SUCCESSFULLY
UNLOADED',0DH,0AH,'$'
     STR_INT_IS_LOADED     DB     'USER     INTERRUPTION     IS
LOADED',0DH,0AH,'$'
     STRENDL DB 0DH,0AH,'$'
DATA ENDS

STACK SEGMENT STACK
     DW 50 DUP (?)
STACK ENDS
 END START
```