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: «Обработка стандартных прерываний»

Студент гр. 7381 _____ Вологдин М.Д.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург
2019

Цель работы.

В архитектуре компьютера существуют стандартные прерывания, за которыми закреплены определённые вектора прерываний. Вектор прерываний хранит адрес подпрограммы обработчика прерываний. При возникновении прерывания, аппаратура компьютера передаёт управление и выполняет соответствующие действия.

В данной лабораторной работе предлагается построить обработчик прерываний сигналов таймера. Эти сигналы генерируются аппаратурой через определённые интервалы времени и, при возникновении такого сигнала, возникает прерывание с определённым значением вектора. Таким образом, управление будет передано функции, чья точка входа записана в соответствующий вектор прерывания.

Описание функций:

PRINT	Вызывает прерывание печати строки
setCurs	Устанавливает курсор в строку dh и столбец dl
getCurs	Возвращает положение курсора: строка в dh, столбец в dl
outputAL	Вывод символа с кодом AL в текущее положение курсора
ROUT	Пользовательский обработчик прерывания, печатающий числа. При помещении в SHOULD_BE_DELETED единицы восстанавливает стандартное прерывание и выгружается из памяти
CHECK_INT	Функция, устанавливающее пользовательский обработчик прерывания, если сигнатуры не совпали, и печатающее 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	Функция установки пользовательской функции прерывания
DEL_INT	Функция 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, установлено ли пользовательское прерывание с вектором 1Ch с помощью сигнатуры ‘AAAA’.
2) Если не установлено, то устанавливает резидентная функция для обработки прерывания и осуществляет выход в DOS с помощью функции 31h прерывания 21h, оставляющей программу прерывания резидентной.
3) Если установлено, то выводит соответствующее сообщение и осуществляет выход в DOS.
4) Если установлено и программа запущена с ключом /un, то выгружает прерывание из памяти и осуществляет выход в DOS.

Тестирование

1) Загрузка прерывания.

```
C:\>LAB4.EXE
User interruption is loaded
```

2) Память после загрузки прерывания.

```
C:\>memcheck.com
Size of available memory: 647792 B
Size of expanded memory: 15360 KB
 # ADDR OWNR      SIZE NAME
 1 016F 0008        16
 2 0171 0000        64
 3 0176 0040       256
 4 0187 0192       144
 5 0191 0192       944 LAB4
 6 01CD 01D8       144
 7 01D7 01D8    647792 MEMCHECK
```

3) Попытка загрузки второго прерывания.

```
C:\>lab4.exe
User interruption is already loaded
```

4) Выгрузка прерывания

```
C:\>lab4.exe /un
User interruption is successfully unloaded
```

5) Память после выгрузки прерывания

```
C:\>memcheck.com
Size of available memory: 648912 B
Size of expanded memory: 15360 KB
 # ADDR OWNR      SIZE NAME
 1 016F 0008        16
 2 0171 0000        64
 3 0176 0040       256
 4 0187 0192       144
 5 0191 0192    648912 MEMCHECK
```

Выводы.

В результате выполнения данной лабораторной работы были исследованы организация и управление прерываниями, был построен обработчик прерывания от сигналов таймера.

Ответы на контрольные вопросы

1. Как реализован механизм прерывания от часов?

Ответ: Прерывание по таймеру вызывается автоматически по каждому тику аппаратных часов – каждые 55мс. После вызова, сохраняется содержимое регистров, затем определяется источник прерывания, по номеру которого определяется смещение в таблице векторов прерываний. Полученный адрес сохраняется в регистры CS:IP, после чего управление передается по этому адресу, т.е. выполняется запуск обработчика прерываний и происходит его выполнение. После выполнения, происходит возврат управления прерванной программе.

2. Какого типа прерывания использовались в работе?

Ответ: В работе использовались аппаратные (прерывания от системного таймера), и программные прерывания.

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
;---------------------------------------
PRINT PROC NEAR
     PUSH AX
     MOV AL,00H
     MOV AH,09H
     INT 21H
     POP AX
     RET
PRINT ENDP

;---------------------------------------
SETCURS PROC
     PUSH AX
     PUSH BX
     PUSH DX
     PUSH CX
     MOV AH,02H
     MOV BH,0
     INT 10H
     POP CX
     POP DX
     POP BX
     POP AX
     RET
SETCURS ENDP
```

```
;---------------------------------------
GETCURS PROC
      PUSH AX
      PUSH BX
      ;PUSH DX
      PUSH CX
      MOV AH,03H
      MOV BH,0
      INT 10H
      POP CX
      ;POP DX
      POP BX
      POP AX
      RET
GETCURS ENDP

;---------------------------------------
OUTPUTAL PROC
      PUSH AX
      PUSH BX
      PUSH CX
      MOV AH,09H
      MOV BH,0
      MOV BL,07H
      MOV CX,1
      INT 10H
      POP CX
      POP BX
      POP AX
      RET
OUTPUTAL ENDP
;---------------------------------------
ROUT PROC FAR
      JMP INT_CODE
```

```
SIGNATURE DB 'AAAA'
KEEP_IP DW 0
KEEP_CS DW 0
KEEP_PSP DW 0
SHOULD_BE_DELETED DB 0
COUNT DB 0
KEEP_SS DW 0
KEEP_SP DW 0
KEEP_AX DW 0
INT_CODE:

MOV 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

CALL GETCURS
PUSH DX
MOV DX,0013H
CALL SETCURS

CMP COUNT,0AH
JL ROUT_SKIP
MOV COUNT,0H
ROUT_SKIP:
MOV AL,COUNT
OR AL,30H
CALL OUTPUTAL
```

```
      POP DX
      CALL SETCURS
      INC COUNT

      POP ES
      POP DS
      POP DX
      MOV AX, KEEP_AX
      MOV AL,20H
      OUT 20H,AL
      MOV SP, KEEP_SP
      MOV SS, KEEP_SS

      IRET
ROUT ENDP
LAST_BYTE:
;---------------------------------------
;
CHECK_INT PROC
           MOV AH,35H
           MOV AL,1CH
           INT 21H

      MOV SI, OFFSET SIGNATURE
      SUB SI, OFFSET ROUT

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

      LABEL_INT_IS_NOT_LOADED:
```

```
        LEA DX, 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,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 DEL_INT
            MOV DX,OFFSET STR_INT_IS_UNLOADED
            CALL PRINT
            RET
CHECK_INT ENDP
;---------------------------------------
DEL_INT PROC
            PUSH DS
            CLI
            MOV DX,ES:[BX+SI+4] ; IP
            MOV AX,ES:[BX+SI+6] ; CS
            MOV DS,AX
            MOV AX,251CH
            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
DEL_INT ENDP
;---------------------------------------
SET_INT PROC
      PUSH DS
      MOV AH,35H
```

```
      MOV AL,1CH
      INT 21H
      MOV KEEP_IP,BX
      MOV KEEP_CS,ES

      MOV DX,OFFSET ROUT
      MOV AX,SEG ROUT
      MOV DS,AX
      MOV AH,25H
      MOV AL,1CH
      INT 21H
      POP DS
      RET
SET_INT ENDP
;--------------------------------------
BEGIN:
      MOV AX,DATA
      MOV DS,AX
      MOV KEEP_PSP,ES

      CALL CHECK_INT

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

CODE ENDS

STACK SEGMENT STACK
      DW 100H DUP (?)
STACK 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
      END START
```