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

ОТЧЕТ

по лабораторной работе №4
по дисциплине «Операционные системы»
Тема: Обработка стандартных прерываний

Студент гр. 8383 _____ Бессуднов Г. И.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург
2020

Цель работы.

Построить обработчик прерываний сигнала таймера. Изучить способы загрузки резидентной программы в память и ее выгрузку

Ход работы.

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

1) Проверяет, установлено ли пользовательское прерывание с вектором 1Ch.
2) Устанавливает резидентную функцию для обработки прерывания и настраивает вектор прерываний, если прерывание не установлено, и осуществляет выход по функции 4Ch прерывания int 21h.
3) Если прерывание установлено, то выводится соответствующее сообщение и осуществляется выход по функции 4Ch прерывания int 21h.
4) 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.

Код пользовательского прерывания должен выполнять следующие функции:

1) Сохранять значения регистров в стеке при входе и восстановить их при выходе.
2) При выполнении тела процедуры накапливать общее суммарное число прерываний и выводить на экран. Для вывода на экран следует использовать прерывание int 10h, которое позволяет непосредственно выводить информацию на экран.

Вывод программы после первого запуска представлен на рис. 1.

Рисунок 1 – Выполнение LR4.EXE в первый раз

Вывод программы LR3_1.COM из лабораторной №3 представлен на рис. 2.

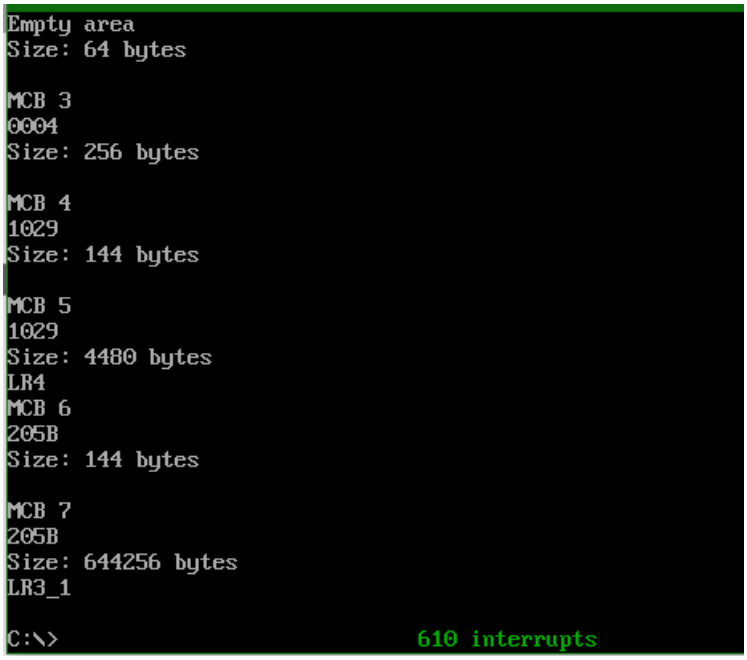

Рисунок 2 – Выполнение LR3_1.COM

Попытка еще раз запустить LR4.EXE представлена на рис. 3.

Рисунок 3 – Запуск lr4.exe во второй раз

Далее программа была запущена с параметром /un дважды, результат представлен на рис. 4.

Рисунок 4 – Запуск lr4.exe с параметром /un дважды

С помощью программы LR3_1.COM была просмотрена информация о блоках MCB, результат выполнения представлен на рис. 5.

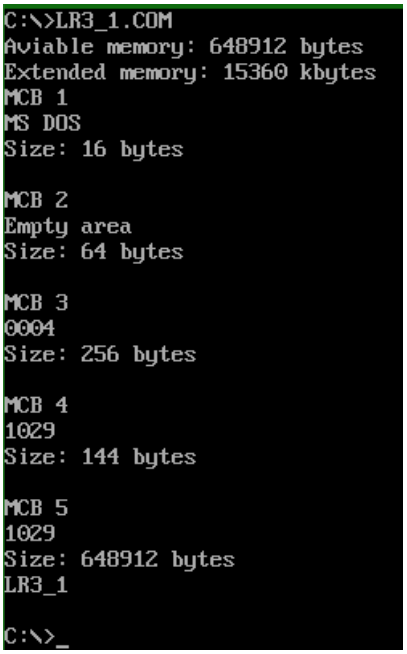

Рисунок 5 – Выполнение LR3_1.COM после выгрузки

На рисунке 5 действительно видно, что память резидентного обработчика была освобождена.

Контрольные вопросы.

1. Как реализован механизм прерывания от часов?

Системный таймер вырабатывает прерывание 8H каждый раз в $\frac{1}{18,2}$ секунды. Обработчик, установленный BIOS при инициализации, каждый раз увеличивает значение тиков таймера на 1. Так же обработчик может вызывать прерывание 1CH. При инициализации 1CH указывает на IRET, т.е. по умолчанию нечего выполняться не будет. В программе же можно вручную установить обработчик этого прерывания, который будет выполнять какие-то действия периодически.

2. Какого типа прерывания использовались в работе?

1) Программные - 21H, 10H

2) Аппаратные - 1CH

Выводы.

В ходе выполнения лабораторной работы была реализована программа, загружающая и выгружающая пользовательское прерывание от системного таймера в память.

ПРИЛОЖЕНИЕ
КОД ПРОГРАММЫ

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

ASTACK  SEGMENT STACK
    DW  128 dup(0)
ASTACK  ENDS

DATA    SEGMENT
       MES_LOADED DB  "Loaded already",10,13,"$"
       MES_NOT_LOADED     DB  "Not loaded",10,13,"$"
    IS_NOT_LOADED      DB  1
    IS_UN              DB  0
DATA    ENDS

;прерывание
MY_INT   PROC    FAR
        jmp     MY_INT_START
       MY_INT_DATA:
             INT_COUNTER        DB  "000 interrupts"
             ID                 DW  6506h
             KEEP_IP             DW  0
             KEEP_CS             DW  0
             KEEP_PSP            DW  0

    MY_INT_START:
             push  AX
             push    BX
             push    CX
             push    DX
             push    SI
        push    ES
        push    DS

             mov    AX, seg INT_COUNTER
             mov    DS, AX

    ;поставить курсор
        mov     AH, 03h
             mov     BH, 0h
             int     10h
        push    DX

        mov     AH, 02h
             mov     BH, 0h
             mov     DX, 1820h
             int     10h

      ;увеличить счетчик
             mov    AX, SEG INT_COUNTER
             push  DS
             mov    DS, AX
             mov    SI, offset INT_COUNTER
             add          SI, 2
             mov    CX, 3
      MY_INT_CYCLE:
```

```
            mov    AH, [SI]
            inc    AH
            mov    [SI], AH
            cmp    AH, ':'
            jne    MY_INT_END_CYCLE
            mov    AH, '0'
            mov    [SI], AH
            dec    SI
            loop   MY_INT_CYCLE
      MY_INT_END_CYCLE:
            pop    DS

      ;печать счетчика
            push   ES
            push   BP
       mov     AX, SEG INT_COUNTER
            mov     ES, AX
            mov     BP, offset INT_COUNTER
            mov     AH, 13h
            mov     AL, 1h
            mov    BL, 2h
            mov     BH, 0
            mov     CX, 14
            int     10h

            pop          BP
            pop          ES

        pop     DX
        mov     AH, 02h
            mov     BH, 0h
            int     10h

            pop     DS
            pop     ES
            pop          SI
            pop     DX
            pop     CX
            pop     BX
            pop          AX

            mov     AL, 20h
            out     20h, AL
      iret
MY_INT    ENDP
    MY_INT_END:

;проверка прерывания на загруженность
MY_INT_CHECK       PROC
            push    AX
            push    BX
            push    SI

            mov     AH, 35h
            mov     AL, 1Ch
            int     21h
            mov     SI, offset ID
            sub     SI, offset MY_INT
            mov     AX, ES:[BX + SI]
            cmp       AX, ID
            jne     MY_INT_CHECK_END
```

```
            mov     IS_NOT_LOADED, 0

      MY_INT_CHECK_END:
            pop     SI
            pop     BX
            pop     AX
      ret
MY_INT_CHECK       ENDP
;загрузка прерывания
MY_INT_LOAD        PROC
        push    AX
            push    BX
            push    CX
            push    DX
            push    ES
            push    DS

        mov     AH, 35h
            mov     AL, 1Ch
            int     21h
            mov     KEEP_CS, ES
        mov     KEEP_IP, BX
        mov     AX, seg MY_INT
            mov     DX, offset MY_INT
            mov     DS, AX
            mov     AH, 25h
            mov     AL, 1Ch
            int     21h
            pop          DS

        mov     DX, offset MY_INT_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
MY_INT_LOAD        ENDP
;выгрузка прерывания
MY_INT_UNLOAD      PROC
        CLI
            push    AX
            push    BX
            push    DX
            push    DS
            push    ES
            push    SI

            mov     AH, 35h
            mov     AL, 1Ch
            int     21h
            mov    SI, offset KEEP_IP
            sub    SI, offset MY_INT
```

```
            mov    DX, ES:[BX + SI]
            mov    AX, ES:[BX + SI + 2]

            push   DS
            mov      DS, AX
            mov      AH, 25h
            mov      AL, 1Ch
            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
MY_INT_UNLOAD      ENDP

;проверяем не был ли передан аргумент /un
UN_CHECK        PROC
        push    AX
            push    ES

            mov      AX, KEEP_PSP
            mov      ES, AX
            cmp      byte ptr ES:[82h], '/'
            jne      UN_CHECK_END
            cmp      byte ptr ES:[83h], 'u'
            jne      UN_CHECK_END
            cmp      byte ptr ES:[84h], 'n'
            jne      UN_CHECK_END
            mov      IS_UN, 1

      UN_CHECK_END:
            pop      ES
            pop      AX
            ret
UN_CHECK        ENDP

WRITE_STR    PROC    NEAR
        push    AX
        mov     AH, 09h
        int     21h
        pop     AX
    ret
WRITE_STR   ENDP
```

```
MAIN PROC
             push    DS
             xor     AX, AX
             push    AX
             mov     AX, DATA
             mov     DS, AX
             mov     KEEP_PSP, ES

             call    MY_INT_CHECK
             call    UN_CHECK
             cmp     IS_UN, 1
             je      UNLOAD
             mov     AL, IS_NOT_LOADED
             cmp     AL, 1
             je      LOAD
             mov     DX, offset MES_LOADED
             call    WRITE_STR
             jmp     MAIN_END
      LOAD:;загружаем перрывание
             call    MY_INT_LOAD
             jmp     MAIN_END
      UNLOAD: ;выгрузка перрывания, если передали /un
             cmp     IS_NOT_LOADED, 1
             je     NOT_EXIST
             call    MY_INT_UNLOAD
             jmp     MAIN_END
      NOT_EXIST:
             mov     DX, offset MES_NOT_LOADED
             call    WRITE_STR
      MAIN_END:
             xor    AL, AL
             mov    AH, 4Ch
             int    21h
      MAIN ENDP
CODE    ENDS
END   MAIN
```