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 _____ Дейнега В.Е.

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 l31.com из лабораторной №3 представлен на рис. 2.

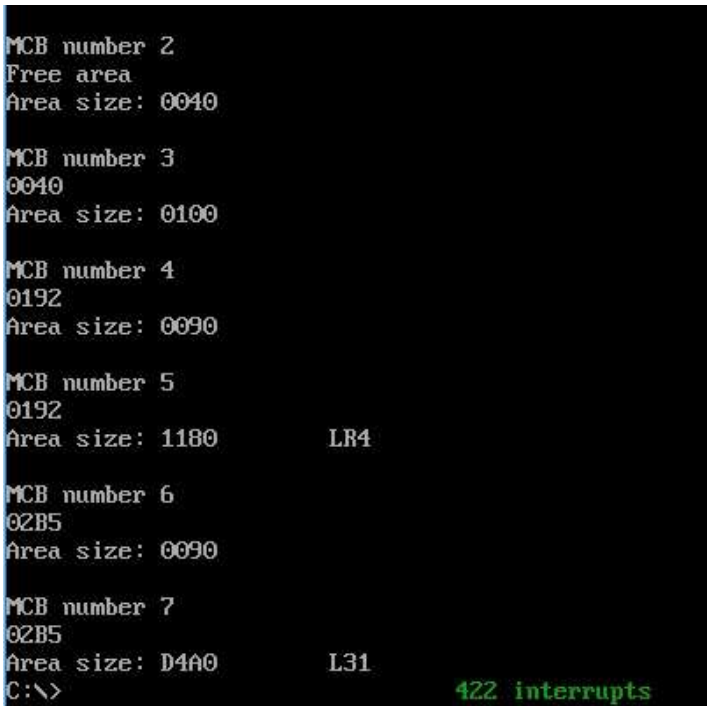

Рисунок 2 – Выполнение l31.com после выполнения lab4.exe

Попытка еще раз запустить lr4.exe представлена на рис. 3.

Рисунок 3 – Запуск lr4.exe во второй раз

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

Рисунок 4 – Запуск lr4.exe с параметром /um

С помощью программы из лабораторной №3 просмотрим информацию о блоках MCB, результат выполнения l31.com представлен на рис. 5.

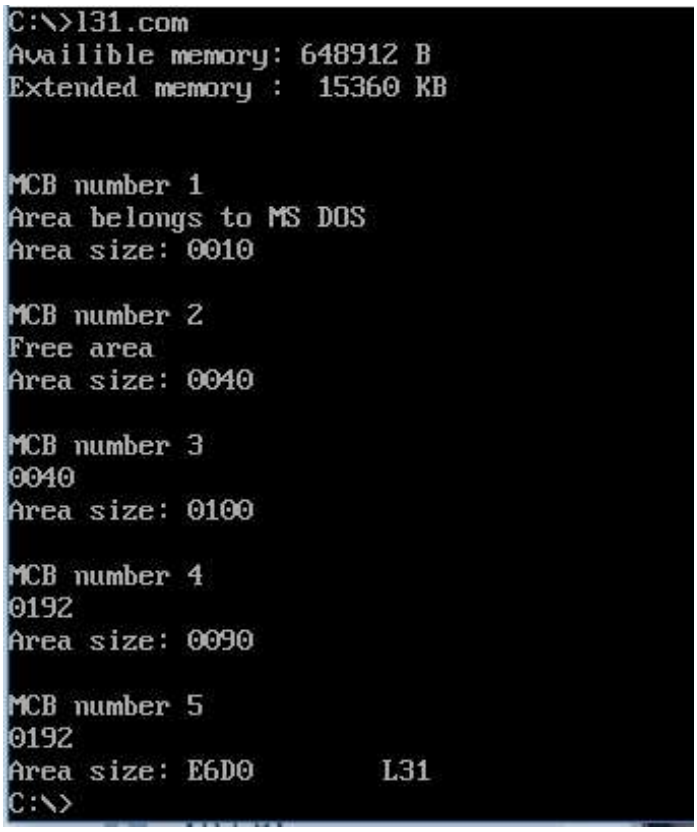

Рисунок 5 – Выполнение l31.com после выгрузки резидента

На рисунке 5 действительно видно, что память резидентного обработчика была освобождена, ранее он занимал блок 5, что видно на рис. 2.

Контрольные вопросы.

1. Как реализован механизм прерывания от часов?

Любой компьютер содержит системный таймер. Это устройство вырабатывает прерывание INT 8h приблизительно 18,2 раза в секунду. При инициализации BIOS устанавливает свой обработчик для прерывания таймера. Этот обработчик каждый раз увеличивает на 1 текущее значение счетчика тиков таймера. В конце этот обработчик вызывает прерывание int 1Ch – пользовательское прерывание по таймеру (по соответствующему адресу в таблице векторов прерываний). После инициализации системы вектор INT 1Ch указывает на команду IRET, однако в реализованной в данной работе

программе вектор указывает на пользовательский обработчик, который выводит на экран значение счетчика вызовов прерываний системного таймера. Во время выполнения int 8h и int 1Ch все аппаратные прерывания не вызываются. После выполнения обработчиков осуществляется возврат к коду, выполнение которого было прервано

2. Какого типа прерывания использовались в работе?

1) Программные (int 21h, int 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

INTER   PROC    FAR
        jmp     INTER_START
      INTER_DATA:
            COUNTER            DB  "000 interrupts"
            SIGNATURE          DW  3158h
            KEEP_IP             DW  0
            KEEP_CS             DW  0
            KEEP_PSP            DW  0

    INTER_START:
            push   AX
            push    BX
            push    CX
            push    DX
            push    SI
        push    ES
        push    DS
            mov    AX, seg COUNTER
            mov    DS, AX

    SET_CURSOR:
        mov     AH, 03h
            mov     BH, 0h
            int     10h
        push    DX

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

      INCREASE:
            mov    AX, SEG COUNTER
            push   DS
            mov    DS, AX
            mov    SI, offset COUNTER
            add          SI, 2
            mov    CX, 3
      INTER_CYCLE:
            mov    AH, [SI]
            inc    AH
            mov    [SI], AH
            cmp    AH, ':'
            jne    INTER_END_CYCLE
            mov    AH, '0'
```

```
            mov    [SI], AH
            dec    SI
            loop   INTER_CYCLE
      INTER_END_CYCLE:
            pop    DS

      PRINT_COUNTER:
            push   ES
            push   BP
        mov     AX, SEG COUNTER
            mov     ES, AX
            mov     BP, offset 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
INTER    ENDP
    INTER_END:

INTER_CHECK       PROC
            push    AX
            push    BX
            push    SI

            mov     AH, 35h
            mov     AL, 1Ch
            int     21h
            mov     SI, offset SIGNATURE
            sub     SI, offset INTER
            mov     AX, ES:[BX + SI]
            cmp       AX, SIGNATURE
```

```
            jne     INTER_CHECK_END
            mov     IS_LOADED, 1

      INTER_CHECK_END:
            pop     SI
            pop     BX
            pop     AX
      ret
INTER_CHECK       ENDP

INTER_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 INTER
            mov     DX, offset INTER
            mov     DS, AX
            mov     AH, 25h
            mov     AL, 1Ch
            int     21h
            pop          DS

        mov     DX, offset INTER_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
INTER_LOAD        ENDP

INTER_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 INTER
            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
INTER_UNLOAD      ENDP

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    INTER_CHECK
            call    UN_CHECK
            cmp     IS_UN, 1
            je      UNLOAD
            mov     AL, IS_LOADED
            cmp     AL, 1
            jne     LOAD
            mov     DX, offset STR_LOADED_ALREADY
            call    WRITE_STR
            jmp     MAIN_END
      LOAD:
            call    INTER_LOAD
            jmp     MAIN_END
      UNLOAD:
            cmp     IS_LOADED, 1
            jne     NOT_EXIST
            call    INTER_UNLOAD
            jmp     MAIN_END
      NOT_EXIST:
            mov     DX, offset STR_NOT_LOADED
            call    WRITE_STR
      MAIN_END:
            xor   AL, AL
            mov   AH, 4Ch
            int   21h
      MAIN ENDP

CODE    ENDS
```

```
ASTACK  SEGMENT STACK
    DW  128 dup(0)
ASTACK  ENDS

DATA    SEGMENT
       STR_LOADED_ALREADY DB  "Interruption loaded already ",10,13,"$"
       STR_NOT_LOADED     DB  "Interruption isn't loaded",10,13,"$"
    IS_LOADED          DB  0
    IS_UN               DB  0
DATA    ENDS

END   MAIN
```