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

ОТЧЕТ

по лабораторной работе №2

по дисциплине «Операционные системы»

Тема: Исследование интерфейсов программных модулей

Студентка гр. 7381 Кревчик А.Б.

Преподаватель Ефремов М.А.

Санкт-Петербург

2019

Цель работы.

Исследование интерфейса управляющей программы и загрузочных модулей. Этот интерфейс состоит в передаче запускаемой программе управляющего блока, содержащего адреса и системный данные. Так загрузчик строит префикс сегмента программы (PSP) и помещает его адрес в сегментный регистр. Исследование префикса сегмента программы (PSP) и среды, передаваемой программе.

Описание функций.

TETR_TO_HEX – вспомогательная функция, переводит из двоичной в шестнадцатеричную систему.

BYTE_TO_HEX – переводит число из регистра AL в шестнадцатиричную систему.

WRD_TO_HEX – переводит число из регистра AX в шестнадцатиричную систему.

PRINT – печатает сообщение на экран.

Результат запуска программы представлен на рис.1.

Рисунок 1 – Результат работы программы

Выводы.

В ходе выполнения данной лабораторной работы были исследованы интерфейс управляющей программы и загрузочных модулей.

Ответы на контрольные вопросы.

Сегментный адрес недоступной памяти.

1. На какую область памяти указывает адрес недоступной памяти?

На границу оперативной памяти и на границу области, доступной для загрузки программ.

2. Где расположен этот адрес по отношению области памяти, отведённой программе?

Адрес располагается сразу за памятью, выделенной для программы.

3. Можно ли в эту область памяти писать?

Можно, т.к. в DOS нет защиты памяти.

Среда передаваемая программе.

1. Что такое среда?

Среда представляет собой область памяти, в которой в виде символьных строк записаны значения переменных, называемых переменными среды.

2. Когда создается среда? Перед запуском приложения или в другое время?

Создается при загрузке DOS, а при запуске программы происходит копирование среды в новую область памяти.

3. Откуда берется информация, записываемая в среду?

Из системного файла autoexec.bat.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

```
     TESTPC        SEGMENT
                   ASSUME  CS:TESTPC, DS:TESTPC, ES:NOTHING, SS:NOTHING
                   ORG       100H
     START:        JMP       BEGIN

     ;ДАННЫЕ
         ADDRES_OF_UNAVALIABLE_MEM  DB 'SEGMENT ADDRESS OF UNAVAILABLE
MEMORY:     ', 13, 10 ,'$'
         ADDRES_OF_ENVIRONMENT           DB 'SEGMENT ADDRESS OF
ENVIRONMENT:     ', 13, 10, '$'
         TAIL                            DB 'TAIL OF COMAND LINE
', 13, 10, 10, '$'
         NEW_LINE                        DB 'EMPTY LINE', 13, 10, 10, '$'
         CONTENT_LINE                         DB 'CONTENT OF THE
ENVIRONMENT:', 10, 13, '$'
         EMPTY                           DB ' ', 13, 10, '$'
         PATH_LINE                       DB 'PATH:', 10, 13, '$'

     ;ПРОЦЕДУРЫ

     TETR_TO_HEX PROC NEAR
                 AND  AL,0FH
                 CMP  AL,09
                 JBE  NEXT
                 ADD  AL,07
     NEXT:       ADD  AL,30H
                 RET
     TETR_TO_HEX ENDP

     BYTE_TO_HEX PROC NEAR          ;БАЙТ В AL ПЕРЕВОДИТСЯ В ДВА СИМВОЛА
ШЕСТН. ЧИСЛА В AX
                 PUSH CX
                 MOV  AH,AL
                 CALL TETR_TO_HEX
                 XCHG AL,AH
                 MOV  CL,4
                 SHR  AL,CL
                 CALL TETR_TO_HEX    ;В AL СТАРШАЯ ЦИФРА
                 POP  CX                  ;В AH МЛАДШАЯ
                 RET
     BYTE_TO_HEX ENDP

     WRD_TO_HEX PROC NEAR ;ПЕРЕВОД В 16 С/С 16-ТИ РАЗРЯДНОГО ЧИСЛА, В AX
- ЧИСЛО, DI - АДРЕС ПОСЛЕДНЕГО СИМВОЛА
                 PUSH BX
```

```
            MOV       BH,AH
            CALL BYTE_TO_HEX
            MOV       [DI],AH
            DEC       DI
            MOV       [DI],AL
            DEC       DI
            MOV       AL,BH
            XOR       AH,AH
            CALL BYTE_TO_HEX
            MOV       [DI],AH
            DEC       DI
            MOV       [DI],AL
            POP       BX
            RET
WRD_TO_HEX      ENDP

BYTE_TO_DEC PROC NEAR       ; ПЕРЕВОД БАЙТА В 10С/С, SI - АДРЕС ПОЛЯ
МЛАДШЕЙ ЦИФРЫ
            PUSH AX         ; AL СОДЕРЖИТ ИСХОДНЫЙ БАЙТ
            PUSH CX
            PUSH DX
            XOR  AH,AH
            XOR  DX,DX
            MOV  CX,10
LOOP_BD: DIV    CX
            OR        DL,30H
            MOV  [SI],DL
            DEC  SI
            XOR  DX,DX
            CMP  AX,10
            JAE  LOOP_BD
            CMP  AL,00H
            JE        END_L
            OR        AL,30H
            MOV  [SI],AL
END_L:  POP     DX
            POP  CX
            POP       AX
            RET
BYTE_TO_DEC ENDP

PRINT PROC NEAR           ;ПЕЧАТЬ СООБЩЕНИЯ НА ЭКРАН
            PUSH AX
            MOV  AH, 09H
            INT  21H
            POP  AX
            RET
PRINT ENDP

ADDRES_OF_MEMORY PROC NEAR
```

```
            PUSH AX
            PUSH DI
            PUSH DX
            MOV       AX,DS:[02H]
            MOV       DI, OFFSET ADDRES_OF_UNAVALIABLE_MEM
            ADD       DI, 43
            CALL WRD_TO_HEX
            MOV       DX, OFFSET ADDRES_OF_UNAVALIABLE_MEM
            CALL PRINT
            POP       DX
            POP       DI
            POP       AX
            RET
ADDRES_OF_MEMORY ENDP

ENVIROMENT_ADDRES PROC NEAR
            PUSH AX
            PUSH DI
            PUSH DX
            MOV       AX, DS:[2CH]
            MOV       DI, OFFSET ADDRES_OF_ENVIRONMENT
            ADD       DI, 36
            CALL WRD_TO_HEX
            MOV       DX, OFFSET ADDRES_OF_ENVIRONMENT
            CALL PRINT
            POP       DX
            POP       DI
            POP       AX
            RET
ENVIROMENT_ADDRES ENDP

GET_TAIL PROC NEAR
            PUSH AX
            PUSH CX
            PUSH DI
            PUSH SI
            XOR       CX, CX
            MOV       CL, DS:[080H]
            MOV       SI, OFFSET TAIL
            ADD  SI,  21
            TEST CL, CL
            JZ        EMPTY1
            XOR       DI, DI
            XOR       AX, AX
      GETTAIL:
            MOV       AL, DS:[081H+DI]
            MOV       [SI], AL
            INC       DI
            INC       SI
            LOOP GETTAIL
```

```
          MOV       DX, OFFSET TAIL
          CALL PRINT
          JMP  EXIT1
     EMPTY1:
          MOV       DX, OFFSET NEW_LINE
          CALL PRINT
     EXIT1:
          POP  SI
          POP       DI
          POP       CX
          POP       AX
          RET
GET_TAIL ENDP

CONTENT PROC NEAR
          PUSH AX
          PUSH DX
          PUSH DS
          PUSH ES
          MOV  DX, OFFSET CONTENT_LINE
          CALL PRINT
          MOV  AH, 02H
          MOV  ES, DS:[2CH]
          XOR  SI, SI
     WRITECONT:
          MOV  DL, ES:[SI]
          INT  21H
          CMP  DL, 0H
          JE        ENDOFLINE
          INC  SI
          JMP  WRITECONT
     ENDOFLINE:
          MOV  DX, OFFSET EMPTY
          CALL PRINT
          INC  SI
          MOV  DL, ES:[SI]
          CMP  DL, 0H
          JNE  WRITECONT
          MOV  DX, OFFSET EMPTY
          CALL PRINT
          POP  ES
          POP  DS
          POP  DX
          POP  AX
          RET
CONTENT ENDP

PATH PROC NEAR
          PUSH AX
          PUSH DX
```

```
          PUSH DS
          PUSH ES
          MOV  DX, OFFSET PATH_LINE
          CALL PRINT
          ADD  SI, 3H
          MOV  AH, 02H
          MOV  ES, DS:[2CH]
     WRITEMSG:
          MOV  DL, ES:[SI]
          CMP  DL, 0H
          JE        ENDOFPATH
          INT  21H
          INC  SI
          JMP  WRITEMSG
     ENDOFPATH:
          POP  ES
          POP  DS
          POP  DX
          POP  AX
          RET
PATH ENDP

BEGIN:
     CALL      ADDRES_OF_MEMORY
     CALL      ENVIROMENT_ADDRES
     CALL      GET_TAIL
     CALL      CONTENT
     CALL      PATH
     XOR       AL,AL
     MOV       AH,4CH
     INT       21H
TESTPC ENDS
END START
```