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

ОТЧЕТ

по лабораторной работе №2
по дисциплине «Операционные системы»
Тема: «Исследование интерфейсов программных модулей»

Студент гр. 7381 _____ Вологдин М.Д.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург
2019

Цель работы.

Исследование интерфейса управляющей программы и загрузочных модулей. Исследование префикса сегмента программы (PSP) и среды, передаваемой программе.

Основные теоретические положения.

Необходимо создать программный модуль типа .COM, который распечатывает следующую информацию:

1. Сегментный адрес недоступной памяти, взятый из PSP, в шестнадцатеричном виде.
2. Сегментный адрес среды, передаваемой программе, в шестнадцатеричном виде.
3. Хвост командной строки в символьном виде.
4. Содержимое области среды в символьном виде.
5. Путь загружаемого модуля.

Описание функций и структур данных

Все функции расположены в табл. 1, структуры данных – в табл.2

Таблица 1 – Описание функций.

PRINT_INACCESIBLE_MEMORY	печатает сегментный адрес недоступной памяти в шестнадцатеричном виде
PRINT_ENV_ADDRES	печатает сегментный адрес среды
PRINT_ARGV	выводит аргументы командной строки или сообщение об их отсутствии
PRINT_ENV	выводит содержимое области среды и путь загружаемого модуля в символьном виде
BYTE_TO_HEX	перевод байта регистра AL в десятичную систему счисления, результат в SI
WRD_TO_HEX	перевод двух байт регистра AX в шестнадцатеричную систему счисления, результат в регистр DI

BYTE_TO_DEC	перевод байта регистра AL в десятичную систему счисления, помещая результат в SI

Таблица 2 – Описание структур данных.

INACCESSMEMADDRINFO	Строка, информирующая о том, что дальше выведется адрес недоступной памяти
INACCESSMEMADDR	Строка для хранения адреса недоступной памяти в символьном виде
ENVADDRINFO	Строка, информирующая о том, что дальше выведется адрес среды окружения
ENVADDR	Строка для хранения адреса среды окружения в символьном виде
TAILPRNTINFO	Строка, информирующая о том, что дальше выведется хвост командной строки
TAIL	Строка из 80 байтов для хранения хвоста командной строки
NOTAIL	Строка, информирующая, что хвоста нет
ENVCONTENTINFO	Строка, информирующая, что далее следует содержимое среды окружения
PRGRMPATHINFO	Строка, информирующая, что далее следует путь программы
_ENDL	Строка, переводящая каретку на начало новой строки

Выполнение работы

1. Определили сегментный адреса недоступной памяти и распечатали его.
2. Определили сегментный адрес среды и распечатали его.
3. Вывели хвост командной строки в символьном виде или сообщение о том, что его нет.
4. Вывели содержимое области среды и путь загружаемого модуля в символьном виде.

Результат работы программы показан на рис. 1.

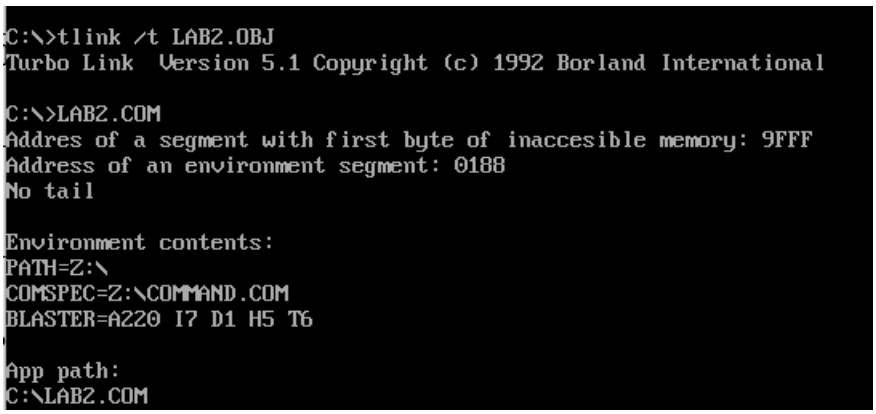

Рисунок 1 – Результат работы программы

Вывод

В ходе выполнения работы было проведено изучение интерфейса управляющей программы и загрузочных модулей, префикса сегмента программы (PSP) и параметров среды, передаваемой программе при её запуске.

Ответы на контрольные вопросы

- Сегментный адрес недоступной памяти программ

1. На какую область памяти указывает адрес недоступной памяти?

   Ответ: Адрес недоступной памяти указывает на область памяти, работа с которой запрещена (ведёт к непредсказуемому поведению).

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

   Ответ: Этот адрес является первым сразу за концом сегмента памяти, отведённой программе, – в нашем случае этим адресом является 9FFF.

3. Можно ли в эту область памяти писать?

   Ответ: При условии отсутствия в управляющей программе операционной системы механизма защиты памяти (как в DOS), запись в эту область возможна.

- Среда, передаваемая программе:

1. Что такое среда?

Ответ: Среда – это набор переменных, хранящих информацию о конфигурации настройках системы в которой запускается приложение.

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

   Ответ: Во время запуска ОС.

3. Откуда берется информация, записываемая в среду?

   Ответ: Эта информация хранится в файле системного реестра. В случае операционной системы MS DOS эта информация берётся из файла AUTOEXEC.BAT.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД

```
TESTPC SEGMENT
 ASSUME CS:TESTPC, DS:TESTPC, ES:NOTHING, SS:NOTHING
 ORG 100H
START: JMP BEGIN
INACCESSMEMADDRINFO DB 'ADDRES OF A SEGMENT WITH FIRST BYTE OF INACCESIBLE MEMORY: $'
ACCESSMEMADDR DB '    $'
ENVADDRINFO DB 'ADDRESS OF AN ENVIRONMENT SEGMENT: $'
ENVADDR DB '    $'
TAILPRNTINFO DB 'TAIL:$'
TAIL DB 50H DUP(' '),'$'
NOTAIL DB 'NO TAIL$'
ENVCONTENTINFO DB 'ENVIRONMENT CONTENTS:',0DH,0AH,'$'
PRGRMPATHINFO DB 'APP PATH:',0DH,0AH,'$'
_ENDL DB 0DH,0AH,'$'
PRINT PROC NEAR
     MOV AH,09H
     INT 21H
     RET
PRINT ENDP

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
     PUSH CX
```

```
      MOV AH,AL
      CALL TETR_TO_HEX
      XCHG AL,AH
      MOV CL,4
      SHR AL,CL
      CALL TETR_TO_HEX
      POP CX
      RET
BYTE_TO_HEX ENDP

WRD_TO_HEX PROC NEAR
      PUSH BX
      MOV BH,AH
      CALL BYTE_TO_HEX
      MOV [DI],AH
      DEC DI
      MOV [DI],AL
      DEC DI
      MOV AL,BH
      CALL BYTE_TO_HEX
      MOV [DI],AH
      DEC DI
      MOV [DI],AL
      POP BX
      RET
WRD_TO_HEX ENDP

BYTE_TO_DEC PROC NEAR
      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
      RET
BYTE_TO_DEC ENDP

GET_INACCESS_MEM_ADDR PROC NEAR
      MOV AX,DS:[2]
      MOV ES,AX
      MOV DI,OFFSET ACCESSMEMADDR+3
      CALL WRD_TO_HEX
      MOV DX,OFFSET INACCESSMEMADDRINFO
      CALL PRINT
      MOV DX,OFFSET ACCESSMEMADDR
      CALL PRINT
      MOV DX,OFFSET _ENDL
      CALL PRINT
      ;MOV AX,01000H
      ;MOV ES:[0H],AX ; WORKS IN DOS
      RET
GET_INACCESS_MEM_ADDR ENDP

GET_ENV_ADDR PROC NEAR
      MOV AX,DS:[2CH]
      MOV DI,OFFSET ENVADDR+3
```

```
      CALL WRD_TO_HEX
      MOV DX,OFFSET ENVADDRINFO
      CALL PRINT
      MOV DX,OFFSET ENVADDR
      CALL PRINT
      MOV DX,OFFSET _ENDL
      CALL PRINT
      RET
GET_ENV_ADDR ENDP

PRINT_TAIL PROC NEAR
      XOR CH,CH
      MOV CL,DS:[80H]

      CMP CL,0
      JNE NOTNIL
            MOV DX,OFFSET NOTAIL
            CALL PRINT
            MOV DX,OFFSET _ENDL
            CALL PRINT
            RET
      NOTNIL:

      MOV DX,OFFSET TAILPRNTINFO
      CALL PRINT

      MOV BP,OFFSET TAIL
      CYCLE:
            MOV DI,CX
            MOV BL,DS:[DI+80H]
            MOV DS:[BP+DI-1],BL
      LOOP CYCLE

      MOV DX,OFFSET TAIL
```

```
      CALL PRINT
      RET
PRINT_TAIL ENDP

PRINT_ENV PROC NEAR
      MOV DX, OFFSET _ENDL
      CALL PRINT
      MOV DX, OFFSET ENVCONTENTINFO
      CALL PRINT

      MOV AX,DS:[2CH]
      MOV ES,AX

      XOR BP,BP
      PE_CYCLE1:
            CMP WORD PTR ES:[BP],0001H
            JE PE_EXIT1
            CMP BYTE PTR ES:[BP],00H
            JNE PE_NOENDL
                  MOV DX,OFFSET _ENDL
                  CALL PRINT
                  INC BP
            PE_NOENDL:
            MOV DL,ES:[BP]
            MOV AH,02H
            INT 21H
            INC BP
      JMP PE_CYCLE1
      PE_EXIT1:
      ADD BP,2

      MOV DX, OFFSET _ENDL
      CALL PRINT
      MOV DX, OFFSET PRGRMPATHINFO
```

```
      CALL PRINT

      PE_CYCLE2:
            CMP BYTE PTR ES:[BP],00H
            JE PE_EXIT2
            MOV DL,ES:[BP]
            MOV AH,02H
            INT 21H
            INC BP
      JMP PE_CYCLE2
      PE_EXIT2:

      RET
PRINT_ENV ENDP

BEGIN:
      CALL GET_INACCESS_MEM_ADDR
      CALL GET_ENV_ADDR
      CALL PRINT_TAIL
      CALL PRINT_ENV
      XOR AL,AL
      MOV AH,4CH
      INT 21H
TESTPC ENDS
 END START
```