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

Студент гр. 7383 _____ Корякин М. П.

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 и структур данных.

Название функции	**Назначение**
BYTE_TO_HEX	переводит число AL в коды символов 16-ой с/с, записывая получившееся в al и ah
TETR_TO_HEX	вспомогательная функция для работы функции BYTE_TO_HEX
WRD_TO_HEX	переводит число AX в строку в 16-ой с/с, записывая получившееся в di, начиная с младшей цифры
BYTE_TO_DEC	переводит байт из AL в десятичную с/с и записывает получившееся число по адресу si, начиная с младшей цифры
FIRST	Начальная функция программы. Находит и выводит на экран: сегментный адрес недоступной памяти, сегментный адрес среды, хвост командной строки, содержимое облаcи среды, путь загружаемого модуля.
PRINT	Вывод на экран

Название	Тип	Назначение
SAUMT_PSP	db	Строка, информирующая о том, что дальше выведется адрес недоступной памяти
FOR_PSP	db	Строка для хранения адреса недоступной памяти в символьном виде
SAMT_P	db	Строка, информирующая о том, что дальше выведется адрес среды окружения
FOR_P	db	Строка для хранения адреса среды окружения в символьном виде
CMLT	db	Хвост командной строки
CE	db	Содержимое среды окружения
LMP	db	Путь программы
ENDL	db	Новая строка

Последовательность действий, выполняемых утилитой.

1. Печатает сегментный адрес первого байта недоступной памяти
2. Печатает сегментный адрес среды, передаваемой программе
3. Печатает хвост командной строки
4. Печатает содержимое области среды в символьном виде
5. Печатает путь загружаемого модуля
6. Выходит в DOS

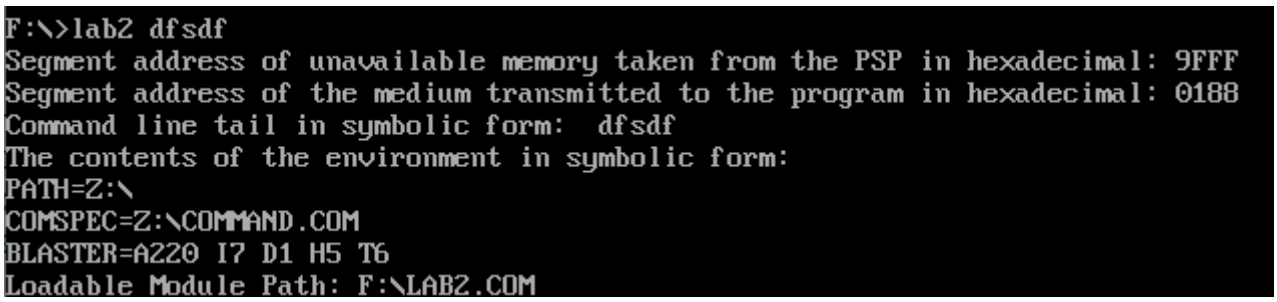

Рисунок 1 – результат работы программы lab2.com

Выводы.

В процессе выполнения данной лабораторной работы были исследованы интерфейс управляющей программы, загрузочных модулей, префикс сегмента программы(PSP) и среды, передаваемой программе.

Ответы на контрольные вопросы.

Сегментный адрес недоступной памяти:

1. На какую область памяти указывает адрес недоступной памяти?

   Этот адрес указывает на границу области основной опертивной памяти и памяти доступной для загрузки программ.

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

   Сразу после области памяти, отведенной программе.

3. Можно ли в эту область памяти писать?

   Да, можно. Потому что в DOS не предусмотрена защита памяти.

Среда, передаваемая программе:

1. Что такое среда?

   Это 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?

   При загрузке DOS, а при запуске приложения копируется в новую область памяти – создаваемую среду программы.

3. Откуда берется информация, записываемая в среду?

   В MS-DOS она берется из системного файла autoexec.bat.

ПРИЛОЖЕНИЕ А

lab2.asm

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

SAUMT_PSP db 'Segment address of unavailable memory taken from the PSP
in hexadecimal: '
FOR_PSP db '    ',0DH,0AH,'$'
SAMT_P db 'Segment address of the medium transmitted to the program in
hexadecimal: '
FOR_P db '    ',0DH,0AH,'$'
CMLT db 'Command line tail in symbolic form: ','$'
CE db 'The contents of the environment in symbolic form: ',0DH,0AH,'$'
LMP db 'Loadable Module Path: ','$'
ENDL db 0DH,0AH,'$'


PRINT PROC
      push ax
      mov ah,09h
      int 21h
      pop ax
      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


FIRST:
;Segment address of unavailable memory taken from the PSP in
hexadecimal
      mov ax,es:[2]
      mov di,offset FOR_PSP+3
      call WRD_TO_HEX
      mov dx, offset SAUMT_PSP
      call PRINT
```

```
;Segment address of the medium transmitted to the program in
hexadecimal
      mov ax,es:[2Ch]
      mov di,offset FOR_P+3
      call WRD_TO_HEX
      lea dx,SAMT_P
      call PRINT

;Command line tail in symbolic form
      mov dx,offset CMLT
      call PRINT
      mov cx,0
      mov cl,es:[80h]
      cmp cl,0
      je START_CE
      mov dx,81h
      mov bx,0
      mov ah,02h
      TAIL_loop:
            mov dl,es:[bx+81h]
            int 21h
            inc bx
      loop TAIL_loop


;The contents of the environment in symbolic form
START_CE:
      mov dx,offset ENDL
      call PRINT
      mov dx, offset CE
      call PRINT
      mov ax, es:[2Ch]
      mov es, ax
      ;xor SI, SI
      mov bx, 0
      mov ah, 02h
out_ce:
      cmp word ptr es:[bx], 0000h
      je ending_ce
      cmp byte ptr es:[bx], 00h
      jne missing
      mov dx,offset ENDL
      call PRINT
      inc bx
missing:
      mov dl, es:[bx]
      int 21h
      inc bx
      jmp out_ce
ending_ce:
      mov dx,offset ENDL
```

```
      call PRINT
;Loadable Module Path
      add bx, 4;
      mov dx, offset LMP
      call PRINT
      mov ah, 02h
out_lmp:
      cmp byte ptr es:[bx], 00h
      je ending_lmp
      mov dl, es:[bx]
      int 21h
      inc bx
      jmp out_lmp
ending_lmp:

      mov dx,offset ENDL
      call PRINT
;finish
      mov ah,4Ch
      int 21H
TESTPC ENDS
 END START
```