**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. 7383 _____ Прокопенко Н.

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.**

Описание функций:

| **Название функции** | **Назначение** |
|---|---|
| GET_adr_off_mem | печатает адрес недоступной памяти |
| GET_Seg_adr | печатает сегментный адрес среды |
| Write | вызывает функцию печати строки |
| TAIL | печатает хвост командной строки |
| SREDA | печатает содержимое области среды в символьном виде |
| 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 |

Описание структур данных:

| **Название** | **Тип** | **Назначение** |
|---|---|---|
| off_mem_ | db | Строка, информирующая о том, что дальше выведется адрес недоступной памяти |
| off_mem | db | Строка для хранения адреса недоступной памяти в символьном виде |
| Seg_adr_ | db | Строка, информирующая о том, что дальше выведется адрес среды окружения |
| Seg_adr | db | Строка для хранения адреса среды окружения в символьном виде |
| TAIL_ | db | Хвост командной строки |
| SREDA_ | db | Содержимое среды окружения |
| PATH_ | 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

Результат запуска программы представлены на рисунке 1.

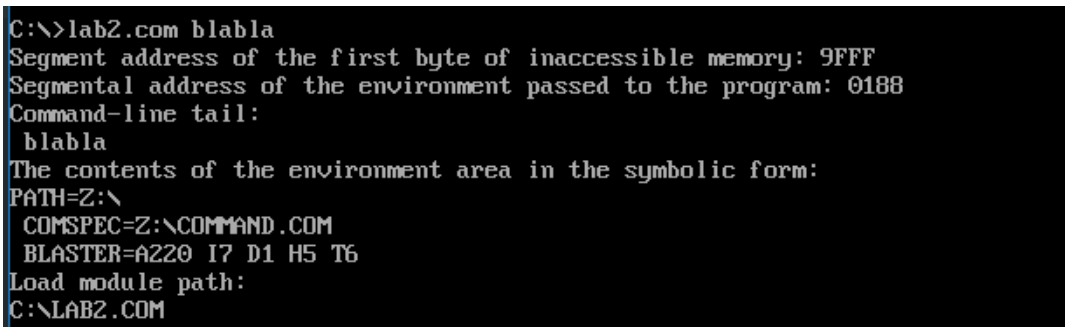


Рисунок 1 – результат работы программы lab2.com

**Выводы.**

В процессе выполнения данной лабораторной работы были исследованы интерфейс управляющей программы и загрузочных модулей. Код программы lab2.asm представлен в приложении А.

**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, отведенной программе?

   Сразу за областью памяти, отведенной программе.

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?

   Среда создается при загрузке DOS, а при запуске приложения копируется в новую область памяти.

3) Откуда берется информация, записываемая в среду?

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

## ПРИЛОЖЕНИЕ А

### lab.asm

```
TESTPC SEGMENT
 ASSUME CS:TESTPC, DS:TESTPC, ES:NOTHING, SS:NOTHING
 ORG 100H
 START: JMP BEGIN
; ДАННЫЕ
off_mem_  db 'Segment address of the first byte of inaccessible memory: '
off_mem   db '    ',0DH,0AH,'$'
Seg_adr_  db 'Segmental address of the environment passed to the program: '
Seg_adr        db '    ',0DH,0AH,'$'
TAIL_          db 'Command-line tail: ',0DH,0AH,'$'
SREDA_         db 'The contents of the environment area in the symbolic form: ',0DH,0AH,'$'
PATH_          db 'Load module path: ',0DH,0AH,'$'
ENDL      db 0DH,0AH,'$'
; ПРОЦЕДУРЫ
;---------------------------------------
Write PROC near
     push ax
     mov ah,09h
     int 21h
     pop ax
     ret
Write ENDP
;---------------------------------------
GET_adr_off_mem PROC near
     mov ax,es:[2]
     mov di,offset off_mem+3
     call WRD_TO_HEX
     lea dx,off_mem_
     call Write
     ret
GET_adr_off_mem ENDP
;---------------------------------------
GET_Seg_adr PROC near
     mov ax,es:[2Ch]
     mov di,offset Seg_adr+3
     call WRD_TO_HEX
     lea dx,Seg_adr_
     call Write
```

```
     ret
GET_Seg_adr  ENDP
;---------------------------------------
TAIL PROC near
     mov dx,offset TAIL_
     call Write
     mov cx,0
     mov cl,es:[80h]
     cmp cl,0
     je TAIL_END
     mov dx,81h
     mov bx,0
     mov ah,02h
     TAIL_loop:
          mov dl,es:[bx+81h]
          int 21h
          inc  bx
     loop TAIL_loop
     mov dx,offset ENDL
     call Write
     TAIL_END:
     ret
TAIL ENDP
;---------------------------------------
SREDA PROC near
     mov dx,offset SREDA_
     call Write
     push es
     ; кладём в es адрес области среды
     mov ax,es:[2Ch]
     mov es,ax
     mov ah,02h
     mov bx,0
     SREDA_loop:
          mov dl,es:[bx]
          int 21h
          inc  bx
          cmp byte ptr es:[bx],00h
          jne SREDA_loop
          mov dx,offset ENDL
          call Write
          cmp word ptr es:[bx],0000h
          jne SREDA_loop
     add bx,4 ; пропускаем 0001
```

```
     mov dx,offset PATH_
     call Write

     SREDA_loop1:
          mov dl,es:[bx]
          int 21h
          inc  bx
          cmp byte ptr es:[bx],00h
          jne SREDA_loop1
     mov dx,offset ENDL
     call Write
     pop es
     ret
SREDA 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
;--------------------------------------
; перевод в 16с/с 16-ти разрядного числа
; в AX - число, DI - адрес последнего символа
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
;--------------------------------------
BEGIN:
      call GET_adr_off_mem
      call GET_Seg_adr
      call TAIL
      call SREDA
      xor AL,AL
      mov AH,4Ch
      int 21H
TESTPC ENDS
 END START
```