

**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. 9383

\_\_\_\_\_

Соседков К.С.

Преподаватель

\_\_\_\_\_

Ефремов М.А.

Санкт-Петербург

2020

### **Цель работы.**

Исследование интерфейса управляющей программы и загрузочных модулей.

### **Задание.**

**Шаг 1.** Для выполнения лабораторной работы необходимо написать и отладить программный модуль типа .COM, который выбирает и распечатывает следующую информацию:

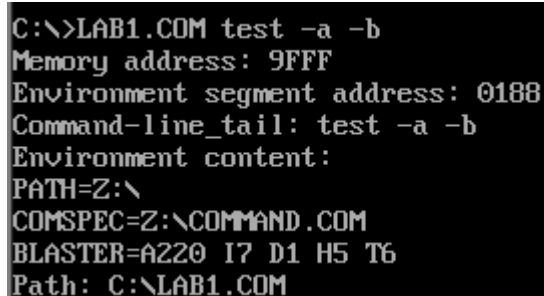
- 1) Сегментный адрес недоступной памяти, взятый из PSP, в шестнадцатеричном виде.
- 2) Сегментный адрес среды, передаваемой программе, в шестнадцатеричном виде.
- 3) Хвост командной строки в символьном виде.
- 4) Содержимое области среды в символьном виде.
- 5) Путь загружаемого модуля.

Сохраните результаты, полученные программой, и включите их в отчет.

**Шаг 2.** Оформление отчета в соответствии с требованиями. В отчет включите скриншот с запуском программы и результатами.

### **Выполнение работы.**

При выполнении работы на языке ассемблера был реализован программный модуль типа COM. Результат исполнения приведен на рисунке 1.



```
C:\>LAB1.COM test -a -b
Memory address: 9FFF
Environment segment address: 0188
Command-line_tail: test -a -b
Environment content:
PATH=Z:\
COMSPEC=Z:\COMMAND.COM
BLASTER=A220 I7 D1 H5 T6
Path: C:\LAB1.COM
```

Рисунок 1: Результат исполнения

### **Контрольные вопросы к шагу 1.**

#### **Сегментный адрес недоступной памяти.**

1) На какую область памяти указывает адрес недоступной памяти?

Адрес недоступной памяти указывает на первый байт сегмента, который расположен сразу после выделенной под программу памяти.

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

Адрес расположен по адресу  $2h$ .

3) Можно ли в эту область памяти писать?

Да, т.к. в DOS отсутствуют механизмы защиты памяти.

#### **Среда передаваемая программе.**

1) Что такое среда?

Область памяти которая хранит символьные строки вида:  
<переменная>=<значение>.

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

Среда создается перед запуском приложения.

3) Откуда берется информация, записываемая в среду?

В DOS для настройки среды используется системный пакетный файл AUTOEXEC.BAT который исполняется интерпретатором командной строки COMMAND.COM.

**Выводы.**

При выполнении лабораторной работы была исследована структура интерфейса управляющей программы и загрузочных модулей.

## ПРИЛОЖЕНИЕ А

### ИСХОДНЫЙ КОД ПРОГРАММ

Название файла: lab2.asm

TESTPC SEGMENT

ASSUME CS:TESTPC, DS:TESTPC, ES:NOTHING, SS:NOTHING

ORG 100H

START: JMP BEGIN

seg\_memory db 'Memory address:     ',0DH,0AH,'\$'

environment\_seg\_address db 'Environment segment address:     ',0DH,0AH,'\$'

command\_line\_tail db 'Command-line\_tail:\$'

environment\_content db 'Environment content:',0DH,0AH,'\$'

path db 'Path: \$'

;-----

BYTE\_TO\_DEC PROC near

push AX

push BX

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 BX
    pop AX
    ret
BYTE_TO_DEC 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
    ; 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
    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:

```

;SEGMENT MEMORY
mov ax, ds:[02h]
mov di, offset seg_memory+19
call WRD_TO_HEX

mov dx, offset seg_memory
mov ah,09h
int 21h

```

```
;Environment segment address
mov ax, ds:[02ch]
mov di, offset environment_seg_address+32
call WRD_TO_HEX
```

```
mov dx, offset environment_seg_address
mov ah,09h
int 21h
```

```
xor cx, cx
mov cl, ds:[080h]
```

```
cmp cl, 00h
je read_environment_content
```

```
;CMD Tail
mov dx, offset command_line_tail
mov ah,09h
int 21h
```

```
xor di, di
mov ah, 02h
cmd_tail_loop:
    mov dl, ds:[081h+di]
    int 21h
    inc di
loop cmd_tail_loop
```



```
;new line
mov dl, 0DH
int 21h
mov dl, 0AH
int 21h
```

```
read_environment_content:
;Environment content
mov dx, offset environment_content
mov ah,09h
int 21h
```

```
mov bx, [2ch]
mov es, [bx]
mov ah, 02h
xor di, di
```

```
env_content_loop:
    mov dl, es:[di]
    cmp dl, 00h
    je env_content_loop_end
    int 21h
    inc di
    jmp env_content_loop
env_content_loop_end:
    inc di
    mov dl, es:[di]
    cmp dl, 00h
    je read_path
    ;new line
    mov dl, 0DH
    int 21h
```

```
mov dl, 0AH
int 21h
jmp env_content_loop
```

```
read_path:
    add di, 3
    ;new line
    mov dl, 0DH
    int 21h
    mov dl, 0AH
    int 21h
```

```
mov dx, offset path
mov ah, 09h
int 21h
mov ah, 02h
read_path_loop:
    mov dl, es:[di]
    cmp dl, 00h
    je exit
    int 21h
    inc di
    jmp read_path_loop
```

```
exit:
    xor al,al
    mov ah,4ch
    int 21h
TESTPC ENDS
END START
```