

**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. 8383

Кормщикова А. О.

Преподаватель

Ефремов М. А.

Санкт-Петербург

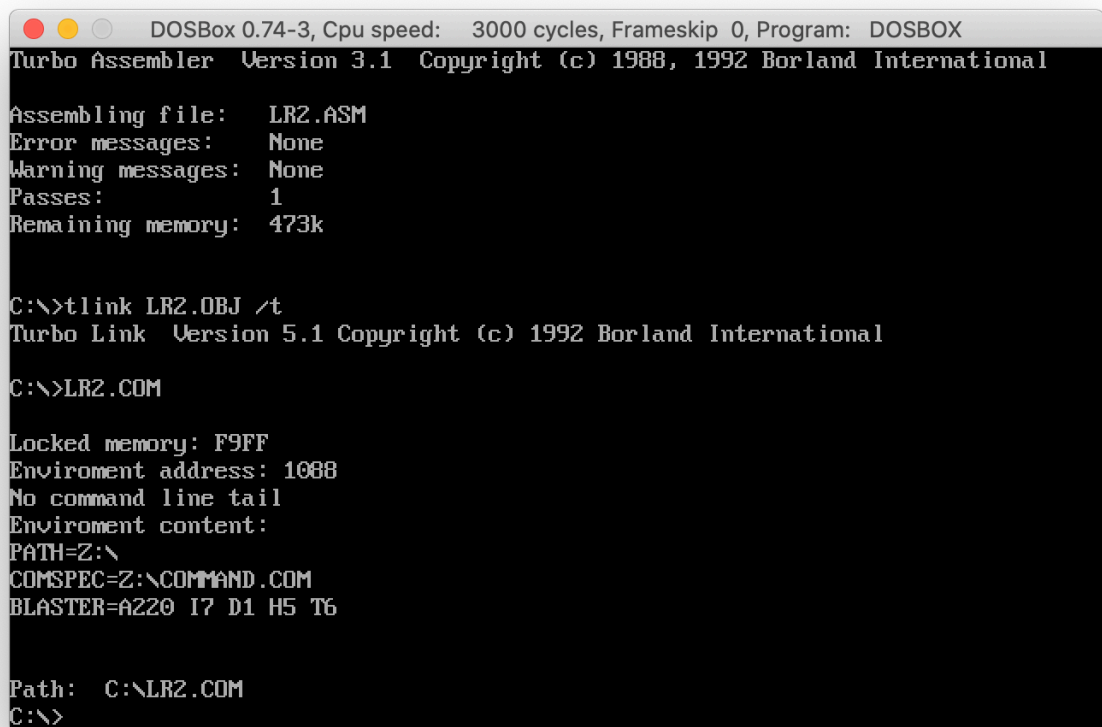
2020

### **Цель работы.**

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

### **Ход выполнения.**

Был написан код исходного .COM модуля (LR2.ASM представлен в приложении А), который выбирает и распечатывает следующую информацию: сегментный адрес недоступной памяти, взятый из PSP, в шестнадцатеричном виде; сегментный адрес среды, передаваемой программе, в шестнадцатеричном виде; хвост командной строки в символьном виде; содержимое области среды в символьном виде; путь загружаемого модуля. Результат работы программы показан на рис. 1.



```
DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: DOSBOX
Turbo Assembler Version 3.1 Copyright (c) 1988, 1992 Borland International
Assembling file: LR2.ASM
Error messages: None
Warning messages: None
Passes: 1
Remaining memory: 473k

C:\>tlink LR2.OBJ /t
Turbo Link Version 5.1 Copyright (c) 1992 Borland International

C:\>LR2.COM

Locked memory: F9FF
Environment address: 1088
No command line tail
Environment content:
PATH=Z:\
COMSPEC=Z:\COMMAND.COM
BLASTER=A220 I7 D1 H5 T6

Path: C:\LR2.COM
C:\>
```

Рисунок 1 - Результат выполнения .COM модуля

### **Сегментный адрес недоступной памяти**

1) На какую область памяти указывает адрес недоступной памяти?

На служебную часть памяти, эта память не может быть выделена DOS под программу.

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

Непосредственно за памятью, отведенной DOS пользовательским программам.

3) Можно ли в эту область памяти писать?

В DOS отсутствуют механизмы защиты памяти, поэтому в эту область памяти можно писать.

### **Среда передаваемая программе**

1) Что такое среда?

Среда - последовательность символьных строк вида имя=параметр, каждая строка заканчивается байтом нулей. Среда также заканчивается байтом нулей, т.е. два нулевых байта являются признаком конца переменных среды.

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

При запуске DOS создается корневая среда, по умолчанию, при запуске пользовательской программы, программа получает копию этой среды. При этом каждая программа может изменить свою копию.

3) Откуда берется информация, записываемая в среду?

Из файла AUTOEXEC.BAT.

### **Выводы.**

В ходе выполнения лабораторной работы были исследованы интерфейсы управляющей программы и загрузочных модулей, а также префикс сегмента программы и среда, передаваемая программе.

## ПРИЛОЖЕНИЕ

### Содержимое файла "хорошего" EXE модуля:

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

;DATA

    LOCKED_MEM db 0DH, 0AH, 'Locked memory: $'
    ENVIROMENT db 0DH, 0AH, 'Enviroment address: $'
    TAIL db 0DH, 0AH, 'Command line tail: $'
    NO_TAIL db 0DH, 0AH, 'No command line tail $'
    ENVIROMENT_C db 0DH, 0AH, 'Enviroment content:', 0DH, 0AH, '$'
    PATH db 0DH, 0AH, 'Path: $'
    NL db 0DH, 0AH, '$'

;-----

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 в 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

;-----

PRINT_LINE PROC near
    push ax
    mov ah, 09h
    int 21h
    pop ax
    ret
PRINT_LINE ENDP

PRINT_SYMBOL PROC near
    push ax
    mov ah, 02h
    int 21h
    pop ax
    ret
PRINT_SYMBOL ENDP

PRINT_HEX PROC near
```

```

        push ax
        push ax
        mov al, ah
        call BYTE_TO_HEX
        mov dl, ah
        call PRINT_SYMBOL
        mov dl, al
        call PRINT_SYMBOL
        pop ax
        call BYTE_TO_HEX
        mov dl, ah
        call PRINT_SYMBOL
        mov dl, al
        call PRINT_SYMBOL
        pop ax
        ret
PRINT_HEX ENDP

BEGIN:

;LOCKED MEMORY
        mov dx, offset LOCKED_MEM
        call PRINT_LINE
        mov ax, ds:[02h]
        call PRINT_HEX

;ENVIROMENT
        mov dx, offset ENVIROMENT
        call PRINT_LINE
        mov ax, ds:[2Ch]
        call PRINT_HEX

;TAIL
        mov cl, ds:[80h]
        cmp cl, 0
        je NTAIL
        mov ch, 0
        mov di, 0
LOOP:
        mov dl, ds:[81h+di]
        call PRINT_SYMBOL
        add di, 1
        loop LOOP
        jmp ENDT

NTAIL:
        mov dx, offset NO_TAIL
        call PRINT_LINE

ENDT:
;Enviroment content
        mov dx, offset ENVIROMENT_C
        call PRINT_LINE
        mov bx, 2ch
        mov es, [bx]
        mov si, 0
EN_C:
        cmp BYTE PTR es:[si], 0h
        je NEXTT
        mov dl, es:[si]
        call PRINT_SYMBOL
        jmp PR

```

```

NEXTT:
    mov dx, offset NL
    call PRINT_LINE
PR:
    add si, 1
    cmp WORD PTR es:[si], 0001h
    je PATH_S
    jmp EN_C

PATH_S:
    mov dx, offset PATH
    call PRINT_LINE
    add si, 2
LOOOOP:
    cmp BYTE PTR es:[si], 00h
    je EXIT
    mov dl, es:[si]
    call PRINT_SYMBOL
    add si, 1
    jmp LOOOOP

;exit to dos
EXIT:
    xor AL,AL
    mov AH,4Ch
    int 21H

LR      ENDS
END START

```