

**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 структур загрузочных модулей**

Студентка гр. 0382

Здобнова К.Д.

Преподаватель

Ефремов М.А.

Санкт-Петербург

2022

### **Цель работы.**

Исследование интерфейса управляющей программы и загрузочных модулей. Этот интерфейс состоит в передаче запускаемой программе управляющего блока, содержащего адреса и системные данные. Так загрузчик строит префикс сегмента программы (PSP) и помещает его адрес в сегментный регистр. Исследование префикса сегмента программы (PSP) и среды, передаваемой программе.

### **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.**

Используя шаблон из методических указаний был написан исходный текст com-модуля (см. приложение А). С помощью программных пакетов TASM и TLINK был получен загрузочный модуль COM, выводящий следующую информацию:

- 1) Сегментный адрес недоступной памяти, взятый из PSP, в шестнадцатеричном виде.

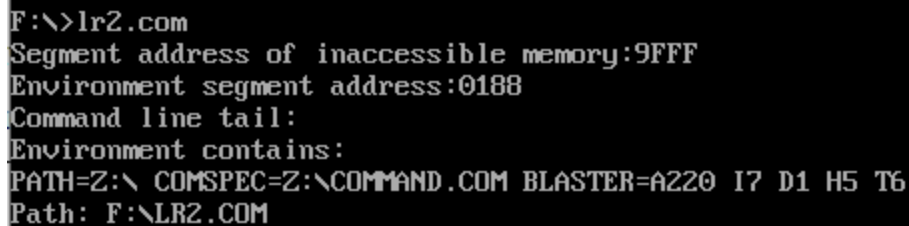
2) Сегментный адрес среды, передаваемой программе, в шестнадцатеричном виде.

3) Хвост командной строки в символьном виде.

4) Содержимое области среды в символьном виде.

5) Путь загружаемого модуля.

Результат вывода программы:



```
F:\>lr2.com
Segment address of inaccessible memory:9FFF
Environment segment address:0188
Command line tail:
Environment contains:
PATH=Z:\ COMSPEC=Z:\COMMAND.COM BLASTER=A220 I7 D1 H5 T6
Path: F:\LR2.COM
```

Рисунок 1. Результат запуска lab2.com.

### Ответы на контрольные вопросы:

Сегментный адрес недоступной памяти

1) На какую область памяти указывает адрес недоступной памяти? – Значение сегментного адреса памяти после памяти, которая была выделена для программы.

2) Где расположен этот адрес по отношению области памяти, отведенной программе? – Первый байт после памяти, которая была выделена для программы.

3) Можно ли в эту область памяти писать? – Доступ к данной памяти не ограничен, поэтому можно.

Среда передаваемая программе

1) Что такое среда? – Среда это последовательность символьных строк вида «имя-параметр», каждая из которых завершается нулевым байтом. Среда также завершается нулевым байтом, что означает, что два нулевых байта - ее конец.

2) Когда создается среда? Перед запуском приложения или в другое время? – При запуске новой программы, она получает в качестве блока среды новый экземпляр, который является копией среды родителя, эта среда может быть изменена. Однако, запуск программ осуществляется с помощью cmd, у которого своя, «корневая» среда, которая создается при запуске ОС. Следовательно, среда создается сразу при загрузке ОС, но может быть изменена в любое время.

3) Откуда берется информация, записываемая в среду? – пакетный файл autoexec.bat.

### **Вывод**

При выполнении данной лабораторной работы был исследован интерфейс управляющей программы и загрузочных модулей.

## ПРИЛОЖЕНИЕ А

### ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: LR2.asm

```
TESTPC SEGMENT
    ASSUME CS:TESTPC, DS:TESTPC, ES:NOTHING, SS:NOTHING
    ORG 100H
START: JMP BEGIN
; ДАННЫЕ
Seg_address_un    db    'Segment address of inaccessible memory: ',
0dh, 0ah, '$'
Seg_address_env    db    'Environment segment address: ', 0dh, 0ah, '$'
Tail_cmd          db    'Command line tail: ', 0dh, 0ah, '$'
Env_cont          db    'Environment contains: ', 0dh, 0ah, '$'
path              db    'Path: $'
nextLine          db    0dh, 0ah, '$'
;ПРОЦЕДУРЫ
;-----
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
; 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
; перевод в 16 с/с 16-ти разрядного числа
; в AX - число, DI - адрес последнего символа
    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
; перевод в 10с/с, SI - адрес поля младшей цифры
        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_1
        or AL,30h
        mov [SI],AL
end_1: pop DX
        pop CX
        ret
BYTE_TO_DEC ENDP
;-----
; КОД
BEGIN:
;Type
        mov ax, es:[0002h]
        mov di, OFFSET Seg_adress_un+42; сдвиг по строке
        call WRD_TO_HEX
        mov dx, OFFSET Seg_adress_un
        call PRINT
        mov dx, OFFSET nextLine
        call PRINT

        mov ax, es:[002Ch]
        mov di, OFFSET Seg_adress_env+31
        call WRD_TO_HEX
        mov dx, OFFSET Seg_adress_env
        call PRINT
        mov dx, OFFSET nextLine
        call PRINT

        mov dx, OFFSET Tail_cmd
        call PRINT
        xor cx, cx
        xor bx, bx
        mov cl, byte PTR es:[80h]
        mov bx, 81h
first:
        cmp cx, 0h

```

```

        je after_first
        mov dl, byte PTR es:[bx]
        mov ah, 02h
        int 21h
        inc bx
        dec cx
        jmp first; Повторяем
after_first:

        push es
        mov dx, OFFSET Env_cont
        call PRINT
        mov bx, es:[002Ch]
        mov es, bx
        xor bx, bx

after_second:
        mov dl, byte PTR es:[bx]
        cmp dl, 0h
        je second
        mov ah, 02h
        int 21h
        inc bx
        jmp after_second
second:
        int 21h
        inc bx
        mov dl, byte PTR es:[bx]
        cmp dl, 0h
        je skip
        jmp after_second

skip:
        mov dx, OFFSET nextLine
        call PRINT

        add bx, 3
        mov dx, OFFSET path
        call PRINT
third:
        mov dl, byte PTR es:[bx]
        cmp dl, 0h
        je skip_last
        mov ah, 02h
        int 21h
        inc bx
        jmp third

skip_last:

; Выход в DOS
        xor AL,AL
        mov AH,4Ch
        int 21h
        ret
TESTPC ENDS
        END START ;конец модуля, START - точка входа

```