

**МИНОБРНАУКИ РОССИИ**  
**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**  
**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**  
**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**  
**Кафедра МОЭВМ**

**ОТЧЕТ**  
**по лабораторной работе №2**  
**по дисциплине «Операционные системы»**  
**Тема: Исследование интерфейсов программных модулей**

Студент гр. 9382

\_\_\_\_\_

Субботин М.О.

Преподаватель

\_\_\_\_\_

Ефремов М.А.

Санкт-Петербург

2021

## 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.

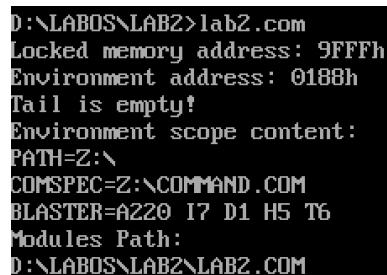
## Основные теоретические положения.

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.

## Ход выполнения:

Были созданы методы PSP\_LOCKED\_MEMORY, PSP\_ENVIRONMENT\_ADDRESS, PSP\_TAIL и PSP\_CONTAINER, которые выводят на экран всю необходимую информацию:



```
D:\LABOS\LAB2>lab2.com
Locked memory address: 9FFFh
Environment address: 0188h
Tail is empty!
Environment scope content:
PATH=Z:\
COMSPEC=Z:\COMMAND.COM
BLASTER=A220 I7 D1 H5 T6
Modules Path:
D:\LABOS\LAB2\LAB2.COM
```

Рисунок 1. Отработка программы (без аргументов)

```
D:\LABOS\LAB2>lab2.com super+default
Locked memory address: 9FFFh
Environment address: 0188h
Tail of command line:  super+default
Environment scope content:
PATH=Z:\
COMSPEC=Z:\COMMAND.COM
BLASTER=A220 I7 D1 H5 T6
Modules Path:
D:\LABOS\LAB2\LAB2.COM
```

Рисунок 2. Оработка программы (с аргументами)

### **Выводы.**

Был исследован интерфейс управляющей программы и загрузочных модулей, префикс сегмента программы (PSP) и среды, передаваемой программе.

## ПРИЛОЖЕНИЕ А

### ОТВЕТЫ НА КОНТРОЛЬНЫЕ ВОПРОСЫ

#### Сегментный адрес недоступной памяти

- 1) На какую область памяти указывает адрес недоступной памяти?

На первый байт, после пространства, отведенного для программы.

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

Расположен сразу же за концом памяти, выделенной программе.

<i>Address</i>	<i>Size</i>	<i>Name</i>
0x0000:0x0000	1024 bytes	Interrupt Vector Table
0x0040:0x0000	256 bytes	<a href="#">BIOS Data Area</a>
0x0050:0x0000	?	Free memory
0x07C0:0x0000	512 bytes	Boot sector code
0x07E0:0x0000	?	Free memory
0xA000:0x0000	64 Kb	Graphics Video Memory
0xB000:0x0000	32 Kb	Monochrome Text Video Memory
0xB800:0x0000	32 Kb	Color Text Video Memory
0xC000:0x0000	256 Kb <sup>1</sup>	ROM Code Memory
0xFFFF:0x0000	16 bytes	More <a href="#">BIOS data</a>

- 3) Можно ли в эту область памяти писать?

Да, в DOS это не запрещено.

#### Среда передаваемая программе

- 1) Что такое среда?

Это область памяти, в которой хранятся переменные этой среды. Есть несколько стандартных переменных среды (PATH).

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

Когда программа запускает другую программу, то новая запущенная программа получает экземпляр блока среды, идентичный родительскому, также можно создать и другую среду.

Изначально среда создается при запуске ОС, и потом в зависимости от требований программ расширяется.

### 3) Откуда берется информация, записываемая в среду?

Из файла AUTOEXEC.BAT, который находится в корневом каталоге загрузочного устройства, также в нем содержатся основные переменные окружения (PATH).

## ПРИЛОЖЕНИЕ Б КОД ПРОГРАММ

### Lab2.asm:

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

LOCKED_MEMORY_ADDRESS DB 'LOCKED MEMORY ADDRESS:
H',0DH,0AH,'$'
ENVIRONMENT_ADDRESS  DB 'ENVIRONMENT ADDRESS:
H',0DH,0AH,'$'
TAIL_COMMAND_LINE    DB 'TAIL OF COMMAND LINE:
',0DH,0AH,'$'
CONTAINER              DB 'ENVIRONMENT SCOPE CONTENT:
',0DH,0AH,'$'
PATH                   DB 'MODULES PATH:
',0DH,0AH,'$'
EMPTY_TAIL             DB 'TAIL IS EMPTY! ',0DH,0AH,'$'
END_STRING            DB 0DH,0AH, '$'

;ПРОЦЕДУРЫ
;-----
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 СС 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 10СС, 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_L
                                OR      AL,30H
                                MOV     [SI],AL
END_L:  POP     DX
        POP     CX
        RET

```

```

BYTE_TO_DEC     ENDP

```

```

;-----

```

```

PSP_LOCKED_MEMORY PROC NEAR
    MOV AX, DS:[02H] ; БЕРЕМ СЕГМЕНТНЫЙ АДРЕС
НЕДОСТУПНОЙ ПАМЯТИ
    MOV DI, OFFSET LOCKED_MEMORY_ADDRESS
    ADD DI, 26
    CALL WRD_TO_HEX
    MOV DX, OFFSET LOCKED_MEMORY_ADDRESS
    MOV AH, 09H
    INT 21H
    RET

```

```

PSP_LOCKED_MEMORY ENDP

```

```

PSP_ENVIRONMENT_ADDRESS PROC NEAR
    MOV AX, DS:[2CH] ; БЕРЕМ ЗНАЧЕНИЕ СЕГМЕНТНОГО АДРЕСА
СРЕДЫ
    MOV DI, OFFSET ENVIRONMENT_ADDRESS
    ADD DI, 24
    CALL WRD_TO_HEX
    MOV DX, OFFSET ENVIRONMENT_ADDRESS
    MOV AH, 09H
    INT 21H
    RET

```

```

PSP_ENVIRONMENT_ADDRESS ENDP

```

```

PSP_TAIL PROC NEAR
    XOR CX, CX
    MOV CL, DS : [80H] ; КЛАДЕМ В CL ЧИСЛО СИМВОЛОВ В ХВОСТЕ
КОМАНДНОЙ СТРОКИ
    MOV SI, OFFSET TAIL_COMMAND_LINE
    ADD SI, 23
    CMP CL, 0H ; СРАВНИВАЕМ КОЛИЧЕСТВО СИМВОЛОВ В ХВОСТЕ С
НУЛЕМ
    JE EMPTY_TAIL_JMP ; ЕСЛИ ХВОСТ ПУСТ
    ; ВЫПОЛНЯЕМ ЕСЛИ ЕСТЬ СИМВОЛЫ
    XOR DI, DI
    XOR AX, AX
READ_TAIL:
    MOV AL, DS : [81H+DI]
    INC DI
    MOV [SI], AL
    INC SI
    LOOP READ_TAIL ; ВЫПОЛНЯЕТСЯ СТОЛЬКО РАЗ, СКОЛЬКО
ЗНАЧЕНИЕ ECX, А ОНО У НАС РАВНО CL
    MOV DX, OFFSET TAIL_COMMAND_LINE
    JMP END_TAIL
EMPTY_TAIL_JMP:
    MOV DX, OFFSET EMPTY_TAIL ; ВЫВОДИМ ИНФОРМАЦИЮ О
ТОМ, ЧТО В ХВОСТЕ НЕТ ЭЛЕМЕНТОВ
END_TAIL:
    MOV AH, 09H
    INT 21H
    RET
PSP_TAIL ENDP

```

```

PSP_CONTAINER PROC NEAR
    MOV DX, OFFSET CONTAINER
    MOV AH, 09H
    INT 21H

    XOR DI, DI
    MOV DS, DS : [2CH] ; ПОЛУЧАЕМ СЕГМЕНТНЫЙ АДРЕС СРЕДЫ

READ_LINE:
    CMP BYTE PTR [DI], 00H
    JZ END_LINE
    MOV DL, [DI]
    MOV AH, 02H ; ВЫВОДИМ
    INT 21H

```



```

        JMP FIND_END
END_LINE:
        CMP BYTE PTR [DI+1],00H
        JZ FIND_END
        PUSH DS

        MOV CX, CS
        MOV DS, CX
        MOV DX, OFFSET END_STRING
        MOV AH, 09H
        INT 21H

        POP DS
FIND_END:
        INC DI
        CMP WORD PTR [DI], 0001H
        JNE READ_LINE ; ЕСЛИ НЕ НАШЛИ 0001H, ЗНАЧИТ ЕЩЕ РАНО
        ДЛЯ МАРШРУТА ЗАГРУЖЕННОЙ ПРОГРАММЫ

        ;ПОЛУЧАЕМ МАРШРУТ ЗАГРУЖЕННОЙ ПРОГРАММЫ
        PUSH DS
        MOV AX, CS
        MOV DS, AX
        MOV DX, OFFSET PATH
        MOV AH, 09H ; ВЫВОДИМ 'MODULES PATH: '
        INT 21H

        POP DS
        ADD DI, 2 ; ПРОПУСКАЕМ 0001H И НАЧИНАЕМ СЧИТЫВАТЬ
        ПУТЬ
LOOP_PATH:
        CMP BYTE PTR [DI], 00H
        JZ END_OF_PATH
        MOV DL, [DI]
        MOV AH, 02H
        INT 21H
        INC DI
        JMP LOOP_PATH
END_OF_PATH:
        RET
PSP_CONTAINER ENDP

```

; КОД

BEGIN:

```
CALL PSP_LOCKED_MEMORY  
CALL PSP_ENVIRONMENT_ADDRESS  
CALL PSP_TAIL  
CALL PSP_CONTAINER
```

;ВЫХОД В DOS

```
XOR     AL,AL  
MOV     AH,4CH  
INT     21H
```

TESTPC

ENDS

END START

;КОНЕЦ МОДУЛЯ, START - ТОЧКА

ВХОДА