

**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. 8382

Щемель Д.А.

Преподаватель

Ефремов М.А.

Санкт-Петербург

2020

## Цель работы

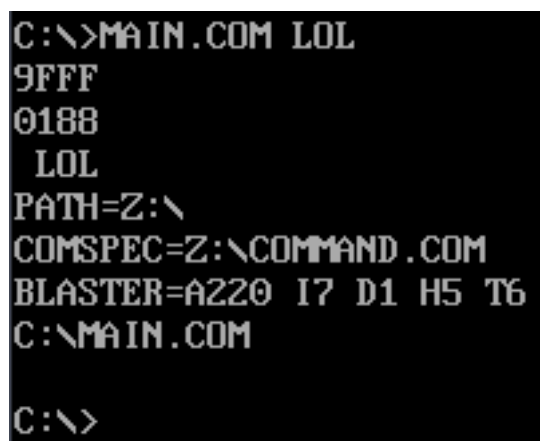
Исследование интерфейса управляющей программы и загрузочных модулей. Этот интерфейс состоит в передаче запускаемой программе управляющего блока, содержащего адреса и системные данные. Так загрузчик стоит префикс сегмента программы (PSP) и помещает его адрес в сегментный регистр. Исследование префикса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

## Ход выполнения работы

Был написан исходный код для .COM-модуля, который выводит следующую информацию:

1. Сегментный адрес недоступной памяти
2. Сегментный адрес среды, передаваемой программе
3. Хвост командной строки
4. Содержимое области среды
5. Путь загружаемого модуля

Результат работы модуля приведён на скриншоте ниже.



```
C:\>MAIN.COM LOL
9FFF
0188
LOL
PATH=Z:\
COMSPEC=Z:\COMMAND.COM
BLASTER=A220 I7 D1 H5 T6
C:\MAIN.COM
C:\>
```

Figure 1: Результат работы программы

## Контрольные вопросы

### Сегментный адрес недоступной памяти

1. На какую область памяти указывает адрес недоступной памяти?

На область памяти, следующей за областью, выделенной программе.

2. Где расположен этот адрес по отношению области памяти, отведённой программе?

За областью, выделенной программе.

3. Можно ли в эту область памяти писать?

Да, потому что нет никаких ограничений.

### **Среда, передаваемая программе**

1. Что такое среда?

Набор именованных переменных

2. Когда создаётся среда? Перед запуском приложения или в другой момент?

Во время загрузки ОС (в случае DOS).

3. Откуда берётся информация, записываемая в среду?

Путём вызова SET NAME=VALUE

### **Вывод**

В ходе выполнения лабораторной работы был изучен интерфейс управляющей программы и загрузочных модулей. А так же PSP и env.

## ПРИЛОЖЕНИЕ А. ИСХОДНЫЙ КОД ПРОГРАММЫ

PSPRESEARCH SEGMENT

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

ORG 100H

START: JMP MAIN

START\_FORBIDEN db '0000', 10, 13, '\$'

ENV\_ADDRESS db '0000', 10, 13, '\$'

CRLF db 10, 13, '\$'

EXIT PROC near

xor AL, AL

mov AH, 4ch

int 21h

ret

EXIT ENDP

PRINT PROC near

push ax

mov ah, 09h

int 21h

pop ax

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

```
    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

BYTE\_TO\_DEC PROC near

```
    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
```

```
MAIN:
```

```
    mov dx, es:[2h]
    mov al, dh
    call BYTE_TO_HEX
    mov si, offset START_FORBIDEN
    mov [si], ax
    mov si, offset START_FORBIDEN
    mov al, dl
    call BYTE_TO_HEX
    mov [si+2], ax
    mov dx, offset START_FORBIDEN
    call PRINT

    mov dx, es:[2ch]
    mov al, dh
    call BYTE_TO_HEX
    mov si, offset ENV_ADDRESS
    mov [si], ax
    mov si, offset ENV_ADDRESS
    mov al, dl
    call BYTE_TO_HEX
    mov [si+2], ax
    mov dx, offset ENV_ADDRESS
    call PRINT

    mov cl, es:[80h]
    mov si, 81h
    mov ah, 2h
    cmp cl, 0
    je FINISH_PRINT_NEXT_CHAR_FROM_CL
PRINT_NEXT_CHAR_FROM_CL:
    mov dl, [si]
    int 21h
```

```

        inc si

        loop PRINT_NEXT_CHAR_FROM_CL
FINISH_PRINT_NEXT_CHAR_FROM_CL:
        mov dx, offset CRLF
        call PRINT

mov es, es:[2ch]
mov dl, es:[0]
mov si, 0
PRINT_NEXT_CHAR_FROM_ENV:
        int 21h
        inc si
        mov dl, es:[si]
        cmp dl, 0
        je PRINT_NEXT_LINE
        jmp PRINT_NEXT_CHAR_FROM_ENV
PRINT_NEXT_LINE:
        mov dx, offset CRLF
        call PRINT
        inc si
        mov dl, es:[si]
        cmp dl, 0
        je FINISH_PRINT_ENV
        jmp PRINT_NEXT_CHAR_FROM_ENV
FINISH_PRINT_ENV:
add si, 3
mov dl, es:[si]
PRINT_NEXT_CHAR_FROM_PATH:
        int 21h
        inc si
        mov dl, es:[si]
        cmp dl, 0
        je FINISH_PRINT_NEXT_CHAR_FROM_PATH
        jmp PRINT_NEXT_CHAR_FROM_PATH
FINISH_PRINT_NEXT_CHAR_FROM_PATH:

```

```
mov dx, offset CRLF
```

```
call PRINT
```

```
call EXIT
```

```
PSPRESEARCH ENDS
```

```
END START
```