

**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. 0382

Гудов Н.Р.

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

### **Выполнение работы.**

При выполнении работы была разработан программный модуль типа **.com**, содержащий следующие процедуры:

Print

Prnt\_Unavailable – Выводит сегментный адрес недоступной памяти, взятый из PSP, в шестнадцатеричном виде.

Prnt\_Env\_Address- Выводит сегментный адрес среды, передаваемой программе, в шестнадцатеричном виде.

Prnt\_Input\_String-Выводит хвост командной строки в символьном виде,

предварительно считывая их количество.

Prnt\_String – Выводит содержимое области среды в символьном виде.

Prnt\_Environment\_Content\_And\_Path – Выводит содержимое области среды и путь содержимого модуля.

### **Контрольные вопросы.**

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

1) На какую область памяти указывает адрес недоступной памяти?

На сегментный адрес основной оперативной памяти, расположенной после программы, т. е. на первый сегмент после памяти, выделенной программе.

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

Сегментный адрес расположен в PSP по адресу 2Ch.

3) Можно ли в эту область памяти писать?

Можно, т. к. в DOS общее адресное пространство.

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

1) Что такое среда?

Среда представляет собой область памяти, в которой в виде символьных строк записаны значения переменных, называемых переменными среды

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

Среда создается при загрузке ОС, но может меняться перед запуском приложения, в соответствии с его требованиями.

3) Откуда берется информация, записываемая в среду?

Из системного пакетного файла AUTOEXEC.BAT, в котором содержатся ключевые переменных окружения, которые учувствуют в создании окружения в простейшем случае.

## **Выводы.**

Был исследован интерфейс управляющей программы и загрузочных модулей. Реализована программа печатающая сегментный адрес недоступной памяти, сегментный адрес среды, хвост командной строки, содержимое области среды и путь загружаемого модуля.

## ПРИЛОЖЕНИЕ А

### ИСХОДНЫЙ КОД ПРОГРАММЫ

```
PC Segment
    Assume CS:PC, DS:PC,
ES:NOTHING, SS:NOTHING
    ORG 100H

START: JMP BEGIN

;ДАННЫЕ

Unavailable_Memory_Msg db 'Unavailable
memory address:      ', 0ah, '$'
Segment_Env_Addres_Msg db 'Environment
address:             ', 0ah, '$'
Input_String db 'Input string:', '$'

;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 шест. 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_l
    or AL,30h
    mov [SI],AL
end_l:
    pop DX
    pop CX
    ret

```

BYTE\_TO\_DEC ENDP

print PROC NEAR

```

    push ax
    mov ah, 09h
    int 21h
    pop ax
    ret

```

print ENDP

Prnt\_Unavailable proc near

```

    push ax
    push di
    push dx

    mov ax, es:[02h]
    mov di, offset Unavailable_Memory_Msg
    add di, 31
    call wrd_to_hex
    mov dx, offset Unavailable_Memory_Msg
    call print

    pop dx
    pop di

```

```

    pop ax
    ret
Prnt_Unavailable endp

Prnt_Env_Address proc near
    push ax
    push di
    push dx

    mov ax, es:[02Ch]
    mov di, offset Segment_Env_Addres_Msg
    add di, 24
    call wrd_to_hex
    mov dx, offset Segment_Env_Addres_Msg
    call print

    pop dx
    pop di
    pop ax
    ret
Prnt_Env_Address endp

Prnt_Input_String proc near
    push dx
    push cx
    push si
    push ax

    mov dx, offset Input_String
    call print
    mov cl, ds:[80h]
    mov si, 081h
    mov ah, 02h
    cmp cl, 0
    je end_

    print_symbol:
        mov dl, [si]
        int 21h
        inc si
        loop print_symbol

    end_:
        mov dl, 0ah
        int 21h

    pop ax
    pop si
    pop cx
    pop dx

    ret
Prnt_Input_String endp

Prnt_String proc near

```

```

push dx
push ax
mov ah, 02h

print_sym:
    mov dl, ds:[si]
    inc si

    cmp dl, 0
    jz end_of_string

    int 21h
    jmp print_sym

end_of_string:
    mov dl, 0ah
    int 21h

pop ax
pop dx
ret
Prnt_String endp

Prnt_Environment_Content_And_Path proc
near
    push ds
    push si
    push ax
    push cx
    push dx

    mov ds, es:[2ch]
    mov si, 0

    Prnt_Strings:
        call Prnt_String
        mov bl, ds:[si]
        cmp bl, 0
        jz print_path
        jmp Prnt_Strings

    print_path:
        add si, 3
        call Prnt_String

    the_end:
        pop dx
        pop cx
        pop ax
        pop si
        pop ds

    ret

Prnt_Environment_Content_And_Path endp

BEGIN:

```



```
    call Prnt_Unavailable
    call Prnt_Env_Address
    call Prnt_Input_String
    call
Prnt_Environment_Content_And_Path
```

```
    xor al, al
    mov ah, 4Ch
    int 21H
```

```
PC    ENDS
      END      START
```