

**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. 9382

\_\_\_\_\_

Павлов Р.В.

Преподаватель

\_\_\_\_\_

Ефремов М.А.

Санкт-Петербург

2021

## Постановка задачи.

Цель работы: исследование интерфейса управляющей программы и загрузочных модулей, исследование префикса сегмента программы (PSP) и среды, передаваемой программе.

Функции и структуры данных:

Название процедуры	Описание процедуры
main	Вызов реализованных процедур, вывод на экран типа ПК
pctype	Формирует строку с указанием типа ПК, помещает в регистр DX её смещение
type_sys_info	Выводит на экран информацию о текущей версии ОС
byte_to_dec	Записывает в строку число размером 1 байт в десятичной системе по указанному в DX адресу
byte_to_hex	Записывает в строку число размером 1 байт в шестнадцатеричной системе по указанному в DX адресу

Последовательность действий:

1. Написание программного модуля .COM, который выводит следующую информацию:
  - 1) Сегментный адрес недоступной памяти 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 работы.

- 1) Взяты сегментные адреса недоступной памяти и среды, передаваемой программе, каждый переведён в символьный вид и занесён в соответствующую строку процедурой **word\_to\_hex**.
- 2) Реализована процедура **print\_tail**, при помощи цикла выводящая хвост консоли в символьном виде.
- 3) Реализована процедура **print\_envir\_path**, которая посимвольно выводит содержимое среды, а затем и путь загружаемого модуля. Концом строк в последовательности служат нулевые байты.
- 4) Программа отлажена, скомпилирована и запущена. Результаты её выполнения представлены ниже.

```
C:\>lab2
Адрес недоступной памяти: 9FFF

Адрес среды: 0188

Хвост консоли: пуст, нет аргументов.

Содержимое среды:
PATH=Z:\
COMSPEC=Z:\COMMAND.COM
BLASTER=A220 I7 D1 H5 T6

Путь:
C:\LAB2.COM

C:\>
```

```
C:\>lab2 tail
Адрес недоступной памяти: 9FFF

Адрес среды: 0188

Хвост консоли: tail

Содержимое среды:
PATH=Z:\
COMSPEC=Z:\COMMAND.COM
BLASTER=A220 I7 D1 H5 T6

Путь:
C:\LAB2.COM

C:\>_
```

## **Ответы на контрольные вопросы.**

### **Сегментный адрес недоступной памяти**

1. На первый и последующие байты памяти, недоступные программе.
2. Она находится за блоком памяти, выделенным программе.
3. Туда можно что-то записать, но делать этого не стоит.

### **Среда, передаваемая программе**

1. Среда — это область памяти, в которой хранится массив строк вида «имя = значение» (переменных среды), которые служат вызываемой программе параметрами.
2. Среда создаётся после загрузки программы в память, но до передачи управления этой программе.
3. Переменные среды берутся из командного файла **autoexec.bat**, который также выполняется при запуске ОС и отвечает за настройку параметров среды.

## **Выводы.**

В результате выполнения лабораторной работы был исследован интерфейс управляющей программы и загрузочных модулей, изучены некоторые фрагменты структуры PSP, исследована среда программы.

## ПРИЛОЖЕНИЕ А. ИСХОДНЫЙ КОД

- имя файла : lab2.asm

```
.model tiny
.code
    org 100h

    main proc far
        xor ax, ax
        push ds
        push ax

        mov ax, ds:[2]
        mov dx, offset unav_mem + 26      ; запись адреса недоступной
        call word_to_hex                  ; памяти в строку, пе-
чатать
        sub dx, 26
        call print_string

        mov ax, ds:[2Ch]
        mov dx, offset envir_address + 13 ; запись адреса среды
        call word_to_hex                  ; в строку, печать
        sub dx, 13
        call print_string

        call print_tail                  ; печать информа-
ции о хвосте консоли

        call print_envir_path            ; печать информации о
среде и пути

        mov dl, 13
        int 21h                          ; возврат
каретки, завершение
        retf
    main endp

    print_string proc near                ; процедура печати
строки
        push ax                          ; (в dx за-
ранее помещён адрес)
        mov ah, 9
        int 21h
        pop ax
        retn
    print_string endp

    byte_to_dec proc near                 ; перевод байтового
значения в 10 CC
        push ax
        push bx
        push cx

        xor ah, ah
```

```

        mov bx, 10                                ; подготовка дан-
ных (bx - делитель)
        xor si, si
        add si, 1

        mov cx, 2
        c1:
            div bl
            add ah, '0'
            push bx
            mov bx, dx
            mov [bx + si], ah                      ; цикл записи числа с
конца
            pop bx
            dec si
            xor ah, ah
            loop c1

        pop cx
        pop bx
        pop ax
        retn
byte_to_dec endp

        byte_to_hex proc near                      ; перевод байтового
значения в 16 CC
        push ax
        push bx
        push cx

        xor ah, ah
        mov bx, 16
        xor si, si
        add si, 1

        mov cx, 2
        c2:
            div bl
            cmp ah, 10
            jl digit1
            add ah, 7
            digit1:
                add ah, '0'                        ; аналогично пре-
дыдущему, но
                push bx                            ; учитыва-
ется возможность появления символов
                mov bx, dx                          ; латинского ал-
фавита
                mov [bx + si], ah
                pop bx
                dec si
                xor ah, ah
                loop c2

        pop cx
        pop bx
        pop ax
        retn
byte_to_hex endp

        word_to_hex proc near                      ; перевод слова в 16 CC

```

```

        push ax
        push bx
        push cx
        push dx

        xor dx, dx
        xor si, si
        add si, 3

        mov cx, 4                                ; 4 повторения
цикла, в отличие от предыдущего
        c3:
            mov bx, 16
            div bx
            cmp dl, 10
            jnl digit2
            add dl, 7
            digit2:
                add dl, '0'
            pop bx
            mov [bx + si], dl
            push bx
            dec si
            xor dx, dx
            loop c3

        pop dx
        pop cx
        pop bx
        pop ax
        retn
word_to_hex endp

        print_tail proc near                      ; вывод информации о
хвосте консоли
            mov dx, offset tail_info
            call print_string

            mov di, 80h
            xor cx, cx
            mov cl, [di]                          ; получение кол-
ва символов
            mov ah, 2                              ; и установка в
режим посимвольного вывода
            cmp cl, 0
            je void
            inc di
            tail:
                mov dl, [di]                      ; цикл работает,
если хвост не пуст
                int 21h
                inc di
                loop tail
            jmp escape_the_void

        void:
            mov dx, offset no_tail                ; если же пуст - выво-
дится сообщение об этом
            call print_string

```

```

        escape_the_void:                                ; в обоих случаях два-
жды печатается                                     ; перенос на но-
        mov dl, 10
вую строку
        int 21h
        int 21h

        retn
print_tail endp

print_envir_path proc near                             ; вывод информации о
среде и пути
        push ds

        mov dx, offset envir_content
        call print_string

        xor di, di
        mov ax, ds:[2Ch]                               ; переход на другой
сегментный адрес
        mov ds, ax
        mov ah, 2                                       ; и установка в
режим посимвольного вывода
        whilee:
            mov dl, [di]
            cmp dl, 0
            je zero                                     ; печать,
пока не встречен 0-байт (конец строки)
            int 21h
            inc di
            jmp whilee

        zero:
            inc di
            mov dl, 10
            int 21h
            mov dl, [di]
            cmp dl, 0                                  ; если и после
конец строки 0-байт
            jne whilee                                 ; то конец инфор-
мации о среде, иначе повторить

        mov dl, 10
        int 21h
        add di, 3
        push ds
        push ax
        mov ax, es
        mov ds, ax
        mov dx, offset path_info                       ; вывод строки с
обозначением пути
        call print_string
        pop ax
        pop ds

        path:
            mov dl, [di]
            cmp dl, 0
            je stop                                     ; также по-
символьный вывод информации

```



```

                                int 21h                                ; о пути
пока не встречен 0-байт
                                inc di
                                jmp path

                                stop:
вую строку                      mov dl, 10                        ; перенос на но-
                                int 21h

                                pop ds
                                retn
print_envir_path endp

unav_mem db 'Адрес недоступной памяти: ', 4 dup(?), 13, 10, 10, '$'
envir_adress db 'Адрес среды: ', 4 dup(?), 13, 10, 10, '$'
tail_info db 'Хвост консоли:$'
no_tail db ' пуст, нет аргументов.$'
envir_content db 'Содержимое среды:', 10, '$'
path_info db 'Путь:', 10, '$'

end main

```