МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра математического обеспечения и применения ЭВМ

ОТЧЕТ

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

Студент гр. 8382 _____ Янкин Д.О.

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 строит префикс сегмента программы (PSP) и помещает его адрес в сегментный регистр. Исследование префикса сегмента программы (PSP) и среды, передаваемой программе.

Ход работы.

Был написан код программы и получен .COM модуль, который печатает информацию о:

1) Сегментном адресе недоступной памяти;
2) Сегментном адресе среды;
3) Хвосте командной строки;
4) Содержимом области среды;
5) Пути загружаемого модуля.

Рисунок 1. Результат работы программы

Контрольные вопросы.

Сегментный адрес недоступной памяти:

1) На какую область памяти указывает адрес недоступной памяти?

   На память, которую программа не должна модифицировать в ходе работы.

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

   Сразу за областью памяти, выделенной программе.

3) Можно ли в эту область памяти писать?

   Можно, но не нужно.

Среда, передаваемая программе:

1) Что такое среда?

   Набор строк вида ИМЯ=ПАРАМЕТР, содержащих какую-либо информацию, в том числе о настройках операционной системы.

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

   Окружение создается при загрузке операционной системы.

3) Откуда берётся информация, записываемая в среду?

   При загрузке программы выделяется область памяти для среды, содержимое которой копируется из среды родительского процесса.

Выводы.

В ходе работы был изучен интерфейс управляющей программы и загрузочных модулей, префикс сегмента программы и среда, передаваемая программе.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ LABASM

```
TESTPC SEGMENT
     ASSUME CS:TESTPC, DS:TESTPC, ES:NOTHING, SS:NOTHING
     ORG 100H

START: JMP BEGIN

; ДАННЫЕ
INACCESSIBLE_MEMORY_MESSAGE          db 'Inaccessible memory: ',
'$'
INACCESSIBLE_MEMORY                  db '0000', 13, 10, '$'

ENVIROMENT_ADRESS_MESSAGE       db 'Enviroment adress: ', '$'
ENVIROMENT_ADRESS                    db '0000', 13, 10, '$'

COMMAND_LINE_TAIL_MESSAGE       db 'Command line tail: ', '$'

ENVIROMENT_MESSAGE                   db 'Enviroment: ', '$'

PATH_MESSAGE                         db 'Path: ', '$'

; ПРОЦЕДУРЫ
;-------------------------------------------------
TETR_TO_HEX PROC near
; младшая шестн. цифра AL в шестн. цифру ASCII
           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 шестн. числа ASCII 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
           ;xchg al, ah          ;; а теперь наоборот!
           ret
BYTE_TO_HEX ENDP

;-------------------------------
WRD_TO_HEX PROC near
; перевод в 16 с/с 16-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 10 с/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
           dec si
end_l:     pop DX
           pop CX
           ret
BYTE_TO_DEC ENDP



;-------------------------------------------------
PRINT_STRING PROC near
; Просто выводит строку с уже указанным в dx смещением, очень
сложная функция
           push ax
```

```
            mov ah, 09h
            int 21h

            pop ax
            ret
PRINT_STRING ENDP

;-------------------------------------------------
PRINT_WORD PROC near
; Выводит регистр AX
            push ax
            push dx

            mov dl, ah
            mov ah, 02h
            int 21h

            mov dl, al
            int 21h

            pop dx
            pop ax
            ret
PRINT_WORD ENDP

;-------------------------------------------------
PRINT_ENDL PROC near
; Выводит 13, 10
            push ax
            push dx

            mov dl, 13
            mov ah, 02h
            int 21h
```

```
            mov dl, 10
            int 21h

            pop dx
            pop ax
            ret
PRINT_ENDL ENDP



;-------------------------------
; КОД
BEGIN:
            ; Недоступная память
            mov dx, offset INACCESSIBLE_MEMORY_MESSAGE
            call PRINT_STRING

            mov bx, 2h
            mov ax, [bx]

            mov di, offset INACCESSIBLE_MEMORY
            add di, 3
            call WRD_TO_HEX

            mov dx, offset INACCESSIBLE_MEMORY
            call PRINT_STRING


            ; Сегментный адрес среды, передаваемый программе
            mov dx, offset ENVIROMENT_ADRESS_MESSAGE
            call PRINT_STRING

            mov bx, 2Ch
```

```
            mov ax, [bx]

            mov di, offset ENVIROMENT_ADRESS
            add di, 3
            call WRD_TO_HEX

            mov dx, offset ENVIROMENT_ADRESS
            call PRINT_STRING


            ; Хвост командной строки
            mov dx, offset COMMAND_LINE_TAIL_MESSAGE
            call PRINT_STRING

            mov bx, 80h
            mov ch, 0
            mov cl, [bx]
            cmp cx, 0
            je COMMAND_LINE_TAIL_END

            mov bx, 81h
            mov ah, 02h

COMMAND_LINE_TAIL_LOOP:
            mov dl, [bx]
            int 21h
            add bx, 1
            loop COMMAND_LINE_TAIL_LOOP
COMMAND_LINE_TAIL_END:
            call PRINT_ENDL


            ; Область среды
            mov dx, offset ENVIROMENT_MESSAGE
```

```
            call PRINT_STRING

            mov bx, 2Ch
            mov es, [bx]
            mov bx, 0

            mov ah, 02h
ENVIROMENT_LOOP:
            mov dl, 0
            cmp dl, es:[bx]
            je ENVIROMENT_END
ENVIROMENT_VARIABLE_LOOP:
            mov dl, es:[bx]
            int 21h
            add bx, 1
            cmp dl, 0
            jne ENVIROMENT_VARIABLE_LOOP
            je ENVIROMENT_LOOP
ENVIROMENT_END:
            call PRINT_ENDL


            ; Путь загружаемого модуля
            mov dx, offset PATH_MESSAGE
            call PRINT_STRING

            add bx, 3
PATH_LOOP:
            mov dl, es:[bx]
            int 21h
            add bx, 1
            cmp dl, 0
            jne PATH_LOOP
            call PRINT_ENDL
```

```
            ret

TESTPC      ENDS
            END START ; конец модуля, START – точка входа
```