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 _____ Черницын П.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 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.

```
C:\>LR2.COM qwerty
Inaccessible memory: 9FFF
Enviroment adress: 0188
Command line tail:  qwerty
Enviroment: PATH=Z:\ COMSPEC=Z:\COMMAND.COM BLASTER=A220 I7 D1 H5 T6
Path: C:\LR2.COM
```

Рисунок 1. Результат работы программы

Контрольные вопросы.

Сегментный адрес недоступной памяти:

1) На какую область памяти указывает адрес недоступной памяти?

   На память, в которой находится транзитная часть командного процессора COMMAND.COM, которая содержит исполнитель внутренних команд и загрузчик программ в оперативную память.

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?

   DOS не проверяет изменение памяти из других процессов, нет защиты памяти, поэтому писать можно. Запись в эту область памяти чревата неправильной работе программы.

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
;DATA
INACCESSIBLE_MEMORY_MSG db 'Inaccessible memory: ', '$'
INACCESSIBLE_MEMORY		db '0000', 13, 10, '$'

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

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

ENVIROMENT_MSG				db 'Enviroment: ', '$'

PATH_MESSAGE			db 'Path: ', '$'

ENDL					db 13, 10, '$'
;--------------------------------------------------------
WriteMsg PROC near
			push ax
	mov ah,09h
	int 21h
			pop ax
	ret
WriteMsg ENDP
;--------------------------------------------------------
TETR_TO_HEX PROC near
	and al,0Fh
	cmp al,09
	jbe NEXT
	add al,07
NEXT:   add al,30h ; код нуля
	ret
TETR_TO_HEX ENDP
;--------------------------------------------------------
BYTE_TO_HEX PROC near
;байт в al переводится в два символа в шест. сс ax
	push cx
	mov ah,al
	call TETR_TO_HEX
	xchg al,ah
	mov cl,4
	shr al,cl
	call TETR_TO_HEX ;al - старшая
	pop cx ;ah - младшая цифра
	ret
BYTE_TO_HEX ENDP
;--------------------------------------------------------
WRD_TO_HEX PROC near
; перевод в 16сс 16ти разрядного числа
; 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
;--------------------------------------------------
BEGIN:
                ;недоступная память
                mov dx, offset INACCESSIBLE_MEMORY_MSG
                call WriteMsg

                mov bx, 2h
                mov ax, [bx]

                mov di, offset INACCESSIBLE_MEMORY
                add di, 3
                call WRD_TO_HEX
                mov dx, offset INACCESSIBLE_MEMORY
                call WriteMsg

                ;адрес среды
                mov dx, offset ENVIROMENT_ADRESS_MSG
                call WriteMsg

                mov bx, 2ch
                mov ax, [bx]

                mov di, offset ENVIROMENT_ADRESS
                add di, 3
                call WRD_TO_HEX
                mov dx, offset ENVIROMENT_ADRESS
                call WriteMsg

                ;хвост ком строки
                mov dx, offset COMMAND_LINE_TAIL_MSG
                call WriteMsg

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

                mov bx, 81h
                mov ah, 02h

CMD_TAIL_LOOP:
                mov dl, [bx]
                int 21h
                inc bx
                loop CMD_TAIL_LOOP

CMD_TAIL_END:
                mov dx, offset ENDL
                call WriteMsg
```

```
                ;Область памяти
                mov dx, offset ENVIROMENT_MSG
                call WriteMsg

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

                mov ah, 02h
                xor dl, dl

ENVIROMENT_LOOP:
                cmp dl, es:[bx]
                je ENVIROMENT_END

ENVIROMENT_VARIABLE_LOOP:
                mov dl, es:[bx]
                int 21h
                inc bx
                cmp dl, 0
                jne ENVIROMENT_VARIABLE_LOOP
                je ENVIROMENT_LOOP

ENVIROMENT_END:
                mov dx, offset ENDL
                call WriteMsg

                ; Путь загружаемого модуля
                mov dx, offset PATH_MESSAGE
                call WriteMsg

                add bx, 3
PATH_LOOP:
                mov dl, es:[bx]
                int 21h
                inc bx
                cmp dl, 0
                jne PATH_LOOP
                mov dx, offset ENDL
                call WriteMsg


                xor al,al
                mov AH,4Ch
                int 21h
TESTPC ENDS
                END START;
```