

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

Колногоров Д.Г.

Преподаватель

Ефремов М.А.

Санкт-Петербург

2020

Цель работы.

Исследование интерфейса управляющей программы и загрузочных модулей, префикса сегмента программы и среды, передаваемой программе.

Выполнение работы.

Был написан программный модуль типа **.COM** (представлен в приложении А), который определяет и распечатывает 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.

Результат исполнения COM модуля представлен на рисунке 1.



```
C:\>LR2.COM
unavailable memory: 9FFF
environment memory: 0188
tail: no tail
environment variables:
PATH=Z:\
COMSPEC=Z:\COMMAND.COM
BLASTER=A220 I7 D1 H5 T6
LR=LR2
path: C:\LR2.COM EXE
```

Рисунок 1 — результат исполнения COM модуля

Сегментный адрес недоступной памяти.

- 1) На какую область памяти указывает адрес недоступной памяти?

Адрес недоступной памяти указывает на первый байт сегмента памяти, который находится за выделенной программе памятью.

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

Этот сегмент расположен сразу после выделенной программе памяти.

3) Можно ли в эту область памяти писать?

Можно.

Среда передаваемая программе.

1) Что такое среда?

Среда — совокупность системных переменных, которые передаются программе при её запуске.

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

При загрузке ОС.

3) Откуда берётся информация, записываемая в среду?

Программа наследует родительскую среду. Для этого при запуске программы выделяется отдельная область памяти, в которую копируется родительская среда.

Вывод.

В ходе выполнения лабораторной работы были исследованы интерфейс управляющей программы и загрузочных модулей, префикса сегмента программы и среды, передаваемой программе.

ПРИЛОЖЕНИЕ А

КОД ИСХОДНОГО СОМ МОДУЛЯ

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

START:       JMP      BEGIN
; data
NEW_LINE      db 10,13,'$'
STR_SEG_MEM    db 'unavailable memory: $'
STR_ENV_ADDR   db 'environment memory: $'
STR_TAIL       db 'tail: $'
STR_NO_TAIL    db 'no tail',10,13,'$'
STR_ENV_VARIABLES db 'environment variables:',10,13,'$'
STR_PATH       db 'path: $'

PRINT_NEW_LINE PROC near
    push DX
    push AX

    mov DX, offset NEW_LINE
    mov AH, 09h
    int 21h

    pop AX
    pop DX
    ret
PRINT_NEW_LINE ENDP

PRINT_BYTE     PROC near
; prints AL as two hex digits
    push BX
    push DX

    call BYTE_TO_HEX
    mov BH, AH

    mov DL, AL
    mov AH, 02h
    int 21h
```

```

        mov DL, BH
        mov AH, 02h
        int 21h

        pop DX
        pop BX
        ret
PRINT_BYTE    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
; AL --> two hex symbols in AX
        push    CX
        mov     AH,AL
        call    TETR_TO_HEX
        xchg    AL,AH
        mov     CL,4
        shr     AL,CL
        call    TETR_TO_HEX ; AL - high digit
        pop     CX          ; AH - low digit
        ret
BYTE_TO_HEX    ENDP
;-----
; CODE
BEGIN:

PRINT_SEG_MEM:
        mov DX, offset STR_SEG_MEM
        mov AH, 09h
        int 21h

        mov BX, DS:[02h]

```

```

    mov AL, BH
    call PRINT_BYTE
    mov AL, BL
    call PRINT_BYTE

    call PRINT_NEW_LINE
PRINT_ENV_ADDR:
    mov DX, offset STR_ENV_ADDR
    mov AH, 09h
    int 21h

    mov BX, DS:[2Ch]
    mov AL, BH
    call PRINT_BYTE
    mov AL, BL
    call PRINT_BYTE

    call PRINT_NEW_LINE
PRINT_TAIL:
    mov DX, offset STR_TAIL
    mov AH, 09h
    int 21h

    mov CH, 0
    mov CL, DS:[80H]
    cmp CL, 0
    je no_tail

    mov BX, 0
tail_loop:
    mov DL, DS:[81H+BX]
    mov AH, 02H
    int 21h

    inc BX
    loop tail_loop

    call PRINT_NEW_LINE
    jmp tail_end
no_tail:
    mov DX, offset STR_NO_TAIL

```

```
mov AH, 09h
int 21h
tail_end:
```

PRINT_ENV_CONTENTS:

```
mov DX, offset STR_ENV_VARIABLES
mov AH, 09h
int 21h
```

```
mov ES, DS:[2Ch]
```

```
mov BX, 0
```

```
print_env_variable:
```

```
mov DL, ES:[BX]
```

```
cmp DL, 0
```

```
je variable_end
```

```
mov AH, 02h
```

```
int 21h
```

```
inc BX
```

```
jmp print_env_variable
```

```
variable_end:
```

```
call PRINT_NEW_LINE
```

```
inc BX
```

```
mov DL, [BX+1]
```

```
cmp DL, 0
```

```
jne print_env_variable
```

PRINT_MODULE_PATH:

```
mov DX, offset STR_PATH
```

```
mov AH, 09h
```

```
int 21h
```

```
add BX, 2
```

```
path_loop:
```

```
mov DL, ES:[BX]
```

```
cmp DL, 0
```

```
jne path_next
```

```
cmp byte ptr ES:[BX+1], 0
```

```
je loop_end
```

```
path_next:
```

```

        mov AH, 02H
        int 21h
        inc BX
        jmp path_loop
loop_end:

; return to DOS
        xor     AL,AL
        mov     AH,4Ch
        int     21H
TESTPC  ENDS
        END     START      ; module end START - entry point

```