

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

Афанасьев Н. С.

Преподаватели

Ефремов М.А.

Санкт-Петербург

2022

Цель работы.

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

Задание.

Написать и отладить программный модуль типа .COM, который выбирает и распечатывает определённую информацию из PSP.

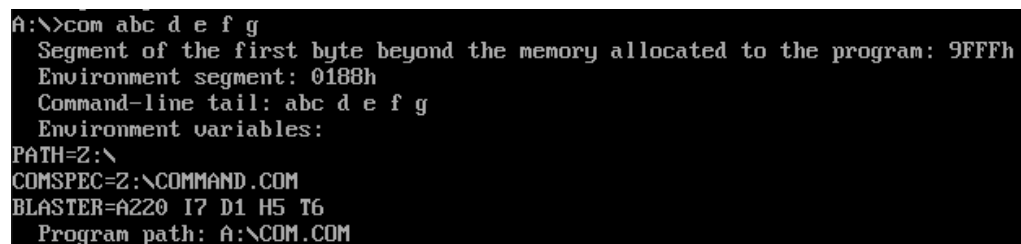
Выполнение работы.

При работе были использованы/созданы следующие процедуры:

- TETR_TO_HEX, BYTE_TO_HEX, WRD_TO_HEX – процедуры, описанные в шаблоне, для перевода двоичных кодов в символы шестнадцатеричных чисел.

- PRINT – процедура для вывода строки, отступ на которую содержится в DX, на экран, используя функцию 09h прерывания 21h.

- PRINT_INFO – процедура для вывода необходимой информации из PSP. Сначала считывается сегментный адрес недоступной памяти и выводится на экран в 16-тиричном виде. Далее считывается сегментный адрес среды, передаваемой программе и так же выводится на экран в 16-тиричном виде. Далее посимвольно считывается и выводится хвост командной строки. Далее посимвольно выводится содержимое области среды, каждая новая переменная среды начинается с новой строки. Наконец, посимвольно считывается и выводится путь загружаемого модуля. По итогу, вся информация будет на экране (рис.1).



```
A:\>com abc d e f g
Segment of the first byte beyond the memory allocated to the program: 9FFFh
Environment segment: 0188h
Command-line tail: abc d e f g
Environment variables:
PATH=Z:\
COMSPEC=Z:\COMMAND.COM
BLASTER=A220 I7 D1 H5 T6
Program path: A:\COM.COM
```

Рис.1 – Пример выполнения программы

Программный код см. в Приложении А

Вопросы.

Сегментный адрес недоступной памяти

- 1) На какую область памяти указывает адрес недоступной памяти?
 - На сегмент оперативной памяти, находящийся за пределами памяти, выделенной программе.
- 2) Где расположен этот адрес по отношению области памяти, отведенной программе?
 - Первый байт сразу после программы
- 3) Можно ли в эту область памяти писать?
 - Да, так как этому ничего не препятствует, нету никаких механизмов защиты памяти.

Среда, передаваемая программе

- 1) Что такое среда?
 - Область памяти, в которой записаны переменные среды (COMPSEC, PATH, PROMPT, SET, BLASTER и т.д.) со значениями.
- 2) Когда создается среда? Перед запуском приложения или в другое время?
 - Изначально среда создаётся при запуске ОС. При запуске какой-либо программы содержимое родительской среды копируется, и также родительская программа может эту среду дополнять.
- 3) Откуда берется информация, записываемая в среду?
 - Системные переменные среды в MS-DOS берутся из файла autoexec.bat при запуске ОС. Остальные переменные могут добавляться перед запуском программы в процессе работы родительских программ.

Выводы.

Был исследован интерфейс управляющей программы и загрузочного модуля, а также префикс сегмента программы (PSP) и среды, передаваемой программе.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЕ КОДЫ ПРОГРАММ

Название файла: com.asm

```
CODE SEGMENT
    ASSUME CS:CODE, DS:CODE, ES:NOTHING, SS:NOTHING
    ORG 100H
    START: jmp BEGIN

    MEM_SEG      DB "    Segment of the first byte beyond the memory
allocated to the program:      h",0DH,0AH,'$'
    ENV_SEG      DB "    Environment segment:      h",0DH,0AH,'$'
    CMD_TAIL     DB "    Command-line tail:","'$'
    ENV_VARS     DB "    Environment variables: ",0DH,0AH,'$'
    PR_PATH      DB "    Program path: ","'$'
    NEWLINE      DB 0DH,0AH,'$'

    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
        push CX
        mov AH,AL
        call TETR_TO_HEX
        xchg AL,AH
        mov CL,4
        shr AL,CL
        call TETR_TO_HEX
        pop CX
        ret
    BYTE_TO_HEX ENDP

    WRD_TO_HEX PROC NEAR
        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
```

```

PRINT PROC NEAR
    push AX
    mov AH, 09h
    int 21h
    pop AX
    ret
PRINT ENDP

PRINT_INFO PROC NEAR
    push AX
    push CX
    push DX
    push DI
    push ES
;External memory segment
    mov DX, offset MEM_SEG
    mov DI, DX
    add DI, 75
    mov AX, CS:[2]
    call WRD_TO_HEX
    call PRINT
;Environment segment
    mov DX, offset ENV_SEG
    mov DI, DX
    add DI, 26
    mov AX, CS:[2Ch]
    call WRD_TO_HEX
    call PRINT
;Command-line tail
    mov DX, offset CMD_TAIL
    call PRINT
    xor CX,CX
    mov CL, CS:[80h]
    cmp CL, 0
    mov AH, 02h
    je lend
    mov DI, 81h
    lstart:
        mov DL, CS:[DI]
        int 21h
        inc DI
        loop lstart
    lend:
    mov DX, offset NEWLINE
    call PRINT
;Environment variables
    mov DX, offset ENV_VARS
    call PRINT
    mov DX, CS:[2Ch]
    mov ES, DX
    mov DI, 0
    _next:
        mov DL, ES:[DI]
    _print:
        int 21h
        inc DI
        cmp DL, 0

```

```

        jne _next
        mov DX, offset NEWLINE
        call PRINT
        mov DL, ES:[DI]
        cmp DL, 0
        jne _print
;Program path
        mov DX, offset PR_PATH
        call PRINT
        add DI, 3
        __next:
            mov DL, ES:[DI]
            int 21h
            inc DI
            cmp DL, 0
            jne __next
        mov DX, offset NEWLINE
        call PRINT
        pop ES
        pop DI
        pop DX
        pop CX
        pop AX
        ret
PRINT_INFO ENDP

BEGIN:
        call PRINT_INFO
        xor AL,AL
        mov AH,4Ch
        int 21H
CODE ENDS
END START

```