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

ОТЧЕТ

по практической работе № 2

по дисциплине «Операционные системы»

Тема: Исследование интерфейсов программных модулей

Студент гр. 9382 _____ Русинов Д.А.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург

2021

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.

Задание.

Шаг 1. Для выполнения лабораторной работы необходимо написать и отладить программный модуль типа .COM, который выбирает и распечатывает следующую информацию:

1) Сегментный адрес недоступной памяти, взятый из PSP, 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.

Сохраните результаты, полученные программой, и включите их в отчет.

Шаг 2. Оформление отчета в соответствии с требованиями. В отчет включите скриншот с запуском программы и результатами.

Выполнение работы.

Была разработана .COM-программа, которая выполняет задание. Результаты выполнения программы приведены для запуска модуля с аргументами и без.

```
F:\MASM>LAB2COM.COM
Unavailable memory:  9FFF
Environment address:  0188
Command tail is: Empty
Environment content: PATH=Z:\
                     COMSPEC=Z:\COMMAND.COM
                     BLASTER=A220 I7 D1 H5 T6
Path is: F:\MASM\LAB2COM.COM
```

```
F:\MASM>LAB2COM.COM SUPER+DEFAULT+ARGUMENT
Unavailable memory:  9FFF
Environment address:  0188
Command tail is:  SUPER+DEFAULT+ARGUMENT
Environment content: PATH=Z:\
                     COMSPEC=Z:\COMMAND.COM
                     BLASTER=A220 I7 D1 H5 T6
Path is: F:\MASM\LAB2COM.COM
```

Ответы на вопросы.

Сегментный адрес недоступной памяти

1) На какую область памяти указывает адрес недоступной памяти?

На первый байт после участка памяти, отведенного под программу

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

Адрес недоступной памяти по отношению к области памяти, отведенной программе, расположен после. В сторону увеличения адресов.

3) Можно ли в эту область памяти писать?

Можно, так как DOS не имеет механизмов защиты перезаписи памяти программами, для которых эта память не выделялась.

Среда передаваемая программе

1) Что такое среда?

Участок памяти, который содержит переменные среды. Переменные среды хранят некоторую информацию о состоянии системы.

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

   Среда создается при запуске ОС. Среда копируется в адресное пространство запущенной программы. Среда может изменяться в соответствии с требованиями программы.

3) Откуда берется информация, записываемая в среду?

   Из файла AUTOEXEC.BAT. Он расположен в корневом каталоге загрузочного устройства.

Выводы.

Был исследован интерфейс управляющей программы и загрузочных модулей. Был исследован префикс сегмента программы (PSP) и среды, передаваемой программе.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Файл: LAB2COM.ASM

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

STARTUP: JMP MAIN


ENV_ADDR         DB      'Environment address:',                  '$'
UNAVAILABLE      DB      'Unavailable memory:',                   '$'
AX_REGISTER      DB      '      ',                        0DH, 0AH, '$'
EMPTY            DB      'Empty',                         0DH, 0AH, '$'
TAIL             DB      'Command tail is: ',                     '$'
ENV              DB      'Environment content: ',                 '$'
ENV_TAB          DB      '                     ',                 '$'
NEW_LINE         DB                                       0DH, 0AH, '$'
PATH             DB      'Path is: ',                             '$'



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


BYTE_TO_DEC PROC NEAR
    PUSH SI
    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

END_L:
    POP DX
    POP CX
```

```
    POP SI
    RET
BYTE_TO_DEC ENDP


WRITESTRING PROC NEAR
    PUSH AX
    MOV AH, 09H
    INT 21H
    POP AX
    RET
WRITESTRING ENDP


CLEARSTRING PROC NEAR
    PUSH SI
    PUSH CX

    XOR SI, SI
    MOV CX, 5

CLEARING:
    MOV [OFFSET AX_REGISTER+SI], 0FF20h
    INC SI
    LOOP CLEARING

    MOV [OFFSET AX_REGISTER+SI], 0FF20h
    POP CX
    POP SI
    RET
CLEARSTRING ENDP


_PRINT_INFO PROC NEAR
    CALL WRITESTRING
    MOV DI, OFFSET AX_REGISTER
    ADD DI, 5
    CALL WRD_TO_HEX
    MOV DX, OFFSET AX_REGISTER
    CALL WRITESTRING
    CALL CLEARSTRING
    RET
_PRINT_INFO ENDP


PRINT_UNAVAILABLE_MEMORY PROC NEAR
    MOV DX, OFFSET UNAVAILABLE
    MOV AX, DS:[02h]
```

```
    CALL _PRINT_INFO
    RET
PRINT_UNAVAILABLE_MEMORY ENDP


PRINT_ENV_ADDR PROC NEAR
    MOV DX, OFFSET ENV_ADDR
    MOV AX, DS:[2Ch]
    CALL _PRINT_INFO
    RET
PRINT_ENV_ADDR ENDP


PRINT_COMMAND_TAIL PROC NEAR
    PUSH CX
    PUSH AX
    XOR CX, CX

    MOV DX, OFFSET TAIL
    CALL WRITESTRING

    MOV CL, DS:[80h]
    CMP CL, 0
    JE _EMPTY_TAIL

    MOV SI, 0

_PRINT_TAIL_SYMBOL:
    MOV DL, DS:[81h+SI]
    MOV AH, 02h
    INT 21h
    INC SI
    LOOP _PRINT_TAIL_SYMBOL

    JMP _EXIT_TAIL_PRINT

_EMPTY_TAIL:
    MOV DX, OFFSET EMPTY
    CALL WRITESTRING

_EXIT_TAIL_PRINT:
    POP AX
    POP CX
    RET
PRINT_COMMAND_TAIL ENDP


PRINT_ENVIRONMENT PROC NEAR
```

```
    PUSH DX
    PUSH AX
    PUSH SI
    PUSH DS

    XOR SI, SI

    MOV DX, OFFSET ENV
    CALL WRITESTRING

    MOV DS, DS:[2CH]

_READ_ENV:
    MOV DL, [SI]
    CMP DL, 0
    JE _EOF

    MOV AH, 02h
    INT 21h

    INC SI
    JMP _READ_ENV

_EOF:
    INC SI
    MOV DL, [SI]
    CMP DL, 0
    JE _END_READING_ENV

    POP DS
    MOV DX, OFFSET NEW_LINE
    CALL WRITESTRING
    MOV DX, OFFSET ENV_TAB
    CALL WRITESTRING
    PUSH DS
    MOV DS, DS:[2Ch]

    JMP _READ_ENV

_END_READING_ENV:
    POP DS
    MOV DX, OFFSET NEW_LINE
    CALL WRITESTRING

    MOV DX, OFFSET PATH
    CALL WRITESTRING
    PUSH DS
    MOV DS, DS:[2Ch]
```

```
            ADD SI, 3

        _READING_PATH:
            MOV DL, [SI]
            CMP DL, 0
            JE _EXIT_PRINT_ENV

            MOV AH, 02h
            INT 21h
            INC SI
            JMP _READING_PATH


        _EXIT_PRINT_ENV:
            POP DS
            POP SI
            POP AX
            POP DX
            RET
        PRINT_ENVIRONMENT ENDP


        MAIN:
            CALL PRINT_UNAVAILABLE_MEMORY
            CALL PRINT_ENV_ADDR
            CALL PRINT_COMMAND_TAIL
            CALL PRINT_ENVIRONMENT
            XOR AL, AL
            MOV AH, 4CH
            INT 21H


        TESTPC ENDS
    END STARTUP
```