**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**

Студентка гр. 7381 Кревчик А.Б.

Преподаватель Ефремов М.А.

Санкт-Петербург

2019

**Цель работы.**

Исследование интерфейса управляющей программы и загрузочных модулей. Этот интерфейс состоит в передаче запускаемой программе управляющего блока, содержащего адреса и системный данные. Так загрузчик строит префикс сегмента программы (PSP) и помещает его адрес в сегментный регистр. Исследование префикса сегмента программы (PSP) и среды, передаваемой программе.

**Описание функций.**

TETR\_TO\_HEX – вспомогательная функция, переводит из двоичной в шестнадцатеричную систему.

BYTE\_TO\_HEX – переводит число из регистра AL в шестнадцатиричную систему.

WRD\_TO\_HEX – переводит число из регистра AX в шестнадцатиричную систему.

PRINT – печатает сообщение на экран.

Результат запуска программы представлен на рис.1.



Рисунок 1 – Результат работы программы

**Выводы.**

В ходе выполнения данной лабораторной работы были исследованы интерфейс управляющей программы и загрузочных модулей.

**Ответы на контрольные вопросы.**

**Сегментный адрес недоступной памяти.**

1. На какую область памяти указывает адрес недоступной памяти?

На границу оперативной памяти и на границу области, доступной для загрузки программ.

1. Где расположен этот адрес по отношению области памяти, отведённой программе?

Адрес располагается сразу за памятью, выделенной для программы.

1. Можно ли в эту область памяти писать?

Можно, т.к. в DOS нет защиты памяти.

**Среда передаваемая программе.**

1. Что такое среда?

Среда представляет собой область памяти, в которой в виде символьных строк записаны значения переменных, называемых переменными среды.

1. Когда создается среда? Перед запуском приложения или в другое время?

Создается при загрузке DOS, а при запуске программы происходит копирование среды в новую область памяти.

1. Откуда берется информация, записываемая в среду?

Из системного файла autoexec.bat.

ПРИЛОжение а

исходный код программы

TESTPC SEGMENT

ASSUME CS:TESTPC, DS:TESTPC, ES:NOTHING, SS:NOTHING

ORG 100H

START: JMP BEGIN

;ДАННЫЕ

ADDRES\_OF\_UNAVALIABLE\_MEM db 'Segment address of unavailable memory: ', 13, 10 ,'$'

ADDRES\_OF\_ENVIRONMENT db 'Segment address of environment: ', 13, 10, '$'

TAIL db 'Tail of comand line ', 13, 10, 10, '$'

NEW\_LINE db 'Empty line', 13, 10, 10, '$'

CONTENT\_LINE db 'Content of the environment:', 10, 13, '$'

EMPTY db ' ', 13, 10, '$'

PATH\_LINE db 'Path:', 10, 13, '$'

;ПРОЦЕДУРЫ

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 ;байт в AL переводится в два символа шестн. числа 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

ret

BYTE\_TO\_HEX ENDP

WRD\_TO\_HEX PROC NEAR ;перевод в 16 с/с 16-ти 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

xor AH,AH

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 байта в 10с/с, SI - адрес поля младшей цифры

push AX ; AL содержит исходный байт

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 AX

ret

BYTE\_TO\_DEC ENDP

PRINT PROC NEAR ;печать сообщения на экран

push AX

mov AH, 09h

int 21h

pop AX

ret

PRINT ENDP

ADDRES\_OF\_MEMORY PROC NEAR

push AX

push DI

push DX

mov AX,DS:[02h]

mov DI, OFFSET ADDRES\_OF\_UNAVALIABLE\_MEM

add DI, 43

call WRD\_TO\_HEX

mov DX, OFFSET ADDRES\_OF\_UNAVALIABLE\_MEM

call PRINT

pop DX

pop DI

pop AX

ret

ADDRES\_OF\_MEMORY ENDP

ENVIROMENT\_ADDRES PROC NEAR

push AX

push DI

push DX

mov AX, DS:[2Ch]

mov DI, OFFSET ADDRES\_OF\_ENVIRONMENT

add DI, 36

call WRD\_TO\_HEX

mov DX, OFFSET ADDRES\_OF\_ENVIRONMENT

call PRINT

pop DX

pop DI

pop AX

ret

ENVIROMENT\_ADDRES ENDP

GET\_TAIL PROC NEAR

push AX

push CX

push DI

push SI

xor CX, CX

mov CL, DS:[080h]

mov SI, OFFSET TAIL

add SI, 21

test CL, CL

jz empty1

xor DI, DI

xor AX, AX

gettail:

mov AL, DS:[081h+DI]

mov [SI], AL

inc DI

inc SI

loop gettail

mov DX, OFFSET TAIL

call PRINT

jmp exit1

empty1:

mov DX, OFFSET NEW\_LINE

call PRINT

exit1:

pop SI

pop DI

pop CX

pop AX

ret

GET\_TAIL ENDP

CONTENT PROC NEAR

push AX

push DX

push DS

push ES

mov DX, OFFSET CONTENT\_LINE

call PRINT

mov AH, 02h

mov ES, DS:[2Ch]

xor SI, SI

writecont:

mov DL, ES:[SI]

int 21h

cmp DL, 0h

je endofline

inc SI

jmp writecont

endofline:

mov DX, OFFSET EMPTY

call PRINT

inc SI

mov DL, ES:[SI]

cmp DL, 0h

jne writecont

mov DX, OFFSET EMPTY

call PRINT

pop ES

pop DS

pop DX

pop AX

ret

CONTENT ENDP

PATH PROC NEAR

push AX

push DX

push DS

push ES

mov DX, OFFSET PATH\_LINE

call PRINT

add SI, 3h

mov AH, 02h

mov ES, DS:[2Ch]

writemsg:

mov DL, ES:[SI]

cmp DL, 0h

je endofpath

int 21h

inc SI

jmp writemsg

endofpath:

pop ES

pop DS

pop DX

pop AX

ret

PATH ENDP

BEGIN:

call ADDRES\_OF\_MEMORY

call ENVIROMENT\_ADDRES

call GET\_TAIL

call CONTENT

call PATH

xor AL,AL

mov AH,4Ch

int 21h

TESTPC ENDS

END START