**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. 6383 |  | Карамышев А.О. |
| Преподаватель |  | Губкин А.Ф. |

Санкт-Петербург

2018

**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 и структур данных.**

|  |  |
| --- | --- |
| **Название функции** | **Назначение** |
| GET\_ADRESS\_OFF\_MEM | печатает адрес недоступной памяти |
| GET\_SEGSR | печатает сегментный адрес среды |
| PRINT | вызывает функцию печати строки |
| TAIL | печатает хвост командной строки |
| SREDA | печатает содержимое области среды в символьном виде |
| BYTE\_TO\_HEX | переводит число AL в коды символов 16-ой с/с, записывая получившееся в al и ah |
| TETR\_TO\_HEX | вспомогательная функция для работы функции BYTE\_TO\_HEX |
| WRD\_TO\_HEX | переводит число AX в строку в 16-ой с/с, записывая получившееся в di, начиная с младшей цифры |
| BYTE\_TO\_DEC | переводит байт из AL в десятичную с/с и записывает получившееся число по адресу si, начиная с младшей цифры |

|  |  |  |
| --- | --- | --- |
| **Название** | **Тип** | **Назначение** |
| OFF\_MEM\_ | db | Cтрока, информирующая о том, что дальше выведется адрес недоступной памяти |
| OFF\_MEM | db | Строка для хранения адреса недоступной памяти в символьном виде |
| SEGSR\_ | db | Строка, информирующая о том, что дальше выведется адрес среды окружения |
| SEGSR | db | Строка для хранения адреса среды окружения в символьном виде |
| TAIL\_ | db | Хвост командной строки |
| SREDA\_ | db | Содержимое среды окружения |
| PATH\_ | db | Путь программы |
| ENDL | db | Новая строка |

**Последовательность действий, выполняемых утилитой.**

1. Печатает сегментный адрес первого байта недоступной памяти
2. Печатает сегментный адрес среды, передаваемой программе
3. Печатает хвост командной строки
4. Печатает содержимое области среды в символьном виде
5. Печатает путь загружаемого модуля
6. Выходит в DOS

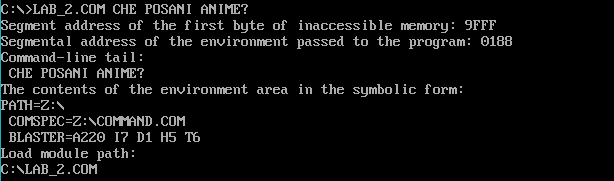
****

Рисунок 1 – результат работы программы lab\_2.com

**Выводы.**

В процессе 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, отведенной программе?

Сразу за областью памяти, отведенной программе.

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?

Среда создается при загрузке DOS, а при запуске приложения копируется в новую область памяти.

1. Откуда берется информация, записываемая в среду?

В MS-DOS она берется из системного файла autoexec.bat.

ПРИЛОЖЕНИЕ А

LAB\_2.ASM

TESTPC SEGMENT

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

ORG 100H

START: JMP BEGIN

; ДАННЫЕ

OFF\_MEM\_ db 'Segment address of the first byte of inaccessible memory: '

OFF\_MEM db ' ',0DH,0AH,'$'

SEGSR\_ db 'Segmental address of the environment passed to the program: '

SEGSR db ' ',0DH,0AH,'$'

TAIL\_ db 'Command-line tail: ',0DH,0AH,'$'

SREDA\_ db 'The contents of the environment area in the symbolic form: ',0DH,0AH,'$'

PATH\_ db 'Load module path: ',0DH,0AH,'$'

ENDL db 0DH,0AH,'$'

; ПРОЦЕДУРЫ

;---------------------------------------

PRINT PROC

push ax

mov ah,09h

int 21h

pop ax

ret

PRINT ENDP

;---------------------------------------

GET\_ADRESS\_OFF\_MEM PROC

mov ax,es:[2]

mov di,offset OFF\_MEM+3

call WRD\_TO\_HEX

lea dx,OFF\_MEM\_

call PRINT

ret

GET\_ADRESS\_OFF\_MEM ENDP

;---------------------------------------

GET\_SEGSR PROC

mov ax,es:[2Ch]

mov di,offset SEGSR+3

call WRD\_TO\_HEX

lea dx,SEGSR\_

call PRINT

ret

GET\_SEGSR ENDP

;---------------------------------------

TAIL PROC

mov dx,offset TAIL\_

call PRINT

mov cx,0

mov cl,es:[80h]

cmp cl,0

je TAIL\_END

mov dx,81h

mov bx,0

mov ah,02h

TAIL\_loop:

mov dl,es:[bx+81h]

int 21h

inc bx

loop TAIL\_loop

mov dx,offset ENDL

call PRINT

TAIL\_END:

ret

TAIL ENDP

;--------------------------------------

SREDA PROC

mov dx,offset SREDA\_

call PRINT

push es

; кладём в es адрес области среды

mov ax,es:[2Ch]

mov es,ax

mov ah,02h

mov bx,0

SREDA\_loop:

mov dl,es:[bx]

int 21h

inc bx

cmp byte ptr es:[bx],00h

jne SREDA\_loop

mov dx,offset ENDL

call PRINT

cmp word ptr es:[bx],0000h

jne SREDA\_loop

add bx,4 ; пропускаем 0001

mov dx,offset PATH\_

call PRINT

SREDA\_loop2:

mov dl,es:[bx]

int 21h

inc bx

cmp byte ptr es:[bx],00h

jne SREDA\_loop2

mov dx,offset ENDL

call PRINT

pop es

ret

SREDA 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

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

;---------------------------------------

; перевод в 16с/с 16-ти разрядного числа

; в AX - число, DI - адрес последнего символа

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

;---------------------------------------

; перевод в 10с/с, SI - адрес поля младшей цифры

BYTE\_TO\_DEC PROC near

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

ret

BYTE\_TO\_DEC ENDP

;---------------------------------------

BEGIN:

call GET\_ADRESS\_OFF\_MEM

call GET\_SEGSR

call TAIL

call SREDA

xor AL,AL

mov AH,4Ch

int 21H

TESTPC ENDS

END START