**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 |  | Михайлов Ю.А. |
| Преподаватель |  | Губкин А.Ф. |

Санкт-Петербург

2018

**Постановка задачи.**

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.

**Описание функций и структур данных**

|  |  |
| --- | --- |
| **Название процедуры** | **Описание процедуры** |
| WRITE\_STR | Вывод строки на экран |
| INFO | Выводит на экран:   1. Сегментный адрес недоступной памяти 2. Сегментный адрес среды 3. Хвост командной строки 4. Содержимое области среды 5. Путь загружаемого модуля |
| TETR\_TO\_HEX | Перевод из четверичной системы счисления в шестнадцатеричную |
| BYTE\_TO\_HEX | Перевод из двоичной системы счисления в шестнадцатеричную |
| WRD\_TO\_HEX | Перевод слова (2 байта) шестнадцатеричную систему счисления |
| BYTE\_TO\_DEC | Перевод из двоичной системы счисления в десятичную |

**Последовательность действий, выполняемых утилитой**

1. Определяется и выводится на экран сегментный адрес недоступной памяти;
2. Определяется и выводится на экран сегментный адрес среды;
3. Определяется и выводится на экран хвост командной строки;
4. Определяется и выводится на экран содержимое области среды;
5. Определяется и выводится на экран путь загружаемого модуля

Результат работы программы представлен на рис.1.



Рисунок 1 – Результат выполнения программы os\_lab\_2.com.

**Вывод.**

В ходе выполнения данной лабораторной работы были исследованы интерфейс управляющей программы, загрузочных модулей, организация PSP, а так же содержание области переменных среды.

**Ответы на контрольные вопросы.**

1. **Сегментный адрес недоступной памяти**
   1. **На какую область памяти указывает адрес недоступной памяти?**

На значение сегментного адреса памяти, следующей за памятью, выделенной программе.

**1.2 Где расположен этот адрес по отношению области памяти, отведенной программе?**

Он расположен за концом памяти, выделенной программе.

* 1. **Можно ли в эту область памяти писать?**

Да, т.к. DOS не имеет механизма защиты памяти.

**2. Среда передаваемая программе**

**2.1 Что такое среда?**

Переменные, в которых хранятся некоторые настройки операционной системы.

**2.2 Когда создается среда? Перед запуском приложения или в другое время?**

Она создается при загрузке DOS. При запуске программы эта среда копируется для нее.

**2.3 Откуда берется информация, записываемая в среду?**

Из реестра ОС(MS Windows).

**КОД ПРОГРАММЫ OS\_LAB\_2.ASM**

TESTPC SEGMENT

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

org 100h

START: JMP BEGIN

; ДАННЫЕ

MEM db 13, 10, "Locked memory address: h$" ; 17 symbols

ENV db 13, 10, "Environment address: h$" ; 23 symbols

TAIL db 13, 10, "Command line tail: $" ; 21 symbols

EMP db 13, 10, "There are no sybmols$"

CONT db 13, 10, "Content:", 13, 10, "$"

ENT db 13, 10, "$"

PATH db 13, 10, "Path:", 13, 10, "$" ; 8 symbols

;ПРОЦЕДУРЫ

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

WRITE\_STR PROC near

push ax

mov ah, 09h

int 21h

pop ax

ret

WRITE\_STR ENDP

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

INFO PROC near

; Memory

mov ax, ds:[02h]

mov di, offset MEM

add di, 28

call WRD\_TO\_HEX

mov dx, offset MEM

call WRITE\_STR

; Environment

mov ax, ds:[2Ch]

mov di, offset ENV

add di, 26

call WRD\_TO\_HEX

mov dx, offset ENV

call WRITE\_STR

; Tail

xor cx, cx

mov cl, ds:[80h]

mov si, offset TAIL

add si, 20

test cl, cl

jz empty

xor di, di

xor ax, ax

readtail:

mov al, ds:[81h+di]

mov [si], al

inc di

inc si

loop readtail

mov dx, offset TAIL

call WRITE\_STR

jmp nextaction

empty:

mov dx, offset EMP

call WRITE\_STR

nextaction: nop

; Envrironment content

mov dx, offset CONT

call WRITE\_STR

xor di, di

mov bx, 2Ch

mov ds, [bx]

readstring:

cmp byte ptr [di], 00h

jz pressenter

mov dl, [di]

mov ah, 02h

int 21h

jmp findend

pressenter:

push ds

mov cx, cs

mov ds, cx

mov dx, offset ENT

call WRITE\_STR

pop ds

findend:

inc di

cmp word ptr [di], 0001h

jz readpath

jmp readstring

readpath:

push ds

mov ax, cs

mov ds, ax

mov dx, offset PATH

call WRITE\_STR

pop ds

add di, 2

pathloop:

cmp byte ptr [di], 00h

jz final

mov dl, [di]

mov ah, 02h

int 21h

inc di

jmp pathloop

final:

ret

INFO 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

; 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

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

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

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

; КОД

BEGIN:

call INFO

mov ah, 10h

int 16h

; Выход в DOS

xor AL,AL

mov AH,4Ch

int 21H

TESTPC ENDS

END START ;конец модуля, START - точка входа