**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра математического обеспечения и применения ЭВМ**

отчет

**по лабораторной работе №2**

**по дисциплине «Операционные системы»**

Тема: **Исследование интерфейсов командных модулей**

|  |  |  |
| --- | --- | --- |
| Студент гр. 8383 |  | Дейнега В.Е. |
| Преподаватель |  | Ефремов М. А. |

Санкт-Петербург

2020

**Цель работы.**

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

**Ход работы.**

В ходе выполнения лабораторной работы был написан и отлажен программный модуль типа **.COM**, который выбирает и распечатывает на экран следующую информацию:

1. Сегментный адрес недоступной памяти 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.

Результат работы представлен на рис. 1 и 2. Исходный код программы содержится в Приложении А.



Рисунок 1 – Работа без входных данных



Рисунок 2 – Работа программы с входными данными

**Контрольные вопросы**

Ниже приведены ответы на контрольные вопросы:

**Сегментный адрес недоступной памяти**

1. На какую область памяти указывает адрес недоступной памяти? Адрес недоступной памяти указывает на служебную часть памяти, программа не может ее использовать.
2. Где расположен адрес по отношению к памяти, выделенной программе? После памяти, отведенной программе.
3. Можно ли в эту область памяти писать? Да, DOS не может запретить делать этого.

**Среда, передаваемая программе**

1. Что такое среда? Среда – последовательность строк вида параметр = значение, хранящие какую-либо информацию, например, данные о настройках системы.
2. Когда создается среда? Среда создается при запуске программы.
3. Откуда берется информация, записываемая в среду? Из файла AUTOEXEC.BAT.

**Выводы.**

В ходе выполнения работы был исследован интерфейс управляющей программы и загрузочных модулей, а так же интерфейс префикса сегмента программы (PSP) и среды, передаваемой программе.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

TESTPC SEGMENT

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

ORG 100H

START: JMP MAIN

LEN\_LOCKED EQU 31

LEN\_ENV EQU 28

STR\_LOCKED db 13,10, "Address of locked memory: $"

STR\_ENV db 13,10, "Address of enviroment: $"

STR\_TAIL db 13,10, "Tail comand\_line: $"

STR\_EMPTY\_TAIL db " no command line $"

STR\_ENVIROMENT\_AREA db 13,10, "Enviroment: $"

STR\_ENTER db 13,10, " $"

STR\_PATH db 13,10, "Path: $"

WRITE\_STR PROC near

push ax

mov ah, 09h

int 21h

pop ax

ret

WRITE\_STR 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

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 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

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

PRINT\_LOCKED\_MEM PROC near

push ax

push dx

mov ax, ds:[02h]

mov di, offset STR\_LOCKED

add di, LEN\_LOCKED

call WRD\_TO\_HEX

mov dx, offset STR\_LOCKED

call WRITE\_STR

pop dx

pop ax

ret

PRINT\_LOCKED\_MEM ENDP

PRINT\_ADDRESS\_ENVIROMENT PROC near

push ax

push dx

mov ax, ds:[2Ch]

mov di, offset STR\_ENV

add di, LEN\_ENV

call WRD\_TO\_HEX

mov dx, offset STR\_ENV

call WRITE\_STR

pop dx

pop ax

ret

PRINT\_ADDRESS\_ENVIROMENT ENDP

PRINT\_TAIL PROC near

push ax

push dx

mov dx, offset STR\_TAIL

call WRITE\_STR

mov cx, 0

mov cl, ds:[80h]

cmp cl, 0

je tail\_empty

mov di, 0

xor dx,dx

print\_tail\_cycle:

mov dl, ds:[81h+di]

mov ah,02H

int 21h

inc di

loop print\_tail\_cycle

jmp end\_print

tail\_empty:

mov dx, offset STR\_EMPTY\_TAIL

call WRITE\_STR

end\_print:

pop dx

pop ax

ret

PRINT\_TAIL ENDP

PRINT\_PATH PROC near

push dx

push ax

push ds

mov dx, offset STR\_ENVIROMENT\_AREA

call WRITE\_STR

mov di, 0

mov es, ds:[2Ch]

cycle\_env:

cmp byte ptr es:[di], 00h

je enter\_

mov dl, es:[di]

mov ah, 02h

int 21h

inc di

jmp cycle\_env

enter\_:

inc di

cmp word ptr es:[di], 0001h

je path\_

mov dx, offset STR\_ENTER

call WRITE\_STR

jmp cycle\_env

path\_:

inc di

inc di

mov DX, offset STR\_PATH

call WRITE\_STR

cycle\_p:

cmp byte ptr es:[di], 00h

je end\_print\_p

mov dl, es:[di]

mov ah, 02h

int 21h

inc di

jmp cycle\_p

end\_print\_p:

pop dx

pop ax

pop ds

ret

PRINT\_PATH ENDP

MAIN:

call PRINT\_LOCKED\_MEM

call PRINT\_ADDRESS\_ENVIROMENT

call PRINT\_TAIL

call PRINT\_PATH

xor al, al

mov AH,4Ch

int 21H

TESTPC ENDS

END START