**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. 8383 |  | Мололкин К.А. |
| 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.**

Во время выполнения работы был написан и отлажен код .СОM модуля, код модуля представлен в приложении А. Данная программа читает и распечатывает сегментный адрес недоступной памяти, сегментный адрес среды, хвост командной строки, содержимое области среды, а также путь загружаемого модуля. Результат выполнения программы показан на рис. 1.

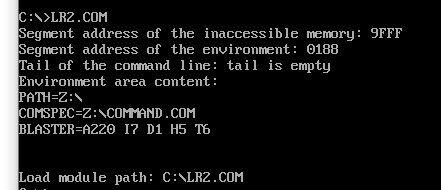


Рисунок 1 – Результат выполнения программы

**Сегментный адрес недоступной памяти**

1. На какую область памяти указывает адрес недоступной памяти?

Адрес недоступной памяти указывает на область памяти, за которым программа не должна модифицировать содержимое.

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

Адрес расположен после памяти, отведенной программе.

1. Можно ли в эту область памяти писать?

В эту область памяти можно писать, так DOS не препятствует этому.

**Среда, передаваемая программе**

1. Что такое среда?

Область среды содержит последовательность строк вида:

имя=параметр

Каждая строка завершается байтом нулей. Под строки выделяется 32 Кбайта памяти.

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

Среда создается перед запуском приложения. Она копируется из COMMAND.COM, которая создается при запуске DOS.

1. Откуда берется информация, записываемая в среду

Из родительской среды.

**Выводы.**

В ходе выполнения лабораторной работы был исследован интерфейс управляющей программы и загрузочных модулей, а также префикс сегмента программы (PSP) и среды, передаваемой программе.

Приложение а

**КОД ПРОГРАММЫ**

LR2 SEGMENT

ASSUME CS:LR2, DS:LR2, ES:NOTHING, SS:NOTHING

ORG 100H

START: JMP BEGIN

LOCKED\_MEMORY\_ADRESS db "Segment address of the inaccessible memory: $"

ENVIRONMENT\_ADRESS db 13, 10, "Segment address of the environment: $"

COMAND\_LINE\_TAIL db 13, 10, "Tail of the command line: $"

EMPTY\_TAIL db "tail is empty$"

ENIRONMENT\_CONTENT db 13, 10, "Environment area content:", 13, 10, '$'

LOAD\_PATH db 13, 10, "Load module path: $"

ENTER\_SYMB db 13, 10, '$'

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

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

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\_STRING PROC near

mov ah, 09h

int 21h

ret

PRINT\_STRING ENDP

PRINT\_SYMBOL PROC near

mov dl, al

mov ah, 02h

int 21h

ret

PRINT\_SYMBOL ENDP

LM\_ADRESS PROC near

mov ax, ds:[02h]

mov cx, offset LOCKED\_MEMORY\_ADRESS

mov di, cx

add di, 47

call WRD\_TO\_HEX

mov dx, cx

call PRINT\_STRING

ret

LM\_ADRESS ENDP

E\_ADRESS PROC near

mov ax, ds:[2Ch]

mov cx, offset ENVIRONMENT\_ADRESS

mov di, cx

add di, 41

call WRD\_TO\_HEX

mov dx, cx

call PRINT\_STRING

ret

E\_ADRESS ENDP

TAIL PROC near

mov dx, offset COMAND\_LINE\_TAIL

call PRINT\_STRING

xor cx, cx

mov cl, ds:[80h]

cmp cl, 0

jne line\_not\_empty

mov dx, offset EMPTY\_TAIL

call PRINT\_STRING

jmp p\_end

line\_not\_empty:

xor si, si

xor dx, dx

cycle:

mov al, DS:[81h + si]

call PRINT\_SYMBOL

inc si

loop cycle

p\_end:

ret

Tail ENDP

E\_CONTENT PROC near

mov dx, offset ENIRONMENT\_CONTENT

call PRINT\_STRING

xor si, si

mov bx, 2Ch

mov es, [bx]

print\_content\_cycle:

cmp BYTE PTR es:[si], 0h

je print\_enter

mov al, es:[si]

call PRINT\_SYMBOL

jmp end\_reading

print\_enter:

mov dx, offset ENTER\_SYMB

call PRINT\_STRING

end\_reading:

inc si

cmp WORD PTR es:[si], 0001h

jne print\_content\_cycle

mov DX, offset LOAD\_PATH

call PRINT\_STRING

add SI, 2

print\_path:

cmp BYTE PTR es:[si], 00h

je p\_endd

mov al, es:[si]

call PRINT\_SYMBOL

inc si

jmp print\_path

p\_endd:

ret

E\_CONTENT ENDP

BEGIN:

call LM\_ADRESS

call E\_ADRESS

call TAIL

call E\_CONTENT

xor AL, AL

mov AH, 4Ch

int 21h

LR2 ENDS

END START