**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра математического обеспечения и применения ЭВМ**

отчет

**по практической работе № 2**

**по дисциплине «Операционные системы»**

Тема: **Исследование интерфейсов программных модулей**

|  |  |  |
| --- | --- | --- |
| Студент гр. 9382 |  | Иерусалимов Н. |
| Преподаватель |  | Ефремов М.А. |

Санкт-Петербург

2021

**Цель работы.**

Исследование интерфейса управляющей̆ программы и загрузочных модулей̆. Этот интерфейс состоит в передаче запускаемой̆ программе управляющего блока, содержащего адреса и системные данные. Так загрузчик строит префикс сегмента программы (PSP) и помещает его адрес в сегментный̆ регистр. Исследование префикса сегмента программы (PSP) и среды, передаваемой̆ программе.

**Задание.**

**Шаг 1.**

Для выполнения лабораторной работы необходимо написать и отладить программный модуль типа .COM, который выбирает и распечатывает следующую информацию:

1. Сегментный адрес недоступной памяти, взятый из PSP, 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.

Сохраните результаты, полученные программой, и включите их в отчет.

**Шаг 2.**

Оформление отчета в соответствии с требованиями. В отчет включите скриншот с запуском программы и результатами.

**Выполнение работы.**

В ходе выполнения работы была реализована СOM программа, которая выводит на экран:

* Cегментный адрес недоступной памяти, данные брались из PSP .
* Адрес среды в шестнадцатеричном виде, получили с использованием WRD\_TO\_HEX в первом пункте.
* Хвост команды (как по мне это можно назвать аргументом.)
* Путь до программы
* Адрес среды в текстовом виде.

Результат работы программы можно увидеть на рисунке 1\2.

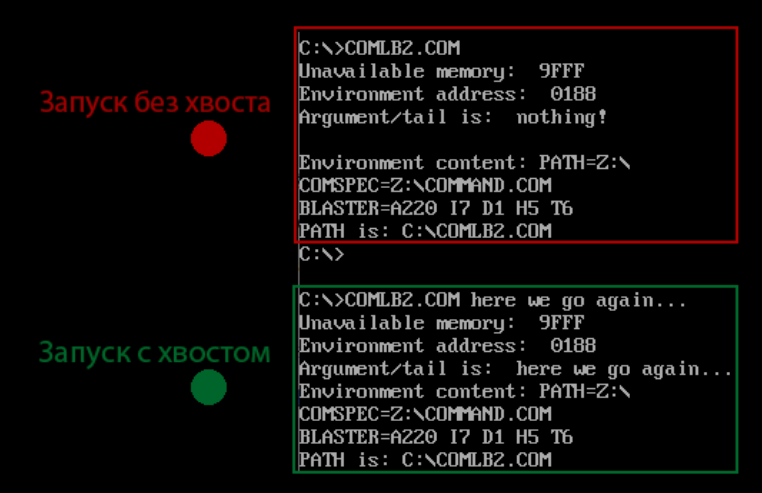


Рис.1 <Пример работы программы>

**Ответы на вопросы.**

**Сегментный адрес недоступной памяти**

1. На какую область памяти указывает адрес недоступной памяти?

На первый байт после участка памяти, отведенного под программу

2) Где расположен этот адрес по отношению области памяти, отведенной

программе?

Адрес недоступной памяти по отношению к области памяти,

отведенной программе, расположен после. В сторону увеличения адресов.

1. Можно ли в эту область памяти писать?

Можно, DOS не защищен от перезаписи памяти, для которых эта память не выделялась.

**Среда передаваемая программе**

1. Что такое среда?

Набор переменных которые хранят некоторую информацию о состоянии системы.

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

При загрузке DOS.

1. Откуда берется информация, записываемая в среду?

Из файла AUTOEXEC.BAT, который расположен в корневом каталоге загрузочного устройства.

**Выводы.**

Был исследован интерфейс управляющей программы и загрузочных модулей. Изучили префикс сегмента программы (PSP) и среды, передаваемой̆ программе.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Файл: comLB2.ASM

testpc segment

assume cs:testpc,ds:testpc,es:nothing,ss:nothing

org 100h

start: jmp main

env\_addr db 'Environment address:', '$'

unavailable db 'Unavailable memory:', '$'

ax\_register db ' ', 0dh,0ah,'$'

ENV db 'Environment content: ', '$'

NEW\_LINE db 0dh,0ah,'$'

PATH db 'PATH is: ', '$'

ZERO\_ARGUMENTS db ' nothing!', 0dh,0ah, 0dh,0ah,'$'

TAIL\_ARGUMENTS db 'Argument/tail is: ', '$'

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 si

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 si

ret

Byte\_to\_dec endp

;///////////////////////////////

;/////////\_Code\_///////////////

;//////////////////////////////

Writestring proc near

push ax

mov ah,09h

int 21h

pop ax

ret

Writestring endp

NULLString proc near

push si

push cx

xor si,si

mov cx,5

Clear:

mov [offset ax\_register+si],0ff20h

inc si

loop Clear

mov [offset ax\_register+si],0ff20h

pop cx

pop si

ret

NULLString endp

Display\_info proc near

call writestring

mov di,offset ax\_register

add di,5

call wrd\_to\_hex

mov dx,offset ax\_register

call writestring

call NULLString

ret

Display\_info endp

Display\_UnMem proc near

mov dx,offset unavailable

mov ax,ds:[02h]

call Display\_info

ret

Display\_UnMem endp

Print\_env\_addr proc near

mov dx,offset env\_addr

mov ax,ds:[2ch]

call Display\_info

ret

Print\_env\_addr endp

Display\_command\_tail proc near

push cx

push ax

xor cx,cx

mov dx,offset TAIL\_ARGUMENTS

call writestring

mov cl,ds:[80h]

cmp cl,0

je \_empty\_tail

mov si,0

Display\_tail\_symbol:

mov dl,ds:[81h+si]

mov ah,02h

int 21h

inc si

loop Display\_tail\_symbol

mov dx,offset NEW\_LINE

call writestring

jmp \_exit\_tail\_print

\_Empty\_tail:

mov dx,offset ZERO\_ARGUMENTS

call writestring

\_Exit\_tail\_print:

pop ax

pop cx

ret

Display\_command\_tail endp

Print\_environment proc near

push dx

push ax

push si

push ds

xor si,si

mov dx,offset ENV

call writestring

mov ds,ds:[2ch]

\_Read\_env:

mov dl,[si]

cmp dl,0

je \_eof

mov ah,02h

int 21h

inc si

jmp \_read\_env

\_Eof:

inc si

mov dl,[si]

cmp dl,0

je \_end\_reading\_env

pop ds

mov dx,offset NEW\_LINE

call writestring

push ds

mov ds,ds:[2ch]

jmp \_read\_env

\_End\_reading\_env:

pop ds

mov dx,offset NEW\_LINE

call writestring

mov dx,offset PATH

call writestring

push ds

mov ds,ds:[2ch]

add si,3

\_Reading\_path:

mov dl,[si]

cmp dl,0

je \_exit\_print\_env

mov ah,02h

int 21h

inc si

jmp \_reading\_path

\_Exit\_print\_env:

pop ds

pop si

pop ax

pop dx

ret

Print\_environment endp

main:

call Display\_UnMem

call print\_env\_addr

call Display\_command\_tail

call print\_environment

xor al,al

mov ah,4ch

int 21h

testpc ends

end start