**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. 9383 |  | Рыбников Р.А. |
| Преподаватель |  | Ефремов М.А. |

Санкт-Петербург

2021

**Цель работы.**

Исследование интерфейса управляющей программы и загрузочных модулей. Этот интерфейс состоит в передаче запускаемой программе управляющего блока, содержащего адреса и системные данные. Так загрузчик состоит префикс сегмента программы (PSP) и помещает его адрес в сегментные регистр. Исследование префикса сегмента программы (PSP) и среды, передаваемой программе.

**Порядок выполнения работы.**

**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 стоки в символьном виде.
4. Содержимое области среды в символьном виде.
5. Путь загружаемого модуля.

**Шаг 2.** Оформление отчета в соответствии с требованиями. В отчет включается скриншот с запуском программы и результатами.

**Выполнение работы.**

Были составлены функции для считывания данных из префикса и преобразования чисел.

В результате выполнения были получены следующие значения:

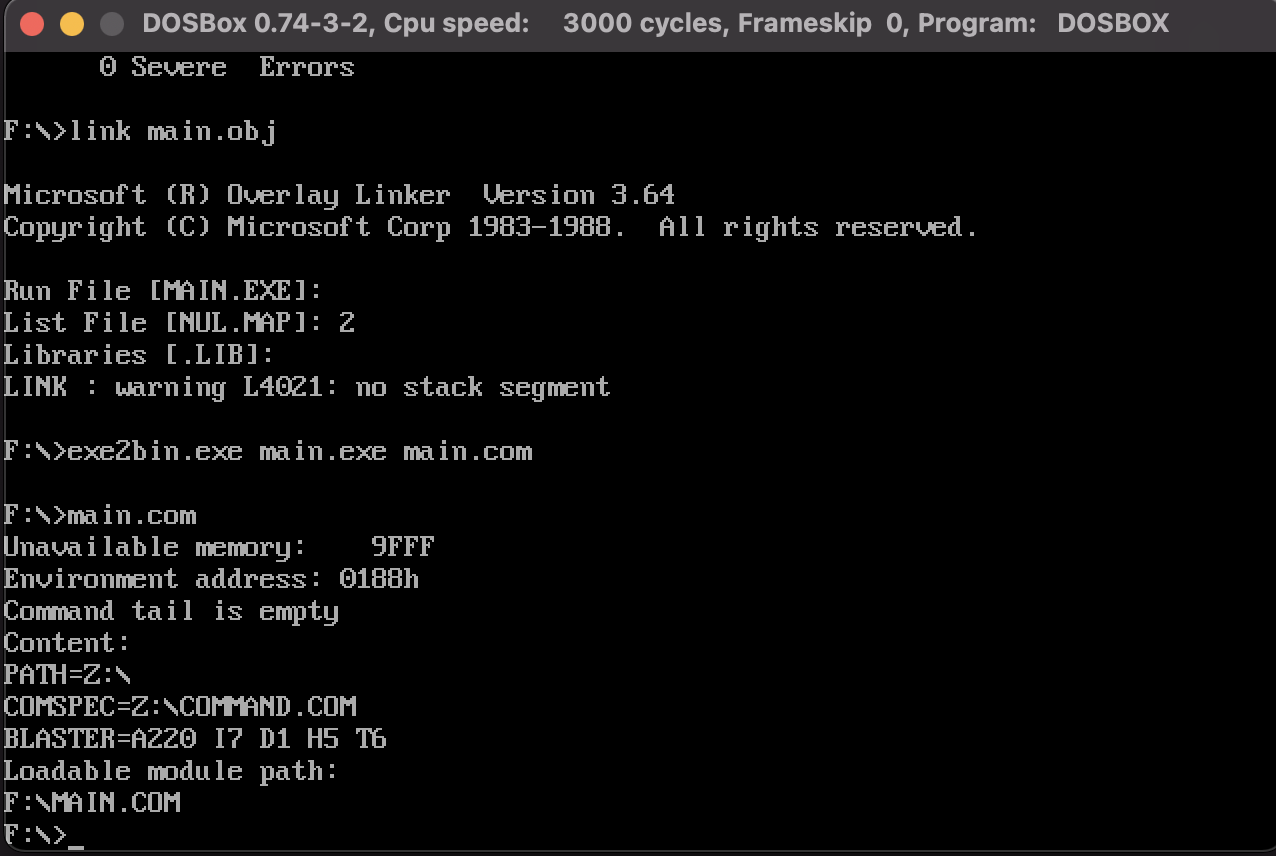


Рисунок 1 -- Пример работы программы.

**Выводы.**

В ходе лабораторной работы была реализована .COM программа, исходный код которой расположен в приложении А.

ОТВЕТЫ НА КОНТРОЛЬНЫЕ ВОПРОСЫ

**Сегментный адрес недоступной памяти:**

1. На какую область памяти указывает адрес недоступной памяти?

На первый байт после памяти после программы.

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

PSP по адресу 2сh.

1. Можно ли в эту область памяти писать?

Да, т.к. в DOS-е нет механизмов защиты.

**Среда, передаваемая программе:**

1. Что такое среда?

Среда -- это область памяти, в которой записаны переменные среды.

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

При загрузке ОС, но перед запуском приложения, она может быть изменена в соответствии с требованиями этого приложения.

1. Откуда берётся информация, записываемая в среду?

Из системного файла AUTOEXEC.BAT, который расположен в [корневом каталоге](https://ru.wikipedia.org/wiki/%D0%9A%D0%B0%D1%82%D0%B0%D0%BB%D0%BE%D0%B3_(%D1%84%D0%B0%D0%B9%D0%BB%D0%BE%D0%B2%D0%B0%D1%8F_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0)) [загрузочного устройства](https://ru.wikipedia.org/w/index.php?title=%D0%97%D0%B0%D0%B3%D1%80%D1%83%D0%B7%D0%BE%D1%87%D0%BD%D0%BE%D0%B5_%D1%83%D1%81%D1%82%D1%80%D0%BE%D0%B9%D1%81%D1%82%D0%B2%D0%BE&action=edit&redlink=1).

ПРИЛОЖЕНИЕ А.

TESTPC SEGMENT

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

ORG 100H

START: jmp BEGIN

MEMORY\_ADDRESS db 'Unavailable memory: h',13,10, 13, 10, '$'

ENV\_ADDRESS db 'Environment address: h',13,10,'$'

NOT\_EMPTY\_TAIL db 'Command line tail: ',13,10,'$'

EMPTY\_TAIL\_STR db 'Command tail is empty',13,10,'$'

CONTENT\_STR db 'Content:',13,10, '$'

END\_OF\_LINE db 13, 10, '$'

PATH db 'Loadable module path: ',13,10,'$'

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

WRITE\_STRING PROC near

mov AH,09h

int 21h

ret

WRITE\_STRING ENDP

UNAVAILABLE\_MEMORY PROC near

mov ax,ds:[02h]

mov di, offset MEMORY\_ADDRESS

add di, 26

call WRD\_TO\_HEX

mov dx, offset MEMORY\_ADDRESS

call WRITE\_STRING

ret

UNAVAILABLE\_MEMORY ENDP

ENVIROMENT\_ADDRESS PROC near

mov ax,ds:[2Ch]

mov di, offset ENV\_ADDRESS

add di, 24

call WRD\_TO\_HEX

mov dx, offset ENV\_ADDRESS

call WRITE\_STRING

ret

ENVIROMENT\_ADDRESS ENDP

COMMAND\_LINE\_TAIL PROC near

xor cx, cx

mov cl, ds:[80h]

mov si, offset NOT\_EMPTY\_TAIL

add si, 19

cmp cl, 0h

je empty\_tail

xor di, di

xor ax, ax

next\_tail:

mov al, ds:[81h+di]

inc di

mov [si], al

inc si

loop next\_tail

mov dx, offset NOT\_EMPTY\_TAIL

jmp TAIL\_END

empty\_tail:

mov dx, offset EMPTY\_TAIL\_STR

TAIL\_END:

call WRITE\_STRING

ret

COMMAND\_LINE\_TAIL ENDP

CONTENT PROC near

mov dx, offset CONTENT\_STR

call WRITE\_STRING

xor di,di

mov ds, ds:[2Ch]

READ\_LINE:

cmp byte ptr [di], 00h

jz END\_LINE

mov dl, [di]

mov ah, 02h

int 21h

jmp find\_end

END\_LINE:

cmp byte ptr [di+1],00h

jz FIND\_END

push ds

mov cx, cs

mov ds, cx

mov dx, offset END\_OF\_LINE

call WRITE\_STRING

pop ds

FIND\_END:

inc di

cmp word ptr [di], 0001h

jz PATH\_READING

jmp READ\_LINE

PATH\_READING:

push ds

mov ax, cs

mov ds, ax

mov dx, offset PATH

call WRITE\_STRING

pop ds

add di, 2

LOOP\_PATH:

cmp byte ptr [di], 00h

jz EXIT

mov dl, [di]

mov ah, 02h

int 21h

inc di

jmp LOOP\_PATH

EXIT:

ret

CONTENT ENDP

BEGIN:

call UNAVAILABLE\_MEMORY

call ENVIROMENT\_ADDRESS

call COMMAND\_LINE\_TAIL

call CONTENT

xor AL,AL

mov AH,4Ch

int 21H

TESTPC ENDS

END START