**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

|  |  |  |
| --- | --- | --- |
| Студентка гр. 6381 |  | Лопатина А.С. |
| Преподаватель |  | Губкин А.Ф. |

Санкт-Петербург

2018

**Постановка задачи**

**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 строит префикс сегмента программы (PSP) и помещает его адрес в сегментный регистр. Исследование префикса сегмента программы (PSP) и среды, передаваемой программе.

**Необходимые сведения для составления программы.**

При начальной загрузке программы формируется PSP, который размещается в начале первого сегмента программы. PSP занимает 256 байт и располагается с адреса, кратного границе сегмента. При загрузке модулей типа .COM все сегментные регистры указывают на адрес PSP. Именно по этой причине значения этих регистров в модуле .EXE следует переопределять.

Формат PSP:

|  |  |  |
| --- | --- | --- |
| Смещение | Длина поля (байт) | Содержимое поля |
| 0 | 2 | int 20h |
| 2 | 2 | Сегментный адрес первого байта недоступной памяти. Программа не должна модифицировать содержимое памяти за этим адресом. |
| 4 | 6 | Зарезервировано |
| 0Ah(10) | 4 | Вектор прерывания 22h (IP, CS) |
| 0Eh(14) | 4 | Вектор прерывания 23h (IP, CS) |
| 12h(18) | 4 | Вектор прерывания 24h (IP, CS) |
| 2Ch(44) | 2 | Сегментный адрес среды, передаваемой программе. |
| 5Ch |  | Область форматируется как стандартный неоткрытый блок управления файлом (FCB) |
| 6Ch |  | Область форматируется как стандартный неоткрытый блок управления файлом (FCB). Перекрывается если FCB с адреса 5Ch открыт. |
| 80h | 1 | Число символов в хвосте командной строки. |
| 81h |  | Хвост командной строки – последовательность символов после имени вызываемого модуля. |

Область среды содержит последовательность символьных строк вида:

*имя=параметр*

Каждая строка завершается байтом нулей.

В первой строке указывается имя COMSPEC, которая определяет используемый командный процессор и путь к COMMAND.COM. Следующие строки содержат информацию, задаваемую командами PATH, PROMT, SET.

Среда заканчивается также байтом нулей. Таким образом, два нулевых байта являются признаком конца переменных среды. Затем идут два байта, содержащих 00h, 01h, после которых располагается маршрут загруженной программы. Маршрут также заканчивается байтом 00h.

**Сведения о функциях и структурах данных управляющей программы.**

Сведения о функциях:

|  |  |
| --- | --- |
| Название | Назначение |
| TETR\_TO\_HEX | перевод полвины байта из 2-ой СС в 16-ую СС |
| BYTE\_TO\_HEX | перевод байта, расположенного в AL, в 2 символа 16-ой числа в AX |
| WRD\_TO\_HEX | перевод в 16-ую СС 16-ти разрядного числа |
| BYTE\_TO\_DEC | перевод в 10-ую СС |
| OUTPUT\_PROC | вывод на экран |
| DETERMINE\_ADDR\_MEM | определение сегментного адреса недоступной памяти |
| DETERMINE\_ADDR\_ENV | определение сегментного адреса среды |
| DETERMINE\_TAIL | определение хвоста командной строки |
| DETERMINE\_CONTENT\_PATH | определение содержимого области среды и пути загружаемого модуля |

Сведения о структурах данных:

|  |  |  |
| --- | --- | --- |
| Название | Тип | Назначение |
| ADDR\_MEM | db | Адрес недоступной памяти |
| ADDR\_ENV | db | Адрес среды |
| TAIL | db | Хвост командной строки |
| CONTENT | db | Содержимое области среды |
| PATH | db | Путь загружаемого модуля |
| NEW\_LINE | db | Переход на новую строку |

**Последовательность действий, выполняемых программой.**

Модуль .СОМ выбирает и выводит на экран:

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.

**Результат работы программы.**

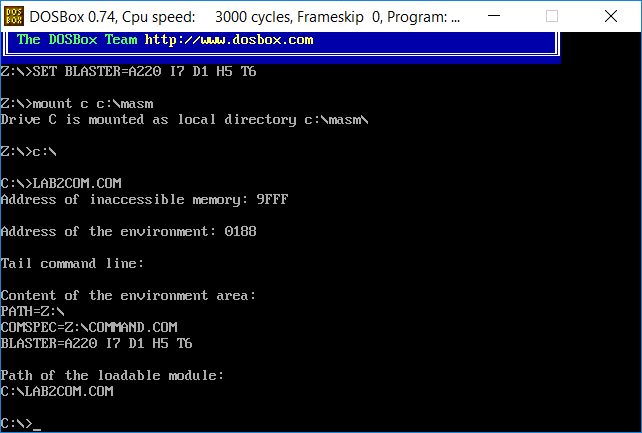


рис.1 Результат работы программы

**Описание результатов исследования проблем, поставленных в лабораторной работе**

**Сегментный адрес недоступной памяти.**

1. На какую область памяти указывает адрес недоступной памяти?

Ответ: адрес недоступной памяти указывает на границу оперативной памяти и на границу области, доступной для загрузки программ;

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

Ответ: адрес расположен сразу за областью памяти, отведенной программе;

1. Можно ли в эту область памяти писать?

Ответ: в эту область памяти писать можно, так как DOS не имеет механизма защиты памяти;

**Среда, передаваемая программе.**

1. Что такое среда?

Ответ: среда – это область памяти, содержащая переменные, в которых хранятся некоторые настройки ОС, записанные в виде строк формата *имя=параметр*;

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

Ответ: среда создается при загрузке DOS, а не перед запуском приложения;

1. Откуда берется информация, записываемая в среду?

Ответ: информация, записываемая в среду, берется из системного файла autoexec.bat.

**Заключение**

В результате выполнения лабораторной работы был исследован интерфейс управляющей программы и загрузочных модулей, а также префикс сегмента программы (PSP) и среда, передаваемая программе.

Приложения

**Код программы** **lab2com**.**asm**

TESTPC SEGMENT

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

ORG 100H

START: jmp BEGIN

;Данные

ADDR\_MEM db 'Address of inaccessible memory: ',0dh,0ah,'$'

ADDR\_ENV db 'Address of the environment: ',0dh,0ah,'$'

TAIL db 'Tail command line: ',0dh,0ah,'$'

CONTENT db 'Content of the environment area: ' , '$'

PATH db 'Path of the loadable module: ' , '$'

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

;Процедуры

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

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 - старша¤, в AH - младша¤

pop cx

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 10 с/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

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

OUTPUT\_PROC PROC NEAR ;Вывод на экран сообщения

push ax

mov ah, 09h

int 21h

pop ax

ret

OUTPUT\_PROC ENDP

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

DETERMINE\_ADDR\_MEM PROC NEAR ;Определение сегментного адреса недоступной памяти

push ax

push di

mov ax, ds:[02h] ;Загружаем адрес

lea di, ADDR\_MEM

add di, 23h ;Загружаем адрес последнего символа в строке

call WRD\_TO\_HEX ;Переводим в 16 СС

pop di

pop ax

ret

DETERMINE\_ADDR\_MEM ENDP

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

DETERMINE\_ADDR\_ENV PROC NEAR ;Определение сегментного адреса среды

push ax

push di

mov ax, ds:[2Ch] ;Загружаем адрес

lea di, ADDR\_ENV

add di, 01Fh ;Загружаем адрес последнего символа в строке

call WRD\_TO\_HEX ;Переводим в 16 СС

pop di

pop ax

ret

DETERMINE\_ADDR\_ENV ENDP

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

DETERMINE\_TAIL PROC NEAR ;Определение хвоста командной строки

push ax

push cx

push dx

push si

push di

xor cx, cx

mov ch, ds:[80h] ; Загружаем число символов в конце командной строки

mov si, 81h

mov di, offset TAIL

add di, 14h ;Загружаем адрес последнего символа в строке

get\_line:

cmp ch, 0 ; Проверяем на пустоту строки

je null\_l

mov al, ds:[si] ; Записываем очередной символ

mov [di], al

inc di

inc si

dec ch

jmp get\_line

null\_l:

mov al, 0h

mov [di], al

pop di

pop si

pop dx

pop cx

pop ax

ret

DETERMINE\_TAIL ENDP

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

DETERMINE\_CONTENT\_PATH PROC near ;Определяем содержимое области среды и путь к загружаемому модулю

push es

push ax

push bx

push cx

lea dx, CONTENT ;Вывод поясняющей строки

call OUTPUT\_PROC

mov es,es:[2ch] ;Записываем начало содержимого области среды

mov bx, 0

xor si,si

new:

lea dx, NEW\_LINE ;Перенос на новую строку

call OUTPUT\_PROC

mov ax,si ;Сохраняем адрес начала названия элемента области среды

count\_env:

cmp byte ptr es:[si], 0

je get\_content ;Доходим до конца элемента области среды

inc si

jmp count\_env

get\_content:

push es:[si]

mov byte ptr es:[si], '$'

push ds

mov cx,es

mov ds,cx

mov dx,ax ;Записываем значение адреса начала строки

call OUTPUT\_PROC ;Выводим полученную строку

pop ds

pop es:[si] ;Возвращаем значение текущей ячейки

cmp bx,0 ;Проверяем на окончание

jne end\_proc ; Если bx!=0 то переходим к концу процедуры

inc si

cmp byte ptr es:[si], 1 ;Проверяем, идет ли дальше информация о пути

jne new

lea dx, NEW\_LINE ;Перенос на новую строку

call OUTPUT\_PROC

lea dx,PATH ;Записываем значение адреса пути

call OUTPUT\_PROC

mov bx,1 ;Сообщаем программе, что дальше пойдет путь

add si,2

jmp new

end\_proc:

lea dx,NEW\_LINE

call OUTPUT\_PROC

pop cx

pop bx

pop ax

pop es

ret

DETERMINE\_CONTENT\_PATH ENDP

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

BEGIN:

call DETERMINE\_ADDR\_MEM ;Определяем адрес памяти

lea dx,ADDR\_MEM

call OUTPUT\_PROC

lea dx,NEW\_LINE

call OUTPUT\_PROC

call DETERMINE\_ADDR\_ENV ;Определяем адрес среды

lea dx,ADDR\_ENV

call OUTPUT\_PROC

lea dx,NEW\_LINE

call OUTPUT\_PROC

call DETERMINE\_TAIL ;Определяем хвост командной строки

lea dx,TAIL

call OUTPUT\_PROC

lea dx,NEW\_LINE

call OUTPUT\_PROC

call DETERMINE\_CONTENT\_PATH ;Определяем содержимое среды и путь модуля

;Выход в DOS

xor al, al

mov ah, 4ch

int 21h

TESTPC ENDS

END START