**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра MOЭBM**

отчет

**по лабораторной работе №2**

**по дисциплине «Операционные системы»**

**Тема: Исследование интерфейсов программных модулей**

|  |  |  |
| --- | --- | --- |
| Студент гр. 6381 |  | Вергейчик Г. Л. |
| Преподаватель |  | Губкин А. Ф. |

Санкт-Петербург

2018

**Цель работы**

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

**Дополнительные сведения**

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

Формат PSP:

* Смещение 0 – int20h
* Смещение 2 – сегментный адрес первого байта недоступно памяти
* Смещение 4 – зарезервировано
* Смещение 0Ah – вектор прерывания 22h
* Смещение 0Eh – вектор прерывания 23h
* Смещение 12h – вектор прерывания 24h
* Смещение 2Ch – сегментный адрес среды, передаваемой программе
* Смещение 5Ch - FCB
* Смещение 6Ch - FCB
* Смещение 80h – число символов в хвосте командной строки
* Смещение 81h – хвост командной строки

**Ход работы**

1. Был написан файл исходного .COM модуля, определяющего сегментный адрес недоступной памяти, сегментный адрес среды, хвост командной строки, содержимое области среды и путь к модулю;

**Функции программы**

* Функция TETR\_TO\_HEX переводит половину байта в символ.
* Функция BYTE\_TO\_HEX переводит байт в AL в два символа в шестн. числа в AX.
* Функция WRD\_TO\_HEX переводит в 16 с/с 16-ти разрядное число в AX - число, DI - адрес последнего символа.

Функция BYTE\_TO\_DEC переводит в 10 с/с, SI - адрес поля младшей цифры.

* Функция GET\_INACCESSIBLE\_MEM достает информацию о количестве недоступной памяти
* Функция GET\_SEGMENT\_ADRESS достает адрес сегмента среды
* Функция GET\_COMMANDLINE\_TAIL достает информацию о хвосте командной строки
* Функция GET\_CONTENT\_AND\_PATH достает информацию о окружении среды и пути к модулю

**Результаты отладки программы**(см. Рис1):

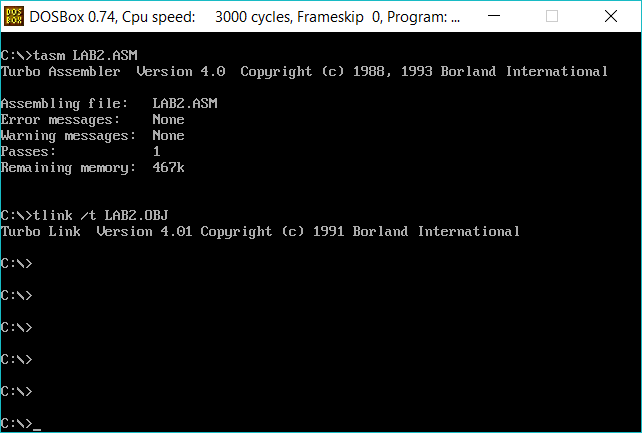


Рис1. Компиляция программы

**Результат работы программы**(см. Рис2):

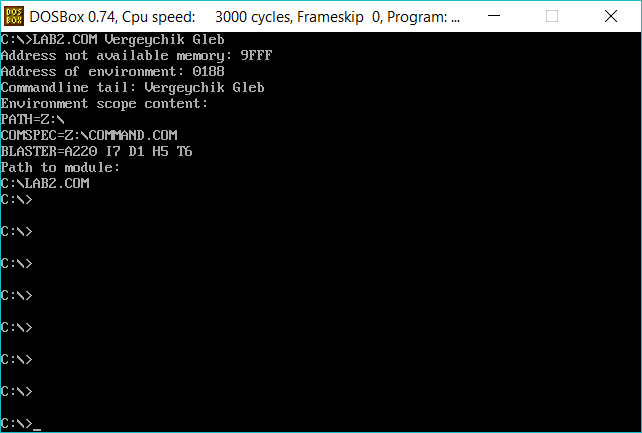


Рис2. Результат работы программы

**Ответы на вопросы**

**Сегментный адрес недоступной памяти**

1. На какую область памяти указывает адрес недоступной памяти?

На область оперативной памяти, которая следует за выделенной для этой программы памятью.

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

За памятью, которая выделена программе.

1. Можно ли в эту область памяти писать?

Можно, потому что DOS не может защитить память.

**Среда, передаваемая программе**

1. Что такое среда?  
   Область с переменными содержащими в себе некоторые настройки операционной системы
2. Когда создается среда? Перед запуском или в другое время?

Создается при загрузке DOS и копируется для программы при ее запуске.

1. Откуда берется информация, записываемая в среду?

Информация, записываемая в среду, берётся из системного файла autoexec.bat.

**Вывод**

В ходе выполнения данной лабораторной работы была исследована структура PSP, а также исследован интерфейс управляющей программы и загрузочных модулей. Также данная лабораторная работа является хорошим средством закрепить в памяти знания о различиях между COM и EXE файлами.

**Приложение**

1. Файл

TESTPC SEGMENT

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

ORG 100H

START: JMP BEGIN

; ДАННЫЕ

ADD\_N db 'Address not available memory: ',0dh,0ah,'$'

ASP\_N db 'Address of environment: ',0dh,0ah,'$'

TAIL db 'Commandline tail:','$'

SOD\_SRED db 'Environment scope content: ' , '$'

PATH db 'Path to module: ' , '$'

ENDL db 0dh,0ah,'$'

NEW\_LINE PROC near

lea dx,ENDL

call Write\_msg

ret

NEW\_LINE ENDP

Write\_msg PROC near

mov ah,09h

int 21h

ret

Write\_msg 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

; 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 ;

pop cx

ret

BYTE\_TO\_HEX ENDP

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

WRD\_TO\_HEX PROC near

; первод 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

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

; Определяем первый байт недоступной памяти

GET\_INACCESSIBLE\_MEM PROC near

push ax

mov ax,es:[2]

lea di,ADD\_N

add di,33

call WRD\_TO\_HEX

pop ax

ret

GET\_INACCESSIBLE\_MEM ENDP

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

; Определяем сегментный адрес среды передаваемой программе

GET\_SEGMENT\_ADRESS PROC near

push ax

mov ax,es:[2Ch]

lea di,ASP\_N

add di,27

call WRD\_TO\_HEX

pop ax

ret

GET\_SEGMENT\_ADRESS ENDP

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

; Определяем хвост командной строки в символьном виде

GET\_COMMANDLINE\_TAIL PROC near

push ax

push cx

xor ax, ax

mov al, es:[80h]

add al, 81h

mov si, ax

push es:[si]

mov byte ptr es:[si+1], '$'

push ds

mov cx, es

mov ds, cx

mov dx, 81h

call Write\_msg

pop ds

pop es:[si]

pop cx

pop ax

ret

GET\_COMMANDLINE\_TAIL ENDP

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

; Определяем содержимое области среды и путь к модулю

GET\_CONTENT\_AND\_PATH PROC near

push es ; Сохраняем

push ax ; изменяемые

push bx ; данные

push cx ; в стеке.

mov bx,1 ; Задаем проверочному регистру, на условие вывода Пути до модуля, значение 1

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

mov si,0 ; Вспомогательному регистру задаем значение 0

RE1:

call NEW\_LINE ; Перенос на новую строчку

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

RE:

cmp byte ptr es:[si], 0 ; Проверяем не 0 ли этот элемент

je NEXT2 ; Как только доходим до конца элемента области среди переходим к метке NEXT2

inc si ; Увеличиваем si на 1

jmp RE ; Прыгаем к метке RE

NEXT2:

push es:[si] ; Сохраняем значение текущей ячейки в стек

mov byte ptr es:[si], '$' ; Присваиваем этой ячейке знак окончания строки

push ds ; Сохраняем занчение регистра ds в стек

mov cx,es ; Заносим в регистр cx значение регистра es

mov ds,cx ; Задаем регистру ds значение регистра cx

mov dx,ax ; Заносим в регистр dx занчение адрес начала строки

call Write\_msg ; Выводим текущую строчку на экран

pop ds ; Возвращаем значение ds

pop es:[si] ; Возвращаем значение текущей ячейки

cmp bx,0 ; Проверка условия о выводе пути до модуля

jz LAST ; Если bx = 0 то переходим к концу процедуры

inc si ; Увеличиваем si на 1

cmp byte ptr es:[si], 01h ; Проверка на то, идет ли дальше информация о пути до модуля

jne RE1 ; Возвращаемся к метке RE1

; call NEW\_LINE ; Перенос строки

lea dx,PATH ; Заносим занчение переменной PATH в dx

call Write\_msg ; Выводим сообщение на экран

mov bx,0 ; Меняем переменную bx на ноль, решая тем самым, что дальше идет адрес пути до модуля

add si,2 ; Пропускаем не нужные символы

jmp RE1 ; Прыгаем к метке RE1

LAST:

; call NEW\_LINE ; Перенос строки

pop cx ; Возвращаем

pop bx ; данные

pop ax ; из

pop es ; стека.

ret

GET\_CONTENT\_AND\_PATH ENDP

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

Write PROC near

mov ah,09h

int 21h

ret

Write ENDP

BEGIN:

call GET\_INACCESSIBLE\_MEM ;Определяем первый байт недоступной памяти

call GET\_SEGMENT\_ADRESS ;и сегментый адрес передаваемой строки

lea dx,ADD\_N

call Write\_msg ;Выводим

; call NEW\_LINE ;на экран

lea dx,ASP\_N ;эти

call Write\_msg ;данные ; call NEW\_LINE

lea dx, TAIL

call Write\_msg ;Выводим слово "Tail:"

call GET\_COMMANDLINE\_TAIL ;Определяем и выводим хвост командной строки в символьном виде

call NEW\_LINE

lea dx,SOD\_SRED

call Write\_msg ;Выводим строку Content of the environment:"

call GET\_CONTENT\_AND\_PATH ;Определяем и выводим содержимое области среды и путь к модулю

; выход в DOS

xor al,al

mov ah, 04Ch

int 21h

ret

TESTPC ENDS

END START