**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МОЭВМ**

отчет

**по лабораторной работе №2**

**по дисциплине «Операционные системы»**

Тема: “ **Исследование интерфейсов программных модулей** ”

|  |  |  |
| --- | --- | --- |
| Студент гр. 7381 |  | Ильясов А.В. |
| Преподаватель |  | Ефремов М.А. |

Санкт-Петербург

2019

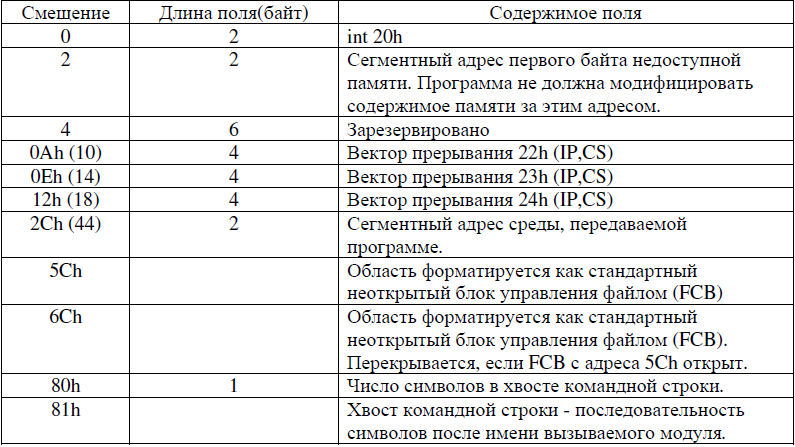
**Цель работы.**

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.

**Основные теоретические положения.**

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. При загрузке модуля типа **.EXE** сегментные регистры DS и ES указывают на PSP. Именно по этой причине значения этих регистров в модуле **.EXE** следует переопределять.

Формат PSP:

Таблица 1 – структура PSP

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, PROMPT, 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.

**Результат выполнение работы.**

Рисунок 1 – результат запуска файла lab2.com

**Выводы.**

В ходе работы было проведено исследование интерфейса управляющей программы и загрузочных модулей, а также исследование префикса сегмента программы (PSP) и среды, передаваемой программе.

**Ответы на контрольные вопросы.**

**Сегментный адрес недоступной памяти**

1) На какую область памяти указывает адрес недоступной памяти?

Ответ: адрес указывает на область оперативной памяти, находящейся сразу после выделенной для программы памятью.

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

Ответ: этот адрес располагается сразу за концом памяти, отведённой программе.

3) Можно ли в эту область памяти писать?

Ответ: можно, так как в DOS нет механизма защиты памяти.

**Среда передаваемая программе**

1) Что такое среда?

Ответ: среда – это область памяти, в которой в виде символьных строк записаны значения переменных (имя=параметр), называемых переменными среды. Они содержат данные о некоторых директориях операционной системы и конфигурации компьютера, которые передаются программе, когда она запускается.

2) Когда создается среда? Перед запуском приложения или в другое время?

Ответ: среда создаётся при загрузке DOS. При запуске программы эта среда только копируется в новую область памяти.

3) Откуда берется информация, записываемая в среду?

Ответ: информация берется из системного файла autoexec.bat.

ПРИЛОжение а

исходный текст .Сom модуля

TESTPC SEGMENT

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

ORG 100H

START: JMP BEGIN

SAofInaccessibleMemory db 'The segment address of the inaccessible memory (from PSP): ', 0DH, 0AH, '$'

SAofEnvironment db 0DH, 0AH, 'The segment address of the environment passed to the program: ', 0DH, 0AH, '$'

TailOfComandLine db 0DH, 0AH,'The tail of comand line: ', 0DH, 0AH, '$'

ContentsOfEnvironment db 0DH, 0AH, 'The contents of the environment:', 0DH, 0AH, '$'

PathOfModule db 0DH, 0AH, 0AH, 'The path of the loaded module:', 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

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

xor ah, ah

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

push ax

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 ax, 00h

jbe end\_l

or al, 30h

mov [si], al

end\_l:

pop ax

pop dx

pop cx

ret

BYTE\_TO\_DEC ENDP

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

;ПРОЦЕДУРЫ ДЛЯ ОПРЕДЕЛЕНИЯ ДАННЫХ

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

FindSAofInaccessibleMemory PROC NEAR

push ax

push di

mov ax, ds:[02h]

mov di, offset SAofInaccessibleMemory

add di, 3Eh

call WRD\_TO\_HEX

pop di

pop ax

ret

FindSAofInaccessibleMemory ENDP

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

FindSAofEnvironment PROC NEAR

push ax

push di

mov ax, ds:[02Ch]

mov di, offset SAofEnvironment

add di, 43h

call WRD\_TO\_HEX

pop di

pop ax

ret

FindSAofEnvironment ENDP

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

FindTailOfComandLine PROC NEAR

push ax

push cx

push dx

push si

push di

xor cx, cx

mov si, 80h

mov ch, byte ptr cs:[si]

mov di, offset TailOfComandLine

add di, 1Bh

inc si

Copy:

cmp ch, 0h

je StopCopy

xor ax, ax

mov al, byte ptr cs:[si]

mov [di], al

inc di

inc si

dec ch

jmp Copy

StopCopy:

xor ax, ax

mov al, 0Ah

mov [di], al

inc di

mov al, '$'

mov [di], al

pop di

pop si

pop dx

pop cx

pop ax

ret

FindTailOfComandLine ENDP

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

FindContOfEnvirAndPathOfMod PROC NEAR

push ax

push dx

push ds

push es

mov dx, offset ContentsOfEnvironment

call PRINT

mov ah, 02h

mov es, ds:[02Ch]

xor si, si

CopyContents:

mov dl, es:[si]

int 21h

cmp dl, 0h

je StopCopyContents

inc si

jmp CopyContents

StopCopyContents:

inc si

mov dl, es:[si]

cmp dl, 0h

jne CopyContents

mov dx, offset PathOfModule

call PRINT

add si, 3h

mov ah, 02h

mov es, ds:[2Ch]

CopyPath:

mov dl, es:[si]

cmp dl, 0h

je StopCopyPath

int 21h

inc si

jmp CopyPath

StopCopyPath:

pop es

pop ds

pop dx

pop ax

ret

FindContOfEnvirAndPathOfMod ENDP

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

PRINT PROC NEAR

push ax

mov ah, 09h

int 21h

pop ax

ret

PRINT ENDP

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

BEGIN:

;1) Сегментный адрес недоступной памяти, взятый из PSP

call FindSAofInaccessibleMemory

mov dx, offset SAofInaccessibleMemory

call PRINT

;2) Сегментный адрес среды, передаваемой программе

call FindSAofEnvironment

mov dx, offset SAofEnvironment

call PRINT

;3) Хвост командной строки

call FindTailOfComandLine

mov dx, offset TailOfComandLine

call PRINT

;4) Содержимое области среды в символьном виде и 5) Путь загружаемого модуля

call FindContOfEnvirAndPathOfMod

xor al, al

mov ah, 4ch

int 21h

TESTPC ENDS

END START