**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 |  | Герасимова Д.В. |
| Преподаватель |  | Губкин А.Ф. |

Санкт-Петербург

2018

**Цель работы:**

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

**Постановка задачи:**

Необходимо написать и отладить программный модуль типа .COM, 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.

**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 поле |
| 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.

**В программе используются следующие процедуры:**

|  |  |
| --- | --- |
| **Название процедуры** | **Назначение** |
| TETR\_TO\_HEX | Перевод половины байта в шестнадцатеричную систему счисления |
| BYTE\_TO\_HEX | Перевод байта регистра AL в шестнадцатеричную систему счисления, помещая результат в AX |
| WRD\_TO\_HEX | Перевод двух байт регистра AX в шестнадцатеричную систему счисления, помещая результат в регистр DI |
| BYTE\_TO\_DEC | Перевод байта регистра AL в десятичную систему счисления, помещая результат в SI |
| SEGMENT\_INACCESS | Определение сегментного адреса недоступной памяти |
| SEGMENT\_ENVIRONMENT | Определение сегментного адреса среды |
| TAIL | Определение хвоста командной строки |
| CONTENT | Определение содержимого области среды |
| PATH | Определение пути загружаемого модуля |
| PRINT | Вывод на экран |

**Определение структуры данных:**

|  |  |  |
| --- | --- | --- |
| **Название поля данных** | **Тип** | **Назначение** |
| \_endl | db | Новая строка |
| \_seg\_inaccess | db | Сегментный адрес недоступной памяти |
| \_seg\_env | db | Сегментный адрес среды |
| \_tail | db | Хвост командной строки |
| \_env | db | Содержимое области среды |
| \_dir | db | Путь загружаемого модуля |
| \_symb | db | Нет символов |

**Результат работы программы:**

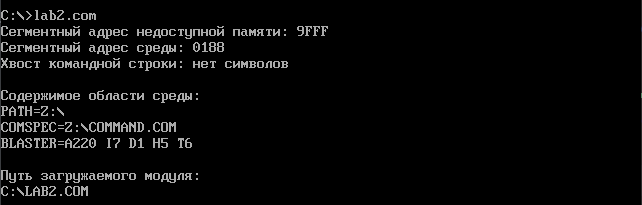


Рис. 1

**Ответы на контрольные вопросы:**

**Сегментный адрес недоступной памяти.**

1. На какую область памяти указывает адрес недоступной памяти?

**Ответ:** адрес недоступной памяти указывает на область оперативной памяти, следующей сразу за выделенной для этой программы памятью. Данная область недоступна для загрузки программ и выделения памяти для них вследствие чего называется недоступной.

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

**Ответ:** он располагается сразу после области памяти, отведённой программе.

1. Можно ли в эту область памяти писать?

**Ответ:** можно, т.к. MS DOS имеет общее адресное пространство и не имеет надёжного механизма управления памятью, поэтому любая программа может воспользоваться данной памятью.

**Среда передаваемая программе.**

1. Что такое среда?

**Ответ:** среда – область памяти, в которой в виде символьных строк (имя=параметр) записаны значения переменных, называемых переменными средами и содержащих данные о некоторых директориях операционной системы и конфигурации компьютера.

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

**Ответ:** при загрузке DOS; при запуске программы происходит лишь копирование среды в новую область памяти.

1. Откуда берется информация, записываемая в среду?

**Ответ:** из системного файла AUTOEXEC.BAT.

**Заключение:**

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

**Приложение**

**lab2.asm**

TESTPC SEGMENT

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

org 100H ; использовать смещение 100h (256 байт) от начала

; сегмента, в который загружена наша программа

START: JMP BEGIN ; START - точка входа

; ДАННЫЕ:

; дополнительные данные

EOF EQU '$'

\_endl db ' ',0DH,0AH,'$' ; новая строка

\_seg\_inaccess db 'Сегментный адрес недоступной памяти: ',0DH,0AH,EOF

\_seg\_env db 'Сегментный адрес среды: ',0DH,0AH,EOF

\_tail db 'Хвост командной строки: ', EOF

\_env db 'Содержимое области среды:',0DH,0AH,EOF

\_dir db 'Путь загружаемого модуля:',0DH,0AH,EOF

\_symb db 'нет символов',0DH,0AH,EOF

; 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

;байт AL переводится в два символа шестн. числа в AX

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 ;в AL - старшая, в AH - младшая

pop CX

ret

BYTE\_TO\_HEX ENDP

;перевод в 16 с/с 16-ти разрядного числа

;в AX - число, DI - адрес последнего символа

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

;перевод в 10с/с, SI - адрес поля младшей цифры

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

; функция определения сегментного адреса недоступной памяти

SEGMENT\_INACCESS PROC NEAR

push ax

push di

mov ax, ds:[02h] ; загружаем адрес

mov di, offset \_seg\_inaccess

add di, 40 ; загружаем адрес последнего символа \_seg\_inacces

call WRD\_TO\_HEX ; переводим ax в 16СС

pop di

pop ax

ret

SEGMENT\_INACCESS ENDP

; функция определения сегментного адреса среды, передаваемого программе

SEGMENT\_ENVIRONMENT PROC NEAR

push ax

push di

mov ax, ds:[2Ch] ; загружаем адрес

mov di, offset \_seg\_env

add di, 27 ; загружаем адрес последнего символа \_seg\_env

call WRD\_TO\_HEX

pop di

pop ax

ret

SEGMENT\_ENVIRONMENT ENDP

; функция определяет хвост программной строки в символьном виде

TAIL PROC NEAR

push ax

push cx

push dx

push si

push di

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

mov si, 81h

mov di, offset \_tail

add di, 20

CopyCmd:

cmp ch, 0h

je NoCmd ; если число символов в хвосте командной строки = 0

;No NoCmd

mov al, ds:[si] ; копируем в di очередной элемент (по адресу si)

mov [di], al ; хвоста командной строки

inc di ; смещаем адреса si и di

inc si ; на один символ вправо

dec ch ; --ch

jmp CopyCmd ; циклически копируем командную строку

NoCmd:

mov al, 0h

mov [di], al

mov dx, offset \_symb

call PRINT

pop di

pop si

pop dx

pop cx

pop ax

ret

TAIL ENDP

; функция определяет содержимое области среды

CONTENT PROC NEAR

push ax

push dx

push ds

push es

; вывод содержимого области среды

mov dx, offset \_env

call PRINT

mov ah, 02h ; будем выводить посимвольно dl

mov es, ds:[2Ch]

xor si, si

WriteCont:

mov dl, es:[si]

int 21h ; вывод

cmp dl, 0h ; проверяем на конец строки

je EndOfLine

inc si ; переходим к рассмотрению след. символа

jmp WriteCont

EndOfLine:

mov dx, offset \_endl ; прыжок на новую строчку

call PRINT

inc si

mov dl, es:[si]

cmp dl, 0h ; проверяем на конец содержимого области среды (если два подряд 0 байта)

jne WriteCont

mov dx, offset \_endl

call PRINT

pop es

pop ds

pop dx

pop ax

ret

CONTENT ENDP

; вывод пути загружаемого модуля

PATH PROC NEAR

push ax

push dx

push ds

push es

mov dx, offset \_dir

call PRINT

add si, 3h

mov ah, 02h

mov es, ds:[2Ch]

WriteDir:

mov dl, es:[si]

cmp dl, 0h

je EndOfDir

int 21h

inc si

jmp WriteDir

EndOfDir:

pop es

pop ds

pop dx

pop ax

ret

PATH ENDP

; функция вывода на экран

PRINT PROC NEAR

push ax

mov ah, 09h

int 21h

pop ax

ret

PRINT ENDP

; КОД

BEGIN:

call SEGMENT\_INACCESS

mov dx, offset \_seg\_inaccess

call PRINT

call SEGMENT\_ENVIRONMENT

mov dx, offset \_seg\_env

call PRINT

mov dx, offset \_tail

call PRINT

call TAIL

mov dx, offset \_endl

call PRINT

call CONTENT

call PATH

mov dx, offset \_endl

call PRINT

; выход в DOS

xor al, al

mov ah, 4ch

int 21h

TESTPC ENDS

END START ; конец модуля