**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра MOЭBM**

отчет

**по лабораторной работе №7**

**по дисциплине «Операционные системы»**

**Тема: Построение модуля оверлейной структуры**

|  |  |  |
| --- | --- | --- |
| Студент гр. 6381 |  | Сергухин В.Ю. |
| Преподаватель |  | Губкин А. Ф. |

Санкт-Петербург

2018

**Цель работы**

Исследование возможности построения загрузочного модуля оверлейной структуры. Приложение в данной работе состоит из нескольких модулей, все модули находятся в одном каталоге и полный путь в этот каталог берется из среды.

**Описание функций**

|  |  |
| --- | --- |
| **Название процедуры** | **Описание** |
| PRINT | Выводит на экран строку |
| FreeSpaceInMemory | Процедура для освобождения места в памяти. Уменьшает отведённый программе блок памяти |
| FindPath | Процедура, формирующая путь до файла оверлея и записывающая этот путь в переменную OvlPath |
| SizeOfOverlay | Процедура, читающая размер файла оверлея, переводящая его в параграфы и выделяющая память, согласно вычисленному размеру, для загрузки оверлея |
| CallOverlay | Процедура, загружающая оверлей в выделенную память, вызывающая оверлейную программу и освобождающая отведённую память |

**Ход работы:**

**Шаг 1:**

Запуск программы, когда текущим каталогом является каталог с разработанными модулями lab7, overlay1 и overlay2 (см. Рис1).

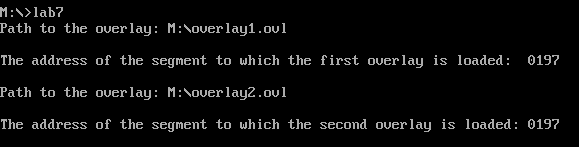


Рис.1 Запуск программы, когда текущим каталогом является каталог с разработанными модулями lab7, overlay1 и overlay2

**Шаг 2:**

Запуск программы, когда текущим каталогом является каталог, отличный от того, в котором находятся модули lab7,first и second.(см. Рис2).

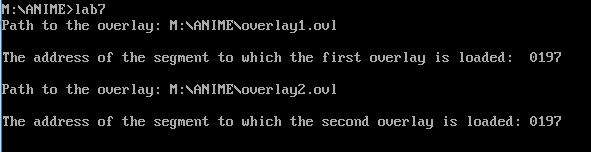


Рис. 2 Запуск программы, когда текущим каталогом является каталог, отличный от того, в котором находятся модули lab7, overlay1 и overlay2.

**Шаг 3:**

Запуск программы, когда в каталоге нет второго оверлея (см. Рис3).

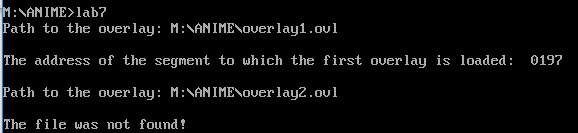


Рис. 3 Запуск программы, когда в каталоге нет второго оверлея.

**Ответы на контрольные вопросы:**

*Как должна быть устроена программа, если в качестве оверлейного сегмента использовать .СОМ модули?*

Так как в com модуле присутствует свой PSP, то его вызов необходимо производить по смещению 100h, также необходимо сохранять регистры и восстанавливать их по завершению работы com модуля.

**Вывод**

В процессе выполнения данной лабораторной работы была исследована возможность построения загрузочного модуля оверлейной структуры.

**Приложение A: Код программы lab7.exe**

ASSUME CS:CODE, DS:DATA, ES:DATA, SS:STACK

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

STACK SEGMENT STACK

DW 64 DUP (?)

STACK ENDS

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

DATA SEGMENT

Mem\_7 DB 0DH, 0AH,'Memory control unit destroyed!',0DH,0AH,'$'

Mem\_8 DB 0DH, 0AH,'Not enough memory to perform the function!',0DH,0AH,'$'

Mem\_9 DB 0DH, 0AH,'Wrong address of the memory block!',0DH,0AH,'$'

OvlPath DB 64 dup (0), '$'

DTA DB 43 DUP (?)

KEEP\_PSP DW 0

Path\_To DB 'Path to the overlay: ','$'

File\_2 DB 0DH, 0AH,'The file was not found!',0DH,0AH,'$'

File\_3 DB 0DH, 0AH,'The route was not found!',0DH,0AH,'$'

Err\_alloc DB 0DH, 0AH,'Failed to allocate memory to load overlay!',0DH,0AH,'$'

SegAdr DW 0

CallAdr DD 0

Load\_1 DB 0DH, 0AH,'The overlay was not been loaded: a non-existent function!',0DH,0AH,'$'

Load\_2 DB 0DH, 0AH,'The overlay was not been loaded: file not found!',0DH,0AH,'$'

Load\_3 DB 0DH, 0AH,'The overlay was not been loaded: route not found!',0DH,0AH,'$'

Load\_4 DB 0DH, 0AH,'The overlay was not been loaded: too many open files!',0DH,0AH,'$'

Load\_5 DB 0DH, 0AH,'The overlay was not been loaded: no access!',0DH,0AH,'$'

Load\_8 DB 0DH, 0AH,'The overlay was not been loaded: low memory!',0DH,0AH,'$'

Load\_10 DB 0DH, 0AH,'The overlay was not been loaded: incorrect environment!',0DH,0AH,'$'

Ovl1 DB 'OVL1.ovl',0

Ovl2 DB 'OVL2.ovl',0

DATA ENDS

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

CODE SEGMENT

START: JMP MAIN

;ПРОЦЕДУРЫ

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

PRINT PROC NEAR ;печать на экран

push ax

mov ah, 09h

int 21h

pop ax

ret

PRINT ENDP

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

FreeSpaceInMemory PROC ;освобождает память для загрузки оверлеев

mov bx,offset LAST\_BYTE ;кладём в ax адрес конца программы

mov ax,es ;es-начало

sub bx,ax ;bx=Размер=Конец-начало

mov cl,4h

shr bx,cl ;переводим в параграфы

;освобождаем место в памяти

mov ah,4Ah ;функция позволяет уменьшить отведённый программе блок памяти

int 21h

jnc NO\_ERROR ;CF=0 - если нет ошибки

;oбработка ошибок CF=1 AX = код ошибки если CF установлен

cmp ax,7 ;разрушен управляющий блок памяти

mov dx,offset Mem\_7

je YES\_ERROR

cmp ax,8 ;недостаточно памяти для выполнения функции

mov dx,offset Mem\_8

je YES\_ERROR

cmp ax,9 ;неверный адрес блока памяти

mov dx,offset Mem\_9

YES\_ERROR:

call PRINT ;выводим ошибку на экран

xor al,al

mov ah,4Ch

int 21H

NO\_ERROR:

ret

FreeSpaceInMemory ENDP

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

FindPath PROC ;ищем путь к файлу оверлея

push ds

push dx

mov dx, seg DTA ;DS:DX = адрес для DTA

mov ds, dx

mov dx,offset DTA ;устанавливаем DS:DX на строку, содержащую имя файла с оверлеем

mov ah,1Ah ;установить адрес DTA

int 21h

pop dx

pop ds

push es ;вычисление пути

push dx

push ax

push bx

push cx

push di

push si

mov es, keep\_PSP ;восстанавливаем PSP

mov ax, es:[2Ch] ;сегментный адрес среды, передаваемой программе

mov es, ax

xor bx, bx ;очищение bx

CopyContents:

mov al, es:[bx] ;берём очередной символ

cmp al, 0h ;проверяем на то что это конец строки

je StopCopyContents ;если конец строки

inc bx ;для перехода к следующему символу

jmp CopyContents

StopCopyContents:

inc bx ;для перехода к следующему символу

cmp byte ptr es:[bx], 0h ;проверяем на то что это конец строки

jne CopyContents ;если всё таки не конец строки (не 2 0-х байта подряд)

add bx, 3h ;два 0-х байта, а потом 00h и 01h

mov si, offset OvlPath

CopyPath: ;путь

mov al, es:[bx] ;берём очередной символ

mov [si], al ;копируем

inc si ;переходим к следующему символу

cmp al, 0h ;проверяем на то что это конец строки

je StopCopyPath

inc bx ;переходим к следующему символу

jmp CopyPath

StopCopyPath:

sub si, 9h ;вычитаем названия программы lab7.exe+1(т.к. мы перешли на следующий после)

mov di, bp ;в di - название оверлея

EntryWay: ;перемещение

mov ah, [di] ;перемещаем символ названия оверлея в ah

mov [si], ah ;записываем в путь

cmp ah, 0h ;если конец имени

je StopEntryWay ;заканчиваем запись

inc di

inc si

jmp EntryWay

StopEntryWay:

mov dx, offset Path\_To

call PRINT

mov dx, offset OvlPath

call PRINT

pop si

pop di

pop cx

pop bx

pop ax

pop dx

pop es

ret

FindPath ENDP

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

SizeOfOverlay PROC ;читает размер файла оверлея и запрашивает объём памяти, достаточный для загрузки

push ds

push dx

push cx

xor cx, cx ;cx - значение байта атрибутов, которое для файла имеет значение 0

mov dx, seg OvlPath ;DS:DX = адрес строки ASCIIZ с именем файла

mov ds, dx

mov dx, offset OvlPath

mov ax,4E00h ;функция 4E прерывания 21h

int 21h ;найти 1-й совпадающий файл

jnc FileFound ;переход, если CF=0, если нет ошибки

cmp ax,3

je Error3

mov dx, offset File\_2 ;файл не найден

jmp exitFileEr

Error3:

mov dx, offset File\_3 ;маршрут не найден

exitFileEr:

call PRINT

pop cx

pop dx

pop ds

xor al,al

mov ah,4Ch

int 21H

FileFound: ;если файл был найден

push es

push bx

mov bx, offset DTA ;смещение на DTA

mov dx,[bx+1Ch] ;старшее слово размера памяти в байтах

mov ax,[bx+1Ah] ;младшее слово размера файла

mov cl,4h ;переводим в параграфы младшее слово

shr ax,cl

mov cl,12

sal dx, cl ;переводим в байты и параграфы

add ax, dx ;складываем

inc ax ;взять большее целое число параграфов

mov bx,ax ;в bx - количество памяти

mov ah,48h ;распределить память (дать размер памяти)

int 21h

jnc MemoryAlloc ;перейти, если CF=0, значит память выделена

mov dx, offset Err\_alloc ;выводим сообщение об ошибке

call PRINT

xor al,al

mov ah,4Ch

int 21H

MemoryAlloc:

mov SegAdr, ax ;сохраняем сегментный адрес распределенного блока

pop bx

pop es

pop cx

pop dx

pop ds

ret

SizeOfOverlay ENDP

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

CallOverlay PROC ;файл оверлейного сегмента загружается и выполняется

push dx

push bx

push ax

mov bx, seg SegAdr ;ES:BX = адрес EPB (EXEC Parameter Block - блока параметров EXEC)

mov es, bx

mov bx, offset SegAdr

mov dx, seg OvlPath ;DS:DX = адрес строки ASCIIZ с именем файла, содержащего программу

mov ds, dx

mov dx, offset OvlPath

mov ax, 4B03h ;загружаем программный оверлей

int 21h

push dx

jnc IsLoad ;переход, если CF=0, значит нет ошибок

cmp ax, 1 ;несуществующий файл

je Er1

cmp ax, 2 ;файл не найден

je Er2

cmp ax, 3 ;маршрут не найден

je Er3

cmp ax, 4 ;слишком много открытых файлов

je Er4

cmp ax, 5 ;нет доступа

je Er5

cmp ax, 8 ;мало памяти

je Er8

cmp ax, 10 ;неправильная среда

je Er10

jmp NoEr

Er1:

mov dx, offset Load\_1

call PRINT

jmp NoEr

Er2:

mov dx, offset Load\_2

call PRINT

jmp NoEr

Er3:

mov dx, offset Load\_3

call PRINT

jmp NoEr

Er4:

mov dx, offset Load\_4

call PRINT

jmp NoEr

Er5:

mov dx, offset Load\_5

call PRINT

jmp NoEr

Er8:

mov dx, offset Load\_8

call PRINT

jmp NoEr

Er10:

mov dx, offset Load\_10

call PRINT

jmp NoEr

IsLoad:

mov AX,DATA ;восстанавливаем ds

mov DS,AX

mov ax, SegAdr

mov word ptr CallAdr+2, ax

call CallAdr ;вызываем оверлейную программу

mov ax, SegAdr

mov es, ax

mov ax, 4900h ;освободить распределенный блок памяти

int 21h

mov AX,DATA ;восстанавливаем ds

mov DS,AX

NoEr:

pop dx

mov es, keep\_PSP

pop ax

pop bx

pop dx

ret

CallOverlay ENDP

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

MAIN:

mov AX,DATA

mov DS,AX

mov KEEP\_PSP, ES

call FreeSpaceInMemory ;1)освобождает лишнюю память для загрузки оверлея

mov bp, offset Ovl1

call FindPath;ищем путь к файлу оверлея

call SizeOfOverlay ;2)читает размер файла оверлея и запрашивает объём памяти, достаточный для загрузки

call CallOverlay ;3)файл оверлейного сегмента загружается и выполняется

;4)освобождается память, отведённая для оверлейного сегмента

mov bp, offset Ovl2

call FindPath;ищем путь к файлу оверлея

call SizeOfOverlay ;2)читает размер файла оверлея и запрашивает объём памяти, достаточный для загрузки

call CallOverlay ;3)файл оверлейного сегмента загружается и выполняется

; ;4)освобождается память, отведённая для оверлейного сегмента

xor al,al

mov ah,4Ch ;выход

int 21h

LAST\_BYTE:

CODE ENDS

END START

**Приложение B: Код программы overlay1.ovl**

ASSUME CS:OVL1,DS:OVL1,SS:NOTHING,ES:NOTHING

OVL1 SEGMENT

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

MAIN1 PROC FAR

push ds

push dx

push di

push ax

mov ax,cs

mov ds,ax

mov bx, offset ForPrint

add bx, 47h

mov di, bx

mov ax, cs

call WRD\_TO\_HEX

mov dx, offset ForPrint

call PRINT

pop ax

pop di

pop dx

pop ds

retf

MAIN1 ENDP

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

PRINT PROC NEAR ;печать на экран

push ax

mov ah, 09h

int 21h

pop ax

ret

PRINT ENDP

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

TETR\_TO\_HEX PROC near ;половина байт AL переводится в символ шестнадцатиричного числа в AL

and al, 0Fh ;and 00001111 - оставляем только вторую половину al

cmp al, 09 ;если больше 9, то надо переводить в букву

jbe NEXT ;выполняет короткий переход, если первый операнд МЕНЬШЕ или РАВЕН второму операнду

add al, 07 ;дополняем код до буквы

NEXT: add al, 30h ;16-ричный код буквы или цифры в al

ret

TETR\_TO\_HEX ENDP

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

BYTE\_TO\_HEX PROC near ;байт AL переводится в два символа шестнадцатиричного числа в AX

push cx

mov ah, al ;копируем al в ah

call TETR\_TO\_HEX ;переводим al в символ 16-рич.

xchg al, ah ;меняем местами al и ah

mov cl, 4

shr al, cl ;cдвиг всех битов al вправо на 4

call TETR\_TO\_HEX ;переводим al в символ 16-рич.

pop cx

ret

BYTE\_TO\_HEX ENDP

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

WRD\_TO\_HEX PROC near ;регистр AX переводится в шестнадцатеричную систему, DI - адрес последнего символа

push bx

mov bh, ah ;копируем ah в bh, т.к. ah испортится при переводе

call BYTE\_TO\_HEX ;переводим al в два символа шестнадцатиричного числа в AX

mov [di], ah ;пересылка содержимого регистра ah по адресу, лежащему в регистре DI

dec di

mov [di], al ;пересылка содержимого регистра al по адресу, лежащему в регистре DI

dec di

mov al, bh ;копируем bh в al, восстанавливаем значение ah

xor ah, ah ;очищаем ah

call BYTE\_TO\_HEX ;переводим al в два символа шестнадцатиричного числа в AX

mov [di], ah ;пересылка содержимого регистра al по адресу, лежащему в регистре DI

dec di

mov [di], al ;пересылка содержимого регистра al по адресу, лежащему в регистре DI

pop bx

ret

WRD\_TO\_HEX ENDP

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

ForPrint DB 0DH,0AH, 'The address of the segment to which the first overlay is loaded: ',0DH,0AH,'$'

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

OVL1 ENDS

END

**Приложение C: Код программы overlay2.ovl**

ASSUME CS:OVL2,DS:OVL2,SS:NOTHING,ES:NOTHING

OVL2 SEGMENT

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

MAIN2 PROC FAR

push ds

push dx

push di

push ax

mov ax,cs

mov ds,ax

mov bx, offset ForPrint

add bx, 47h

mov di, bx

mov ax, cs

call WRD\_TO\_HEX

mov dx, offset ForPrint

call PRINT

pop ax

pop di

pop dx

pop ds

retf

MAIN2 ENDP

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

PRINT PROC NEAR ;печать на экран

push ax

mov ah, 09h

int 21h

pop ax

ret

PRINT ENDP

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

TETR\_TO\_HEX PROC near ;половина байт AL переводится в символ шестнадцатиричного числа в AL

and al, 0Fh ;and 00001111 - оставляем только вторую половину al

cmp al, 09 ;если больше 9, то надо переводить в букву

jbe NEXT ;выполняет короткий переход, если первый операнд МЕНЬШЕ или РАВЕН второму операнду

add al, 07 ;дополняем код до буквы

NEXT: add al, 30h ;16-ричный код буквы или цифры в al

ret

TETR\_TO\_HEX ENDP

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

BYTE\_TO\_HEX PROC near ;байт AL переводится в два символа шестнадцатиричного числа в AX

push cx

mov ah, al ;копируем al в ah

call TETR\_TO\_HEX ;переводим al в символ 16-рич.

xchg al, ah ;меняем местами al и ah

mov cl, 4

shr al, cl ;cдвиг всех битов al вправо на 4

call TETR\_TO\_HEX ;переводим al в символ 16-рич.

pop cx

ret

BYTE\_TO\_HEX ENDP

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

WRD\_TO\_HEX PROC near ;регистр AX переводится в шестнадцатеричную систему, DI - адрес последнего символа

push bx

mov bh, ah ;копируем ah в bh, т.к. ah испортится при переводе

call BYTE\_TO\_HEX ;переводим al в два символа шестнадцатиричного числа в AX

mov [di], ah ;пересылка содержимого регистра ah по адресу, лежащему в регистре DI

dec di

mov [di], al ;пересылка содержимого регистра al по адресу, лежащему в регистре DI

dec di

mov al, bh ;копируем bh в al, восстанавливаем значение ah

xor ah, ah ;очищаем ah

call BYTE\_TO\_HEX ;переводим al в два символа шестнадцатиричного числа в AX

mov [di], ah ;пересылка содержимого регистра al по адресу, лежащему в регистре DI

dec di

mov [di], al ;пересылка содержимого регистра al по адресу, лежащему в регистре DI

pop bx

ret

WRD\_TO\_HEX ENDP

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

ForPrint DB 0DH,0AH, 'The address of the segment to which the second overlay is loaded: ',0DH,0AH,'$'

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

OVL2 ENDS

END