**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра математического обеспечения и применения ЭВМ**

отчет

**по практической работе №7**

**по дисциплине «Операционные системы»**

**Тема**: **Построение модуля оверлейной структуры**

|  |  |  |
| --- | --- | --- |
| Студент гр. 8381 |  | Киреев К.А. |
| Преподаватель |  | Ефремов М.А. |

Санкт-Петербург

2020

**Цель работы.**

Исследование возможности построение загрузочного модуля оверлейной структуры. Исследуется структура оверлейного сегмента и способ загрузки и выполнения оверлейных сегментов. Для запуска вызываемого оверлейного модуля используется функция 4B03h прерывания int 21h. Все загруженные и оверлейные модули находятся в одном каталоге.

В этой работе также рассматривается приложение, состоящее из нескольких модулей, поэтому все модули помещаются в один каталог и вызываются с использованием полного пути.

**Выполнение работы.**

Написан текст исходного EXE модуля, который выполняет следующие функции. Освобождает память для загрузки оверлеев. Читает размер файла оверлея и запрашивает объем памяти, достаточный для его загрузки. Файл оверлейного сегмента загружается и выполняется. Освобождается память, отведенная для оверлейного сегмента. Затем предыдущие действия выполняются для следующего оверлейного сегмента. Также были написаны и отлажены оверлейные сегменты.

Полученный модуль был отлажен. Далее отлаженная программа была запущена. Результаты выполнения программы представлены на рис. 1.

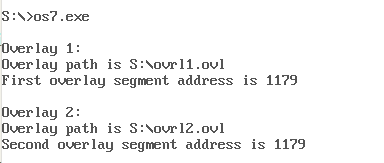


Рисунок 1 – Результат выполнения OS6.EXE.

Видно, что оверлеи запускаются с одного адреса.

Далее программа была запущена из другого каталога. Результаты выполнения представлены на рис. 2.

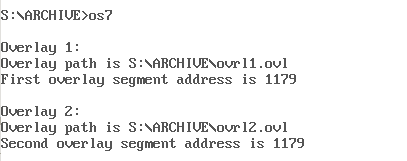


Рисунок 2 – Результат выполнения OS6.EXE из другого каталога.

Далее отлаженная программа была запущена, когда одного оверлея не было в каталоге. Результаты выполнения программы представлены на рис. 3.

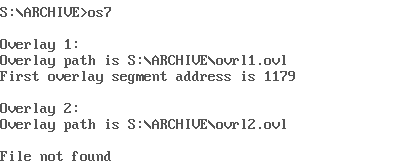


Рисунок 3 – Результат выполнения OS6.EXE, когда одного оверлея нет в каталоге.

**Контрольные вопросы**

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

При использовании в качестве оверлейного сегмента .COM модуля, необходимо вызывать его по смещению 100h, так как в .COM файлах код располагается с адреса 100h. В ином случае PSP не будет сформирован.

**Вывод.**

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

**ПРИЛОЖЕНИЕ А**

**ИСХОДНЫЙ КОД ПРОГРАММЫ. OS7.ASM**

Astack segment stack

dw 256 dup(?)

Astack ends

data segment

dos\_4ah\_err7 db 13,10,'Memory control block is destroyed',13,10,'$'

dos\_4ah\_err8 db 13,10,'Not enough memory to execute function',13,10,'$'

dos\_4ah\_err9 db 13,10,'Invalid memory block address',13,10,'$'

dos\_4eh\_err2 db 13,10,'File not found$',13,10,'$'

dos\_4eh\_err3 db 13,10,'Route not found$',13,10,'$'

dos\_48h\_err db 13,10,'Failed to allocate memory to load overlay!',13,10,'$'

dos\_4b03h\_err db 13,10,'Overlays did not load',13,10,'$'

dos\_4b03h\_err1 db 13,10,'Function number is invalid',13,10,'$'

dos\_4b03h\_err2 db 13,10,'File not found',13,10,'$'

dos\_4b03h\_err3 db 13,10,'File not found',13,10,'$'

dos\_4b03h\_err4 db 13,10,'File not found',13,10,'$'

dos\_4b03h\_err5 db 13,10,'Disk error',13,10,'$'

dos\_4b03h\_err8 db 13,10,'Insufficient memory',13,10,'$'

dos\_4b03h\_err10 db 13,10,'Wrong environment string',13,10,'$'

keep\_psp dw 0

mesto db 16 dup (0)

overlays\_path db 64 dup (0),'$'

DTA db 43 DUP (?)

overlay\_seg\_address dw 0

overlay\_prs dd 0

ovrl1 db 'ovrl1.ovl',0

ovrl2 db 'ovrl2.ovl',0

msg\_ovrl1 db 13,10,'Overlay 1:',13,10,'$'

msg\_ovrl2 db 13,10,'Overlay 2:',13,10,'$'

msg\_overlay\_path db 'Overlay path is ','$'

data ends

code segment

assume cs:code, ds:data, ss:Astack

pushall macro

irp case, <ax,bx,cx,dx,si,di,es,ds,sp,bp>

push &case&

endm

endm

popall macro

irp case, <bp,sp,ds,es,di,si,dx,cx,bx,ax>

pop &case&

endm

endm

print proc near

push ax

mov ah, 09h

int 21h

pop ax

ret

print endp

word\_to\_str proc near

;на входе ax число 16 бит

;si указатель на строку

;bx разрядность результата

pushall

cmp bx, 16

ja end\_wts

cmp ax, 7FFFh

jna plus

mov byte ptr [si], '-'

inc si

not ax

inc ax

plus:

xor cx, cx

jmp manipulation

manipulation:

xor dx, dx

div bx

mov di, ax

mov al, dl

cmp al, 10

sbb al, 69h

das

push di

lea di, mesto

add di, cx

mov byte ptr [di], al

pop di

mov ax, di

inc cx

test ax, ax

jz endrep

jmp manipulation

endrep:

lea di, mesto

add di, cx

copyrep:

dec di

mov dl, byte ptr [di]

mov byte ptr [si], dl

inc si

loop copyrep

end\_wts:

popall

ret

word\_to\_str endp

prep proc near

pushall

mov bx, offset zseg

mov ax, es

sub bx, ax

mov cl, 4

shr bx, cl

mov ah, 4Ah

int 21h

jnc exec\_dos\_4ah

irpc case, 789

cmp ax, &case&

je type\_dos\_4ah\_err&case&

endm

irpc case, 789

type\_dos\_4ah\_err&case&:

lea dx, dos\_4ah\_err&case&

call print

mov ax, 4C00h

int 21h

endm

exec\_dos\_4ah:

popall

ret

prep endp

prep\_overlay\_path proc

push ds

push dx

mov dx, seg DTA

mov ds, dx

lea dx, DTA

;Dos Fn 1Ah Int 21h: позволяет определить адрес DTA для последующих операций

;ds:dx = адрес DTA

mov ah, 1Ah

int 21h

pop dx

pop ds

pushall

mov es, keep\_psp

mov es, es:[2Ch]

xor si, si

lea di, overlays\_path

skip\_env:

mov dl, es:[si]

cmp dl, 00

je end\_env

inc si

jmp skip\_env

end\_env:

inc si

mov dl, es:[si]

cmp dl, 00

jne skip\_env

add si, 3

path\_change:

mov dl, es:[si]

cmp dl, 00

je final\_name

mov [di], dl

inc si

inc di

jmp path\_change

final\_name:

mov si, bp

child\_name:

mov dl, byte ptr [si]

mov byte ptr [di-7], dl

inc di

inc si

test dl, dl

jne child\_name

popall

ret

prep\_overlay\_path endp

prep\_overlay\_size proc

pushall

xor cx, cx

mov dx, seg overlays\_path

mov ds, dx

lea dx, overlays\_path

;Dos Fn 4Eh Int 21h: нахождение первого файла, соответсвующего заданной спецификации

;cx=атрибуты искомых файлов

;ds:dx=адрес спецификации искомого файла

mov ah,4Eh

int 21h

jnc exec\_dos\_4eh

irpc case, 23

cmp ax, &case&

je type\_dos\_4eh\_err&case&

endm

irpc case, 23

type\_dos\_4eh\_err&case&:

lea dx, dos\_4eh\_err&case&

call print

mov ax, 4C00h

int 21h

endm

exec\_dos\_4eh:

mov si, offset DTA

mov ax, [si+1Ah]

mov bx, [si+1Ch]

mov cl, 4

shr ax, cl

mov cl, 12

shl bx, cl

add ax, bx

add ax, 2

mov bx, ax

;Dos Fn 48h Int 21h: выделение блока памяти

;bx=требуемое число параграфов памяти

;на выходе: ax=сегментный адрес выделенного блока

mov ah, 48h

int 21h

jnc exec\_dos\_48h

lea dx, dos\_48h\_err

call print

mov ax, 4C00h

int 21h

exec\_dos\_48h:

mov overlay\_seg\_address, ax

popall

ret

prep\_overlay\_size endp

load\_overlay proc

pushall

push ss

push sp

mov bx, seg overlay\_seg\_address

mov es, bx

lea bx, overlay\_seg\_address

mov dx, seg overlays\_path

mov ds, dx

lea dx, overlays\_path

;Dos Fn 4803h Int 21h: загрузка оверлея в отведенную область памяти

;ds:dx=строка, содержащая путь к оверлею

;es:bx=указатель на блок параметров

mov ax, 4B03h

int 21h

jnc exec\_dos\_4b03h

lea dx, dos\_4b03h\_err

call print

irp case, <1,2,3,4,5,8,10>

cmp ax, &case&

je type\_dos\_4b03h\_err&case&

endm

irp case, <1,2,3,4,5,8,10>

type\_dos\_4b03h\_err&case&:

lea dx, dos\_4b03h\_err&case&

call print

jmp OVL\_RET

endm

exec\_dos\_4b03h:

mov ax, overlay\_seg\_address

mov word ptr overlay\_prs+2, ax

call overlay\_prs

mov es, ax

;Dos Fn 49h Int 21h: освобождение блока памяти

;es=сегментный адрес освобождаемого блока

mov ax, 4900h

int 21h

OVL\_RET:

mov es, keep\_psp

pop sp

pop ss

popall

ret

load\_overlay endp

main proc far

push ds

xor ax, ax

push ax

mov ax, DATA

mov ds, ax

mov keep\_psp, es

call prep

lea dx, msg\_ovrl1

call print

lea bp, ovrl1

call prep\_overlay\_path

lea dx, msg\_overlay\_path

call print

lea dx, overlays\_path

call print

call prep\_overlay\_size

call load\_overlay

lea dx, msg\_ovrl2

call print

lea bp, ovrl2

call prep\_overlay\_path

lea dx, msg\_overlay\_path

call print

lea dx, overlays\_path

call print

call prep\_overlay\_size

call load\_overlay

mov ax, 4C00h

int 21h

ret

main endp

code ends

zseg segment

zseg ends

end main

**ПРИЛОЖЕНИЕ Б**

**ИСХОДНЫЙ КОД ПРОГРАММЫ. OVRL1.ASM**

ovrl1 segment

assume cs:ovrl1

main: jmp processing

info db 'First overlay segment address is ',13,10,'$'

mesto db 16 dup (0)

processing proc far

push ax

push bx

push dx

push ds

push si

mov ax, cs

mov ds, ax

lea si, info+33

mov bx, 16

call word\_to\_str

lea dx, info

call print

pop si

pop ds

pop dx

pop bx

pop ax

retf

processing endp

print proc near

push ax

mov ah, 09h

int 21h

pop ax

ret

print endp

word\_to\_str proc near

;на входе ax число 16 бит

;si указатель на строку

;bx разрядность результата

push ax

push bx

push cx

push dx

push di

push si

cmp bx, 16

ja end\_wts

cmp ax, 7FFFh

jna plus

mov byte ptr [si], '-'

inc si

not ax

inc ax

plus:

xor cx, cx

jmp manipulation

manipulation:

xor dx, dx

div bx

mov di, ax

mov al, dl

cmp al, 10

sbb al, 69h

das

push di

lea di, mesto

add di, cx

mov byte ptr [di], al

pop di

mov ax, di

inc cx

test ax, ax

jz endrep

jmp manipulation

endrep:

lea di, mesto

add di, cx

copyrep:

dec di

mov dl, byte ptr [di]

mov byte ptr [si], dl

inc si

loop copyrep

end\_wts:

pop si

pop di

pop dx

pop cx

pop bx

pop ax

ret

word\_to\_str endp

ovrl1 ends

end main

**ПРИЛОЖЕНИЕ В**

**ИСХОДНЫЙ КОД ПРОГРАММЫ. OVRL2.ASM**

ovrl2 segment

assume cs:ovrl2

main: jmp processing

info2 db 'Second overlay segment address is ',13,10,'$'

mesto db 16 dup (0)

processing proc far

push ax

push bx

push dx

push ds

push si

mov ax, cs

mov ds, ax

lea si, info2+34

mov bx, 16

call word\_to\_str

lea dx, info2

call print

pop si

pop ds

pop dx

pop bx

pop ax

retf

processing endp

print proc near

push ax

mov ah, 09h

int 21h

pop ax

ret

print endp

word\_to\_str proc near

;на входе ax число 16 бит

;si указатель на строку

;bx разрядность результата

push ax

push bx

push cx

push dx

push di

push si

cmp bx, 16

ja end\_wts

cmp ax, 7FFFh

jna plus

mov byte ptr [si], '-'

inc si

not ax

inc ax

plus:

xor cx, cx

jmp manipulation

manipulation:

xor dx, dx

div bx

mov di, ax

mov al, dl

cmp al, 10

sbb al, 69h

das

push di

lea di, mesto

add di, cx

mov byte ptr [di], al

pop di

mov ax, di

inc cx

test ax, ax

jz endrep

jmp manipulation

endrep:

lea di, mesto

add di, cx

copyrep:

dec di

mov dl, byte ptr [di]

mov byte ptr [si], dl

inc si

loop copyrep

end\_wts:

pop si

pop di

pop dx

pop cx

pop bx

pop ax

ret

word\_to\_str endp

ovrl2 ends

end main