

МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МОЭВМ

ОТЧЕТ
по лабораторной работе №7
по дисциплине «Операционные системы»
Тема: «Построение модуля динамической структуры»

Студент гр. 7381

Адамов Я.В.

Преподаватель

Ефремов М.А.

Санкт-Петербург

2019

Цель работы.

Исследование возможности построения 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 загрузочные 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.

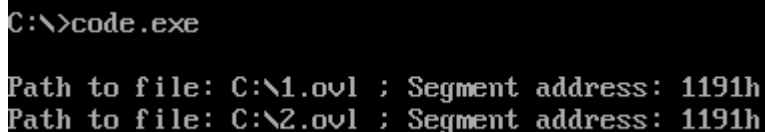
Описание функций.

Название функции	Описание
PrintMsg	Печать строки, адрес которой помещен в DX.
TETR_TO_HEX	Перевод 4-битного числа в код символа в 16 с/с. Вспомогательная функция для BYTE_TO_HEX.
BYTE_TO_HEX	Байт в AL переводится в два символа шестн. числа в AX.
WRD_TO_HEX	Перевод в 16 с/с 16-ти разрядного числа, в AX – число, DI – адрес последнего символа.
DTAset	Устанавливает адреса DTA блока.
MEMORY_CLEAN	Освобождение лишней памяти.
FINDINGname	Извлечение полного имени файла из среды.
addMEMovl	Выделение памяти для оверлея.
OVLrun	Вызов программы оверлея.

Описание работы утилиты.

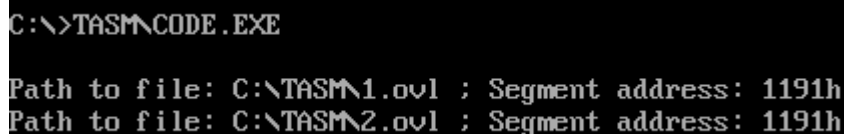
Программа освобождает память для загрузки оверлеев, после чего находит путь к ним. Вычислив размер оверлея и выделив память для его загрузки, программа запускает его. После выполнения происходит освобождение памяти, отведённой для оверлейного сегмента.

Демонстрация работы программы представлена на рис. 1-3.



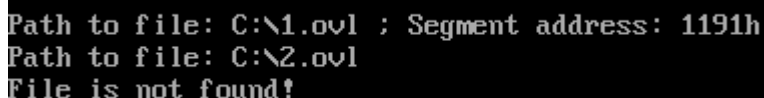
```
C:\>code.exe  
Path to file: C:\1.ovl ; Segment address: 1191h  
Path to file: C:\2.ovl ; Segment address: 1191h
```

Рисунок 1 – работа программы.



```
C:\>TASM\CODE.EXE  
Path to file: C:\TASM\1.ovl ; Segment address: 1191h  
Path to file: C:\TASM\2.ovl ; Segment address: 1191h
```

Рисунок 2 – запуск из другого каталога.



```
Path to file: C:\1.ovl ; Segment address: 1191h  
Path to file: C:\2.ovl  
File is not found!
```

Рисунок 3 – запуск при отсутствии одного из файлов.

Вывод.

В ходе выполнения лабораторной работы была исследована возможности построения загрузочного модуля оверлейной структуры.

Ответы на контрольные вопросы.

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

В .COM модулях код начинается с адреса 100h, следовательно, при использовании в качестве оверлейного сегмента .COM модуля необходимо вызывать его по смещению 100h.

Приложение А. Код программы.

```
AStack SEGMENT STACK
```

```
    DW 100h DUP(?)
```

```
AStack ENDS
```

```
; _____
```

```
DATA SEGMENT
```

```
    PATH_ db 'Path to file: $'
```

```
    NAME_ db 64 DUP(0)
```

```
    DTA_BLOCK db 43 DUP(0)
```

```
    SEGMENTofOVL dw 0
```

```
    ADDRESSofOVL dd 0
```

```
    KEEP_PSP dw 0
```

```
    fileERR1 db 0DH,0AH,'File is not found! ',0DH,0AH,'$'
```

```
    fileERR2 db 0DH,0AH,'Path is not found! ',0DH,0AH,'$'
```

```
    numberERR db 0DH,0AH,'Incorrect number! ',0DH,0AH,'$'
```

```
    ERROfFILE db 0DH,0AH,'File is not found! ',0DH,0AH,'$'
```

```
    diskERR db 0DH,0AH,'Disk error! ',0DH,0AH,'$'
```

```
    memERR db 0DH,0AH,'Not enough memory! ',0DH,0AH,'$'
```

```
    numberERR0 db 0DH,0AH,'Incorrect environment! ',0DH,0AH,'$'
```

```
    mcb_ERR      db      0DH,0AH,      'Block      of      memory      is  
destroyed! ',0DH,0AH,'$'
```

```
    mem_ERR_func db      0DH,0AH,      'Not      enough      memory      for  
function! ',0DH,0AH,'$'
```

```
    adr_ERR      db      0DH,0AH,      'Wrong      adress      of      the      block      of  
memory! ',0DH,0AH,'$'
```

```
    addmemERR db 'Error by adding memory! ',0DH,0AH,'$'
```

```
OVL_1 db '1.ovl',0
OVL_2 db '2.ovl',0
ENDSTR db 0DH,0AH,'$'
```

```
DATA ENDS
```

```
;_____
```

```
CODE SEGMENT
```

```
    ASSUME CS:CODE, DS:DATA, ES:DATA, SS:AStack
```

```
PrintMsg PROC near
```

```
    push ax
    mov ah,09h
    int 21h
    pop ax
    ret
```

```
PrintMsg ENDP
```

```
DTAset PROC
```

```
    push dx
    lea dx,DTA_BLOCK
    mov ah,1Ah
    int 21h
    pop dx
```

```
DTAset ENDP
```

```

MEMORY_CLAEN PROC
    lea bx, LAST_BYTE
    mov ax, es
    sub bx, ax
    mov cl, 4
    SHR bx, cl
    mov ah, 4Ah
    int 21h
    jnc correctSTEP
    cmp ax, 7
    lea dx, mcb_ERR
    je MEM_ERROR
    cmp ax, 8
    lea dx, adr_ERR
    je MEM_ERROR
    cmp ax, 9
    lea dx, adr_ERR
MEM_ERROR:
    call PrintMsg
    xor al, al
    mov ah, 4Ch
    int 21h
correctSTEP:
    ret
MEMORY_CLAEN ENDP

```

FINDINGname PROC

```
push es
mov es,es:[2Ch]
xor si,si
lea di,NAME_
```

STEP1:

```
inc si
cmp word ptr es:[si],0000h
jne STEP1
add si,4
```

STEP2:

```
cmp byte ptr es:[si],00h
je STEP3
mov dl,es:[si]
mov [di],dl
inc si
inc di
jmp STEP2
```

STEP3:

```
dec si
dec di
cmp byte ptr es:[si],'\ '
jne STEP3

inc di
```



```
mov si,bx
push ds
pop es
```

STEP4:

```
lodsb
stosb
cmp al,0
jne STEP4
mov byte ptr [di],'$'
lea dx,PATH_
call PrintMsg
lea dx,NAME_
call PrintMsg
pop es
ret
```

FINDINGname ENDP

addMEMov1 PROC

```
push ds
push dx
push cx
XOR cx,cx
lea dx,NAME_
mov ah,4Eh
INT 21h
JNC WAY2
```

```
cmp ax,3
lea dx,fileERR2
je WAY1
lea dx,fileERR1
```

WAY1:

```
call PrintMsg
pop cx
pop dx
pop ds
xor al,al
mov ah,4Ch
int 21h
```

WAY2:

```
push es
push bx
lea bx,DTA_BLOCK
mov dx,[bx+1Ch]
mov ax,[bx+1Ah]
mov cl,4h
SHR ax,cl
mov cl,12
SAL dx,cl
add ax,dx
inc ax
mov bx,ax
mov ah,48h
```

```
int 21h
jc WAY3
mov SEGMENTofOVL,ax
pop bx
pop es
pop cx
pop dx
pop ds
ret
```

WAY3:

```
lea dx,addmemERR
call PrintMsg
mov ah,4Ch
int 21h
```

addMEMovl ENDP

OVLrun PROC

```
push dx
push bx
push ax
mov bx, SEG SEGMENTofOVL
mov es,bx
lea bx,SEGMENTofOVL
lea dx,NAME_
mov ax,4B03h
int 21h
```

```
jnc GOODway  
call CHECKING  
jmp QUITov1
```

GOODway:

```
mov ax,DATA  
mov ds,ax  
mov ax,SEGMENTofOVL  
mov word ptr ADDRESSofOVL+2, ax  
call ADDRESSofOVL  
mov ax,SEGMENTofOVL  
mov es,ax  
mov ax,4900h  
int 21h  
mov ax,DATA  
mov ds,ax
```

QUITov1:

```
mov es,KEEP_PSP  
pop ax  
pop bx  
pop dx  
ret
```

OVLrun ENDP

CHECKING PROC

```
cmp ax,1
```

```

    lea dx,numberERR
    je PRINT_ERR
    cmp ax,2
    lea dx,ERROfFILE
    je PRINT_ERR
    cmp ax,5
    lea dx,diskERR
    je PRINT_ERR
    cmp ax,8
    lea dx,memERR
    je PRINT_ERR
    cmp ax,10
    lea dx,numberERR0
PRINT_ERR:
    call PrintMsg
    ret
CHECKING ENDP

```

```

BEGIN:
    mov ax,DATA
    mov ds,ax
    mov Keep_psp,es
    lea dx,ENDSTR
    call PrintMsg
    call MEMORY_CLAEN
    call DTASET
    lea bx,OVL_1

```

```
call FINDINGname  
call addMEMovl  
call OVLrun  
lea bx,OVL_2  
call FINDINGname  
call addMEMovl  
call OVLrun  
mov ah,4Ch  
int 21h
```

LAST_BYTE:

CODE ENDS

END BEGIN

Приложение Б. Код оверлейного сегмента.

OVL SEGMENT

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

BEGIN PROC FAR

push ds

push ax

push di

push dx

push bx

mov ds,ax

lea bx,cs:mes

add bx,22

mov di, bx

mov ax, cs

call WRD_TO_HEX

lea dx, cs:mes

call PrintMsg

pop bx

pop dx

pop di

pop ax

pop ds

retf

BEGIN ENDP

```
mes db '; Segment address:      h',0DH,0AH,'$'
```

```
PrintMsg PROC near
```

```
    push ax
```

```
    mov ah,09h
```

```
    int 21h
```

```
    pop ax
```

```
    ret
```

```
PrintMsg 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
```

```
    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
    call BYTE_TO_HEX
    mov [di],ah
    dec di
    mov [di],al
    pop bx
    ret
WRD_TO_HEX ENDP
```

```
OVL ENDS
    END BEGIN
```