

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

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

Студентка гр.7381

\_\_\_\_\_

Машина Ю.Д.

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 int21h. 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.

### **Функции и структуры данных управляющей программы**

Таблица 1 – Функции управляющей программы

Имя функции	Описание функции
Print	Функция выводит на экран строку.
Errors	Функция обработки ошибок при вызове функции 4Ah прерывания int 21h для освобождения места в памяти.
Clear_Memory	Функция освобождения памяти. В случае возникновения ошибок при освобождении памяти вызывается функция Errors.
Path	Функция для формирования пути к оверлею. Данная функция вызывает функции Variables и Path.
ovl_size	Функция определяет размер оверлея с помощью функции 4Eh.

Bhfuncerrors	Функция обработки ошибок при вызове функции 4B03h.
Run_ovl	Функция, которая загружает оверлей.
MAIN PROC	Основная функция.

Таблица 2 – Структура данных управляющей программы

Имя	Тип	Назначение
memryblockdestroyed	db	Вывод строки 'Memory control block destroyed'
notenough	db	Вывод строки 'Not enough memory to perform the function'
wrongaddress	db	Вывод строки 'Wrong memory address'
nonexist	db	Вывод строки 'Error: Non-existent function'
filenf	db	Вывод строки 'Error: File not found'
pathnf	db	Вывод строки 'Error: Path not found'
manyfiles	db	Вывод строки 'Error: Too many opened files'
noaccess	db	Вывод строки 'Error: No access'
notenoughmemry	db	Вывод строки 'Error: Not enough memory'
incenv	db	Вывод строки 'Error: Incorrect environment'
str_overlay1	db	Название первого оверлея
str_overlay2	db	Название второго оверлея
DTA	db	Организация в программе области дисковой передачи данных.
Overlay_Path	db	Путь до оверлея
KEEP_PSP	dw	Переменная для сохранения PSP
OVERLAY_ADDR	dd	Адрес для запуска оверлея

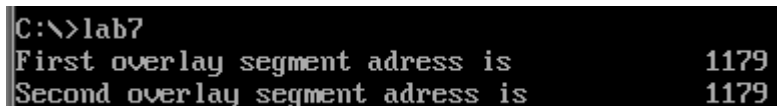
## Ход работы.

Был написан программный модуль типа .EXE, 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.
- Предыдущие действия выполняются для следующего оверлейного сегмента.

Были написаны и отлажены оверлейные сегменты. Оверлейные сегменты выводят адрес сегмента, в который они загружены.

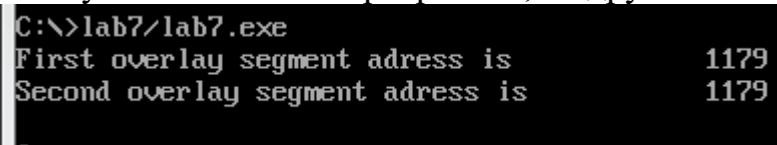
Был выполнен запуск отлаженной программы. Оверлейные сегменты загружаются с одного адреса, перекрывая друг друга.



```
C:\>lab7
First overlay segment adress is      1179
Second overlay segment adress is     1179
```

Рисунок 1 – Запуск программы лабораторной работы №7.

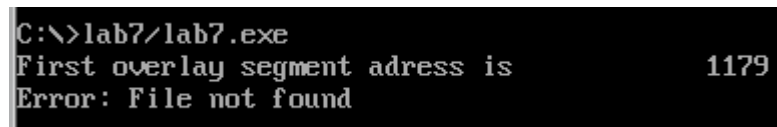
Был выполнен запуск отлаженной программы, из другого каталога.



```
C:\>lab7/lab7.exe
First overlay segment adress is      1179
Second overlay segment adress is     1179
```

Рисунок 2 – Запуск программы лабораторной работы №7.

Был выполнен запуск отлаженной программы, когда одного оверлея нет в каталоге.



```
C:\>lab7/lab7.exe
First overlay segment adress is      1179
Error: File not found
```

Рисунок 3 – Запуск программы лабораторной работы №7.

### **Ответы на контрольные вопросы.**

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

*Ответ:* При обращении к оверлейному сегменту необходимо учитывать смещение 100h, так как. Это связано с тем, что в .COM модуле присутствует PSP.

### **Выводы.**

В ходе выполнения лабораторной работы №7 был построен загрузочный модуль оверлейной структуры, а также оверлей. Изучены дополнительные функции работы с памятью и способы загрузки и выполнения оверлейных сегментов.

## ПРИЛОЖЕНИЕ А

### КОД ИСХОДНОЙ ПРОГРАММЫ

```
AStack SEGMENT STACK
    dw 64 dup(?)
AStack ENDS
```

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

```
DATA SEGMENT
    wrongadress db 'Wrong memory address',13, 10,'$'
    nonexistent db 'Error: Non-existent function', 13, 10, '$'
    manyfiles db 'Error: Too many opened files', 13, 10, '$'
    noaccess db 'Error: No access', 13, 10, '$'
    notenoughmemry db 'Error: Not enough memory', 13, 10, '$'

    incenv db 'Error: Incorrect environment', 13, 10, '$'
    memryblockdestroyed db 'Memory control block destroyed',13, 10,'$'
    notenough db 'Not enough memory to perform the function',13, 10,'$'
    filenf db 'Error: File not found', 13, 10, '$'
    pathnf db 'Error: Path not found', 13, 10, '$'

    str_overlay1 db 'LAB7OVL1.OVL', 0
    str_overlay2 db 'LAB7OVL2.OVL', 0
    DTA db 43 dup (0), '$'
    Overlay_Path db 100h dup (0), '$'
    OVERLAY_ADDR dd 0
    KEEP_PSP dw 0
    Overlay_Address dw 0
DATA ENDS
```

```
Print PROC NEAR
    push ax
    mov ah, 09h
    int 21h
    pop ax
    ret
Print ENDP
```

```
OVL_Path PROC NEAR
    push ax
    push bx
    push cx
    push dx
    push si
    push di
    push es
    mov es, KEEP_PSP
    mov ax, es:[2Ch]
    mov es, ax
    mov bx, 0
```

```
mov cx, 2
call Variables
call Path
```

```
get_way:
mov ah, [di]
mov [si], ah
cmp ah, 0
jz check_way
inc di
inc si
jmp get_way
```

```
check_way:
pop es
pop di
pop si
pop dx
pop cx
pop bx
pop ax
ret
OVL_Path ENDP
```

```
Path PROC NEAR
get_path:
mov al, es:[bx]
mov [si], al
inc si
inc bx
cmp al, 0
jz check_path
jmp get_path
```

```
check_path:
sub si, 9
mov di, bp
ret
Path ENDP
```

```
Clear_Memory PROC NEAR ; освобождение памяти для загрузки оверлеев
mov bx, offset LAST_BYTE
mov ax, es
sub bx, ax
mov cl, 4h
shr bx, cl
mov ah, 4Ah ; освобождение памяти перед загрузкой оверлея
int 21h
jnc end_clear
```

```
call Errors
xor al, al
mov ah, 4Ch
```

```

int 21h
end_clear:
ret
Clear_Memory ENDP

```

```

Errors PROC NEAR
cmp ax,7
mov dx,offset memryblockdestroyed
je print1
cmp ax,8
mov dx,offset notenough
je print1
cmp ax,9
mov dx,offset wrongadress
je print1

```

```

print1:
call Print
ret
Errors ENDP

```

```

Variables PROC NEAR
get_variables:
inc cx
mov al, es:[bx]
inc bx
cmp al, 0
jz check_end
loop get_variables

```

```

check_end:
cmp byte PTR es:[bx], 0
jnz get_variables
add bx, 3
mov si, offset Overlay_Path
ret
Variables ENDP

```

```

ovl_size PROC NEAR ; чтение размера файла оверлея
push bx
push es
push si

```

```

push ds
push dx
mov dx, SEG DTA
mov ds, dx
mov dx, offset DTA
mov ax, 1A00h ; в области памяти буфера DTA со смещением 1Ah будет
находиться младшее слово размера файла
int 21h
pop dx
pop ds

```



```

push ds
push dx
xor cx, cx
mov dx, SEG Overlay_Path
mov ds, dx
mov dx, offset Overlay_Path
mov ax, 4E00h ; определение размера оверлея
int 21h
pop dx
pop ds

jnc no_err_size
cmp ax, 2
je err1
cmp ax, 3
je err2
jmp no_err_size

err1:
mov dx, offset filenf
call Print
jmp exit
err2:
mov dx, offset pathnf
call Print
jmp exit

no_err_size:
push es
push bx
push si
mov si, offset DTA
add si, 1Ch ; в слове со смещением 1Ch в DTA будет находиться старшее
слово размера памяти в байтах
mov bx, [si]

sub si, 2
mov bx, [si]
push cx
mov cl, 4
shr bx, cl
pop cx
mov ax, [si+2]
push cx
mov cl, 12
sal ax, cl
pop cx
add bx, ax
add bx, 2
mov ax, 4800h ; отведение памяти
int 21h
mov Overlay_Adress, ax

```

```
pop si
pop bx
pop es
```

```
exit:
pop si
pop es
pop bx
ret
ovl_size ENDP
```

Bhfuncerrors PROC NEAR ; от обращения к функции 4B03h:

```
cmp ax, 1 ; несуществующая функция
mov dx, offset nonexistent
je print3
cmp ax, 2 ; файл не найден
mov dx, offset filenf
je print3
cmp ax, 3 ; маршрут не найден
mov dx, offset pathnf
je print3
cmp ax, 4 ; слишком много открытых файлов
mov dx, offset manyfiles
je print3
cmp ax, 5 ; нет доступа
mov dx, offset noaccess
je print3
cmp ax, 8 ; мало памяти
mov dx, offset notenoughmemory
je print3
cmp ax, 10 ; неправильная среда
mov dx, offset incenv
je print3
```

```
print3:
call Print
ret
Bhfuncerrors ENDP
```

```
NO_ERROR_RUN PROC NEAR
mov ax, SEG DATA
mov ds, ax
mov ax, Overlay_Adress
mov WORD PTR OVERLAY_ADDR+2, ax
call OVERLAY_ADDR
mov ax, Overlay_Adress
mov es, ax
mov ax, 4900h ; освобождение памяти после отработки оверлея
int 21h
mov ax, SEG DATA
mov ds, ax
ret
NO_ERROR_RUN ENDP
```

```

Run_ovl PROC NEAR
push bp
push ax
push bx
push cx
push dx
mov bx, SEG Overlay_Adress
mov es, bx
mov bx, offset Overlay_Adress

mov dx, SEG Overlay_Path
mov ds, dx
mov dx, offset Overlay_Path
push ss
push sp

mov ax, 4B03h ; для запуска вызываемого оверлейного модуля
int 21h
jnc no_error_way

call Bhfuncerrors
jmp exit_way
no_error_way:
call NO_ERROR_RUN

exit_way:
pop sp
pop ss
mov es, KEEP_PSP
pop dx
pop cx
pop bx
pop ax
pop bp
ret
Run_ovl ENDP

MAIN PROC FAR
mov ax, seg DATA
mov ds, ax
mov KEEP_PSP, es
call Clear_Memory
mov bp, offset str_overlay1
call OVL_Path
call ovl_size
call Run_ovl
mov bp, offset str_overlay2
call OVL_Path
call ovl_size
call Run_ovl

```

```
xor al, al
mov ah, 4Ch
int 21H
ret
MAIN ENDP
CODE ENDS

LAST_BYTE SEGMENT
LAST_BYTE ENDS

END MAIN
```