

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. 6383

Иолшина В.

Преподаватель

Ефремов М.А.

Санкт-Петербург

2019

Цель работы.

Исследование возможности построение загрузочного модуля оверлейной структуры. Также нужно исследовать способ загрузки и выполнения оверлейных сегментов. Для запуска вызываемого оверлейного модуля используется функция 4B03h прерывания int 21h

Таблица 1 - Описание процедур.

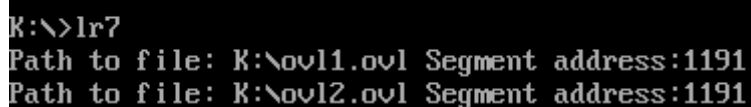
Название процедуры	Назначение
PRINT	вызывает функцию печати строки
DTAset	устанавливает адрес DTA блока
MEMORY_CLEAN	освобождает лишнюю память
FINDINGname	извлекает полное имя файла из среды
addMEMovl	выделяет память для оверлея
OVLrun	вызывает программу оверлея
CHECKING	находит ошибку и выводит сообщение о ней

Действия, выполняемые программой.

1. Освобождение памяти для загрузки оверлеев.
2. Нахождение пути к оверлею.
3. Вычисление размера оверлея и выделение памяти для его загрузки.
4. Загрузка и выполнение оверлейного сегмента.
5. Освобождение памяти, отведённой для оверлейного сегмента.
6. Вышеперечисленные пункты повторяются для второго оверлея.

Результаты выполнения программы представлены на рис. 1-7.

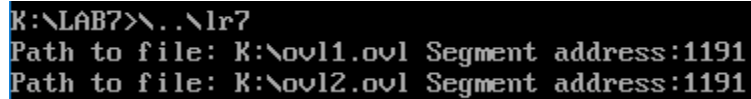
1. Запуск программы lr7.exe изображён на рис. 1.



```
K:\>lr7
Path to file: K:\ovl1.ovl Segment address:1191
Path to file: K:\ovl2.ovl Segment address:1191
```

Рисунок 1 – Результат выполнения программы lr7.exe

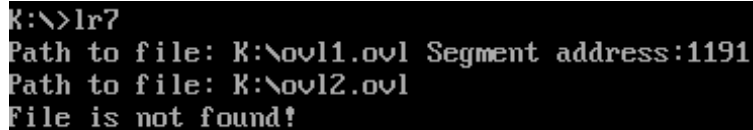
2. На рис. 2 представлен запуск приложения из другого каталога.



```
K:\LAB7>\\.\lr7
Path to file: K:\ovl1.ovl Segment address:1191
Path to file: K:\ovl2.ovl Segment address:1191
```

Рисунок 2 – Результат выполнения программы lr7.exe

3. Запустим программу снова, но при отсутствии в текущем каталоге одного из оверлейных модулей. Действие программы показано на рис. 3.



```
K:\>lr7
Path to file: K:\ovl1.ovl Segment address:1191
Path to file: K:\ovl2.ovl
File is not found!
```

Рисунок 3 – Результат выполнения программы lr7.exe

Выводы.

В процессе выполнения данной лабораторной работы была исследована возможность построения загрузочного модуля оверлейной структуры, а также исследован способ загрузки и выполнения оверлейных сегментов. Было создано приложение, состоящее из нескольких модулей, все модули которого помещаются в один каталог и вызываются с использованием полного пути.

Ответы на контрольные вопросы.

- 1.** 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. Также важно сохранять регистры и восстанавливать их при завершении действия **.COM** модуля.

ПРИЛОЖЕНИЕ А

7.asm

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 'ovl1.ovl', 0
OVL_2          DB 'ovl2.ovl', 0
```

DATA ENDS

STACK SEGMENT STACK

```
DW 100 DUP (0)
```

STACK ENDS

CODE SEGMENT

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

PRINT PROC

```
MOV AH, 09H
INT 21H
RET
```

PRINT 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 PRINT
    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 PRINT
        LEA DX, NAME_
        CALL PRINT
        POP ES
        RET

FINDINGname ENDP

addMEMovl 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 PRINT
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 PRINT
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 QUITovl

```

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

```

QUITovl:

```

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 PRINT
    ret
CHECKING ENDP

BEGIN PROC FAR
    mov ax, DATA
    mov ds, ax
    mov Keep_psp, ES
    CALL MEMORY_CLAEN
    CALL DTAsset
    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:
BEGIN ENDP

CODE ENDS
    END BEGINpa

```

ПРИЛОЖЕНИЕ Б

ovl1.asm

FIRSTOVL segment

ASSUME cs:FIRSTOVL, 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, 19
    mov di, bx
    mov ax, cs
    call WRD_TO_HEX
    lea dx, cs:mes
    call PRINT
    pop bx
    pop dx
    pop di
    pop ax
    pop ds
    retf
```

BEGIN ENDP

```
mes          db      'Segment address:  ', 10, 13, '$'
```

PRINT PROC

```
    mov ah, 09h
    int 21h
    ret
```

PRINT 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

FIRSTOVL ENDS

END BEGIN

ПРИЛОЖЕНИЕ В

ovl2.asm

SECONDOVL segment

ASSUME cs:SECONDOVL, 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, 19
    mov di, bx
    mov ax, cs
    call WRD_TO_HEX
    lea dx, cs:mes
    call PRINT
    pop bx
    pop dx
    pop di
    pop ax
    pop ds
    retf
```

BEGIN ENDP

```
mes                db    'Segment address:  ', 10, 13, '$'
```

PRINT PROC

```
    mov ah, 09h
    int 21h
    ret
```

PRINT 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

SECONDOVL ENDS

END BEGIN