**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**

|  |  |  |
| --- | --- | --- |
| Студент гр. 6383 |  | Никитин К.В. |
| Преподаватель |  | Губкин А.Ф. |

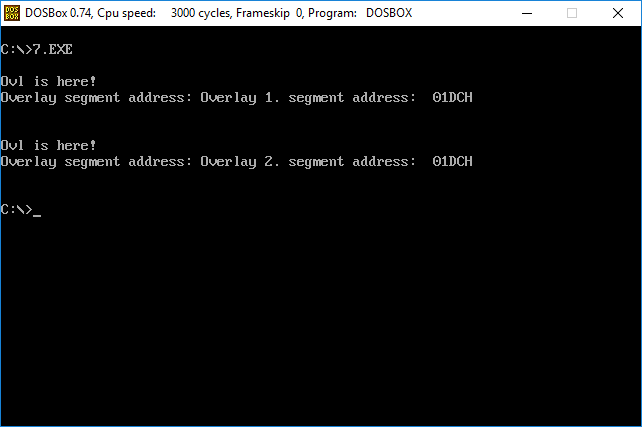
Санкт-Петербург

2018

1. **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.

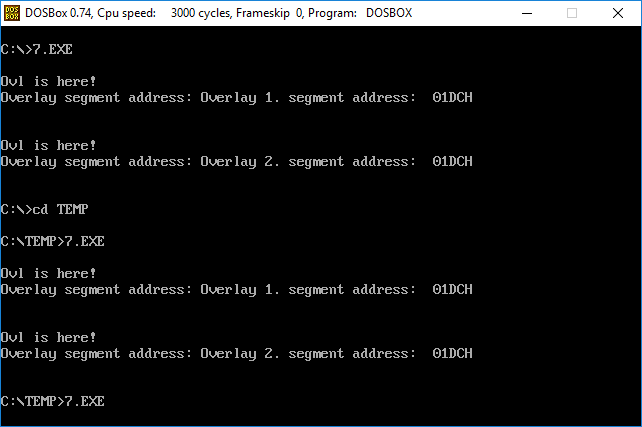
1. **Ход работы.**
2. Запуск **7.EXE**. Оверлейные сегменты загружаются с одного и того же адреса, перекрывая друг друга. (Рис.1)



**Рисунок 1**

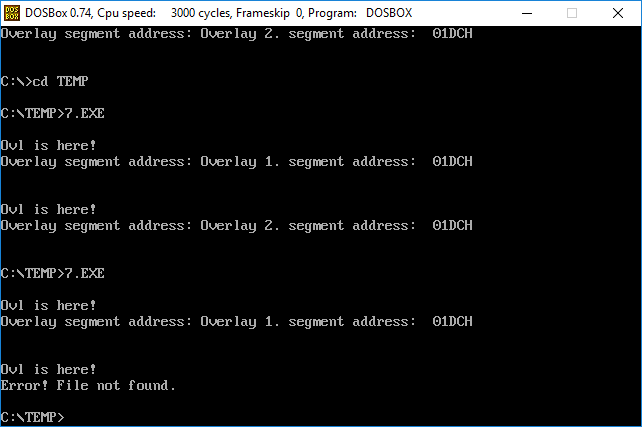
1. Запуск **7.EXE** из другого каталога. Приложение выполнено успешно. (Рис.2)

Результат:



**Рисунок 2**

1. Запуск **7.EXE** в случае, когда одного оверлея нет в каталоге. Приложение завершено аварийно. (Рис.3)



**Рисунок 3**

1. **Ответы на контрольные вопросы**
2. *Как должна быть устроена программа, если в качестве оверлейного сегмента использовать .COM модули?*

Так как в COM модуле присутствует PSP, то нужно обращаться к оверлейному сегменту со смещением на 100h.

1. **Вывод**

В ходе данной лабораторной работы была исследована возможность построение загрузочного модуля оверлейной структуры, а также изучена структура оверлейного сегмента и способ загрузки и выполнения оверлейных сегментов.

**ПРИЛОЖЕНИЕ А**

**LAB7.ASM**

DATA SEGMENT

PSP dw ?

path db 100 dup (?)

dta db 43 dup (?)

overlay dw 0

epb dw ?

\_ss dw ?

\_sp dw ?

Count db 0

OVL\_FOUND db 'Ovl is here! $'

ADRESS db 'Overlay segment address: $'

ERR1 db 'Error! File not found.$'

ERR2 db 'Error! Path not found.$'

OVL1 db 'ovl1.ovl',0

OVL2 db 'ovl2.ovl',0

EMPTYLINE db 13, 10, '$'

DATA ENDS

STACK SEGMENT STACK

DW 100 DUP (0)

STACK ENDS

CODE SEGMENT

ASSUME DS:DATA, SS:STACK, CS:CODE, ES:NOTHING

.386

PRINT proc near

push ax

push dx

mov ah, 09h

int 21h

pop dx

pop ax

ret

PRINT endp

ENDL proc near

push ax

push dx

mov dx, offset EMPTYLINE

mov ah, 09h

int 21h

pop dx

pop ax

ret

ENDL endp

JOBS:

mov ax, data

mov ds, ax

mov PSP, es

mov es, es:[002Ch]

xor bx, bx

STEP:

mov dl, byte PTR es:[bx]

cmp dl, 0h

je FIRST\_

inc bx

jmp STEP

FIRST\_:

inc bx

mov dl, byte PTR es:[bx]

cmp dl, 0h

je SECOND\_

jmp STEP

SECOND\_:

add bx,3

push si

mov si, offset path

STEP1:

mov dl, byte PTR es:[bx]

mov [si], dl

inc si

inc bx

cmp dl, 0

jne STEP1

STEP2:

mov al, [si]

cmp al, '\'

je STEP3

dec si

jmp STEP2

STEP3:

inc si

push di

mov di, offset OVL1

STEP4:

mov ah, [di]

mov [si], ah

inc si

inc di

cmp ah, 0

jne STEP4

mov ah,'$'

mov [si],ah

pop di

call ENDL

mov dx, offset OVL\_FOUND

call PRINT

call ENDL

mov ax, PSP

mov es, ax

mov bx, offset last\_byte

shr bx, 4

add bx, 50

mov ah, 4Ah

int 21h

mov dx, offset dta

mov ah, 1Ah

int 21h

CYCLE:

mov dx, offset path

mov ah, 4Eh

mov cx, 0h

int 21h

jnc NO\_ERROR

cmp ax, 2

jne ERROR1

mov dx, offset ERR1

call PRINT

call ENDL

jmp TODOS

ERROR1:

cmp ax, 3

jne ERROR2

mov dx, offset ERR2

call PRINT

call ENDL

jmp TODOS

ERROR2:

cmp ax, 18

jne TODOS

mov dx, offset ERR1

call PRINT

call ENDL

jmp TODOS

NO\_ERROR:

mov ebx, dword ptr [ offset dta + 1Ah ]

shr ebx, 4

inc ebx

mov ah, 48h

int 21h

mov epb, ax

mov ax, ds

mov es, ax

mov bx, offset epb

mov dx, offset path

mov \_sp, sp

mov \_ss, ss

mov ax, 4B03h

int 21h

mov ax, data

mov ds, ax

mov ss, \_ss

mov sp, \_sp

mov dx, offset ADRESS

call PRINT

push ds

call dword ptr overlay

call ENDL

pop ds

mov ax, epb

mov es, ax

mov ah, 49h

int 21h

mov al, Count

cmp al, 1

je TODOS

call ENDL

mov di, 0

STEP5:

mov al, [si]

cmp al, '\'

je STEP6

dec si

jmp STEP5

STEP6:

inc si

push di

mov di, offset OVL2

STEP7:

mov ah, [di]

mov [si], ah

inc si

inc di

cmp ah, 0

jne STEP7

mov ah,'$'

mov [si],ah

pop di

pop si

mov Count, 1

mov dx, offset OVL\_FOUND

call PRINT

call ENDL

jmp CYCLE

TODOS:

xor al, al

mov ah, 4Ch

int 21h

last\_byte:

CODE ENDS

END JOBS

**ПРИЛОЖЕНИЕ Б**

**OVL1.ASM**

CODE SEGMENT

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

OVERLAY PROC FAR

push AX

push DI

push DS

mov AX,CS

mov DS,AX

mov DI,OFFSET STR\_ADDR + 32

call WORD2HEX

call PUT\_ADDRESS

pop DS

pop DI

pop AX

RETF

OVERLAY ENDP

PUT\_ADDRESS PROC NEAR

push AX

push DX

mov DX,OFFSET STR\_ADDR

mov AH,09H

int 21H

pop DX

pop AX

ret

PUT\_ADDRESS ENDP

TETR2HEX PROC NEAR

and AL,0FH

cmp AL,09

JBE NEXT

add AL,07

NEXT: add AL,30H

ret

TETR2HEX ENDP

BYTE2HEX PROC NEAR

push CX

mov AH,AL

call TETR2HEX

XCHG AL,AH

mov CL,4

shr AL,CL

call TETR2HEX

pop CX

ret

BYTE2HEX ENDP

WORD2HEX PROC NEAR

push BX

mov BH,AH

call BYTE2HEX

mov [DI],AH

dec DI

mov [DI],AL

dec DI

mov AL,BH

call BYTE2HEX

mov [DI],AH

dec DI

mov [DI],AL

pop BX

ret

WORD2HEX ENDP

STR\_ADDR DB 'Overlay 1. segment address: H',0DH,0AH,'$'

CODE ENDS

END OVERLAY

**ПРИЛОЖЕНИЕ В**

**OVL2.ASM**

CODE SEGMENT

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

OVERLAY PROC FAR

push AX

push DI

push DS

mov AX,CS

mov DS,AX

mov DI,OFFSET STR\_ADDR + 32

call WORD2HEX

call PUT\_ADDRESS

pop DS

pop DI

pop AX

RETF

OVERLAY ENDP

PUT\_ADDRESS PROC NEAR

push AX

push DX

mov DX,OFFSET STR\_ADDR

mov AH,09H

int 21H

pop DX

pop AX

ret

PUT\_ADDRESS ENDP

TETR2HEX PROC NEAR

and AL,0FH

cmp AL,09

JBE NEXT

add AL,07

NEXT: add AL,30H

ret

TETR2HEX ENDP

BYTE2HEX PROC NEAR

push CX

mov AH,AL

call TETR2HEX

XCHG AL,AH

mov CL,4

shr AL,CL

call TETR2HEX

pop CX

ret

BYTE2HEX ENDP

WORD2HEX PROC NEAR

push BX

mov BH,AH

call BYTE2HEX

mov [DI],AH

dec DI

mov [DI],AL

dec DI

mov AL,BH

call BYTE2HEX

mov [DI],AH

dec DI

mov [DI],AL

pop BX

ret

WORD2HEX ENDP

STR\_ADDR DB 'Overlay 2. segment address: H',0DH,0AH,'$'

CODE ENDS

END OVERLAY