**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МОЭВМ**

**отчет**

по лабораторной работе №**7**

по дисциплине «Операционные системы»

на тему: Построение модуля оверлейной структуры

|  |  |  |
| --- | --- | --- |
| Студент гр. 6381 |  | Кухарев М.А. |
| Преподаватель |  | Губкин А.Ф. |

Санкт-Петербург

2018

**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.

**Ход выполнения работы:**

***Шаг 1:***

Напишем и отладим программный модуль типа exe, который выполняет функции:

1. Освобождает память для загрузки оверлеев.
2. Читает размер файла оверлея и запрашивает объем памяти, достаточный для его загрузки.
3. Файл оверлейного сегмента загружается и выполняется.
4. Освобождается память, отведенная для оверлейного сегмента.
5. Затем действия 1-4 выполняются для следующего оверлейного сегмента.

***Шаг 2:***

Напишем и отладим оверлейные сегменты. Оверлейный сегмент выводит адрес сегмента, в который он загружен. Имена файлов оверлеев имеют вид .OVL

***Шаг 3:***

Запустим программу.

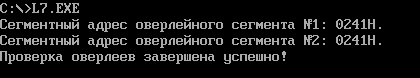


Рис.1. Результат работы программы L7.EXE

Заметим, что оверлеи запускаются с одного и того же адреса.

***Шаг 4:***

Запустим программу из другого каталога.

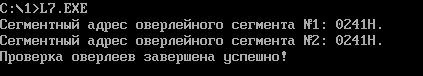


Рис.2. Результат запуска программы L7.EXE из другого каталога

Приложение выполняется успешно.

***Шаг 5:***

Уберем из каталога второй оверлей и запустим программу.

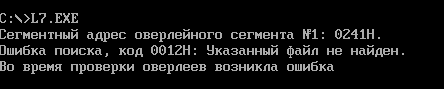


Рис.3. Результат запуска программы L7.EXE, когда в каталоге нет второго оверлея

Программа завершилась аварийно с кодом 0241H.

**Ответы на контрольные вопросы.**

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

**Ответ:** При использовании в качестве оверлейного сегмента .COM модуля, необходимо вызывать его по смещению 100h, так как в .COM файлах код располагается с адреса 100h.

**Приложение A: Код программы L7.ASM**

.SEQ

CODE SEGMENT

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

START: jmp START\_

DATA SEGMENT

PSP\_SIZ = 10h

STK\_SIZ = 10h

OVL\_END\_SUCCESS db 'Проверка оверлеев завершена успешно!', 0Dh, 0Ah, '$'

OVL\_END\_ERROR db 'Во время проверки оверлеев возникла ошибка', 0Dh, 0Ah, '$'

ERR\_FUNC\_48H db 'Ошибка функции 48H прерывания 21H, код ошибки: H.', 0Dh, 0Ah, '$'

ERR\_FUNC\_49H db 'Ошибка функции 49H прерывания 21H, код ошибки: H.', 0Dh, 0Ah, '$'

ERR\_FUNC\_4AH db 'Ошибка функции 4AH прерывания 21H, код ошибки: H.', 0Dh, 0Ah, '$'

ERR\_SIZE\_ABOVE db 'Ошибка: Размер оверлея превышает 1048560 (FFFF0H) байт!', 0Dh, 0Ah, '$'

ERR\_FIND\_02H db 'Ошибка поиска, код 0002H: Указанный путь не существует.', 0Dh, 0Ah, '$'

ERR\_FIND\_12H db 'Ошибка поиска, код 0012H: Указанный файл не найден.', 0Dh, 0Ah, '$'

ERR\_FIND\_UNKNOWN db 'Ошибка поиска, код H', 0Dh, 0Ah, '$'

ERR\_EXE\_01H db 'Ошибка загрузки, код 0001H: Неверный номер подфункции.', 0Dh, 0Ah, '$'

ERR\_EXE\_02H db 'Ошибка загрузки, код 0002H: Указанный файл не найден.', 0Dh, 0Ah, '$'

ERR\_EXE\_03H db 'Ошибка загрузки, код 0003H: Указанный путь не существует.', 0Dh, 0Ah, '$'

ERR\_EXE\_04H db 'Ошибка загрузки, код 0004H: Открыто слишком много файлов.', 0Dh, 0Ah, '$'

ERR\_EXE\_05H db 'Ошибка загрузки, код 0005H: Ошибка доступа к файлу.', 0Dh, 0Ah, '$'

ERR\_EXE\_08H db 'Ошибка загрузки, код 0008H: Не хватает свободной памяти.', 0Dh, 0Ah, '$'

ERR\_EXE\_0AH db 'Ошибка загрузки, код 000AH: Блок среды превышает 32 Кб.', 0Dh, 0Ah, '$'

ERR\_EXE\_0BH db 'Ошибка загрузки, код 000BH: Некорректный формат файла.', 0Dh, 0Ah, '$'

ERR\_EXE\_UNKNOWN db 'Ошибка загрузки, код H', 0Dh, 0Ah, '$'

ABS\_NM1 db 100h dup (?)

OVL\_NM1 db 'LAB7\_OV1.OVL', 00h

ABS\_NM2 db 100h dup (?)

OVL\_NM2 db 'LAB7\_OV2.OVL', 00h

DTA\_BUF db 2Bh dup (?)

OVLN\_IP dw 00h

OVLN\_CS dw 00h

EPB\_DW1 dw 00h

EPB\_DW2 dw 00h

CHR\_EOT = '$'

DATA ENDS

STACK SEGMENT STACK

db STK\_SIZ \* 10h dup (?)

STACK ENDS

TETR\_TO\_HEX PROC NEAR

and AL, 0Fh

cmp AL, 09h

jbe NEXT

add AL, 07h

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, 04h

shr AL, CL

call TETR\_TO\_HEX

pop CX

ret

BYTE\_TO\_HEX ENDP

WRD\_TO\_HEX PROC NEAR

push AX

push BX

push DI

mov BH, AH

call BYTE\_TO\_HEX

mov DS:[DI], AH

dec DI

mov DS:[DI], AL

dec DI

mov AL, BH

call BYTE\_TO\_HEX

mov DS:[DI], AH

dec DI

mov DS:[DI], AL

pop DI

pop BX

pop AX

ret

WRD\_TO\_HEX ENDP

PR\_STR\_BIOS PROC NEAR

push AX

push BX

push CX

push DX

push DI

push ES

mov AX, DS

mov ES, AX

mov AH, 0Fh

int 10h

mov AH, 03h

int 10h

mov DI, 00h

dsbp\_nxt: cmp byte ptr DS:[BP+DI], CHR\_EOT

je dsbp\_out

inc DI

jmp dsbp\_nxt

dsbp\_out: mov CX, DI

mov AH, 13h

mov AL, 01h

int 10h

pop ES

pop DI

pop DX

pop CX

pop BX

pop AX

ret

PR\_STR\_BIOS ENDP

START\_: mov BX, DATA

mov DS, BX

mov BX, STACK

add BX, STK\_SIZ

sub BX, CODE

add BX, PSP\_SIZ

mov AH, 4Ah

int 21h

jc ERR\_4A

jmp PREPARE\_ALL

ERR\_4A: lea DI, ERR\_FUNC\_4AH

add DI, 50

call WRD\_TO\_HEX

mov BL, 07h

lea BP, ERR\_FUNC\_4AH

call PR\_STR\_BIOS

jmp DOS\_QUIT\_ERROR

PREPARE\_ALL: mov AH, 1Ah

lea DX, DTA\_BUF

int 21h

mov SI, 00h

jmp NEXT\_OVERLAY

NEXT\_OVERLAY: inc SI

jmp OVERLAY\_1

OVERLAY\_1: cmp SI, 01h

jne OVERLAY\_2

lea CX, OVL\_NM1

lea DX, ABS\_NM1

jmp PREPARE\_NAM

OVERLAY\_2: cmp SI, 02h

jne OVERLAY\_END

lea CX, OVL\_NM2

lea DX, ABS\_NM2

jmp PREPARE\_NAM

OVERLAY\_END: mov BL, 07h

lea BP, OVL\_END\_SUCCESS

call PR\_STR\_BIOS

jmp DOS\_QUIT

PREPARE\_NAM: push SI

push ES

mov ES, ES:[2Ch]

xor SI, SI

PREPARE\_EEL: cmp word ptr ES:[SI], 0000h

je PREPARE\_LSI

inc SI

jmp PREPARE\_EEL

PREPARE\_LSI: add SI, 04h

mov DI, SI

xor AX, AX

PREPARE\_LSL: cmp byte ptr ES:[DI], 00h

je PREPARE\_CPI

cmp byte ptr ES:[DI], "/"

je PREPARE\_SLS

cmp byte ptr ES:[DI], "\"

je PREPARE\_SLS

jmp PREPARE\_LSN

PREPARE\_SLS: mov AX, DI

PREPARE\_LSN: inc DI

jmp PREPARE\_LSL

PREPARE\_CPI: mov DI, DX

PREPARE\_CPL: cmp SI, AX

ja PREPARE\_CNI

mov BL, ES:[SI]

mov DS:[DI], BL

inc SI

inc DI

jmp PREPARE\_CPL

PREPARE\_CNI: pop ES

mov SI, CX

PREPARE\_CNL: cmp byte ptr DS:[SI], 00h

je FIND\_OVERLAY

PREPARE\_CNS: mov BL, DS:[SI]

mov DS:[DI], BL

inc SI

inc DI

jmp PREPARE\_CNL

FIND\_OVERLAY: pop SI

mov AH, 4Eh

mov CX, 00h

int 21h

jc FIND\_ERR\_02

jmp GET\_SIZE

FIND\_ERR\_02: cmp AX, 02h

jne FIND\_ERR\_12

mov BL, 07h

lea BP, ERR\_FIND\_02H

call PR\_STR\_BIOS

jmp DOS\_QUIT

FIND\_ERR\_12: cmp AX, 12h

jne FIND\_ERR\_UNKNOWN

mov BL, 07h

lea BP, ERR\_FIND\_12H

call PR\_STR\_BIOS

jmp DOS\_QUIT\_ERROR

FIND\_ERR\_UNKNOWN:

lea DI, ERR\_FIND\_UNKNOWN

add DI, 22

call WRD\_TO\_HEX

mov BL, 07h

lea BP, ERR\_FIND\_UNKNOWN

call PR\_STR\_BIOS

jmp DOS\_QUIT\_ERROR

GET\_SIZE: lea BP, DTA\_BUF

mov AX, DS:[BP+1Ah]

mov BX, DS:[BP+1Ch]

xchg DX, BP

mov DX, BX

and DX, 0000000000001111b

cmp DX, BX

jne SIZE\_ERR

mov CL, 0Ch

shl DX, CL

mov DL, AL

and DL, 00001111b

cmp DL, 00000000b

je GET\_PAIR

mov DL, 01h

jmp GET\_PAIR

GET\_PAIR: mov CL, 04h

shr AX, CL

add AX, DX

jc SIZE\_ERR

xchg DX, BP

jmp GET\_FMEM

SIZE\_ERR: mov BL, 07h

lea BP, ERR\_SIZE\_ABOVE

call PR\_STR\_BIOS

jmp DOS\_QUIT\_ERROR

GET\_FMEM: mov BX, AX

mov AH, 48h

int 21h

jc ERR\_48

mov OVLN\_CS, AX

jmp MAKE\_PAR

ERR\_48: lea DI, ERR\_FUNC\_48H

add DI, 50

call WRD\_TO\_HEX

mov BL, 07h

lea BP, ERR\_FUNC\_48H

call PR\_STR\_BIOS

jmp DOS\_QUIT\_ERROR

MAKE\_PAR: mov BX, seg EPB\_DW1

push ES

mov ES, BX

lea BX, EPB\_DW1

mov EPB\_DW1, AX

mov EPB\_DW2, AX

jmp LOAD\_OVERLAY

LOAD\_OVERLAY: mov CH, 01h

mov AH, 4Bh

mov AL, 03h

int 21h

pop ES

jc LOAD\_ERR\_01

mov CH, 00h

jmp EXECUTE\_OVERLAY

LOAD\_ERR\_01: cmp AX, 01h

jne LOAD\_ERR\_02

mov BL, 07h

lea BP, ERR\_EXE\_01H

call PR\_STR\_BIOS

jmp EXECUTE\_FMEM

LOAD\_ERR\_02: cmp AX, 02h

jne LOAD\_ERR\_03

mov BL, 07h

lea BP, ERR\_EXE\_02H

call PR\_STR\_BIOS

jmp EXECUTE\_FMEM

LOAD\_ERR\_03: cmp AX, 03h

jne LOAD\_ERR\_04

mov BL, 07h

lea BP, ERR\_EXE\_03H

call PR\_STR\_BIOS

jmp EXECUTE\_FMEM

LOAD\_ERR\_04: cmp AX, 04h

jne LOAD\_ERR\_05

mov BL, 07h

lea BP, ERR\_EXE\_04H

call PR\_STR\_BIOS

jmp EXECUTE\_FMEM

LOAD\_ERR\_05: cmp AX, 05h

jne LOAD\_ERR\_08

mov BL, 07h

lea BP, ERR\_EXE\_05H

call PR\_STR\_BIOS

jmp EXECUTE\_FMEM

LOAD\_ERR\_08: cmp AX, 08h

jne LOAD\_ERR\_0A

mov BL, 07h

lea BP, ERR\_EXE\_08H

call PR\_STR\_BIOS

jmp EXECUTE\_FMEM

LOAD\_ERR\_0A: cmp AX, 0Ah

jne LOAD\_ERR\_0B

mov BL, 07h

lea BP, ERR\_EXE\_0AH

call PR\_STR\_BIOS

jmp EXECUTE\_FMEM

LOAD\_ERR\_0B: cmp AX, 0Bh

jne LOAD\_ERR\_UNKNOWN

mov BL, 07h

lea BP, ERR\_EXE\_0BH

call PR\_STR\_BIOS

jmp EXECUTE\_FMEM

LOAD\_ERR\_UNKNOWN:

lea DI, ERR\_EXE\_UNKNOWN

add DI, 24

call WRD\_TO\_HEX

mov BL, 07h

lea BP, ERR\_EXE\_UNKNOWN

call PR\_STR\_BIOS

jmp EXECUTE\_FMEM

EXECUTE\_OVERLAY:

call dword ptr DS:[OVLN\_IP]

jmp EXECUTE\_FMEM

EXECUTE\_FMEM: mov AX, OVLN\_CS

push ES

mov ES, AX

mov AH, 49h

int 21h

pop ES

jc ERR\_49

cmp CH, 00h

jne DOS\_QUIT\_ERROR

jmp NEXT\_OVERLAY

ERR\_49: lea DI, ERR\_FUNC\_49H

add DI, 50

call WRD\_TO\_HEX

mov BL, 07h

lea BP, ERR\_FUNC\_49H

call PR\_STR\_BIOS

jmp DOS\_QUIT\_ERROR

DOS\_QUIT\_ERROR:

mov BL, 07h

lea BP, OVL\_END\_ERROR

call PR\_STR\_BIOS

DOS\_QUIT: mov AH, 4Ch

int 21h

CODE ENDS

END START

**Код программы LAB7\_OV1.asm**

OVL\_CODE SEGMENT

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

OVERLAY\_PROC PROC FAR

start: push AX

push DX

push DI

push DS

push ES

mov AX, CS

mov DS, AX

mov ES, AX

lea DI, OVL\_MSG

add DI, 45

call WRD\_TO\_HEX

lea DX, OVL\_MSG

call PRINT\_STR

pop ES

pop DS

pop DI

pop DX

pop AX

retf

OVERLAY\_PROC ENDP

OVL\_MSG db 'Сегментный адрес оверлейного сегмента №1: H.', 0Dh, 0Ah, '$'

TETR\_TO\_HEX PROC NEAR

and AL, 0Fh

cmp AL, 09h

jbe NEXT

add AL, 07h

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, 04h

shr AL, CL

call TETR\_TO\_HEX

pop CX

ret

BYTE\_TO\_HEX ENDP

WRD\_TO\_HEX PROC NEAR

push AX

push BX

push DI

mov BH, AH

call BYTE\_TO\_HEX

mov DS:[DI], AH

dec DI

mov DS:[DI], AL

dec DI

mov AL, BH

call BYTE\_TO\_HEX

mov DS:[DI], AH

dec DI

mov DS:[DI], AL

pop DI

pop BX

pop AX

ret

WRD\_TO\_HEX ENDP

PRINT\_STR PROC NEAR

push AX

mov AH, 09h

int 21h

pop AX

ret

PRINT\_STR ENDP

OVL\_CODE ENDS

END

**Код программы LAB7\_OV2.asm**

OVL\_CODE SEGMENT

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

OVERLAY\_PROC PROC FAR

start: push AX

push DX

push DI

push DS

push ES

mov AX, CS

mov DS, AX

mov ES, AX

lea DI, OVL\_MSG

add DI, 45

call WRD\_TO\_HEX

lea DX, OVL\_MSG

call PRINT\_STR

pop ES

pop DS

pop DI

pop DX

pop AX

retf

OVERLAY\_PROC ENDP

OVL\_MSG db 'Сегментный адрес оверлейного сегмента №2: H.', 0Dh, 0Ah, '$'

TETR\_TO\_HEX PROC NEAR

and AL, 0Fh

cmp AL, 09h

jbe NEXT

add AL, 07h

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, 04h

shr AL, CL

call TETR\_TO\_HEX

pop CX

ret

BYTE\_TO\_HEX ENDP

WRD\_TO\_HEX PROC NEAR

push AX

push BX

push DI

mov BH, AH

call BYTE\_TO\_HEX

mov DS:[DI], AH

dec DI

mov DS:[DI], AL

dec DI

mov AL, BH

call BYTE\_TO\_HEX

mov DS:[DI], AH

dec DI

mov DS:[DI], AL

pop DI

pop BX

pop AX

ret

WRD\_TO\_HEX ENDP

PRINT\_STR PROC NEAR

push AX

mov AH, 09h

int 21h

pop AX

ret

PRINT\_STR ENDP

OVL\_CODE ENDS

END