**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.

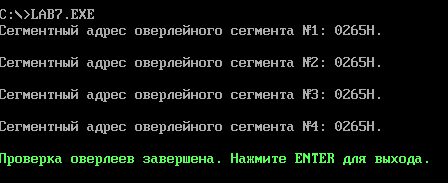


Рис.1. Результат работы программы LAB7.EXE

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

***Шаг 4:***

Запустим программу из другого каталога.

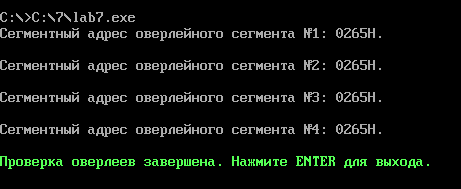


Рис.2. Результат запуска программы LAB7.EXE из другого каталога

Приложение выполняется успешно.

***Шаг 5:***

Уберем из каталога третий оверлей и запустим программу.

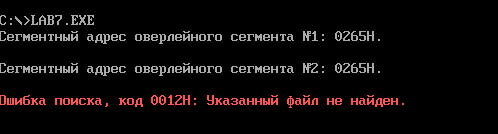


Рис.3. Результат запуска программы LAB7.EXE, когда в каталоге нет второго оверлея

Программа завершилась аварийно с кодом 0012H.

**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 LAB7.ASM**

L7\_CODE SEGMENT

ASSUME CS: L7\_CODE, DS: L7\_DATA, ES: NOTHING, SS: L7\_STACK

START: jmp l7\_start

L7\_DATA SEGMENT

PSP\_SIZ = 10h

STK\_SIZ = 10h

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

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

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

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

OVL\_END db 'Проверка оверлеев завершена. Нажмите ENTER для выхода.', 0Dh, 0Ah, '$'

FND\_E02 db 'Ошибка поиска, код 0002H: Указанный путь не существует.', 0Dh, 0Ah, '$'

FND\_E12 db 'Ошибка поиска, код 0012H: Указанный файл не найден.', 0Dh, 0Ah, '$'

FND\_EUN db 'Ошибка поиска, код H: < Неизвестный код ошибки >', 0Dh, 0Ah, '$'

EXE\_E01 db 'Ошибка загрузки, код 0001H: Неверный номер подфункции.', 0Dh, 0Ah, '$'

EXE\_E02 db 'Ошибка загрузки, код 0002H: Указанный файл не найден.', 0Dh, 0Ah, '$'

EXE\_E03 db 'Ошибка загрузки, код 0003H: Указанный путь не существует.', 0Dh, 0Ah, '$'

EXE\_E04 db 'Ошибка загрузки, код 0004H: Открыто слишком много файлов.', 0Dh, 0Ah, '$'

EXE\_E05 db 'Ошибка загрузки, код 0005H: Ошибка доступа к файлу.', 0Dh, 0Ah, '$'

EXE\_E08 db 'Ошибка загрузки, код 0008H: Не хватает свободной памяти.', 0Dh, 0Ah, '$'

EXE\_E0A db 'Ошибка загрузки, код 000AH: Блок среды превышает 32 Кб.', 0Dh, 0Ah, '$'

EXE\_E0B db 'Ошибка загрузки, код 000BH: Некорректный формат файла.', 0Dh, 0Ah, '$'

EXE\_EUN db 'Ошибка загрузки, код H: < Неизвестный код ошибки >', 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

ABS\_NM3 db 100h dup (?)

OVL\_NM3 db 'LAB7\_OV3.OVL', 00h

ABS\_NM4 db 100h dup (?)

OVL\_NM4 db 'LAB7\_OV4.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 = '$'

MSG\_CLR = 07h

HLP\_CLR = 09h

INF\_CLR = 0Eh

ERR\_CLR = 0Ch

OVL\_CLR = 0Ah

L7\_DATA ENDS

L7\_STACK SEGMENT STACK

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

L7\_STACK ENDS

;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

; Процедуры

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

; Перевод байта из AL в два символа HEX числа в AX

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

; Перевод в HEX слова из AX

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

; Вывод текста

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

;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

; Код программы

; Освобождение неиспользуемой памяти

l7\_start: mov BX, L7\_DATA

mov DS, BX

mov BX, L7\_STACK

add BX, STK\_SIZ

sub BX, L7\_CODE

add BX, PSP\_SIZ

mov AH, 4Ah

int 21h

jc error\_4A

jmp prep\_all

; Вывод информации об ошибке

error\_4A: lea DI, ERR\_4AH

add DI, 50

call WRD\_TO\_HEX

mov BL, ERR\_CLR

lea BP, ERR\_4AH

call PR\_STR\_BIOS

jmp dos\_quit

; Установка адреса блока

prep\_all: mov AH, 1Ah

lea DX, DTA\_BUF

int 21h

mov SI, 00h

jmp next\_ovl

; Выбор имени очередного файла оверлея

next\_ovl: inc SI

jmp name\_ov1

name\_ov1: cmp SI, 01h

jne name\_ov2

lea CX, OVL\_NM1

lea DX, ABS\_NM1

jmp prep\_nam

name\_ov2: cmp SI, 02h

jne name\_ov3

lea CX, OVL\_NM2

lea DX, ABS\_NM2

jmp prep\_nam

name\_ov3: cmp SI, 03h

jne name\_ov4

lea CX, OVL\_NM3

lea DX, ABS\_NM3

jmp prep\_nam

name\_ov4: cmp SI, 04h

jne name\_eol

lea CX, OVL\_NM4

lea DX, ABS\_NM4

jmp prep\_nam

name\_eol: mov BL, OVL\_CLR

lea BP, OVL\_END

call PR\_STR\_BIOS

jmp dos\_quit

; Подготовка абсолютного имени файла

prep\_nam: push SI

push ES

mov ES, ES:[2Ch]

xor SI, SI

prep\_eel: cmp word ptr ES:[SI], 0000h

je prep\_lsi

inc SI

jmp prep\_eel

prep\_lsi: add SI, 04h

mov DI, SI

xor AX, AX

prep\_lsl: cmp byte ptr ES:[DI], 00h

je prep\_cpi

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

je prep\_sls

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

je prep\_sls

jmp prep\_lsn

prep\_sls: mov AX, DI

prep\_lsn: inc DI

jmp prep\_lsl

prep\_cpi: mov DI, DX

prep\_cpl: cmp SI, AX

ja prep\_cni

mov BL, ES:[SI]

mov DS:[DI], BL

inc SI

inc DI

jmp prep\_cpl

prep\_cni: pop ES

mov SI, CX

prep\_cnl: cmp byte ptr DS:[SI], 00h

je find\_ovl

prep\_cns: mov BL, DS:[SI]

mov DS:[DI], BL

inc SI

inc DI

jmp prep\_cnl

; Поиск первого подходящего файла

find\_ovl: pop SI

mov AH, 4Eh

mov CX, 00h

int 21h

jc find\_e02

jmp get\_size

find\_e02: cmp AX, 02h

jne find\_e12

mov BL, ERR\_CLR

lea BP, FND\_E02

call PR\_STR\_BIOS

jmp dos\_quit

find\_e12: cmp AX, 12h

jne find\_eun

mov BL, ERR\_CLR

lea BP, FND\_E12

call PR\_STR\_BIOS

jmp dos\_quit

find\_eun: lea DI, FND\_EUN

add DI, 22

call WRD\_TO\_HEX

mov BL, ERR\_CLR

lea BP, FND\_EUN

call PR\_STR\_BIOS

jmp dos\_quit

; Получение размера файла

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\_para

mov DL, 01h

jmp get\_para

get\_para: mov CL, 04h

shr AX, CL

add AX, DX

jc size\_err

xchg DX, BP

jmp get\_fmem

size\_err: mov BL, ERR\_CLR

lea BP, ERR\_SIZ

call PR\_STR\_BIOS

jmp dos\_quit

; Выделение блока памяти под оверлей

get\_fmem: mov BX, AX

mov AH, 48h

int 21h

jc error\_48

mov OVLN\_CS, AX

jmp make\_par

; Вывод информации об ошибке

error\_48: lea DI, ERR\_48H

add DI, 50

call WRD\_TO\_HEX

mov BL, ERR\_CLR

lea BP, ERR\_48H

call PR\_STR\_BIOS

jmp dos\_quit

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\_ovl

; Загрузка оверлейного сегмента

load\_ovl: mov CH, 01h

mov AH, 4Bh

mov AL, 03h

int 21h

pop ES

jc load\_e01

mov CH, 00h

jmp exec\_ovl

load\_e01: cmp AX, 01h

jne load\_e02

mov BL, ERR\_CLR

lea BP, EXE\_E01

call PR\_STR\_BIOS

jmp exe\_fmem

load\_e02: cmp AX, 02h

jne load\_e03

mov BL, ERR\_CLR

lea BP, EXE\_E02

call PR\_STR\_BIOS

jmp exe\_fmem

load\_e03: cmp AX, 03h

jne load\_e04

mov BL, ERR\_CLR

lea BP, EXE\_E03

call PR\_STR\_BIOS

jmp exe\_fmem

load\_e04: cmp AX, 04h

jne load\_e05

mov BL, ERR\_CLR

lea BP, EXE\_E04

call PR\_STR\_BIOS

jmp exe\_fmem

load\_e05: cmp AX, 05h

jne load\_e08

mov BL, ERR\_CLR

lea BP, EXE\_E05

call PR\_STR\_BIOS

jmp exe\_fmem

load\_e08: cmp AX, 08h

jne load\_e0A

mov BL, ERR\_CLR

lea BP, EXE\_E08

call PR\_STR\_BIOS

jmp exe\_fmem

load\_e0A: cmp AX, 0Ah

jne load\_e0B

mov BL, ERR\_CLR

lea BP, EXE\_E0A

call PR\_STR\_BIOS

jmp exe\_fmem

load\_e0B: cmp AX, 0Bh

jne load\_eun

mov BL, ERR\_CLR

lea BP, EXE\_E0B

call PR\_STR\_BIOS

jmp exe\_fmem

load\_eun: lea DI, EXE\_EUN

add DI, 24

call WRD\_TO\_HEX

mov BL, ERR\_CLR

lea BP, EXE\_EUN

call PR\_STR\_BIOS

jmp exe\_fmem

; Выполнение кода оверлейного сегмента

exec\_ovl: call dword ptr DS:[OVLN\_IP]

jmp exe\_fmem

; Освобождение выделенного блока памяти

exe\_fmem: mov AX, OVLN\_CS

push ES

mov ES, AX

mov AH, 49h

int 21h

pop ES

jc error\_49

cmp CH, 00h

jne dos\_quit

jmp next\_ovl

; Вывод информации об ошибке

error\_49: lea DI, ERR\_49H

add DI, 50

call WRD\_TO\_HEX

mov BL, ERR\_CLR

lea BP, ERR\_49H

call PR\_STR\_BIOS

jmp dos\_quit

; Выход

dos\_quit: mov AH, 01h

int 21h

mov AH, 4Ch

int 21h

L7\_CODE ENDS

END START

**Код программы LAB7\_OV1.asm**

OVL\_CODE SEGMENT

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

; Код оверлейной процедуры

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

mov AH, 01h

int 21h

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

; Вывод текста прерыванием 21/9

PRINT\_STR PROC NEAR

push AX

mov AH, 09h

int 21h

pop AX

ret

PRINT\_STR ENDP

OVL\_CODE ENDS

END