**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. 8383 |  | Шишкин И.В. |
| Преподаватель |  | Ефремов М.А. |

Санкт-Петербург

2020

**Цель работы.**

Исследование возможности построения загрузочного модуля оверлейной структуры.

**Ход работы.**

Был написан программный модуль типа .ЕХЕ, который выполняет следующие функции:

1. Освобождает память для загрузки оверлея.
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.

Были написаны оверлейные модули OVERLAY1 и OVERLAY2, которые выводят свой сегментный адрес.

Была запущена отлаженная программа. Результат выполнения программы приведен на рис. 1.

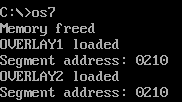


Рисунок 1 – Результат работы программы в 1-м случае

Затем, программа была запущена из другого каталога. Результат выполнения программы приведен на рис. 2.

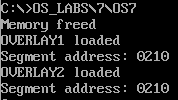


Рисунок 2 – Результат работы программы во 2-м случае

Была запущена отлаженная программа, когда второго оверлея нет в каталоге. Результат выполнения программы приведен на рис. 3.

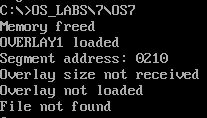


Рисунок 3 – Результат работы программы в 3-м случае

**Контрольные вопросы.**

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

Нужно будет, чтобы обращение к модулю происходило со смещением 100h.

**Выводы.**

В ходе выполнения лабораторной работы была исследована возможность построения загрузочного модуля оверлейной структуры.

ПРИЛОЖЕНИЕ А

КОД ПРОГРАММЫ OS7

AStack SEGMENT STACK

dw 128 dup(0)

Astack ENDS

DATA SEGMENT

PARAMETER\_BLOCK dw 0 ;сегментный адрес среды

dd 0 ;сегмент и смещение командной строки

dd 0 ;сегмент и смещение FCB

dd 0 ;сегмент и смещение FCB

STR\_OVERLAY1\_NAME db 'OVERLAY1.OVL$'

STR\_OVERLAY2\_NAME db 'OVERLAY2.OVL$'

PROGRAM\_PATH db 50 dup(0)

OFFSET\_OVL\_NAME dw 0

DTA\_BUFF db 43 dup(0)

OVL\_PARAM\_SEG dw 0

OVL\_ADDRESS dd 0

IS\_MEMORY\_FREED db 0

STR\_FUNCTION\_COMPLETED db 'Memory freed', 13, 10, '$'

STR\_FUNC\_NOT\_COMPLETED db 13, 10, 'Memory is not freed$'

ERROR\_CODE\_7 db 13, 10, 'Memory control block destroyed$'

ERROR\_CODE\_8 db 13, 10, 'Not enough memory to execute function$'

ERROR\_CODE\_9 db 13, 10, 'Invalid memory block address$'

GET\_SIZE\_ERROR db 13, 10, 'Overlay size not received$'

GET\_SIZE\_ERROR\_CODE\_2 db 13, 10, 'File not found$'

GET\_SIZE\_ERROR\_CODE\_3 db 13, 10, 'Path not found$'

LOADING\_ERROR db 13, 10, 'Overlay not loaded$'

LOADING\_ERROR\_CODE\_1 db 13, 10, 'Non-existent function$'

LOADING\_ERROR\_CODE\_2 db 13, 10, 'File not found$'

LOADING\_ERROR\_CODE\_3 db 13, 10, 'Path not found$'

LOADING\_ERROR\_CODE\_4 db 13, 10, 'Too many open files$'

LOADING\_ERROR\_CODE\_5 db 13, 10, 'No access$'

LOADING\_ERROR\_CODE\_8 db 13, 10, 'No memory$'

LOADING\_ERROR\_CODE\_10 db 13, 10, 'Wrong environment$'

END\_OF\_DATAA db 0

DATA ENDS

CODE SEGMENT

ASSUME CS:CODE, DS:DATA, SS:AStack

;--------------------------------------------------

PRINT PROC near

push AX

mov AH, 09h

int 21h

pop AX

ret

PRINT ENDP

;----------------------------------------------

OVERLAY\_LOADING PROC near

call MAKE\_FULL\_FILE\_NAME

call GET\_OVERLAY\_SIZE

call OVERLAYY

ret

OVERLAY\_LOADING ENDP

;----------------------------------------------

MAKE\_FULL\_FILE\_NAME PROC near

push AX

push DI

push SI

push ES

;mov OFFSET\_OVL\_NAME, AX

;mov BX, AX

mov ES, ES:[2Ch] ;смещение до сегмента окружения (environment)

xor DI, DI

NEXT: ;ищем 2 нуля - т.к. строка запуска программы за ними

mov AL, ES:[DI]

;inc DI

cmp AL, 0

je AFTER\_FIRST\_0

inc DI

jmp NEXT

AFTER\_FIRST\_0:

inc DI

mov AL, ES:[DI]

cmp AL, 0

jne NEXT

add DI, 3h ;нашли 2 нуля, пропускаем 3 цифры

mov SI, 0

WRITE\_NUM:

mov AL, ES:[DI]

cmp AL, 0

je DELETE\_FILE\_NAME

mov PROGRAM\_PATH[SI], AL

inc DI

inc SI

jmp WRITE\_NUM

DELETE\_FILE\_NAME:

dec si

cmp PROGRAM\_PATH[SI], '\'

je READY

jmp DELETE\_FILE\_NAME

READY:

mov DI, -1

ADD\_FILE\_NAME:

inc SI

inc DI

;mov AL, OFFSET\_OVL\_NAME[DI]

mov AL, BX[DI]

cmp AL, '$'

je END\_OF\_COMMAND\_LINE

mov PROGRAM\_PATH[SI], AL

jmp ADD\_FILE\_NAME

END\_OF\_COMMAND\_LINE:

pop ES

pop SI

pop DI

pop AX

ret

MAKE\_FULL\_FILE\_NAME ENDP

;----------------------------------------------

GET\_OVERLAY\_SIZE PROC near

push AX

push BX

push CX

push DX

push SI

mov AH, 1Ah

mov DX, offset DTA\_BUFF

int 21h

; определение размера требуемой памяти

mov AH, 4Eh

mov DX, offset PROGRAM\_PATH

mov CX, 0

int 21h

jnc NO\_ERRORS

mov DX, offset GET\_SIZE\_ERROR

call PRINT

mov DX, offset GET\_SIZE\_ERROR\_CODE\_2

cmp AX, 2

je WRITE\_TYPE\_OF\_ERROR

mov DX, offset GET\_SIZE\_ERROR\_CODE\_3

cmp AX, 3

je WRITE\_TYPE\_OF\_ERROR

jmp END\_OF\_OVL\_SIZE

WRITE\_TYPE\_OF\_ERROR:

call PRINT

jmp END\_OF\_OVL\_SIZE

NO\_ERRORS:

mov SI, offset DTA\_BUFF

add SI, 1Ah

mov BX, [SI]

shr BX, 4

mov AX, [SI + 2]

shl AX, 12

add BX, AX

add BX, 2

mov AH, 48h

int 21h

jnc SAVE\_SEG

mov DX, offset STR\_FUNC\_NOT\_COMPLETED

call PRINT

jmp END\_OF\_OVL\_SIZE

SAVE\_SEG:

mov OVL\_PARAM\_SEG, AX

END\_OF\_OVL\_SIZE:

pop SI

pop DX

pop CX

pop BX

pop AX

ret

GET\_OVERLAY\_SIZE ENDP

;----------------------------------------------

OVERLAYY PROC NEAR

push AX

push BX

push DX

push ES

mov DX, offset PROGRAM\_PATH

push DS

pop ES

mov BX, offset OVL\_PARAM\_SEG

mov AX, 4B03h

int 21h

jnc NO\_LOADING\_ERRORS

mov DX, offset LOADING\_ERROR

call PRINT

mov DX, offset LOADING\_ERROR\_CODE\_1

cmp AX, 1

je WRITE\_LOADING\_OVL\_ERROR

mov DX, offset LOADING\_ERROR\_CODE\_2

cmp AX, 2

je WRITE\_LOADING\_OVL\_ERROR

mov DX, offset LOADING\_ERROR\_CODE\_3

cmp AX, 3

je WRITE\_LOADING\_OVL\_ERROR

mov DX, offset LOADING\_ERROR\_CODE\_4

cmp AX, 4

je WRITE\_LOADING\_OVL\_ERROR

mov DX, offset LOADING\_ERROR\_CODE\_5

cmp AX, 5

je WRITE\_LOADING\_OVL\_ERROR

mov DX, offset LOADING\_ERROR\_CODE\_8

cmp AX, 8

je WRITE\_LOADING\_OVL\_ERROR

mov DX, offset LOADING\_ERROR\_CODE\_10

cmp AX, 10

je WRITE\_LOADING\_OVL\_ERROR

jmp END\_OF\_OVERLAYY

WRITE\_LOADING\_OVL\_ERROR:

call PRINT

jmp END\_OF\_OVERLAYY

NO\_LOADING\_ERRORS:

mov AX, OVL\_PARAM\_SEG

mov ES, AX

mov WORD PTR OVL\_ADDRESS + 2, AX

call OVL\_ADDRESS

mov AH, 49h

int 21h

END\_OF\_OVERLAYY:

pop ES

pop DX

pop BX

pop AX

ret

OVERLAYY ENDP

;----------------------------------------------

FREEING\_UP\_MEMORY PROC near

push AX

push BX

push CX

push DX

mov BX, offset END\_OF\_PROGRAM

mov AX, offset END\_OF\_DATAA

add BX, AX

add BX, 40Fh

mov CL, 4

shr BX, CL

mov AX, 4A00h ;сжать или расширить блок памяти

int 21h

jnc FUNCTION\_COMPLETED

mov DX, offset STR\_FUNC\_NOT\_COMPLETED

call PRINT

mov IS\_MEMORY\_FREED, 0

cmp AX, 7

je IF\_ERROR\_CODE\_7

cmp AX, 8

je IF\_ERROR\_CODE\_8

cmp AX, 9

je IF\_ERROR\_CODE\_9

IF\_ERROR\_CODE\_7:

mov DX, offset ERROR\_CODE\_7

call PRINT

jmp END\_OF\_FREEING

IF\_ERROR\_CODE\_8:

mov DX, offset ERROR\_CODE\_8

call PRINT

jmp END\_OF\_FREEING

IF\_ERROR\_CODE\_9:

mov DX, offset ERROR\_CODE\_9

call PRINT

jmp END\_OF\_FREEING

FUNCTION\_COMPLETED:

mov DX, offset STR\_FUNCTION\_COMPLETED

call PRINT

mov IS\_MEMORY\_FREED, 1

END\_OF\_FREEING:

pop DX

pop CX

pop BX

pop AX

ret

FREEING\_UP\_MEMORY ENDP

;----------------------------------------------

BEGIN PROC FAR

xor AX, AX

push AX

mov AX, DATA

mov DS, AX

mov BX, DS

call FREEING\_UP\_MEMORY

cmp IS\_MEMORY\_FREED, 1

jne ENDD

; 1-я оверлейная программа

mov BX, offset STR\_OVERLAY1\_NAME

call OVERLAY\_LOADING

; 2-я оверлейная программа

mov BX, offset STR\_OVERLAY2\_NAME

call OVERLAY\_LOADING

ENDD:

xor AL, AL

mov AH, 4Ch

int 21h

BEGIN ENDP

END\_OF\_PROGRAM:

CODE ENDS

END BEGIN

ПРИЛОЖЕНИЕ Б

КОД ПРОГРАММЫ OVERLAY1

CODE SEGMENT

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

;----------------------------------------------

BEGIN PROC FAR

push AX

push DX

push DS

push DI

mov AX, CS

mov DS, AX

mov DX, offset IF\_LOAD

call PRINT

mov DI, offset SEGMENT\_ADDRESS

add DI, 22

call WRD\_TO\_HEX

mov DX, offset SEGMENT\_ADDRESS

call PRINT

pop DI

pop DS

pop DX

pop AX

RETF

BEGIN ENDP

;--------------------------------------------------

PRINT PROC near

push AX

mov AH, 09h

int 21h

pop AX

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

;байт в AL переводится в два символа шест. числа в AX

push CX

mov AH, AL

call TETR\_TO\_HEX

xchg AL, AH

mov CL, 4

shr AL, CL

call TETR\_TO\_HEX ;в AL старшая цифра

pop CX ;в AH младшая

ret

BYTE\_TO\_HEX ENDP

;----------------------------------------------

WRD\_TO\_HEX PROC near

;перевод в 16 с/с 16-ти разрядного числа

; в AX - число, DI - адрес последнего символа

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

;--------------------------------------------------

IF\_LOAD db 13, 10, 'OVERLAY1 loaded$'

SEGMENT\_ADDRESS db 13, 10, 'Segment address: $'

;--------------------------------------------------

CODE ENDS

END BEGIN

ПРИЛОЖЕНИЕ В

КОД ПРОГРАММЫ OVERLAY2

CODE SEGMENT

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

;----------------------------------------------

BEGIN PROC FAR

push AX

push DX

push DS

push DI

mov AX, CS

mov DS, AX

mov DX, offset IF\_LOAD

call PRINT

mov DI, offset SEGMENT\_ADDRESS

add DI, 22

call WRD\_TO\_HEX

mov DX, offset SEGMENT\_ADDRESS

call PRINT

pop DI

pop DS

pop DX

pop AX

RETF

BEGIN ENDP

;--------------------------------------------------

PRINT PROC near

push AX

mov AH, 09h

int 21h

pop AX

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

;байт в AL переводится в два символа шест. числа в AX

push CX

mov AH, AL

call TETR\_TO\_HEX

xchg AL, AH

mov CL, 4

shr AL, CL

call TETR\_TO\_HEX ;в AL старшая цифра

pop CX ;в AH младшая

ret

BYTE\_TO\_HEX ENDP

;----------------------------------------------

WRD\_TO\_HEX PROC near

;перевод в 16 с/с 16-ти разрядного числа

; в AX - число, DI - адрес последнего символа

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

;--------------------------------------------------

IF\_LOAD db 13, 10, 'OVERLAY2 loaded$'

SEGMENT\_ADDRESS db 13, 10, 'Segment address: $'

;--------------------------------------------------

CODE ENDS

END BEGIN