

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. 9382

Балаева М.О.

Преподаватель

Ефремов М.А

Санкт-Петербург

2021

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.

Описание функций и структур данных.

Название функции	Назначение
PRINT	Печать строки
MAKEPATH	Определение пути к первому оверлейному сегменту
MAKEPATH_	Определение пути ко второму оверлейному сегменту
READ_OVERLAY	Определяет размер оверлея и запрашивает нужное количество памяти
LOAD_OVERLAY	Запуск оверлея
BYTE_TO_HEX	Переводит число в 2 шестнадцатеричных числа
FREE_MEM	Освобождение лишней памяти
CLEAN_MEM	Очистка памяти, занятой оверлеем
TETR_TO_HEX	вспомогательная функция для работы функции BYTE_TO_HEX

Ход работы.

1. Запуск программы lab7.exe, на рисунке видно, как загружается оверлей, в какой сегмент и как очищаются (рис.1):

```
C:\>lab7.exe
Оверлей успешно загружен
Сегментный адрес Оверлея1:047DH
Память успешно очищена
Оверлей успешно загружен
Сегментный адрес Оверлея2:047DH
Память успешно очищена
```

Рисунок 1

2. Повторный запуск уже из другого каталога (рис.2):

```
C:\NEW>C:\LAB7.EXE
Оверлей успешно загружен
Сегментный адрес Оверлея1:047DH
Память успешно очищена
Оверлей успешно загружен
Сегментный адрес Оверлея2:047DH
Память успешно очищена
```

Рисунок 2

3. Повторный запуск без оверлеев (рис.3):

```
C:\NEW>LAB7.EXE
Ошибка: Файл не найден
Ошибка: Файл не найден
```

Рисунок 3

Заключение.

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

Ответы на контрольные вопросы.

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

Так как при загрузке com модуля-оверлея com-сегмент загружается без смещения в 100h, то требуется вызвать функцию не по нулевому смещению, а по смещению 100h. Компенсировать такое смещение нужно уменьшением сегментного адреса на 10h.

ПРИЛОЖЕНИЕ А

LAB7.ASM

.286

```
ASTACK      SEGMENT  STACK
            DW 100 DUP (0)
ASTACK      ENDS

CODE        SEGMENT
            ASSUME    CS:CODE, DS:DATA, SS:ASTACK, ES:NOTHING

MAKEPATH    MACRO    FILE,PATH
            push      ES
            push      BX
            push      DX
            push      DI
            push      SI
            mov       ES,ES:[2Ch]
            xor       BX,BX
mark1:      mov       DX,ES:[BX]
            cmp       DX,0000h
            jz        read_path
            add       BX,1 ;//
            jmp       mark1
read_path:  add       BX,4
            mov       DI,OFFSET PATH
path_mark1:
            mov       DL,ES:[BX]
            mov       [DI],DL
            inc       DI
            inc       BX
            cmp       DL,00h
            jnz       path_mark1
path_mark2: dec       DI
            mov       DL,[DI]
            cmp       DL,92
            jne       path_mark2
            mov       SI,OFFSET FILE
putname:
            inc       DI
            mov       DL,[SI]
            mov       [DI],DL
            inc       SI
            cmp       DL,00h
            jnz       putname
            pop       SI
            pop       DI
            pop       DX
            pop       BX
            pop       ES
            ENDM

PRINT      MACRO    STRING
            push      AX
            push      DX
            mov       DX,OFFSET STRING
            mov       AH,09h
            int       21h
            pop       DX
            pop       AX
            ENDM
```

```

MAKEPATH_ MACRO      FILE_,PATH
    push    ES
    push    BX
    push    DX
    push    DI
    push    SI
    mov     ES,ES:[2Ch]
    xor     BX,BX
mark1_:
    mov     DX,ES:[BX]
    cmp     DX,0000h
    jz      read_path_
    inc     BX
    jmp     mark1_
read_path_:
    add     BX,4
    mov     DI,OFFSET PATH
path_mark1_:
    mov     DL,ES:[BX]
    mov     [DI],DL
    inc     DI
    inc     BX
    cmp     DL,00H
    jnz     path_mark1_
path_mark2_:
    dec     DI
    mov     DL,[DI]
    cmp     DL,92
    jne     path_mark2_
    mov     SI,OFFSET FILE_
putname_:
    inc     DI
    mov     DL,[SI]
    mov     [DI],DL
    inc     SI
    cmp     DL,00h
    jnz     putname_
    pop     SI
    pop     DI
    pop     DX
    pop     BX
    pop     ES
    ENDM

FREE_MEM  PROC      NEAR
    push    BX
    push    AX
    mov     BX,OFFSET PROGEND
    mov     AH,4Ah
    int     21h
    jnc     FREE_MEM_EXIT
    mov     free_excode,1
    cmp     AX,07h
    jne     tmp1
    PRINT   err6
    jmp     FREE_MEM_EXIT
tmp1:
    cmp     AX,08h
    jne     tmp2
    PRINT   err7
    jmp     FREE_MEM_EXIT

```

```

tmp2:      cmp      AX,09h
           jne      UNERRFREE
           PRINT    err8
           jmp      FREE_MEM_EXIT
UNERRFREE: PRINT    err10
FREE_MEM_EXIT:
           pop      AX
           pop      BX
           ret
FREE_MEM   ENDP

READ_OVERLAY PROC      NEAR
           push     BP
           push     AX
           push     BX
           push     DX
           push     CX
           mov      AH,1Ah
           mov      DX,OFFSET buffer
           int      21h
           mov      AH,4Eh
           mov      DX,OFFSET path
           mov      CX,0
           int      21h
           jnc      BYTE_TO_PAR
           mov      read_excode,1
           cmp      AX,12h
           jne      tmp3
           PRINT    err2
           jmp      READEXIT
tmp3:      cmp      AX,02h
           jne      UNERRREAD
           PRINT    err3
           jmp      READEXIT
UNERRREAD: PRINT    err10
           jmp      READEXIT
BYTE_TO_PAR:
           mov      BP,OFFSET buffer
           mov      BX,DS:[BP+1AH]
           mov      AX,DS:[BP+1CH]
           shr      BX,4
           shl      AX,12
           add      BX,AX
           inc      BX
           mov      AX,DS:[BP+1CH]
           and      AX,0FFF0H
           cmp      AX,0000H
           jz       REQ_MEM
           mov      read_excode,1
           PRINT    err7
           jmp      READEXIT
REQ_MEM:
           mov      AH,48h
           int      21h
           jnc      READSAVE
           mov      read_excode,1
           PRINT    err7
           jmp      READEXIT
READSAVE:  mov      overlay_seg,AX
           mov      overlay_seg1,AX
           mov      reloc,AX

```

```

READEXIT: pop      CX
           pop      DX
           pop      BX
           pop      AX
           pop      BP
           ret
READ_OVERLAY  ENDP

LOAD_OVL  PROC      NEAR
           push     AX
           push     BX
           push     DX
           push     ES
           mov      DX,OFFSET path
           push     DS
           pop      ES
           mov      BX,OFFSET char_block
           mov      AX,4B03h
           int      21h
           jc       load_err
           PRINT    ov_load
           call     DWORD PTR ov_address
           jmp      LOADEXIT
load_err:  mov      load_excode,1
           cmp      AX,01h
           jne      error2
           PRINT    err1
           jmp      LOADEXIT
error2:   cmp      AX,02h
           jne      error3
           PRINT    err2
           jmp      LOADEXIT
error3:   cmp      AX,03h
           jne      error4
           PRINT    err3
           jmp      LOADEXIT
error4:   cmp      AX,04h
           jne      error5
           PRINT    err4
           jmp      LOADEXIT
error5:   cmp      AX,05h
           jne      error7
           PRINT    err5
           jmp      LOADEXIT
error7:   cmp      AX,08h
           jne      error9
           PRINT    err7
           jmp      LOADEXIT
error9:   cmp      AX,0Ah
           jne      UNERRLOAD
           PRINT    err9
           jmp      LOADEXIT
UNERRLOAD:PRINT    err10
LOADEXIT: pop      ES
           pop      DX
           pop      BX
           pop      AX
           ret
LOAD_OVL  ENDP

CLEAN_MEM  PROC      NEAR

```

```

        push    AX
        push    ES
        mov     AX,overlay_seg
        mov     ES,AX
        mov     AH,49h
        int     21h
        jc      CLEANERR
        PRINT   mem_clean
        jmp     CLEANEXIT
CLEANERR: mov     clean_excode,1
        PRINT   err11
CLEANEXIT: pop     ES
        pop     AX
        ret
CLEAN_MEM    ENDP

MAIN        PROC    NEAR
        mov     AX,DATA
        mov     DS,AX
        call    FREE_MEM
        cmp     free_excode,0
        jne     NEXT
        MAKEPATH file_ovl1,path
        mov     CX,1
        call    READ_OVERLAY
        cmp     read_excode,0
        jne     NEXT
        call    LOAD_OVL
        call    CLEAN_MEM
        cmp     load_excode,0
        jne     NEXT
        cmp     clean_excode,0
        jne     NEXT

NEXT:       MAKEPATH_ file_ovl2,path
        mov     read_excode,0
        mov     load_excode,0
        mov     clean_excode,0
        mov     CX,1
        call    READ_OVERLAY
        cmp     read_excode,0
        jne     EXIT
        call    LOAD_OVL
        call    CLEAN_MEM
        cmp     load_excode,0
        jne     EXIT
        cmp     clean_excode,0
        jne     EXIT
EXIT:       xor     AL,AL
        mov     AH,4Ch
        int     21h
        ret
MAIN        ENDP
PROGEND:
CODE        ENDS

DATA        SEGMENT
err1         DB    'ERROR: Invalid function number',0DH,0AH,'$'
err2         DB    'Ошибка: Файл не найден',0DH,0AH,'$'
err3         DB    'Ошибка: Путь не найден',0DH,0AH,'$'

```



```

err4      DB      'Ошибка: Слишком много открытых
файлов',0DH,0AH,'$'
err5      DB      'Ошибка: Доступ закрыт',0DH,0AH,'$'
err6      DB      'Ошибка: Блоки управления памятью
разрушены',0DH,0AH,'$'
err7      DB      'Ошибка: Недостаточно памяти',0DH,0AH,'$'
err8      DB      'Ошибка: Неверный адрес блока
памяти',0DH,0AH,'$'
err9      DB      'Ошибка: Недопустимая среда',0DH,0AH,'$'
err10     DB      'Ошибка: Неизвестно',0DH,0AH,'$'
err11     DB      'Ошибка: очистка не завершена',0DH,0AH,'$'
ov_load   DB      'Оверлей успешно загружен',0DH,0AH,'$'
mem_clean DB      'Память успешно очищена',0DH,0AH,'$'
file_ovl1 DB      'lab7_1.ovl',0
file_ovl2 DB      'lab7_2.ovl',0
path      DB      128 DUP (0)
buffer    DB      43 DUP (0)
char_block EQU    $
overlay_seg DW     ?
reloc      DW     ?
ov_address EQU    $
ov_ofst    DW     0
overlay_seg1 DW     ?
free_excode DB     0
read_excode DB     0
load_excode DB     0
clean_excode DB     0
DATA      ENDS
END       MAIN

```

ПРИЛОЖЕНИЕ Б

OVL1.OVL

```

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 ADDRESS + 30
call      WORD_TO_HEX
call      PRINT
pop       DS
pop       DI
pop       AX
RETF
OVERLAY   ENDP

PRINT     PROC      NEAR
push      AX
push      DX
mov       DX,OFFSET ADDRESS
mov       AH,09H
int       21H
pop       DX
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
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

WORD_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
WORD_TO_HEX ENDP

ADDRESS DB      'Сегментный адрес Оверлея1:  H',0DH,0AH,'$'
CODE     ENDS
        END     OVERLAY

```

ПРИЛОЖЕНИЕ В

OVL2.OVL

```

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 ADDRESS + 30
          call      WORD_TO_HEX
          call      PRINT
          pop       DS
          pop       DI
          pop       AX
          RETF
OVERLAY   ENDP

PRINT     PROC      NEAR
          push      AX
          push      DX
          mov       DX,OFFSET ADDRESS
          mov       AH,09H
          int       21H
          pop       DX
          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
          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

WORD_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
WORD_TO_HEX ENDP

ADDRESS DB      'Сегментный адрес Оверлея2:  H',0DH,0AH,'$'
CODE    ENDS
        END     OVERLAY

```