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. 7383 _____ Прокопенко Н.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург

2019

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.

Постановка задачи:

Шаг 1. Для выполнения лабораторной работы необходимо написать и отладить 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, отведённая для оверлейного сегмента

5) Затем действия 1) – 4) выполняются для следующего оверлейного сегмента

Шаг 2. Также необходимо написать и отладить оверлейные сегменты. Оверлейный сегмент выводит адрес сегмента, в который он загружен

Шаг 3. Запустите отлаженное приложение. Оверлейные сегменты должны загружаться с одного и того же адреса, перекрывая друг друга.

Шаг 4. Запустите приложение из другого каталога. Приложение должно быть выполнено успешно.

Шаг 5. Запустите приложение в случае, когда одного оверлея нет в каталоге. Приложение должно закончиться аварийно.

Шаг 6. Занесите полученные результаты в виде скриншотов в отчет. Оформите отчет в соответствии с требованиями.

Описание функций и структур данных.

Описание процедур и структур данных представлены в табл. 1 и табл.2 соответственно.

Таблица 1 – описание процедур

Название процедуры	Назначение
TETR_TO_HEX	Перевод 2-х байт в 16-ую систему счисления
BYTE_TO_HEX	Перевод байта в 10-ую систему счисления
WRD_TO_HEX	Перевод слова (2 байта) в шестнадцатеричную систему счисления
PR_STR_BIOS	Вывод текста

Таблица 2 – описание структур данных

Название поля данных	Тип	Назначение
OVL_END_SUCCESS	db	Проверка оверлеев завершена успешно
OVL_END_ERROR	db	Во время проверки оверлеев возникла ошибка
ERR_FUNC_48H	db	Ошибка функции 48H прерывания 21H, код ошибки:
ERR_FUNC_49H	db	Ошибка функции 49H прерывания 21H, код ошибки:
ERR_FUNC_4AH	db	Ошибка функции 4AH прерывания 21H, код ошибки:
ERR_SIZE_ABOVE	db	Ошибка: Размер оверлея превышает 1048560 (FFFF0H) байт!

Окончание таблицы 2

ERR_FIND_02H	db	Ошибка поиска, код 0002H: Указанный путь не существует
ERR_FIND_12H	db	Ошибка поиска, код 0012H: Указанный файл не найден.
ERR_FIND_UNKNOWN	db	Ошибка поиска, код:
ERR_EXE_01H	db	Ошибка загрузки, код 0001H: Неверный номер подфункции.
ERR_EXE_02H	db	Ошибка загрузки, код 0002H: Указанный файл не найден.
ERR_EXE_03H	db	Ошибка загрузки, код 0003H: Указанный путь не существует.
ERR_EXE_04H	db	Ошибка загрузки, код 0004H: Открыто слишком много файлов.
ERR_EXE_05H	db	Ошибка загрузки, код 0005H: Ошибка доступа к файлу.
ERR_EXE_08H	db	Ошибка загрузки, код 0008H: Не хватает свободной памяти.
ERR_EXE_0AH	db	Ошибка загрузки, код 000AH: Блок среды превышает 32 Кб.
ERR_EXE_0BH	db	Ошибка загрузки, код 000BH: Некорректный формат файла.
ERR_EXE_UNKNOWN	db	Ошибка загрузки, код:

Примеры работы программы по шагам представлены на рисунке 1, рисунке 2, рисунке 3, рисунке 4.

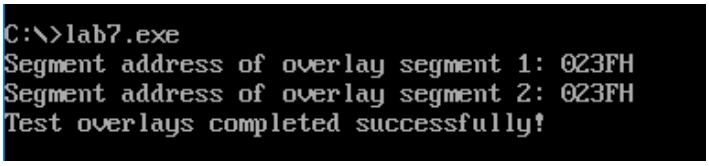

Рисунок 1 – Запуск программы из каталога с разработанными оверлеями.

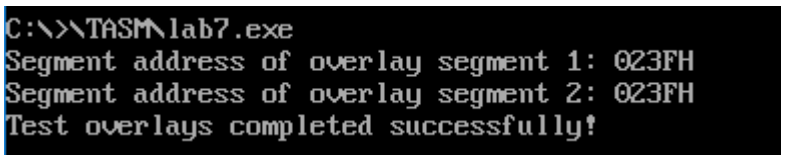

Рисунок 2 – Запуск программы во время нахождения в другом каталоге.

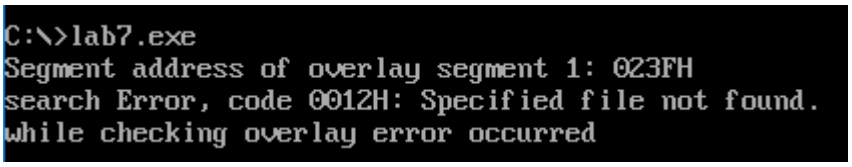

Рисунок 3 – Запуск программы при отсутствии одного из оверлеев. Программа завершилась аварийно.

Выводы.

В ходе лабораторной работы был построен загрузочный модуль оверлейной структуры, а также оверлеи. Изучены дополнительные функции работы с памятью и способы загрузки и выполнения оверлейных сегментов.

Код программы lab7.ASM представлен в приложении А.

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

```
.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  'Test  overlays  completed  successfully!',
0Dh, 0Ah, '$'
        OVL_END_ERROR  db  'while  checking  overlay  error  occurred',
0Dh, 0Ah, '$'
        ERR_FUNC_48H   db  'Interrupt  function  48H  ERROR  21H,  error
code: H.', 0Dh, 0Ah, '$'
        ERR_FUNC_49H   db  'Interrupt  function  error  49H  21H,  error
code: H.', 0Dh, 0Ah, '$'
        ERR_FUNC_4AH   db  'interrupt  function  4AH  Error  21H,  error
code: H.', 0Dh, 0Ah, '$'
        ERR_SIZE_ABOVE      db  'Error:  overlay  Size  exceeds  1048560
(FFFF0H) bytes!', 0Dh, 0Ah, '$'
        ERR_FIND_02H   db  'search  Error,  code  0002H:  the  Specified
path does not exist.', 0Dh, 0Ah, '$'
        ERR_FIND_12H   db  'search  Error,  code  0012H:  Specified  file
not found.', 0Dh, 0Ah, '$'
        ERR_FIND_UNKNOWN db 'search Error code H', 0Dh, 0Ah, '$'
        ERR_EXE_01H db 'boot Error code 0001H: Wrong number of sub-
functions.', 0Dh, 0Ah, '$'
        ERR_EXE_02H db 'download Failed, code 0002H: the Specified
file was not found.', 0Dh, 0Ah, '$'
        ERR_EXE_03H db 'Error loading, code 0003H: Specified path
does not exist.', 0Dh, 0Ah, '$'
        ERR_EXE_04H db 'download Error, code 0004H: Too many files
opened.', 0Dh, 0Ah, '$'
        ERR_EXE_05H  db  'download  Error,  code  0005H:  file  access
Error.', 0Dh, 0Ah, '$'
        ERR_EXE_08H db 'download Error, code 0008H: Not enough free
memory.', 0Dh, 0Ah, '$'
        ERR_EXE_0AH db 'boot Error code 000AH: the environment Block
is larger than 32 KB.', 0Dh, 0Ah, '$'
```

```
            ERR_EXE_0BH db 'Error loading code 000BH: Incorrect file
format.', 0Dh, 0Ah, '$'
            ERR_EXE_UNKNOWN db 'load Error, code H', 0Dh, 0Ah, '$'
            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
```

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, 41
                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 'Segment address of overlay segment 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
```

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, 41
                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 'Segment address of overlay segment 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
```