МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра математического обеспечения и применения ЭВМ

ОТЧЕТ

по практической работе №7

по дисциплине «Операционные системы»

Тема: Построение модуля оверлейной структуры

Студентка гр. 7382 _____ Чемова К.А.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург

2019

Цель лабораторной работы.

Исследование возможности построения 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 загрузочные и оверлейные модули находятся в одном каталоге.

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, отведенная 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 отлаженную программу. 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, когда одного оверлея нет в каталоге. Приложение должно закончиться аварийно.

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

Процедуры используемые в программе.

- PRINT – выводит сообщение на экран;

- DTA_SET – установка адреса области обмена с диском (DTA блока);
- TETR_TO_HEX – вспомогательная функция, которая переводит из двоичной в шестнадцатеричную систему счисления, используется для работы функции BYTE_TO_HEX;
- BYTE_TO_HEX – переводит байтовое число из регистра AL в шестнадцатеричную систему счисления;
- WRD_TO_HEX – переводит число из регистра AX в шестнадцатеричную систему счисления;
- FREE_MEMORY – освобождает место в памяти, используя функцию 4Ah прерывания int 21h;
- FIND_PATH – извлечение полного имени файла оверлея из среды;
- ALLOCATE_MEMORY_FOR_OVL – выделение памяти под оверлей;
- PROGRAM_CALL_OVL – вызов программы оверлея;
- PROCESSING – обработка оверлея: нахождение полного пути, выделение памяти под файл и вызов программы.

Структуры данных.

Таблица 2 – Структуры данных используемые в программе

Название	Тип	Назначение
Mem_7	db	Строка – сообщение об ошибке освобождения памяти: 7 – разрушен управляющий блок памяти; 8 – недостаточно памяти для выполнения функции; 9 – неверный адрес блока памяти
Mem_8	db	
Mem_9	db	
Load_1	db	Строка-сообщение об ошибке загрузки файла оверлея: 1 – несуществующая функция; 2 – файл не найден; 3 – маршрут не найден; 4 – слишком много открытых файлов; 5 – нет доступа; 8 – мало памяти; 10 – неправильная среда
Load_2	db	
Load_3	db	
Load_4	db	
Load_5	db	
Load_8	db	
Load_10	db	
Err_alloc	db	Строка-сообщение об ошибке выделения памяти для загрузки оверлея
Path	db	Строка-сообщение для вывода вычисленного пути до оверлея

File_2	db	Строка-сообщение об ошибке поиска запускаемого файла оверлея: 2 – файл не найден; 3 – маршрут не найден
File_3	db	
OvlPath	db	Строка, содержащая полный путь до запускаемого оверлея
DTA	db	Переменная, содержащая сведения об области обмена с диском (DTA блок)
Keep_psp	dw	Переменная для сохранение содержимого регистра PSP
SegAdr	dw	Переменная, для сохранения сегментного адреса освобождённого для оверлея блока памяти

Ход работы.

Результат запуска программы с двумя оверлеями в каталогепоказан на рисунке 1.

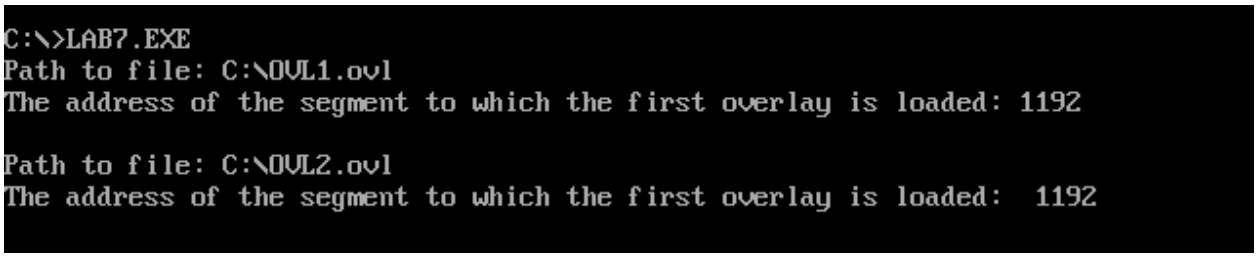

Рисунок 1 – Результат шага 1

Результаты запуска программы, когда в каталоге только один оверлей, представлены на рис. 2 и 3.

Рисунок 2 – Нет первого оверлея

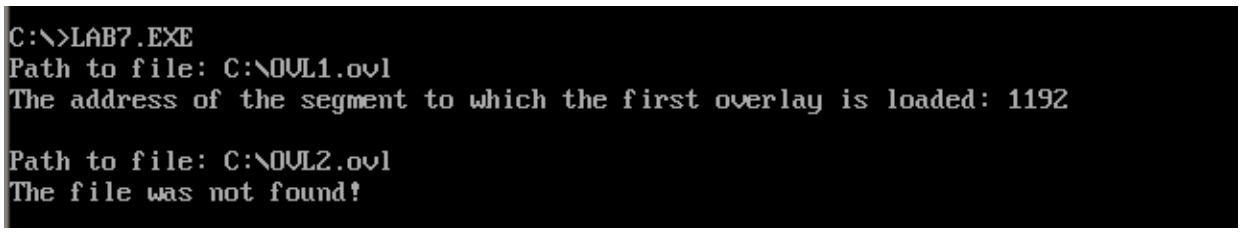

Рисунок 2 – Нет второго оверлея

В случае, когда ни одного из оверлеев нет в каталоге, программа завершает свою работу аварийно.

Вывод.

В результате выполнения данной лабораторной работы были исследованы организация загрузочных модулей оверлейной структуры. Была написана программа, в которой ошибок не обнаружено.

Ответы на контрольные вопросы.

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

В COM-модуле после записи значений регистров в стек, необходимо поместить значение регистра CS в регистр DS, так как адрес сегмента данных совпадает с адресом сегмента кода, кроме того необходимо добавить 100h, т. к. изначально данные сегменты настроены на PSP.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ LAB7.ASM

```
ASTACK SEGMENT STACK
     DW 64 DUP (?)
ASTACK ENDS
;ДАННЫЕ
;-------------------------------------------------------------------------
-----------------------
DATA SEGMENT
     Load_1     db   0DH, 0AH,'The overlay was not been loaded: non-
existent function!', 0DH, 0AH, '$'
     Load_10     db    0DH, 0AH,'The overlay was not been loaded:
incorrect environment!', 0DH, 0AH, '$'
     Load_4    db  0DH, 0AH,'The overlay was not been loaded: too many
open files!', 0DH, 0AH, '$'
     Load_3     db  0DH, 0AH,'The overlay was not been loaded: route
not found!', 0DH, 0AH, '$'
     Load_2    db  0DH, 0AH,'The overlay was not been loaded: file not
found!', 0DH, 0AH, '$'
     Load_8      db   0DH, 0AH,'The overlay was not been loaded: low
memory!', 0DH, 0AH, '$'
     Load_5      db   0DH, 0AH,'The overlay was not been loaded: no
access!', 0DH, 0AH, '$'
     Mem_8         db    0DH, 0AH,'Not enough memory to perform the
function!', 0DH, 0AH, '$'
     Err_alloc  db    0DH,  0AH,'Failed to allocate memory to load
overlay!', 0DH, 0AH, '$'
     Mem_9     db  0DH, 0AH,'Wrong address of the memory block!', 0DH,
0AH, '$'
     Mem_7       db   0DH, 0AH,'Memory control unit destroyed!', 0DH,
0AH, '$'
     File_3    db  0DH, 0AH,'The route was not found!', 0DH, 0AH, '$'
     File_2    db  0DH, 0AH,'The file was not found!', 0DH, 0AH, '$'
     Path      db  'Path to file: ', '$'
     Ovl1   db  'OVL1.ovl', 0000h
     Ovl2   db  'OVL2.ovl', 0000h
     OvlPath   db  64 DUP (?), '$'
     DTA       db  43 DUP (?)
     Keep_psp  dw  0000h
     SegAdr    dw  0000h
     CallAdr      dd  0000h
DATA ENDS
```

```
;-------------------------------------------------------------------------
-----------------------
CODE SEGMENT
        ASSUME CS:CODE, DS:DATA, ES:DATA, SS:ASTACK
;ПРОЦЕДУРЫ
;-------------------------------------------------------------------------
-----------------------
PRINT      PROC NEAR
;вывод строки
      mov   AH,  0009h
      int   21h
      ret
PRINT  ENDP
;-------------------------------------------------------------------------
-----------------------
DTA_SET PROC NEAR
;установка адреса DTA блока
        push    DX
        lea     DX,   DTA   ;вычисление эффективного адреса, DS:DX на
строку, содержащую имя файла с оверлеем
        mov     AH,   1Ah   ;установить адрес области обмена с диском
(DTA)
        int     21h       ;вызов функции
        pop     DX
DTA_SET ENDP
;-------------------------------------------------------------------------
-----------------------
FREE_MEMORY PROC NEAR                                   ;освобождение
места в памяти
                                                    ;перед      вызовом
функции надо определить объём памяти, необходимый lr6
      mov   BX,  offset LAST_BYTE                    ;кладём в BX адрес
конца программы
      mov   AX,   ES                                    ;ES - начало
программы
      sub   BX,   AX                                  ;BX = BX - ES,
число параграфов, которые будут выделяться прграмме
      mov   CL,  0004h
      shr   BX,   CL                                    ;переводим в
параграфы
      mov   AH,  4Ah                                  ;функция сжатия
блока памяти
      int   21h
      jnc   WITHOUT_ERROR                             ;флаг CF = 0, если
нет ошибки
```

```
      jmp   WITH_ERROR                                    ;обработка ошибок
CF=1 AX = код ошибки, если CF установлен
;*************************************************************************
**********************
WITHOUT_ERROR:
      ret
;*************************************************************************
**********************
WITH_ERROR:
;*************************************************************************
**********************
MEM_7_ERROR:
      cmp   AX, 0007h                                ;разрушен управляющий
блок памяти
      jne   MEM_8_ERROR
      mov   DX, offset Mem_7
      jmp   END_ERROR
;*************************************************************************
**********************
MEM_8_ERROR:
      cmp   AX, 0008h                               ;недостаточно памяти для
выполнения функции
      jne   MEM_9_ERROR
      mov   DX, offset Mem_8
      jmp   END_ERROR
;*************************************************************************
**********************
MEM_9_ERROR:
        mov     DX, offset Mem_9                          ;неверный адрес
блока памяти
;*************************************************************************
**********************
END_ERROR:                                          ;вывод ошибки на экран
      call  PRINT
      xor   AL, AL
      mov   AH, 4Ch
      int   21h
FREE_MEMORY ENDP
;-------------------------------------------------------------------------
----------------------
FIND_PATH PROC NEAR                                       ;поиск пути к
файлу оверлея
      push  ES
      mov   ES,  ES:[2CH]                               ;сегментный адрес
среды, передаваемой программе
```

```
      xor   SI,  SI
      lea   DI,  OvlPath
;*************************************************************************
**********************
FIRST:
      inc   SI                                          ;взятие следующего
символа строки
         cmp     word ptr ES:[SI], 0000h                ;проверка на то,
что это конец строки
      jne   FIRST                                     ;переход, если не конец
строки
      add   SI, 0004h                                   ;взятие следующего
символа строки
;*************************************************************************
**********************
SECOND:
      cmp   byte ptr ES:[SI], 0000h                  ;проверка на то, что это
конец строки
      je    THIRD                                     ;если конец строки (два
нулевых байта подряд)
      mov   DL, ES:[SI]
      mov   [DI], DL
      inc   SI
      inc   DI
      jmp   SECOND
;*************************************************************************
**********************
THIRD:
      dec   SI                                          ;взятие предыдущего
символа строки
      dec   DI
      cmp   byte ptr ES:[SI], '\'
      jne   THIRD                                     ;переход, если символ
текущий не "\"
      inc   DI                                          ;взятие следующего
символа строки
      mov   SI, BX
      push  DS
      pop   ES
;*************************************************************************
**********************
FOURTH:
      lodsb                                           ;чтение байта из строки
по адресу DS:SI в AL
```

```
      stosb                                          ;запись байта в строку,
сохранить AL по адресу ES:DI
      cmp   AL, 0000h                               ;проверка на то, что это
конец строки
      jne   FOURTH                                   ;переход, если не конец
строки
      mov   byte ptr [DI], '$'
      mov   DX, offset Path
      call  PRINT
      lea   DX, OvlPath
      call  PRINT
      pop   ES
      ret
FIND_PATH ENDP
;-------------------------------------------------------------------------
-----------------------
ALLOCATE_MEMORY_FOR_OVL  PROC  NEAR                          ;считывание
размера оверлея и выделение памяти под этот файл
      push  DS
      push  DX
      push  CX
      xor   CX, CX                                        ;CX - значение
байта атрибутов для сравнения (для файла 0)
      lea   DX, OvlPath                                     ;вычисление
эффективного адреса, DS:DX = адрес ASCIIZ-строки с именем файла
                                                    ;ASCIIZ-строка    -
способ представления строки, при котором используется массив
символов,а конец - нуль-символ
      mov   AH, 4Eh                                         ;функция 4Eh -
поиск первого совпадающего символа
      int   21h                                          ;вызов функции
      jnc   FILE_IS_FOUND                                ;переход, если флаг
CF=0 (нет ошибок, найдено совпадение)
      cmp   AX, 0003h                                     ;в AX хранится код
ошибки, если CF установлен
      je    ERROR_3                                          ;переход, если
ошибка с кодом 0003h
      mov   DX, offset File_2                             ;если файл не был
найден
      jmp   EXIT_ERROR
;*************************************************************************
**********************
ERROR_3:
      mov   DX, offset File_3                          ;если маршрут не был
найден
```

```
;*************************************************************************
**********************
EXIT_ERROR:                                     ;вывод ошибки на экран
     call  PRINT
     pop   CX
     pop   DX
     pop   DS
     xor   AL, AL
     mov   AH, 4Ch
     int   21h
;*************************************************************************
**********************
FILE_IS_FOUND:                                  ;если файл был найден
     push  ES
     push  BX
     mov   BX, offset DTA                        ;смещение на DTA
     mov   DX, [BX + 1Ch]                         ;старшее слово размера
памяти в байтах
     mov   AX, [BX + 1Ah]                         ;младшее слово размера
файла
     mov   CL, 0004h                               ;перевод в параграфы
младшего слова
     shr   AX, CL                                  ;сдвиг бит операнда
вправо
     mov   CL, 000Ch
     sal   DX, CL                                  ;переводим в байты и
параграфы, сдвиг влево
     add   AX, DX
     inc   AX                                    ;взятие большего целого
числа параграфов
     mov   BX, AX                                    ;BX - запрошенное
количество памяти в 16-байтовых параграфах
     mov   AH, 48h                                 ;распределить память
(дать размер памяти)
     int   21h
     jnc   MEMORY_ALLOCATED                        ;переход, если CF=0
(память выделена)
     mov   DX, offset Err_alloc                     ;вывод сообщеня об
ошибке
     call  PRINT
     xor   AL, AL
     mov   AH, 4Ch
     int   21h
;*************************************************************************
**********************
```

```
MEMORY_ALLOCATED:
      mov   SegAdr, AX                              ;SegAdr - сегментный
адрес распределенного блока
      pop   BX
      pop   ES
      pop   CX
      pop   DX
      pop   DS
      ret
ALLOCATE_MEMORY_FOR_OVL ENDP
;-------------------------------------------------------------------
----------------------
PROGRAM_CALL_OVL PROC NEAR                          ;вызов программы
оверлея
      push  DX
      push  BX
      push  AX
      mov   BX, seg SegAdr
      mov   ES, BX
      lea   BX, SegAdr                             ;ES:BX = адрес EPB
(EXEC Parameter Block - блока параметров EXEC)
      lea   DX, OvlPath                               ;DS:DX = адрес
строки ASCIIZ с именем файла, содержащего программу
      mov   AX, 4B03h                               ;функция, которая
загружает программный оверлей
      int   21h
      jnc   IS_LOADED                              ;переход, если нет
ошибок
;*******************************************************************
**********************
ERROR_CHECK:
      cmp   AX, 0001h                          ;несуществующий файл
      lea   DX, Load_1
      je    PRINT_ERROR
      cmp   AX, 0002h                          ;файл не найден
      lea   DX, Load_2
      je    PRINT_ERROR
      cmp   AX, 0003h                          ;маршрут не найден
      lea   DX, Load_3
      je    PRINT_ERROR
      cmp   AX, 0004h                           ;слишком много открытых
файлов
      lea   DX, Load_4
      je    PRINT_ERROR
      cmp   AX, 0005h                          ;нет доступа
```

```
      lea   DX, Load_5
      je    PRINT_ERROR
      cmp   AX, 0008h                           ;мало памяти
      lea   DX, Load_8
      je    PRINT_ERROR
      cmp   AX, 000Ah                           ;неправильная среда
      lea   DX, Load_10
;***************************************************************************
**********************
PRINT_ERROR:                                    ;вывод сообщения об
ошибки на экран
      call  PRINT
      jmp   FINISH
;***************************************************************************
**********************
IS_LOADED:
      mov   AX, DATA                            ;восстанавливаем DS
      mov   DS, AX
      mov   AX, SegAdr
      mov   word ptr CallAdr + 0002h, AX
      call  CallAdr                             ;вызываем оверлейную
программу
      mov   AX, SegAdr
      mov   ES, AX
      mov   AX,  4900h                                  ;освободить
распределенный блок памяти
      int   21h
      mov   AX, DATA
      mov   DS, AX
;***************************************************************************
**********************
FINISH:
      mov   ES, Keep_psp
      pop   AX
      pop   BX
      pop   DX
      ret
PROGRAM_CALL_OVL ENDP
;---------------------------------------------------------------------------
----------------------
PROCESSING PROC NEAR
;обработка оверлея
        call    FIND_PATH
        call    ALLOCATE_MEMORY_FOR_OVL
        call    PROGRAM_CALL_OVL
```

```
     ret
PROCESSING ENDP
;-------------------------------------------------------------------------
-----------------------
MAIN:
        mov     AX, DATA
     mov  DS, AX
     mov  Keep_psp, ES
     call FREE_MEMORY
     call DTA_SET
     lea  BX, Ovl1
     call PROCESSING
     lea  BX, Ovl2
     call PROCESSING
     xor  AL, AL
     mov  AH, 4Ch
     int  21h
LAST_BYTE:
CODE ENDS
        END     MAIN
```

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ OVL1.ASM

```
LR7_OVL1 SEGMENT
ASSUME CS:LR7_OVL1, DS:LR7_OVL1, ES:NOTHING, SS:NOTHING
;-------------------------------------------------------------------------
-----------------------
BEGINNING PROC FAR
      push  DS
      push  DX
      push  DI
      push  AX
      mov   AX, CS
      mov   DS, AX
      lea   BX, StrForPrint
      add   BX, 46h
      mov   DI, BX
      mov   AX, CS
      call  WRD_TO_HEX
      lea   DX, StrForPrint
      call  PRINT
      pop   AX
      pop   DI
      pop   DX
      pop   DS
      retf
BEGINNING ENDP
;-------------------------------------------------------------------------
-----------------------
PRINT PROC NEAR
      mov  AH, 0009h
      int  21h
      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

WRD_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
WRD_TO_HEX ENDP
;-------------------------------------------------------------------------
-----------------------
StrForPrint db 0DH, 0AH, 'The address of the segment to which the
first overlay is loaded:                ', 0DH, 0AH, '$'
;-------------------------------------------------------------------------
-----------------------
LR7_OVL1 ENDS
END
```

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ OVL2.ASM

```
LR7_OVL1 SEGMENT
ASSUME CS:LR7_OVL1, DS:LR7_OVL1, ES:NOTHING, SS:NOTHING
;-------------------------------------------------------------------------
-----------------------
BEGINNING PROC FAR
      push  DS
      push  DX
      push  DI
      push  AX
      mov   AX, CS
      mov   DS, AX
      lea   BX, StrForPrint
      add   BX, 47h
      mov   DI, BX
      mov   AX, CS
      call  WRD_TO_HEX
      lea   DX, StrForPrint
      call  PRINT
      pop   AX
      pop   DI
      pop   DX
      pop   DS
      retf
BEGINNING ENDP
;-------------------------------------------------------------------------
-----------------------
PRINT PROC NEAR
      mov  AH, 0009h
      int  21h
      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

WRD_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
WRD_TO_HEX ENDP
;-------------------------------------------------------------------------
-----------------------
StrForPrint db 0DH, 0AH, 'The address of the segment to which the
first overlay is loaded:                ', 0DH, 0AH, '$'
;-------------------------------------------------------------------------
-----------------------
LR7_OVL1 ENDS
END
```