МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МОЭВМ

ОТЧЕТ

по лабораторной работе №7

по дисциплине «Операционные системы»

Тема: «Построение модуля оверлейной структуры»

Студент гр. 7381 _____ Вологдин М.Д.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург

2019

Цель 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.

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:

print	Макрос вызова прерывания, печатающего строку
printl	Макрос, печатающий строку и переводящий каретку на новую строку консоли
TETR_TO_HEX	Вспомогательная функция для работы функции BYTE_TO_HEX
BYTE_TO_HEX	Переводит число AL в коды символов 16-ой с/с, записывая получившееся в al и ah
FREE_MEM	Процедура освобождения лишней памяти
ALLOC_MEM	Процедура выделения памяти для загрузки оверлея
FIND_OVERLAY_PATH	Процедура, проверяющая, есть ли хвост, и если есть, то устанавливающая его в качестве имени загружаемого оверлея, иначе используется имя “overlay.exe”. Устанавливает DS:DX на строку с именем оверлея, если таковой нашелся
RUN_OVERLAY	Процедура, которая вызывает процедуры FIND_OVERLAY_PATH, ALLOC_MEM; загружает оверлей в память, запускает его, и по окончании работы оверлея освобождает память, выделенную для него. На вход принимает указатель на строку с именем загружаемого оверлея в регистре BP.
PROCESS_LOADER_ERRORS	Процедура, обрабатывающая ошибки OS

Описание структур данных:

STR_ERR_*	Строки оповощения об ошибках
STR_FREE_MEM	Строка, оповещающая о работе функции FREE_MEM
STR_ALLOC_MEM	Строка, оповещающая о работе функции ALLOC_MEM
STR_FIND_OVERLAY_PATH	Строка, оповещающая о работе функции FIND_OVERLAY_PATH
STR_RUN_OVERLAY	Строка, оповещающая о работе функции RUN_OVERLAY
STR_PROC_DONE	Строка, оповещающая, что действие успешно завершено
STR_RMV_OVERLAY	Строка, оповещающая о начале освобождения памяти от оверлея
STRENDL	Массив символов, переводящих каретку на начало новой строки
OVERLAY_ADDR	Первое слово – для хранения IP оверлея, второе – для сегмента
PARAMBLOCK	Блок параметров, содержащий сегментный адрес, по которому загружается оверлей
OVERLAY_PATH	Строка, используемая для хранения имени загружаемого оверлея, если есть хвост командной строки

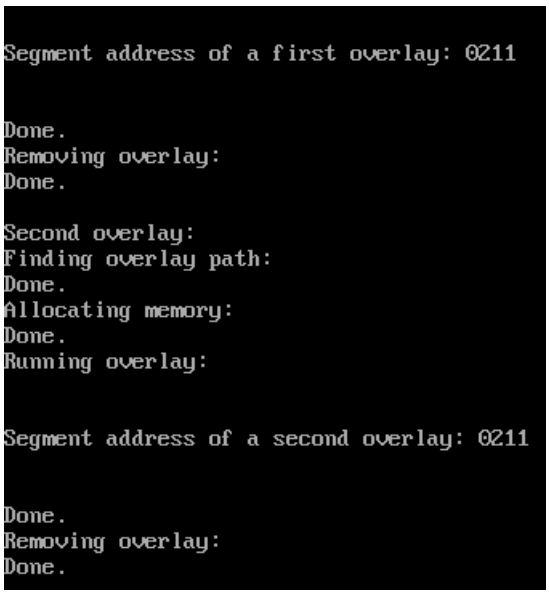
Выполнение работы.

Был написан программный модуль .EXE, который выполняет следующие функции:

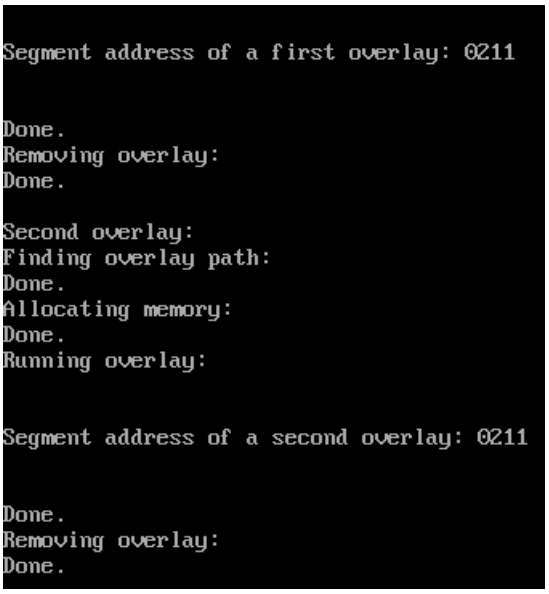
1. Освобождает память для загрузки оверлеев.
2. Читает размер файла оверлея и запрашивает объем памяти, достаточный для его загрузки.
3. Файл оверлейного сегмента загружается и выполняется.
4. Освобождается память, отведенная для оверлейного сегмента.
5. Предыдущие действия выполняются для следующего оверлейного сегмента.

Тестирование

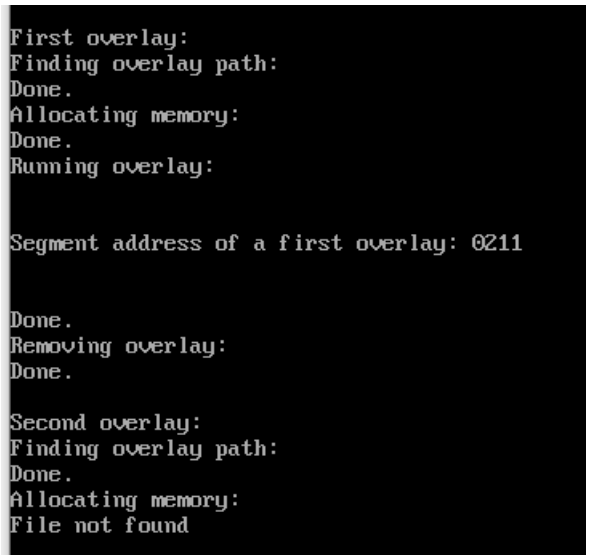
1) Запуск отлаженной программы, Оверлейные сегменты загружаются с одного адреса, перекрывая друг друга.



2) Запуск отлаженной программы из другого каталога.



3) Запуск отлаженной программы, когда одного оверлея нет в каталоге



Выводы

В ходе данной лабораторной работы была исследованы возможность построения загрузочного модуля оверлейной структуры, структура оверлейного сегмента и способ загрузки и выполнения оверлейных сегментов.

Ответы на контрольные вопросы

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

Ответ: При использовании .COM модуля необходимо:

- перед загрузкой оверлейного модуля сгенерировать сегмент PSP в начале выделенной памяти
- выделить память для стека, корректно установить регистры SS и SP; количество выделенной памяти нужно установить, как у стандартного загрузчика (64 кб)
- при переходе в оверлей адрес требуется сместить на 100h для того, чтобы блок PSP не выполнялся как код

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД

```
PRINT MACRO
      PUSH AX
      MOV AX,0900H
      INT 21H
      POP AX
ENDM
PRINTL MACRO A
      PUSH AX
      PUSH DX
      LEA DX,A
      MOV AX,0900H
      INT 21H
      LEA DX,STRENDL
      INT 21H
      POP DX
      POP AX
ENDM
DEBUG MACRO R
      PUSHF
      PUSH AX
      PUSH DI
      MOV AX,R
      LEA DI,STRTEST+3
      CALL WRD_TO_HEX
      PRINTL STRTEST
      POP DI
      POP AX
      POPF
ENDM
CODE SEGMENT
 ASSUME CS:CODE, DS:DATA, ES:DATA, SS:STACKSEG
START: JMP BEGIN
```

```
;---------------------------------------
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 FAR
     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
;---------------------------------------
FREE_MEM PROC
      PUSH AX
      PUSH BX
      PUSH DX

            PRINTL STR_FREE_MEM
            MOV AX,STACKSEG
            MOV BX,ES
            SUB AX,BX
            ADD AX,10H
            MOV BX,AX
            MOV AH,4AH
            INT 21H
            JNC FREE_MEM_SUCCESS

            MOV DX,OFFSET STR_ERR_FREE_MEM
            PRINT
            CMP AX,7
            MOV DX,OFFSET STR_ERR_MCB_DESTROYED
            JE FREE_MEM_PRINT_ERROR
            CMP AX,8
            MOV DX,OFFSET STR_ERR_NOT_ENOUGH_MEM
            JE FREE_MEM_PRINT_ERROR
            CMP AX,9
            MOV DX,OFFSET STR_ERR_WRNG_MEM_BL_ADDR

            FREE_MEM_PRINT_ERROR:
            PRINT
            MOV DX,OFFSET STRENDL
```

```
            PRINT

            XOR AL,AL
            MOV AH,4CH
            INT 21H

      FREE_MEM_SUCCESS:
      PRINTL STR_PROC_DONE
      POP DX
      POP BX
      POP AX
      RET
FREE_MEM ENDP
;---------------------------------------
ALLOC_MEM PROC
      PUSH DS
      PUSH DX
            PRINTL STR_ALLOC_MEM

            XOR CX,CX
            MOV AX,4E00H
            INT 21H
            JNC ALLOC_MEM_FILE_FOUND
                  PRINTL STR_ERR_FL_NOT_FND
                  POP DX
                  POP DS
                  JMP ALLOC_MEM_ERR_EXIT
            ALLOC_MEM_FILE_FOUND:
            PUSH ES
            PUSH BX

            MOV AH,2FH
            INT 21H
```

```
            ADD BX,1AH
            MOV AX,WORD PTR ES:[BX]
            MOV DX,WORD PTR ES:[BX+2]
            MOV BX,10H
            DIV BX
            INC AX

            MOV AH,48H
            INT 21H
            JNC ALLOC_MEM_DONE
                  PRINTL STR_ERR_ALLOC_MEM
                  CALL PROCESS_LOADER_ERRORS
                  MOV AX,4C00H
                  INT 21H
            ALLOC_MEM_DONE:
            POP ES
            POP BX
      POP DX
      POP DS
      PRINTL STR_PROC_DONE
      MOV WORD PTR OVERLAY_ADDR+2,AX
      MOV PARAMBLOCK,AX
      RET

      ALLOC_MEM_ERR_EXIT:

      RET
ALLOC_MEM ENDP

FIND_OVERLAY_PATH PROC
            PRINTL STR_FIND_OVERLAY_PATH

      PUSH ES
      PUSH AX
```

```
        PUSH SI
        PUSH DI
        PUSH CX

        MOV AX, ES:[2CH]
        MOV ES, AX

        MOV AL, 0
        MOV DI, 0
        FIND_OVERLAY_SKIP_BEGIN:
        MOV CX, 512
REPNE SCASB
        CMP ES:[DI],AL
        JNE FIND_OVERLAY_SKIP_BEGIN

        ADD DI, 3
        PUSH DI
        MOV CX,100
REPNE SCASB
        MOV AX,DI
        POP DI
        SUB AX,DI
        DEC AX
        MOV CX,AX


        PUSH DS
        PUSH ES
        POP DS
        POP ES
        MOV SI,DI
        LEA DI,OVERLAY_PATH
REP MOVSB
        MOV BYTE PTR ES:[DI],0
```

```
      MOV CX,100
      MOV AL,'\' ; ^É¥¬ Á¨¬¢®« '\'
      STD
REPNE SCASB
      CLD
      ADD DI,2
      MOV CX,13
      MOV AX,DATA
      MOV DS,AX
      MOV SI,BP

REP MOVSB

      POP CX
      POP DI
      POP SI
      POP AX
      POP ES

      LEA DX,OVERLAY_PATH

      PRINTL STR_PROC_DONE
      RET
FIND_OVERLAY_PATH ENDP
;---------------------------------------
RUN_OVERLAY PROC
      PUSH DS
      PUSH ES
      PUSH AX
      PUSH BX
      PUSH CX
      PUSH DX
```

```
CALL FIND_OVERLAY_PATH

CALL ALLOC_MEM
JNC OVERLAY_FOUND
     JMP OVERLAY_ERR
OVERLAY_FOUND:
MOV CS:KEEP_SP, SP
MOV CS:KEEP_SS, SS
MOV CS:KEEP_DS, DS
MOV CS:KEEP_ES, ES

MOV PARAMBLOCK,AX
PUSH DS
POP ES
LEA BX, PARAMBLOCK

PRINTL STR_RUN_OVERLAY
     MOV AX,4B03H
     INT 21H
     JNC OVL_LOADED
          CALL PROCESS_LOADER_ERRORS
     OVL_LOADED:
     PRINTL STRENDL
     CALL DWORD PTR OVERLAY_ADDR
     PRINTL STRENDL
MOV SP,CS:KEEP_SP
MOV SS,CS:KEEP_SS
MOV DS,CS:KEEP_DS
MOV ES,CS:KEEP_ES
JNC OVERLAY_SUCCESS

     CALL PROCESS_LOADER_ERRORS
```

```
        OVERLAY_SUCCESS:
        PRINTL STR_PROC_DONE
        PRINTL STR_RMV_OVERLAY
        PUSH ES
        MOV AX,PARAMBLOCK
        MOV ES,AX
        MOV AX,4900H
        INT 21H
        POP ES
        JNC OVERLAY_DELETE_SUCCESS
        CALL PROCESS_LOADER_ERRORS
        OVERLAY_DELETE_SUCCESS:
        PRINTL STR_PROC_DONE
        OVERLAY_ERR:
        POP DX
        POP CX
        POP BX
        POP AX
        POP ES
        POP DS
        RET

        KEEP_SP DW 0
        KEEP_SS DW 0
        KEEP_DS DW 0
        KEEP_ES DW 0
RUN_OVERLAY ENDP
;--------------------------------------
PROCESS_LOADER_ERRORS PROC
        CMP AX,1
        LEA DX,STR_ERR_WRNG_FNCT_NUMB
        JE PROCESS_LOADER_ERRORS_PRINT
        CMP AX,2
        LEA DX,STR_ERR_FL_NOT_FND
```

```
JE PROCESS_LOADER_ERRORS_PRINT
CMP AX,3
LEA DX,STR_ERR_PATH_NOT_FOUND
JE PROCESS_LOADER_ERRORS_PRINT
CMP AX,4
LEA DX,STR_ERR_TOO_MANY_FILES
JE PROCESS_LOADER_ERRORS_PRINT
CMP AX,5
LEA DX,STR_ERR_NO_ACCESS
JE PROCESS_LOADER_ERRORS_PRINT
CMP AX,6
LEA DX,STR_ERR_INVLD_HNDL
JE PROCESS_LOADER_ERRORS_PRINT
CMP AX,7
LEA DX,STR_ERR_MCB_DESTROYED
JE PROCESS_LOADER_ERRORS_PRINT
CMP AX,8
LEA DX,STR_ERR_NOT_ENOUGH_MEM
JE PROCESS_LOADER_ERRORS_PRINT
CMP AX,9
LEA DX,STR_ERR_WRNG_MEM_BL_ADDR
JE PROCESS_LOADER_ERRORS_PRINT
CMP AX,10
LEA DX,STR_ERR_WRONG_ENV
JE PROCESS_LOADER_ERRORS_PRINT
CMP AX,11
LEA DX,STR_ERR_WRONG_FORMAT
JE PROCESS_LOADER_ERRORS_PRINT
CMP AX,12
LEA DX,STR_ERR_INV_ACCSS_MODE
JE PROCESS_LOADER_ERRORS_PRINT
LEA DX,STR_ERR_UNKNWN
PROCESS_LOADER_ERRORS_PRINT:
PRINT
```

```
      MOV AX,4C00H
      INT 21H
PROCESS_LOADER_ERRORS ENDP
;---------------------------------------
BEGIN:
      MOV AX,DATA
      MOV DS,AX
      CALL FREE_MEM

      PUSH DX
      LEA DX,STRENDL
      PRINT
      POP DX
      PRINTL STR_FIRST_OVERLAY

      LEA BP, STR_OVL1
      CALL RUN_OVERLAY

      PUSH DX
      LEA DX,STRENDL
      PRINT
      POP DX
      PRINTL STR_SECOND_OVERLAY

      LEA BP, STR_OVL2
      CALL RUN_OVERLAY

      XOR AL,AL
      MOV AH,4CH
      INT 21H
CODE ENDS
; „€□□›…
DATA SEGMENT
```

```
                STR_ERR_FREE_MEM              DB  'ERROR  WHEN  FREEING
MEMORY: $'
                STR_ERR_ALLOC_MEM             DB      'ERROR      WHILE
ALLOCATING MEMORY$'
                STR_ERR_TOO_LONG_TAIL         DB 'TAIL IS TOO LONG$'

                STR_ERR_WRNG_FNCT_NUMB        DB  'FUNCTION  NUMBER  IS
WRONG$'
                STR_ERR_FL_NOT_FND            DB 'FILE NOT FOUND$'
                STR_ERR_PATH_NOT_FOUND        DB 'PATH NOT FOUND$'
                STR_ERR_TOO_MANY_FILES        DB    'TOO    MANY    OPEN
FILES$'
                STR_ERR_NO_ACCESS             DB 'ACCESS DENIED$'
                STR_ERR_INVLD_HNDL            DB 'INVALID HANDLE$'
                STR_ERR_MCB_DESTROYED         DB    'MEMORY    CONTROL
BLOCKS DESTROYED$'
                STR_ERR_NOT_ENOUGH_MEM        DB         'INSUFFICIENT
MEMORY$'
                STR_ERR_WRNG_MEM_BL_ADDR  DB   'INVALID   MEMORY   BLOCK
ADDRESS$'
                STR_ERR_WRONG_ENV             DB              'INVALID
ENVIRONMENT$'
                STR_ERR_WRONG_FORMAT      DB 'INVALID FORMAT$'
                STR_ERR_INV_ACCSS_MODE        DB    'INVALID    ACCESS
MODE$'
                STR_ERR_UNKNWN                DB 'UNKNOWN ERROR$'



                STR_FIRST_OVERLAY         DB 'FIRST OVERLAY:$'
                STR_SECOND_OVERLAY        DB 'SECOND OVERLAY:$'
                STR_FREE_MEM              DB 'FREEING MEMORY:$'
                STR_ALLOC_MEM             DB 'ALLOCATING MEMORY:$'
                STR_FIND_OVERLAY_PATH     DB 'FINDING OVERLAY PATH:$'
                STR_RUN_OVERLAY           DB 'RUNNING OVERLAY:$'
```

```
        STR_PROC_DONE           DB 'DONE.$'
        STR_RMV_OVERLAY         DB 'REMOVING OVERLAY:$'

    STRENDL                 DB 0DH,0AH,'$'
    STRTEST                     DB '    $'

    OVERLAY_ADDR DD 0

    PARAMBLOCK DW 0
               DD 0
    OVERLAY_PATH        DB 105 DUP ('$')
    STR_OVL1            DB 'OVERLAY1.OVL',0
    STR_OVL2            DB 'OVERLAY2.OVL',0

DATA ENDS
; ‘’…Š
STACKSEG SEGMENT STACK
    DW 80H DUP (?) ; 100H ¡ ©Â
STACKSEG ENDS
 END START
```