МИНОБРНАУКИ РОССИИ

САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ

ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)

Кафедра МО ЭВМ

ОТЧЁТ

по лабораторной работе №7

по дисциплине «Операционные системы»

Тема: Построение модуля оверлейной стуктуры

Студент гр.8382 _____ Фильцин И.В.

Преподаватель _____ Ефремов М.А..

Санкт-Петербург

2020

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 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.

Ход работы

В ходе лабораторной программы был написан программный модуль типа .EXE, который освобождает память для загрузки оверлеев, читает размер файла оверлея и запрашивает объем памяти, достаточный для его загрузки. Затем файл оверлейного сегмента загружается и выполняется. Затем освобождается память, отведенная для оверлейного сегмента.

Результат запуска программы см. на рис. 1

Результат запуска программы из другой дирректории см. на рис. 2

Запуск программы при отсутствии 1-ого оверлея см. на рис. 3

Запуск программы при отсутствии 2-ого оверлея см. на рис. 4

Контрольные вопросы

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

Отличие от обычного оверлейного сегмента состоит в том, что код в .COM модуле начинается со смещением 100h. Следовательно, для того, чтобы использовать .COM модуль необходимо учитывать это смещение.

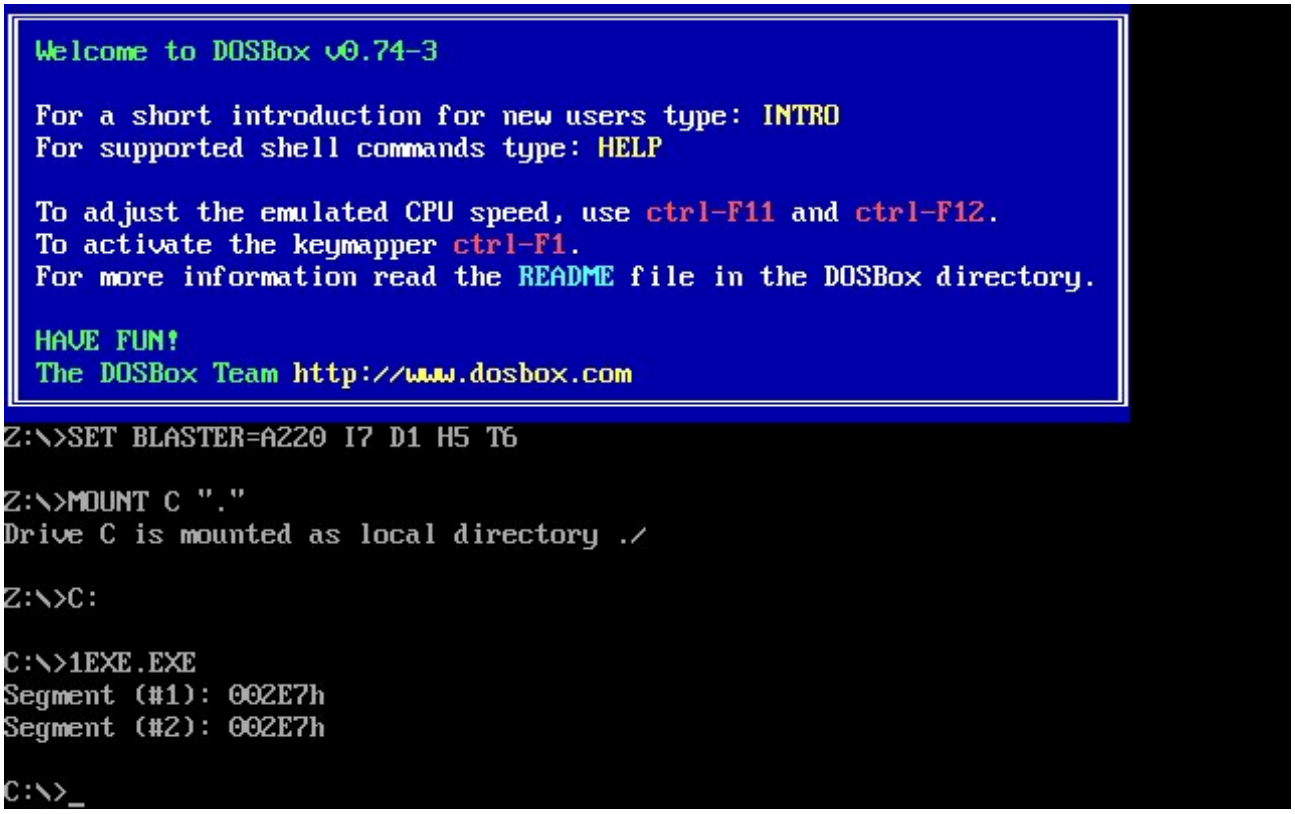

Рис. 1: Запуск программы

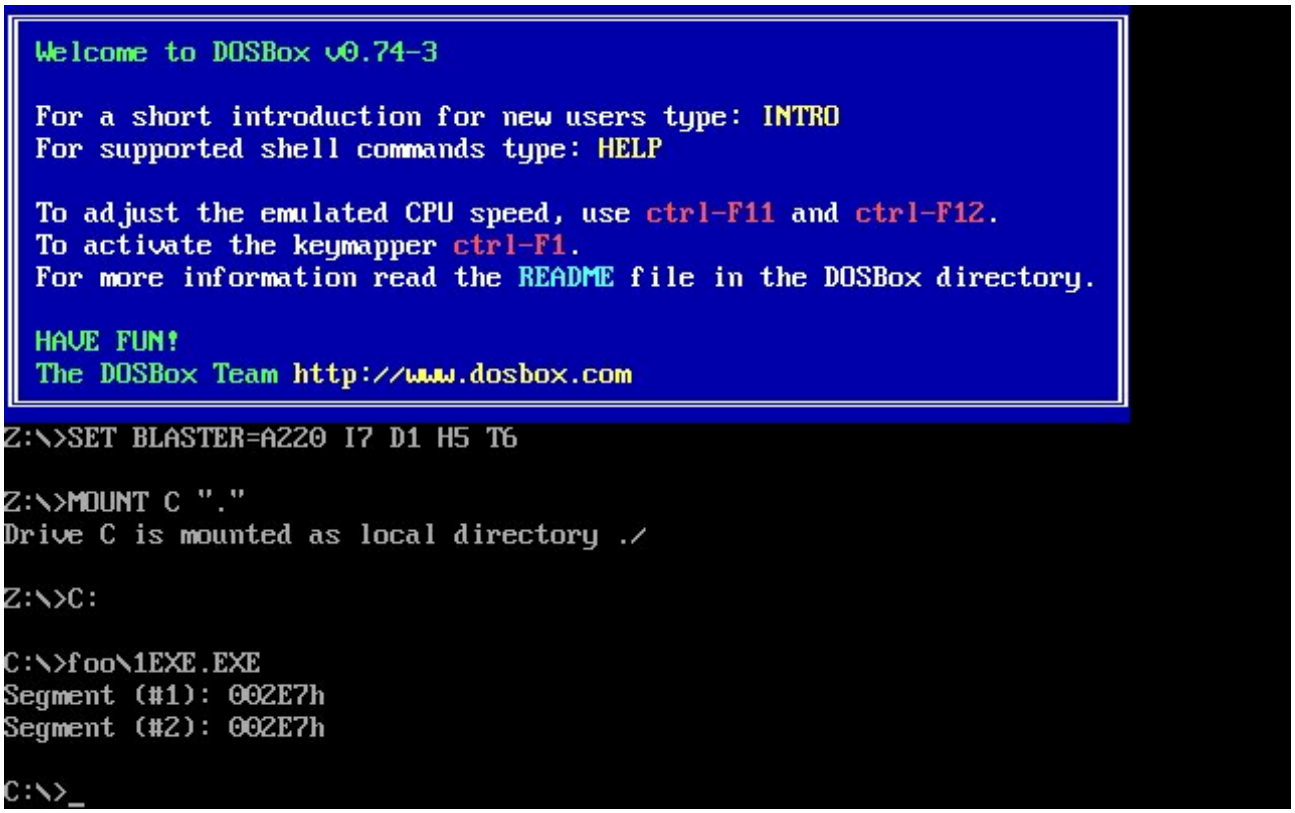

Рис. 2: Результат запуска программы из другой дирректории

Рис. 3: Запуск программы при отсутствии 1-ого оверлея

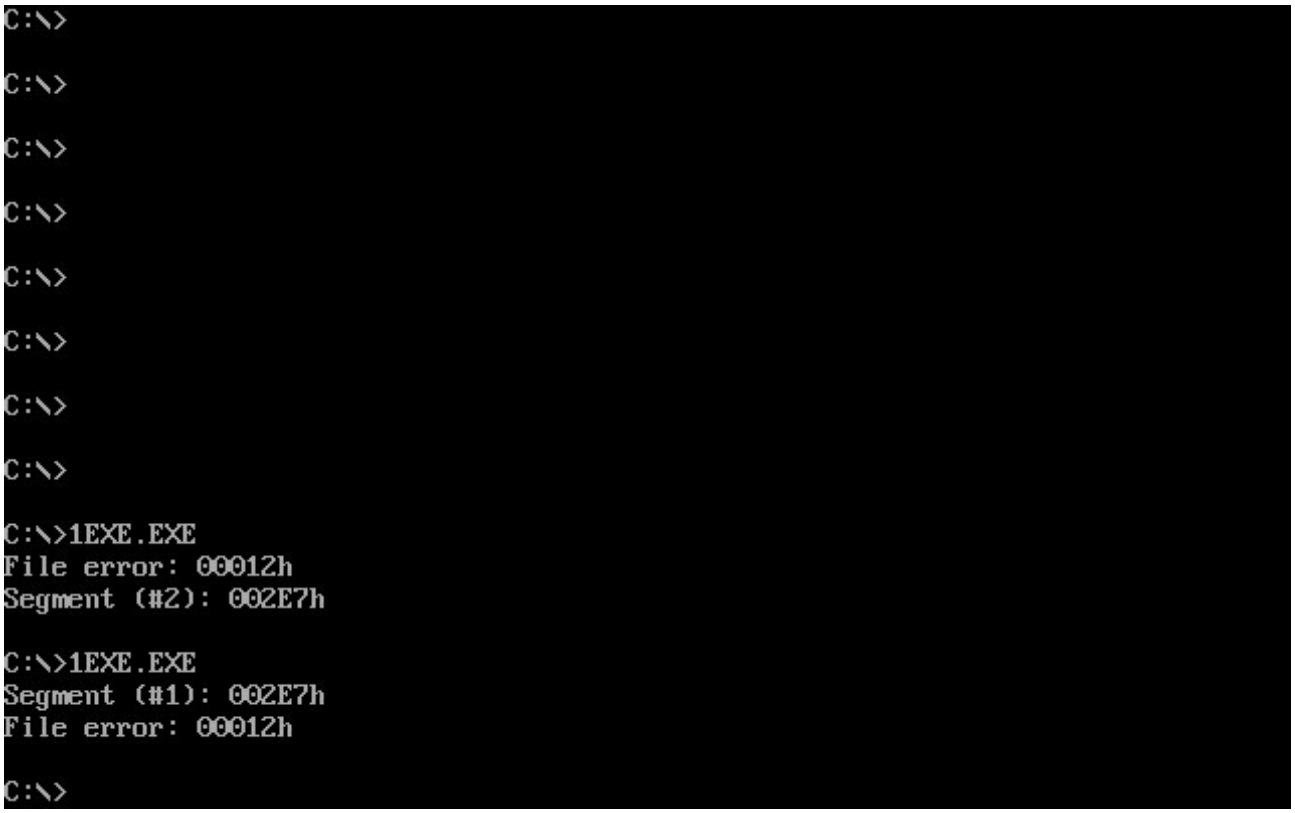

Рис. 4: Запуск программы при отсутствии 2-ого оверлея

Вывод

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

Приложение А. Исходный код программы

```
.model small

.stack 100h

.data
  error_dealloc db 'Dealloc error: 00000h', 13, 10, '$'
  error_file db 'File error: 00000h', 13, 10, '$'
  error_alloc db 'Alloc error: 00000h', 13, 10, '$'
  error_run db 'Run error: 00000h', 13, 10, '$'
  program_path db 100 dup(?)
  dta db 43 dup(0)
  ovl_seg dw 0
  ovl_addr dd 0

.code

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

call_ovl proc near
  push es
  mov si, 02ch
  mov es, es:[si]

  mov si, 0

  out_print:
    mov dl, es:[si]
    cmp dl, 0
    je finish_1
  while_print:
    mov dl, es:[si]
    inc si
    inc bp
    cmp dl, 0
    je out_print
    jmp while_print
  finish_1:

  add si, 3
  mov bp, offset program_path

  print_for:
    mov dl, es:[si]
```

```
  mov ds:[bp], dl
  cmp dl, 0
  je finish_print
  inc si
  inc bp
  jmp print_for
finish_print:

sub bp, 8
mov ds:[bp], byte ptr 'o'
mov ds:[bp + 1], byte ptr 'v'
mov ds:[bp + 2], ax
mov ds:[bp + 3], byte ptr '.'
mov ds:[bp + 4], byte ptr 'o'
mov ds:[bp + 5], byte ptr 'v'
mov ds:[bp + 6], byte ptr 'l'
mov ds:[bp + 7], byte ptr 0

mov dx, offset program_path
xor cx, cx
mov ah, 04eh
int 21h
jnc good_file

mov di, offset error_file
add di, 16
call wrd_to_hex
```

```
mov dx, offset error_file
mov ah, 09h
int 21h
jmp finish

good_file:

mov bx, offset dta
mov ax, [bx + 01ch]
mov bx, [bx + 01ah]

mov cl, 4
shr bx, cl
mov cl, 12
shl ax, cl

add bx, ax
inc bx

mov ah, 048h
int 21h
jnc good_alloc

mov di, offset error_alloc
add di, 17
call wrd_to_hex
```

```
mov dx, offset error_alloc
mov ah, 09h
int 21h
jmp finish

good_alloc:
mov ovl_seg, ax
mov ax, @data
mov es, ax
mov bx, offset ovl_seg
mov dx, offset program_path
mov ax, 04b03h
int 21h
jnc good_run

mov di, offset error_run
add di, 15
call wrd_to_hex

mov dx, offset error_run
mov ah, 09h
int 21h
jmp finish

good_run:
mov ax, ovl_seg
```

```
  mov word ptr ovl_addr + 2, ax
  push ds
  call ovl_addr
  pop ds
  mov ax, ovl_seg
  mov es, ax
  mov ah, 049h
  int 21h

  finish:
    pop es
    ret
call_ovl endp

main:
  mov ax, @data
  mov ds, ax

  mov dx, offset dta
  mov ah, 01ah
  int 21h


  mov bx, offset last_byte

  mov ah, 04ah
  int 21h
```

```
  jnc good_dealloc

  mov di, offset error_dealloc
  add di, 17
  call wrd_to_hex

  mov dx, offset error_dealloc
  mov ah, 09h
  int 21h
  jmp finish_prog

  good_dealloc:
  mov ax, '1'
  call call_ovl
  mov ax, '2'
  call call_ovl

  finish_prog:
  mov ah, 04ch
  int 21h

last_byte:
end main
```

Приложение А. Исходный код оверлея

```
code segment
assume cs:code, es:nothing, ds:nothing, ss:nothing

main proc far
  push ax
  push dx
  push ds
  push di

  mov ax, cs
  mov ds, ax

  mov di, offset seg_label
  add di, 18
  call wrd_to_hex

  mov dx, offset seg_label
  mov ah, 09h
  int 21h

  pop di
  pop ds
  pop dx
  pop ax
  retf
main endp
```

```
seg_label db 'Segment (#1): 00000h', 13, 10, '$'

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

code ends
end main
```