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

ОТЧЕТ

по лабораторной работе №6
по дисциплине «Операционные системы»
Тема: Построение модуля динамической структуры

Студент гр. 9383 Мосин К.К.

Преподаватель Ефремов М.А.

Санкт-Петербург
2021

Цель работы.

Исследование возможности построения загрузочного модуля динамической структуры.

Задание.

Шаг 1. Написать и отладить модуль типа .EXE, который выполняет следующие функции:

1) Подготавливает параметры для запуска загрузочного модуля из того же каталога, в котором находится он сам.

2) Вызываемый модуль запускается с использованием загрузчика.

3) После запуска проверяется выполнение загрузчика, а затем результат выполнения вызываемой программы.

Шаг 2. Запуск отлаженной программы, когда текущим каталогом является каталог с разработанными модулями. Ввод символа A-Z.

Шаг 3. Запуск отлаженной программы, когда текущим каталогом является каталог с разработанными модулями. Ввод комбинации Ctrl-C.

Шаг 4. Запуск отлаженной программы, когда текущим каталогом является любой другой каталог. Повторный ввод комбинаций.

Шаг 5. Запуск отлаженной программы, когда модули находятся в разных каталогах.

Выполнение работы.

Запуск разработанных модулей из исходного каталога, а также из другого каталога и когда программы находятся в разных каталогах представлены на изображениях 1, 2 и 3 соответственно.

Рис. 1 - запуск модулей из этого же каталога

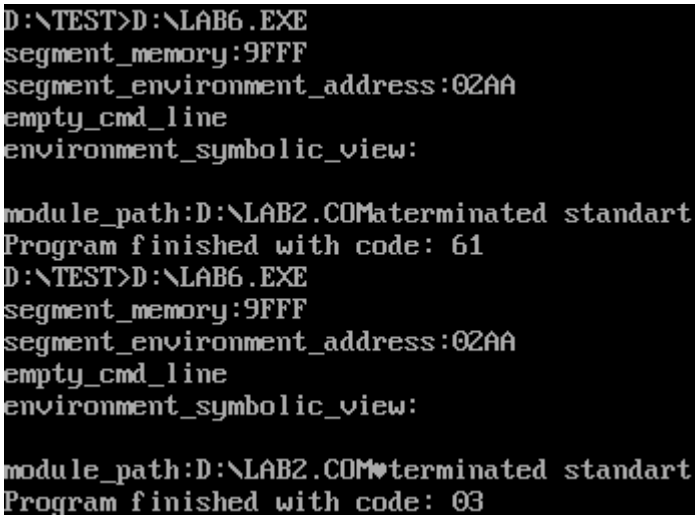

Рис. 2 - запуск модулей из другого каталога

Рис. 3 - запуск программы, когда модули находятся в разных каталогах

Контрольные вопросы.

1) Как реализовано прерывание Ctrl-C?

По нажатию Ctrl-C вызывается прерывание int 23h, после управление передается коду по адресу 008C, который ведет к остановке текущей программы.

2) В какой точке заканчивается вызываемая программа, если код причины завершения 0?

Программа заканчивается в точке вызова прерывания 21h с функцией выхода в DOS.

3) В какой точке заканчивается вызываемая программа по прерыванию Ctrl-C?

В точке ожидания ввода символа с клавиатуры прерывания 21h в модуле lab2.asm.

Выводы.

В ходе выполнения лабораторной работы был построен загрузочный модуль динамической системы.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: lab6.asm

```
stack_ segment stack
dw 128 dup(?)
stack_ ends

data_ segment
execute_param      dw 0
db 0
db 0
db 0
file_path db 64 dup(0)
file_name db 'lab2.com',0
end_line db ' ',0dh,0ah,'$'

exit_code db 'Program finished with code: $'
error_1 db 'wrong number of function'
error_2 db 'file path not found',0dh,0ah,'$'
error_5 db 'wrong disk',0dh,0ah,'$'
error_8 db 'not enough memory',0dh,0ah,'$'
error_10 db 'wrong environment string',0dh,0ah,'$'
error_11 db 'wrong format',0dh,0ah,'$'
error_free_memory db 'fail free memory',0dh,0ah,'$'
error_unknown db 'unknown',0dh,0ah,'$'

termination_0 db 'terminated standart',0dh,0ah,'$'
termination_1 db 'terminated using ctrl-c',0dh,0ah,'$'
termination_2 db 'terminated by device error',0dh,0ah,'$'
termination_3 db 'terminated by 31h',0dh,0ah,'$'
```

```
keep_ss dw 0
keep_sp dw 0

data_ ends

code_ segment
assume cs:code_, ds:data_, ss:stack_

print proc near
push ax
mov ah,09h
int 21h
pop ax
ret
print endp

print_end_line proc near
push dx
mov dx,offset end_line
call print
pop dx
ret
print_end_line 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

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

main proc far
mov ax,data_
mov ds,ax
;free memory
mov ax,es
mov bx,offset main_end_prt
sub bx,ax
add bx,100h
mov ah,4ah
int 21h
jc free_memory_error
jmp find_path
free_memory_error:
mov dx,offset error_free_memory
call print
```

```
find_path:
mov es,es:[2ch]
mov bx,0
xor si,si
loop_:
cmp byte ptr es:[bx],00h
jne next_byte
cmp byte ptr es:[bx+1],00h
jne next_byte
jmp path_found
next_byte:
inc bx
jmp loop_

path_found:
add bx,4
lea di,file_path

read:
mov dl,es:[bx+si]
cmp byte ptr es:[bx+si],0
je insert_file_name
mov [di],dl
inc di
inc si
jmp read

insert_file_name:
sub di,8
mov cx,8
lea si,file_name
```

```
copy:
mov al,[si]
mov [di],al
inc si
inc di
loop copy
mov [di],byte ptr '$'

mov bx,offset execute_param
mov dx,offset file_path

;push sp
;push ss
mov ax,4b00h
int 21h
mov cx,ax
;pop ss
;pop sp

jnc execute
cmp ax,1
je print_error_1
cmp ax,2
je print_error_2
cmp ax,5
je print_error_5
cmp ax,8
je print_error_8
cmp ax,10
je print_error_10
cmp ax,11
```

```
je print_error_11
mov dx,offset error_unknown
call print
jmp exit

print_error_1:
mov dx,offset error_1
call print
jmp exit
print_error_2:
mov dx,offset error_2
call print
jmp exit
print_error_5:
mov dx,offset error_5
call print
jmp exit
print_error_8:
mov dx,offset error_8
call print
jmp exit
print_error_10:
mov dx,offset error_10
call print
jmp exit
print_error_11:
mov dx,offset error_11
call print
jmp exit

execute:
```

```
;call print_end_line
mov ax,4d00h
int 21h

cmp ah,0
je print_termination_0
cmp ah,1
je print_termination_1
cmp ah,2
je print_termination_2
cmp ah,3
je print_termination_3

print_termination_0:
mov dx,offset termination_0
call print
jmp complete
print_termination_1:
mov dx,offset termination_1
call print
jmp exit
print_termination_2:
mov dx,offset termination_2
call print
jmp exit
print_termination_3:
mov dx,offset termination_3
call print
jmp exit

complete:
```

```
mov dx,offset exit_code
call print
call byte_to_hex
push ax
mov ah,02h
mov dl,al
int 21h
pop ax
xchg ah,al
mov ah,02h
mov dl,al
int 21h

exit:
mov ah,4ch
int 21h
main_end_prt:
main endp

code_ ends
end main
```