**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра MOЭBM**

отчет

**по лабораторной работе №6**

**по дисциплине «Операционные системы»**

**Тема: Построение модуля динамической структуры**

|  |  |  |
| --- | --- | --- |
| Студент гр. 6381 |  | Вергейчик Г. Л. |
| Преподаватель |  | Губкин А. Ф. |

Санкт-Петербург

2018

**Цель работы**

Исследование возможности построения загрузочного модуля динамической структуры. В отличии от предыдущих лабораторный работ в этой работе рассматривается приложение, состоящее из нескольких модулей, а не из одного модуля простой структуры. В этом случае разумно предположить, что все модули приложения находятся в одном каталоге и полный путь в этот каталог можно взять из среды, как это делалось в работе 2. Понятно, что такое приложение должно запускаться в соответствии со стандартами ОС.

В работе исследуется интерфейс между вызывающим и вызываемым модулями по управлению и по данным. Для запуска вызываемого модуля используется функция 4B00h прерывания int 21h. Все загрузочные модули находятся в одном каталоге. Необходимо обеспечить возможность запуска модуля динамической структуры из любого каталога.

**Описание функций**

* TETR\_TO\_HEX - Получение одного из двух символов 16-ричного числа в AX.
* BYTE\_TO\_HEX - Перевод байта из регистра AL в два символа 16-ричного числа в AX.
* Mem\_proc - Функция, необходимая для подготовки и освобождения памяти.
* Param\_block - Функция для заполнение блока параметров.
* Write - Функция для вывода на экран
* PRINT - Функция печати на экран.

**Ход работы:**

**Шаг 1:**

Программа Lab6.exe вызывает программу LR2.com, которая останавливается, ожидая символ с клавиатуры(см. Рис1).

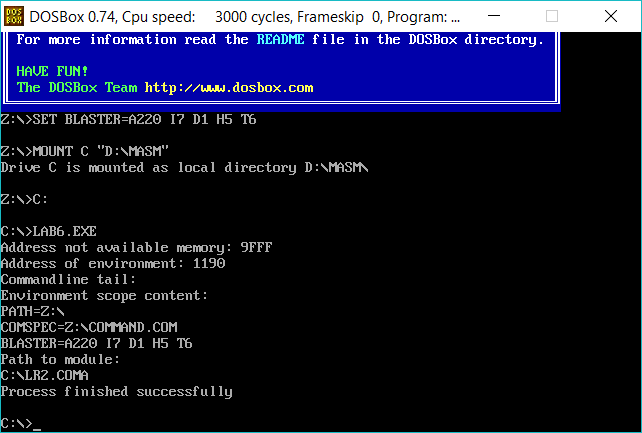


Рис.1 Вызов LR2.COM и ожидание ввода символа

**Шаг 2:**

Программа Lab6.exe вызывает программу LR2.com, которая останавливается, ожидая символ с клавиатуры (см. Рис2).

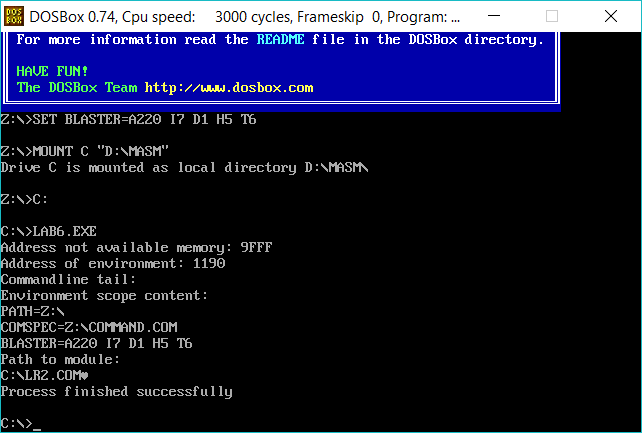


Рис. 2 Вызов LR2.COM и ожидание ввода символа

**Шаг 3:**

Запуск программы, когда текущий каталог другой(см. Рис3).

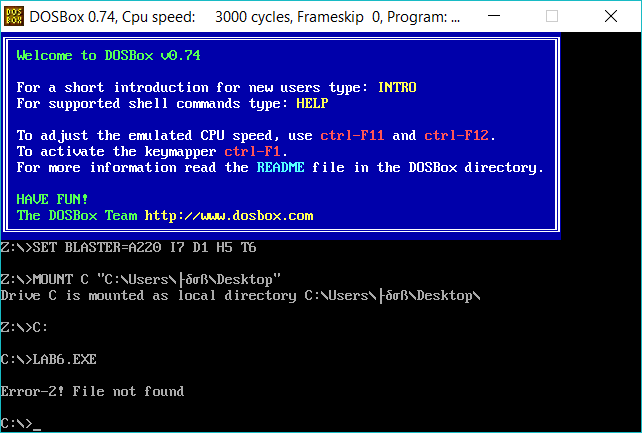


Рис. 3 Запуск программы, когда текущий каталог другой

**Шаг 4:**

Запуск программы, когда модули находятся в разных каталогах(см. Рис4).

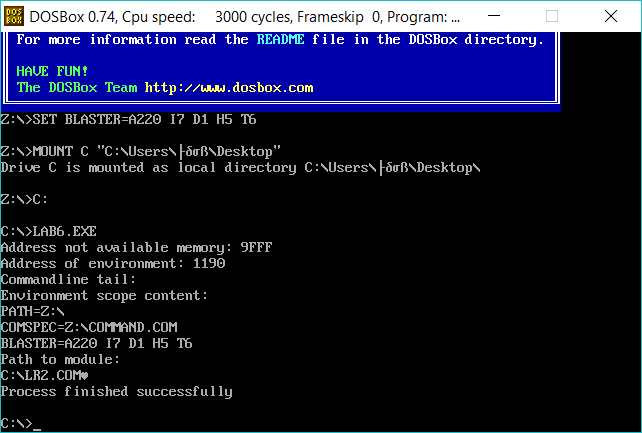


Рис. 4 Запуск программы, когда модули находятся в разных каталогах

**Ответы на контрольные вопросы:**

1. Как реализовано прерывание ctrl+c?

Если было нажато сочетание клавиш ctrl+c и флаг break поставлен в значение on, то управление передаётся по адресу 0000:008Ch. Этот адрес копируется в PSP функциями 26h и 4Ch и восстанавливается из него при выходе из программы.

1. В какой точке заканчивается вызываемая программа, если код причины завершения 0?

В месте вызова функции 4Ch прерывания 21h.

1. В какой точке заканчивается вызываемая программа по прерыванию ctrl+c?

В месте вызова функции 01h прерывания 21h.

**Вывод**

В процессе выполнения данной лабораторной работы была исследована возможность построения загрузочного модуля динамической структуры.

**Приложение**

1. Файл lab6.asm

.MODEL SMALL

;-----------------------------------------------------

CODE SEGMENT

ASSUME CS:CODE, DS:DATA, ES:DATA, SS:ASTACK

;-----------------------------------------------------

START:

JMP BEGIN

;ПРОЦЕДУРЫ

;-----------------------------------------------------

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

; байт в AL переводится в два символа шестн. числа в AX

push CX

mov AH,AL

call TETR\_TO\_HEX

xchg AL,AH

mov CL,4

shr AL,CL

call TETR\_TO\_HEX ;в AL старшая цифра

pop CX ;в AH младшая

ret

BYTE\_TO\_HEX ENDP

;-----------------------------------------------------

;подготовка места и освобождение памяти

Mem\_proc PROC

lea bx,Last\_byte

mov ax,es

sub bx,ax

mov cl,4h

shr bx,cl

mov ah,4Ah

int 21h

jnc End\_of\_free\_mem ;проверка CF

cmp ax,7

lea dx, Mem\_error7

je Error\_mem\_found

cmp ax,8

lea dx, Mem\_error8

je Error\_mem\_found

cmp ax,9

lea dx, Mem\_error9

Error\_mem\_found:

call Write

xor al,al

mov ah,4Ch

int 21H

End\_of\_free\_mem:

ret

Mem\_proc ENDP

;-----------------------------------------------------

;заполнение блока параметров

Param\_block PROC

mov ax, es

mov Params,0

mov Params+2, 80h

mov Params+4, ax

mov Params+6, 5Ch

mov Params+8, ax

mov Params+10, 6Ch

mov Params+12, ax

ret

Param\_block ENDP

;-----------------------------------------------------

Call\_LR2 PROC

;подготовка строки, содержащей путь и имя вызываемой программы

lea dx, Path

push ds

pop es

lea bx, Params

;сохранение содержимого регистров SS и SP в переменных

mov Keep\_sp, SP

mov Keep\_ss, SS

;вызов загрузчика OS

mov ax,4B00h

int 21h

jnc No\_error

push ax

mov ax,DATA

mov ds,ax

pop ax

mov SS,Keep\_ss

mov SP,Keep\_sp

cmp ax,1

lea dx, Error1

je Error

cmp ax,2

lea dx, Error2

je Error

cmp ax,5

lea dx, Error5

je Error

cmp ax,8

lea dx, Error8

je Error

cmp ax,10

lea dx, Error10

je Error

cmp ax,11

lea dx, Error11

Error:

call Write

xor al,al

mov ah,4Ch

int 21h

No\_error:

mov ax,4d00h

int 21h

cmp ah,0

lea dx, End0

je Exit

cmp ah,1

lea dx, End1

je Exit

cmp ah,2

lea dx, End2

je Exit

cmp ah,3

lea dx, End3

Exit:

call Write

ret

Call\_LR2 ENDP

;-----------------------------------------------------

Write PROC NEAR

push ax

mov ah, 09h

int 21h

pop ax

ret

Write ENDP

;-----------------------------------------------------

BEGIN:

mov ax,DATA

mov ds, ax

push ES

mov ES, ES:[2CH]

xor SI, SI

lea DI, Path

Skip:

inc SI

cmp WORD PTR ES:[SI], 0000H

jne Skip

add SI, 4

FileName:

cmp BYTE PTR ES:[SI], 00H

je EndFN

mov DL, ES:[SI]

mov [DI], DL

inc SI

inc DI

jmp FileName

EndFN:

sub DI, 8

mov [DI], 'rl'

add DI, 2

mov [DI], '.2'

add DI, 2

mov [DI], 'OC'

add DI, 2

mov BYTE PTR[DI], 'M'

inc DI

mov DL, '$'

mov [DI], DL

pop ES

call Mem\_proc

call Param\_block

call Call\_LR2

xor al,al

mov ah,4Ch ;выход

int 21h

Last\_byte:

CODE ENDS

;-----------------------------------------------------

ASTACK SEGMENT STACK

DW 64 DUP (?)

ASTACK ENDS

;-----------------------------------------------------

;Данные

DATA SEGMENT

Params dw ? ;сегментный адрес среды

dd ? ;сегмент и смещение командной строки

dd ? ;сегмент и смещение первого FCB

dd ? ;сегмент и смещение второго FCB

Mem\_error7 DB 0DH, 0AH,'Error! MCB destroyed',0DH,0AH,'$'

Mem\_error8 DB 0DH, 0AH,'Error! Not enough memory',0DH,0AH,'$'

Mem\_error9 DB 0DH, 0AH,'Error! Wrong address',0DH,0AH,'$'

Error1 DB 0DH, 0AH,'Error-1! Wrong number of function',0DH,0AH,'$'

Error2 DB 0DH, 0AH,'Error-2! File not found',0DH,0AH,'$'

Error5 DB 0DH, 0AH,'Error-5! Disk error',0DH,0AH,'$'

Error8 DB 0DH, 0AH,'Error-8! Not enough memory',0DH,0AH,'$'

Error10 DB 0DH, 0AH,'Error-10! Incorrect environment string',0DH,0AH,'$'

Error11 DB 0DH, 0AH,'Error-11! Error with format',0DH,0AH,'$'

End0 DB 0DH, 0AH,'Process finished successfully',0DH,0AH,'$'

End1 DB 0DH, 0AH,'Ctrl-Break end!',0DH,0AH,'$'

End2 DB 0DH, 0AH,'Device error end!',0DH,0AH,'$'

End3 DB 0DH, 0AH,'Function 31h end!',0DH,0AH,'$'

Path DB 50 dup(0)

Keep\_ss DW 0

Keep\_sp DW 0

DATA ENDS

END START