**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра MOЭBM**

отчет

**по лабораторной работе №7**

**по дисциплине «Операционные системы»**

**Тема: Построение модуля оверлейной структуры**

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

Санкт-Петербург

2018

**Цель работы**

Исследование возможности построения загрузочного модуля оверлейной структуры. Приложение в данной работе состоит из нескольких модулей, все модули находятся в одном каталоге и полный путь в этот каталог берется из среды.

**Описание функций**

* CUT\_MEM – Процедура, которая освобождает избыточную для данного приложения память с помощью функции 4AH прерывания 21h;
* GET\_FILE\_NAME – процедура для извлечения полного имени файла из среды программы;
* SET\_DTA\_BLOCK – процедура для установки адреса DTA блока;
* WHAT\_ERROR\_IN\_CALL – данная процедура вызывается, если при запуске подпрограммы произошла ошибка. Выводит на экран информацию об ошибке.
* GET\_MEM\_FOR\_OVL – данная процедура выделяет память для оверлея.
* CALL\_OVL – вызов оверлея;
* WRITE– вывод сообщения;

**Ход работы:**

**Шаг 1:**

Запуск программы, когда текущим каталогом является каталог с разработанными модулями lab7, first и second. (см. Рис1).

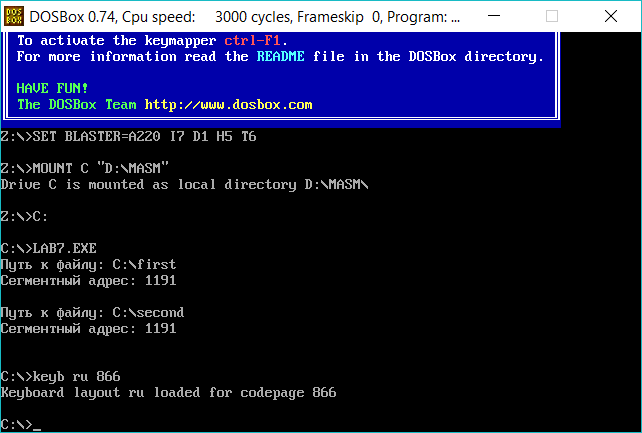


Рис.1 Запуск программы, когда текущим каталогом является каталог с разработанными модулями lab7, first и second

**Шаг 2:**

Запуск программы, когда текущим каталогом является каталог, отличный от того, в котором находятся модули lab7,first и second.(см. Рис2).

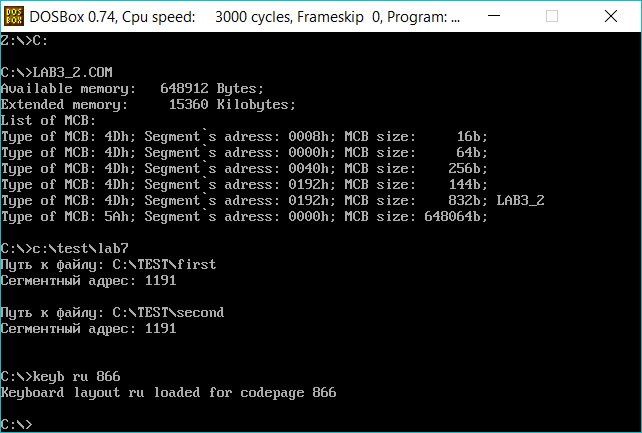


Рис. 2 Запуск программы, когда текущим каталогом является каталог, отличный от того, в котором находятся модули lab7,first и second.

**Шаг 3:**

Запуск программы, когда в каталоге нет второго оверлея(см. Рис3).

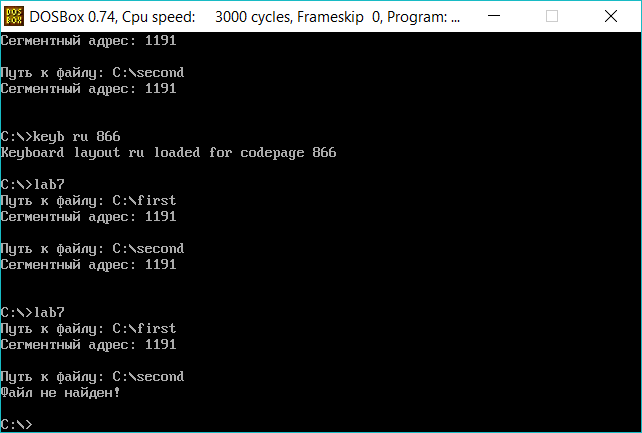


Рис. 3 Запуск программы, когда в каталоге нет второго оверлея.

**Ответы на контрольные вопросы:**

Как должна быть устроена программа, если в качестве оверлейного сегмента использовать **.СОМ** модули?

Так как в com модуле присутствует свой PSP, то его вызов необходимо производить по смещению 100h, также необходимо сохранять регистры и восстанавливать их по завершению работы com модуля.

**Вывод**

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

**Приложение**

1. Файл lab7.asm

DATA SEGMENT

PREFIX DB 'Путь к файлу: $'

FILENAME DB 64 DUP(0)

OVL\_1 DB 'first',0

OVL\_2 DB 'second',0

DTA\_BLOCK DB 43 DUP(0) ;буфер для заполнения при поиске файла

OVL\_SEGMENT DW 0

OVL\_ADRESS DD 0 ;дальний адрес оверлея

KEEP\_PSP DW 0

;Сообщения об ошибках при поиске файла

ERR\_FIND\_NUM\_2 DB 0DH, 0AH,'Файл не найден!',0DH,0AH,'$'

ERR\_FIND\_NUM\_3 DB 0DH, 0AH,'Маршрут не найден!',0DH,0AH,'$'

;Сообщения об ошибках запуска программы

ERR\_CALL\_NUM\_1 DB 0DH, 0AH,'Неверный номер функции!',0DH,0AH,'$'

ERR\_CALL\_NUM\_2 DB 0DH, 0AH,'Файл не найден!',0DH,0AH,'$'

ERR\_CALL\_NUM\_5 DB 0DH, 0AH,'Ошибка диска!',0DH,0AH,'$'

ERR\_CALL\_NUM\_8 DB 0DH, 0AH,'Недостаточный объем памяти!',0DH,0AH,'$'

ERR\_CALL\_NUM\_10 DB 0DH, 0AH,'Неправильная строка среды!',0DH,0AH,'$'

;Сообщение об ошибке при выделении памяти

ERR\_MEM\_AL DB 'Ошибка при выделении памяти!',0DH,0AH,'$'

DATA ENDS

ASTACK SEGMENT STACK

DW 64 DUP (0)

ASTACK ENDS

CODE SEGMENT

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

;Процедура для вывода сообщения

WRITE PROC

PUSH AX

MOV AH, 09H

INT 21H

POP AX

RET

WRITE ENDP

;установка адреса DTA\_BLOCK

SET\_DTA\_BLOCK PROC

PUSH DX

LEA DX, DTA\_BLOCK

MOV AH,1AH

INT 21H

POP DX

SET\_DTA\_BLOCK ENDP

;Процедура для освобождения лишней памяти

CUT\_MEM PROC

LEA BX, END\_OF\_THIS\_PROGRAMM

MOV AX, ES

SUB BX, AX

MOV CL, 4

SHR BX, CL

MOV AH, 4AH

INT 21H

JNC NOT\_MEMORY\_ERROR

LEA DX, ERR\_MEM\_AL

CALL WRITE

MOV AH, 4CH

INT 21H

NOT\_MEMORY\_ERROR:

RET

CUT\_MEM ENDP

;функция для определения имени запускаемой программы

;в BX необходимо переместить смещения строки с именем файла

GET\_FILE\_NAME PROC

PUSH ES

MOV ES, ES:[2CH]

XOR SI, SI

LEA DI, FILENAME

SKIP\_ENV\_PART:

INC SI

CMP WORD PTR ES:[SI], 0000H

JNE SKIP\_ENV\_PART

ADD SI, 4

PATH\_SYMBOL:

CMP BYTE PTR ES:[SI], 00H

JE END\_FILE\_NAME

MOV DL, ES:[SI]

MOV [DI], DL

INC SI

INC DI

JMP PATH\_SYMBOL

END\_FILE\_NAME: ;поиск последней директории

DEC SI

DEC DI

CMP BYTE PTR ES:[SI], '\'

JNE END\_FILE\_NAME

INC DI

MOV SI, BX ;Загрузка из регистра строки

PUSH DS

POP ES

SIMP\_OVL: ;заполнение имени файла

LODSB

STOSB

CMP AL, 0

JNE SIMP\_OVL

MOV BYTE PTR [DI], '$'

LEA DX, PREFIX

CALL WRITE

LEA DX, FILENAME

CALL WRITE

POP ES

RET

GET\_FILE\_NAME ENDP

;Выделение памяти для оверлея

GET\_MEM\_FOR\_OVL PROC

PUSH DS

PUSH DX

PUSH CX

XOR CX, CX

LEA DX, FILENAME

MOV AH,4EH ;поиск файла

INT 21H

JNC FILE\_FOUNDED

CMP AX,3

LEA DX, ERR\_FIND\_NUM\_3

JE ERROR\_FIND\_EXIT

LEA DX, ERR\_FIND\_NUM\_2

ERROR\_FIND\_EXIT: ;выход при обнаружении ошибки

CALL WRITE

POP CX

POP DX

POP DS

XOR AL,AL

MOV AH,4CH

INT 21H

FILE\_FOUNDED: ;файл найден

PUSH ES

PUSH BX

LEA BX, DTA\_BLOCK

MOV DX,[BX+1CH]

MOV AX,[BX+1AH]

MOV CL,4H

SHR AX,CL

MOV CL,12

SAL DX, CL

ADD AX, DX

INC AX

MOV BX,AX

MOV AH,48H

INT 21H

JC ERR\_EXIT

MOV OVL\_SEGMENT, AX

POP BX

POP ES

POP CX

POP DX

POP DS

RET

ERR\_EXIT:

LEA DX, ERR\_MEM\_AL

CALL WRITE

MOV AH, 4CH

INT 21H

GET\_MEM\_FOR\_OVL ENDP

;Вызов оверлея

CALL\_OVL PROC

PUSH DX

PUSH BX

PUSH AX

MOV BX, SEG OVL\_SEGMENT

MOV ES, BX

LEA BX, OVL\_SEGMENT

LEA DX, FILENAME

MOV AX, 4B03H

INT 21H

JNC NO\_ERROR\_IN\_CALL

CALL WHAT\_ERROR\_IN\_CALL

JMP EX\_FROM\_CALL

NO\_ERROR\_IN\_CALL:

MOV AX,DATA

MOV DS,AX

MOV AX, OVL\_SEGMENT

MOV WORD PTR OVL\_ADRESS+2, AX

CALL OVL\_ADRESS

MOV AX, OVL\_SEGMENT

MOV ES, AX

MOV AX, 4900H

INT 21H

MOV AX,DATA

MOV DS,AX

EX\_FROM\_CALL:

MOV ES, KEEP\_PSP

POP AX

POP BX

POP DX

RET

CALL\_OVL ENDP

;Процедура для определения ошибки, если программа не запустилась

WHAT\_ERROR\_IN\_CALL PROC

CMP AX,1

LEA DX, ERR\_CALL\_NUM\_1

JE WRITE\_ERR

CMP AX,2

LEA DX, ERR\_CALL\_NUM\_2

JE WRITE\_ERR

CMP AX,5

LEA DX, ERR\_CALL\_NUM\_5

JE WRITE\_ERR

CMP AX,8

LEA DX, ERR\_CALL\_NUM\_8

JE WRITE\_ERR

CMP AX,10

LEA DX, ERR\_CALL\_NUM\_10

WRITE\_ERR:

CALL WRITE

ret

WHAT\_ERROR\_IN\_CALL ENDP

BEGIN PROC FAR

mov ax, DATA

mov ds, ax

mov Keep\_psp, ES

CALL CUT\_MEM

CALL SET\_DTA\_BLOCK

LEA BX, OVL\_1

CALL GET\_FILE\_NAME ;

CALL GET\_MEM\_FOR\_OVL

CALL CALL\_OVL

LEA BX, OVL\_2

CALL GET\_FILE\_NAME

CALL GET\_MEM\_FOR\_OVL

CALL CALL\_OVL

MOV AH, 4CH

INT 21H

END\_OF\_THIS\_PROGRAMM:

BEGIN ENDP

CODE ENDS

END BEGIN

1. Файл first.asm

OVL\_SEG SEGMENT

ASSUME CS:OVL\_SEG, DS:nothing, SS:nothing, ES:nothing

MAIN PROC FAR

PUSH DS

PUSH AX

PUSH DI

PUSH DX

MOV AX, CS

MOV DS, AX

LEA DI, SEG\_ADR

ADD DI, 23

CALL WRD\_TO\_HEX

LEA DX, SEG\_ADR

MOV AH, 09H

INT 21H

POP DX

POP DI

POP AX

POP DS

RETF

MAIN ENDP

SEG\_ADR db 0DH, 0AH,'Сегментный адрес: ', 0DH, 0AH,0DH,0AH, '$'

TETR\_TO\_HEX PROC

and AL,0Fh

cmp AL,09

jbe NEXT

add AL,07

NEXT:

add AL,30h

ret

TETR\_TO\_HEX ENDP

BYTE\_TO\_HEX PROC

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

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

OVL\_SEG ENDS

END MAIN