

# **Лабораторная работа №6. Основы работы с Midnight Commander (mc). Структура программы на языке ассемблера NASM**

**Архитектура ЭВМ**

Плескачева Елизавета Андреевна НММбд-02-22

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Выполнение лабораторной работы</b>	<b>6</b>
2.1	Подключение внешнего файла . . . . .	9
<b>3</b>	<b>Задания для самостоятельной работы</b>	<b>13</b>
3.1	Задание 1 . . . . .	13
3.2	Задание 2 . . . . .	14
<b>4</b>	<b>Выводы</b>	<b>15</b>

# Список иллюстраций

2.1	Midnight Commander . . . . .	6
2.2	Создание папки . . . . .	6
2.3	команда touch . . . . .	6
2.4	Созданный файл . . . . .	7
2.5	Открытие пустого файла . . . . .	7
2.6	Код введенный в файл . . . . .	8
2.7	Окно сохранение файла . . . . .	8
2.8	Содержимое файла lab6-1.asm . . . . .	9
2.9	Запуск и ввод из консоли . . . . .	9
2.10	Папка Downloads справа . . . . .	10
2.11	Копирование файла in_out_asm . . . . .	10
2.12	Копирование файла lab6-2.asm . . . . .	10
2.13	Скопированные файлы . . . . .	11
2.14	Исходный текст для lab6-2.asm . . . . .	12
2.15	Вывод программы на экран . . . . .	12
3.1	Программа lab6-1-samostoyatel'naya . . . . .	13
3.2	Запуск lab6-1-samostoyatel'naya . . . . .	14
3.3	Программа выводящая строку на экран с использованием внешнего файла . . . . .	14
3.4	Вывод программы на экран . . . . .	14

## Список таблиц

# 1 Цель работы

Приобретение практических навыков работы в Midnight Commander. Освоение инструкций языка ассемблера `mov` и `int`.

## 2 Выполнение лабораторной работы

Откроем терминал и введем mc, что бы запустить Midnight Commander.

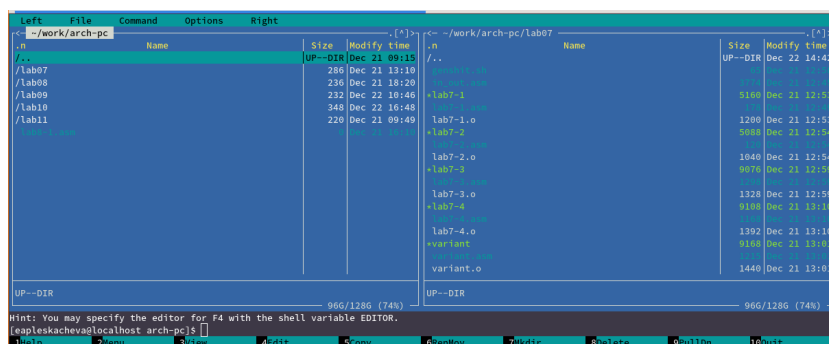


Рис. 2.1: Midnight Commander

Создадим папку lab06, нажав F7

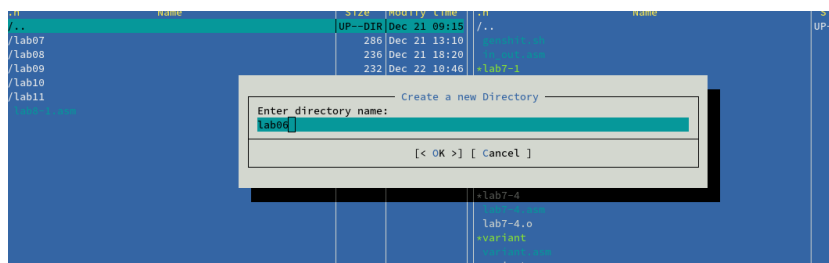


Рис. 2.2: Создание папки

Создадим файл lab06-1.asm с помощью строки ввода

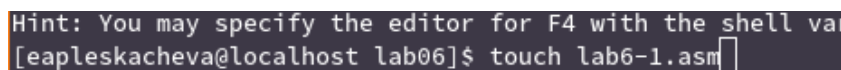


Рис. 2.3: команда touch

Файл появился в MC

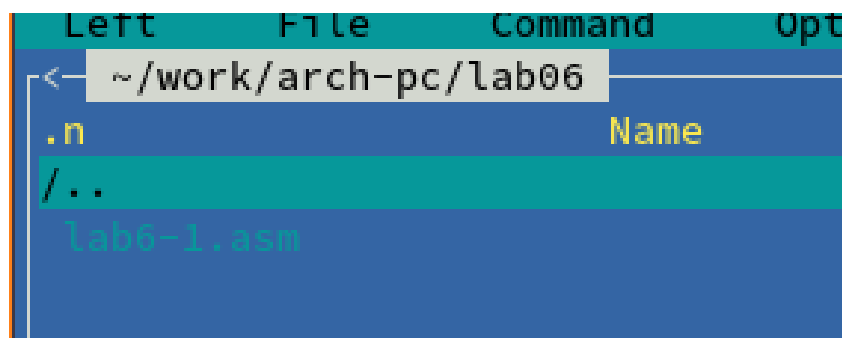


Рис. 2.4: Созданный файл

Откроем этот файл, нажав F4

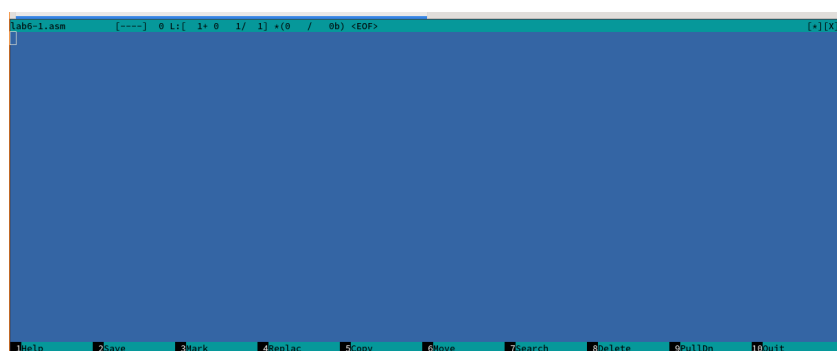


Рис. 2.5: Открытие пустого файла

Введем в файл код из листинга 6.1

```

lab6-1.asm      [-M--]  0 L:[  1+ 5   6/ 40] *(127 /1460b) 0109 0x06D
; input output string program

; --- Variable declaration --- ;

SECTION .data                ;initializing data section
msg:    DB 'Введите строку:', 10 ; message + \n
msgLen: EQU $-msg            ; len of 'msg'

SECTION .bss                 ; no initialized data section
buf1:   RESB 80              ; 80 bytes size buff

; --- Programm text --- ;

SECTION .text                ; Code itself
GLOBAL _start                ; Beginning of the programm
_start:                       ; entry point

; --- 'write' syscall --- ;
; printing string from 'msg' with len 'msgLen'

mov eax, 4                    ; Syscal for 'sys_write'
mov ebx, 1                    ; descriptor - std output
mov ecx, msg                  ; mov addres of 'msg' to 'ecx'
mov edx, msgLen               ; mov size of 'msg' to 'edx'
int 80h                       ; kernel call

; --- 'read' syscall --- ;
mov eax, 3                    ; syscall for 'sys_read'
mov ebx, 0                    ; descriptor 0 - std input
mov ecx, buf1                 ; mov buffer addres for input to ecx
mov edx, 80                   ; mov string len to edx
int 80h                       ; kernel call

; --- 'exit' syscall ---;
mov eax, 1                    ; syscall for 'sys_exit'
mov ebx, 0                    ; move 0 return code to ebx
int 80h                       ; kernel call

```

Рис. 2.6: Код введенный в файл

Сохраним файл

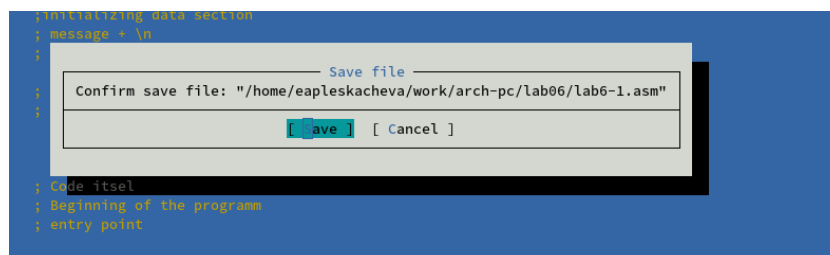


Рис. 2.7: Окно сохранение файла

Откроем сохраненный файл через МС, нажав F3



```

/home/eapleskacheva/work/arch-pc/lab06/lab6-1.asm
; input output string program

; --- Variable declaration --- ;

SECTION .data                                ;initializing data section
msg:    DB 'Введите строку:', 10             ; message + \n
msglen: EQU $-msg                            ; len of 'msg'

SECTION .bss                                ; no initialized data section
buf1:   RESB 80                             ; 80 bytes size buff

; --- Programm text --- ;

SECTION .text                                ; Code itsel
GLOBAL _start                               ; Beginning of the programm
_start:                                     ; entry point

; --- `write` syscall --- ;
; printing string from 'msg' with len 'msglen'

mov eax, 4                                  ; Syscal for `sys_write`
mov ebx, 1                                  ; descriptor - std output
mov ecx, msg                                ; mov addres of 'msg' to 'ecx'
mov edx, msglen                             ; mov size of 'msg' to 'edx'
int 80h                                     ; kernel call

```

Рис. 2.8: Содержимое файла lab6-1.asm

Текст файла сохранился

Выйдем из МС, нажав F10

Создадим исполняемый файл и запустим его:

```

nasm -f elf ./lab6-1.asm
ld -m elf_i386 -o ./lab6-1 ./lab6-1.o
./lab6-1

Введите строку:
Плескачева Лиза
[eapleskacheva@localhost lab06]$ 

```

Рис. 2.9: Запуск и ввод из консоли

## 2.1 Подключение внешнего файла

Откроем папку Downloads через МС

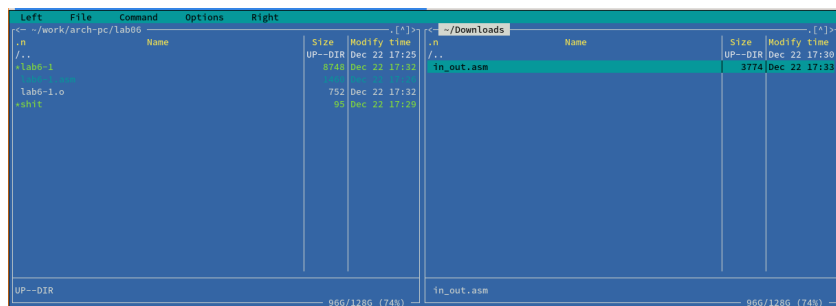


Рис. 2.10: Папка Downloads справа

Скопируем файл in\_out.asm в lab06

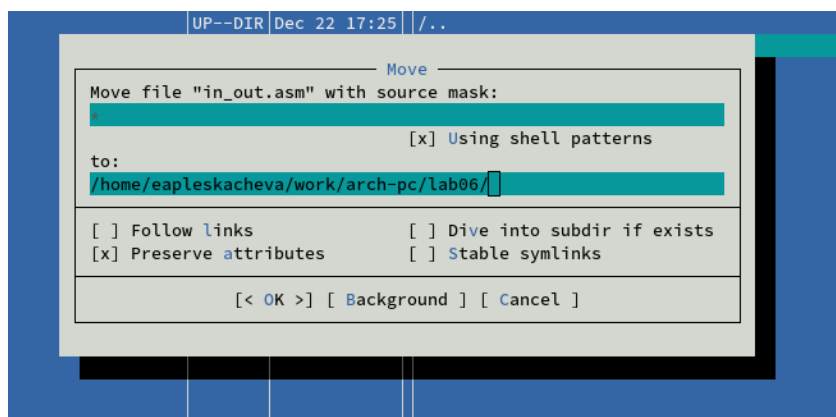


Рис. 2.11: Копирование файла in\_out\_asm

Скопируем файл lab6-1.asm в lab6-2.asm

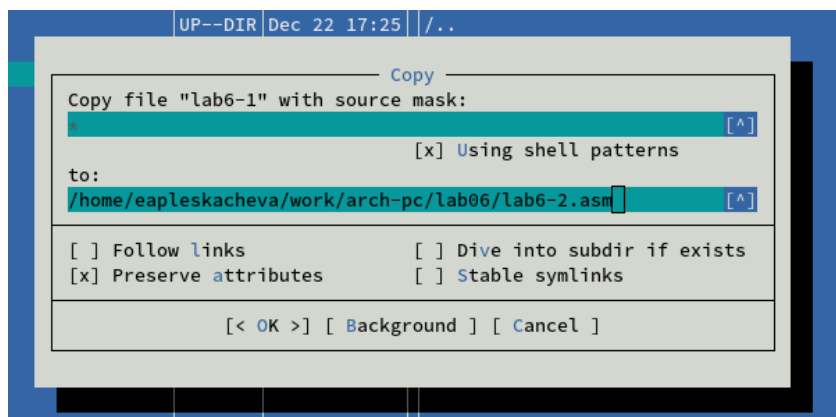


Рис. 2.12: Копирование файла lab6-2.asm

Как видим, оба файла появились в папке lab06

```
../  
in_out.asm  
*lab6-1  
lab6-1.asm  
lab6-1.o  
*lab6-2.asm  
*shit
```

Рис. 2.13: Скопированные файлы

Введем файл из листинга 6.2 в lab6-2.asm

```

include 'in_out.asm'

SECTION .data
msg:    DB 'Введите строку:', 10

SECTION .bss
buf1:   RESB 80

SECTION .text
GLOBAL _start
_start:

    mov    eax, msg
    call   sprint

    mov    ecx, buf1
    mov    edx, 80
    call   sread

    call   quit

```

Рис. 2.14: Исходный текст для lab6-2.asm

Создадим исполняемый файл и запустим

```

nasm -f elf ./lab6-2.asm
ld -m elf_i386 -o ./lab6-2 ./lab6-2.o
./lab6-2

Введите строку:
Лиза Плескачева
[earleskacheva@localhost lab06]$

```

Рис. 2.15: Вывод программы на экран

Мы ввели свое ФИО и программа завершилась.

Если бы мы написали вместо sprint, sprintLF, то после сообщения “Введите строку” был бы еще и перенос строки.

## 3 Задания для самостоятельной работы

### 3.1 Задание 1

Напишем программу, которая будет выводить на экран введенное сообщение, без использования `in_out.asm`

```
SECTION .data
msg:    DB 'Введите строку:', 10
msgLen: EQU $-msg

SECTION .bss
buf1:   RESB 80

SECTION .text
GLOBAL _start
_start:

    mov eax, 4
    mov ebx, 1
    mov ecx, msg
    mov edx, msgLen
    int 80h

    mov eax, 3
    mov ebx, 0
    mov ecx, buf1
    mov edx, 80
    int 80h

    mov eax, 4
    mov ebx, 1
    mov ecx, msg+msgLen
    mov edx, 80
    int 80h

    mov eax, 1
    mov ebx, 0
    int 80h
```

Рис. 3.1: Программа lab6-1-samostoyatel'naya

Запустим ее.

```
[eapleskacheva@localhost lab06]$ nasm -f elf ./lab6-1-samostoyatel'naya.asm
[eapleskacheva@localhost lab06]$ ld -m elf_i386 -o ./lab6-1-samostoyatel'naya.o ./lab6-1-samostoyatel'naya.o
[eapleskacheva@localhost lab06]$ ./lab6-1-samostoyatel'naya
Введите строку:
Лиза Плескачева
Лиза Плескачева
[eapleskacheva@localhost lab06]$
```

Рис. 3.2: Запуск lab6-1-samostoyatel'naya

## 3.2 Задание 2

Напишем такую же программу, используя in\_out.asm

```
lab6-2-samostoyatel'naya.asm [----] 0 L: [ 1+ 0 1/ 22] *(0 / 278b) 0037 0x025
#include 'in_out.asm'
SECTION .data
msg: DB 'Введите строку:', 10

SECTION .bss
buf1: RESB 80
SECTION .text
GLOBAL _start
_start:

    mov     eax, msg
    call    sprintf

    mov     ecx, buf1
    mov     edx, 80
    call    sread

    mov     eax, buf1
    call    sprintf

    call    quit
```

Рис. 3.3: Программа выводющая строку на экран с использованием внешнего файла

Скомпилируем и запустим программу

```
[eapleskacheva@localhost lab06]$ nasm -f elf ./lab6-2-samostoyatel'naya.asm
[eapleskacheva@localhost lab06]$ ld -m elf_i386 -o ./lab6-2-samostoyatel'naya.o ./lab6-2-samostoyatel'naya.o
[eapleskacheva@localhost lab06]$ ./lab6-2-samostoyatel'naya
Введите строку:
Лиза Плескачева
Лиза Плескачева
[eapleskacheva@localhost lab06]$
```

Рис. 3.4: Вывод программы на экран

Как видим, вывод происходит с переносами строки, так как мы используем sprintf, а не printf

## 4 Выводы

Мы приобрели практические навыки в работе с Midnight Commander и освоили инструкции `mov` и `int` языка ассемблера NASM