

# **labrotary work №5**

**computer architecture**

Ali hosseinabadi

# **Content**

<b>1 Objective of the Work</b>	<b>5</b>
<b>2 Task</b>	<b>6</b>
<b>3 Theoretical Introduction</b>	<b>7</b>
3.1 Conducting the Laboratory Work . . . . .	8
3.1.1 Basics of Working with Midnight Commander . . . . .	8
3.1.2 Connecting an External File . . . . .	11
3.2 Self-Study Task . . . . .	13
<b>4 Conclusions</b>	<b>18</b>
<b>References</b>	<b>19</b>

# **Список иллюстраций**

3.1 Opening Midnight Commander . . . . .	8
3.2 Midnight Commander Interface . . . . .	8
3.3 Opened Directory arch-pc . . . . .	9
3.4 Creating Working Subdirectory . . . . .	9
3.5 Creating a File in Midnight Commander . . . . .	10
3.6 Editing the File in Midnight Commander . . . . .	10
3.7 Checking Saved Changes . . . . .	11
3.8 Running the Modified Program . . . . .	11
3.9 Copying the File to the Working Directory . . . . .	12
3.10 Changing the Program . . . . .	12
3.11 Launching the Modified Program . . . . .	12
3.12 Running the Modified Program with Different Subroutine . . . . .	13
3.13 Editing the Copy . . . . .	13
3.14 Running the Program . . . . .	14
3.15 Editing a copy . . . . .	15
3.16 Running my program . . . . .	16

# **Список таблиц**

# **1 Objective of the Work**

The objective of this laboratory work is to acquire practical skills in using Midnight Commander and to learn the basic instructions of the assembly language: `mov` and `int`.

## **2 Task**

1. Basics of working with Midnight Commander
2. Structure of a program written in NASM assembly language
3. Including an external file
4. Completing the assignments for self-study

### 3 Theoretical Introduction

Midnight Commander (or simply mc) is a program that allows users to browse directory structures and perform basic file system management operations. Thus, mc functions as a file manager. Midnight Commander makes working with files more convenient and visually understandable.

A program written in the NASM assembly language typically consists of three sections: the program code section (SECTION .text), the initialized data section (known at compile time) (SECTION .data), and the uninitialized data section (which is allocated memory at compile time but assigned values during program execution) (SECTION .bss).

To declare initialized data in the .data section, the directives DB, DW, DD, DQ, and DT are used, which reserve memory and specify which values should be stored in that memory:

- DB (define byte) – defines a variable of 1 byte;
- DW (define word) – defines a variable of 2 bytes (word);
- DD (define double word) – defines a variable of 4 bytes (double word);
- DQ (define quad word) – defines a variable of 8 bytes (quad word);
- DT (define ten bytes) – defines a variable of 10 bytes.

### **3.1 Conducting the Laboratory Work**

### **3.1.1 Basics of Working with Midnight Commander**

By entering the appropriate command in the terminal (Figure -fig. 3.1), I open Midnight Commander (Figure -fig. 3.2).

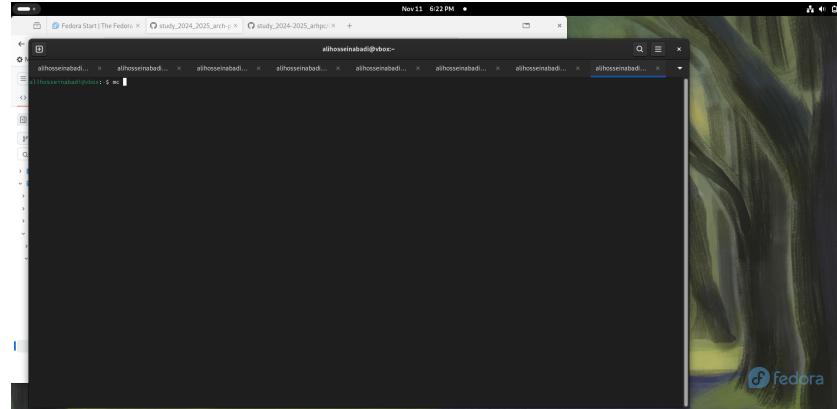


Рис. 3.1: Opening Midnight Commander

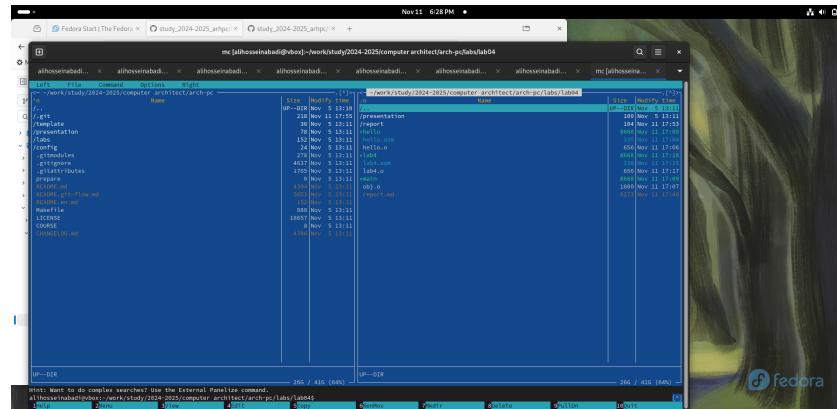


Рис. 3.2: Midnight Commander Interface

I navigate to the directory created in the previous laboratory work (Figure -fig. 3.3).

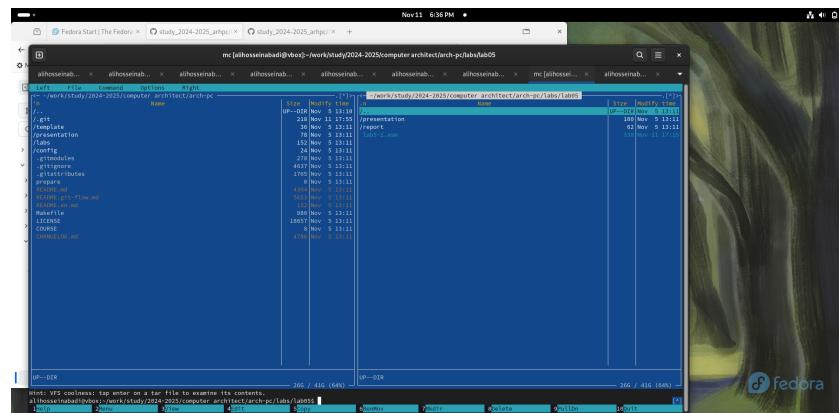


Рис. 3.3: Opened Directory arch-pc

Using the function key, I create a subdirectory lab05, where I will work (Figure - fig. 3.4).

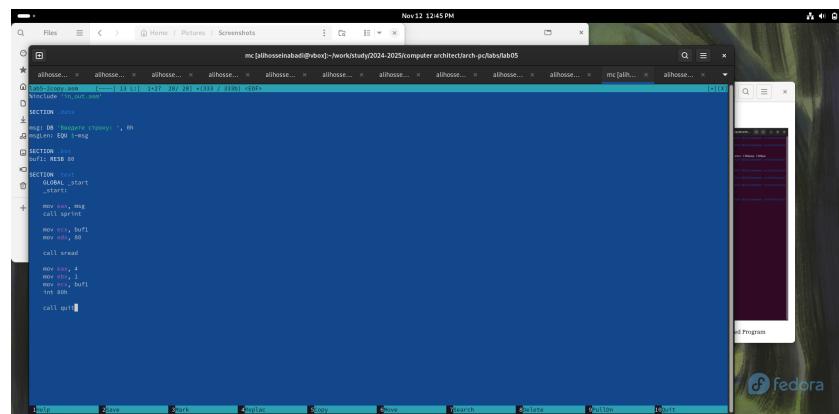


Рис. 3.4: Creating Working Subdirectory

In the input line, I enter the command touch and create a file (Figure -fig. 3.5).

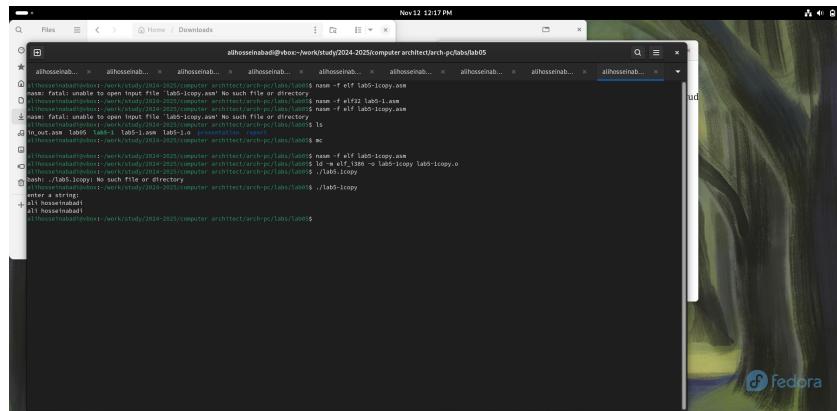


Рис. 3.5: Creating a File in Midnight Commander

I use F4 to open the newly created file and enter the code from the listing (Figure -fig. 3.6).

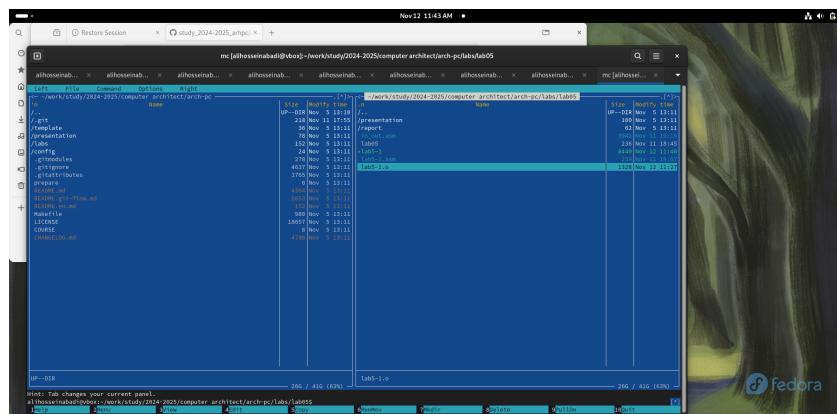


Рис. 3.6: Editing the File in Midnight Commander

I check the saved changes using the F3 key (Figure -fig. 3.7).

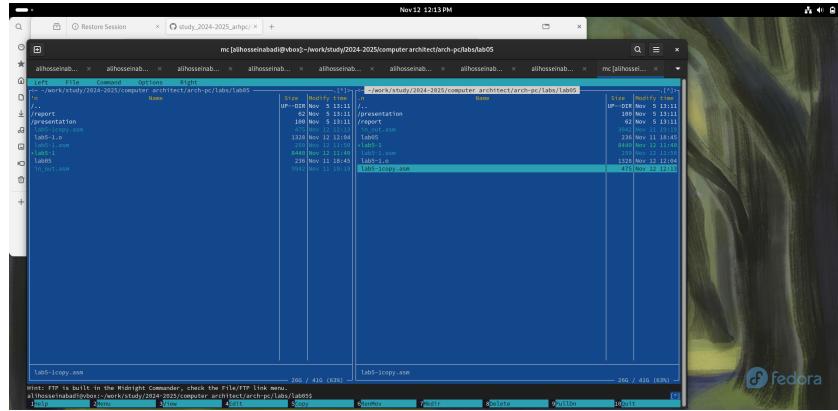


Рис. 3.7: Checking Saved Changes

I compile and execute the modified file (Figure -fig. 3.8).

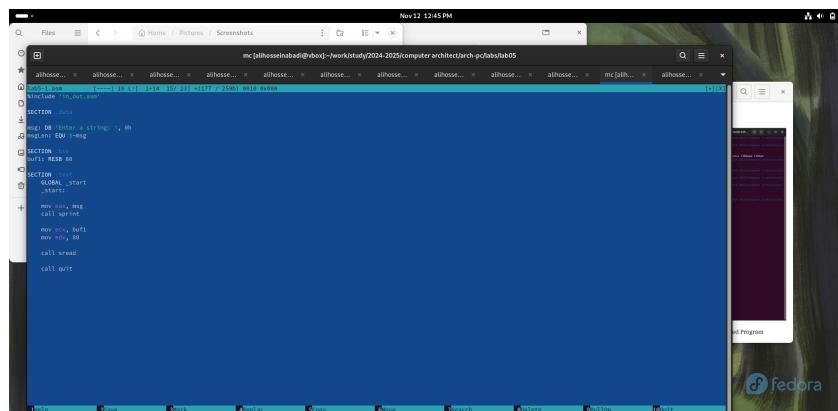


Рис. 3.8: Running the Modified Program

### 3.1.2 Connecting an External File

I save the file downloaded from TUIS to a shared folder on my computer, then in the virtual machine, I go to the directory of the shared folder, copy the file to the working subdirectory (Figure -fig. 3.9).

Рис. 3.9: Copying the File to the Working Directory

I include subroutines from the included file in the copy of the file (Figure -fig. 3.10).

The screenshot shows a terminal window titled "Nov12 12:30 PM" running on a Fedora system. The command entered is:

```
mc [aihossaini@vbox]~/work/study/2024-2025/computer/arch/pc/lab05
```

The assembly code displayed is:

```
alhossaini@vbox:~/work/study/2024-2025/computer/arch/pc/lab05$ nasm -f elf32 -o lab05.o lab05.asm
alhossaini@vbox:~/work/study/2024-2025/computer/arch/pc/lab05$ ld -m elf_i386 -o lab05 lab05.o
alhossaini@vbox:~/work/study/2024-2025/computer/arch/pc/lab05$ ./lab05
Hello, world!
alhossaini@vbox:~/work/study/2024-2025/computer/arch/pc/lab05$
```

The assembly code itself is as follows:

```
section .data
msg DB "Hello, user! string", 0
newline DB 10, 13, 0

section .text
global _start
_start:
    mov ax, msg
    call sprint
    mov ectx, buf1
    mov edx, 80
    call read
    call quit
```

Рис. 3.10: Changing the Program

I translate, compose, and launch the program with the included file (Figure -fig. 3.12).

Рис. 3.11: Launching the Modified Program

I edit the file and replace the `sprintLF` subroutine with `sprint`. The difference between the two subroutines is that the second one prompts the input on the same line (Figure -fig. 3.13).

```

Nov 12 22:45 PM
mc@alhosseiniabadi:~/work/study/2024-2025/computer architect/arch-pc/labs/lab05
$ nasm -f elf lab5-1.asm
$ ld -o lab5-1 lab5-1.o
$ ./lab5-1
al

```

Рис. 3.12: Running the Modified Program with Different Subroutine

## 3.2 Self-Study Task

I create a copy of `lab5-1.asm`, editing it so that the string I entered from the keyboard is displayed at the end (Figure -fig. 3.14).

```

Nov 12 22:45 AM
alhosseiniabadi:~/work/study/2024-2025/computer architect/arch-pc/labs/lab05
$ nasm -f elf lab5-1.asm
$ ld -o lab5-1 lab5-1.o
$ ./lab5-1
alhosseiniabadi:~/work/study/2024-2025/computer architect/arch-pc/labs/lab05$ al

```

Рис. 3.13: Editing the Copy

I translate, compose, and run my program (Figure -fig. 3.15).

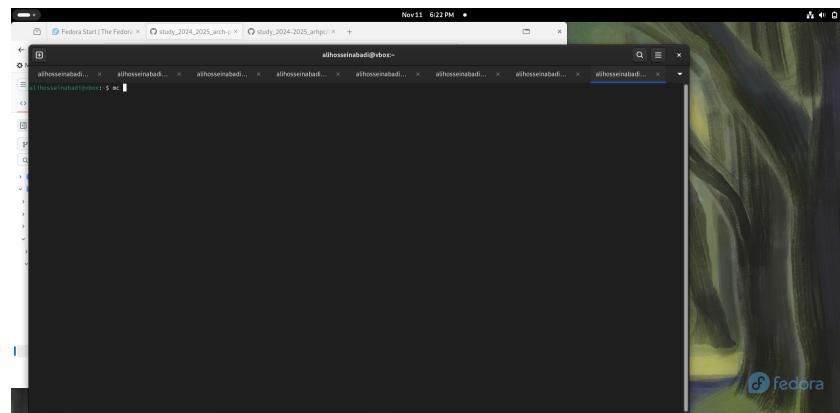


Рис. 3.14: Running the Program

Here is the code:

```
SECTION .data
msg: DB 'write a string:', 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, buf1
mov    edx, buf1
int    80h
```

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

I create a copy of lab5-2.asm, edit it so that the line I entered from the keyboard is displayed at the end (Fig. -fig. 3.16).

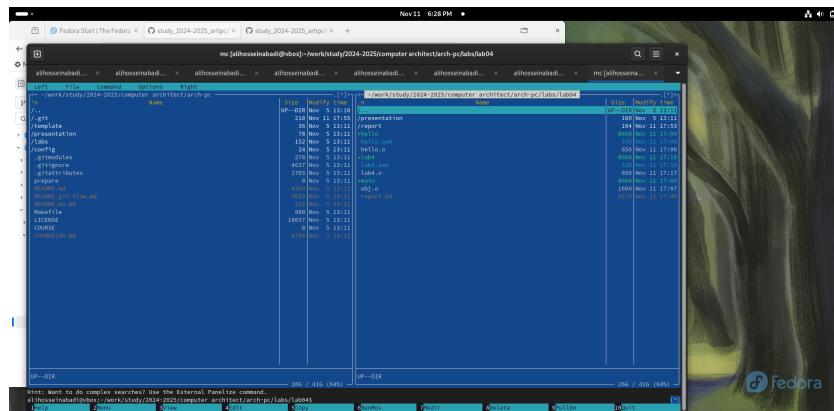


Рис. 3.15: Editing a copy

I translate, compose and run my program (Fig. -fig. ??).

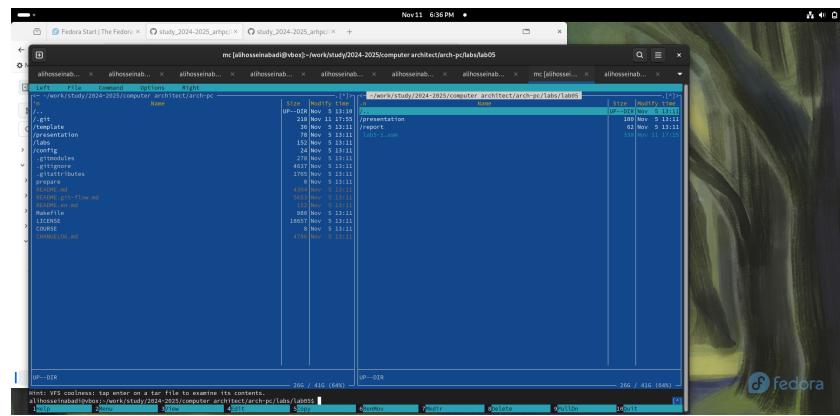


Рис. 3.16: Running my program

code:

```
%include 'in_out.asm'
```

```
SECTION .data
```

```
msg: DB 'write a string: ', 0h
msgLen: EQU $-msg
```

```
SECTION .bss
```

```
buf1: RESB 80
```

```
SECTION .text
```

```
GLOBAL _start
```

```
_start:
```

```
    mov eax, msg
    call sprint
```

```
    mov ecx, buf1
    mov edx, 80
```

```
call sread
```

```
mov eax, 4
mov ebx, 1
mov ecx, buf1
int 80h
```

```
call quit
```

## **4 Conclusions**

During this lab I gained practical skills in working in Midnight Commander and also mastered the assembly language instructions mov and int.

# **References**

1. sample
2. course on tuis
3. labrotary work №5
4. prograaming in nasmlanguage