Report on Laboratory Work No. 4

Discipline: Computer Architecture

Erfan Hosseinabadi

Table of Contents

List of Illustrations

List of Tables

1 Objective of the Work

The objective of this laboratory work is to master the procedures for compiling and building programs written in NASM assembly language.

2 Assignment

1. Create a Hello World program!
2. Work with the NASM translator.
3. Work with the extended syntax of the NASM command line.
4. Work with the LD linker.
5. Run the executable file.
6. Complete independent work assignments.

3 Theoretical Introduction

The primary functional elements of any computer are the central processor, memory, and peripheral devices.

The interaction of these devices occurs through a shared bus to which they are connected. Physically, the bus consists of a large number of conductors that connect devices to each other. In modern computers, these conductors are made in the form of conductive tracks on the motherboard.

The main task of the processor is to process information and organize the coordination of all computer nodes. The central processor includes the following units:

- Arithmetic Logic Unit (ALU) — performs logical and arithmetic operations necessary to process information stored in memory;
- Control Unit (CU) — provides control and checks all devices of the computer;
- Registers — ultra-fast temporary storage within the processor for storing intermediate results of instructions; processor registers are divided into two types: general-purpose registers and special registers.

To write programs in assembly language, it is necessary to know what processor registers exist and how they can be used. Most commands in assembly programs use registers as operands. Practically all commands represent transformations of data stored in processor registers, such as moving data between registers or between registers and memory, and performing mathematical or logical operations on data stored in registers.

Access to registers is done by names, rather than by addresses like with main memory. Each register in the x86 architecture has a name consisting of 2 or 3 Latin letters.

For example, here are the names of the main general-purpose registers (these registers are most frequently used in programming):

- RAX, RCX, RDX, RBX, RSI, RDI — 64-bit
- EAX, ECX, EDX, EBX, ESI, EDI — 32-bit
- AX, CX, DX, BX, SI, DI — 16-bit
- AH, AL, CH, CL, DH, DL, BH, BL — 8-bit

Another important node of a computer is Random Access Memory (RAM). RAM is a fast volatile memory that directly interacts with processor nodes, intended for storing programs and data with which the processor works at the current moment. RAM consists of uniform numbered memory cells. The number of a memory cell is the address of the data stored in it.

The peripheral devices of a computer include:

- External memory devices intended for long-term storage of large volumes of data.
- Input-output devices that ensure interaction between the CPU and the external environment.

The computational process of a computer is based on the principle of program control. This means that a computer solves a given task as a sequence of actions recorded in the form of a program.

Instruction codes represent multi-digit binary combinations of 0s and 1s. In the machine instruction code, two parts can be distinguished: operational and address. The operational part stores the code of the command that needs to be executed. The address part contains the data or addresses of the data involved in performing the operation.

When executing each command, the processor carries out a certain standard sequence of actions known as the processor's command cycle. This involves:

1. Forming the address in memory of the next command;

2. Reading the command code from memory and its decoding;
3. Executing the command;
4. Transitioning to the next command.

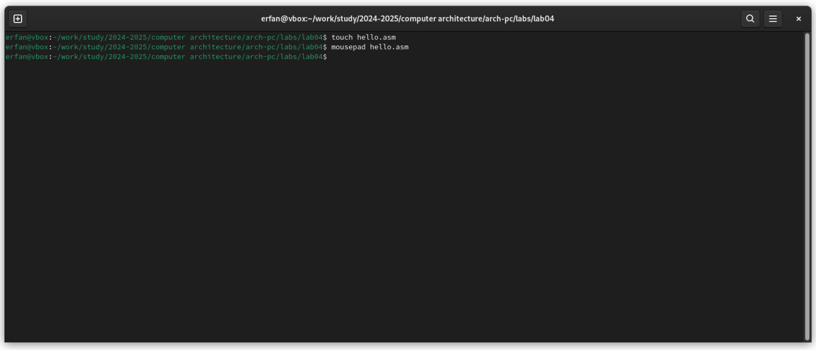
Assembly language (abbreviated as asm) is a low-level machine-oriented language.

NASM is an open assembler project with versions available for different operating systems, which enables the generation of object files for these systems. NASM uses Intel syntax and supports x86-64 instructions.

4 Conducting Laboratory Work

4.1 Hello World Program

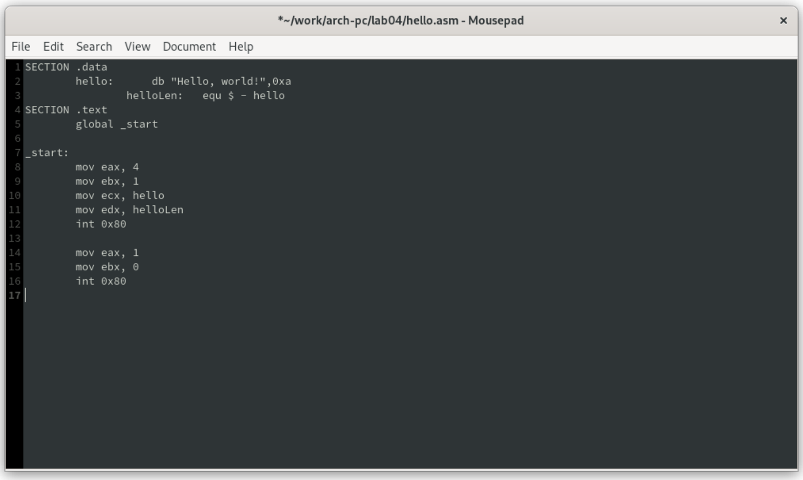
I create a file hello.asm in which I will write a program in assembly language. (Fig. -fig. 4.2)



```
erfan@vbox:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab04$ touch hello.asm
erfan@vbox:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab04$ mousepad hello.asm
erfan@vbox:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab04$
```

Fig. 4.1: Creating .asm file

Using the editor, I write the program in the created file. (Fig. -fig. 4.2)

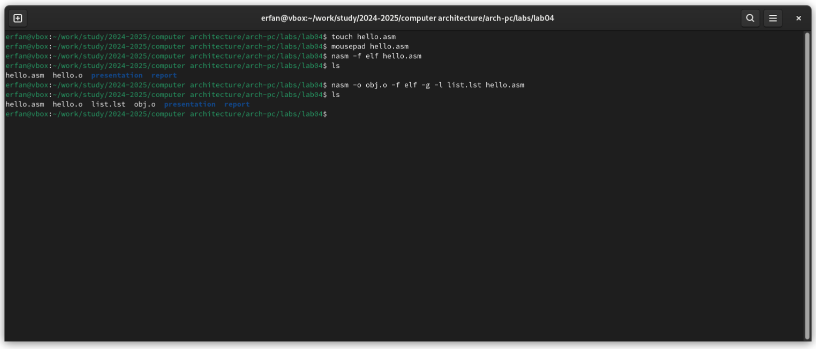

*~/work/arch-pc/lab04/hello.asm - Mousepad
File Edit Search View Document Help

```
SECTION .data
        hello:      db "Hello, world!",0xa
                helloLen:   equ $ - hello
SECTION .text
        global _start

_start:
        mov eax, 4
        mov ebx, 1
        mov ecx, hello
        mov edx, helloLen
        int 0x80

        mov eax, 1
        mov ebx, 0
        int 0x80
```

Fig. 4.2: Editing the file

4.2 NASM Translator

I compile my program using NASM. (Fig. -fig. 4.3)

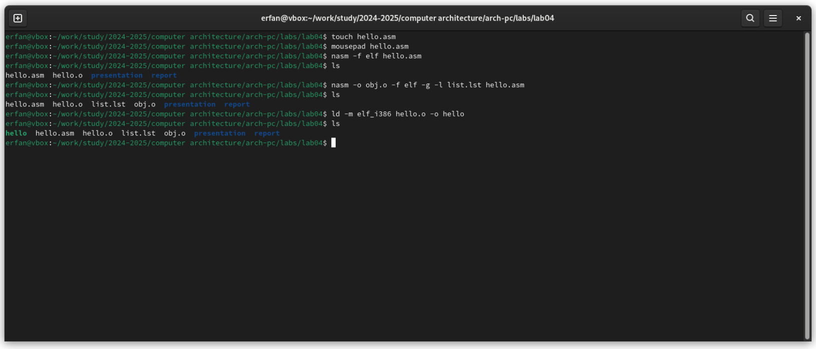


```
erfan@vbox:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab04$ touch hello.asm
erfan@vbox:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab04$ mousepad hello.asm
erfan@vbox:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab04$ nasm -f elf hello.asm
erfan@vbox:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab04$ ls
hello.asm  hello.o  presentation  report
erfan@vbox:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab04$ nasm -o obj.o -f elf -g -l list.lst hello.asm
erfan@vbox:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab04$ ls
hello.asm  hello.o  list.lst  obj.o  presentation  report
erfan@vbox:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab04$
```

Fig. 4.3: Compiling the program

4.3 Extended NASM Command Line Syntax

I execute the command shown in (Fig. -fig. 4.4), which compiles the source file hello.asm into obj.o; the .o extension indicates that the file is an object file. Additionally, the flags -g and -l prepare the debugging file and listing, respectively.

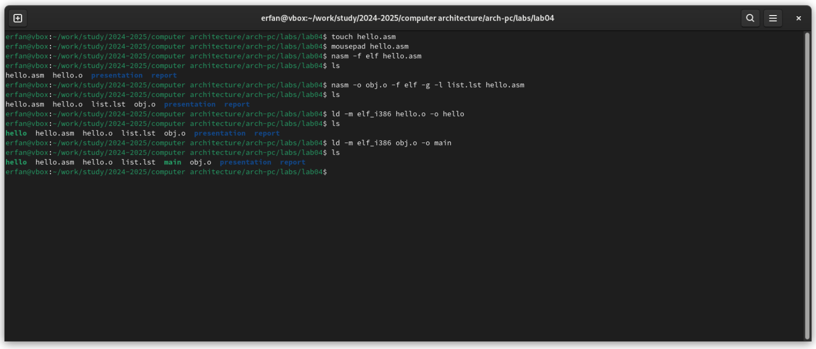


```
erfan@vbox:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab04
erfan@vbox:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab04$ touch hello.asm
erfan@vbox:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab04$ mousepad hello.asm
erfan@vbox:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab04$ nasm -f elf hello.asm
erfan@vbox:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab04$ ls
hello.asm  hello.o  presentation  report
erfan@vbox:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab04$ nasm -o obj.o -f elf -g -l list.lst hello.asm
erfan@vbox:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab04$ ls
hello.asm  hello.o  list.lst  obj.o  presentation  report
erfan@vbox:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab04$ ld -m elf_i386 hello.o -o hello
erfan@vbox:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab04$ ls
hello  hello.asm  hello.o  list.lst  obj.o  presentation  report
erfan@vbox:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab04$ 
```

Fig. 4.4: NASM syntax capabilities

4.4 LD Linker

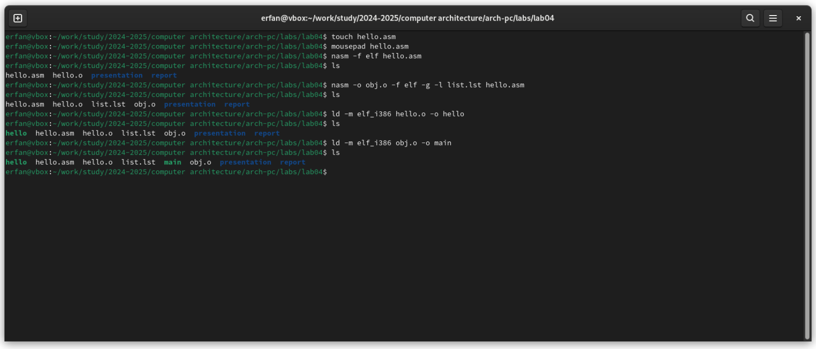
Then I need to transfer the object file to the linker, which I do using the ld command. (Fig. -fig. 4.5)



```
erfan@vbox:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab04
erfan@vbox:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab04$ touch hello.asm
erfan@vbox:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab04$ mousepad hello.asm
erfan@vbox:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab04$ nasm -f elf hello.asm
erfan@vbox:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab04$ ls
hello.asm  hello.o  presentation  report
erfan@vbox:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab04$ nasm -o obj.o -f elf -g -l list.lst hello.asm
erfan@vbox:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab04$ ls
hello.asm  hello.o  list.lst  obj.o  presentation  report
erfan@vbox:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab04$ ld -m elf_i386 hello.o -o hello
erfan@vbox:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab04$ ls
hello  hello.asm  hello.o  list.lst  obj.o  presentation  report
erfan@vbox:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab04$ ld -m elf_i386 obj.o -o main
erfan@vbox:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab04$ ls
hello  hello.asm  hello.o  list.lst  main  obj.o  presentation  report
erfan@vbox:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab04$
```

Fig. 4.5: Sending file to the linker

Next, I execute the following command …, the result of this command will be a created file main, compiled from the object file obj.o. (Fig. -fig. 4.6)

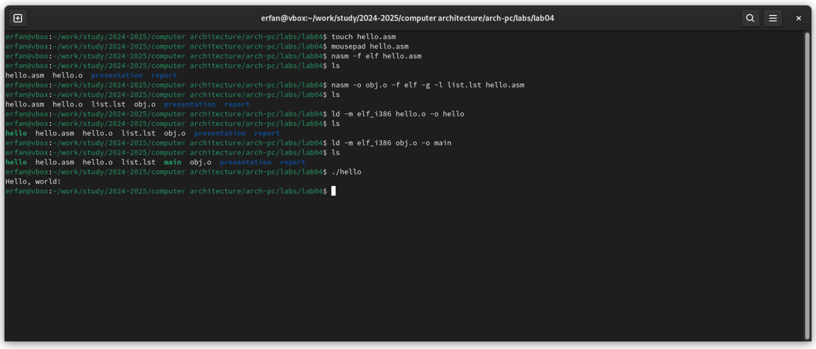


```
erfan@vbox:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab04
erfan@vbox:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab04$ touch hello.asm
erfan@vbox:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab04$ mousepad hello.asm
erfan@vbox:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab04$ nasm -f elf hello.asm
erfan@vbox:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab04$ ls
hello.asm  hello.o  presentation  report
erfan@vbox:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab04$ nasm -o obj.o -f elf -g -l list.lst hello.asm
erfan@vbox:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab04$ ls
hello.asm  hello.o  list.lst  obj.o  presentation  report
erfan@vbox:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab04$ ld -m elf_i386 hello.o -o hello
erfan@vbox:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab04$ ls
hello  hello.asm  hello.o  list.lst  obj.o  presentation  report
erfan@vbox:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab04$ ld -m elf_i386 obj.o -o main
erfan@vbox:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab04$ ls
hello  hello.asm  hello.o  list.lst  main  obj.o  presentation  report
erfan@vbox:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab04$
```

Fig. 4.6: Creating executable file

4.5 Running the Executable File

I run the executable file from the current directory. (Fig. -fig. 4.7)

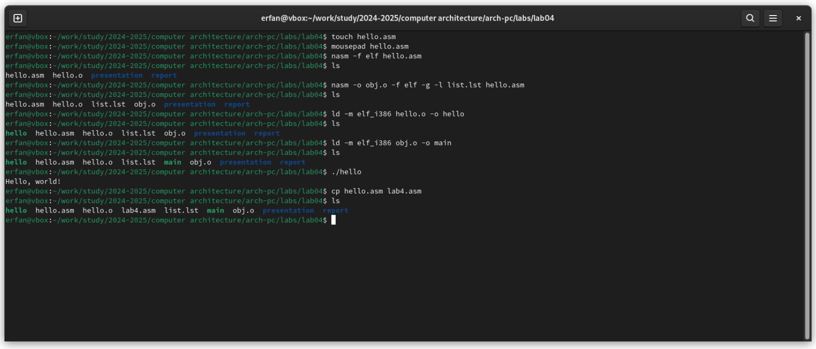


```
erfan@vbox:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab04
erfan@vbox:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab04$ touch hello.asm
erfan@vbox:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab04$ mousepad hello.asm
erfan@vbox:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab04$ nasm -f elf hello.asm
erfan@vbox:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab04$ ls
hello.asm  hello.o  presentation  report
erfan@vbox:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab04$ nasm -o obj.o -f elf -g -l list.lst hello.asm
erfan@vbox:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab04$ ls
hello.asm  hello.o  list.lst  obj.o  presentation  report
erfan@vbox:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab04$ ld -m elf_i386 hello.o -o hello
erfan@vbox:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab04$ ls
hello  hello.asm  hello.o  list.lst  obj.o  presentation  report
erfan@vbox:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab04$ ld -m elf_i386 obj.o -o main
erfan@vbox:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab04$ ls
hello  hello.asm  hello.o  list.lst  main  obj.o  presentation  report
erfan@vbox:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab04$ ./hello
Hello, world!
erfan@vbox:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab04$
```

Fig. 4.7: Running the program

4.6 Independent Work Assignments

I create a copy of the file for subsequent work. (Fig. -fig. 4.8)

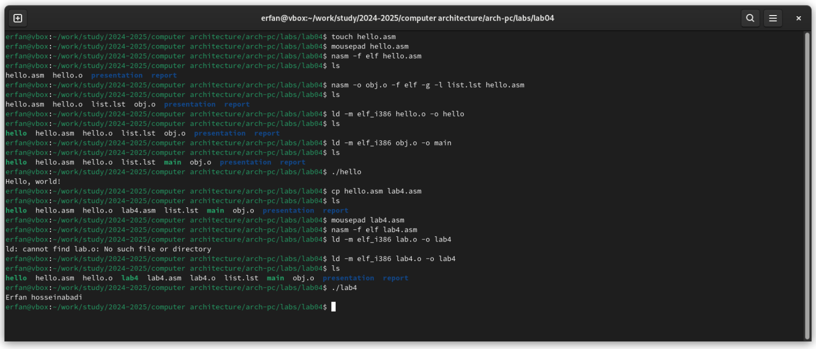


```
erfan@vbox:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab04
erfan@vbox:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab04$ touch hello.asm
erfan@vbox:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab04$ mousepad hello.asm
erfan@vbox:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab04$ nasm -f elf hello.asm
erfan@vbox:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab04$ ls
hello.asm  hello.o  presentation  report
erfan@vbox:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab04$ nasm -o obj.o -f elf -g -l list.lst hello.asm
erfan@vbox:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab04$ ls
hello.asm  hello.o  list.lst  obj.o  presentation  report
erfan@vbox:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab04$ ld -m elf_i386 hello.o -o hello
erfan@vbox:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab04$ ls
hello  hello.asm  hello.o  list.lst  obj.o  presentation  report
erfan@vbox:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab04$ ld -m elf_i386 obj.o -o main
erfan@vbox:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab04$ ls
hello  hello.asm  hello.o  list.lst  main  obj.o  presentation  report
erfan@vbox:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab04$ ./hello
Hello, world!
erfan@vbox:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab04$ cp hello.asm lab4.asm
erfan@vbox:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab04$ ls
hello  hello.asm  hello.o  lab4.asm  list.lst  main  obj.o  presentation  report
erfan@vbox:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab04$
```

Fig. 4.8: Creating a copy

I edit the copy of the file, replacing the text with my first and last name. (Fig. -fig. 4.9)

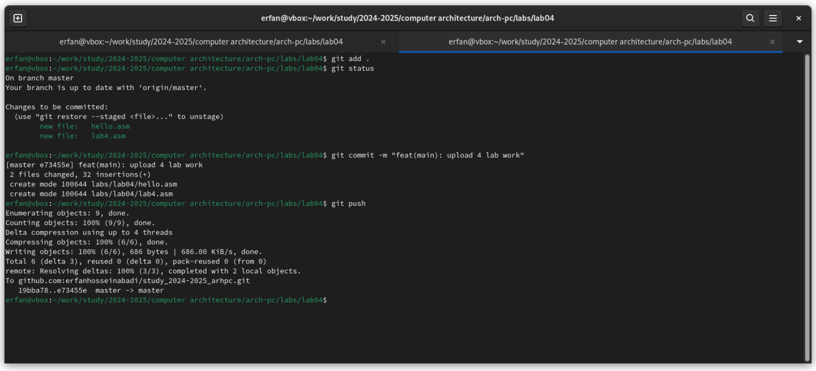
I translate the copy of the file into an object file, link it, and run it. (Fig. -fig. 4.9)



```
erfan@vbox:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab04
erfan@vbox:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab04$ touch hello.asm
erfan@vbox:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab04$ mousepad hello.asm
erfan@vbox:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab04$ nasm -f elf hello.asm
erfan@vbox:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab04$ ls
hello.asm  hello.o  presentation  report
erfan@vbox:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab04$ nasm -o obj.o -f elf -g -l list.lst hello.asm
erfan@vbox:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab04$ ls
hello.asm  hello.o  list.lst  obj.o  presentation  report
erfan@vbox:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab04$ ld -m elf_i386 hello.o -o hello
erfan@vbox:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab04$ ls
hello  hello.asm  hello.o  list.lst  obj.o  presentation  report
erfan@vbox:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab04$ ld -m elf_i386 obj.o -o main
erfan@vbox:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab04$ ls
hello  hello.asm  hello.o  list.lst  main  obj.o  presentation  report
erfan@vbox:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab04$ ./hello
Hello, world!
erfan@vbox:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab04$ cp hello.asm lab4.asm
erfan@vbox:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab04$ ls
hello  hello.asm  hello.o  lab4.asm  list.lst  main  obj.o  presentation  report
erfan@vbox:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab04$ mousepad lab4.asm
erfan@vbox:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab04$ nasm -f elf lab4.asm
erfan@vbox:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab04$ ld -m elf_i386 lab.o -o lab4
ld: cannot find lab.o: No such file or directory
erfan@vbox:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab04$ ld -m elf_i386 lab4.o -o lab4
erfan@vbox:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab04$ ls
hello  hello.asm  hello.o  lab4  lab4.asm  lab4.o  list.lst  main  obj.o  presentation  report
erfan@vbox:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab04$ ./lab4
Erfan hosseinabadi
erfan@vbox:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab04$
```

Fig. 4.9: Checking the functionality of the compiled program

Having ensured the program works correctly, I copy the working files into my local repository. (Fig. -fig. 4.10)



```
erfan@vbox:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab04
erfan@vbox:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab04    erfan@vbox:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab04
erfan@vbox:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab04$ git add .
erfan@vbox:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab04$ git status
On branch master
Your branch is up to date with 'origin/master'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   hello.asm
        new file:   lab4.asm

erfan@vbox:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab04$ git commit -m "feat(main): upload 4 lab work"
[master e73455e] feat(main): upload 4 lab work
 2 files changed, 32 insertions(+)
 create mode 100644 labs/lab04/hello.asm
 create mode 100644 labs/lab04/lab4.asm
erfan@vbox:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab04$ git push
Enumerating objects: 9, done.
Counting objects: 100% (9/9), done.
Delta compression using up to 4 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (6/6), 686 bytes | 686.00 KiB/s, done.
Total 6 (delta 3), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (3/3), completed with 2 local objects.
To github.com:erfanhosseinabadi/study_2024-2025_arhpc.git
   19bba78..e73455e  master -> master
erfan@vbox:~/work/study/2024-2025/computer architecture/arch-pc/labs/lab04$
```

Fig. 4.10: Sending files to local repository

Uploading changes to my remote repository on GitHub.

5 Conclusions

In carrying out this laboratory work, I learned the procedures for compiling and building programs written in the NASM assembly language.

6 References

1. Example of laboratory work execution
2. Course at RUDN University
3. Laboratory Work No. 4
4. Programming in NASM Assembly Language by A.V. Stolyarov