**Ministerul Educaţiei și Cercetării al Republicii Moldova Universitatea Tehnică a Moldovei**

**Facultatea Calculatoare, Informatică și Microelectronică**

COMPUTER ARCHITECTURE

Laboratory work 4:

Introduction in Assembly Language

Elaborated:

st. gr. FAF-213 Konjevic Alexandra

Verified:

asist. Univ Vladislav Voitcovschi

Chişinău – 2023

**Objective:**

Getting familiar with the basic concepts of Assembly Language – how it works, its syntax; understanding concepts like registers, memory, instructions etc.

**Introduction:**

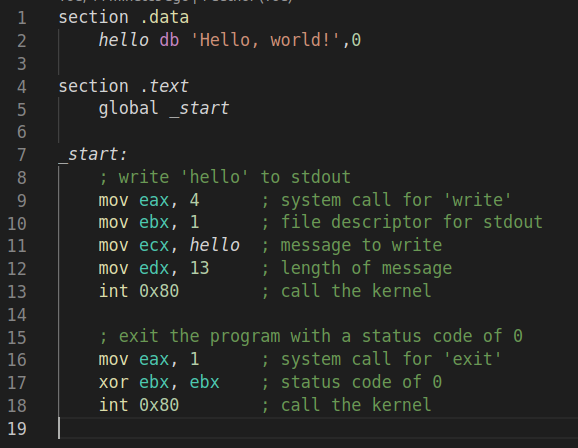
Assembly language is a low-level programming language that allows direct control over a computer's hardware. It is a vital tool in computer engineering and is widely used for developing software in various applications. In this report, we explore the fundamentals of assembly language programming, its syntax, and its application in computer systems. Specifically, this report presents beginner programs in assembly language, with the goal of providing a comprehensive understanding of the language. The laboratory work accompanying this report involves practical exercises aimed at reinforcing the theoretical concepts discussed.

**Tasks:**

1. Familiarize yourself with the basics of Assembly Language: Before diving into NASM, it's important to understand the fundamentals of Assembly Language. Start by learning the basic concepts such as registers, memory, instructions, and the syntax used to write Assembly code.
2. Install NASM: The first step towards learning NASM is to install it on your machine. NASM is available for multiple platforms like Windows, Linux, and macOS. Download and install the version that is compatible with your system.

*NASM installed*

1. Write simple programs: Start by writing simple programs in NASM to get a feel for the language. Start with basic programs like printing messages on the screen, reading input from the user, and performing arithmetic operations. This will help you understand how NASM works and get comfortable with the syntax.
2. Hello world program

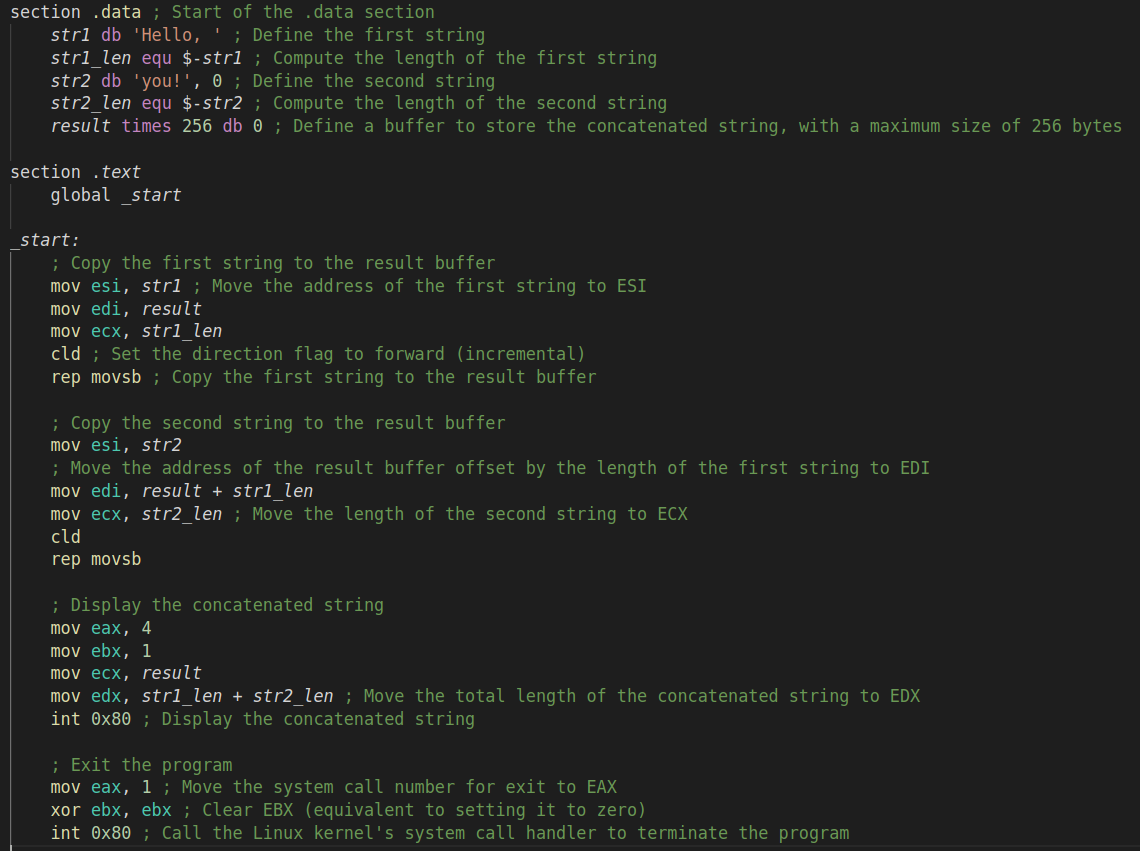


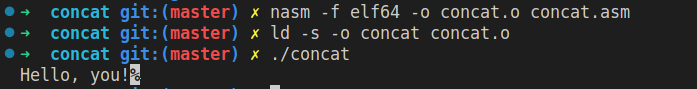
*Hello world program in Assembly*



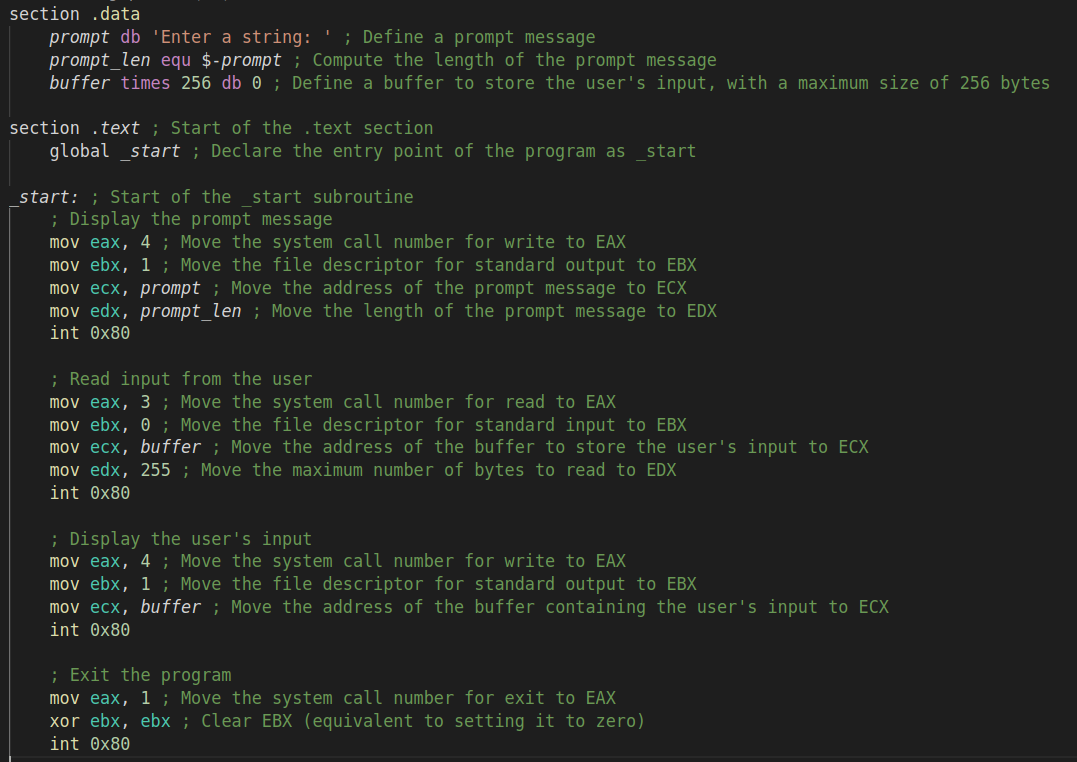
*Hello world program output*

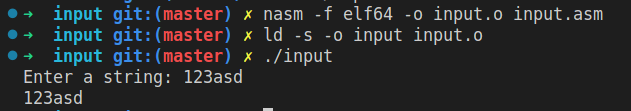
1. Concatenate strings program

*Concatenate characters program*

*Concatenate program output*

1. Read input from user

*Input program output*

*Input program output*

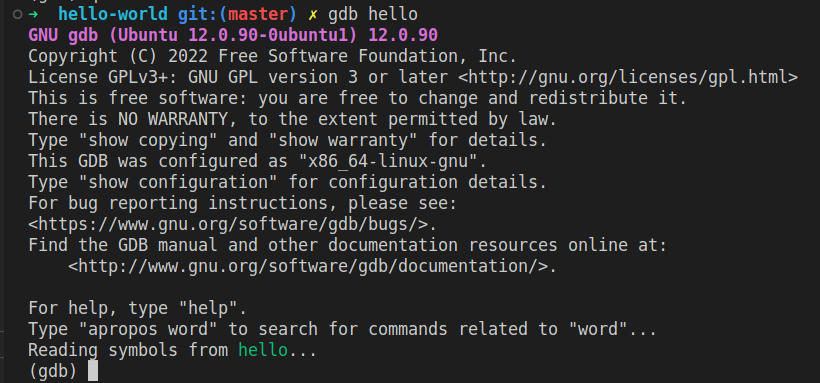
1. Debug your programs: Debugging is an essential part of programming, and NASM is no exception. Learn how to use debugging tools like GDB to identify and fix errors in your code. This will help you become more efficient in your programming and also give you a better understanding of how your code works:

The ability to effectively use a debugger is a crucial skill for any software developer. NASM supports several debuggers, such as GDB and DDD. To get familiar with debugging, I used GDB.

To use GDB to debug NASM programs, I followed the steps below (I debugged the hello-world program):

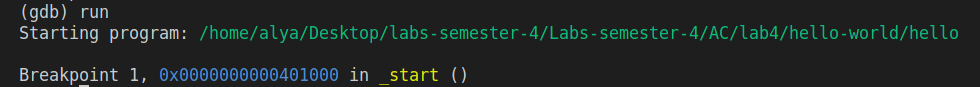
- Compiled the program with the -g flag, to include debugging information in the object file.

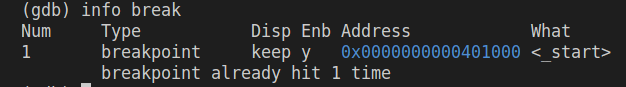
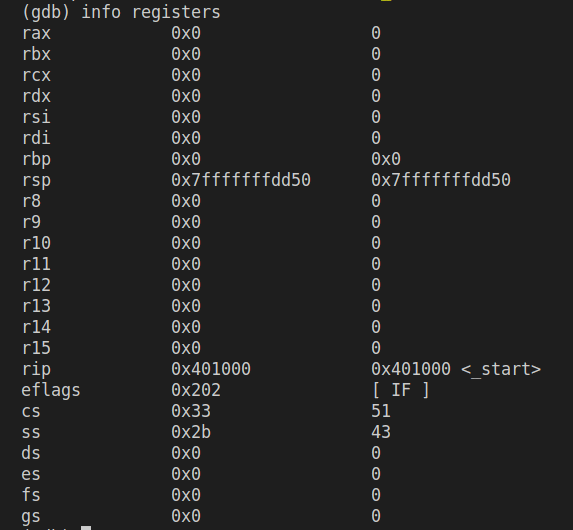
- Started the GDB by running the command: `gdb hello`

- I set a breakpoint at the begging of the program:



- I ran the program:



**Conclusion:**

In conclusion, this report has provided an overview of assembly language programming, including its syntax, structure, and application in computer systems. Through practical exercises, I have gained hands-on experience in writing programs using assembly language, providing a foundation for further exploration in this field.

By working with assembly language, I have gained insight into the underlying operations of a computer system, and the role of low-level programming in controlling hardware. This knowledge is critical in the development of software and applications for various domains, including embedded systems, operating systems, and game development.

Assembly language programming requires a thorough understanding of computer architecture and hardware, as well as a keen attention to detail. However, with practice and dedication, it is a powerful tool for developers to optimize performance and implement functionality that may not be possible using higher-level languages.

In conclusion, this laboratory work has provided a solid foundation for further exploration of assembly language programming.