Technical University of Moldova Faculty of Computers, Informatics and Microelectronics

Software Engineering and Automation Department

**REPORT**

Laboratory work N.4

Discipline: Computer architecture

Theme: Assembly Language

Done by: st. Sirghi Tudor

gr. FAF-213

Verified by : C. Fiștic   
 V.Voitcovschi

Chișinău 2023

**Basics of Assembly Language:**

NASM (Netwide Assembler) is a popular low-level programming language used for developing software applications at the system level. It is designed to work on a variety of operating systems and architectures, including Windows, Linux, and macOS. NASM is known for its flexibility, high performance, and the ability to generate executable files with minimal overhead. It supports a wide range of assembly syntaxes, including Intel and AT&T, making it a versatile choice for both beginners and experienced developers. With NASM, you can write code that directly interacts with the hardware, making it a powerful tool for system-level programming tasks such as developing device drivers or operating system kernels.

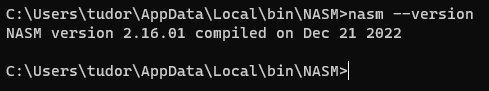
**Control Registers:**

In NASM (Netwide Assembler), a control register is a special register used for controlling specific aspects of the processor's behavior. Control registers are typically used in system-level programming tasks, such as developing operating system kernels or device drivers. There are several control registers in modern x86 processors, including the following:

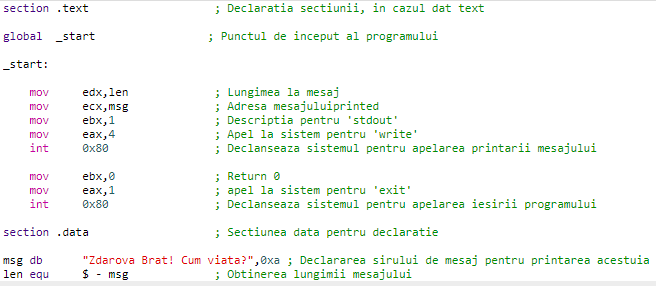
1. CR0 (Control Register 0): This register controls various processor operations, including memory protection, paging, and system management mode (SMM).
2. CR2 (Control Register 2): This register stores the linear address of the last page fault that occurred in the system.
3. CR3 (Control Register 3): This register holds the physical address of the top-level page table used by the processor's memory management unit (MMU).
4. CR4 (Control Register 4): This register controls various processor features, including virtual-8086 mode, protection keys, and machine check exceptions.

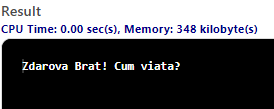
To work with control registers in NASM, you typically use the MOV instruction to move values to and from the control registers. However, accessing control registers can be complex, and it is important to have a solid understanding of the x86 architecture and the specific control registers you are working with before attempting to modify them in your code.

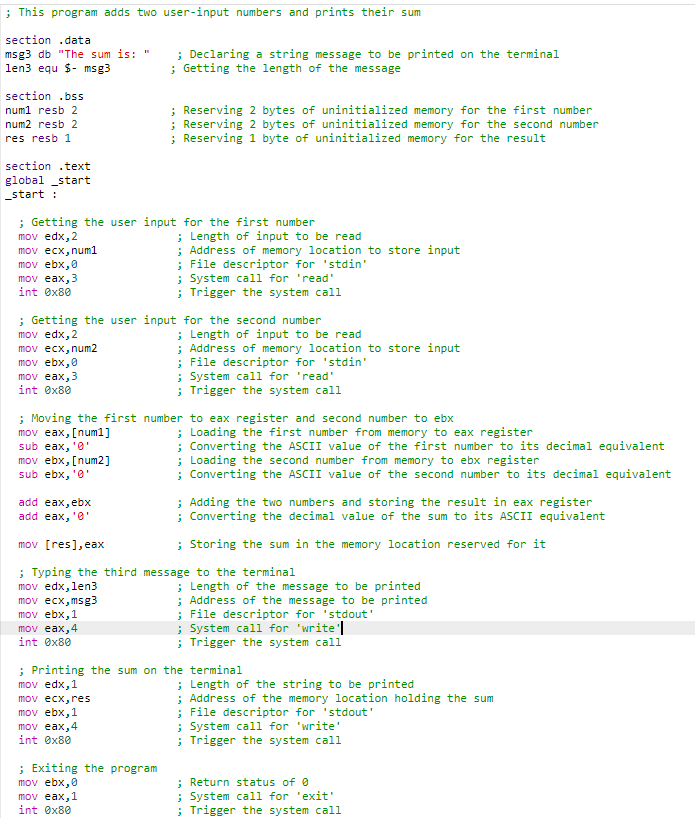
**NASM installation :**

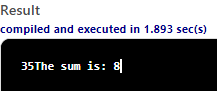
****

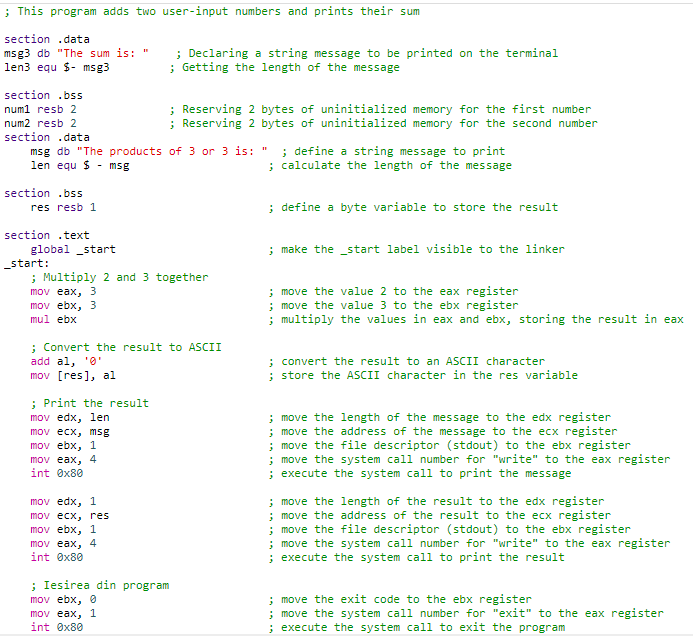
**Exercises:**













**Debugging:**

Debugging is an essential part of any programming task, including assembly language programming with NASM (Netwide Assembler). Fortunately, there are several tools and techniques you can use to debug your code and find and fix errors.

Here are some tips for debugging code in NASM:

1. Use a debugger: Debuggers are powerful tools that allow you to step through your code, set breakpoints, and examine the contents of registers and memory locations. Some popular debuggers for assembly language programming include GDB, OllyDbg, and WinDbg.
2. Use printf-style debugging: You can use the PRINTF macro in NASM to print out values of variables or register contents during runtime. This can help you identify where the problem is occurring and what the values of variables are at that point.
3. Check for syntax errors: Syntax errors are a common cause of bugs in assembly language programming. Make sure that your code is free from typos, misspelled labels or instructions, and other syntax errors.
4. Check for logical errors: Logical errors are more difficult to find and can be caused by incorrect algorithms or incorrect use of instructions. It's a good idea to review your code and make sure that you are using the correct instructions and algorithms.
5. Use a disassembler: A disassembler can help you examine the machine code generated by your assembly language code. This can be especially helpful when trying to understand what your code is doing and why it might not be working as expected.

In summary, debugging assembly language code in NASM can be challenging, but with the right tools and techniques, you can identify and fix errors in your code. Be patient, review your code carefully, and use the available tools to help you.

**Conclusion:**

In conclusion, NASM (Netwide Assembler) is a powerful and flexible assembly language programming tool that is widely used for system-level programming tasks such as developing device drivers, operating system kernels, and other low-level applications. As an assembly language programming tool, NASM provides a direct interface to the hardware, allowing for precise control over the system and high performance.

Assembly language programming with NASM requires a good understanding of the x86 architecture and the specific control registers and instructions used in system-level programming tasks. Debugging assembly language code can be challenging, but with the right tools and techniques, it is possible to identify and fix errors in your code.

In my opinion, I can say that assembly language programming with NASM is a useful and powerful tool for developers working at the system level. However, it is important to note that assembly language programming can be complex and time-consuming, and it may not be the best choice for every programming task. Ultimately, the decision to use NASM or any other assembly language programming tool depends on the specific requirements of the project and the experience and expertise of the developer.