|  |  |
| --- | --- |
|  | **Namal University Mianwali**  **Department of Computer Science** |

**Lab Manual**

|  |  |  |  |
| --- | --- | --- | --- |
| Course | CSC-241-L Computer Organization and Assembly Language | | |
| Instructor | Ramzan Shahid Khan | Session / Semester | 2024-2028 (3rd) |

|  |  |
| --- | --- |
| Lecture # | 01 |
| Topic | Introduction to MASM, Lab Setup and Basic Assembly Commands |
| Objective | The objective of this first lab is to introduce students to Microsoft Assembler (MASM), guide them through the installation and setup process, and familiarize them with basic assembly commands. |

**What is MASM?**

MASM (Microsoft Macro Assembler) is a widely used assembler for the x86 family of microprocessors. It enables developers to write programs in assembly language, which directly interacts with the hardware of the computer. MASM is especially popular for its compatibility with Intel processors and its support for both high-level constructs and low-level hardware operations.

**Why Use Assembly Language?**

Assembly language offers direct control over hardware, enabling programmers to optimize their code for speed and memory usage. It is often used in systems programming, real-time applications, and situations where performance is critical.

**MASM Setup and Installation**

1. **Download MASM:** You can download MASM from Microsoft's official website or use the MASM32 SDK, a free version that contains all tools needed to write and compile assembly programs. [Download Link: <https://www.masm32.com/>]
2. **Installation:** After downloading, follow the installation wizard to install MASM. Ensure that the path to the assembler (ml.exe) is added to your system's environment variables to compile programs from the command line.
3. **Text Editor (optional): Install the Visual Studio code from given link and install two extensions 1). MASM and 2). x86 and x86\_64 Assembly. [Download Link:** <https://code.visualstudio.com/download>**]**
4. **Verifying Installation:** Open the command prompt and type ml to verify if MASM is correctly installed. If installed correctly, you should see MASM's version details.

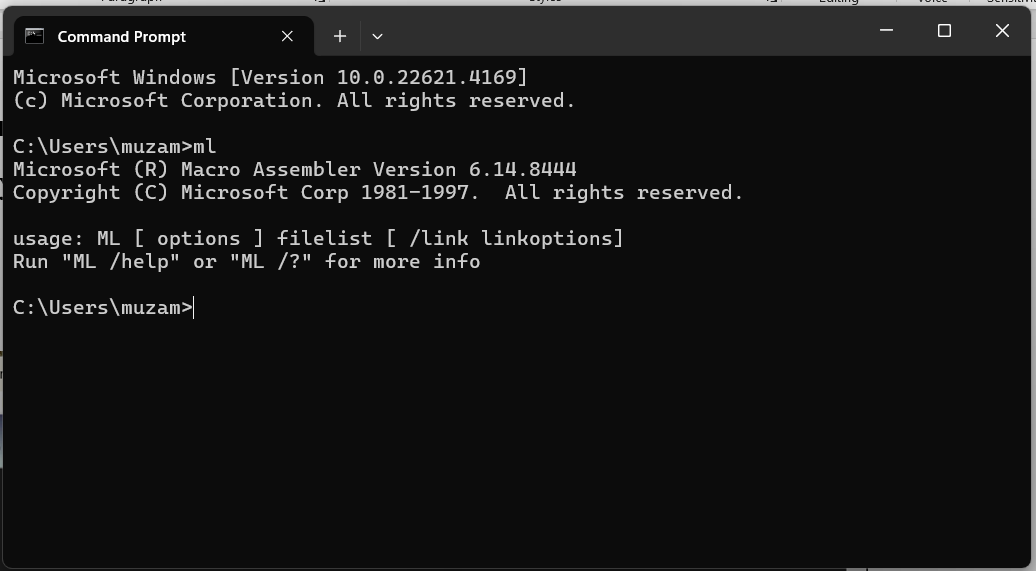


Figure . Verifying MASM 32 Installation

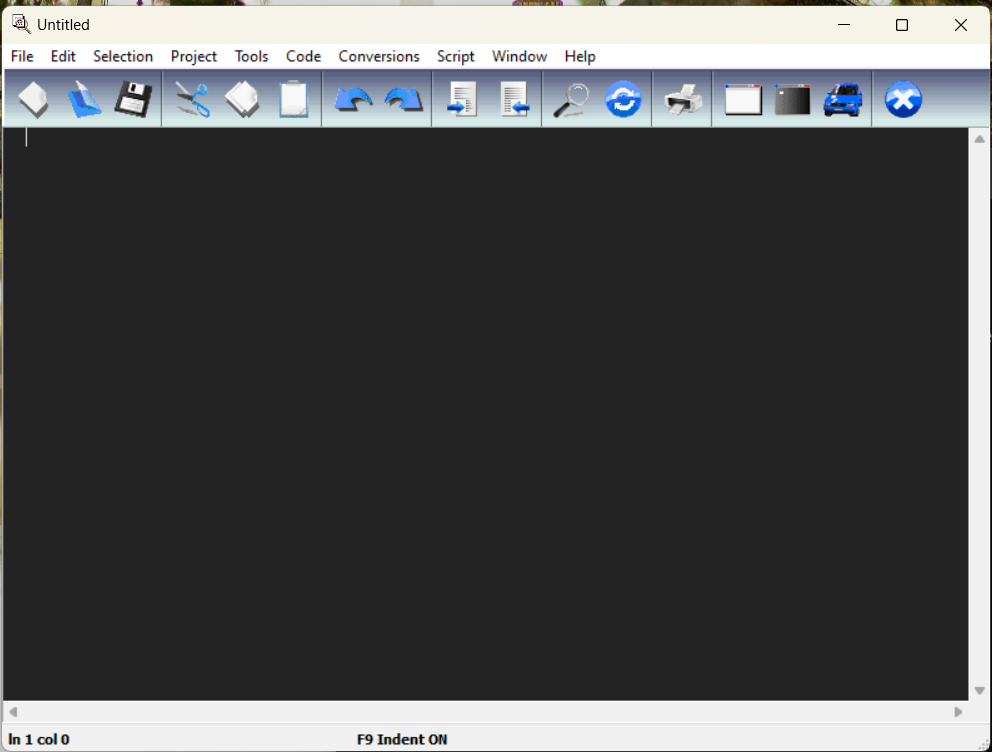


Figure . Interface of MASM32 Editor

**Assemble-Link-Execute cycle**

**Figure 3 shows the** **Assemble-Link-Execute cycle to process of editing, assembling, linking, and executing assembly language programs. Following is a detailed description of each step.**

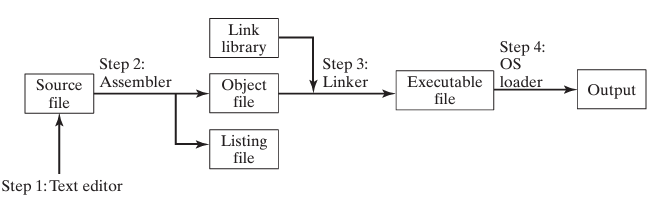
****

Figure . The Assemble-Link-Execute Cycle

**Step 1: A programmer uses a text editor to create an ASCII text file named the source file.**

**Step 2: The assembler reads the source file and produces an object file, a machine-language translation of the program. Optionally, it produces a listing file. If any errors occur, the programmer must return to Step 1 and fix the program.**

**Step 3: The linker reads the object file and checks to see if the program contains any calls to procedures in a link library. The linker copies any required procedures from the link library, combines them with the object file, and produces the executable file.**

**Step 4: The operating system loader utility reads the executable file into memory and branches the CPU to the program’s starting address, and the program begins to execute.**

**How to use MASM for Hello World! Program**

**Editing** **the Program**

**You write the assembly code in a text editor and save it with an appropriate file extension (usually .asm). The assembly code contains instructions, data definitions, and possibly directives that the assembler will translate into machine code.**

**Example of a simple hello\_world.asm program for x86 architecture:**

|  |
| --- |
| **; Include the necessary MASM32 runtime library**  **include \masm32\include\masm32rt.inc**  **.data**  **; Data section (currently empty)**  **.code**  **; Code section begins**  **start:**  **call main ; Call the main procedure**  **inkey ; Wait for a key press before exiting**  **exit ; Exit the program**  **main proc**  **cls ; Clear the screen**  **print "Hello World", 13, 10 ; Print "Hello World" followed by a newline**  **ret ; Return from the procedure**  **main endp ; End of the main procedure**  **end start ; Marks the program entry point** |

**Assembling the Program**

The assembler converts the human-readable assembly code (.asm) into **machine code** (binary format) and produces an object file (typically .obj). The object file contains the translated machine code instructions. Command to assemble using MASM (Microsoft Macro Assembler):

ml /c /Zd /coff "hello\_world.asm"

* ml assembles and links assembly language code into an object file or executable.an object file (.obj), not an executable file.
* /c switch tells MASM to assemble the code only, without linking. The output will be an object file (.obj), not an executable file.
* /Zd generates debugging information in the object file. The debugging information helps in debugging the program with a debugger, as it includes source file references (line numbers, variables, etc.).
* This is useful for tracking down issues in your assembly code during development.
* /coff tells MASM to generate the object file in COFF (Common Object File Format), which is a standard format used for object files on Windows.
* COFF is compatible with modern linkers and debuggers in Windows environments.

**Output**: An object file (e.g., hello\_world.obj).

**Linking the Program**

The linker takes the object file (hello\_world.obj) and combines it with system libraries or other object files to produce an **executable** file. This step resolves any external references (e.g., library calls) and prepares the final binary. Command to link using MASM's linker:

Link /SUBSYSTEM:CONSOLE "hello\_world.obj"

* /SUBSYSTEM:CONSOLE: Specifies the program will run as a console application.

**Output**: An executable file (e.g., hello\_world.exe).

**Executing the Program**

The final step is to run the assembled and linked program (the .exe file). In the case of DOS or Windows console applications, you can run the executable from the command prompt.

Hello world.exe

If the program is correct, it will execute and display the output (e.g., "Hello, World!" in the console).

**Assembly Program to add two Registers**

|  |
| --- |
| include \masm32\include\masm32rt.inc  .data  sumMessage db "The sum is: ", 0 ; String message to display before the sum  buffer db 11 dup(0) ; Buffer to hold the string representation of the sum (up to 10 digits plus null terminator)  .code  start:  ; Initialize registers with values  mov eax, 30 ; Load 30 into register EAX  mov ebx, 20 ; Load 20 into register EBX  add ecx, eax ; Add the value in EAX (30) to ECX (ECX = 30)  add ecx, ebx ; Add the value in EBX (20) to ECX (ECX = 30 + 20 = 50)  ; Convert the value in ECX (the sum) to a string and store it in the buffer  invoke dwtoa, ecx, addr buffer  ; Output the sumMessage ("The sum is: ") to the console  invoke StdOut, addr sumMessage  ; Output the string in buffer (the sum as text) to the console  invoke StdOut, addr buffer  ; Exit the program with a status code of 0  invoke ExitProcess, 0  end start |

**Lab Tasks**

**Task 1:** Write the assembly language program given in the Lab handout, assemble it and run

it as per the instructions in the handout. CLO-1

**Task 2:** Note down the contents of registers EAX, EBX and ECX as displayed by the program. CLO-1

**Task 3:** Do the contents of register ECX match the expected result? If not, what step needs to be taken? CLO-1

**Task 4:** Modify the source code to get the right result in the register ECX, re-assemble, and re-run the program. CLO-1

**Task 5:** Verify that the contents of the ECX register are now correct. CLO-1

**Submission Guidelines**

1. Create a folder named with your name + registration number with two subdirectories:
   * **Lab Task 1 Code**
   * **Lab Task 1 Report**
2. The **Code** subdirectory must include the following files:
   * program\_name.asm (Assembly source code)
   * program\_name.obj (Object file)
   * program\_name.exe (Executable file)
3. The **Report** subdirectory should contain a detailed explanation of each task, including relevant assembly code snippets with proper comments.
4. **Academic Integrity**: Plagiarism is strictly prohibited, including the use of AI tools such as ChatGPT. Any instances of plagiarism will result in a zero score for the last two lab submissions.
5. Submit your work through the **QOBE portal** in a ZIP file format, along with a hard copy of the report.
6. Adhere to the submission deadlines. Late submissions will receive zero marks.

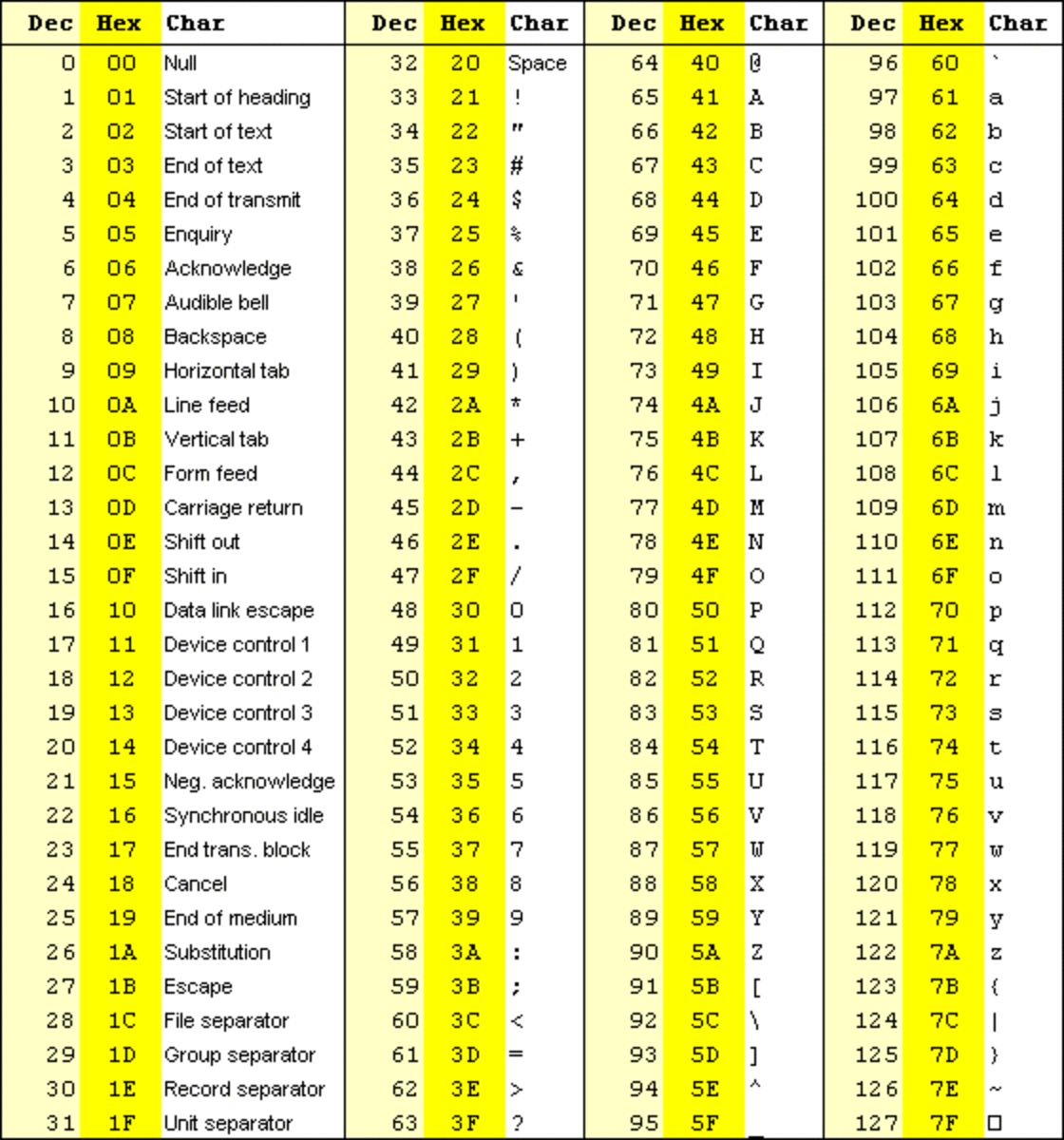
****

Figure . Standard ASCII Codes