**NUST SCHOOL OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE**

|  |  |
| --- | --- |
| Faculty Member:\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Date: \_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| Semester:\_\_\_\_\_\_\_\_\_\_\_\_\_ | Section: \_\_\_\_\_\_\_\_\_\_\_\_\_\_ |

Department of Electrical Engineering

EE- 222: Microprocessor Based Systems

**LAB 01: Programming in Assembly Language using MASM**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Student name | Reg. No. | Lab Report Marks / 10 | Viva Marks / 5 | Total/15 |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

**LAB 01: Programming in Assembly Language using MASM**

**Objective:** The aim of the first lab is familiarization with MASM and writing and testing our first program in Assembly Language.

**INTRODUCTION TO MASM**

The **Microsoft Macro Assembler** (**MASM**) is an x86 assembler for Microsoft Windows that uses the Intel syntax. Assembly language is a great tool to understand how a computer works and with the help of MASM you will be able to assemble and run your programs written in Assembly language.

**Writing Assembly Language Programs:**

You can write Assembly language programs in any text editor e.g. Notepad etc. However, you have to make sure that you save your programs with an extension of asm i.e. if you name your file example then it should be saved by going to saveAs and then typing **example.asm** in the file name and selecting All Files in the file Type.

Once you have installed MASM on your PC and written your program then you have to assemble and link your programs before they can be executed.

**Assemble – Link – Execute Cycle:**

The process of editing, assembling, linking, and executing assembly language programs is summarized in Figure below. Following is a detailed description of each step.

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

***Step 2:*** The **assembler** (file ML.exe)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** (file Link32.exe) 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*. Optionally, the linker can produce a *map 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.

****

At the end the useful files generated are as follows:

Example.obj

Example.lst

Example.exe

Example.exe is the executable file that can now be run by typing example on the DOS prompt and pressing enter.

**How to use MASM**

**Step1**

Open Notepad copy example code given below to Notepad and Save As the fie with *.asm* extension .Make sure you saved the file in the directory “C:\masm615” which is directory in which masm is installed.

**Example Program:**

TITLE Add two registers (example.asm)

; The comments are given after the semi colon on a line

; This program adds 32-bit unsigned

; integers and stores the sum in the ecx register

Include irvine32.inc

.data

;variable declarations go here

.code

Main Proc

;instructions go here

Mov eax, 30 ;Assembly Language is NOT case sensitive

Mov ebx, 20

Add edx, eax

Add edx, ebx

Call dumpregs ;displays the result on the screen by displaying all register values

Exit

Main endp

End main

**Step2**

Open the command prompt by typing *cmd* in Run and change your current path to “C:\masm615”

By typing following commands in command prompt

*cd c:\*

*cd masm615*

or

*cd c:\masm615*

**Step3**

Use make32.bat file to assembling and linking by typing following in command prompt

*make32 example*

This will create following files

Example.obj

Example.lst

Example.exe

*Note: make32.bat is batch file containing list of commands for assembling, linking and setting the paths.*

**Step 4**

Run the exe file by typing following in command prompt

Example.exe

This will show the output of your code.

**CPU REGISTERS:**

Registers are special memory locations on the CPU. One important difference between older and later processors is that the pre-386 processors are 16-bit instead of 32-bit. There are 8 32-bit general purpose registers. The first 4, eax, ebx, ecx, and edx can also be accessed using 16 or 8-bit names. ax gets the first 16 bits of eax, al gets the first 8 bits, and ah gets bits 9-16. The fig below shows all the general purpose and special purpose registers and their sizes.

****

****

**Exercise 1:** Write the assembly language program given in the Lab handout, assemble it and run it as per the instructions in the handout.

**Exercise 2:** Note down the contents of registers EAX, EBX and EDX as displayed by the program:

**Exercise 3:** Check the contents of registers against the instructions.

**Exercise 4:** Do the contents of register EDX match the expected result?

**Exercise 5:** If not, what step needs to be taken?

**Exercise 6:** Modify the source code to get the right result in the register EDX, re-assemble, and re-run the program.

**Exercise 7:** Verify that the contents of the EDX register are now correct.