**Department of Electrical Engineering**

|  |  |
| --- | --- |
| **Faculty Member: Ma'am Quratulain** | **Dated: October 15, 2020** |
|  |  |
| **Course/Section: BSCS-9B** | **Semester: 3rd** |
|  |  |

**Computer Organization and**

**Assembly Language (CS235)**

**Lab #1 Programming in Assembly Language using MASM**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | | **PLO4** | | **PLO5** | **PLO8** | **PLO9** |  |
| **Name** | **Roll number** | **Viva /Quiz/ Lab performance**  **5 marks** | **Analysis of data in lab report**  **5 marks** | **Modern tool Usage**  **5 marks** | **Ethics and Safety**  **5 marks** | **Individual and team work**  **5 marks** | **Total**  **25 marks** |
| **Fatima Seemab** | **291310** |  |  |  |  |  |  |
| **Mahum Samar** | **290647** |  |  |  |  |  |  |
| **Maryam Fatima** | **290479** |  |  |  |  |  |  |

**LAB No.1**

**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:**

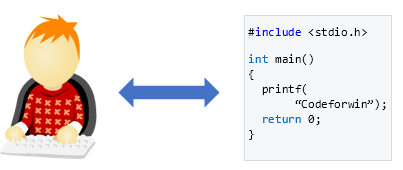
You can write Assembly language programs in any text editor e.g. Notepad etc. However, you have to make sure that you save your programs with an extension of asm i.e. if you name your file example then it should be saved by going to saveAs and then typing **example.asm** in the file name and selecting 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 summarizedin Figure below. Following is a detailed description of each step.

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

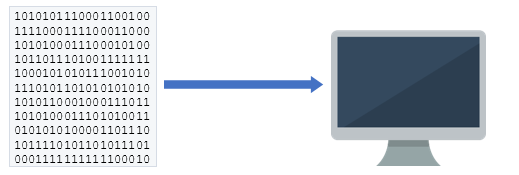


***Step 2:*** The **assembler** (file ML.exe)reads the source file and produces an *object file,* a machine-language translation of the program. Object code is a sequence of computer instructions in an intermediate language. Optionally, it produces a *listing file*. If any errors occur, the programmer must returnto 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 .

**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*

**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. Oneimportant difference between older and later processors is that the pre-386 processorsare 16-bit instead of 32-bit.There are 8 32-bit general purpose registers. The first 4, eax, ebx, ecx, and edx can alsobe accessed using 16 or 8-bit names. ax gets the first 16 bits of eax, al gets the first 8bits, and ah gets bits 9-16. The fig below shows all the general purpose and special purpose registers and their sizes.

****

****

**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 ecx, eax

Add ecx, ebx

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

Exit

Main endp

**Exercise 1:** Write the assembly language program given in the Lab handout, assemble it and run it as per the instructions in the handout. (3 marks lab report)

TITLE Add two registers

; 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

Mov ebx, 20

Add ecx, eax

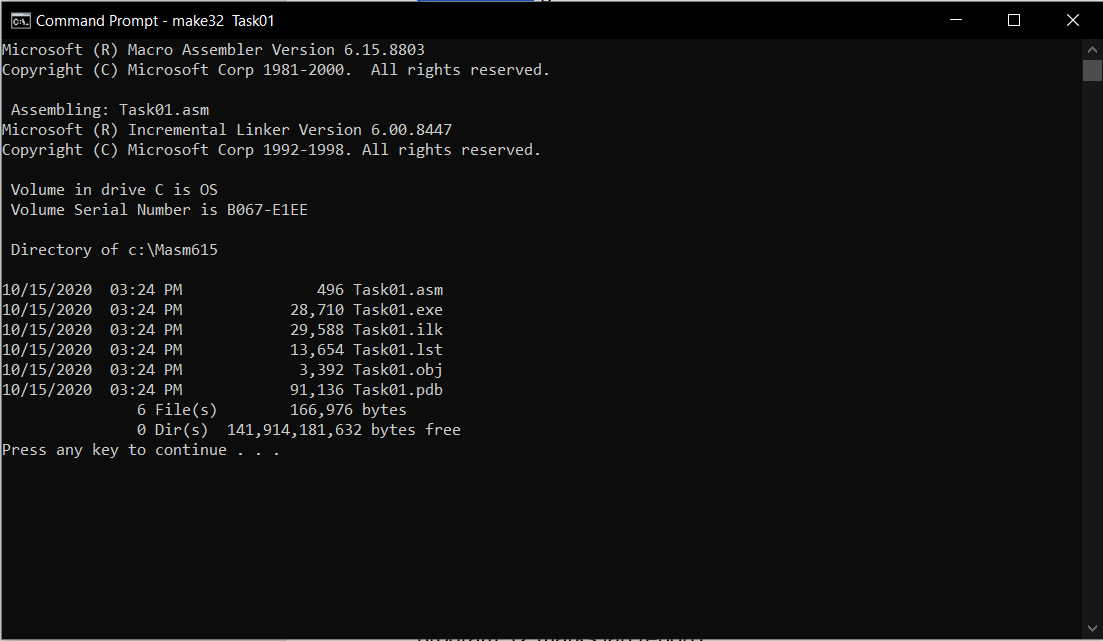
Add ecx, ebx

Call dumpregs ; displays the all register values on screen

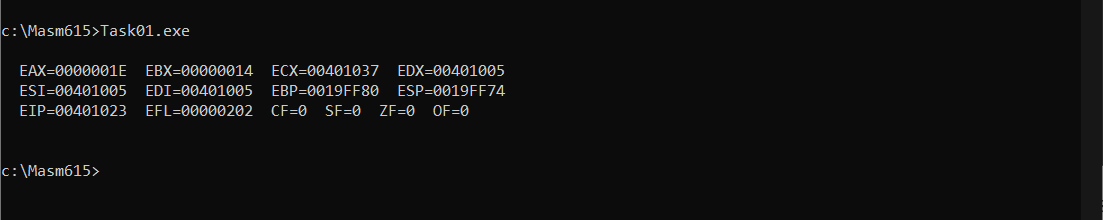
Exit

Main endp

end Main



**Exercise 2:** Note down the contents of registers EAX, EBX and ECX as displayed by the program: (2 marks lab report)



**Exercise 3**: Check the contents of registers against the instructions. (3 marks Modern tool)

* **Mov eax, 30**

This instruction moves the value 30 in the EAX register.

* **Mov ebx, 20**

This instruction moves the value 20 in the EBX register.

* **Add ecx, eax**

This instruction adds the value of ECX and EAX, and stores it in the ECX register.

* **Add ecx, ebx**

This instruction adds the value of ECX and EBX, and stores it in the ECX register.

* **Call dumpregs**

This instruction displays all register values on the screen.

**Exercise 4:** Do the contents of register ECX match the expected result? If not, what step needs to be taken? (1marks Modern tool)

No, the contents of the ECX doesn’t match the expected result. It is due to the garbage value in the ECX register. So ECX register should be initialized with zero before the calculations.

**Exercise 5:** Modify the source code to get the right result in the register ECX, re-assemble, and re-run the program. (3 marks Team work)

TITLE Add two registers

; 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 ecx,0

Mov eax, 30

Mov ebx, 20

Add ecx, eax

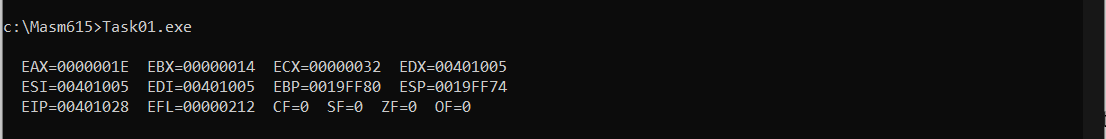
Add ecx, ebx

Call dumpregs ; displays the all register values on screen

Exit

Main endp

end Main



**Exercise 6:** Verify that the contents of the ECX register are now correct. (1 marks Modern toolwork)

In decimal value,

ECX = 3 \* 16^1 + 2 \* 16^0 = 50

Hence the contents of the ECX register are correct

**Exercise 7**: Perform (100-30) +20 operation using minimum number of registers. (2 marks Teamwork)

TITLE Perform (100-30) + 20 operation

Include irvine32.inc

.data ; variable declarations go here

.code

Main Proc ; instructions go here

Mov al, 100

SUB al, 30

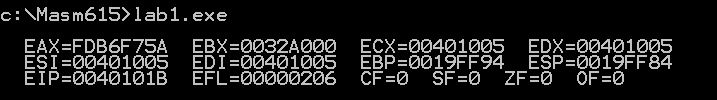
Add al, 20

Call dumpregs ; displays all register values on screen

Exit

Main endp

End main

****

This program moves 100 in al, first 8-bits of eax. Then subtracts 30 from al and stores result integer and finally adds 20, again storing the result in the al. The result is shown by two least signuficant bits of eax (al, here 5A) and rest of bits of eax output is garbage value, we are not concerned with.

In decimal value,

AL = 5 \* 16^1 + 10 \* 16^0 = 90

Hence operation is performed using single register(eax) and minimum bits(8).