# Lab 3&4 – Intro to EMU8086 & Using Basic Instructions

## Objectives

In this lab, students will learn how to use EMU8086 to write and assemble programs in assembly language. Also, students will learn how to use instructions MOV, ADD, INC, DEC, and NEG in an assembly language program.

## Introduction

Emu8086 is a program that compiles the source code (assembly language) and executes it. You can watch registers, flags and memory while your program executes. Arithmetic & Logical Unit (ALU) shows the internal work of the central processor unit (CPU). Emulator runs programs on a Virtual PC; this completely blocks your program from accessing real hardware, such as hard-drives and memory, 8086 machine code is fully compatible with all next generations of Intel's microprocessors.

**Where to start?**

1. Start Emu8086 by selecting its icon from the start menu, or by running Emu8086.exe.

2. Select "Samples" from "File" menu.

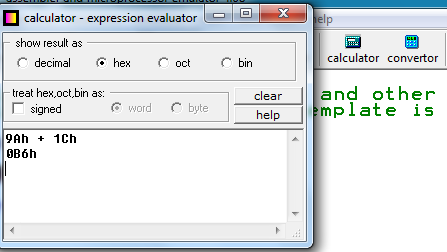
3. Click [Compile and Emulate] button (or press F5 hot key).

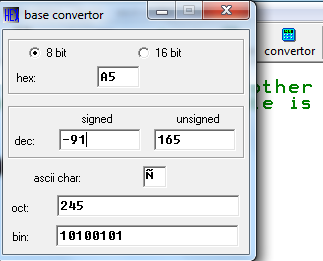
4. Click [Single Step] button (or press F8 hot key), and watch how the code is being executed.

5. Try opening other samples, all sample programs are heavily commented, so it's a great learning tool.

**TASKS**

1. Familiarize yourself with EMU8086 GUI. Explore different menu options. Learn how to use calculator and base converter. Also explore the ASCII table.





1. Press F1, to open the tutorial in browser. Read the following sections:
2. Where to Start?
3. Working with the Editor
4. How to Compile the Code
5. Assembly Language Tutorials (Numbering System, Part 1: What is Assembly Language)
6. Registers are high-speed storage locations inside the microprocessor. They are classified according to the function they performed. Each register has a name. One of the general types of register is Data Registers (that can hold data). AX (accumulator register) is a data register. Find out what other data registers can be used.
7. MOV is an instruction used in assembly language to the copy contents of source operand to destination operand. As a result, destination operands are modified but source contents remain unchanged. For example, if you execute following instruction in EMU8086, the contents will be copied in AX register.

**MOV AX, 41h**

Where MOV is an assembly language instruction, AX is a data register (destination operand) and 41h is the source operand you want to copy.

## Tasks

**Task 1:** Execute the following instructions (single-step) in EMU8086 and observe the changes in contents of destination operand.

1. MOV AX, ‘B’ (**Output:** AX: 0042)
2. MOV AX, 42h (**Output:** AX: 0042)
3. MOV AX, 42 (**Output:** AX: 002A)
4. MOV AX, 42d (**Output:** AX: 002A)
5. MOV AX, 0100 0010b (**Output:** AX: 03EA)
6. MOV AX, 123Ah (**Output:** 123A)
7. MOV AX, 0FE44h (**Output:** FE44)

**Task 2:** Now execute the following instructions (single-step) in EMU8086 and observe the changes in contents of destination operand. If any instruction gives error, correct that error.

|  |  |  |
| --- | --- | --- |
| **No.** | **Instructions** | **Register Contents** |
|  | MOV AL, 128 | AL: 80 AX: 0080 |
|  | MOV AX, F1ABh  **Corrected:** MOV AX, 0F1ABh | **Error:** No zero Prefix in h.  **Corrected Error:** AX: F1AB |
|  | MOV AX, -123 | AX: FF85 |
|  | MOV BX, 123 | BX: 007B |
|  | MOV AH, 010010001b | AH: 91 AX: 9100 |
|  | MOV 1234h, BX | Assembler take 1234h as address. |
|  | MOV DX, 33h | DX: 0033 |
|  | MOV CX, ‘AB’ | CX: 4142 |
|  | MOV CH, AL | CH: 00 |
|  | MOV DL, BL | DL: 00 |
|  | MOV AH, BL | AH:00 |
|  | MOV AX, CL  **Corrected:** mov ax,cx | **Error:** Operands do not match 16bit and 8bit registers.  **Corrected**: AX: 0000 |

**Task 3:** For each of the following assembly language instructions, what will be the possible register value (in hex)? Assume all statements are being executed in a sequence and initially all registers contain zero.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **No.** | **Instructions** | **AX** | | **BX** | | **CX** | | **DX** | |
| **AH** | **AL** | **BH** | **BL** | **CH** | **CL** | **DH** | **DL** |
|  | MOV AX, 011b | 0003 | | 0000 | | 001A | | 0000 | |
|  | MOV BH, 10H | 0003 | | 1000 | | 001A | | 0000 | |
|  | MOV CX, 10 | 0003 | | 1000 | | 000A | | 0000 | |
|  | MOV DX, 1234H | 0003 | | 1000 | | 000A | | 1234 | |
|  | ADD DH, DL | 0003 | | 1000 | | 000A | | 4634 | |
|  | ADD AX, BX | 1003 | | 1000 | | 000A | | 4634 | |
|  | INC AX | 1004 | | 1000 | | 000A | | 4634 | |
|  | INC BL | 1004 | | 1001 | | 000A | | 4634 | |
|  | DEC DX | 1004 | | 1001 | | 000A | | 4633 | |
|  | DEC CH | 1004 | | 1001 | | FF0A | | 4633 | |
|  | NEG CX | 1004 | | 1001 | | 00F6 | | 4633 | |
|  | NEG BL | 1004 | | 10FF | | 00F6 | | 4633 | |