**Group: Tera-Bytes**

**Topic:** Introduction to assembly language.

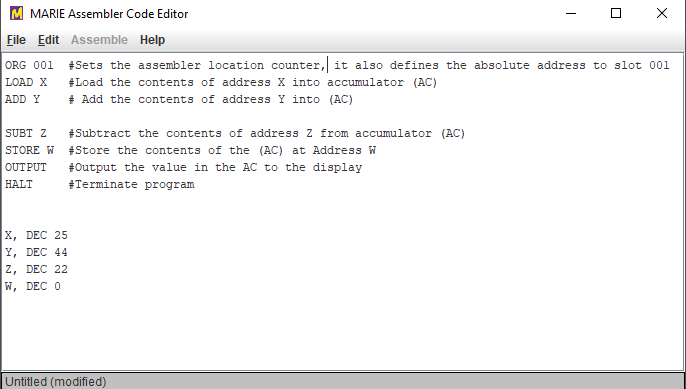
**Aim:** To construct a simple assembly program and run it in the Marie simulator and inspect the data path followed and the various registers involved in its execution.

**Introduction:**

MarieSim is a computer architecture simulator that is based on the MARIE (Machine Architecture that is really Intuitive and Easy) architecture that helps in the understanding of the basic computer organizational structure and its operations. The graphical interface of MarieSim also provides a data path simulator that allows users to trace the instructions as they are transferred through buses to different components of the central processing unit. It aids users in witnessing first-hand how assembly language statements affect the registers and memory of the computer system.

**Body:**

The following code was written in the MarieSim text editor then saved, loaded and executed by the simulator (has accompanying comments as explanations):

Figure 1: MarieSim code editor

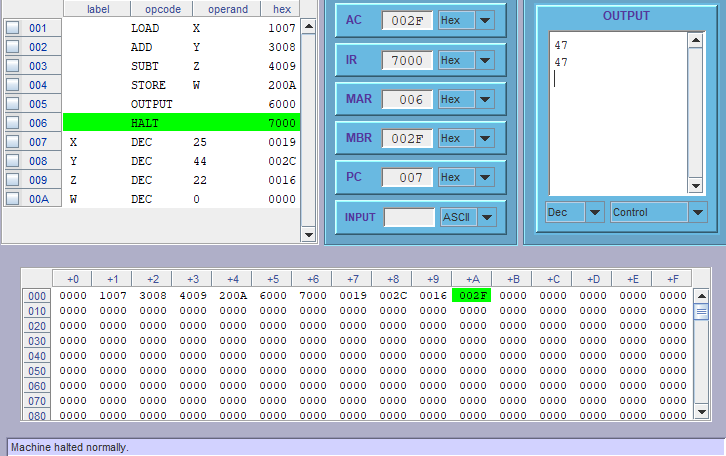


Figure2: Shows execution of assembly code and output

Figure 2 Shows the assembly language statements as they are executed line by line, the green highlight shifts with every phase of execution highlighting the line being executed. Their hexadecimal equivalent is also shown on the right hand-side.

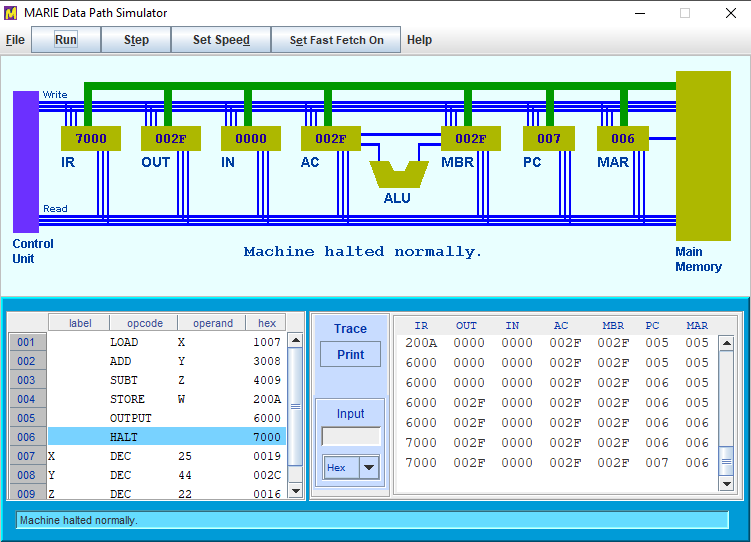


Figure 3: MarieSim data path simulator:

Figure 3 shows a visual representation of all the registers that the instructions go through during execution. During execution the instruction being executed is highlighted in green in the program monitor table. As the instructions move from one register to another their path is drawn graphically to trace their movement.

List of components:

* IR: holds instruction code.
* OUT: out register, holds output character
* IN: input register, holds input character
* AC: accumulator, processor register
* ALU: arithmetic logic unit, executes logical calculations and operations
* MBR: memory buffer register, stores data being stored to and from immediate access storage.
* PC: program counter, register that contains address of instruction being executed.
* MAR: memory address register, stores the addresses of instructions to be executed.

**Conclusion:**

The Marie simulator simplifies the learning of a computer’s underlying architecture and organization, this was clearly show in figure 3 where the data path of instructions was clearly drawn in real time and the various registers responding and changing as instructions were transferred from one register to another.