**CISM 314**

**KTD GROUP**

**LAB 2 Report**

**Title: Introduction to Assembly language and MARIE language and architecture.**

**Aim: Writing an assembly program using Marie Stimulator.**

Assembly language is a low level programming language for computers. There are different kinds of assembly language and each language it has a certain computer architecture that it correspond with or works with. For the purpose of this practical we will be working with MARIE architecture and we will be demonstrate how it works with a simple assembly program.  
MARIE Architecture is made of components that we call registers; which are **AC** or Accumulator: which is there for intermediate data is storing within the AC, **PC** or Program Counter which stores the current position of the instruction, with each instruction having its own address, **MAR** or Memory Access Register which stores or fetches the data at the given address, **MBR** or Memory Buffer Register which stores the data when being transferred to or from memory and **IR** or Instruction Register which holds the current instruction. This architecture is designed easily for student to understand how the computer instructions are processed from a simple high level programming language up until to assembly language.

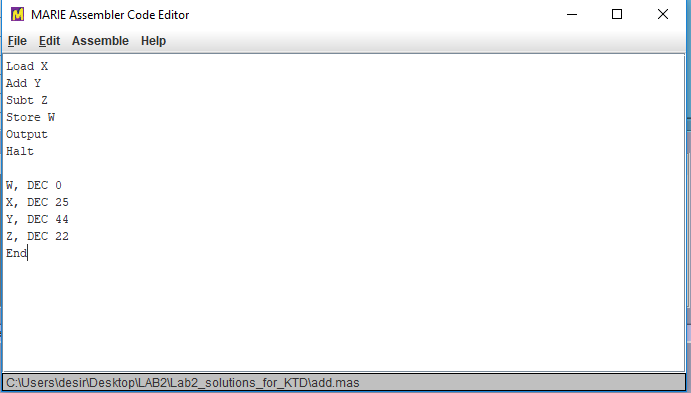


Figure 1: Assembly code in the MARIE code editor

Figure 1 is a screenshot that shows the assembly program in the MARIE code editor, this assembly program, will add the add X and Y and subtract Z, which are assigned the values 25, 44 and 23.We firstly load x into the accumulator (AC).Secondly we add Y to the accumulator which will give us X+ Y, from there we subtract Z from the sum of X and Y in the accumulator. And then we store the value of X + Y –Z the W which will show us the output. And we include HALT so that the program can end.

From there we declare our variables and we use the DEC to indicate that we are working with integers not decimals while declaring. After this save the program then assemble it.Leave the Marie code editor window and go to MARIE Simulator and load the program.

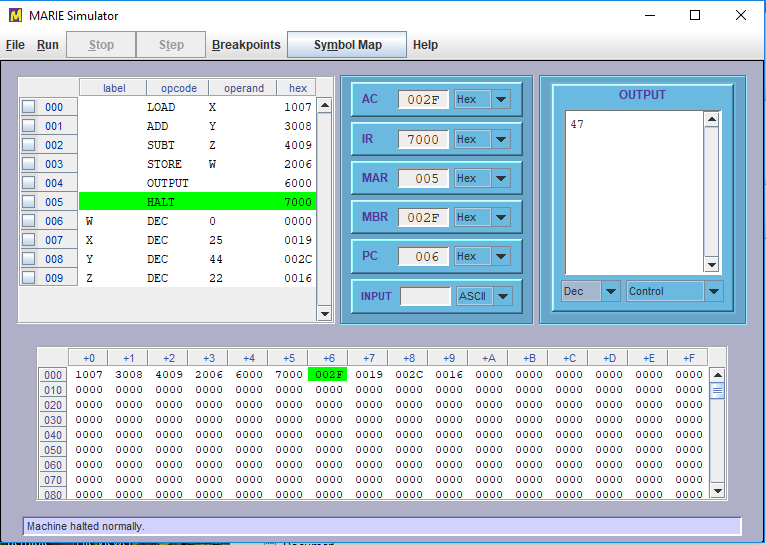


Figure 2: Assembly code loaded in the MARIE simulator

Figure 2 shows the MARIE simulator after an executable file has been loaded. The program window shows the assembly language statements as they were written, along with their hexadecimal equivalents. At the left-hand side of the program screen, there are addresses of the program statements. The statement that has just been executed by the simulator is shown in green highlight. When the program is first loaded, the green highlight will be on the statement at the first address of your program. After you will also notice that the PC register is set at the address of the first statement in the program, this indicates that this is the *next* statement that will run.

The memory monitor has the hexadecimal program instructions in the addresses that match to those in the program window. A green highlight will move to different memory location as the program runs, accessing various storage locations.

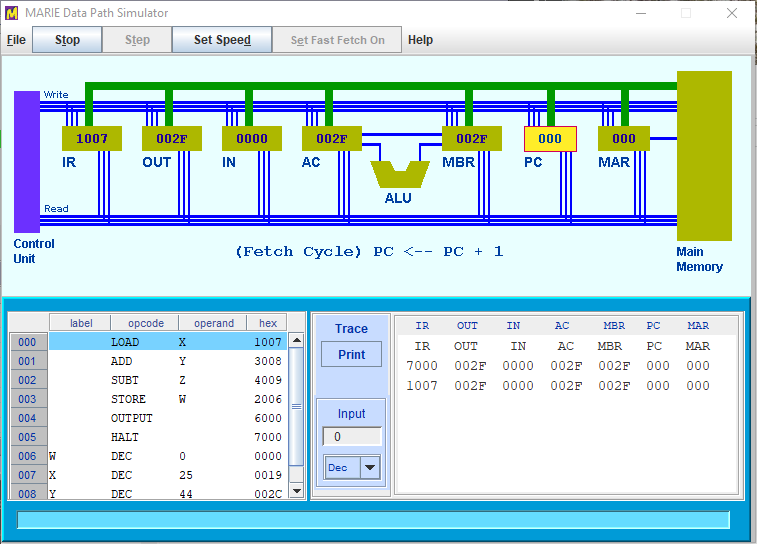


Figure 3 : Assembly code in the graphical environment of the MARIE data path simulator

Figure 3 shows the graphical environment of the MARIE data path simulator. It show data movement within the MARIE machine. As the program executes, the instruction being executed is highlighted in the program monitor table. The corresponding data movement operations are drawn in the graphical area of the screen. Whenever a component is participating in a data movement operation, it is reduced in a brighter color.

The control unit at the left of the graphical area uplifts control lines that enable and disable the components of the system. There are two sets of register control lines, one set that enables reading from a register, and another that enables writing to a register. The binary value of the number of the register is placed on the appropriate control lines.

During each operation when a register value changes, the contents of all registers are printed in the trace window at the lower right-hand side of the monitor area.

In conclusion assembly language is a programming language that is closest to machine language, that is binary 0s and 1s.All the computer instructions are processed in the ALU and each personal computer has a microprocessor that manages the computer's arithmetical, logical, and control activities. Each assembly language is specific to a particular computer architecture. And for the purpose of this practical we were working MARIE architecture which has the components accumulator, program counter, memory access register, memory buffer register and instruction register: and we demonstrated how these components(registers) work together with the sum and subtraction assemble code.