NORTH WEST UNIVERSITY

FACULTY OF NATURAL AND AGRICULTURE, SCIENCE

DEPARTMENT OF COMPUTER SCIENCE

TUMELO MOIPOLAI

26122197

JULIA MASENYA

26315718

REMOFILWE MOSADI

26718545

CISM 314

COMPILER DESIGN

Practical Lab 2

Assembly Language (Marie simulator)

Introduction

In this practical we make use of assembly language using a simulator called MARIE Machine Simulator. Marie simulator was written in the java language so that it can be portable to any platform for which a java virtual Machine (JVM) is available. As the simulator executes a program, the instruction the monitor area are highlighted along with any memory in the memory area that the instructions is accessing. During the course of executing a program instructions, status message may appear in the message area at the bottom of the screen. When your program ends you will see either a “program halted normally” or “program halted abnormally” message. If you never see the message, either your program hasn’t started running yet, or it is in a loop and you’ll need to halt it normally.

The Marie Editor

During the execution of our program, we first set our origin to 112, that’s where our program will be located on our memory area. Secondly, we stored in memory the parameters “W” as is going to store the sum of “x” and “Y” and subtract “z” which are declared as 25, 44 and 22 respectively and the “X” Dec 25 represents the value of X with value of 25 in decimal form. The output is used to display the results by printing them on the output section before the program terminates, and the halt operator is used to terminate the program. There after our code was converted to assembly language by pressing then save the file as



Figure 1: MARIE Assembler Code Editor

The assembled code was loaded in the MARIE Simulator by selecting an option File then Load Menu from the Simulator. The program monitor window displays the assembly language statements as they are written alongside with their respective statements in Hexadecimal form. The left side of the monitor shows the program addresses and the statements which has been executed by the simulator is highlighted in a green colour and when the program is loaded for the first time the green highlight will be on the statement of the first address on the program. The green highlight will move to different memory locations once the program runs, accessing many different storage locations.

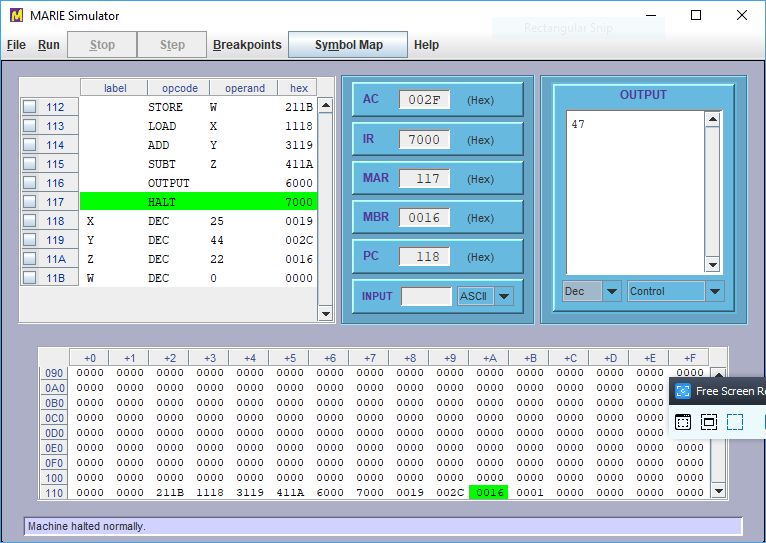


Figure 2: MARIE Simulator

The Data Path Simulator is to show data movement within the MARIE machine. The MARIE data path runs only programs that have been assembled by the MARIE assemble and it specifically load object code that have been saved with the .MEX extension. During the execution of the program the instructions been executed are highlighted in the program monitor table. The corresponding data movement operation is then animated in the graphical section of the screen. Every time a component is participating in the data movement operation it is rendered in a bright colour.

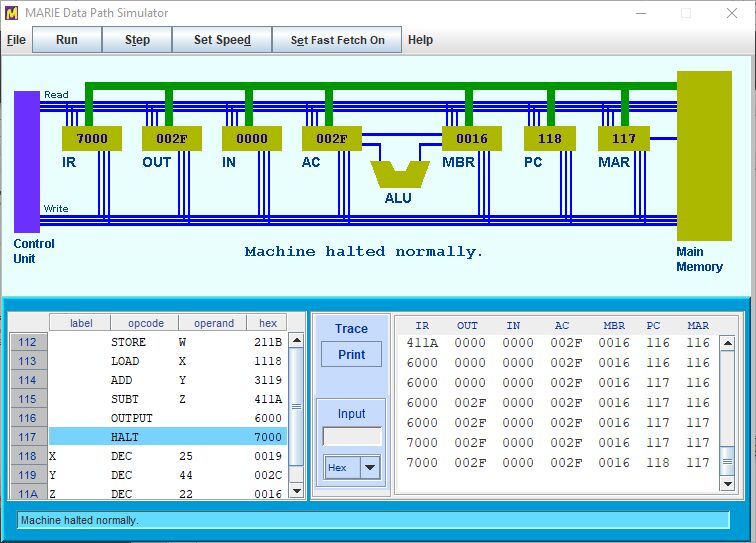


Figure 3: MARIE Data Path Simulator

Conclusion

The purpose of the practical lab was met, which was to see how the MARIE simulator works by using a simple program The group learned with demonstrations, how the Marie simulator works. The current address accessed was highlighted in green colour which made it possible and quick for us to track the movement of data within the MARIE machine.