



In this experimental work, we will work on MARIE software. MARIE ('Machine Architecture that is Really Intuitive and Easy') is a machine architecture and assembly language from The Essentials of Computer Organization and Architecture. First, the installation process will be given and then usage and some examples will be explained in the following parts of the lab sheet.

1. MARIE Installation

the publisher provides a set of simulator programs for the machine, written in Java. MARIE.js is a JavaScript version implementation of MARIE. For that reason, we must install **JDK** (Java Development Kit) and **JRE** (Java Runtime Environment) first. This two software are provided by Oracle free and can be downloaded from the web addresses given below.



JRE Download web address:

<https://www.oracle.com/java/technologies/javase-jre8-downloads.html>



JDK Download web address:

<https://www.oracle.com/java/technologies/javase-jdk14-downloads.html>

After downloading and installing JDK and JRE, we must add the file locations (where you installed jdk and jre) of these software to the "**Environment Variables**". Then;

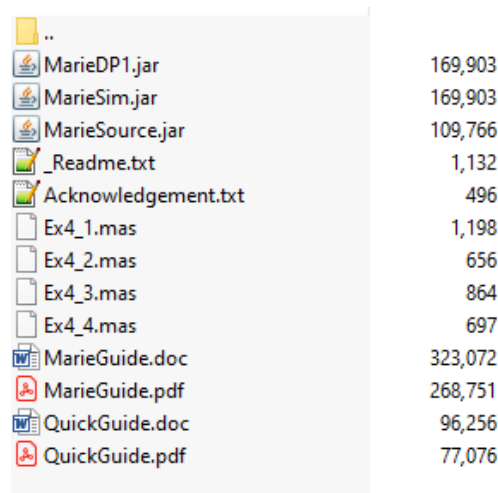
Right Click on This PC -> Properties -> Advanced System Settings -> Advanced Tab -> Environment Variables

On the new windows that is opened, we add new variable by clicking "NEW..." button and paste the copied file path. This process must be done for both JRE and JDK.

Finally, we can download "MARIE and Datapath Simulator" from this link (<http://computerscience.jbpub.com/ecoa/3e/simulators.aspx>). The downloaded file will be a .zip file. Now we are able to run the MARIE.

2. Running MARIE

To run MARIE, all the files in the downloaded .zip file must be extracted in a folder somewhere in the computer. Following image shows the content of zip file.



File Name	Size
..	
MarieDP1.jar	169,903
MarieSim.jar	169,903
MarieSource.jar	109,766
_Readme.txt	1,132
Acknowledgement.txt	496
Ex4_1.mas	1,198
Ex4_2.mas	656
Ex4_3.mas	864
Ex4_4.mas	697
MarieGuide.doc	323,072
MarieGuide.pdf	268,751
QuickGuide.doc	96,256
QuickGuide.pdf	77,076

The following operations will be done on command prompt. To do this, we open “cmd” in the same directory with above files.

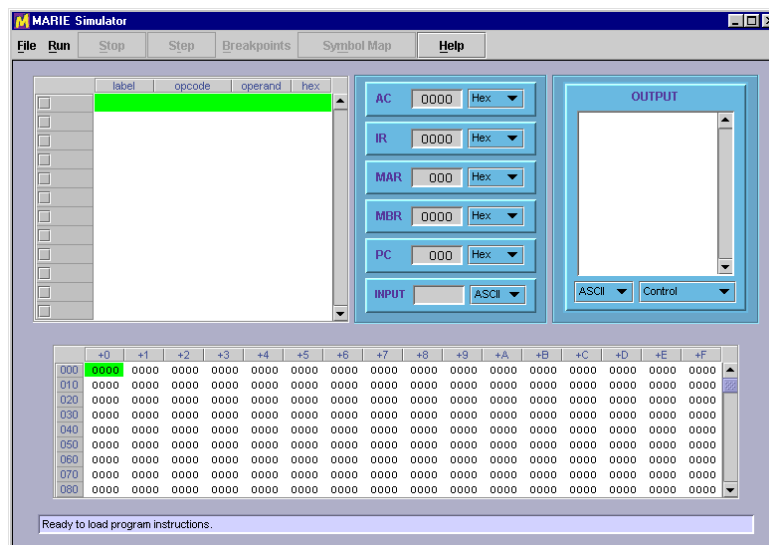
The second step is uncompressing the MARIE Simulator. After we opened cmd, the following commands is entered

```
jar xvf MarieSim.jar
```

After pressing enter, a new folder named “MarieSimulator” will be created. To run the Marie Simulator, in the same command prompt, we execute the following command;

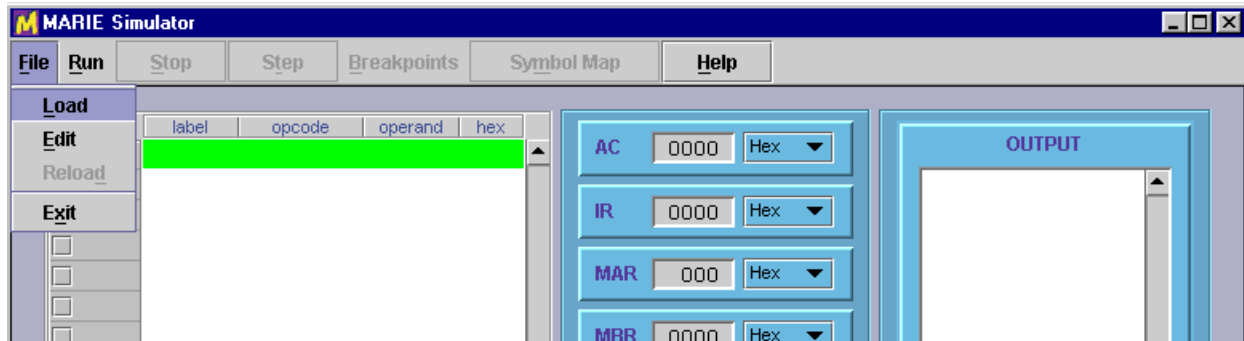
```
java MarieSim1
```

The above command will execute the simulator which shown in following image.

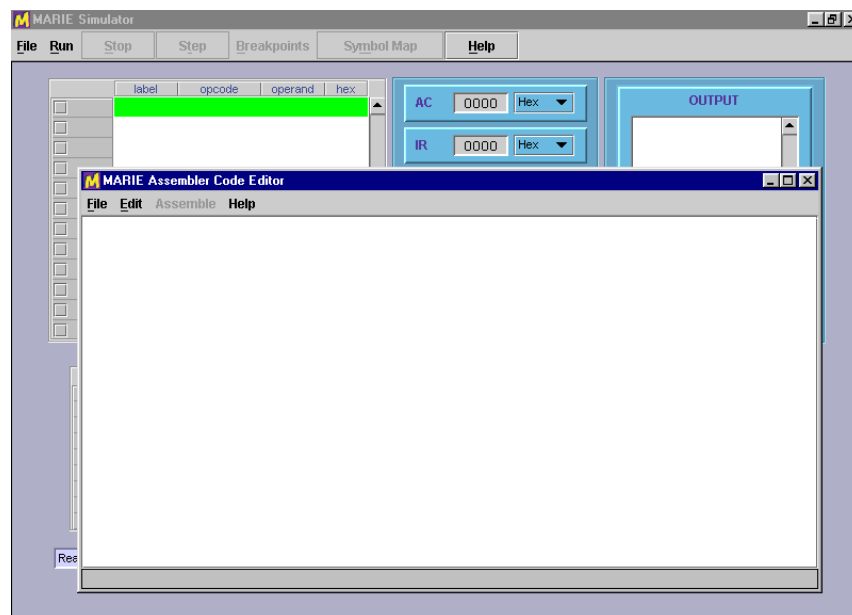


3. Making Simulations on MARIE

To write a program from scratch, you should select the File -> Edit option shown in the following image. The Edit option gives you a simple way to write and assemble programs in MARIE assembly language.

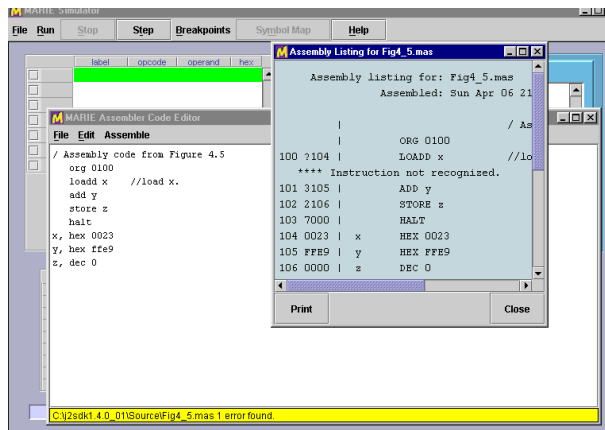


Although you can use any plain text editor to create your source code, the simulator's built-in editor gives you one-button access to the assembler. The MARIE editor frame is shown in the image below.

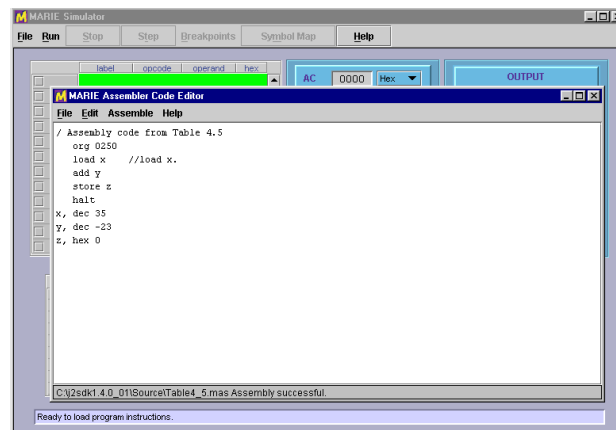


MARIE assembly code source files must have an ".mas" extension, for MARIE Assembler. Both the editor and the assembler recognize files of this type. Once you have saved a file with an ".mas" extension, the Assemble menu option becomes enabled and you can assemble your program by selecting the Assemble current file menu pick. If you load an existing ".mas" file, the Assemble button is automatically enabled. Any modifications that you have made to your assembly-language file are automatically saved by the editor prior to its invoking the assembler.

If the assembler detects errors in your program, the editor sends you a message and the assembly listing file appears in a popup frame as shown in image below on the left. All that you need to do is correct your program and press the Assemble current file button once more. If the file contains no other assembler errors, you will see the screen shown in image below on the right. If you wish, you can display or print the assembly listing file, by using the editor or any text-processing program.



An Unsuccessful Assembly

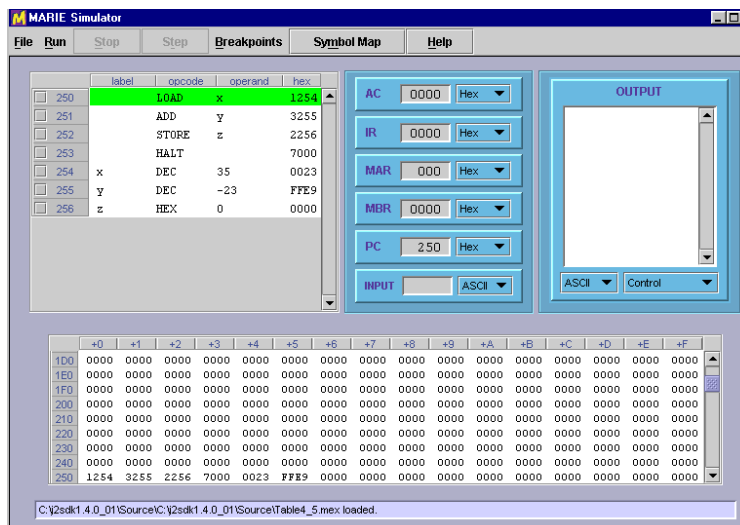


A Successful Assembly

4. Loading Your Program

After you have successfully assembled your program, you must load it into the simulator by selecting the File | Load menu option from the simulator. This option brings up a file chooser panel that lists all of the MARIE executable files in your current directory, and the names of other directories that are available to you. All you need to do is highlight or type the name of the file that you wish to run.

Note: Each time you reassemble a file, you must reload it.



The above image shows the MARIE simulator after an executable file has been loaded. The program monitor window shows the assembly language statements as they were written, along with their hexadecimal equivalents. At the left-hand side of the program monitor, you will see the addresses of the program statements. The statement that has just been executed by the simulator is shown in green highlight, so that you can see the effect that the instruction has had upon the state of the machine. Of course, when the program is first loaded, the green highlight will be on the statement at the first address of your program. You will also notice that the PC register is set at the address of the first statement in your program, indicating that this is the next statement that will be run.

5. Example Program to Run

```

Load X
Add Y
Subt Z
Store Result
Output
Halt
X, Dec 10
Y, Dec 20
Z, Dec 5
Result, Dec 0

```

6. Marie Instruction Set Cheat Sheet

Mnemonic	Hex	Description
Add X	3	Add the contents of address X to AC
AddI X	B	Add indirect: Use the value at X as the actual address of the data operand to add to AC
Clear	A	Put all zeros in AC
Input	5	Input a value from the keyboard into AC
Halt	7	Terminate program
Jump X	9	Load the value of X into PC
JumpI X	C	Use the value at X as the address to jump to
JnS X	0	Store the PC at address X and jump to X+1
Load X	1	Load contents of address X into AC
LoadI X	D	Load indirect: Use the value at X as the address of the value to load.
Output	6	Output the value in AC to the display
Skipcond X	8	Skip next instruction on condition (See note below.)
Store X	2	Store the contents of AC at address X
StoreI X	E	Store indirect: Use X the value at X as the address of where to store the value.
Subt X	4	Subtract the contents of address X from AC

Note regarding use of SKIPCOND:

The two address bits closest to the opcode field, bits 10 and 11 specify the condition to be tested. If the two address bits are 00, this translates to "skip if the AC is negative". If the two address bits are 01, this translates to "skip if the AC is equal to 0". Finally, if the two address bits are 10 (or 2), this translates to "skip if the AC is greater than 0".

Example: the instruction Skipcond 800 will skip the instruction that follows if the AC is greater than 0.

Prepared by;

Emre Rifat Yıldız,
Ph.D. Candidate