A Quick Start Guide for the MARIE Machine Simulator Environment

MarieSim is a rich graphical machine simulation environment. Its features are best appreciated after you have experimented with a simple program or two. This guide presents the basic steps required to enter, assemble, and execute a program.

Be sure that the Java virtual machine version 1.4 (JRE) or later is installed on your machine. Visit Sun's Java site, java.sun.com, to get a copy.

Using the Executable JAR File

It is not necessary to unpack (unJAR) the MARIE machine simulator and its accompanying datapath simulator in order to run them. All that you need to do is double click on the MarieSim.jar icon to invoke the MARIE simulator, or the MarieDP1.jar file to invoke the datapath simulator. You may wish to copy, or drag and drop, the jar files to a convenient location on your system (such as the desktop). If you double click and the software will not run, check the information in the complete MarieGuide documentation on path and classpath.

Using the Uncompressed MARIE Files

If you desire to do so, you may uncompress MarieSim.jar (case sensitive) using the command:

```
jar xvf MarieSim.jar
```

The MARIE simulator is invoked using the command:

```
java MarieSim1
```

On some machines, you may have to also compile the source before the simulator will run properly. You do this through the command:

```
javac MarieSim1.java
```

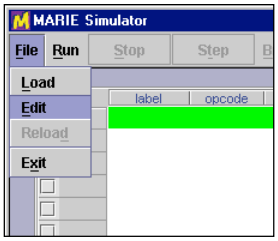
For help with the above steps, please see the complete MarieGuide documentation.

Quick Start Steps

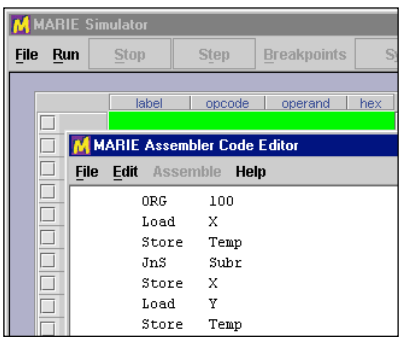
1. Start the simulator.
To invoke MarieSim using the executable JAR file, simply double click on the MarieSim JAR icon. If you've uncompressed the simulator files and prefer to run the simulator from a command prompt, log into the directory where you have installed MarieSim, and type:

```
java MarieSim1
```

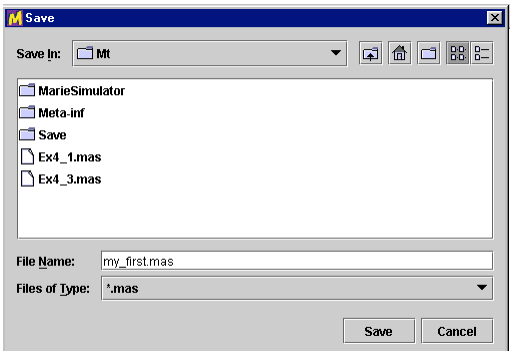
2. Invoke the code editor.
From the [File] menu on the menu bar, select [Edit] as shown in the illustration.



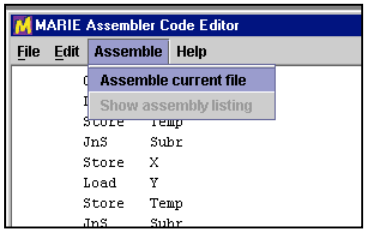
3. Enter the program source code.
Using the editor pop up window, type your MARIE assembly instructions. The MARIE editor supports basic editing functions such as cut and paste.



4. Save the program source code.
Select [Save] from the menu bar of the editor. The simulator automatically saves the file with a .mas extension.

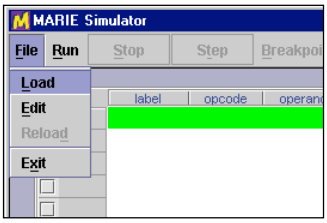


5. Assemble the program source code.
Select [Assemble] from the menu bar. If your program contains no errors, a message displays at the bottom of the editor telling you that the assembly was successful. If your file contains errors, the assembly listing will pop up on the screen, and the message at the bottom of the editor will be highlighted.



6. Close the editor.
Close the editor window by selecting [File] [Exit] or by clicking on the close icon at the upper right-hand corner of the editor panel.

7. Load the program.
From the main simulator pane select [Load] from the [File] menubar option. The file dialog pops up, showing you all of the MARIE executable files (.mex) files in the current directory. You can navigate through your directory structure if needed.

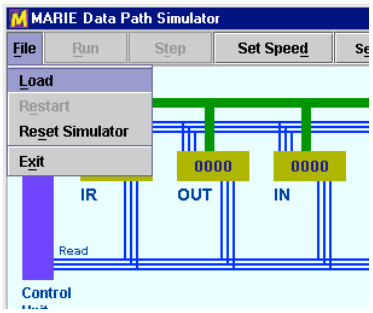


8. Run the program.
Select any of the run mode options from the [Run] menu. This menu allows you to turn step mode off and on, and set a delay between each instruction.

9. Observe data path instruction execution.
Type:

java MarieDP1

10. Load an executable Marie file.
Select [Load] from the [File] menu on the menu bar. A file dialog will appear that shows all of the MARIE executable files in the currently logged directory. (You can navigate the directory structure as needed.) Note: The data path simulator will load only executable (.mex) files that have been assembled in the Marie simulator.



11. Run the program.
To begin execution, you can select [Run] or [Step]. If you select [Step], a single fetch-decode-execute cycle is processed in the simulator. [Run] executes the program without pausing. (You can stop the program during its execution.) If you find that observing the instruction fetch part of the simulation becomes boring, you can turn "fast fetch mode" on. This feature reduces the delay between machine cycles during instruction fetch only. The machine cycle delay is restored for the rest of the instruction execution.