Lab 01
MARS tutorial

1 Objective

This lab provides a quick tutorial for running an assembly program in MIPS Assembler and Runtime Simulator (MARS).

2 Assumptions

It is assumed that you have Java J2SE 1.5 or later installed in your system.

3 Background

MARS is an interactive, user-friendly simulation environment for the Microprocessor without Interlocked Pipeline Stage (MIPS) Microprocessor (μP) developed by Missouri State University for educational purposes. MARS assembles and simulates 155 basic instructions of the MIPS-32 instruction set, 370 pseudo-instructions and 17 system calls used for console and file Input/Output (IO).

4 Tutorial

This section provides the instructions necessary for downloading MARS and simulating a test program.

4.1 Download MARS

MARS homepage is `http://www.cs.missouristate.edu/MARS/`. Your first task is to download the latest version of MARS, which is 4.5 at the moment of writing this tutorial, from `https://courses.missouristate.edu/KenVollmar/MARS/download.htm`. Since MARS runs on Java, it is not necessary to perform any installation steps. In order to open MARS, you simply need to open the recently downloaded file `Mars4_5.jar`.

4.2 Simulating a test program

For this part of the tutorial, you are provided with the test program `Fibonacci.asm`. This file is also available for downloading from MARS website. Once you open MARS, perform the following steps in order to assemble and execute a test program.

1. Load the test program by selecting **File** → **Open** and selecting `Fibonacci.asm`. Alternatively, use the icon shown in Figure 1.

Figure 1: Open icon.

2. In the top menu select **Run** → **Assemble** in order to assemble the provided program. Alternatively, use the icon shown in Figure 2.

Figure 2: Assemble icon.

3. You should see the message `Assemble: operation completed successfully.` in the **Mars Messages** window on the bottom part of the user interface. In addition to this, you should see that the assembly code has been loaded into the **Text Segment** window within the **Execute** panel, as shown in Figure 3.

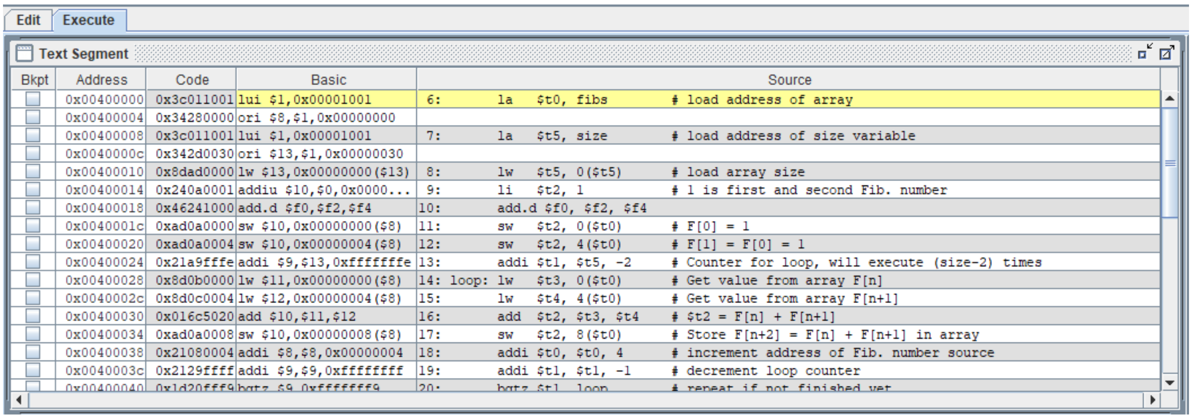

Figure 3: Text Segment window.

4. MARS allows us to select the speed at which the simulation will be run. For this part of the tutorial, select the maximum speed by sliding the **Run speed** slider to the farthest right, as shown in Figure 4.

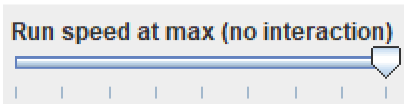

Figure 4: Run speed slider.

5. Click on the run icon shown in Figure 5. This will execute the test program until it reaches its end.

Figure 5: Run icon.

The result of the execution of the program should appear in the **Run I/O** window in the bottom part of the user interface, as shown in Figure 6.

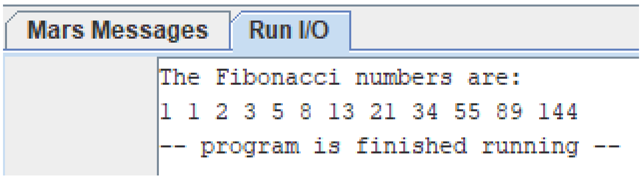

Figure 6: Result of program execution.

6. Restart the simulation by clicking on the icon shown in Figure 7.

Figure 7: Restart icon.

7. Modify the run speed slider in order to execute the program at a ratio of `1 inst/sec`.
8. Click on the run icon and observe.
9. In this case, MARS is executing the program one instruction per second. Here, the executed instruction, register values and main memory data are highlighted in real time in the **Execute**, **Registers** and **Data Segment** panels, respectively.
10. Restart the simulation again and this time, run the simulation step-by-step by clicking on the run step icon of Figure 8.

Figure 8: Run step icon.

Here, you have total control over the execution of the program. Observe how the registers and data memory update at each step of the simulation.

11. You may undo the last simulated step by clicking on the undo icon of Figure 9.

Figure 9: Step back icon.

12. Try to understand the test program and to understand what's going on inside the MIPS at each simulation step. For this purpose, MARS provides an extremely useful **Help**, which may be accessed by clicking on the help icon shown in Figure 10.

Figure 10: Help icon.

The help menu of Figure 11 presents a description of the basic and extended instructions, the directives, syscalls, exceptions and macros available in the MIPS. This should help you to understand the test program provided, particularly the use of `syscall` instructions.

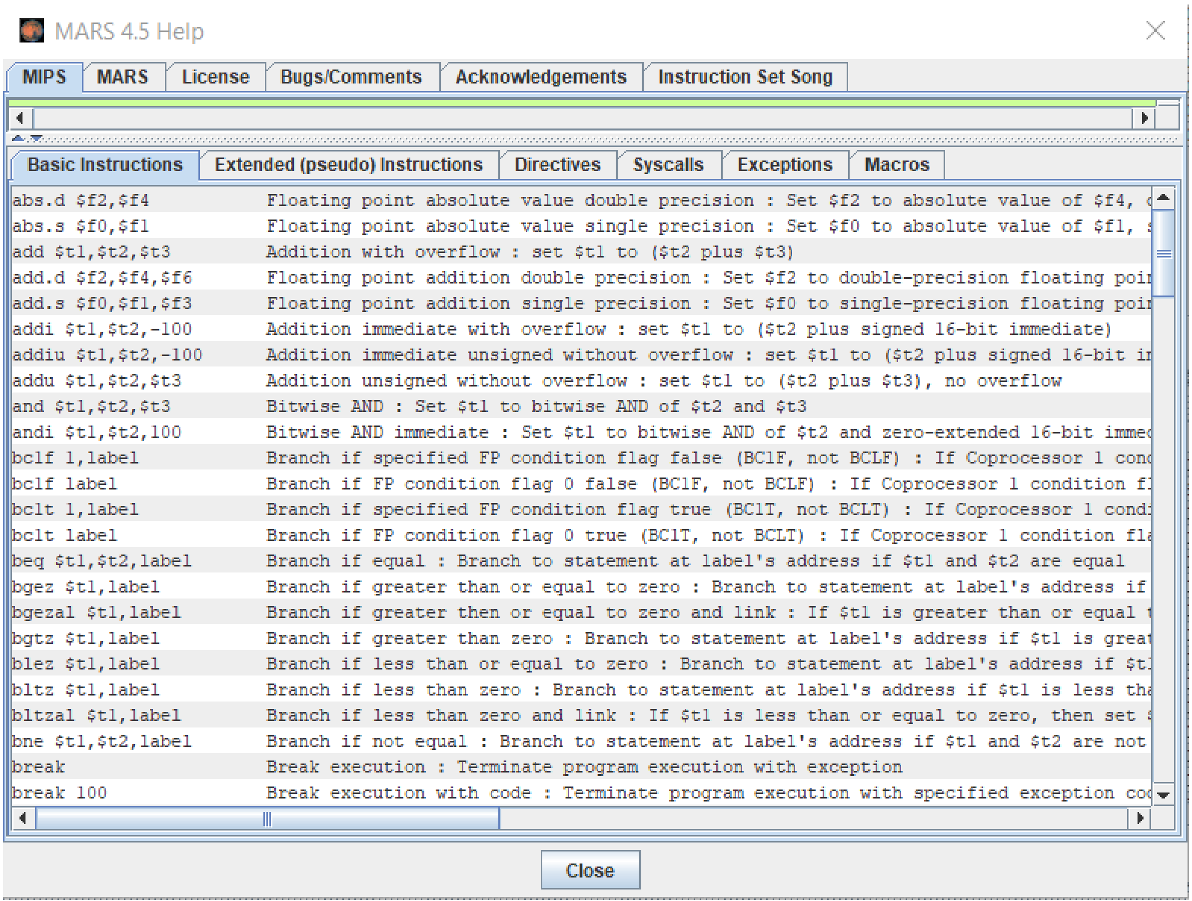

Figure 11: Help menu.

5 Lab work

The following sections describe two basic exercises for understanding both MARS and MIPS assembly language.

5.1 Exercise 1 - 40%

Create a copy of `Fibonacci.asm` and name it `Fibonacci_reverse.asm`. This version of the program should accomplish the following specifications.

1. Compute the first 20 numbers of the Fibonacci series.
2. Print out the Fibonacci series in reverse order, *i.e.*, your program should start by printing the value 6765 and finish with the value 1.

5.2 Exercise 2 - 60%

The objective of this exercise is to translate a piece of high-level language pseudo-code into assembly language. Your task is to generate an assembly language program that implements the pseudo-code of Listing 1.

```
int N = 100;
int x = [N];
x[0] = 0;

for(i = 1; i <= N; i++){
     x[i] = x[i-1] + i;
}
```

Listing 1: Exercise 2 high-level pseudo-code

Your program should compute and place all values of `x` in contiguous memory addresses. Additionally, your program should print all values of `x` in the **Run I/O** panel using `syscall` instructions.

5.3 Deliverables

Prepare a single `.zip` file with the following files.

1. `Fibonacci_reverse.asm` of Section 5.1.
2. A screenshot of **Run I/O** panel after executing `Fibonacci_reverse.asm` of Section 5.1.
3. Your assembly code for Section 5.2.
4. A screenshot of **Data Segment** panel after executing your assembly code of Section 5.2. This screenshot should demonstrate that all `N=100` values of `x` are stored in contiguous memory locations.
5. A screenshot of **Run I/O** panel after executing your assembly code of Section 5.2.

Submit your assignment through Canvas before 23:59 hours on Monday August 31st 2020. Please send any questions to isaac.perez.andrade@tec.mx.