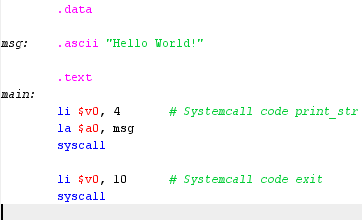
# Lab 5

# MARS Introduction

In this lab you will be introduced to the MARS (MIPS Assembler Runtime Simulator) application. When you complete this lab you will be able to assemble simple MIPS assembly programs.

Part 1.

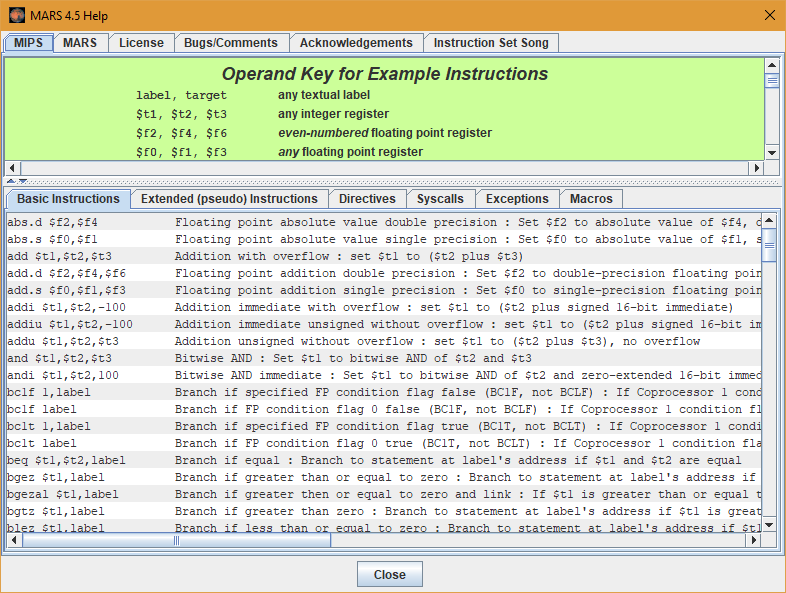
Before you begin. Be sure to save all your work in this lab for future reference. Future assignment may make use of the material in the labs.

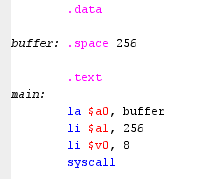
Open MARS. You will be operating MARS in 1 of 2 possible states. Edit or Execute. Obviously, Edit mode is where you will by entering your source code. Let’s start by entering a simple “Hello World” program. From Edit Mode create a new source file File -> New. Save it as “HelloWorld.asm”.

Once entered, you can assemble the program, Run -> Assemble. If you had no errors, the screen will flip into Run Mode.

Run the Program. At the bottom of the screen you should see hello world.

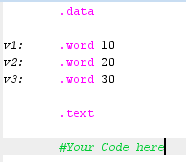
Part 2.

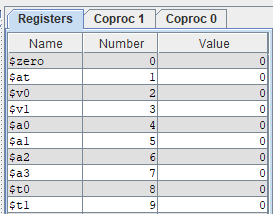
MARS has many features, including an extensive help facility. This can be accessed with F1. It should be used as a reference where needed. Note that the syscalls which make up the bulk of the IO capability are extensively listed. Let’s expand on the hello world program. We will write a program to read in a string and then print it back out.



Create a new program, and using “Hello World” as a template, add the above code. Buffer is an empty memory space of size 256 bytes. We will read our string into the buffer, which will add the null terminator. $a1 is the max number of bytes to read. Run your program.

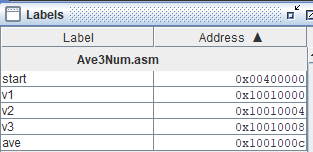
Part 3.

Write a program which will compute the average of 3 numbers, the numbers come from stored values in memory, v1, v2 and v3. Ave will be of type integer. To the right is the start of the program. You will be required to fill in the details as a lab exercise. Notice, I have not define where the solution is to be stored. For now, compute the solution to a register, if you wish, output the result to the console.

Once completed and you run your program, notice to the far right of the run screen a list of the registers. These will contain the results as you go through your computation. You should be able to locate the final solution within 1 of these registers. Note: everyone will likely of used different registers, these would be programmer specific.

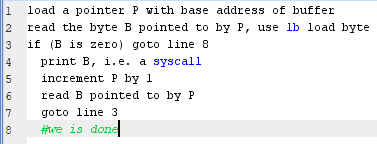
Let’s further the investigation. Reset the program, reset command under Run. F7 and F8 allow you to step through the code 1 line at a time. Try this and watch the register values change. Notice the source line is highlighted as you step.

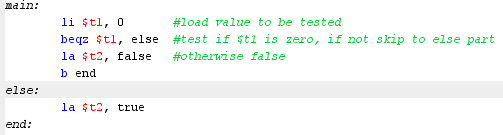
Part 4.

Modify the above code to write the result back into memory using an appropriate variable name. If you have not done so, print the average to the console. Once run, you can directly look at the info stored in memory. Any label defined - references a memory location. Clicking on the label, will highlight the memory location and thus the data saved to that location.

Part 5.

Let’s go back to the 2nd program. Make a copy of the program call it “printByChar.asm” or something descriptive. We want to read the string as you did in part 2, except this time we will replace the print string with: print the string char by char. Note: all strings will have a null character at the end, this is effectively a zero. The pseudo code will look as follows:



Work on this for the rest of your lab.

Comment your code with Name and Student number at the top. Groups of assembly code can be given a nice descriptive semantic comment. The example to the right describes an if else statement. The comments should enhance the understanding of code blocks and not repeat the code.