Lab 05: Intro to MIPS
CSE-031-01 **Points: 20**

Overview

In this lab, we will get ourselves familiar with the programming environment we will use for MIPS programming.

Getting Started

Before we begin any activities, create a directory (`Lab_5`) inside the `CSE31` directory we created in the first lab. You will save all your work from this lab here. **Note that all the files shown in green below are the ones you will be submitting for this assignment.**

You must have a clear idea of how to answer the lab activities before leaving lab to receive participation score.

What is MARS?

Since we cannot manipulate the CPU inside our computers directly (none of us use MIPS CPU, and for security reasons), we need a software to simulate a MIPS CPU for us. The simulator we are using is called **MARS**.

TPS (Think-Pair-Share) activity 1: Discuss questions 1 – 4 (25 minutes) while paired with your classmates assigned by your TA (you will be assigned to groups of 3-4 students) and record your answers in a text file named `tpsAnswers.txt` under a section labelled "TPS 1" (***you will continue to use this file to record your answers to all the TPS questions that follow in the lab handout***):

1. Before we use a new tool, we will need to find out how to use it. MARS **DOES NOT** mean that MIPS is an alien language (well, sort of). Work with your partner and find out what MARS stands for.
2. Since you have found out what MARS stands for, you probably have found out the webpage of MARS as well. Visit the download page and download MARS in your computer. To run MARS, just double-click the downloaded ***jar*** file. You will need Java to run it. Note: MARS is not pre-installed in the lab computers, so you need to download it if you are using a lab computer. But before running the jar file you will have to *mark* the file as user-executable. To do this, navigate to the folder where you have downloaded the jar file and run the command `chmod u+x Mars4_5.jar`. After this, you may double-click the jar file to run it.
3. From the Tutorial materials page (you can find the link to it from the home page), save both tutorial materials (***MARS feature map*** and ***MARS tutorial***) as well as `Fibonacci.asm` in your Lab_5 folder.
4. Follow **Part 1 : Basic MARS Use** in the tutorial using `Fibonacci.asm` and discuss the following questions:
   a. How do you load an assembly file?
   b. How do you assemble (compile) the program?
   c. How do you run the assembled program?
   d. Can you run a program before assembling it?
   e. If you want to run the assembled program line by line, how to do it?
   f. How do you run the program again after it has finished running?

A simpler Fibonacci

The sample program from the tutorial looks too complicated. Let us use a simpler version of it. Load `fib.s` (yes, you can save your assembly programs as `.s` files instead) into MARS and assemble the code. Note that Fibonacci number calculation is as follows:

```
fib[0] = 0;
fib[1] = 1;
```

```
fib[n] = fib[n-1] + fib[n-2];
```

TPS activity 2: Discuss questions 1 – 8 (25 minutes) with your TPS partners in your assigned group and record your answers in `tpsAnswers.txt` under a section labelled "TPS 2":

1. What do the `.data`, `.word`, `.text` directives mean (i.e., what do you put in each section)?
2. What does line 10 do?
3. What does line 15 do?
4. How do you set a breakpoint in MARS? Set breakpoint on line 15 and list the steps of achieving this.
5. After your program stops because of a breakpoint, how do you continue to execute your code? How do you step through your code?
6. How can you find out the content of a register? How do you modify the value of a register manually while running the program?
7. At what address is `n` stored in memory? Calculate the 13th fib number by modifying this memory location.
8. Line 19 and 21 use the `syscall` instruction. What is it and how do you use it?

Your TA will "invite" one of you randomly after the activity to share what you have discussed.

Individual Assignment 1: Create `myFirstMIPS.s`

Write a new piece of MIPS code that, given a value in `$s0`, assign the following to the various `$tX` registers:

```
$t0 = $s0
$t1 = $t0 - 1
$t2 = $t1 - 2
$t3 = $t2 - 3
...
$t7 = $t6 - 7
```

In other words, for each register from `$t1` to `$t7`, you program should store the difference of the previous `$tX` register value and an incremental constant. The `$s0` register contains the initial value. Do not set the value of `$s0` in your code. Instead, learn how to set it manually with MARS (Hint: question 6 in TPS 2). Save your code as `myFirstMIPS.s`.

Collaboration

You must credit anyone you worked with in any of the following three different ways:

1. Given help to
2. Gotten help from
3. Collaborated with and worked together

What to hand in

When you are done with this lab assignment, submit all your work through CatCourses.

Before you submit, make sure you have done the following:

- Attached `myFirstMIPS.s` and `tpsAnswers.txt`.
- Filled in your collaborator's name (if any) in the "Comments…" textbox at the submission page.

Also, remember to demonstrate your code to the TA or instructor before the end of the grace period.