

CSE31: Lab #6 –MIPS programming

Overview

In this lab, we will continue to familiarize ourselves with programming MIPS using MARS. We will focus on **memory accesses** and **control flow**.

Getting started

Before we begin any activities, create a directory (Lab_6) inside the CSE31 directory you created during Lab #1. You will save all your works from this lab here.

(Exercise) Memory in MARS?

MARS not only simulates a MIPS CPU, it also simulates the memory so the CPU can work with it. We will revisit **fib.s** from Lab #5 in this exercise. Load **fib.s** into MARS and assemble the code.

TPS (Think-Pair-Share) activity 1 Paired with the classmate sitting next to you and do the following tasks (25 minutes):

1. Once **fib.s** is assembled, open the **Execute** tab (it should be opened by default after assembled). Two segments of memory are displayed here: *Text Segment* and *Data Segment*. What is the starting addresses of *Text Segment* and *Data Segment*? **Give your answers in Hex format.**
2. The *Text Segment* shows you how each actual machine code is stored in the memory (again it is simulated) and its corresponding MIPS code. Two types of MIPS code are shown here: *Basic* and *Source*. We call the *Basic* one **True Assembly Language**, and the *Source* one **MIPS Assembly Language**. From the display, what can you tell about the difference between the two in terms of their relationship with the machine code? We will cover this topic in future lectures.
3. Now, let's take a look at the *Data Segment*. How much difference in bytes are there between 2 address locations (again, the addresses are in Hex)?
4. For each address location, how many columns are there?
5. What can you tell about the relationship between the address difference and the number of columns at each address location?

6. From the source code, how do you create a new variable/label named ***m*** and set it to 20?
7. Save and assemble your file. At what address is the value of ***m*** stored?
8. Besides numbers, we can also initialize strings for the program to use. Search from the Internet on how to declare a string named ***str1*** and set it to ***"I love CSE31!"***
9. Insert the declaration of ***str1*** in your code and assemble it. From the *Data Segment*, we can see that the string is occupying 3 address locations. At what addresses is ***str1*** stored?
10. ***str1*** is stored as numerical values in the memory. Check the ***ASCII box*** and observe how it is stored. Does the display of characters agree with what you have learned from Lab #4 about how an array of characters is stored?
11. In order to print ***str1***, we will need to use ***syscall*** function. Search the Internet to find out how to print ***str1***.
12. Now let's go back to the program. Search from the Internet to find out what ***"la \$t3, n"*** does. What will be the value stored in ***\$t3*** after running this instruction? From this we can see that we cannot use the initialized variables (labels) directly in our MIPS program. We need to use ***la***, then ***lw*** to save the value into a register.

(Exercise) Create compare.s

Write a new piece of MIPS program named ***compare.s*** and complete the following exercise.

TPS (Think-Pair-Share) activity 2 Paired with the same classmate and answer the following questions (25 minutes):

1. From lectures, we have learned that we can perform different inequality comparisons (<, >, <=, >=) by using only ***slt***, ***beq***, and ***bne*** instructions. Why not having one instruction for each inequality in MIPS?
2. Declare a new variable/label (***n***) and set it to **25**.
3. Insert instructions to declare the following strings:
 - a. str1: *"Less than\n"*
 - b. str2: *"Less than or equal to\n"*
 - c. str3: *"Greater than\n"*

d. str4: "Greater than or equal to\n"

4. Insert instructions to read in an integer from users. Search from the Internet on how to use **syscall** to do it.
5. Insert code so the program will compare if the user input is **less than n**. If it is, print "**Less than**".
6. Insert code so the program will compare if the user input is **greater than or equal to n**. If it is, print "**Greater than or equal to**".
7. Now comment out your code from *part 5 and 6*. Insert code so the program will compare if the user input is **greater than n**. If it is, print "**Greater than**".
8. Insert code so the program will compare if the user input is **less than or equal to n**. If it is, print "**Less than or equal to**".



(Assignment 1, individual) Create sumAll.s

Create a new MIPS program that will ask a user to enter a number repeatedly. This program will calculate **2** sums based on the sign of the inputted numbers: **pos_sum** (sum of all positive numbers) and **neg_sum** (sum of all negative numbers). The program will stop when the user enters a '0' and print out both sums (even if they are zero).

You will be using a loop to determine when to end the program. In MIPS, ALL types of loop can be implemented with a branch statement (beq or bne). Therefore, write a pseudocode to describe your approach to this problem before writing the MIPS code. It is VERY SIMILAR to the **average.c** you have created in Lab #1.

Save your code as **sumAll.s**.

Here are the examples of the output (input is in *italic* and **bold**):

Please enter a number: **1**
Please enter a number: **2**
Please enter a number: **3**
Please enter a number: **-1**
Please enter a number: **-2**
Please enter a number: **-3**
Please enter a number: **0**
Sum of positive numbers is:
6
Sum of negative numbers is:
-6

Please enter a number: **1**
Please enter a number: **2**
Please enter a number: **3**
Please enter a number: **0**
Sum of positive numbers is:
6
Sum of negative numbers is:
0

What to submit

When you are done with this lab assignments, you are ready to submit your work. Make sure you have included the following **before** you press Submit:

- Your ***compare.s***, ***sumAll.s***, answers to the TPS activities in a text file, and a list of Collaborators.
- Your assignment is closed **7 days after this lab is posted** (at 11:59pm).
- You must demo your submission to your TA **within 14 days** (preferably during next lab so you will have a chance to make correction and re-submit/re-demo.)

