Lab 08 Specification – Exploring Assembly language programming with MIPS and Mars.
Due (via your git repo) no later than 8 a.m., Friday, 7th December 2018.
50 points
Last Lab for the semester

Lab Goals

- Practice Assembly language programming
- Understanding the use of registers

Suggestions for Success

- Take a look at the suggestions for successfully completing the lab assignment, which is available at: `https://www.cs.allegheny.edu/sites/amohan/resources/suggestions.pdf`

Learning Assignment

To do well on this assignment, you should also read

- Chapter 02 Computer Organization and Design by Patterson and Hennessy

Assignment Details

In this lab assignment, you will explore the assembly language programming world, get familiar with the use of Mars, and practice some MIPS concepts discussed in class dicussions,

Part 1: Exploring MIPS code

By this point, you have gained a bit of experience in thinking about writing low-level programs in MIPS. Let's take that a step further in today's lab. Given a prompt and some framework code, can you solve a more complicated programming challenge?

Beginning with the `StringLength.asm` code provided in the GitHub repository, please try each of the following changes and answer the accompanying questions. Write up in a short (PDF document) a response to each question.

1. Add another `.asciiz` string to the `.data` section with the text "The string length is: " and output that text before outputting the length we computed in $t1. What does your output look like now?
2. How can you make this new string appear on a different line than the length of the string?
3. What happens to the program when you change the datatype for both strings from `.asciiz` to `.ascii`? Why is this the case? (You can undo this change after you try.)

4. What happens to the program when you remove the exit syscall lines? Why is this the case? (You'll probably want to undo this change too.)

5. In addition to the `lb` instruction to **l**oad a **b**yte, you can also **l**oad a **w**ord (4 bytes) with `lw` and **l**oad and **h**alfword (2 bytes) with `lh`. What happens when you replace the `lb` instruction with each of these? Speculate on how you might fix the resulting errors. (Once again, you can undo this change after experimenting.)

6. Let's say that we want to store our string length back into memory. First, we need to add some space in the `.data` section to save this information. After both strings, insert the instruction "`length:  .word 0`," to provide space for a 4-byte integer (a "word") in memory. Next, before we exit the program, we will want to **s**tore that **w**ord into memory with the instruction "`sw $t1, length`." Which memory cell in the data section is the length saved into after you run the program? Is this where you expected?

Part 2: String Reverse

In the GitHub repo for the lab, you will find a file called `StringReverseTemplate.asm`. This file is a variation to the MIPS code we have discussed so far in our last few classes, but has been repurposed to print the string in reverse. The basic algorithm for printing this string in reverse is the following:

```
Start at the beginning of the string
While we have not found the end of the string:
    Keep stepping one character to the right
We have walked off the end of the string, so take one step back to the left
While we have not found the beginning of the string:
    Print the current character
    Take one step back to the left
Exit
```

The `StringReverseTemplate.asm` file contains comments in ALL-CAPS, which designate where one or more lines of code need to be added to create a program that will print the `str` string in reverse. You may either use this template file, or start writing a new solution from scratch. In the end, you should have a new program called `StringReverse.asm` which prints "!emosewa si SPIM" given input "MIPS is awesome!"

In order to print a character, you will want to use `syscall` #11. There is an additional note on the use of this `syscall` on the MARS Syscall webpage that you can ignore.

Submission Details

1. Your lab assignment solution file must be uploaded to your repository by the due date and time.
2. Questions about the lab? Bring them to class on Tuesday morning!

Before you turn in this assignment, you must ensure that the course instructor has read access to your git repository that is named according to the convention `cs200F2018-<your user name>`.

Grading Rubric

1. If you complete Part 1 fully, as per the requirement outlined above, you will receive a total of 30 points. There will be 5 points awarded for each individual task in Part 1.

2. If you complete Part 2 fully, as per the requirement outlined above, you will receive a total of 20 points.
3. Failure to upload the lab assignment solution file to your git repo, will lead you to receive ”0” points given for the lab submission. In this case, there is no solid base to grade the work.
4. There will be a partial credit (30% of the points allocated for the part) given if the solution had presented all the intermediate steps correctly but unable to produce the final expected result.
5. It is highly recommended to validate if the correct version of the submission is being submitted before due date and make sure to follow the honor code policy described in the course syllabus. If it is a late submission, then it is the student’s responsibility to let the professor know about it after the final submission had been made to GitHub. In this way, an updated version of student’s submission will be used for grading. If the student did not communicate about the late submission, then automatically, the most updated version before the submission deadline will be used for grading purposes. If the student had not made any submission, then in this case, there is 0 points given to the student, automatically for the lab work.
6. If you needed any clarification on your lab grade, talk to the Professor. The lab grade may be changed if deemed appropriate.