Brian Patino
CSE 2304 - Computer Architecture
Lab 03 - Report
2016-03-05

When I first started programming in MIPS, it seemed strange and new to be working directly with lots of binary bits. I thought, “how can I print anything or do anything if I’m just working with numbers?”. As the labs progressed, writing in Assembly became more natural. I started thinking “oh I can use 4 in $v0 to print out strings and I can use 5 to read integer inputs”. Doing things in Assembly that at first seemed daunting is now very simple.

I noticed that there is a great focus on the stack and stack pointer for this assignment. So I made it my prime objective to fully understand the stack and how it is used. I learned that we can use the stack to store register values and retrieve the stored value. But I wondered, “How is this helpful, why would I use this?”. To my curiosity, I tried using a register value over subroutine calls without saving the value in the stack. I used QtSpim’s debugging to go line by line to see what the registers were doing. I noticed that the register that I originally assigned was indirectly altered by the MIPS processing. I soon realized that the stack and stack pointer can be used to solve this problem.

There wasn’t certain part in this lab that was particularly difficult. In fact, I found problem 3 to be very useful. After having read up on the stack and stack pointer, it seemed very simple to create a push and pop subroutine, almost too simple, until I had to create an IsEmpty subroutine. I wasn’t too sure how I could get this done but I did have a few ideas. My first assumption was that the stack was first initialized to 0. This was clearly wrong and I realized this when I was debugging. I did however notice that the stack is initialized to some incredibly high number, so what if I saved that initialized value in a register and used it as the base line? This worked very well and I was able to create the subroutine.

The last problem was rather long but having created push, pop, and IsEmpty beforehand, greatly sped up development. I noticed that this problem asked to create an isPalindrome subroutine which is a classic high-level programming challenge. Using a stack to push and pop values really does help simplify things. To complete this part of the assignment, I used *lb* to load a single character from the given string and then I used *seq* to compare the two characters. Comparing two of the characters isn’t quite enough. I also used a counter to keep track of the character count to make sure the length of the strings are the same.