Jason Hoang

CSCE 2610 – Dr. Burke

March 8, 2015

Project 1 - Report

The MIPS version of the circular doubly linked list was easily one of the hardest programs to initially think about and begin writing down. Mostly because of my understanding of how pointers worked in assembly language was non-existent. So of course, after much collaboration with my peers in class, I finally figured out how to start writing the entire program. Most of the pointer issues came from the very first function, “AddItem” where I had to figure out how to get pointers to move in the list so that I could link the nodes appropriately. Once I mastered the use of pointers the next issue was sorting the nodes so that I could link them in order from least to greatest key value. This required me to begin using most of the $s registers at my disposal since I needed to keep these values passing through multiple functions so that they can be compared and their locations can be linked in the end. There were several logical obstacles that made the thinking behind building the code more and more complex. The main problem that I came across was within my linking. My sorting system wasn’t working properly in that it kept replacing nodes that were just added with the new nodes, thus I only had about 3 or 4 nodes in my list at a time. The problem was within the linking system that I had where I made the list be circularly linked from the beginning, and this just made the logic behind the linking system incredibly too complicated for me to dig into and fix. So my fix was to not connect the ends circularly until the very end, that way I can deal with beqz which helps to traverse the list until we get to the NULL ends instead of running into the logical issues with the circle. This pretty much cleared every issue I had with program and allowed me to progress and finish.

There were some other minor complications that came about from the overall linking system that I used, but after I successfully built the code for the additem function, the rest of the program was very easy to write and put together. I do wish that there were more $s registers for me to use as it would be easier for me to keep track of, but instead I have to reuse them throughout. However, I did make sure that they didn’t conflict with other functions and data, and I also made sure that the registers that I used handled the same types of data throughout.

From this program I can say that I learned almost everything that I could about how pointers worked and are used in MIPS code. I do believe that future endeavors into other similar programs or even any other type of program would be more easily achieved with the understanding of pointers. Of course, that would depend on the other techniques within MIPS coding that I have yet to learn or explore.