ECE 4270 - Computer Architecture

Lab 2 - MIPS Assembly to Hex and Assembly Problem Solving

Brandon Gallo and Jacob Higgins

Group 4

10/1/2020

a. Objective

The objective of this assignment was to convert a text file full of assembly MIPS instructions into their hexadecimal representation. Also a big part of the lab was to write two MIPS assembly programs. One program would perform a bubble sort, and one would find the fibonacci number of some given value. The assembly program can be used as an input to the first part of the lab, and the hex output can be used as an input to the solution to Lab 1.

b. Milestones, Work Distribution, and Implementations

This lab has two fairly distinct parts that are intended to work together once both are complete. For the first part, the assembly to hex converting, the first milestone was correctly reading in strings from a .txt file and breaking up all parts of the string removing the '$' and commas from the string. Then it was on to building a 4 byte hex representation of the instruction. The order of milestones for getting this done went as follows: converting the instruction name into its hex representation, then converting each register to its hex representation, then converting the immediate or offset areas to their hex representation. Lastly, the completed hex representation was written to the terminal and to an output file to be used as an input to the MIPS simulator from lab 1.

As far as important implementations for the assembly to hexadecimal portion of the lab, first was the conversion of the instruction name into its hex representation. This was done by many 'if statements' checking for a string match of all instructions with the current string representation of the instruction name. If a match was found, the 4 byte hex representation was OR'd with the proper value that represented that instruction in hex. While the many 'if statements' could have been used again for scanning for the register string name, instead a faster method of using a hash table was implemented. So a hash table is created and then initialized in main with all 32 register names and their corresponding hex value, so that when its time to search for a register string name, e.g. "$v0", it can be hashed and mapped in close to O(n) time, instead of many many 'if statements'. These were the two important implementations revolving around handling the parsed string representations of the instruction from the text file.

As far as who did what, help and discussion was had between the two parts of the lab throughout, though we split up the two parts for one to focus on each. I, Jacob H, focused mainly on the reading, parsing, converting, and writing, of assembly instruction strings from the input text file, to writing them to the output.txt file. This included all above mentioned implementations regarding converting the instruction to hex representation. I, Brandon G, focused on the assembly language instructions for the bubble sort and fibonacci problems that were then converted into hex to be read by the simulator. This involved converting a C algorithm into assembly language based on the MIPS instruction set architecture. For testing, I converted each instruction individually into hex format and ran it through the working simulation program to see what was going on in memory and the registers.