CPE 315-01 Computer Architecture

Lab 6

Submitted 2/28/2018

Samuel Sachnoff and Michael Silin

Professor David Retz

Cal Poly San Luis Obispo

**• Introduction**

The purpose of this lab is to build upon our previous lab and to implement a full MIPS CPU simulator. We will execute the previous MIPS instructions and run the countbin function. We will also keep track of the Clock cycles used in order to run the function.

**• Functional Requirements**

We accomplished this by first copying all of the MIPS instructions into our C file. We used the reference card which had the code partially written in C as our guide.

**• Approach**

The first thing we did was copy all of the instructions from the mips reference card into our C file. We used the reference card as a guide and we didn’t return anything. We used the arguments as pointers to our three registers - the source, destination, and target. After some slight manipulation, we saved our final register - either the target or the destination. All of our mips functions are in the executeFunctions.c file.

In order to implement our solution, we created a 32 entry array for our registers. We used a byte pointer and an integer index(our pc) in order to access our array.

We used a 1024 byte array for our memory. We initialized the beginning of our array(ie. The starting pc value) to 0. In order to read or write this, we used an offset to get to our values.

Each instruction increments the PC.

Furthermore, we used a MultiCycleState struct to keep track of the number of instruction run, number of memory references, and total number of clock cycles.

**• Discussion of any difficulties encountered in the implementation, and information relative to issues of Reliability, Maintainability, and Security.**

Some of the instructions for accessing the memory were a little tricky. Furthermore, I noticed that we had some error from the last lab that we had to change. Firstly, I didn’t realize jump addresses were relative to the pc and thus I had to change that. Secondly, for r type instruction the rd register was off because we were shifting it by 1 too much. Finally, there was a lot of repetitive code. Therefore, if given more time I would have like to add more functions and/ or structure in order to simplify the solution.

**• A summary of what you learned from the lab.**

In this lab, we learned how the MIPS instructions are converted to a language like C. We became more familiar with each of the MIPS instructions by converting the instructions to C. We also became familiar with keeping track of clock cycles and memory accesses when running our instructions.

**\*\*\*Source Code \*\*\***

**Please see the attached zip file. Thank you**