CPE 315-01 Computer Architecture

Lab 5

Submitted 2/20/2018

Samuel Sachnoff and Michael Silin

Professor David Retz

Cal Poly San Luis Obispo

**• Introduction**

The purpose of this lab is to gain an understanding of how machine instructions are decoded and executed. In this lab we converted a binary file to a memory array.

**• Functional Requirements**

Our program must convert a binary file into MIPS instructions. We must indicate whether the instruction is a R, J, or I instruction and the provide the associated opcode. We must include the function code if there is one. We also decoded the registry target, source, destination and listed these values as well. We indicated the immediate values of immediate instructions and address values of branch functions.

All other functions were declared to be invalid.

**• Approach**

Our first step of converting the binary file was to validate the header. We read each 32 bit instruction line by line and stored them in our memory array.

In our main, we then proceeded to sift through each 4 bytes of memory. We checked the first 6 bits to see if the instruction was an R type, J type, or an I type instruction. From there, we used one of our three print functions in order to further convert the binary instructions to mips assembly code.

Depending on the op code being used, we then checked for the specific instruction by sifting through the remaining bits/function codes.

If we went through all of the functions available and didn’t find an associated function, then we proceeded to label the instruction as invalid.

Finally we compiled and tested our code versus the instructor’s test files.

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

The main challenge we had for this program was reducing the complexity of our code. We used the “copy paste” method in order to work our way through the assignment. We could have made our program a lot smaller and simpler if we spent extra time writing sub-functions - for example converting the Registry values. If given more time, I would have liked to create some additional functions as there appeared to be a lot of repetitive or similar portions of code.

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

In this lab we learned how to hard code our own mips binary assembler. This program gave us experience with the C programming language. Specifically, it gave us experience loading binary files, creating and converting to memory arrays, and sifting through our data. This lab gave us a feel for the first step of creating a MIPS assembler by converting Binary or Hex to MIPS code.

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

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