Lab 5

EE 234

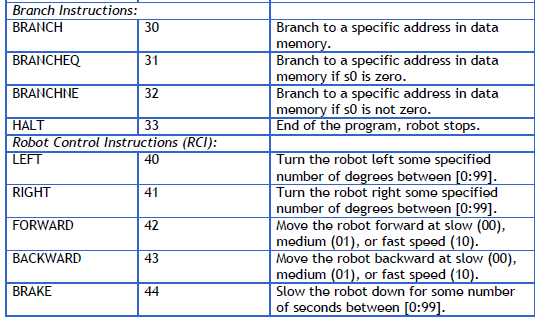
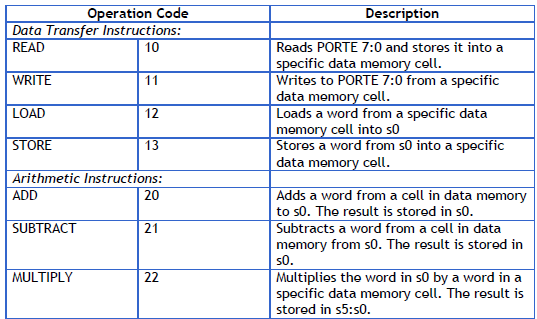
Jonathan Quisenberry

Partner: Mike Mentele

October 16, 2012

**Introduction**

In this lab the main focus is to teach the student about the fetch, decode, and execute cycle. The idea is to create a program that will do all three of these one at a time in MIPS. Ideally we could have the fetch running at the same time as the decode and the execute but for the MIPS it will not work as a pipeline because it runs linearly with MIPS. The design of the fetch, decode, and execute will be explained more in depth in the software design. The important part of this lab will be writing sub routines that can be implemented in future labs. The table below summarizes the op-code for the lab. The operator will be in the description of the table below. The lab is to write code that will follow instructions based on user inputs. So making a square will be an instruction set and the code below is what is entered into the robot.



**Software Design**

The design of the software is shown in the flowchart below.



The software design is bulky because there are many options that must be branched to for selections. Without interrupts to jump to the specific location the program must compare values one at a time until it finds a match and then jumps to the correct procedure. This program will first need to fetch the code from the data memory cell. This cell is controlled by the user input, who will load the instruction set into the $s1 register. This entails the fetch part of the cycle. The next part of the cycle will be the decode.

**Test Procedure and Results**

Overall the board functioned as expected in the demo. In order to test the different RCI commands the full set of RCI was also set up as a instruction set. This showed the TA that all the functions of the RCI worked. For the rest of the car there were no actual LED patterns to follow. So in order to test these, a step through was implemented in the code. Each step the values in the registers were checked against what they were thought to be. If any numbers did not match up then issues would have been found and taken care of. The idea is to test each sub routine in depth in order to see it has the correct functionality. If all the parts of a system work correctly, then when all the parts are put together, hopefully the whole system will work correctly. That is so long as the things that need to be preserved are not changed in the code. So in future code if the $s0 registers are used then it will need to be noted that they cannot be changed from their value. This is where the stack will be implemented more.

**Answers to Questions**

1.Explain how the operation code of each instruction indicates the category of instruction to which it belongs.

It takes the value and branches to the correct place. Therefore a 10 will jump the read and a 11 will jump to a write. The code could have been broken down father into 1 as data values and then looked at the second number and went to the read or write, but it would have added extra lines and seemed unnecessary.

2. Describe how your design would change if instruction were 16-bits instead of 32-bits.

If the intrusion size were 16 bits the lab would have shifted less in the instructions. The multiplier would have only 32 bit length. The instruction lengths would have been different and changed the code. Overall it would give us less capability with 16 bits, but the extra capability was wasted anyways because of the small instructions sizes.

3. Imagine that you were going to control a robotic car. Expand on the ROBO-MAL language and design five other instructions that would be helpful in controlling the robotic car. These do NOT have to be robot control instructions; they may be data transfer, arithmetic, or branch instructions. Also, provide the operation code for the instructions. Feel free to be creative!!!

**Conclusion – Why I Care…**

Overall the project was a success. The fetch, decode, and execute cycle is implemented in the MIPS assembly level and takes instructions from the user and runs a program based on the inputs. The main idea was to create a program that can be used again later in the class and will help to alleviate some of the work needed to run a robot. This code will most likely be used or referenced in the future. Overall the use of the stack pointer would have been helpful in some instances. Instead of looking at the .data location we could have just saved the address of the instruction onto the stack and then used it as we wanted and saved it back to after it was used. The stack helps to preserve and should be used when something needs to be changed from something and back again.