Verilog: Single Cycle Processor

Instructions: Now that we have a collection of processor components, we shall create our implementation of the MIPS 32 processor.

Note: Follow the class slides to make your processor!

Part 0:

If you have not already, pull down the latest from the course repository:

1. Open Bash on Ubuntu on Windows in your repository (make sure you are in your repository!)
2. If you haven't already, add upstream: 'git remote add upstream https://github.com/csu-cs/csci-330-spring-2017.git'.
3. To get the latest from the course do: 'git pull upstream master'

Part 1:

Setup the Instruction memory such that it will load your program during reset. It was easiest for me to use the $readmemh function of Verilog.

This has been done for you. What you should hook up first is incrementing the pc counter so that the cpu will halt when it reads the halt instruction.

Part 2:

Next decide on the order of instructions you wish to implement. It is probably best to start with the basic and then build up to the more complex. Here is a recommended ordering:

1. halt (to make sure your program will halt!)
2. lw (so you can load some data values into your register file.)
3. add
4. nor
5. noop
6. sw
7. beq

As you implement each instruction remember to test! We are not implementing multiply at this time.

Part 2:
Be sure to test some loops and other complex programs. I have provided one in the programs directory.

How to turn in:
Code shall be turned in via GitHub. Ensure the file(s) are in your lab04 directory and push via the command line:

- $ git add <files>
- $ git commit
- $ git push

Teamwork: No teamwork. You may use any resource you can find (except classmates), but please cite your resources.