Project 1 16-bit Processor

Objectives

The project requires you to design a simplified processor in Verilog. The project is broken up into three assignments. In the first assignment you will design the purely combinational modules of the processor. In successive assignments you will add the sequential logic components, and finally integrate all the components.

Before integration, each component will be tested independently. Testbenches to exercise these component modules are provided. When integrated, the processors should be able to execute assembly language programs written in its instruction set.

The instruction set of the architecture to be implemented is described in the document **processorDocs**. It details the set of instructions to be supported by the processor. The instruction format describes how each instruction is represented as a 16-bit machine instruction that the processor must decode and execute. The document **processorComponents** shows the input/output ports of the three *combinational* logic *modules* to be designed in this assignment. These are:

(i) **decoder** : the instruction decoder parses each 16-bit instruction and generates output signals to be used by other modules

(ii) **alu**: executes the arithmetic and logic instructions on its input operands and outputs the results

(iii) **branch**: checks the conditions of branch instructions and generates a branch/no-branch output

These must be implemented in the above order, since testing a module requires a working earlier module. In addition, the **alu** and **branch** modules require inputs from a register file module (called **register**). A skeletal implementation that is sufficient to drive your modules in this assignment is provided. In the next assignment you will implement a real register file to replace the mock-up used here.

The three modules to be designed in this assignment are described in the file **moduleDocs.**

Assignment

Verilog stubs for the three modules to be implemented, testbenches, and the mock-up register module can be found in the tar file. It is suggested that you make separate directories for each module that holds all the files relevant to that module. See the README file on instruction to compile and test your modules.

For submission upload the three verilog files (one per module) to Canvas. Please do not change the file names. Within each skeletal module provided do not change any of the existing code in any way.

Due Date: Wednesday October 7, 11:59pm for all three modules