Assignment 1

Due date: 11:55 PM, Friday, September 28th. 2018

Purpose:

Generate a detailed understanding of the operation of the MIPS pipeline. This is intended to solidify your understanding of the fundamental concepts of

I. Data dependencies between instructions and how they can interact in the pipeline to produce hazards

II. Control flow and its impact on the operation of thepipeline.

Part I: Understanding the Model

This part does not have a submission requirement. The warm-up can be discussed with your classmates. It is intended to help you get up to speed quickly and to familiarize yourself with the model, and compile and execute a few simple programs. Start by studying the trace and execution of the program that is already in instruction memory and determine if the program is correctly executed. Check the documentation associated with the model. The model support lw, sw, beq and the set of r-format instructions from the text and class notes.

- Is this program execution correct? If not, why not?
- The datapath does **not** provide hardware support for correctly executing branches. How many delay slots does the pipeline have?
- What is the initial value of the stack pointer?
- What value is placed on the read output of memory when the data memory is not being read?
- If ALUSrc is undefined, what is the value of the BInput of the ALU in the execute stage.
- How large (in bits) are the EX/MEM and IF/ID pipeline registers?
- If the opcode is 0x00 the effect is a nop. Look at *ps_control.vhd* and convince yourself you understand why this is produces a nop.

Part II: Pipeline Functionality (100 pts)

This assignment is comprised of the following steps. **You may discuss solutions with colleagues. However the coding and testing must be performedindependently!**

1. Add support for the *load-to-use* stall. For example, if a lw instruction produces data used by an immediately following instruction, one stall cycle should be included between the lw and the immediately following instruction.
2. Implement data forwarding to the EX stage.
3. Implement flushing in the pipeline. Assume branches are predicted as not taken. Thus, the fetch stage will keep fetching instructions through PC+4 until the branch condition is resolved. Note that if the branch is taken, then the instructions which were speculatively fetched would have to be flushed.

Test cases for both load-to-use stall and data forwarding will be given a few days before the assignment is due.

Grading Guidelines

Partial credit is evenly distributed among the three functional additions.

1) Compiles and executes: **20** pts.
2) Execution has correct output: **60** pts.
3) Demo your code to the TAs: **20** pts.

Submissions Instructions

Code: In a single zip file submit (lastName_firstName.zip)

The complete VHDLcode for the modified data path. It should be in a form whereby all that needs to be done is compiling & execution (as in the case of the model that was provided to you).

Document your code:

a. Include your name/GTID in each VHDL module (even the un-modified ones) in the header.
b. At the top of each VHDL module, include brief comments describing changes you made within that module.
c. All changes you make should be documented with comments.

Create a zip file: (Not tar or any other format please!) with ALL VHDL modules in your project and submit. This zip file should contain all the VHDL code for the modified data-path. This means that you submit **all** VHDL modules in your project. You may have modified some, and others may have been unchanged, but you should submit **all** the VHDL modules (including the unchanged ones) that your ModelSim project used. Name the zip file *first-name_last-name_A1_vhdl.zip*

Programming submissions should be electronic. Submissions must time stamped by the duedate and time. Submissions will be via Canvas.

Part III: Extra Credit (20 Points)

Update the code from II.3 and add a 2-bit branch predictor.

Also write your own test case (with at least 4 branch instructions that show the change of predictor states).

Submit this part as a separate zip file: *first-name_last-name_A1-XC_vhdl.zip* (otherwise your main submission will not pass the test cases for II.3 which uses a simple Not Taken branch predictor).

Points will be provided only if you can successfully demo the code with your test case to the TAs.