CPSC 313: Computer Hardware and Operating Systems

Assignment #4

Due: Sunday, March 16, 2014 at 22:00

Introduction and Objectives

In this assignment, you will implement data forwarding and branch prediction for the `Y86-Pipe` implementation of the CPU. The objective of this assignment is to give you a taste for the challenges that arise when we implement data forwarding and branch prediction in a CPU. We recommend that you work with a partner on this assignment as you did in the previous assignments.

Problems

1. Implement `Pipe` with data forwarding and jump prediction as described in class and in the textbook: register–register data hazards should result in zero stalls, load-use hazards should result in one stall, ret instructions should result in three stalls, and you should predict that jumps are taken (and shoot instructions down when it is discovered that the guess was incorrect).

   Implement your solution by modifying the class
   `arch.Y86.machine.pipe.Student.CPU`.
   Your coding can be confined to the following helper methods (although you are allowed to make changes to any method you want):
   `pipelineHazardControl`,
   `fetch_SelectPC`,
   `fetch_PredictPC`, and
   `decode_ReadRegisterWithForwarding`.
   Clearly indicate the code you added or changed, using comments in the code.

2. Use `pipe-test.s` to test your solution. The main method for this implementation is in the class `SimpleMachine$Y86PipeStudent` (in the default package). For your solution to be correct it must execute programs correctly and with the appropriate number of stalls. Document the results of each test by indicating whether or not your code passed the test. When your solution runs the `pipe-test.s` should produce the same values in the registers and memory your implementation for assignment 3 did, but

it should use less stalls and bubbles. Make sure that your implementation produces the right number of stalls and bubbles as described in the notes and the text.

Note: An obfuscated version of the Pipe simulator is provided in the **`SimpleMachine313Pipe.jar`** file. Download and run the `pipe-test.s` file to see how a correct implementation should handle the various hazards.

3. Repeat question 2 from the previous assignment for your Pipe implementation. That is, use the cCnt and iCnt processor registers to document the pipeline efficiency for executing the programs sum.s and max.s included with the assignment. The cCnt field is incremented once per clock cycle and the iCnt field is incremented whenever an instruction other than a bubble is retired. Present pipeline efficiency as average number of cycles per instruction (CPI) for each program.

Deliverables

Only one person in each group will submit the assignment using the department's handin tool.

To submit the assignment you should create the directory **`a4`** inside the `cs313` directory, and place in it the following files:

1. A copy of the text file `coverpage.txt` filled with the information of both partners
2. The completed `CPU.java` file (with comments).
3. A file in either text or PDF format that contains the following information:
   - A description of the results for the 8 tests in file `pipe-test.s`, including the number of stalls you observed, and whether or not the test produced the result you expected.
   - Your CPI results from question 3.

When all the files listed above are in the directory `~/cs313/a4`, to submit your assignment just type the following command at the unix prompt:

```
> handin   cs313   a4
```

`Last Updated: 11/03/2014 6:14 PM`