CPSC 313: Computer Hardware and Operating Systems

Assignment #3

Due: Monday, Mar 3, 2014 at 22:00

Introduction and Objectives

This assignment is all about the Y86 pipeline. You will work with an implementation of the pipeline: `Y86-PipeMinus`, which handles hazards by stalling (in the following assignments, you will work with `Y86-Pipe`, which handles hazards using data forwarding and jump prediction).

The objective of this assignment is to help you read and understand the pipeline control logic for stalling. You will achieve this by finding and fixing two bugs in the `Y86-PipeMinus` implementation of the CPU provided with the assignment. The `PipeMinus` implementation is in the class `Arch.Y86.machine.PipeMinus.Student.CPU`. The main method for this implementation is in the class `SimpleMachine$Y86PipeMinusStudent`.

Problem

The `Arch.Y86.machine.PipeMinus.Student.CPU` class we provided you with has two bugs. These bugs were introduced by simply deleting two pieces of code. No other changes were made. The bugs are in the method `pipelineHazardControl`.

1. Use the provided program `pipe-test.s` (inside the `code.zip` file) or your own test programs to identify the symptoms of the bugs. To find a bug, first write down the values that the registers should have at the end of one of the tests. Run the simulator on that test, and then compare the values actually stored in the registers. If they differ from the values you expected, you may have found a bug. To run a test, double-click on the address of the first instruction of the test (this should set the value of the pc register in the `Fetch` stage to this address), and then click on `Run` or `Step` as usually.

   Carefully describe the erroneous execution and explain your theory for what is happening. When you find a bug, describe it carefully, fix it in the code, and describe your solution. Then re-run the test to demonstrate that you correctly fixed the bug. The descriptions you provide of the symptom, theory, bug and solution are as important as fixing the bugs.

Note: An obfuscated version of the PipeMinus simulator is provided in the **`SimpleMachine313PipeMinus.jar`** file. Download and run the `pipe-test.s` file to see how a correct implementation should handle the various hazards.

2. Use the `cCnt` and `iCnt` processor registers to document the pipeline efficiency for executing the programs `sum.s`, `max.s` and `selection-sort.s` which are included in the `code.zip` file. 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 **`a3`** 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, the hours each partner spent on the assignment and any acknowledgements you may have.
2. The corrected CPU.java file (with comments).
3. A file in either text or PDF format that contains the following information:
   - For each of the two bugs you were asked to find, a description of the symptoms, an explanation of what was wrong with the code in the CPU.java file, and one sentence describing your solution.
   - Your CPI results from question 2.

When all the files listed above are in the directory `~/cs313/a3`, to submit your assignment just type the following command at the unix prompt:

```
> handin   cs313  a3
```

Last Updated: 19/02/2014 10:13 PM