CPSC 313: Computer Hardware and Operating Systems
Assignment #3, due Due Friday, May 25 at midnight. Late penalty of 20% per day for up to 3 days.

Introduction and Objectives

This assignment is all about the `Y86` pipeline. You will work with an implementation of the pipeline: `Y86-PipeMinus`, that handles hazards by stalling (in the following assignment, you will work with `Y86-Pipe`, that 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 three 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`.

Source code for this assignment is provide in code.zip on the course web page.

Problem

The `arch.y86.machine.pipeminus.student.CPU` class we provided you with has three bugs. These bugs were introduced by simply deleting a section of code. No other changes were made. The three bugs are all in method `pipelineHazardControl`, or in its helper `isDataHazardOnReg`.

1. Use the provided program `pipe-test.s` 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 usual.

   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.

   Please run the tests in the order in which they appear in file `pipe-test.s`, and fix each bug before moving on to the next test (some of the later tests might fail not only because of the bug they are meant to reveal, but also because of those revealed by earlier tests).

2. Use the `cCnt` and `iCnt` processor registers to document the pipeline efficiency for executing the programs `sum.s`, `max.s` (both included with the simulator's source code)

and `heapsort-student.s` (your completed version from assignment #1; if you could not get it to work, you can copy a friend's solution). 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

You should use the `handin` program to submit the first part of this assignment. The assignment name is `a3`, and the files to submit for this part are:

1. The corrected `CPU.java` file (with comments to indicate the code you added).
2. A file in either text or PDF format that contains the following information:
   - Your name and student number.
   - For each of the three 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.
   - How long it took you to complete this assignment (not including any time you may have spent revising before starting to work on it).

Each bug found, explained and corrected will be worth 9 marks, the CPI results will be worth 2 marks, and the amount of time you spent on the assignment 1 mark, for a total of 30 marks.