CSEP548: Computer Architecture
Assignment 1
Due: Wednesday, April 20

The purpose of this assignment is to supplement your knowledge of the design of dynamic branch prediction schemes with intuition about their relative performance. To evaluate the performance of different prediction schemes, you will be using the sim-outorder simulator that is part of the SimpleScalar tool set. Sim-outorder is an instruction-level simulator that implements many of the architectural features we will study this quarter. But for this assignment, we will use it as though it looks very much like a simple processor that executes one instruction at a time, in the order that they were fetched.

Here's what the assignment involves:

1. Pick an application from the application directory to instrument. All these programs are taken from the SPEC95 benchmark suite, which was the industry-picked standard workload for architecture research two suite-generations ago (we are now on SPEC2000). They have already been precompiled for sim-outorder and you can use them as input to the simulator. Douglas will send you email if it turns out that any of the applications are not appropriate for this assignment or that one is particularly good.
2. Set sim-outorder configuration parameters to reflect a computer that has the following configuration:
    - a pipeline that fetches, decodes, issues, executes and commits one instruction/cycle, no matter what the instruction type
    - only one of each type of functional unit
    - 8KB, two-way set-associative L1 instruction and data caches with 32 byte blocks
    - a 256KB, direct-mapped L2 unified cache with 32 byte blocks
    - an 8-way, 128-entry data TLB
    - a 4-way, 64-entry instruction TLB
    - All caches and the TLB have an LRU block replacement policy.
    - The page size is 4KB.

    For your experiments, vary the branch prediction strategy between:

    - static backward branches taken and forward branches not taken prediction
    - 2-bit dynamic branch prediction
    - correlating branch prediction, using the default number of history bits and pattern history table size

This means you will have separate simulations for the three branch prediction techniques.

SimpleScalar determines the branch target address for both branch prediction schemes with a branch target buffer.

3. Generate and analyze your results. In particular, address the following issues:
   - Branch frequency.
     - How often are branches executed in your programs?
     - How does this compare with the "average" frequency of every 4-6 instructions?
     - If there is a difference, do you have an hypothesis to explain it?
   - Branch characterization.
     - For conditional branches, record the frequencies of each element of the Cartesian product: (branch *forward*/*backward*) x (branch *taken*/*not taken*). You can express your answer in a table like this one:

	taken	not taken
forward	#	#
backward	#	#

     Do you need to generate additional stats to do this? If so, do so.

     - Discuss whether or not your results support the use of the traditional static scheme of backward branches taken and forward branches not taken.
   - Prediction accuracy.
     - Record the number of correct predictions for all branch prediction schemes. Do the results for the static scheme confirm your branch characterization results?
     - Which prediction scheme has the most correct predictions? Why? If you need to run additional simulations to justify your analysis and conclusion, do so.
4. Is there a difference between the number of instructions fetched and the number of instructions committed? Why or why not?
5. Is there a limit to the number of useful history bits, or is more history always better? Perform simulations to determine what the situation really is. What do you think explains your data?

   Keep in mind that when doing a sensitivity analysis of this kind, you should keep all factors constant except the one you are varying.
6. Write up your experiments, the results and an analysis of the results in a report, as outlined in the report handout and illustrated in the sample report. In the results section, devote a different subsection to each of the issues listed above in items 3 through 5. Use tables and graphs to illustrate the results.
7. In addition to the experimental portion of the assignment, answer the following question. For a (3,2) correlated prediction scheme which has a global history register and one pattern

history table, show the contents of the history register and the PHT after the following sequence of branch executions have taken place: 1 not-taken branch, 2 taken branches, 2 not- taken branches, 2 taken branches, and 2 not-taken branches.

- The prediction bits in the pattern history table use 2-bit prediction, implemented with saturating counters. The counters are incremented when a branch is taken and decremented when a branch is not taken. Each counter is initialized to the weakly taken state, as illustrated on the lecture slide for 2-bit branch prediction. Assume this state has the value 10.
- The global history register is initialized to zero.
- The addresses of the ten branch instructions are: ffff0000, ffff0001, ffff0010, ffff0000, ffff0000, ffff0000, ffff0011, ffff0010, ffff0000.

For this assignment I'd like you to work in teams of 2 people. (By the way most assignments will involve 2-person teams. For each one, try to work with a different partner, so begin planning ahead for your partners.) Turn in one report per pair and remember to put both partners' names on the report. If you decide to divide up the work, say who did what.