CSE471: Computer Design & Organization
Assignment 2
Due: Thursday, April 19

The purpose of this assignment is to supplement your knowledge of the design of different dynamic branch prediction schemes with intuition about their relative performance.

You are expected to work in teams of 2 people, hopefully with a different partner than the last assignment.

1. For this assignment your applications should be the two applications *gcc2000* and *gzip2000* from SPEC2000 with respective inputs: *integrate.i* and *input.source*.
2. Simulate the same microarchitecture as in the first assignment, but vary the branch prediction strategy between:
   - 2-bit dynamic branch prediction with 1K counters (the bimod option)
   - correlating branch prediction GAg, using a number of history bits and pattern history table (PHT) size such that the PHT requires the same amount of storage as the 2-bit predictor above
   - a gshare predictor using also the same amount of storage (setting the xor bit in the configuration parameters of the predictor should do the trick)

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

   FYI: 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?
     - If there is a difference between the two programs, do you have an hypothesis to explain it?
     - How does this compare with the "average" frequency of every 4-7 instructions?
     - If there is a difference from the average, do you have an hypothesis to explain it?
   - Branch characterization.
     - For conditional branches, record the frequencies of each element of the Cartesian product: (*forward*, *backward*) x (*taken*, *not taken*). You can express your answer in a table like this one:

	taken	not taken
forward	#	#
backward	#	#

Do you need SimpleScalar to generate additional stats to do this? If so, write the code to do so.

- Prediction accuracy.
  - Record the number of correct predictions for all branch prediction schemes, including what the static schemes would have achieved (for the static schemes, you do not have to run a simulation, just use the branch characterization results). In particular, discuss whether or not your results support the use of the traditional static scheme of backward branches taken and forward branches not taken
  - Which prediction scheme has the most correct predictions for each program? Why? If you need to run additional simulations to justify your analysis and conclusion, do so. (Don't be overly surprised if your results do not coincide with conventional wisdom since you are executing a limited number of instructions at the beginning of an application).
  - This question is only applicable if each of your programs performed best with a different branch prediction scheme. If you had to pick only one prediction scheme that would be used for both programs, which would it be and why?

4. Is there a limit to the number of useful counters and/or history bits, or are more counters/ history always better? Perform simulations to determine what the situation really is. Consider all three cases: bimodal, GAg and gshare. Limit yourself to about one order of magnitude more storage than the base case. 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.

5. 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 reports. In the results section, devote a different subsection to each of the issues listed above in items 3 and 4. **Use tables and graphs to illustrate the results.**

6. 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 taken branch, 2 not-taken branches, 2 taken branch, 2 not-taken branches, and 2 taken branches. The PHT should be shown as updated after the execution of the last branch.
   - 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 in your book and the lecture slides. Assume this state has the value 10.
   - The global history register is initialized to zero.
   - The addresses of the nine branch instructions are: ffff0000, ffff0001, ffff0010, ffff0000, ffff0000, ffff0000, ffff0011, ffff0010, ffff0000. … but do you need this information?

7. Extra credit only, meaning you don't have to do this if you prefer not to:

After getting a sense of how prediction correctness varies with changes in the size of the hardware data structures (cf. Question in paragraph 4), the question that microarchitects usually face, is:

> "If I have a certain amount of chip area that I can devote to branch prediction, how should I divide it among the various branch data structures for correlating branch prediction, the global history registers, the pattern history table, (and the branch target buffer)?"

For extra credit, you can try to answer that question for a bit budget of 2K bits plus a few extra

bits (say, less than 64) for the branch prediction only (i.e., not the branch target buffer). In essence, you have to come up with a design SAs where S is the number of history registers and s is the number of PHT tables of adequate sizes.