CSE 586: Computer Architecture
Assignment 2
Due: Tuesday April 22

The purpose of this assignment is to (1) supplement your knowledge of the design of today's dynamic branch prediction schemes with intuition about their performance relative to simpler techniques, and (2) teach you how to write academically-oriented technical reports.

You are expected to work in teams of 2 people.

1. For this assignment you will use the Blis simulator and the environment in which it executes. Blis is an instruction-level simulator that implements many of the architectural features we will study this quarter. It executes the Alpha instruction set architecture (i.e., Alpha binaries) on a pipelined, out-of-order superscalar that closely models the later Alpha implementations.

2. Blis simulates a machine whose default configuration parameter values appear in ~/Configurations/standard4wide.cfg. You can change all default parameters by editing this file Make sure that the configuration parameters reflect a computer that has the following configuration:

   - a 7-stage pipeline that fetches, decodes, register-renames, issues, register-accesses, executes, and commits up to four instruction/cycle. Memory operations will take 8 stages.
   - 4 integer units, 1 floating-point unit.
   - a memory subsystem with:
     - a 64KB, four-way set-associative L1 instruction and data caches with 32 byte blocks.
     - a write-through L1 data cache.
     - no L2 cache.

   The rest of the parameters should be left with their default values.

3. Your application in this assignment, i.e., the input to the simulator, should be `pearl`; Evan will tell you where `pearl` lives and what its input should be. `pearl` was taken from the SPEC2000 benchmark suite, which is currently one of the standard workloads for architecture research. Even will send you email with the number of instructions you should simulate.

4. Simulate the microarchitecture above, but vary the branch prediction strategy between:

   - the traditional static scheme of backward branches taken and forward branches not taken.
   - 2-bit dynamic branch prediction.
   - 2-level correlated branch prediction, the gshare flavor, using the default number of history bits, pattern history table size, prediction bits, branch address bits and overlap (which are all found in ~/Configurations/standard4wide.cfg).
   - a tournament predictor, in which the 2-bit scheme is the local predictor and gshare is the

global predictor.

Gshare is the branch prediction mechanism implemented in Blis; there is a rumor that 2-bit is also implemented, but incorrectly. You will have to implement the others yourselves. This means you'll have separate simulations for the branch prediction techniques. For all branch prediction mechanism, use the default branch target buffer.

5. Generate and analyze your results. To do this, you may need to add metrics to the simulator. 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 a 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 the following table:

	taken	not taken
forward	#	#
backward	#	#

     - Discuss whether your results support the use of the static branch prediction algorithm or not.
   - Prediction accuracy.
     - Record the number of correct predictions for all branch prediction schemes.
     - Which prediction scheme has the most correct predictions? Why?

6. Is there a difference between the number of instructions fetched and the number of instructions committed? If so, how much and why? If not, don't bother with this question.

7. Comparing alternate gshare branch predictor implementations.

     - Change some of the default configuration parameters of the gshare predictor to improve its performance. Is there a limit to the improvement, i.e., are larger sizes always good?
     - Which of the parameters you changed has more effect on the quality of the predictions? Any hypotheses why?

8. Write up your experiments, the results and an analysis of the results in a report, as outlined in the report handout. In the results section, devote a different subsection to each of the issues listed above in item 3. Use tables and graphs to illustrate the results.