Winter ’12 CIS 429/529 Program 2 – 110/100 points possible – Due Monday, 2-13, 11:59 PM

For this programming assignment you will implement a suite of branch prediction simulations in a programming language of your choice. You will run each branch prediction simulation using two data sets: gcc_cbr_trace and go_cbr_trace. Each line of these data sets contains a relative displacement to a 32-bit Program Counter (PC) followed by a space followed by a 1 if the branch was taken and a 0 otherwise. Your simulations will assume a starting PC of 0. As each line of the trace is processed, the PC should be updated by adding the current displacement. The branch predictor will then make a prediction, which you will check against that actual branch result in the trace.

1. [10] Implement a driver class for your simulations. This class should take a trace filename as a command-line argument, load and parse the trace file, and then run all available branch prediction simulations on that trace. For each simulation, keep track of the total number of branches and the number of branch mispredictions so that your program can compute the misprediction rate.

2. [10] Implement a “never taken” predictor which always predicts that the branch will not be taken.

3. [10] Implement an “always taken” predictor which always predicts that a branch will be taken.

4. [10] Implement a one-bit predictor which keeps track of the most recent branch result and predicts that the current branch will have the same result. Assume that the predictor initially predicts that branches will not be taken.

5. [10] Implement a two-bit saturating predictor like that depicted in P&H Figure C.18 (p.C-28). The predictor requires two consecutive mispredictions before changing state (i.e., if the predictor is predicting that branches will not be taken, two consecutive branches must be actually taken before the predictor will predict that the next branch will be taken). Assume that the predictor initially predicts that branches will not be taken.

6. [15] Implement a local predictor which uses the \( n \) low-order bits of the PC >> 2 (i.e., because the PC is specified in bytes but we’ll assume instructions are stored as 4-byte words) to select one of \( 2^n \) two-bit saturating predictors to make the prediction and update with the actual branch result. Assume that the predictors initially predict that branches will not be taken. Hint: Be careful when using bitwise shift right on signed integers (i.e., shift right logical versus shift right arithmetic).
7. [15] Implement a \((n, 2)\) global correlating predictor which concatenates the results of the last \(n\) branches onto the \(m\) low-order bits of the PC \(>>\) 2 to select one of \(2^(n + m)\) two-bit suturing predictors. For example, consider a \((4, 2)\) global predictor which uses the 8 low-order bits of the PC \(>>\) 2; if the 8 low-order bits of the PC \(>>\) 2 are 1111 0000 and the last four branch results are 0001 (i.e., three branches not taken followed by a branch taken), the index of the predictor to be selected would be 1111 0000 0001, or 3841. After making a prediction, update the selected predictor with the actual branch result, then shift the global history left 1 and use the branch result as the new low-order bit of the history. In other words:

\[
\text{history} = (\text{history} << 1) \mid \text{result}
\]

Assume that the global history initially contains all zeros and that the predictors initially predict that branches will not be taken.

8. [20] Collect misprediction rate data for each trace using the following predictors: not taken, taken, one bit, two bit, local using 10 PC bits, local using 12 PC bits, \((2, 2)\) global using 8 PC bits, \((4, 2)\) global using 6 PC bits, \((2, 2)\) global using 10 PC bits, and \((4, 2)\) global using 8 PC bits (i.e., for a total of 10 predictors). Prepare a report aggregating and presenting your misprediction rate data using tables and/or figures. Your report should include written analysis and conclusions of your collected data. You may optionally add more data points to strengthen your arguments.

9. [+10] (Extra credit) Implement a tournament predictor which uses a two-bit saturating predictor to select between two distinct predictors. Experiment with different types of predictors to find configurations which improve overall performance. Analyze your results along with your results from part 8.

I’m including my output to aid in debugging, but note that your output does not need to match mine for full credit; you output need only be consistent with your own stated assumptions.

The following is my output from go_cbr_trace for use as validation:

read 2000001 lines.

evaluating: Not-Taken Predictor
total branches: 2000001
mispredictions: 1058771
misprediction rate: 0.5293852
evaluating: Taken Predictor
total branches: 2000001
mispredictions: 941230
misprediction rate: 0.47061476

evaluating: 1-Bit Predictor
total branches: 2000001
mispredictions: 956027
misprediction rate: 0.47801325

evaluating: 2-Bit Predictor
total branches: 2000001
mispredictions: 941269
misprediction rate: 0.47063425

evaluating: Local Predictor 1024 entries
total branches: 2000001
mispredictions: 536896
misprediction rate: 0.26844788

evaluating: Local Predictor 4096 entries
total branches: 2000001
mispredictions: 495365
misprediction rate: 0.24768238

evaluating: Global Predictor (2, 2) 1024 entries
total branches: 2000001
mispredictions: 524024
misprediction rate: 0.26201186
evaluating: Global Predictor (4, 2) 1024 entries
total branches: 2000001
mispredictions: 556644
misprediction rate: 0.27832186

evaluating: Global Predictor (2, 2) 4096 entries
total branches: 2000001
mispredictions: 467452
misprediction rate: 0.23372589

evaluating: Global Predictor (4, 2) 4096 entries
total branches: 2000001
mispredictions: 456060
misprediction rate: 0.22802989