Homework 2

Instructions:

- Please type your solutions into a document and convert it into a PDF file. Your solution document should contain your name, student ID, the course name, and homework number. Please submit your solution PDF via Canvas.
- Make reasonable assumptions where necessary and clearly state them!
- You may discuss concepts with your classmates. This fosters group learning and improves the class' progress as a whole. However, make sure to submit your own independent and individual solutions.

Problem 1

In this assignment, you will explore the effectiveness of branch direction prediction (taken vs not taken) on an actual program. We've used Pin, a binary instrumentation tool, to generate a trace of branches and their outcomes. Your task will be use this representative trace to evaluate the effectiveness of a few simple branch prediction schemes. To do this, you'll write a program that reads in the trace and simulates different branch prediction schemes and different predictor sizes. You can write the program in whatever language you like; although we are going to ask for a printout of your source code, to end result of this assignment are the experimental results (and not the program source code).

The trace we're giving you is a trace of 16 million conditional branches. These are the conditional branches from an execution of the program GCC (Gnu C Compiler) from the SPECint2000 benchmark suite. As unconditional branches are always taken, they are excluded from the trace. Each line of the trace file has two fields. Below are the first four lines of a trace file:

```
3086629576 T
3086629604 T
3086629599 N
3086629604 T
```

The first field is the address of the branch instruction. The second field is the character "T" or "N" for branch taken or not taken. **The trace is provided on Canvas under file directory.**

1. A) Assume 1K (1024) entry for Pattern History Table (PHT) with 2-bit Counter per entry (predictor index size of 2 bits), implement and compare the mis-prediction of the following branch predictors:

    1. One level branch Prediction
    2. Two Level Global Branch Prediction
    3. Two Level Gshare Branch Predictor

4. Two Level Local Branch Prediction (assume 128 entry for Local History Register Table)

You should able to come up with the bit-width of local and global history registers. Although initialization doesn't effect the results in any significant way, your code should initialize the predictor to "strongly not taken".

Plot the mis-prediction results across branch predictions and analyze the achieved results. Which one is the most efficient branch predictor (50 pts)?

1.B) Repeat the Problem 1.A, this time with variable entry size (n-bit Counter) with size of 2 bits, 3 bits, 4 bits, 5 bits, ... 20 bits. Generate a dedicated line plot of the data using MS Excel or some other graphing program for each branch predictor. On the y-axis, plot "percentage of branches mis-predicted". On the x-axis plot the the predictor size (basically, the width of n-bit counter). Draw a separate plot for each branch predictor. Analyze the data and argue the effect of increasing counter bits on branch predictor performance. In your view how many bits seems sufficiently enough for branch predictor (50 pts)?

Extra Credit on Quizzes (10pts)

Implement a branch predictor that combine the benefits of Gshare and Two Level Local Branch Predictor. Plot your result and compare it with previous branch predictor. Make sure to argue and analyze you achieved results.