Mahindra University, Spring 2023
EE507 Advanced Computer Architecture
Branch Prediction Programming Assignment
Submission Deadline: 10/03/2021

Section I: Introduction

In this assignment, you are required to implement several Branch Prediction algorithms and analyze their effectiveness on real-world program traces. For this purpose, you will write programs **(C/C++/Python)** to read in the trace and simulate different branch prediction schemes and determine the prediction accuracy.

Information about the traces are given in Section II. The algorithms you required to implement are listed in Section III. Some of the algorithms are mandatory to implement for all students. Apart from this, students have the liberty to explore research papers to implement algorithms to take the prediction accuracy as close to an ideal Oracle predictor as possible i.e. 100% prediction. A leaderboard will be created and shared to keep track of the best algorithm that the class has designed and implemented. This is described in Section IV.

Section II: About the Traces

The **traces** you have been given are generated from runs of real-world programs. They only contain the dynamic instances of conditional branches alone. 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:

```
0x40fc96 1
0x40fc96 0
0x40fcb9 0
0x40fcf2 1
0x40fcf2 1
```

The first field is the address of the branch instruction. The second field is "1" if the branch was taken during the execution, or "0" for branch not being taken. There are a total of **16** traces. *If you are going to use any ML algorithm for branch prediction, you can split the traces into training and testing data and perform cross-validation.*

The datasets have been taken from the following two links. The traces have been standardized to follow the format given above. You can peruse them links for more information and some source codes for reading the files.

- https://www.cis.upenn.edu/~milom/cis371-Spring09/homework/hw2/
- https://github.com/linbinyang/BranchPredictor-Using-Perceptron

Section III: The Branch Prediction Algorithms

For each of the following branch prediction algorithms, you will have to determine the prediction accuracy, the hardware/software requirements, and the kind of program constructs for which the algorithm will perform better. With this also mention the inferences about given each program traces.

1) Static ALWAYS TAKEN Branch Predictor (**Static-T**): All branches are predicted to be taken

2) Static ALWAYS NOT TAKEN Branch Predictor (**Static-NT**): All branches are predicted to be not taken.

3) Dynamic Last Taken Branch Predictor (**Dynamic-LT**): A branch is predicted as taken if it was taken in its last dynamic instance, and vice versa.

4) Dynamic BIMODAL Branch Predictor (**Dynamic-BM**): The simplest dynamic branch direction predictor is an array of **2^n two-bit saturating counters**. Each counter includes one of four values: *strongly taken* (ST), *weakly taken* (WT), *weakly not taken* (WNT), and *strongly not taken* (SNT).

   Prediction: To make a prediction, the predictor selects a counter from the table using the **lower-order *n* bits of the instruction's address** (its program counter value). The prediction is made based on the value of the counter (Taken for ST or WT, Not Taken for WNT or SNT). *Note that multiple instruction addresses will share the counter which will lead to inaccuracy.*

   Training: After *each* branch (correctly predicted or not), the hardware increments or decrements the corresponding counter to bias the counter toward the actual branch outcome (the outcome given in the trace file). As these are 2-bit *saturating* counters, decrementing a minimum counter or incrementing a maxed out counter should have no impact. Initialize the counter to any value. Owing to the large number of branches in each trace, the starting value should not matter.

   Analysis: Vary "n" between some reasonable values and check how the accuracy and the hardware requirements scale with n. Present prediction accuracy for each trace separately and cumulatively. Try answering whether the Bimodal Predictor will saturate in accuracy.

5) Dynamic BIMODAL Branch Predictor with Global Branch History (**Dynamic-GSHARE**): A Dynamic-GSHARE predictor (or gshare) is a more advanced dynamic branch predictor that uses the history of recently executed branches to predict the next branch. gshare uses a history register to records the taken/not-taken history of the last ***h*** branches. It does this by shifting in the most recent conditional branch outcome into the low-order bits of the branch history register. It them hashes the branch history and the PC of the branch when making predictions.

   Prediction. Whereas bimodal uses the lowest *n* bits of the program counter to index into the table, gshare uses the lowest *n* of the **xor** of the program counter and the history register. Except for this different index into the table, the prediction mechanism is otherwise unchanged from a bimodal predictor. In essence a gshare predictor with zero history bits is basically just a bimodal predictor.

   Training. The counter used for prediction (selected by history XOR program counter) is trained just as in a bimodal predictor.

Analysis: Since gshare also uses the Bimodal predictor, repeat the same analysis as for the Dynamic-Bimodal Predictor. Additionally, vary the size of the history register and perform the same analysis.

Section IV: Branch Predictor Competition

Apart from the 4 branch predictors given in Section III that you are required to implement and analyze, you will have to implement your own branch predictor. You can use any algorithm in literature. There are 1000s of publications on branch predictors. Any one of those can be chosen. However, you will have to present the prediction accuracy for the given instruction traces and also the hardware requirement of the algorithm. I have created another trace of around 10 million branches which I will use to evaluate your programs.

Section V: Submission Format

Your submission should be a compressed file (<10 MB) with the following contents:

a) Source code of all the predictors
b) Readme file on how to run the code.
c) Document listing the performance and analysis of all the branch predictors implemented. The document should be in the same format as this assignment sheet.

Link for submission will be posted on the course's discord server before the deadline.

PLEASE NOTE:

- **The code should be properly commented.**
- **I should not be required to edit anything in the code in order for me to run it.**
- **The trace files have to be taken as command line parameters.**