

# Assignment 2

ECE511

Due 09/26/2017

## 1 Objectives

For this assignment, you are going to dive into the micro-architecture level and change a part of the processor. You are going to learn about dynamic branch predictors, and implement the Gshare, YAGS and Perceptron branch predictors in gem5. You will evaluate the efficiency of each by comparing the IPC and branch prediction accuracy. You will be using the soplex and libquantum benchmarks, available on [the course website](#). For more information branch predictors for this assignment you need to take a look at [4]; be sure to review all the sources before attempting to implement the predictors. You have to use the default gem5 L1i, L1d, and L2 caches - not the ones from assignment 1; these should be restored when pulling the code for assignment 2.

## 2 Implementation

The files you will be modifying are located in `src/cpu/pred`. You will probably find it helpful to refer to some of the branch predictors already available in gem5, located in the same folder.

Each branch predictor has two files, a `.cc` and a `.hh` file, which have already been created for you - you will be modifying the files for the Gshare, YAGS and Perceptron predictors. Gem5 has already been set up to include the branch predictors you will be using, so you can focus on their implementation.


## 3 Bi-mode Branch Predictor

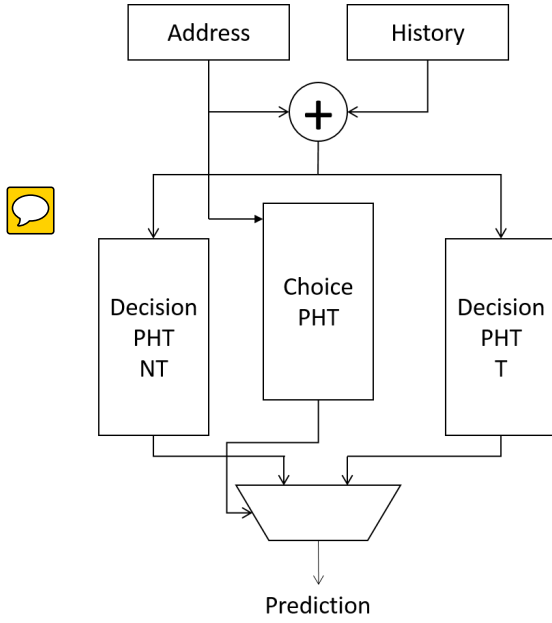
Before you implement the branch predictors that we mentioned earlier, you need to look at the Bi-Mode branch predictor inside gem5.

The Bi-Mode branch predictor (Fig. 1a) has 3 Pattern History tables (PHTs): choice, taken, and not taken. The choice PHT is indexed by the branch address, and the other 2 direction PHTs are indexed by the XOR of the branch address and the branch history register.

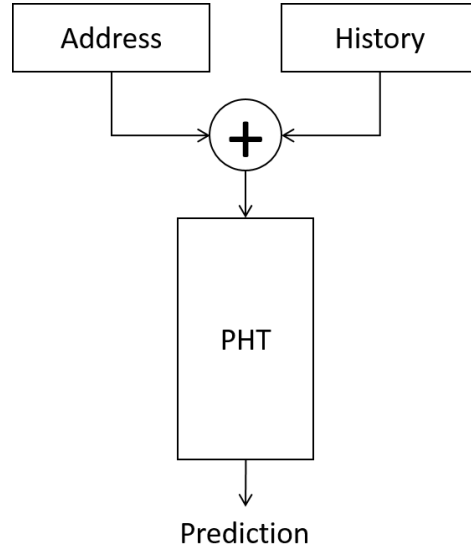
The choice PHT output chooses between the output of the taken direction PHT and the output of not taken direction PHT as the final prediction result. The branch outcome updates the choice PHT, unless the choice PHT output is opposite to the branch outcome but the selected direction PHT output matches the correct final outcome. At the same time, the branch outcome updates only the selected direction PHT.

## 4 Gshare Branch Predictor

The Gshare predictor (Fig. 1b) is a simplified version of the Bi-Mode branch predictor. It has no choice PHT, and only one direction PHT - the outcome of which is the final prediction outcome. You should set the PHT size to 2048 and 4096 for your report. 



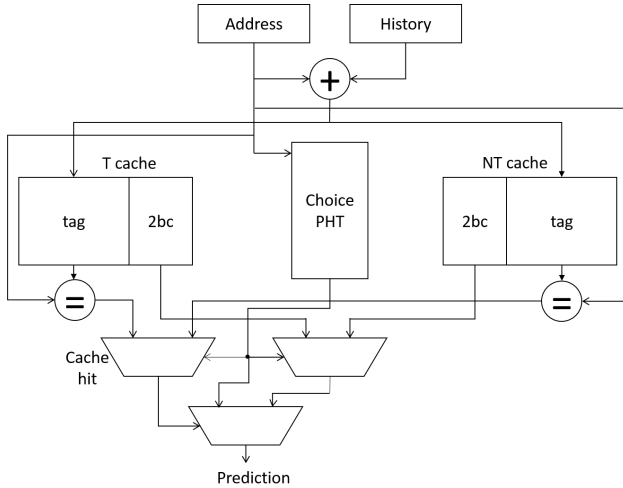
(a) Bi-mode branch predictor [1]



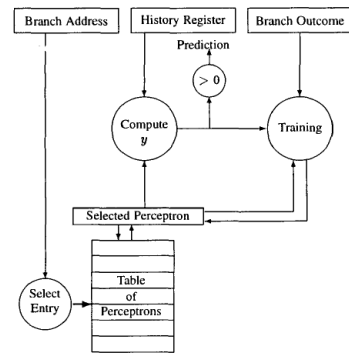
(b) Gshare branch predictor [4]

## 5 YAGS Branch Predictor

YAGS branch predictor (Fig. 2a) separates the taken and not-taken branch history tables. Similar to Bi-Mode branch predictor, the PHT is indexed by the branch instruction address. Instead of using masking, a tag is used - you have to use the same technique to determine the tag size as you would for a cache; this avoids aliasing. Each branch history table uses a 2bit pattern, similar to the PHT for each tag entry. You should set the PHT size to 2048 and 4096, and each cache to 1024 and 2048.



(a) YAGS branch predictor [2]



(b) Perceptron branch predictor [3]

## 6 Perceptron Branch Predictor

The Perceptron branch predictor (Fig. 2b) uses neural networks in order to make more accurate predictions. By using a simple neural network, one which can be efficiently implemented in hardware, the predictor can use a much larger history than other implementations [3].

Similar to the analysis done in [3], vary the length of the history to show the optimal length for the gem5 configuration - use a history lengths of 12, 24, and 48.

## 7 Running the simulation

Use the gem5 template available on the assignment2Template branch of gem5\_master.

After extracting the provided tar.gz file to the gem5\_master directory, change the chosen predictor and its configuration in configs/common/cores/arm/O3\_ARM\_v7a.py, at line 102 - where the O3\_ARM\_v7a\_BP class is declared. Then, run gem5 using the run script provided, rungem5.sh:

```
$ sh rungem5.sh [benchmark]
```

The available predictors are:

- bimode - included in gem5 by default
- tournament - included in gem5 by default
- 2bit - included in gem5 by default
- gshare - will not work until you've implemented it
- yags - will not work until you've implemented it
- perceptron - will not work until you've implemented it

The benchmarks you'll run for each are:

- soplex
- libquantum

**Be prepared for each run to take a significant amount of time.**

## 8 Submission

Write a report, including the IPC, and branch accuracy comparison between, Gshare, YAGS, Perceptron, 2bit local, Tournament, and Bi-Mode branch predictors. You have to use both graphs and tables to demonstrate the difference. Push your code to your repository within the assignment2 group on gitlab. If you have used any source code or reference for your coding they must be referenced in your report.

Overall, you will need to discuss results for:

- Bi-Mode Predictor
- Tournament Predictor
- 2-Bit Predictor
- YAGS Predictor, with the *four* different configurations listed above.
- Gshare Predictor, with the *two* different configurations listed above.
- Perceptron Predictor, with the *three* different configurations listed above.

## References

- [1] I-CK Chen Chih-Chieh Lee and Trevor N. Mudge.
- [2] Avinoam N. Eden and Trevor Mudge.
- [3] Daniel a. Jiménez and Calvin Lin. Dynamic branch prediction with perceptrons. *HPCA*, page 197, 2001.
- [4] Chris Feucht Matt Ramsay and Mikko H. Lipasti. Exploring efficient smt branch predictor design.