Branch Predictor Design

Chih-Hsuan Chen, 110550029

Keywords—Branch Predictor; BPU; One-level; Two-level; BHT; Processor; CPU; Aquila

I. Introduction

A branch predictor is a critical component in a pipelined CPU to reduce a large number of stall cycles. However, the different types of branch predictors will lead to different performance. The goal of this homework is to analyze the current branch predictor of Aquila, which is bimodal, and change it to a two-level branch predictor that takes global information into account to improve the performance of Aquila. All the tests and evaluations are based on CoreMark.

II. Pre-analysis

A. The Effect of the BPU

The CoreMark score of Aquila with BPU is 82.433299, while it is 74.259713 if disable the BPU in Aquila. This shows the importance of the BPU for the performance.

B. The Size of Branch History Table

The BHT in Aquila is implemented as a direct-mapping cache, and uses some bits ($log_2(num\ of\ entries)$ bits from the second bit) of the PC of the branch instructions as the index. If there are some different branch instructions have the same index, the stored information will be erased, which will lower the performance of predictions. Generally, a larger BTH can reduce such scenario since there are more entries to map the branch instructions. The original size of the BHT in the BPU in Aquila is 32, and the following is the comparison of the CoreMark score of Aquila with different BHT size.

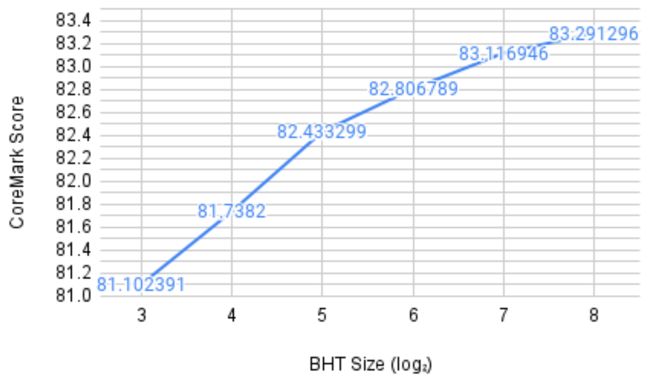


Fig. 1. The CoreMark score of Aquila with different BHT size

We can find that the larger BHT actually improves the performance. However, the impact decreases as the BHT size increases, which may imply that the size is already large enough. It seems that the default size of the BHT is a wise choice since it doesn't use too large space to store but also has a certain level of performance.

C. Statistics of Different Types of Branches

The following is some statistics of different types of branches:

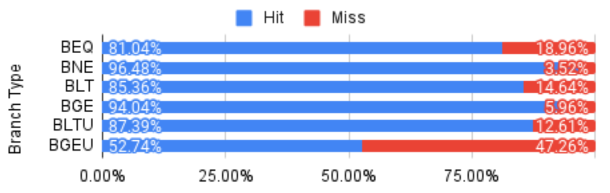


Fig. 2. The hit rate and miss rate of different types of branches

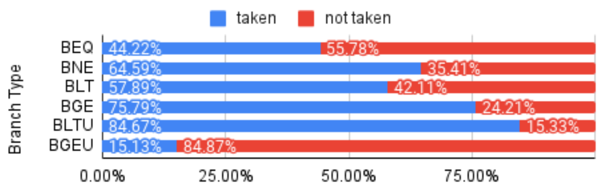


Fig. 3. The taken rate and not taken rate of different types of branches

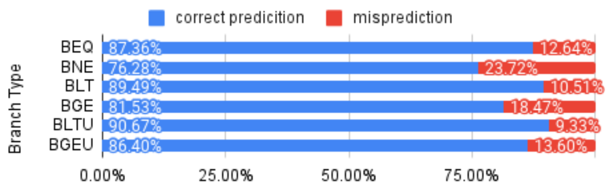


Fig. 4. The corretctness of the predictions of different types of branches

We can find that BGEU instructions have a particularly high miss rate, but they also have a particularly high not taken rate, which means that it doesn't have to branch when the instruction is BGEU most of the time. In the branch mechanism of the BPU in Aquila, if it is not a hit when the PCU searches from the BHT, the default decision is "not taken". This is highly probably the reason why the BGEU instructions can still have a relatively low misprediction rate.

The BEQ instructions have nearly equal numbers of times of branching and not branching and high hit rate; however, the correctness of the predictions is relatively not bad. This may imply that the default 2-bit predictors have a certain level of performance of BEQ instructions in CoreMark. On the other hand, the BNE and BGE instructions have the lowest miss rates but also have the highest misprediction rates, which may indicate that the default 2-bit predictors have poor performance of BNE and BGE instructions.

III. Implementation

I implemented three different types of two-level BPU from the paper "Combining Branch Predictors" by Scott McFarling. The two of them is "Global Predictor with Index Selection" and its slightly modified version from the spec PPT of this assignment, and the other is “Global Predictor with Index Sharing” (gshare). All of them use a single shift register GR to record the branch history of nearby branches and use the global information with different local information to index the pattern history table (PHT), whose entries contain 2-bit saturating counters.

A. *Global Predictor with Index Selection (PC)*

The local information used here is the PC address. The last n bits of GR and the last m bits of the PC address (except for the last two bits) are concatenated to be the index to find the corresponding 2-bit saturating counter (total 2^{n+m} entries).

B. *Global Predictor with Index Selection (Local History)*

Similar to the above branch predictor, but the information used here is local history. There is a local history table with size 2^{i} and uses the last i bits of PC address (except for the last two bits) as the index. Each entry of this table records the branch history of a branch instruction. The last n bits of GR and the last m bits of the local history are concatenated to index the pattern history table (total 2^{n+m} entries).

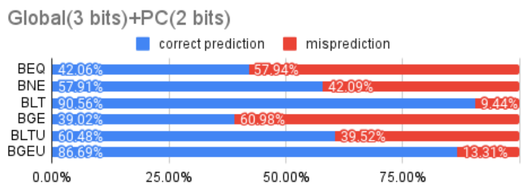
C. *Global Predictor with Index Sharing (gshare)*

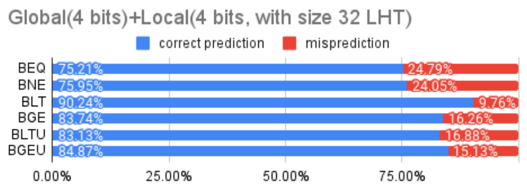
The local information used here is the PC address too, while the last j bits of GR and the last j bits of PC address (except for the last two bits) using XOR operations to generate the index of the 2-bit saturating counters (total 2^{j} entries).

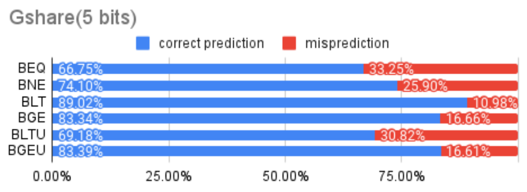
IV. RESULT

A. *Comparison of the Three Methods*

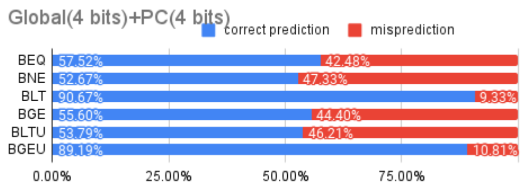
a) *5-bits to Index the PHT*

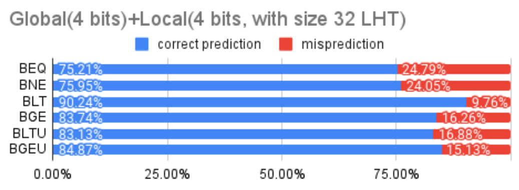


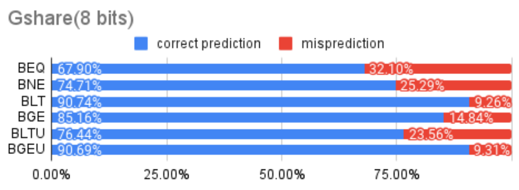




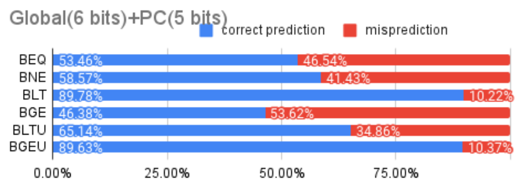
b) *8-bits to Index the PHT*

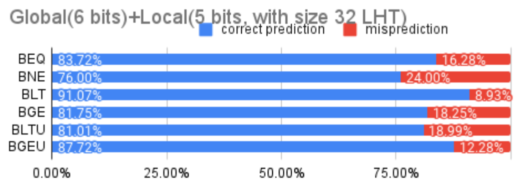






c) *11-bits to Index the PHT*





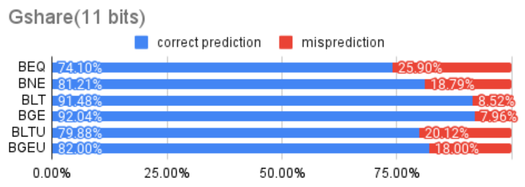


TABLE I. THE COREMARK SCORE

	Global + PC	*Global + Local (with size 32 LHT)*	*Gshare*
5-bits Index	74.954172	80.346442	79.012436
8-bits Index	76.322827	80.503955	79.665573
11-bits Index	75.231484	81.68572	80.84911

From Table I., it is obviously that the global predictor with local history performs best for it has the highest CoreMark score regardless the number of index bits.

We can also notice that the global predictor with PC address predicts the BLT and BGEU instructions far better than the other branches. It has extremely high, nearly 40%-50% of misprediction rate in every case when predicts the other branches.

On the other hand, the gshare predictor does best on BLT and BGE instructions, and the global predictor with local history predicts BLT instructions best

B. *The Number of Global Bits and Local Bits*

TABLE II.

	Global (8 bits) + Local (3 bits)	*Global (3 bits) + Local (8 bits)*	*Global (1 bits) + Local (10 bits)*
CoreMark	81.651522	82.088695	82.165647

Table II. shows the CoreMark score of different combination of global history bits and local history bits, while the total number of bits is fixed. We can observe that the more local history bits there are, the higher CoreMark score the predictor has. It seems that the information of local branch history is more important than global branch history.

C. *The Size of Local History Table*

TABLE III.

	Size 32 LHT	***Size 256 LHT***
CoreMark	82.165647	83.312676

Table III. shows the CoreMark score of the global predictor with different local history table size while the global and local history bits are fixed (1 global history bit and 10 local history bits). In theory, if the predictor has larger LHT, it can store more local branch history of different branch instructions, thus it can predict more precisely. And the real test result indeed agreed with this speculation.

D. *Comparison of the Final Result and the Original BPU*

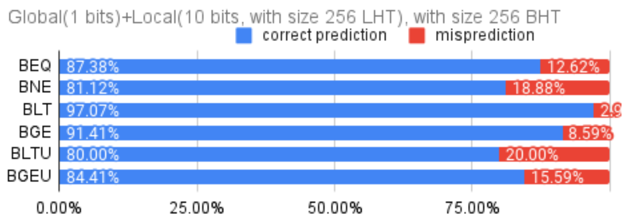


Fig. 5. The corretctness of the predictions from the final result predictor

TABLE IV.

	Final	***Original***	***Original (with size 256 BHT)***
CoreMark	84.173591	82.433299	83.291296

The final result uses a global predictor with 1 bit global history and 10 bits local branch history as the index for PHT, and the size of LHT and BHT are both 256. Table IV. Shows that this predictor has higher CoreMark score than the original predictor (bimodal predictor with size 32 BHT), and even the original predictor with 256 bits BHT.

However, the final result predictor needs much more space than the original one to store the branch history. There is a trade-off between space and performance.

V. Discussion

This assignment is aimed to implement a two-level predictor that considers the global branch history, and it expects that it performs better than the original bimodal one, which only considers the local information. However, the result shows that the final two-level predictor indeed has better performance, but it seems that rather than global branch history, the more bits of local branch history it has, the better performance it has.

But it is too early to draw the conclusion that the global branch history is useless now. It may have to further use another performance evaluate tool to test, and evaluate these branch predictors from more other aspects.

References

[1] RISC-V, ISA Specifications, vol. 1, December 2019

[2] S. McFarling, “Combining Branch Predictors,” WRL Technical Note TN-36, Digital Equipment Corporation, June 1993