Branch Predictor Project

A59005904: Junxian Qu (Teamate: Shangqing Gu)

ABSTRACT

This project is option 2, branch predictor. In this report, the working, advantages and disadvantages, and evolution history of three kinds of branch predictors are discussed in detail. And the three kinds of popular branch predictors, G-share predictor, tournament predictor and perceptron predictor, are implemented by C language based in docker environment. Finally, this report analyzes the result of implementation.

1. INTRODUCTION

When the processor does not have branch prediction, all programs have to stall for avoiding hazard, until the former one finished its execute stage (always 2-cycle penalty). This leads to pretty low efficiency of handling instructions in processor. To reduce stalls and improve processor's performance, branch prediction comes up[1]. Branch prediction is to predict the way the program will choose, and the predicting place is always in the digital circuit[2]. In general, the branch predictor makes prediction on whether the instruction will be taken or not according to the taken history of this program.

This project aims to calculate the performance of some branch predictors. According to the requirement on web, g-share predictor, tournament predictor and a custom predictor will be discussed in this report. The implementation of these three predictors will be shown in the next section. All predictors live in the docker environment, coding by C language. And finally the will demonstrate and analyze the result.

2. IMPLEMENTATION

This section will implement g-share predictor, tournament predictor as well as a custom predictor. For each predictor, first is to show the theorem and the way this project choose for implementation. Then their advantage and disadvantage will be discussed. And their evolution history is the last part.

All of these three predictors make prediction based on the taken history of the program, while their ways of using history information are quite different[3]. There are two history approaches, local history and correlated history. Local history only reflects the relation between a branch and its last branch, which always changes. And correlated history includes recent branches relations[4].

2.1 G-Share Predictor

•Working

The g-share predictor, whose full name is global share predictor, catches the global relations between branches. G-share has a multi-bit predicting register named Branch History Register (BHR). Using XOR to compare the branch address and the global history, BHR gets values of all bits. Then the processor load information in the BHR to predict whether the next branch will be taken or not. After prediction, processor operating 'XOR' again to update the information in BHR. This is the way that g-share approach predicts the taken status of all branches. The Figure 1 displays the process of predicting in a 2-bit g-share predictor.

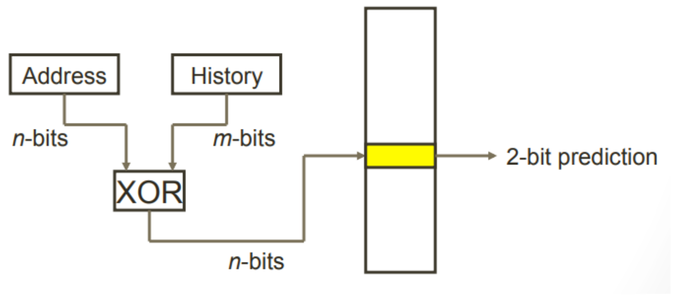

Figure 1: 2-bit g-share predictor process[3]

Initialize the global history as 13 bits. Predict the branch by using exclusive OR between address and global history. Train the predictor by the outcome and its current status.

$$\text{The system buget} = \text{global history } (2^{13} \text{ bits}) = 16 \text{ kb}$$

•Advantage&Disadvantage

The advantage of g-share is obvious. With the global history, predictor can handle a larger scale of branches, which broaden the application of predictors. However, though g-share approach can be valid dealing with a large number of branches, it underperforms local history in small predictors. G-share contains too much computations for a small predictor, especially whose higher order address bits are always sparse[5].

•Evolution

The g-share method evolves from g-select method. G-select predictor has fewer global history bits, thus has a narrower range of prediction. But g-select is still better than simple global method in large predictor and local method in small predictor. The advancement of processor requires a wider range of branch predictor, so researchers invented g-share approach to replace g-select predictor[4].

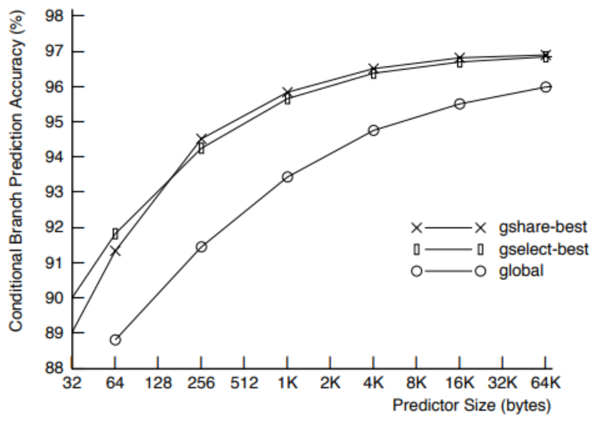

Figure 2: G-share predictor performance[5]

2.2 Tournament Predictor

•Working

Tournament predictor, also known as local/g-share predictor and combined predictor, tracks both local history and global history. It can be simply regarded as the combination of two predictors. One is local predictor, which is to predict branches independent from history. The other is a correlated predictor or a global predictor that is similar to g-share predictor. The way combines them is multiplexer, whose chooser depends on global history.The specific process is shown in Figure 3.In a tournament predictor, either g-share predicting part or local predicting part will be used, which can handle almost every popular predicting situation.

Initialize the global history as 9 bits, the local branch history as 10 bits, the pc index as 10 bits, and the predictor as 10 bits. Predict the branch by choosing global branch predictor or local branch predictor by a selector. Train the global branch predictor as g-share, the local branch predictor by outcome only, and choosing selector by those two predictor and the outcome.

The system budget= unsigned8 global history ($8*2^9$ bits) + unsigned4 local history ($4*2^{10}$ bits) + unsigned2 phistory ($2*2^{10}$ bits) + unsigned4 predict($4*2^{10}$ bits) = 14 kb.

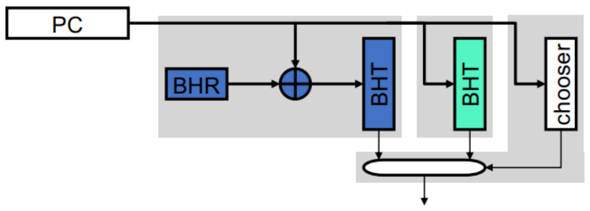

Figure 3: Tournament predictor process[4]

•Advantage&Disadvantage

The advantage of tournament predictor is that it can handle complex branches prediction. For example, one program has 20% history independent branches and 80% correlated branches, then using tournament predictor is appropriate. Its disadvantage is kind of similar to the g-share predictor, which is its logic and circuit complexity causing time-consuming and resource-consuming problem when dealing with simple branches or small number of predictions.

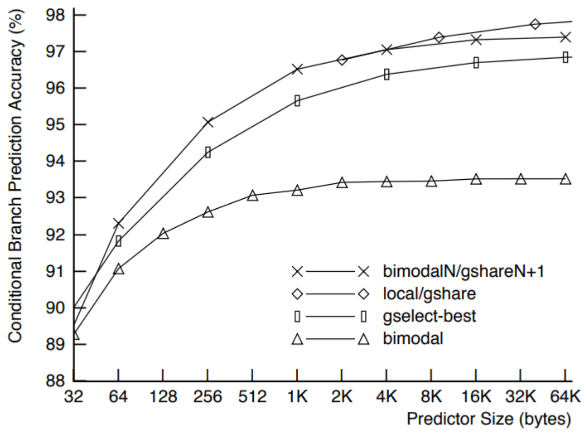

Figure 4: Tournament predictor performance[4]

•Evolution

Tournament predictor evolves from g-share predictor. G-share predictor outperforms global predictor and local predictor, but its ability in predicting history independent branches needs to be improved. To boost the performance of g-share predictor hashing local branches, tournament predictor was created. Combining g-share and local predictor, tournament can be implemented to much complex branches.

2.3 Perceptron Predictor

•Working

In this project, perceptron predictor is selected as the custom predictor. Perceptron algorithm derives from machine learning, and always appears in neural network. This algorithm plays a point-wise threshold role, restricting the input range[6]. In general, Perception predictor operates prediction by learning from branch history.

For branch predictor, perceptron method is quite similar to that of machine learning field. The process of prediction is shown as Figure 5.

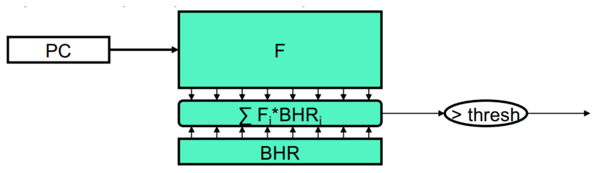

Figure 5: Process of perceptron predictor[7]

In the predicting part of branch prediction, perceptron approach builds a matrix of perceptron[7]. Each perceptron has a vector $(\omega_1, \omega_2 ... \omega_W)$ for storing the weight of branch history and ω_0 as bias. Let $H = (h_{W-1}, h_{W-2} ... h_1)$ be the input vector of branch history. The prediction p can be expressed as (1). Then compare p with the value of the threshold and update the perceptron.

$$p = \omega_0 + \sum_{i=1}^{W-1} (2h_i - 1)\,\omega_i \tag{1}$$

And for the training part, the perceptron is trained by following pseudo-code:

```
IF(prediction != outcome AND abs(p) < threshold):
    FOR (each bit in parallel):
        IF outcome = $h_i$ then $\omega_i$++ else $\omega_i$–
```

Initialize the number of perceptron as 512, the size of global history as 14, and threshold as 0. Predict TAKEN if p is positive, otherwise false. Train the perceptron if the prediction does not equal to the outcome or p is smaller than threshold.

In this project, each perceptron is a vector which consists of 14 global history parameters and 1 bias. The type of parameter in the perceptron vector is 8-bit unsigned int.Then the whole program of perceptron predictor can be implemented.

system budget = perceptron(512) * unsigned8(8 bits) * W(15) + threshold(int) = 60 kb + 256 b

•Advantage&Disadvantage

The main advantage of the perceptron predictor is the ability of capturing the correlation among branches in a long history. Its large scalability outperforms global predictor. Besides, unlike tournament predictor, which requires a metapredictor to compare the local history and global history, perceptron predictor uses multi-accumulate tree as the metapredictor.

The disadvantage is that perceptron predictor can only capture the linear correlations[8], since the multi-accumulate tree can only work for linear coefficient. Moreover, the logic and hardware of perceptron predictor is much more complex, which needs more system resource to implement.

•Evolution

The perceptron predictor evolves from bimodal/g-share predictor. For example, the Alpha EV8 predictor derives from the 2bcskew predictor, which can not handle a long history branch[9]. Researchers exploited machine learning algorithm on it and tried to capture the correlation in a larger scale of global history. Then the perceptron predictor was created.

3. OBSERVATION

Coding the g-share predictor, tournament predictor and perceptron predictor, the result of branch prediction is attainable. Set g-share global history = 13 bits, tournament global history = 9 bits, local history = 10 bits and index = 10 bits, perceptron history = 12 bits and index = 10 bits. Then the result table of the 6 sample data sets is shown as follows.

The misprediction rates are plotted in Figure 6. In table 1 and Figure 6, the misprediction rate of custom (perceptron) predictor beats that of g-share predictor and tournament for all six groups.

Table 1: Result of three predictor

Data	Parameter	G-share	Tournament	Perceptron
fp_1	Branches:	1546797		
	Incorrect:	12765	15297	12605
	Mis Rate:	0.825	0.989	0.815
fp_2	Branches:	2422049		
	Incorrect:	40641	74713	36688
	Mis Rate:	1.678	3.085	1.515
int_1	Branches:	3771697		
	Incorrect:	521958	485690	353486
	Mis Rate:	13.839	12.877	9.372
int_2	Branches:	3755315		
	Incorrect:	15776	17296	12830
	Mis Rate:	0.420	0.461	0.342
mm_1	Branches:	3014850		
	Incorrect:	201871	153860	112036
	Mis Rate:	6.696	5.103	3.716
mm_1	Branches:	2563897		
	Incorrect:	259929	222978	171576
	Mis Rate:	10.138	8.697	6.692

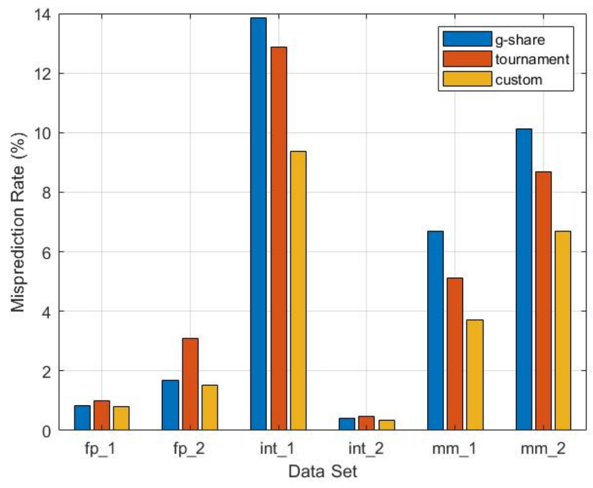

Figure 6: Misprediction rate

To discuss the relation between input and the misprediction rate, this project will show the curve while the input parameter varying.

First is the correlation between the misprediction rate of g-share predictor and its global history size. The input varies from 13 to 6, and then plot the curve. This is shown in Figure 7. In the image, predictor makes more misprediction when the global history size decreases. Therefore, for a g-share predictor, it would be more accurate almost surely when its global history is larger.

As for the tournament predictor, all its three inputs will be plotted with respect to misprediction rate. In the graph, reducing the global history size and index size will lead to a higher error rate. But the error would not change when the

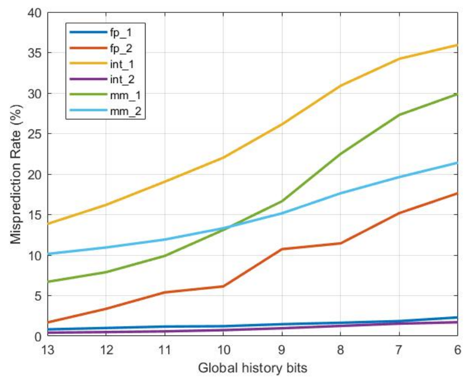

Figure 7: Correlation of misprediction rate and input of g-share predictor

local history changes. This is because no matter how large the size of local history is, only the last 1 or 2 bits of local history are used to store data.

Changing the number of perceptron and global history

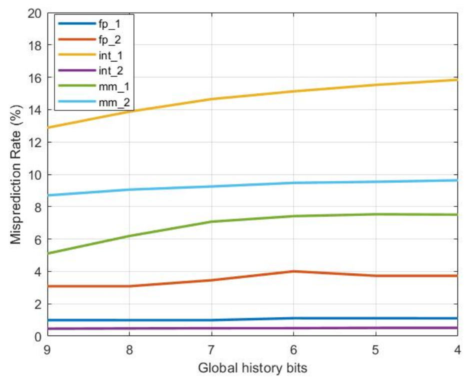

Figure 8: Correlation of misprediction rate and global history size of tournament predictor

size of perceptron predictor, the curve about those two input parameters can be displayed. The correlation of misprediction rate and the input parameter is shown in Figure 11-12. The image demonstrates that both perceptron size and global history size affect the rate of misprediction. And as their size become smaller, the rate slightly increase. Perceptron predictor is much more steady than g-share predictor and tournament predictor.

4. CONCLUSION AND RESULT

By the result of section 3, we can conclude that the perceptron predictor outperforms g-share predictor and tournament predictor even the input size decreases. This is because of its comprehensive prediction ability.

In the working part of perceptron predictor, we have discussed that this predictor has a perceptron part for learning correlations among data in a much longer history than global history in g-share and tournament, and also equips a local predictor for history-independent branches. Therefore, perceptron predictor can beat the other two for given parameters.

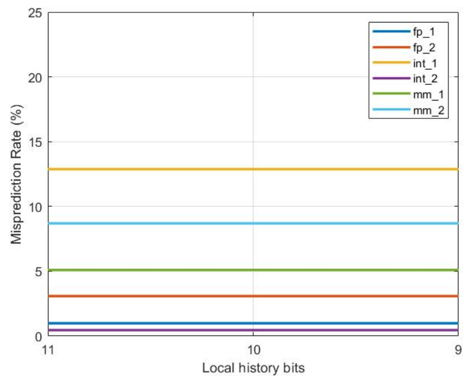

Figure 9: Correlation of misprediction rate and local history size of tournament predictor

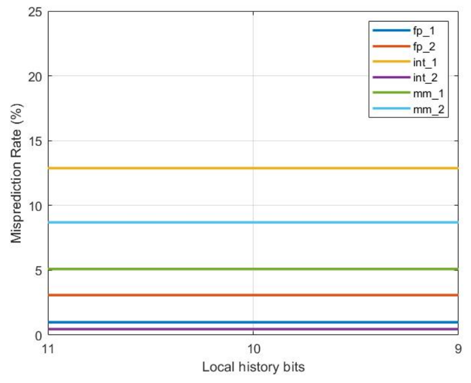

Figure 10: Correlation of misprediction rate and index size of tournament predictor

Besides, since perceptron predictor captures the correlation of branches in a longer history, it still owns enough information than g-share and tournament method even the input size decreases to less than half of initial size. In this way, perceptron predictor keeps a low misprediction rate and thus variation curves are more steady than the other two.

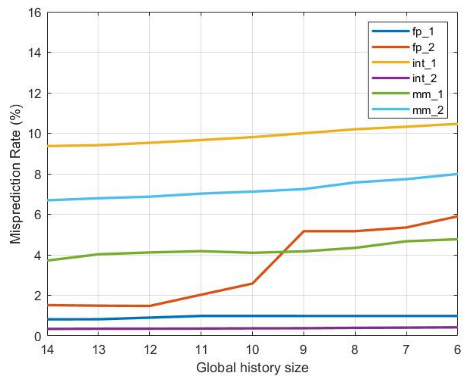

Figure 11: Correlation of misprediction rate and global history size of perceptron predictor

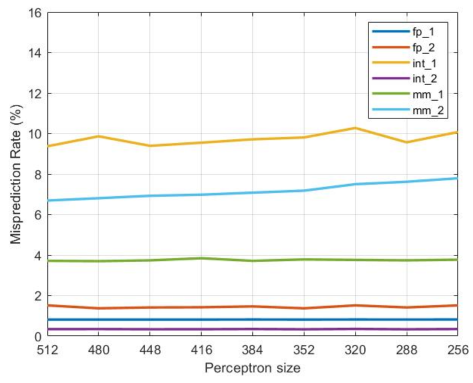

Figure 12: Correlation of misprediction rate and perceptron size of perceptron predictor

5. REFERENCES

[1] I. Bate and R. Reutemann, “Efficient integration of bimodal branch prediction and pipeline analysis,” in *11th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA’05)*, pp. 39–44, 2005.

[2] Wikipedia contributors, “Branch predictor — Wikipedia, the free encyclopedia,” 2021. [Online; accessed 11-June-2021].

[3] Jishen Zhao, “Branch prediction and hardware multithreading,” 2021.

[4] Scott McFarling, “Combining branch predictors,” 1993.

[5] D. W. Wall, “Global register allocation at link time,” in *Proceedings of the 1986 SIGPLAN Symposium on Compiler Construction*, SIGPLAN ’86, (New York, NY, USA), p. 264–275, Association for Computing Machinery, 1986.

[6] Wikipedia contributors, “Perceptron — Wikipedia, the free encyclopedia,” 2021. [Online; accessed 12-June-2021].

[7] A. S. R. Dolbeau, “Revisiting the perceptron predictor,” 2004.

[8] L. Faucett, “Fundamentals of neural networks: Architectures,” 1994.

[9] V. K. Andre ́ Seznec, Stephen Felix and Y. Sazeides, “Design tradeoffs for the ev8 branch predictor,” 2002.