3J-5

A Stack-based Solution for Alias Problem in Branch Prediction

Sijie YIN, Huatao ZHAO, Takahiro WATANABE
Graduate school of Information, Productions And Systems. Waseda university.

Abstract:

In modern embedded systems, improving accuracy in branch prediction is one of the most crucial problems. It's well known that branch alias has become one of the most serious problem that affects the accuracy in 2-level adaptive predictor. In this paper, we propose a stack-based solution which can alleviate alias problem significantly and improve the accuracy in branch prediction. Our proposed solution extends 4 bits on the conventional PHT's higher bits. Experiments are carried on Simple-scalar3.0e and the performance is verified by using SPEC2006 benchmarks. The result shows that the proposed structure can achieve about 10.5% improvement on IPC on average compared to the conventional prediction, and its extra hardware cost is negligible.

1. Introduction

There are mainly three problems that would decrease the accuracy and even make the whole system's performance become low. Alias is one of them which means that different PCs finally index to the same entry in the Pattern History Table(PHT).

Previously a concept of level in a program was proposed which means that we can divide the program into several levels according to some kinds of branch instructions such as function call, loop statement and so on.

According to the coherence between instructions in a problem, we usually consider that a branch instruction is relevant with other instructions which belong to the same level or in the lower level with the current branch instruction, as to those instructions whose level is higher than the current level, we take it as irrelevant.

The flowchart of fetch process in pipeline is shown in Figure 1. If PC fetches a branch instruction from cache, the branch will be delivered into the branch prediction module so as to make a prediction.

" A stack-based solution for alias problem in branch prediction"
The authors are with the Graduate School of Information, Productions and Systems.
E-Mail: yinsijie@ruri.waseda.jp

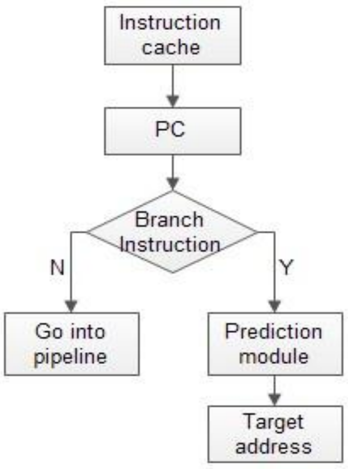

Figure 1: The flowchart of fetch process in pipeline

2. Proposed structure:

As shown in figure 2, a PC recorder, a stack and a comparator and 4 bits extended on PHT's higher bits are added on the conventional 2-level adaptive predictor. PC recorder is used to record the PC corresponding to the bits in Global History Register. Its update is at the same time with GHR. Comparator would produce a mask code via comparing the current PC and the element stored in PC recorder. As to the stack, it is mainly responsible for storing the branch instruction which has already been taken so as to record the relationship of levels. The extending 4 bits in PHT is used for storing the relationship of levels for each entry and waiting to be compared with the current PC's level.

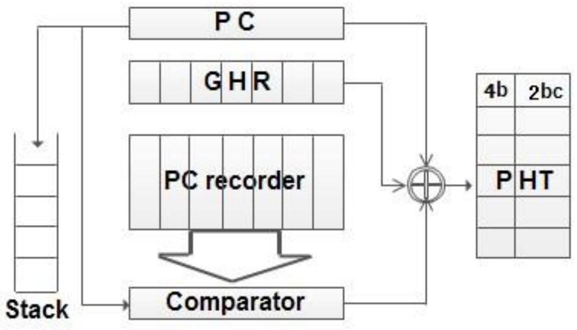

Figure 2: Structure of stack-based prediction model

Prediction process can be divided into four parts:

1: Compare current PC with the elements in PC recorder. If the element in PC recorder is in the same layer with the current PC, the comparator outputs “1”, otherwise it outputs “0”.

2: Compute the index for PHT. the new proposed index is calculated with GHR bits, current PC and mask bits.——Index=（GHR and mask）XOR PC

3: Compare the higher 4 bits in PHT with the current PC’s layer. If match, we access the entry. Otherwise, we access the next entry.

4: Update and push the stack after the current branch instruction is finished. We update the result to GHR and PC recorder. But if the result is “taken”, we also need to push the branch instruction into the stack and save the layer relationship.

3. Simulation

To evaluate the performance of this solution, the extended modification of sim-outorder simulator from simplescalar 3.0e is used as the basic simulation module, and load 8-integer, 2-floating point benchmarks to do 10 million instructions for each. This experimented architecture is shown as table 1.

Table 1: Architecture of the simulated model

Fetch	6 ins/cycle, 64-kbyte,2-way set associative I-cache, local and global branch predictor
Issue	Issues up to 4 integer and 2 floating-point, ins/cycle, 32-entries load/store queue
Pre-scheduling	Up to 64 ins, 6 ins/schedule-line
Functional units	8 integer ALU, 2 integer mult, 2mem ports, 8 FPALU,1 FP mult.

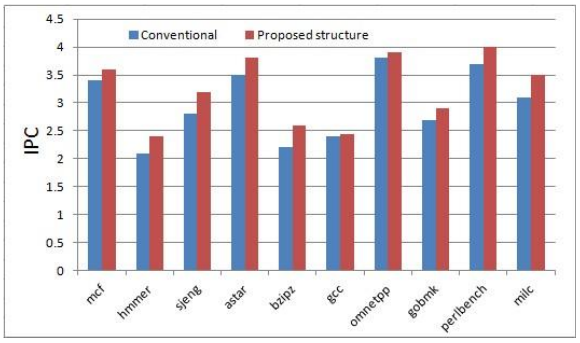

Figure3 : IPC of the conventional prediction and the proposed structure

Final result is shown in figure 3 and we can see that compared to the conventional prediction, gcc and mcf gain slight IPC improvement while omnetpp and bzipz obtain significant IPC improvement. On the whole, the proposed structure can achieve about 10.5% improvement on IPC on average.

4. Conclusion:

This paper proposes a stack-based solution for branch alias problem by adding a PC recorder, a comparator, a stack and extending 4 bits on PHT's higher bits. On the basis of simulation verification, the proposed structure can improve the IPC by 10.5% on average contrasting to the conventional prediction model.

Future work:

Referring to other parts in branch prediction, like the optimization of BTB, a more optimal structure and another set of parameters for PHT and BTB are needed to be done in the future.

References:

[1] Akkary . H, Srinivasan .S.T et al, Perceptron-Based Branch Confidence Estimation, Proceedings of the 10th International Symposium on High Performance Computer Architecture, IEEE Computer Society, pp:256-265, 2004.
[2] Seznec, A. et al, Design tradeoffs for the Alpha EV8 conditional branch predictor, Proceedings of the 29th annual international symposium on Computer architecture, Alaska, IEEE Computer Society, pp:295-306, 2002.
[3] SPEC2006, Standard Performance Evaluation Corporation, http://www.specbench.org.
[4] SimpleScalar LLC, Infrastructure for hardware modeling and software analysis, http://www.simplescalar.com.
[5] Abhishek Bhattacharjee, Thread criticality predictors for dynamic performance, power and resource management in chip muitiprocessors, Proceedings of the 36th annual international symposium on Computer architecture, ACM, pp:290-301, 2009.
[6] Edward Brekelbaum, Hierarchical Scheduling Windows. Proceedings of the 35th annual ACM / IEEE international symposium on Micro-architecture, Turkey, IEEE Computer Society Press, pp:27-36, 2002.