Department of Computer Science and Engineering
UNIVERSITY OF COLORADO
DENVER | ANSCHUTZ MEDICAL CAMPUS

Term Project Proposal

Pipeline Branch Prediction Strategies

[Team Name: IFElse](Bogki Yun, Yuguang Qiu, Qian Mai)

CSCI 5593 Advanced Computer Architecture
Computer Science and Engineering
University of Colorado Denver

Abstract

It is known to all that Pipelining is the major organizational technique that computers use to achieve high performance. Ideally, a pipeline uniprocessor can run at a rate that is limited by its slowest stage and increases instruction throughput by performing multiple operations in parallel, but does not reduce instruction latency as it still must go through all steps to complete a single instruction from start to finish. Therefore, we need a magic tool to guess which way a branch will go before this is known for sure. This is called the “branch prediction scheme”. It plays a critical role in achieving high effective performance in many modern pipelined microprocessor architectures such as x86.

Furthermore, a processor with an implementation of branch prediction that usually makes correct predictions can minimize the performance penalty from branching. However, if branches are predicted poorly, it may create more work for the processor, such as flushing from the pipeline the incorrect code path that has begun execution before resuming execution at the correct location. Based on this fact, our project discusses and implements a simulator of branch prediction strategies with the goal of finding a method to maximize the rate of correct predict.

Branches in the instruction stream disrupt the pipeline, by stalling and/or flushing of the pipeline, and reduce the processor performance well below ideal. Since branch instruction constitute a significant percentage of dynamic workload, techniques are needed to reduce the cost due to branches. Therefore, programs written for a pipelined processor try to avoid branching to minimize possible loss of the speed.

In this term project, our research goal is to build a simulator that can be used to evaluate the performance of several existing prediction schemes by both the accuracy of prediction. In order to demonstrate that how to predict the branch in the pipelining, and then show the performance of each predictor and find some efficient methods to improve the branch prediction strategies, our paper includes basic ideas and detailed discussions on such topic.

Motivation and Problem

Currently, Branch instruction constitute 15 to 30% of instruction executed on a typical machine. It can also break the smooth flow of instruction fetching and execution. Obviously, this kind of method

introduced a significant delay, because a branch that is taken changes the location of instruction fetches, besides, the issuing of instruction must often wait until conditional branch decisions are made.

Meanwhile, one of the major problems in designing a CPU pipeline is to ensure a steady flow of instructions. Such a flow can be impeded or interrupted by a change in the expected sequence of instructions. On high performance machine, such instructions consume a larger fraction of time because they cause pipeline stalls and flushes. On machines with well-defined instruction set, the frequency of branch instructions can be very high. Although the percentage of branch instructions in RISC machines may be lower, it is still essential to keep the cost of branch instructions low. In RISC structure, branch instructions are the major barrier to the goal of initialization of one instruction per cycle. What computer architects pressing for is to know the outcome of the branch and the target address as early as possible in the pipeline.

Solutions

1. Research and Comparison the performance

It is important that pipeline predicts the branch with an accurate and efficient prediction method. However, it is too difficult to predict which is taken or not taken because of lots of branch instructions. In this environment, in order to compare which branch predictor has a good efficiency, we need to know what kinds of branch there are. Therefore, we choose several branch predictions based on from basic method to well known in a recent year, which are smith branch predictor[1], two-level adaptive branch predictor (TLA) [2], statically taken / not taken, bimodal, branch history table (BHT), combination [3], correlation, skewed [4], Gskewed Branch Predictor [5], and gshare [6] branch prediction schemes. Like this, many branch prediction schemes have been proposed. In our project, we will use selected benchmark programs from the SPEC95 to compare several well-known branch prediction schemes.

2. Suggestion for accuracy and efficiency

A static branch prediction scheme relies on the information incorporated in prior to runtime whereas the dynamic (adaptive) branch prediction logic tries to extract the information and predicts the behavior of branches in run-time. Most of the adaptive predictors use these two parts of information.

The forward and backward information of the branch target is helpful to increase the accuracy. Supported by the profile, a compiler may be able to reduce the need for branch predictors.

3. Simulation Platform

We will build a simulation platform for this project. This platform uses selected benchmark programs from the SimpleSclar [7] tool suite written by Todd M. Austin to compare several well-known branch prediction schemes that described above. This tool set consists of compiler, assembler, linker, simulation, and visualization tools for the SimpleScalar architecture. In addition, this simulation tool is advanced in many aspects, e.g., it can execute one million branch instructions in a short time. And implementation of these schemes included various modifications to the bpred.c, bpred.h and sim-inorder.c files. We will run this simulation linux based PC.

Project Milestone

- 3.01 ~ 3.11: Proposal of the project - Pipeline Branch prediction strategies

- 3.12 ~ 3.25: Research and Study on relative knowledge

- 3.26 ~ 4.10: Establish simulation platform and run the program

- 4.11 ~ 4.16: Analyze data graph and draw conclusions

- 4.17 ~ 4.30: Complete paper thesis and project presentation

Reference

[1] Smith, J.E., "A Study of Branch Prediction Strategies", *Proceedings of the 8th annual symposium on Computer Architectures 1981, IEEE Computer Society Press*: Minneapolis, Minnesota, USA. p. 135-148

[2] Yeh, T.-Y. and Y.N. Patt, "Two-level adaptive training branch prediction", *Proceedings of the 24th annual international symposium on Microarchitecture* 1991.

[3] McFarling, "Combining Branch Predictors", *WRL Technical Note TN-36, Digital Equipment Corporation*, June 1993

[4] Michaud, Pierre, Andre Siznec, Richard Uhlig, "Skewed branch predictors", *Tech. report, IRISA publ. int. 1031*, June 1996

[5] Michaud, P., et al., "Trading conflict and capacity Aliasing in conditional branch predictors", *Proceedings of the 24th annual international symposium on Computer architecture*, 1997.

[6] McFarling, S., "Combining branch predictors", *Compaq Computer Corporation Western Research Laboratory*, 1993.

[7] D. Burger, T.M.A., and S.Bennett, "Evaluating future micro-processors: the SimpleSclar tool set", *Technical Report CS-TR-1996-1308, University of Wisconsin-Madison*, 1996