Branch Prediction Models Exploration

Daniela Trevina
Department of EECS
Texas A&M University-Kingsville
Kingsville, USA
daniela.trevino@students.tamuk.edu

Daniela Lopez
Department of EECS
Texas A&M University-Kingsville
Kingsville, USA
daniela.lopez@students.tamuk.edu

Mengxiang Jiang
Department of EECS
Texas A&M University-Kingsville
Kingsville, USA
mengxiang.jiang@students.tamuk.edu

Samah Allahyani
Department of EECS
Texas A&M University-Kingsville
Kingsville, USA
samah.allahyani@students.tamuk.edu

Ugochukwu Onyeakazi
Department of EECS
Texas A&M University-Kingsville
Kingsville, USA
ugochukwu.onyeakazi@students.tamuk.edu

***Abstract*—**

I. Introduction

Branch prediction is a crucial component in modern computer architecture to improve performance and energy efficiency. When executing a program, the processor encounters branches in the control flow, where it must decide which path to take based on a conditional statement. Incorrect branch predictions result in wasted computation cycles and negatively impact the overall performance of the system. To mitigate this issue, branch predictors are employed to predict the outcome of the conditional statements, and enable the processor to pre-fetch instructions from the predicted path, reducing the overhead of branching.

Over the years, a variety of branch prediction models have been developed to improve the accuracy of predictions. These models range from static branch predictors that use heuristics to predict the outcome of branches, to more sophisticated dynamic branch predictors (all models tested in this paper are dynamic) that rely on past history to make predictions. Furthermore, the emergence of machine learning and artificial intelligence has led to the development of more advanced branch prediction techniques.

The aim of this paper is to explore the various branch prediction models and compare their performances. We will examine the underlying principles of each model, the algorithms used to make predictions and their implementations in a simulated processor. We will also discuss the challenges associated with branch prediction, such as the trade-off between prediction accuracy and hardware complexity.

II. Models

In this section, we will cover the various branch predictor models used in the simulation.

A. Bimodal Predictor

This model stores two bits in the branch target buffer (BTB) indicating whether this particular branch was taken or not taken the last two time. The bimodal branch predictor's accuracy will increase exponentially if the branch history size is increased. Most of the predictor's branches are taken or not taken and are not random. The main goal of this model is to distribute the branch behavior and predict which are the taken branches. The bimodal branch has a default size of a 2-bit counter which is organized in the low-order address-bit program counter section. When the branch is categorized as taken, the counter is increased. When the branch is categorized as not taken, then the counter is decreased. The counter value is limited and must be within the range of zero to three. This allows the branch predictor model to be categorized as taken or not taken using repeated values.

B. Perceptron Predictor

This model is a single-layer version of an artificial neural network that can identify and classify patterns, first applied to branch prediction by Jimenez and Lin [2]. In the model shown in Fig. 1 are the input vector (x), the weight vector (w), and the output (y). The inputs correspond to values taken from the global history register, with the exception of x_0, the bias, always set to 1. The output's sign determines the prediction. If the sign is negative, the branch is not taken, otherwise it is taken. Equation (1) shows the calculation of the output:

$$y = w_o + \sum_{i=1}^{h} x_i w_i \tag{1}$$

The weights of the perceptron are trained using the algorithm

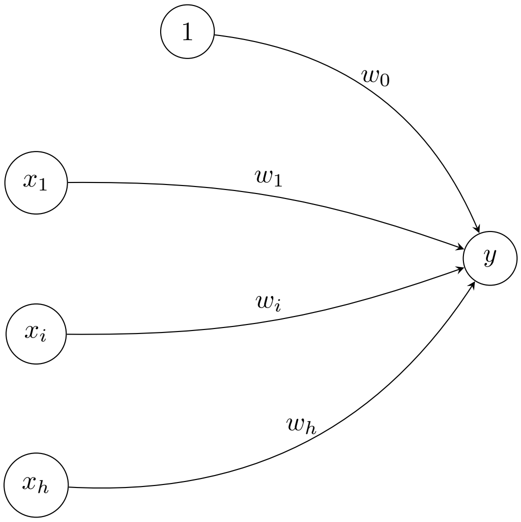

Fig. 1. Perceptron Predictor

below:

```
if sgn(y) != t or abs(y) <= theta  then
    for i := 0 to n do
        w[i] := w[i] + t*x[i]
    end
end
```

t is 1 if the branch is actually taken or -1 if not. θ is the threshold to determine when training should stop. w_i is incremented if t and x_i agree, and decremented if they disagree.

C. Hashed Perceptron Predictor

This predictor is based off of the perceptron branch predictor from the previous section(**??**) but with a few important changes. The first change is instead of a single table for the weights, multiple independently indexed tables of perceptron weights are used, introduced by Jimenez [1]. The second is using the hash of the branch history rather than just the branch history for indexing in order to reduce the number of independent tables, contemporaneously by Seznec [5], Tarjan and Skadron [7], and Loh and Jimenez [3]. The third is instead of using a fixed history length, exponentially increasing history lengths are used for indexing, by Seznec [4] [6]. This idea is actually one of the key parts of the TAGE predictor described later. The last is dynamically adjust the θ value for the training, also by Seznec [4] [6].

References

[1] Daniel A Jiménez. Fast path-based neural branch prediction. In *Proceedings. 36th Annual IEEE/ACM International Symposium on Microarchitecture, 2003. MICRO-36.*, pages 243–252, 2003.

[2] Daniel A Jiménez and Calvin Lin. Dynamic branch prediction with perceptrons. In *Proceedings HPCA Seventh International Symposium on High-Performance Computer Architecture*, pages 197–206. IEEE, 2001.

[3] Gabriel H Loh and Daniel A Jiménez. Reducing the power and complexity of path-based neural branch prediction. In *Proceedings of the 5th Workshop on Complexity Effective Design (WCED5)*, pages 1–8. Citeseer, 2005.

[4] André Seznec. The o-gehl branch predictor. *The 1st JILP Championship Branch Prediction Competition (CBP-1)*, 2004.

[5] André Seznec. Revisiting the perceptron predictor. *PI-1620, IRISA, May*, 2004.

[6] André Seznec. Analysis of the o-geometric history length branch predictor. In *32nd International Symposium on Computer Architecture (ISCA'05)*, pages 394–405. IEEE, 2005.

[7] Kevin Skadron and David Tarjan. Revisiting the perception predictor again. 2004.