ADVANCED COMPUTER ARCHITECTURE FALL 2018

BRANCH TARGET BUFFER PROJECT

TEAM CANNON LAKE

Members:
Md Aminul Hoque
Shantonu Das
Aqeeb Iqbal Arka

Introduction

Branch prediction is a technique used in CPU design that attempts to guess the outcome of a conditional operation and prepare for the most likely result. When a conditional operation needs to be processed, the branch predictor "speculates" what condition is most likely to be met. It then executes the operations required by the most likely result ahead of time so that they are already complete if and when the guess was correct. At runtime, if the guess turns out not to have been correct, the CPU executes the other branch of operation, incurring a slight delay. But if the guess was correct, speed is significantly increased. The first time a conditional operation is seen, the branch predictor does not have much information to use as the basis of a guess. But the more frequently the same operation is used, the more accurate its guess can become.

In this project, the branch target buffer (BTB) is coded and simulated in MATLAB. At first, the branch prediction algorithm is tested on a small program trace and then tested on an integer program and floating-point benchmark. The predictors used in the algorithm are 2-bit (0,2) and 2-history (2,2). The BTB size is fixed at 1024 entries. We have implemented local prediction, global prediction with history, and finally, a tournament prediction to compare between them. Hit rate, miss rate, right prediction, wrong prediction, etc. are calculated as performance metrics and compared for different approaches.

Requirements for the Project

Specifications:

- Local BTB size: 1024 Entries (0,2) predictor
- Global BTB size: 1024 Entries (2,2) predictor
    i. One-way BTB
- 2 Benchmarks (sample trace is not considered a benchmark)
- Predictors: 2-bit (0,2) and 2-history 2-bit (2,2)
    i. Below is the predictor’s 2-bit state machine
    ii. The 2-bit history will be the global history:
        1. It was taken (1) or Not taken (0)
        2. This requires four predictors per entry
    iii. Selector state machine is below
- Hit rate: This is defined as follows

$$Hit\ rate\ (\%) = \frac{number\ of\ hits}{(number\ of\ hits) + (number\ of\ misses)}$$

- Prediction accuracy (or miss prediction):

$$Accuracy\ (\%) = \frac{number\ of\ correct\ (right)\ predictions}{(number\ of\ hits)}$$

Description

The state machine for local predictor is shown below:

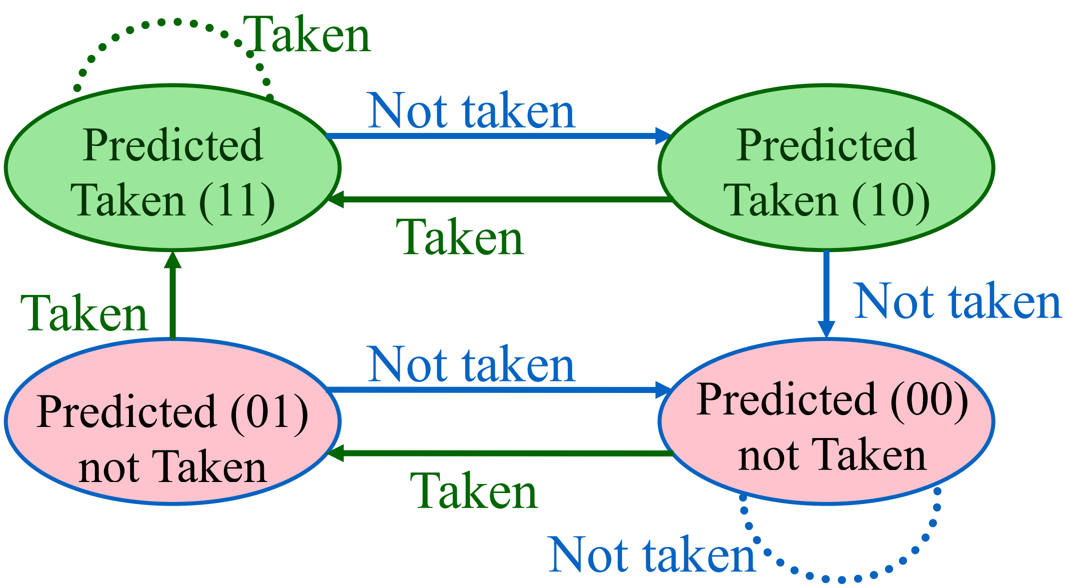

Figure 1: State machine for local predictor

Two other designated state machines are shown below, which are used for global prediction. State machine A initializes with strongly taken and state machine B initializes with weakly taken. Based on the two state machines, prediction accuracy will be compared.

State Machine A:

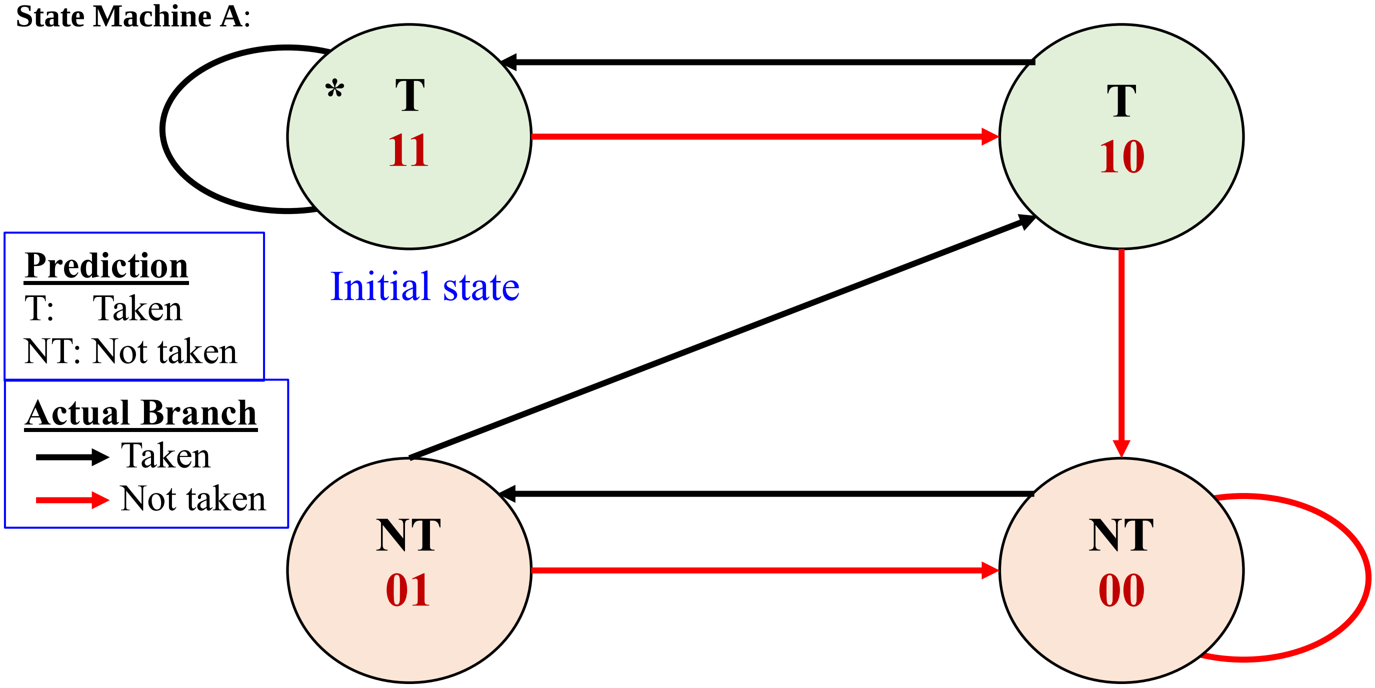

Figure 2: State machine for global predictor A

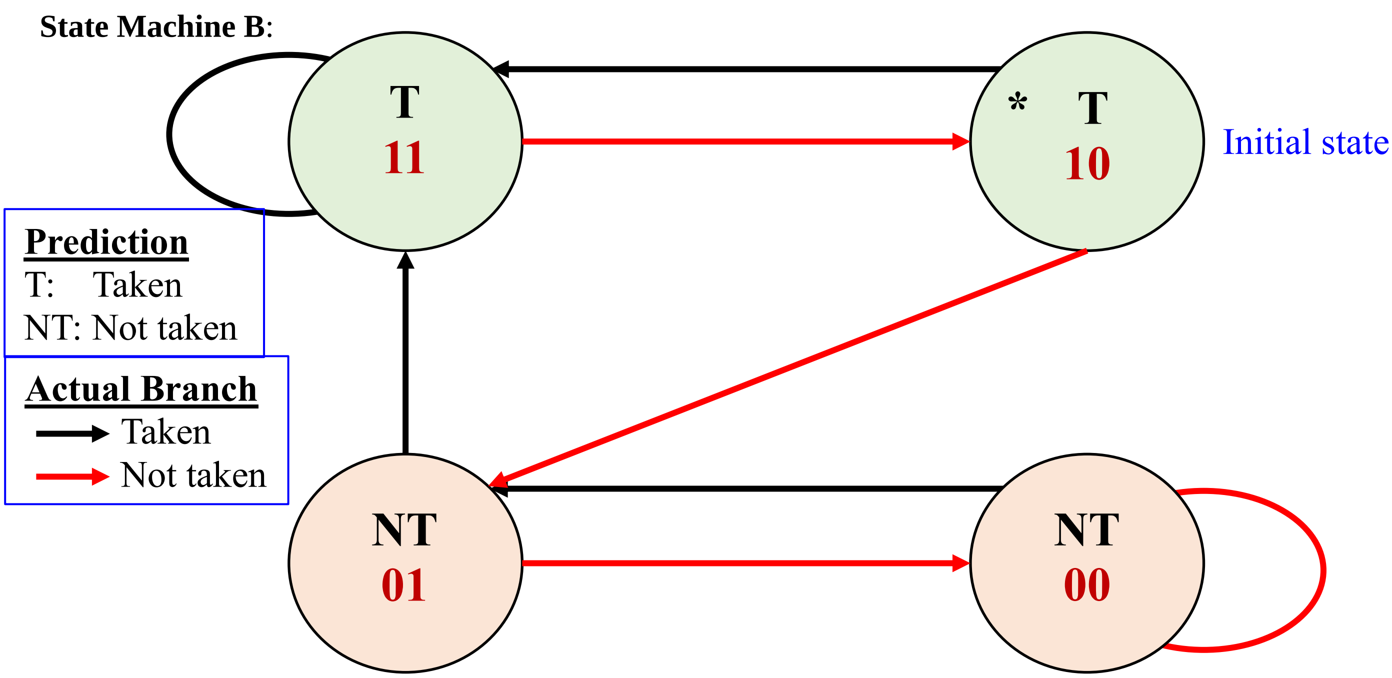

Figure 3: State machine for global predictor B

Results

The following table summarizes the results obtained from simulation of different benchmark tests:

Benchmark	Predictor	Global State machine	No. of hits	Hit rate	No. of miss	Wrong Address	Collision	Right Prediction	Wrong Prediction	Accuracy
Espresso_int	Local only	-	130256	99.6542	452	671	111	123986	6270	95.1864
	Global only	A	130256	99.6542	452	671	111	123977	6279	95.1795
		B	130256	99.6542	452	671	111	124062	6194	95.2447
	Tournament	A	130256	99.6542	452	671	111	124067	6189	95.2486
		B	130256	99.6542	452	671	111	124094	6162	95.2693
Spice_FP	Local only	-	141760	93.3584	10085	10848	9486	135008	6752	95.2370
	Global only	A	141760	93.3584	10085	10848	9486	136446	5314	96.2514
		B	141760	93.3584	10085	10848	9486	136587	5173	96.3509
	Tournament	A	141760	93.3584	10085	10848	9486	136470	5290	96.2683
		B	141760	93.3584	10085	10848	9486	136525	5235	96.3071

The number of hits and misses are the same for different kind of predictors. Because of that, the number of hits, misses, collisions are the same for different predictors, as shown in the table. The number of right predictions, wrong predictions, and accuracy differs for different predictors. The following graphs illustrate the accuracy of different predictors:

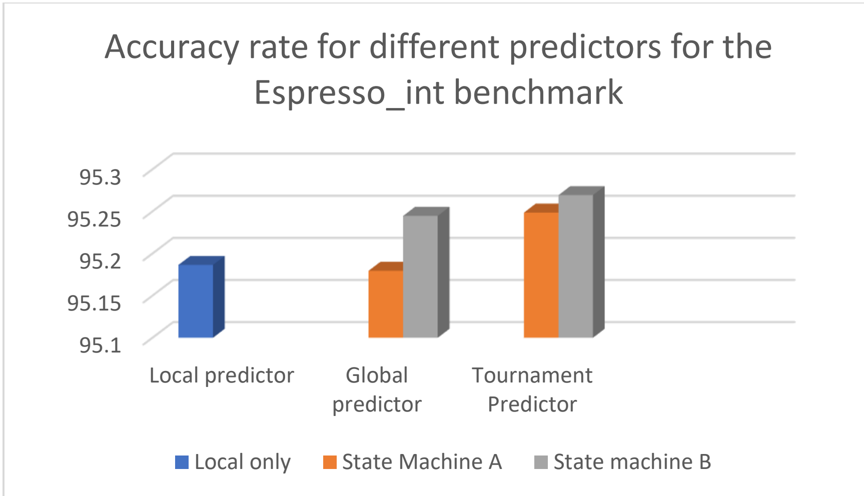

Figure 4: Comparison in accuracy rate for different predictors for Espresso_int

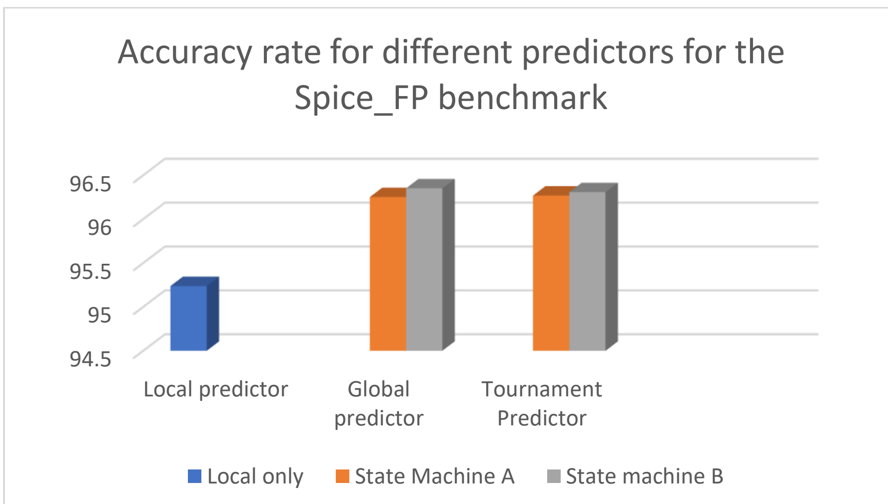

Figure 5: Comparison in accuracy rate for different predictors for Spice_FP

From the figures above, we can observe that for both benchmarks, there is slight improvement when we change the predictor from local only to global. However, the highest accuracies for both benchmarks are achieved when tournament predictor is used. Compared to the integer benchmark, the overall prediction is more accurate for the floating-point benchmark.

Discussion:

1. The hit rate is decently high for both benchmarks due to the relatively large size of the BTB. This size has been shown to achieve a proper balance of performance improvement and timing penalty, which is why this size is chosen for this project. However, as the hit

rate is the characteristic of the BTB rather than the prediction scheme, it does not change for different predictors.

2. There is not a significant performance improvement going from Local to Global to Tournament prediction scheme. The improvement mainly comes from combining Local and Global prediction schemes that can provide better results for different parts of the trace. Therefore, the tournament prediction is a good idea.
3. However, from our results, we can see that there is not a significant amount of improvement when moving from Local to Tournament prediction schemes. Adjusting the state machines of the global predictor can also help improve the performance of the predictor. The higher computation needed for tournament prediction schemes can cause the timing penalty to be more substantial than the simpler Local prediction scheme. So, in many instances, a simpler prediction scheme is preferable.