CS422: Programming Assignment 1

Shubham Singh[1], Aniket Pandey[2], and Nidanshu Arora[3]

[1]170689, @singhshu
[2]160113, @aniketp
[3]160443, @nidanshu

ABSTRACT

Implementation of L-TAGE Branch Predictor, which was proposed by Andre Seznec in 2007. We used ChampSim to simulate to L-TAGE predictor.

Division of Labour

Roll No.	Name	Distribution	Individual Contribution
170689	Shubham Singh	60%	Implemented Loop-table, CSR. Bug-fixes, performance & storage optimizations
160113	Aniket Pandey	30%	Implemented the base TAGE predictor. Bug-fixes, optimizations
160443	Nidanshu Arora	10%	Worked on final predictor optimizations

Insights

Shubham Singh

1. Instead of Increasing prediction accuracy, LOOP table is degrading the performance. One of the possible reason maybe is that loop-table is capturing some if-else instructions inside a loop, which may distort the global history, important for TAGE table. Therefore, we are using loop table only if the number of iteration is more than 2. There is no technical reason for choosing that number, there was no significant change in performance if we choose 3 or 4, but without implementing any threshold on iterations, performance decrement was significant.

2. Using fixed TAGE tag width of 12-bit for all tables improves the accuracy, but the storage overhead was significant which can't be ignored. We experimented with various combination of tag width, and finalized one which was least degrading($\widetilde{=}$-0.3%) in performance wrt to fixed TAGE tag width.

3. Similar to Tag width, fixing number of entries for all table to 2^{12}, results in good accuracy, but due to storage overhead, we fix different number of entries for different tables.

4. Increasing the length of global history table for TAGE results in slightly degradation of performance. Therefore we experimented with various values of α (for $L(i) = (int)(\alpha^{i-1} * L(1) + 0.5)$) and choose $\alpha = 1.7$, $L(1) = 5$.

Aniket Pandey

1. The scheme to avoid ping-pong phenomenon as mentioned in the LTAGE paper with gradually decreasing probabilities of entry allocation in an eligible table is particularly complex and hence a simpler method was implemented. Instead of decreasing the probablity to 0.25 for the third table, we let it be a constant value of 0.5 and iterate until the condition is satisfied. We observe a similar performance for our implementation.

2. We skip the improvement presented in LTAGE paper for updating useful counter and stick to the original algorithm from TAGE. The resetting of bits of useful counter is preserved, however we refrain from flipping the signification of bits as there is no visible improvements in performance.