International Journal of Engineering Technology Science and Research
IJETSR
www.ijetsr.com
ISSN 2394 – 3386
Volume 4, Issue 8
August 2017

# A Study on Improving Branch Prediction Accuracy in the context of Conditional Branches

**AN. Sai Parasanna** **Dr. R. Raghunatha Sarma** **Dr. S. Balasubramanian**
Sri Sathya Sai Institute of Higher Learning

**ABSTRACT:**
*High-performance micro-architectures use deep super scalar pipelines to help speed up execution. The importance of an accurate branch predictor to the efficacy of pipeline in the presence of conditional branches is well-known. Literature contains a number of proposals of effectively using the branch history and path history bits along with the branch instruction address to predict conditional branches. In spite of significant research in the field of branch prediction, microprocessor industry is heading towards zero cycle latency branch prediction with 100% accuracy which is not the current reality. The work embodied in this thesis deals with the study of various branch prediction techniques for conditional branches and their accuracy in predicting the same. We have used Intel championship branch prediction (CBP) simulator [1] and the traces provided along with [2] as the experimental framework. We implemented Gshare [3], a global predictor and two-level adaptive training [4], a local predictor. Experimental results of perceptron predictor [5], a dynamic branch predictor based on neural networks demonstrates more accuracy than the local and global predictors. State-of-the-art dynamic branch predictor TAGE [6] was implemented which outperformed prior implemented predictors. TAGE considers a tagless base bimodal predictor along with several partially tagged tables indexed with branch history, whose lengths form a geometric series. Geometric series of history lengths allows us to use very long history lengths to index predictor tables as well as on recent branches, while still allocating most of the storage space to shorter global history predictor tables. We improvised the prediction efficiency of TAGE through the effective use of hysteresis bits for the base bimodal predictor. We have proposed perceptron-TAGE (P-TAGE) which uses perceptron predictor as the tagless base predictor in TAGE. Our experimental results show that P-TAGE outperformed TAGE for hardware budgets 4Kb and 8Kb for generic workloads. Moreover, P-TAGE performs better than TAGE for floating point workloads for hardware budgets in the range 4Kb-64Kb.*

***KEYWORDS:***
***Gshare, TAGE, P-TAGE, perceptron, branch prediction, pipeline***

## INTRODUCTION:

Branch prediction accuracy remains to be critical for high performance and low power consumption in modern processors. A lot of research and study has been done in this area taking into account the performance degradation due to a pipeline stall on modern processors on a misprediction. With advanced architecture techniques such as multiple issue, instruction re- order, branch prediction accuracy reaching 99% and more with zero cycle latency to predict branches is where the microprocessor industry is heading towards. Though the microprocessor industry has been quite successful in predicting conditional branches with almost 99% accuracy, there is scope for improvement. Moreover, it has a long way to go with respect to indirect branches. Branch instructions are broadly classified into four classes, namely,

1. Conditional branches
2. Subroutine return branches
3. Immediate unconditional branches
4. Unconditional branches on registers

The last two classes of branches are called indirect branches. It is all the more difficult to predict indirect branches than conditional/direct branches. To understand the difficulty in predicting indirect branches, we need to understand the steps involved in branch prediction. Branch Prediction involves a twostep prediction.

International Journal of Engineering Technology Science and Research
IJETSR
www.ijetsr.com
ISSN 2394 – 3386
Volume 4, Issue 8
August 2017

1. Direction: taken/not taken
2. Target Address: correct/wrong target

The key approach adopted towards minimizing branch penalty and maximizing instruction flow throughput is to speculate on both branch direction and branch target address. Branch direction speculation involves the use of branch predictors to predict whether the branch is taken/not taken. But once the direction of prediction is known, the target PC address should be known to fetch the corresponding instruction. This involves branch target address speculation. Branch target speculation involves the use of a structure similar to that of caches, called Branch Target Buffer (BTB) to store previous branch target addresses.

With respect to conditional branches, based on condition code (CC), either branch is taken/not taken. In case of branch being taken, target Address is just current PC+4 or PC+8 depending on the instruction length as the next instruction needs to be fetched and executed. If the branch is not taken, then from the information provided by the compiler, offset from the current PC is calculated and the target address is PC + α (α: offset). Subroutine return branches can be predicted by using a return address stack.

Target address of an immediate unconditional branch is calculated by adding the offset in the instruction to the PC; therefore, the target address can be generated immediately. Unconditional branches on registers have to wait for the register value, i.e., the target address for the jump is stored in the register which can be known only in the later stages of pipeline. Moreover, indirect branches are always taken, i.e. direction prediction is trivial but calculating target address is not trivial. The register value might have been changed from the previous time of prediction and does not follow a pattern as followed by conditional branches. Thus, when it comes to indirect branches, the only speculation that needs to be done after decoding the instruction is target address, using branch table buffer (BTB).

## RELATED WORK:

*1. Branch Target Buffer:*

Some of the design issues of BTB and optimization for dynamically predicted branches is discussed here [20, 21]. On the execution of static branch for the first time, an entry in the BTB is allocated. The corresponding instruction address of the branch instruction is stored in the BIA field, and the target address of the branch instruction is stored in the BTA field. Let's assume for simplicity that BTB is fully associative cache, the BIA field is used for associative access of the BTB. Concurrent accesses of I-cache and BTB is carried out. A hit results in the BTB, when the current PC matches with the BIA entry in the BTB. This implies that the current instruction being executed has been executed before and is a branch instruction. On the result of a hit in the BTB, the BTA field of the hit entry is accessed and is used as the NPC value if the branch is predicted taken. Hence, by accessing the BTB using the branch instruction PC address, in one pipeline stage, NPC value is predicted. However, if the prediction goes wrong, branch misprediction recovery must be invoked, essentially pushing the pipeline of the uncommitted instructions executed before the branch instructions.

*2. Gshare predictor:*

Two-Level Adaptive training prediction scheme, a global branch prediction scheme has the disadvantage that, it fails to capture the locality information related to a branch. From architectural perspective, a branch instruction's outcome may be completely or partially dependent on its locality, i.e., PC address of the branch instruction. Not all branches are correlated with some other branch. Thus one solution could be to associate one global branch predictor with every branch instruction (associated PC address). However, most of the entries in each PT of the global predictor will not be populated. To visualize, imagine the set of all combinations of history of branches taken before coming to a particular branch. This can be captured with one global predictor. Now extend the same to all branch instructions. This needs N pattern tables, where N is the total number of branch instructions in a program. But to get the best of both worlds, one simple logical operator can be used. XOR operator can effectively, in one table, capture both local and global in- formation i.e., XOR PC address and BHR for any given branch history bits and PC address [7]. This predictor is called the Gshare predictor. This is definitely practical and efficient for implementation in hardware.

International Journal of Engineering Technology Science and Research
IJETSR
www.ijetsr.com
ISSN 2394 – 3386
Volume 4, Issue 8
August 2017

*3. Perceptron based predictor:*

Similar to the manner in which a processor keeps the table of two-bit counters in fast SRAM, the table of perceptrons is kept. Hardware budget drives the limit on the number of perceptrons, N, and the number of weights which in turn is determined by the branch history length. The following are the steps that a processor encounters conceptually when a branch instruction is encountered in the fetch stage.

- Branch address is hashed which produces as index i ∈ 0...N-1. This index is used to index into the table of perceptrons.
- From the table of perceptrons, the $i^{th}$ perceptron fetched is stored into a vector register, $P_{0..n}$. This register contains the weights of the $i^{th}$ perceptron.
- The dot product of P with the GHR gives the output y.
- Branch is predicted not taken when y is negative; when y is positive, branch is predicted as taken.
- Once the actual outcome of the branch is known, the training algorithm updates the weights in P based on the actual outcome and the value of y.
- The updated value of P is written back to the $i^{th}$ entry in the table of perceptrons.

*4. TAGE predictor:*

Figure 1 illustrates a TAGE predictor. The TAGE predictor characterizes a base predictor $T_0$ which provides a basic prediction and a group of partially tagged predictor components $T_i$. These tagged predictor components $T_i$, $1 \le i \le M$ are indexed using different history lengths which form a geometric series, i.e., L(i) = (int) ($\alpha i-1$ * L(1) + 0.5). The order of arrangement of the tagged tables is such that, the table using the longest history is $T_1$, followed by $T_2$ using the next largest history length, and so on with TM being the table using the smallest history length in the geometric series. The base predictor is a simple bimodal table with 2-bit counters, indexed using PC value. A single entry in the tagged component consists of a signed counter ctr whose sign provides the prediction, an unsigned useful counter u and a partial tag. u is a 2-bit counter and ctr is a 3-bit counter. For a given budget, the number of entries in the base predictor, the length of the tag bits and number of entries in the tagged component used are as given in the Table 1.

**Configuration of 5-component, 8-component and 14-component predictors**

| No of components | Storage budget | $T_0$ entries | Tag width | u + ctr | Tagged $T_i$ entries |
|---|---|---|---|---|---|
| 5 | $2^n$ bits | $2^{n-3}$ bits | 9 bits | 2 + 3 bits | $2^{n-6}$ bits |
| 8 | $2^n$ bits | $2^{n-3}$ bits | 11 bits | 2 + 3 bits | $2^{n-7}$ bits |
| 14 | $2^n$ bits | $2^{n-3}$ bits | 9 bits | 2 + 3 bits | $2^{n-7}$ bits |

Table 1

**5-component TAGE predictor**

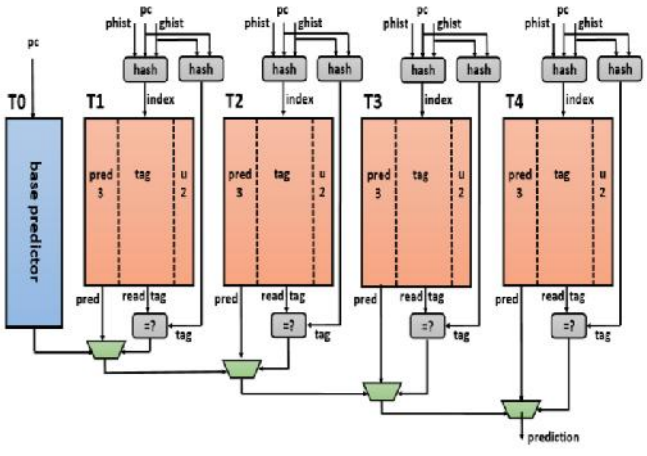


Figure 1

International Journal of Engineering Technology Science and Research
IJETSR
www.ijetsr.com
ISSN 2394 – 3386
Volume 4, Issue 8
August 2017

**OUR CONTRIBUTIONS:**

- An extensive literature study of the various branch prediction techniques, branch predictor algorithms and available simulator/ framework for implementation.
- In-depth understanding of The Championship Branch Prediction(CBP) simulator [7, 1].
- Implemented following branch predictors using CBP simulator.
  - Two-level adaptive training branch predictor.
  - Gshare branch predictor.
  - Perceptron branch predictor.
  - TAgged GEometric (TAGE) history length predictor.
- Improved the results of the TAGE predictor [6] by efficiently using hysteresis bits for the base bimodal predictor.
- Proposed a new Perceptron-TAGE (P-TAGE) predictor by integrating the TAGE predictor with a neural network base predictor.

The codes implemented as part of this work are pushed onto GitHub and can be pulled from https://github.com/SaiPrasannaAnnamalai/BranchPredictors. The implemented codes run to around 3000 lines.

**P-TAGE PREDICTOR:**

The P-TAGE predictor consists of a perceptron predictor as the base predictor and partially tagged components indexed using geometric history lengths of branch (path) history and branch address (PC value) of the TAGE predictor. We compare the performance of the P-TAGE predictor with the TAGE predictor and the scenarios where each of the predictors outperform the other.

It is clearly seen from the Figure 2, that 8-component TAGE outperforms perceptron, Gshare and two-level adaptive predictor for hardware budgets in the range of 32Kb-1024Kb. But on closer observation it can be seen that, the slope of the perceptron predictor predominantly remains close to constant as the hardware budget is increased as opposed to TAGE and other predictors whose misprediction/KI improves largely with increase in hardware budget. For 32Kb and 64Kb budget, perceptron predictor performs close to the 8-component TAGE predictor. One reason could be the inherent ability of the perceptron predictor to consider longer history lengths at shorter budget. Moreover, the reason for the constant slope could be the longer training times with larger predictor tables before a correct prediction is given. Besides only workstations, servers can accommodate branch prediction logic in the size of megabytes around the processor. Moreover, in the two realistic tracks of the CBP tournament, the submitted predictors should be of size 4Kb and 32Kb respectively. The need for efficient lower hardware budget branch predictors coupled with the performance of perceptron predictors close enough to the TAGE predictor at smaller hardware budget gave us the hint towards developing a predictor that couples these two.

**Comparison between existing branch predictors**

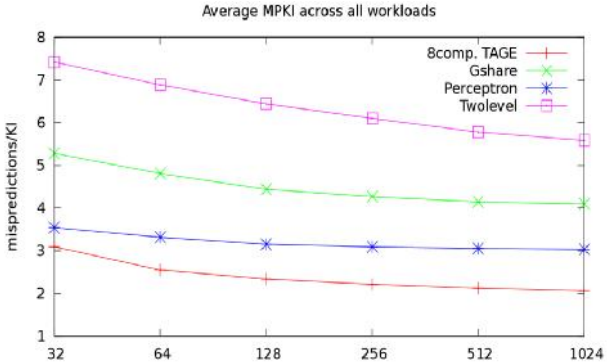


Figure 2

International Journal of Engineering Technology Science and Research
IJETSR
www.ijetsr.com
ISSN 2394 – 3386
Volume 4, Issue 8
August 2017

Figure 3 illustrates a P-TAGE predictor. It is similar to that of a TAGE predictor with the change that, instead of using a bimodal predictor for base prediction, perceptron predictor is used. Initialization, update policy and prediction mechanism is inherited from the TAGE predictor. Perceptron predictor is updated only when there is no tag match at all the tagged tables in which case the actual prediction is given by the perceptron predictor (given the same scenario, bimodal predictor would have been updated in TAGE). We compare the results of the 8-component TAGE with 8-component P- TAGE as it was shown earlier that 8-component TAGE outperforms other predictors.

**A 5 –component P-TAGE predictor**

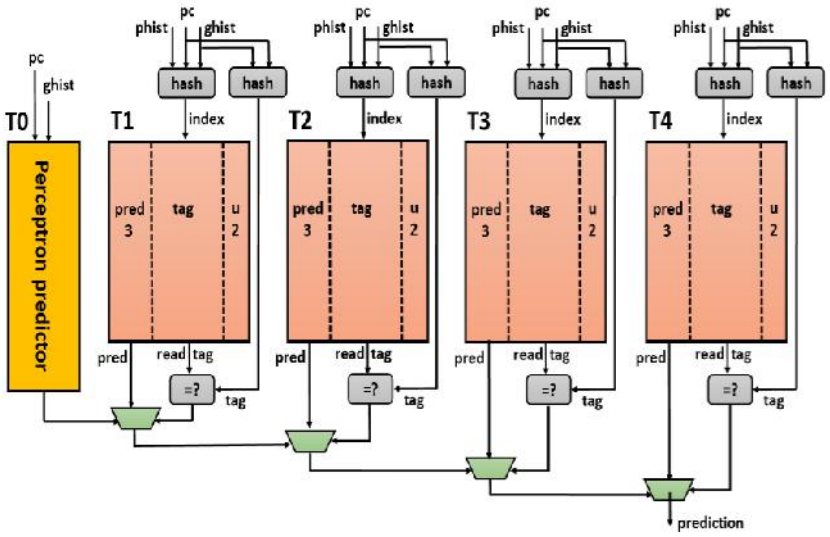


Figure 3

**Best amount of global history to be kept for perceptron predictor**

| Hardware budget in KB | History length: perceptron |
|---|---|
| 1 | 12 |
| 2 | 22 |
| 4 | 28 |
| 8 | 34 |
| 16 | 36 |
| 32 | 59 |
| 64 | 59 |
| 128 | 62 |
| 256 | 62 |
| 512 | 62 |
| 1024 | 62 |

Table 2

Given the budget for perceptron predictor, how many entries are chosen for the predictor table and what is the history length considered. It was given in the Table 1, the history lengths to be considered for a given hardware budget for perceptron predictor, which directly determines the number of entries. In P-TAGE for a given budget for the base perceptron predictor, the history length chosen is that of the entry matching the closest budget in the Table 2.

International Journal of Engineering Technology Science and Research
IJETSR
www.ijetsr.com
ISSN 2394 – 3386
Volume 4, Issue 8
August 2017

## EXPERIMENTAL RESULTS:

In this section, first we compare the experimental results of the perceptron predictor with the TAGE predictor for lower hardware budgets. Then we compare the experimental results of the 8-component P-TAGE predictor against 8-component TAGE predictor for hardware budgets in the range 4Kb-1024Kb for integer, floating point, multimedia and server traces. Figure 4 illustrates the results of the accuracy of the perceptron predictor versus 8-component TAGE predictor in the hardware budget range 4Kb-1024Kb. It is clearly seen that for lower budgets in the range 4Kb- 32Kb, perceptron predictor performs close to 8-component TAGE predictor.

Figure 5 illustrates the accuracy of the 8-component P-TAGE and 8- component TAGE against all workloads. The prediction accuracy for 8- component P-TAGE outperforms that of 8-component TAGE for 4Kb and 8Kb hardware budgets. Moreover, the prediction accuracy of P-TAGE is very close to that of TAGE across all other budgets.

Figure 6 illustrates the accuracy of the 8-component P-TAGE predictor against 8-component TAGE predictor for floating point traces. It can be seen that for the hardware budgets 4Kb-64Kb, 8-component P-TAGE out-performs 8-component TAGE predictor. For 128Kb budget, P-TAGE predictor performs close to TAGE predictor after which the accuracy of P-TAGE degrades.

Figures 7, 8 and 9 illustrate the accuracy of the 8-component P- TAGE predictor against 8-component TAGE predictor for integer, multimedia and server traces respectively. For integer workloads (Fig. 6),4Kb and 8Kb P-TAGE outperforms TAGE. For 16Kb-1024Kb budget, P-TAGE performs very close to TAGE predictor. Similar results can be seen for server workloads (Fig. 7). For multimedia workloads (Fig. 8), except for16Kb-64Kb hardware budget, the accuracy of P-TAGE is close to that of TAGE.

**Perceptron Vs 8-component TAGE branch prediction accuracy against all workloads**

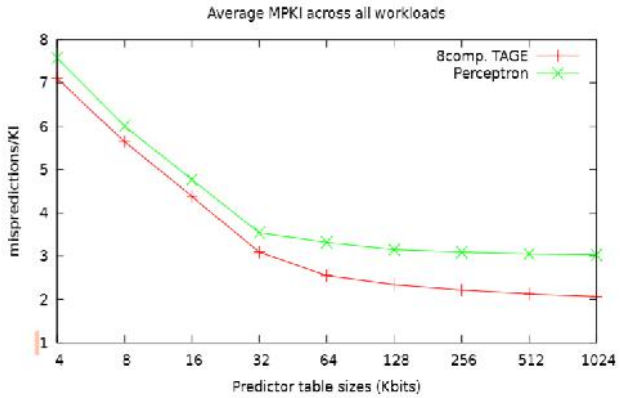


Figure 4

**Consolidated P-TAGE Vs TAGE branch prediction accuracy against all workloads**

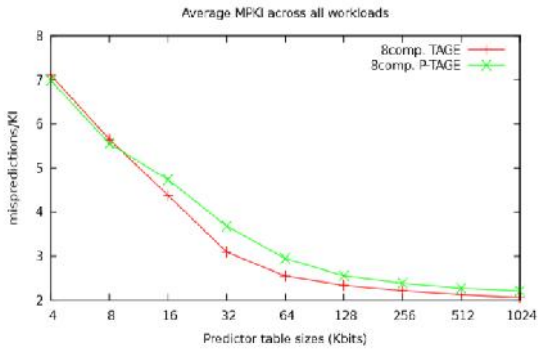


Figure 5

International Journal of Engineering Technology Science and Research
IJETSR
www.ijetsr.com
ISSN 2394 – 3386
Volume 4, Issue 8
August 2017

**P-TAGE Vs TAGE branch prediction accuracy against floating point workload**

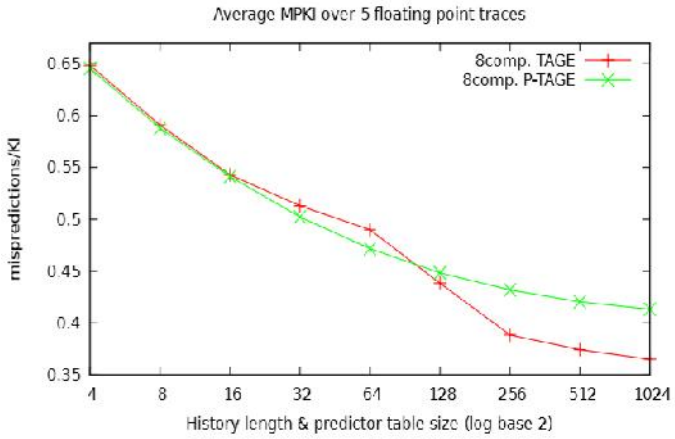


Figure 6

**P-TAGE Vs TAGE branch prediction accuracy against integer workload**

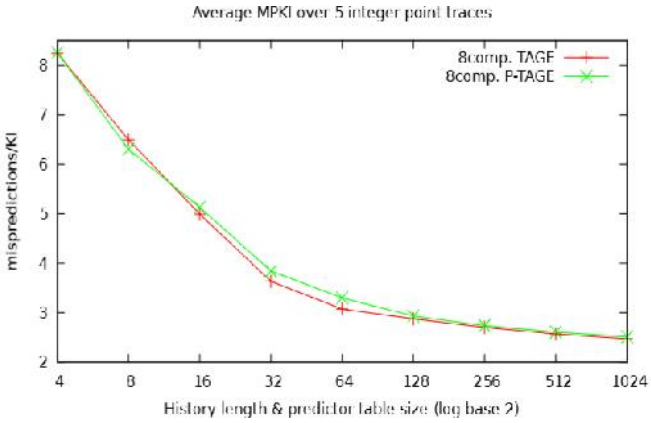


Figure 7

**P-TAGE Vs TAGE branch prediction accuracy against server workload**

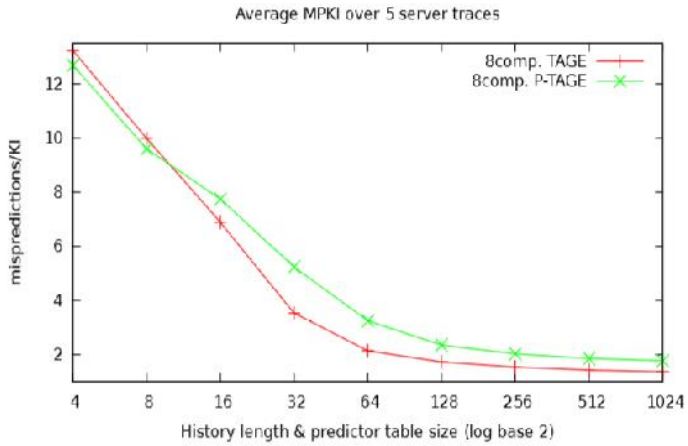


Figure 8

International Journal of Engineering Technology Science and Research
IJETSR
www.ijetsr.com
ISSN 2394 – 3386
Volume 4, Issue 8
August 2017

**P-TAGE Vs TAGE branch prediction accuracy against multimedia workload**

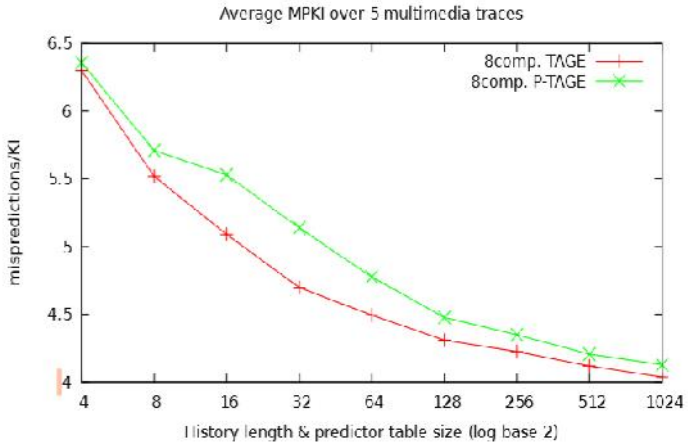


Figure 9

**CONCLUSIONS:**

Pipelining is a powerful implementation technique to enhance the system throughput without the requirement of massive replication of hardware resources. However, the efficiency of pipelining depends upon the efficient handling of pipeline hazards of which, branch prediction is one of the most successful and sought-after techniques to overcome control hazard caused by uncertainty of instruction execution path. Moreover, branch prediction attained greater importance to increase pipeline throughput and processor efficiency with the advent of super scalar pipelines. In spite of significant research in the field of branch prediction, microprocessor industry is heading towards zero cycle latency branch prediction with 100% accuracy which is not the reality. In this work, we have made an honest effort to address some of the issues in this direction for predicting conditional branches efficiently. We used Intel championship branch prediction (CBP) simulator [1] and the traces provided along with [2] to evaluate our implemented predictors. We first implemented Gshare predictor [3] and two-level adaptive training predictor [4] to gain deeper understanding of the evaluation framework. Gshare predictor performed better than two-level predictor. Then a perceptron neural predictor [5] was implemented whose results were better than Gshare and two-level adaptive predictors. Tagged geometric history length predictor (TAGE) [6], a state-of-the-art predictor for conditional branches was then implemented. We improvised the prediction efficiency of TAGE through the effective use of hysteresis bits for the base bimodal predictor. TAGE out-performed perceptron, Gshare and two-level predictors. We proposed perceptron-TAGE (P-TAGE) which uses perceptron predictor as the base predictor in TAGE. The block diagram of the P-TAGE can be seen in Figure 2. P-TAGE outperformed TAGE for hardware budgets 4Kb and 8Kb for generic workloads. P-TAGE performs better than TAGE for floating point workloads for hardware budgets in the range 4Kb-64Kb. The codes implemented as part of this work are pushed onto GitHub and can be pulled from this link.

**FUTURE WORK:**

- In our work, we employ a generic predictor for the given workloads. This can be extended to specialized predictors for specific workloads, say one for floating point traces, one for integer traces etc. exploiting the apriori information available.
- Currently our P-TAGE predicts conditional branches accurately at lower budget. However, P-TAGE can be extended to efficiently predict indirect branches.
- Perceptron predictor can be improved to predict linearly inseparable branches thereby improving P-TAGE predictor accuracy for conditional and indirect branch prediction.

- CBP simulator is a trace-driven simulator. The implemented predictors, especially P-TAGE can be ported to execution-driven simulators in which, the time taken to predict becomes a deciding factor from the perspective of processor fabrication.

**REFERENCES:**

[1] CBP2, http://www.jilp.org/cbp2014/.
[2] traces, http://www.jilp.org/cbp/traces-with-values/.
[3] Bhavya Daya, "Analysis of combined bimodal and gshare branch prediction schemes", Tech. Rep., University of Florida, 2005.
[4] Tse-Yu Yeh and Yale N Patt, "Two-level adaptive training branch pre- diction," in Proceedings of the 24th annual international symposium on Microarchitecture. ACM, 1991, pp. 51-61.
[5] Daniel A Jiménez and Calvin Lin, "Dynamic branch prediction with perceptrons", in High-Performance Computer Architecture, 2001. HPCA. The Seventh International Symposium on. IEEE, 2001, pp. 197-206.
[6] André Seznec and Pierre Michaud, "A case for (partially) tagged geo- metric history length branch prediction", Journal of Instruction-Level Parallelism, vol. 8, no. 1, pp. 1-23, 2006.
[7] Scott McFarling, "Combining branch predictors", Tech. Rep., Technical Report TN-36, Digital Western Research Laboratory, 1993.