View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by University of Hertfordshire Research Archive

Unbiased Branches: An Open Problem

Arpad Gellert[1], Adrian Florea[1], Maria Vintan[1], Colin Egan[2], and Lucian Vintan[1]
[1] Computer Science Department, "Lucian Blaga" University of Sibiu, Emil Cioran Street, No. 4, 550025 Sibiu, Romania
{arpad.gellert, adrian.florea, lucian.vintan}@ulbsibiu.ro
[2] School of Computer Science, University of Hertfordshire, Hatfield, College Lane, AL10 9AB UK,
c.egan@herts.ac.uk

Abstract. The majority of currently available dynamic branch predictors base their prediction accuracy on the previous *k* branch outcomes. Such predictors sustain high prediction accuracy but they do not consider the impact of unbiased branches, which are difficult-to-predict. In this paper, we evaluate the impact of unbiased branches on prediction accuracy. In this paper we evaluate the impact of unbiased branches on a range of branch difference predictors using prediction by partial matching, multiple Markov prediction and neural-based prediction. Our simulation results, with the SPEC2000 integer benchmark suite, are interesting even though they show that unbiased branches still restrict the ceiling of branch prediction and therefore accurately predicting unbiased branches remains an open problem.

Keywords: branch prediction, unbiased branch, branch difference value prediction

1. Introduction

In a previous paper [1] we showed that a branch in a certain dynamic context is difficult-to-predict when that branch is unbiased and its outcomes are non-deterministically shuffled. A branch is unbiased if its behaviour does not demonstrate a tendency to either the taken or the not taken path. We quantified and demonstrated that the percentages of difficult-to-predict branches in the SPECcpu2000 benchmarks suite [2] are significant (averaging between 6% and 24%, depending on the type of branch prediction context and the prediction context length). We considered the ceiling of history context-based prediction to be around 94% if the feature set length of 28 bits is used. Furthermore, we showed that many current state-of-the-art conventional branch predictors are unable to accurately predict these unbiased branches. This is because current branch predictors only use a limited amount of prediction information, such as local- or/and global-correlations and path-based information. The use of such limited information means that unbiased branches cannot be predicted to a high degree of accuracy. Consequently, other information is

required to predict branches which have been classified as unbiased. In this paper we investigate the use of a branch condition sign. The condition sign can be either positive, negative or zero. The condition sign is the difference between the data operands held within each source register. For example, a positive condition sign is computed if the datum in the first source register is greater than the datum in the second source register, and vice-versa for a negative condition sign, and zero if the data show equality. We show that branch behaviour is predictable by predicting the condition sign, but the impact of unbiased branches remains significantly high.

2. Related Work

Smith [3] showed that the majority of mispredicted branches come from few static branches. He also showed that a context-predictor where the last '*n*' (as low as 2) data values produced or consumed are used in combination with a closing outer-loop counter can achieve better prediction accuracy than a conventional *gshare* predictor.

Heil [4] introduced the idea of a Branch Difference Predictor (BDP) which simply holds branch source register differences. Heil used these data-value differences as inputs into a Rare Event Predictor (REP). The Rare Event Predictor was used to predict difficult-to-predict branches and the majority of easy-to-predict branches were predicted with a conventional *gshare* predictor. In Heil's study a difficult-to-predict branch was a branch that was mispredicted by a conventional *gshare* predictor. In contrast to Heil, we define in [1] a difficult-to-predict branch to be a branch with a low degree of polarisation since that tends to shuffle between taken and not-taken and is therefore unbiased. Heil used the differences in the register data values as inputs to the REP (up to a maximum of 3 value differences), whereas in our study we use the sign of the differences (up to a history of 256 sign differences) between the register data values. We therefore use less storage and our simulation results show that we achieve better prediction accuracy.

In [5], González introduced the concept of branch prediction through value prediction (BPVP). The idea was to pre-compute a branch outcome by speculatively predicting the source operand as each branch is dynamically encountered. González prediction strategy was to use a conventional *gshare* in conjunction with a BPVP. The inclusion of the BPVP was to predict the branches that were difficult-to-predict by the conventional *gshare* predictor. González therefore has a similar approach to Heil.

Vintan [6] proposed pre-computing branches by determining a branch outcome as soon that branch's operands were available. The basis behind such pre-computation was that the instruction that produced the last branch source operand would also trigger the branch condition estimation. This means that as soon as this operation was completed then the branch outcome could be immediately resolved. Even though this concept would provide (almost) perfect prediction accuracy, there was a heavy timing penalty in the case when a branch instruction is dynamically executed immediately after the last source operand has been computed, in fact this is a common case.

Gao [7] implemented a Prediction by Partial Matching (PPM) predictor that predicts branch outcomes by combining multiple partial matches through an adder

tree. The Prediction by combining Multiple Partial Matches (PMPM) algorithm selects up to *L* confident longest matches and sums the corresponding counters that are used to furnish a prediction. A bimodal predictor is used to predict branches that are completely biased (either always taken or always not taken) and the PMPM predictor is used to furnish a prediction when a branch is not completely biased. In this study we also implement a PPM predictor, but our PPM predictor has three significant differences. First, our Branch Difference Prediction by Combining Multiple Partial Matches (BPCMP) furnishes predictions for unbiased branches as described in our previous work in [1, 8] not on completely biased branches. Second, in Gao's study global branch history information was used, whereas we use a combination of global and local branch difference history information. Finally, Gao used an adder tree algorithm to combine multiple Markov predictions, we use one of two voting algorithms.

Jiménez [9] proposed a neural predictor that uses fast single-layer perceptrons. In his first perceptron-based predictor the branch address is hashed to select the perceptron, which is then used to furnish a prediction based on global branch history. Jiménez [10] furthered his work by developing a perceptron-based predictor that uses both local and global branch history information. We also evaluate a perceptron-based predictor, but unlike Jiménez our inputs are based on global and local branch operand difference information. In [11] Jiménez developed a piecewise linear predictor using a piecewise linear function the idea being to exploit different paths leading to the branch undergoing prediction. We have also evaluated a piecewise linear predictor using global and local branch operand difference information as inputs as described in [12].

3. Unbiased Branches

In [1] we define an unbiased branch to be a branch that does not demonstrate a bias to either the taken or the not taken path which means unbiased branches show a low degree of polarisation towards a specific prediction context (by which we mean, a local prediction context or a global prediction context or a path-based prediction context) and are therefore difficult-to-predict by that particular prediction context.

We also identified branches that were unbiased on their local and global history contexts and, on their global history XORed with the branch address. Our results showed that even with a feature set length of 28 bits the number of unbiased branches remained significantly high at just over 6%. We therefore considered the ceiling of history context-based prediction to be around 94%.

3.1. Condition-History-Based Branch Prediction Using Markov Models

A context-based predictor [13] predicts the next datum value based on a particular stored pattern that is repetitively generated in the values' sequence. This means that, a context-based predictor could predict any stochastic repetitive sequence. Value predictors that implement the PPM algorithm represent an important class of context-based predictors. In a PPM predictor, if a prediction cannot be furnished by order *k*

then the pattern length is shortened and the Markov predictor of order *k-1* is used to furnish the prediction and if this order cannot furnish a prediction the order is further reduced to *k-2* and so on until either a prediction is furnished or the Markov predictor is of the order *0*.

3.1.1. Local Branch Difference Predictor

In Figure 1 we show the mechanism of our local PPM Branch Difference Predictor. The Branch Difference History Table (BDHT) is indexed by the branch address (B_0). In the case of a hit in the BDHT, the last *h* dynamic source operand differences are furnished. To save storage space, sign operand differences are recorded as +1, -1 or 0. For each dynamic branch encountered, a positive difference is recorded if the first source operand is greater than the second, a negative difference is recorded if the first source operand is less than the second source operand and zero is recorded if both operands are the same. The *h* difference fields of the BDHT entry are then used as inputs into our complete-PPM predictor. The PPM predictor furnishes the predicted sign value of the branch undergoing execution (B_0) of order *k*, where $k<h$. Speculative execution of the branch (B_0) only occurs in the case that the pattern length *k* is repeated in the last *h* differences with a frequency greater than or equal to a threshold value.

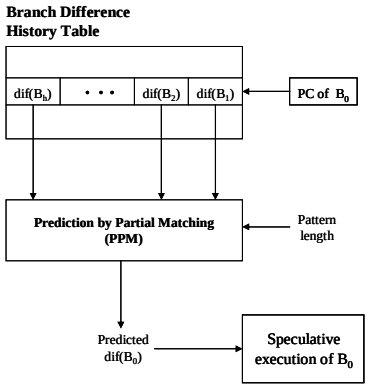

Figure 1. A local complete-PPM branch-difference predictor

3.1.2 Combined Global and Local Branch Difference Predictor

Figure 2 shows the branch prediction mechanism using a combined global and local PPM-based branch-difference predictor. The Global History Register (GHR) contains the global branch difference history pattern. Every global branch history pattern has its own BDHT and the GHR history pattern is used as an index to its BDHT. Each BDHT is configured as a local BDHT and is accessed as described in section 3.1.1.

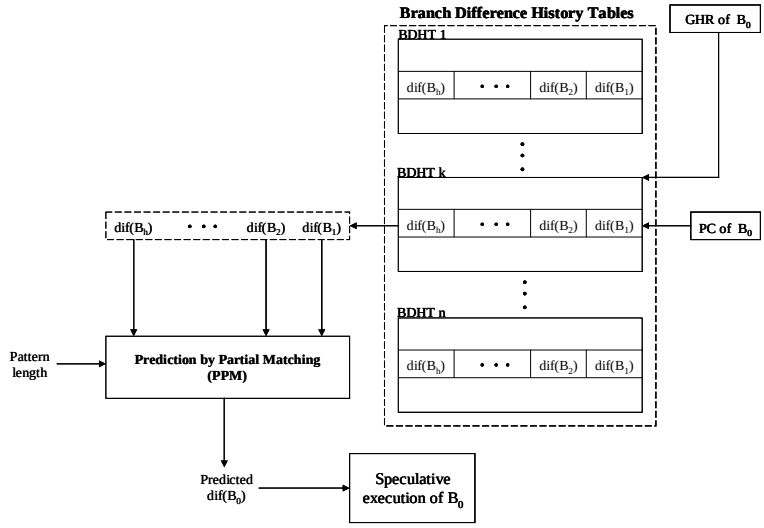

Figure 2. A global and local complete-PPM branch-difference predictor

3.1.3. Branch Difference Prediction by Combining Multiple Partial Matches

Figure 3 shows our branch prediction mechanism using the Branch Difference Prediction by Combining Multiple Partial Matches (BPCMP). An entry in the BDHT is accessed by the method described in section 3.1.1, but now the *h* branch differences are used as inputs into multiple Markov predictors of different orders (*n* where $n < h$). Each Markov predictor furnishes a predicted sign value (+1, -1 or 0) and these multiple predictions are passed to a voter. The final value prediction is then furnished as the greatest sign frequency that was input into the voter.

We have also investigated a confidence-based voting mechanism. The function field of each entry in the BDHT holds *n* saturated confidence counters, in the range -4 to +4, which are associated with the *n* Markov predictors. For a pattern length *k*, where $1 \leq k \leq n$, the Markov predictors will furnish a value prediction if that repeating pattern is stored at least once in its *h* history values. In the case of a correctly predicted branch, its confidence saturating counter is incremented and decremented in the case of a misprediction. The Markov prediction is then replicated to match its confidence counter, so long as that confidence counter is >0. These multiple value predictions are then passed to the voter, which furnishes the most frequent value prediction.

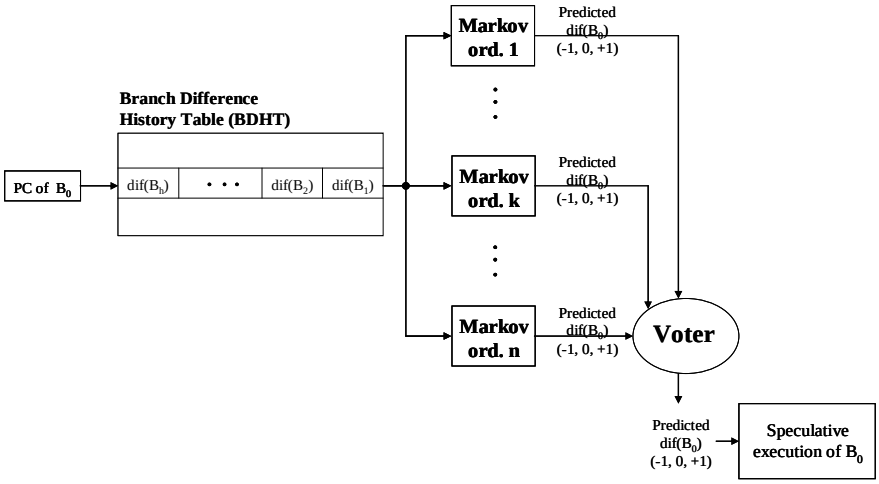

Figure 3. Multiple Markov branch-difference prediction

4. Simulations

We have developed a number of simulators (as described in section 3) which extend the sim-bpred simulator provided in SimpleSim-3.0 [14]. We also include implementations to identify unbiased branches as presented in [1, 8]. We have evaluated our simulators using the unbiased branches we identified in [1] on the SPEC2000 benchmark suite [2]. All simulation results are reported on 1 billion dynamic instructions skipping the first 300 million instructions.

4.1. Local Branch Difference Prediction

We set out to determine the optimal local branch difference predictor. We asked ourselves 5 questions. Would the operand sign value difference algorithm achieve better prediction accuracy than the operand value difference? Which local history register length would provide the best prediction accuracy? Which pattern length would achieve the best prediction accuracy? What is the most suitable threshold value? What is the ideal number of local BDHT entries?

In Figure 4 we answer the first two questions: What would be the most suitable sign algorithm to use and, which history register length achieves the best prediction accuracy? We identified unbiased branches the same way as in our previous work [1], and we evaluated the impact of these unbiased branches using a complete PPM predictor with a local BDHT. The BDHT we used was sufficiently large to ensure that every static branch had its own entry thereby eliminating any possibility of collisions. The pattern length was set to 3, the threshold value was set to 1, and the local history register length was varied from 8-signs to 64-signs in

increments of 8. Our results show that better prediction accuracy is achieved by the operand sign difference algorithm rather than the operand value difference algorithm and that beyond a local history register length of 24-signs there is only marginal improvement in prediction accuracy.

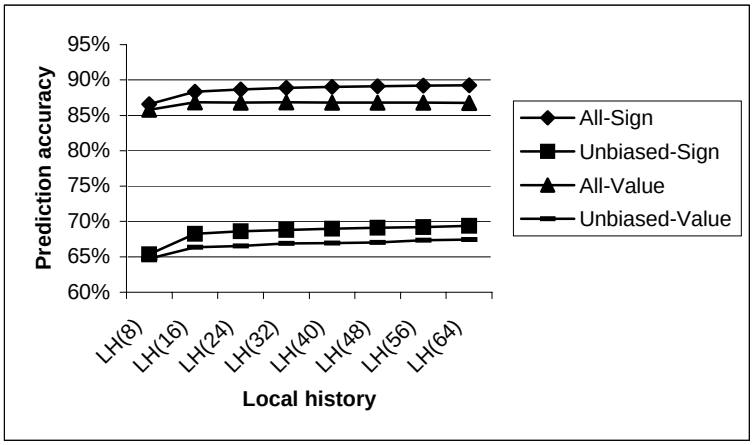

Figure 4. Average difference prediction accuracy with increasing local history register length of the sign difference and operand difference algorithms

In Figure 5 we answer the third question: Which pattern length would achieve the best prediction accuracy? We used a complete PPM predictor with the operand sign difference algorithm, a local history register length of 24-signs and the threshold value was set to 1. Our results show that initially prediction accuracy improves with increasing pattern length and then decreases and these results confirm that our original pattern length of 3 achieves the best prediction accuracy.

In Figure 6 we answer the fourth question: What is the most suitable threshold value? We used the same parameters as Figure 5, but the pattern length was now set to 3 and the threshold value varied. Our results show that prediction accuracy improves with an increasing threshold value, but there is marginal, if any, benefit of increasing the threshold value beyond 7.

In Figure 7 we answer final the question: What would be the optimal number of entries in the local BDHT? We used the same parameters as Figure 6, and the number of entries in the local BDHT was varied from 64 entries to 256 entries in increments of 64. We also include an unlimited local BDHT. Our results show that the impact of the so called 3Cs (capacity, collisions and cold-start) to be minimal with a 256 entry local BDHT and that there is minimal prediction accuracy gain by increasing the number of entries beyond 256 entries where the increased number of cold-start mispredictions may impact on prediction accuracy.

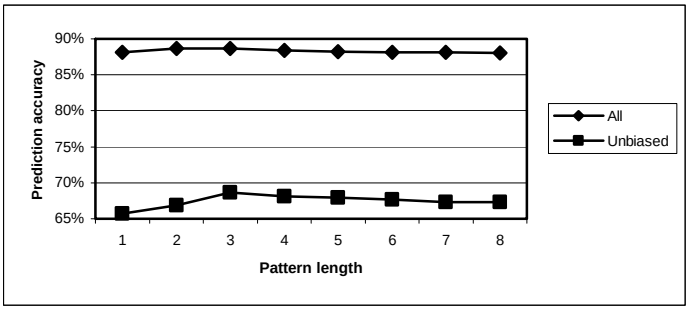

Figure 5. Average difference prediction accuracy with increasing pattern length

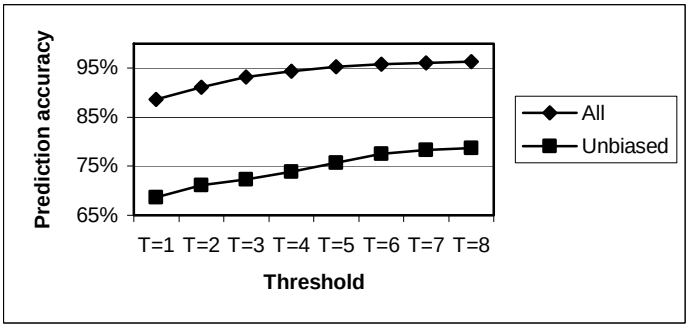

Figure 6. Average difference prediction accuracy with increasing threshold value

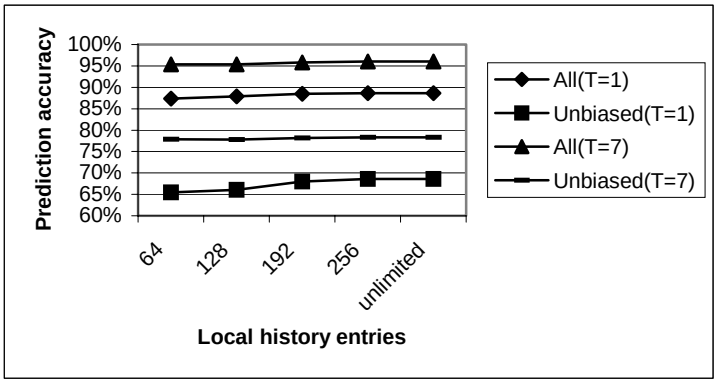

Figure 7. Average difference prediction accuracy with an increase in the number of local BDHT entries

We investigated the branch prediction accuracies of the individual SPEC2000 benchmarks using our optimal local branch difference predictor. We used the operand sign difference algorithm, with a local history register length of 24-signs, a pattern length of 3, and we use a local 256 entry BDHT. In our results we compare two

threshold values, 1 and 7. When the threshold value is 1, we achieve an average branch prediction accuracy of 90.55% and the unbiased branches have an average branch prediction accuracy of 71.76%. When the threshold value is increased to 7, we achieve an average branch prediction accuracy of 96.43% and the unbiased branches have a prediction accuracy of 76.69%. These results show the significance of the threshold value on prediction accuracy and the impact of unbiased branches. Consequently, unbiased branches in this local context remain difficult-to-predict.

4.2. Combined Global and Local Branch Difference Prediction

We consider the high number of unbiased branches and their impact on prediction accuracy to be due to their high degree of shuffling. To alleviate the problem of shuffled branch behaviour of unbiased branches we have developed a combined global and local branch difference predictor which will convert an unbiased branch in a local context into a biased branch in a global context, and therefore a difficult-to-predict branch in a local context will be an easy-to-predict branch in a global context.

In our global and local branch difference predictor, each global history register pattern is used to point to its own local BDHT as described in section 3.1.1 and shown in Figure 2. Consequently, we restrict the global history register length to a maximum of 4-signs. The parameters of each of the local BDHTs were the same as those which achieved the results shown in Figure 7, except we used a 256 entry BDHT.

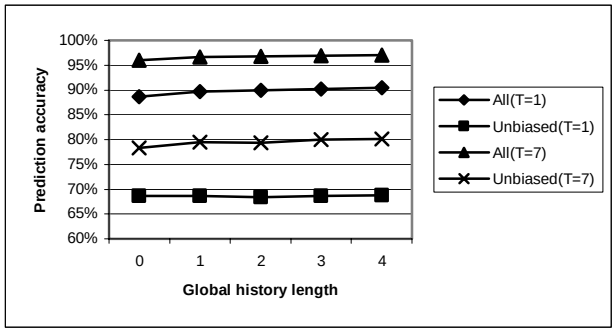

Figure 8. Combined global and local difference prediction accuracy

In Figure 8 the global history register length of 0 represents the optimal local branch difference predictor whose results are provided in Figure 7, with a 256 entry BDHT. With the combined global and local difference predictor, as the global history register length is increased there is a marginal improvement in prediction accuracy. With a global history register length of 4-signs and a threshold value of 1, the combined global and local branch difference predictor achieves an average prediction accuracy of 90.47%, but the unbiased branches only achieve an average prediction accuracy of 68.81% showing a marginal improvement over the local branch difference predictor. When the threshold value is increased to 7, the average prediction accuracy improves to 97.44% and the average prediction accuracy of unbiased branches is still significant

at 81.25%. Even though there is some improvement in prediction accuracy, these results show that the impact of unbiased branches still remains significant and therefore implies that alternative approaches are required.

4.3. Branch Difference Prediction by Combining Multiple Partial Matches

Our first alternative approach was to develop a branch difference predictor using five Markov predictors of orders ranging between 1 and 5 (as described in section 3.1.3 and shown in Figure 3). Again, we use a 256 entry local BDHT, a local history register length of 24-signs; we compare the prediction accuracy of two voting algorithms, a simple voting algorithm and a confidence voting algorithm. Our results show that the average prediction accuracy of the confidence voting algorithm is marginally better than the simple voting algorithm, as shown in Figure 9.

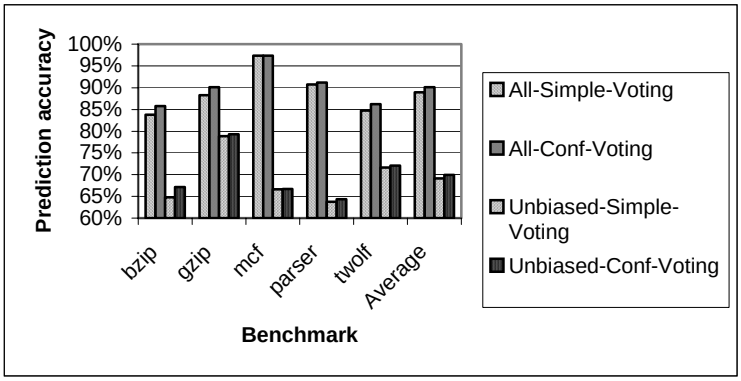

Figure 9. Difference prediction accuracies by combining multiple partial matches through simple voting and confidence-based voting

4.4. Neural-based Branch Difference Global and Local Prediction

In our second alternative approach we developed a family of neural-based branch difference predictors. Our neural predictors are fast single-layer perceptron predictors similar to those developed by Jiménez [9]. For a fair comparison with our 256 entries local BDHT we use a perceptron table with 256 entries. Our single-layer perceptron predictors use global history information only or local history information only or a combination of global and local history information.

To determine our optimal single-layer perceptron predictor, we vary the input history register lengths. Not surprisingly, the combination of global and local history information outperforms the other two predictors. We found that the best average prediction accuracy of 92.58% was achieved with a 40-global history signs combination with 28-local history signs. However, the unbiased branches still have a significant impact with an average prediction accuracy of 73.46%.

Finally, we considered the impact of unbiased branches on a piecewise linear predictor based on [11], but with sign difference inputs as described in [12]. We dynamically changed the global history input from 18- to 48-signs combined with local history input from 1- to 16-signs. We achieved an average prediction accuracy of 94.2% on all branches but the impact of unbiased branches still remained significant at 77.3%.

4.5. Comparing all of the optimal predictors

In Figure 10, we bring together the impact that unbiased branches have on all of the optimal predictors we have developed (local-PPM, combined-PPM and multiple Markov combined- perceptron and the piecewise linear branch predictor). Our results show that unbiased branches have a severe impact on all branch predictors and in all cases unbiased branches only have an average branch prediction accuracy of between 71.54% (local-PPM) and 77.3% (piecewise linear branch predictor).

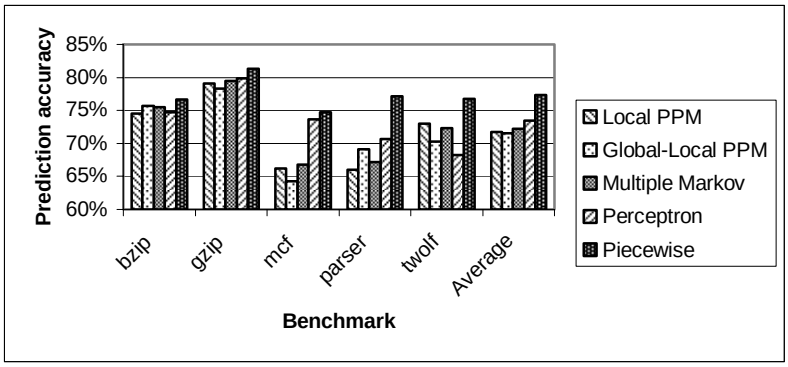

Figure 10. Branch prediction accuracy on unbiased branches

5. Conclusions

In this study we have validated our previous findings in [1, 8] that current start-of-the-art correlate either insufficient information or wrong information in the prediction of unbiased branches. This led us to consider alternative approaches: the branch difference predictors using PPM and multiple Markov predictors and Neural-based perceptron predictors. Our results show that unbiased branches still limit prediction accuracy even with these alternative approaches. The most effective branch predictor was the piecewise linear branch predictor, but even this predictor only achieved a prediction accuracy of 77.3% on the unbiased branches.

However, we have shown that the sign difference algorithm achieves better prediction accuracy than the operand difference algorithm. We also show that

combined global and local information achieves better prediction accuracy than global information alone or local information alone.

In our opinion, the most optimal local branch difference predictor uses the operand sign difference algorithm, with a local history register length of 24, a pattern length of 3, a threshold value of 7 and a local BDHT with 256 entries. This predictor achieves an average prediction accuracy of 96.43% on all branches but on the unbiased branches only achieve a prediction accuracy of 79.69%.

Also in our opinion, the impact of unbiased branches significantly restricts prediction accuracy. This means that accurate branch prediction of unbiased branches remains an open problem and such branches will continue to limit the ceiling of dynamic branch prediction. Perhaps an alternative mechanism might be to hand-shake scheduler support with dynamic branch prediction. The idea of the scheduler would be to remove as many branch instructions from the static code as possible and leave the remaining branches to be dynamically predicted.

References

[1] Vintan L., Gellert A., Florea A., Oancea M., Egan C., Understanding Prediction Limits through Unbiased Branches, Lecture Notes in Computer Science, vol. 4186-0480, Springer-Verlag, ISSN 0302-9743, Berlin Heidelberg, (2006), pp. 480-487.

[2] SPEC2000, The SPEC benchmark programs, http://www.spec.org.

[3] Smith Z., Using Data Values to Aid Branch-Prediction, MSc Thesis, Wisconsin-Madison, USA, (1998), pp. 1-103.

[4] Heil T.H., Smith Z., Smith J.E., Improving Branch Predictors by Correlating on Data Values, The 32nd International Symposium on Microarchitecture, (1999), pp. 28-37.

[5] González J., González A., Control-Flow Speculation through Value Prediction, IEEE Transactions on Computers, Vol. 50, No. 12, (2001), pp. 1362-1376.

[6] Vintan L., Sbera M., Mihu I.Z., Florea A., An Alternative to Branch Prediction: Pre-Computed Branches, ACM SIGARCH Computer Architecture News, Vol.31, Issue 3, ISSN: 0163-5964, ACM Press, NY, USA, (2003), pp. 20-29.

[7] Gao H., Zhou H., PMPM: Prediction by Combining Multiple Partial Matches, The 2nd Journal of Instruction-Level Parallelism Championship Branch Prediction Competition (CBP-2), Orlando, Florida, USA, (2006), pp. 19-24.

[8] Gellert A., Prediction Methods Integrated into Advanced Architectures, Technical Report, Computer Science Department, "Lucian Blaga" University of Sibiu, (2006), pp. 37-70.

[9] Jiménez D., Lin C., Dynamic Branch Prediction with Perceptrons, In Proceedings of the Seventh International Symposium on High Performance Computer Architecture (HPCA-7), (2001), pp. 197-206.

[10] Jiménez D., Lin C., Neural Methods for Dynamic Branch Prediction, ACM Transactions on Computer Systems, Vol. 20, No. 4, New York, USA, (2002), pp. 369-397.

[11] Jiménez D., Idealized Piecewise Linear Branch Prediction, Journal of Instruction-Level Parallelism, Vol. 7, (2005), pp. 1-11.

[12] Gellert A., Integration of some advanced prediction methods into speculative computing systems, Technical Report, Computer Science Department, "Lucian Blaga" University of Sibiu, (2007), pp. 1-40.

[13] Sazeides Y., Smith J. E., The Predictability of Data Values, Proceedings of the 30th Annual International Symposium on Microarchitecture, (1997), pp. 248-258.

[14] Simplescalar, The SimpleSim Tool Set, ftp://ftp.cs.wisc.edu/pub/sohi/Code/simplescalar.