THE PUBLISHING HOUSE
OF THE ROMANIAN ACADEMY

PROCEEDINGS OF THE ROMANIAN ACADEMY, Series A,
Volume 5, Number 3/2004, pp. 000-000

VALUE PREDICTION AND SPECULATION INTO THE NEXT MICROPROCESSORS GENERATION

Lucian N. VINȚAN

"*Lucian Blaga*" University, Computer Science Department, Emil. Cioran Str., No. 4, Sibiu-550025, ROMANIA,
Tel./Fax: +40-269-212716, E-mail: Lucian.Vintan@ulbsibiu.ro

Value Prediction (VP) is a relatively recent developed processing technique. Its main aim is to early predict the instructions' results, during their fetch or decode pipeline stages, and, therefore, to speculatively execute the instruction flow. The predicted value can then be used as an input to some subsequent dependent instructions so that they can execute earlier. If the prediction is incorrect, recovery mechanism must be employed to squash speculative results and re-execute all instructions that used the incorrectly predicted values. Through this paper we briefly presented this emergent paradigm and we investigated the value locality degree present in real-world programs, and extended the value locality and prediction concepts to all general-purpose registers (MIPS architecture). As a consequence, the encouraging obtained results facilitates implementation of much simpler prediction structures, significantly reducing the hardware cost and complexity.

Key words: advanced computer architectures, value locality, dynamic instruction value prediction, simulation, benchmarking.

1. INTRODUCTION

According to [15], Value Locality represents a third facet of the statistical locality concepts used in computer engineering (close by the well known, and used, temporal and spatial locality concepts). Value Locality means the statistical probability to produce a value belonging to the previous *k* value instruction's instances. The concept is strongly related with the redundant computing concepts including here the relatively recently introduced Dynamic Instruction Reuse techniques [25].

In my opinion, the main difference between temporal and spatial locality and, respectively, value locality consists in the fact that the first two metrics are focused on item's (data's or instruction's) address rather than on the instruction's result. More precisely, temporal locality means a significant probability that the current accessed memory address to be accessed again in the nearest future. Value locality means a bit more: the fetched instruction will produce a result belonging to the previous instances, with a significant probability, too. Therefore, it appears that value locality is a consequence of temporal locality. Obviously, value locality might be exploited through value prediction. Measurements using SPEC benchmarks show that value locality on Load instructions is about 50% using a history of one (producing the same value like the previous one) respectively 80%, using a history of 16 previous instances. The value locality can be exploited to bypass the redundant operation through value prediction or dynamic instruction reuse. This is especially beneficial when the operations that are on the critical path can be skipped [30].

2. GENERALISED VALUE PREDICTION

In [21] the authors developed an empirical practical classification related to the generated instruction values. According to them, there are 3 distinct types of instruction values: constant, incremental and non-incremental. Of course, we can imagine also some hybrid classes, based on these 3 main classes. Also, we can consider repetitive and, respectively, non-repetitive sequence values.

Recommended by Mihai DRĂGĂNESCU, member of the Romanian Academy

These classes involve at least two distinct main categories of predictors: computational and contextual. *Computational predictors* are predicting based on some previous values in a computational manner, therefore according to a deterministic recurrence formula. An incremental predictor belongs to the computational class. As it can be easy observed, a computational predictor doesn't directly derive from the value locality concept. A *contextual predictor* predicts the next value based on a particular stored pattern (context) that is repetitively generated in the value sequence. Theoretically contextual predictors can predict any repetitive sequences. A context predictor is of order *k* if its context information includes the last *k* values, and, therefore, the search is done using this pattern of *k* values length [30].

Computational Predictors

We'll briefly present here only the incremental predictors, the most used belonging to the computational class. Considering that v_{n-1} and v_{n-2} are the last generated values, the new value v_n will be calculated using the formula: $v_n = v_{n-1} + (v_{n-1} - v_{n-2})$, where $(v_{n-1} - v_{n-2})$ is the stride. Of course, the stride might be variable in time (constant only for some "real-time" periods). The stored stride is modified based on a confidence mechanism. The corresponding confidence counter is incremented/decremented according to the prediction result, respectively (correct/incorrect). Another hysteresis strategy is the so called "*2 – delta*" method. In this case there are stored two strides (s_1 and s_2), $s_1 = v_n - v_{n-1}$. Stride s_2 is currently used in the prediction process. Only when the same value is generated two times consecutively, it is done the transfer $s_2 \leftarrow s_1$. In the case of repetitive incremental sequences, both strategies reduced the number of wrong predictions from two, to one.

Contextual Predictors

In this case the prediction will be done based on the most frequent values that follow a pattern context in the string of history values. An important contextual predictor class implements, with some simplifications, the general "*Prediction by Partial Matching*" (PPM) algorithm (used also in speech recognition, data compression, etc.). PPM works on a set of *(k+1)* Markov predictors of order *k, k-1, k-2,…,2, 1 ,0* [18]. A *Markov predictor* might be represented by a graph of states and transitions between states. Each transition from state X to state Y in the corresponding graph has assigned a weight representing the fraction of all X references that are followed by a reference Y. Obviously, the sum of weights belonging to a certain state must be equal to 1 (normalisation equation). If the predictor of order *k* can't predict anything, it will be used the predictor of order *(k-1)* and so on, up to the 0th order predictor.

It's obvious that for any value sequence that can be generated through an algorithm, we can develop a corresponding predictor that will correctly predict the next value. Considering, for example, a loop that calculates *Fibonacci*'s recurrent string ($v_n = v_{n-1} + v_{n-2}$), it must be activated a corresponding hardware predictor, implemented by the previous formula. Unfortunately, it is not possible to predict non-repetitive and non-stride value sequences generated by some algorithms under the present-day VP paradigm (mainly consisting in hardware prediction structures accessed by information derived from the object code). In order to predict these difficult sequences, perhaps it's needed a new hardware-software interface implemented through a new Instruction Set Architecture - ISA. In other words, the compiler must transmit to the predictor, through the ISA, some High Level Language (HLL) semantic information, related to the HLL processed structures and processed application. Unfortunately, in the present-day approach, after compilation, the resulted object code, lose almost all of the HLL semantics. Therefore I believe it is strongly needed a **semantic predictor** based on a new code generation paradigm, on a new hardware-software interface that will include more HLL application semantics. This will be an important further challenge in dynamic VP domain.

Two Level Adaptive Value Predictors

Some authors proposed – through an analogy with the well-known Two Level Adaptive Branch Predictors [33] – some Two Level Adaptive Value Predictors [31]. The scheme presented in figure 1 stores in the first level (Value History Table - VHT) the last 4 values per each dynamic instruction. A selection mechanism might choose one of these 4 values as the predicted value. These 4 values are binary codified {00, 01, 10, 11}. While an instruction produces one of the 4 stored values, eventually one of these values will be predicted. Otherwise, the new produced value will be introduced in VHT and another value belonging to VHT will be evacuated (ex. through a LRU algorithm).

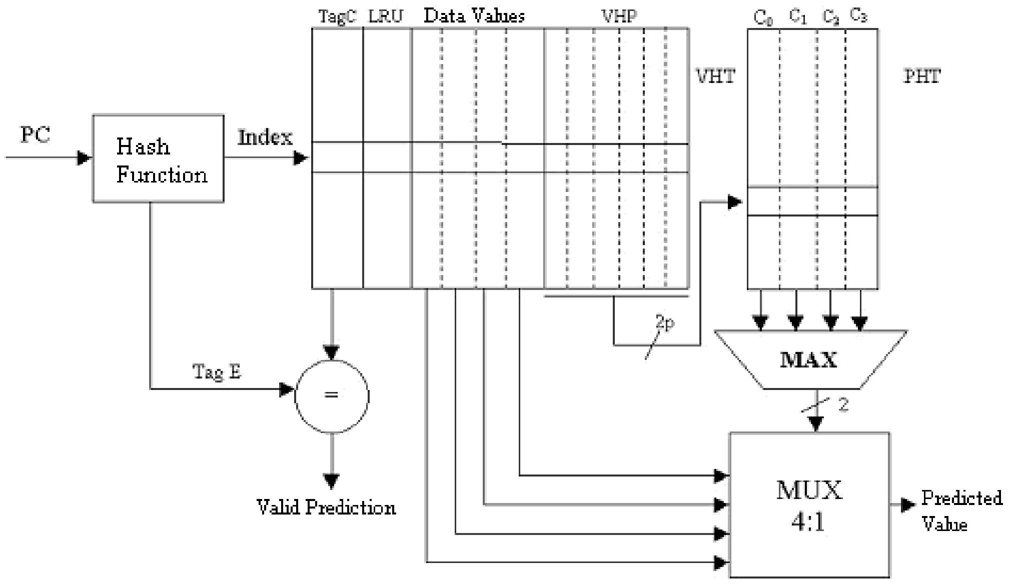

Figure 1. A Two Level Adaptive Value Predictor.

The VHP (*Value History Pattern*) field belonging to the VHT stores a binary pattern on *2p* bits. This pattern codes the last *p* results generated by that instruction (each result is coded using two bits). VHP will update like a shift logic left register, according to the corresponded predicted value. The VHP field address the PHT table (*Pattern History Table*). A PHT line contains 4 independent saturated counters (C_0, C_1, C_2, C_3). If the current instruction produces the value associated with the counter C_K, (k=0,1,2,3) belonging to the corresponding PHT line, then C_K is incremented (+3), and, the other 3 counters are decremented (-1). The MAX circuit works as follows: if MAX (C_0, C_1, C_2, C_3) = C_K and C_K >= A, where A is a certain threshold, then MAX circuit generates the binary value of *K,* codified on 2 bits. Otherwise, if C_K < A, it will be not generated any prediction (C_K value associated with the VHP pattern hasn't a sufficient higher frequency). It can be observed that – through the proposed codification mechanism - VHP index represents a compressed CPU context. This involves reasonable PHT capacities.

The scheme presented in figure 1 represents a simplified feasible implementation of the PPM generic predictor. Each instruction has associated its own context stored in VHT, representing the last *p* binary codified values produced by this instruction (VHP pattern). Using this pattern, the Two Level Predictor calculates – using C_0 to C_3 counters – the most frequent generated value in the last *p* instances of the instruction. Now it's clear why this predictor belongs to the most general PPM contextual predictor, previously described. On the other hand, someone might answer: why the VHT structure stores only the last 4 instruction's values? There is a statistical explanation: 15% - 45% from the instructions produce only one value in their last 16 dynamic instances and 28% - 67% produce maximum 4 distinct values considering their last 16 instances.

A particular type of predictor predicts some instruction values better than other types. Therefore, the hybrid prediction idea seems to be very natural in order to obtain the optimal trade-off and synergism, too. That means two or many predictors working together, selected in a dynamic adaptive manner [31].

3. SOME RESULTS OBTAINED THROUGH SIMULATION

We are presenting now some simulation results obtained using a dedicated simulator developed at University "*L. Blaga*" of Sibiu. All the simulations used *SimpleScalar Tool ver. 3.0* and SPEC benchmarks [1], [27]. There were simulated 5.000.000 dynamic instructions for each SPEC benchmark. We performed several experiments to evaluate the value locality exhibited by MIPS CPU integer registers (unstudied until now as far as we know). Based on this, we further developed some predictors associated with the favourable registers (having important value locality degree). As it can be seen, these predictors are inspired from those described in the previous paragraph.

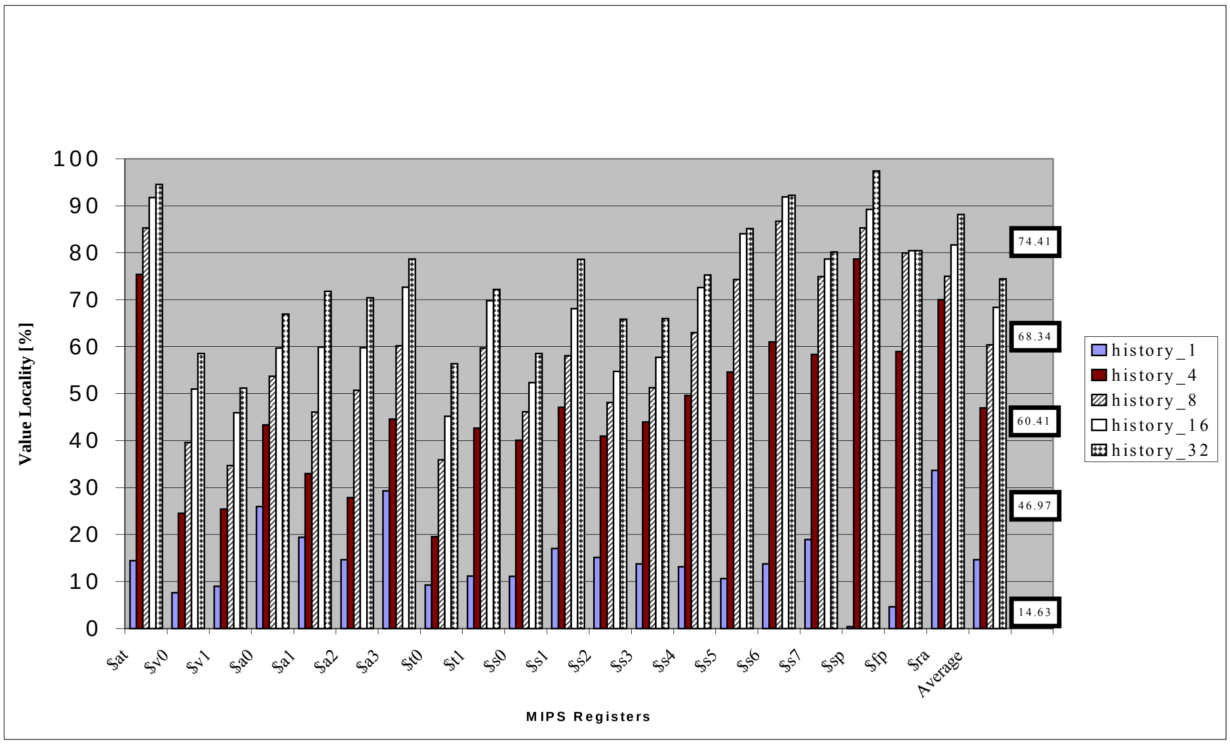

Figure 2. Value Locality from a Register Centric Point of View (MIPS Processor).

Figure 2 presents some value locality degrees centred on MIPS CPU registers. As far as we know this is the first value locality measurement centred on general processor registers. As it can be observed, the value locality degree is high for some registers. Taking into account the simulated MIPS CPU architecture, some possible qualitative explanations are:

- $at register is frequently used in data memory address in order to access some data structures.
- $sp (29) register is the MIPS stack pointer and, thus, it has little variations.
- $fp (30) is the frame pointer register.
- $ra (31) stores the return address for a *jal* (*jump and link*) instruction. There are few calling points in SPEC benchmarks.

These results encouraged us in implementing a small predictor centred on the registers rather than on the instructions, with some important cost / complexity advantages [10]. The predictor's table has only 32 value predictors instead of thousands of predictors required by the instruction centric paradigm. Figure 3 presents the prediction value accuracies obtained through a hybrid predictor (PPM & Incremental).

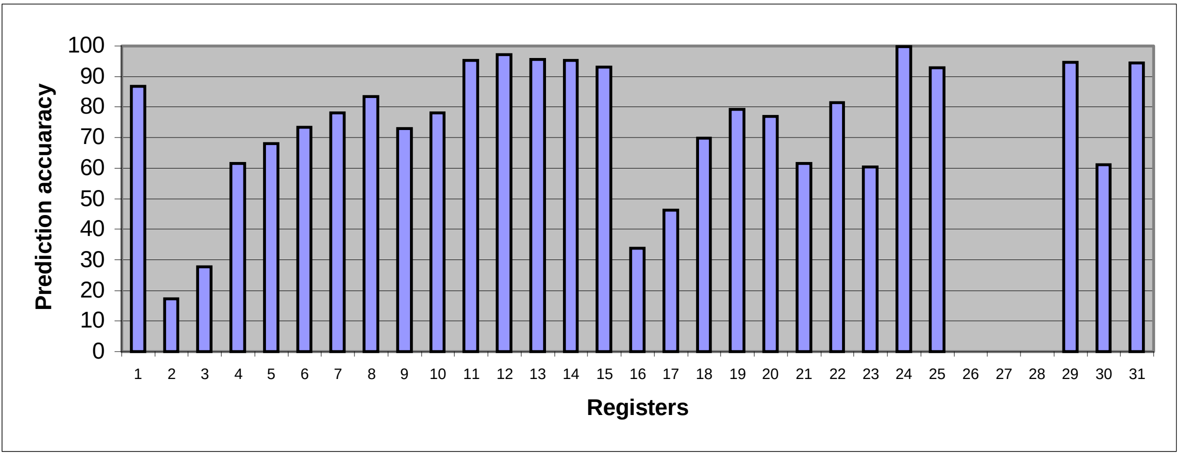

Figure 3. Value Prediction Accuracy on MIPS Registers using a hybrid PPM – Incremental Predictor (256 history values, search pattern length = 4).

Considering now only the favourable registers (18 registers having a prediction accuracy over 70% in figure 3) we obtained the following prediction results, using 4 distinct predictors working together in a dynamic adaptive manner (see figure 4). As it can be observed, the hybrid predictor is the best (85% average prediction accuracy), exploiting the synergism between the 3 particular predictors.

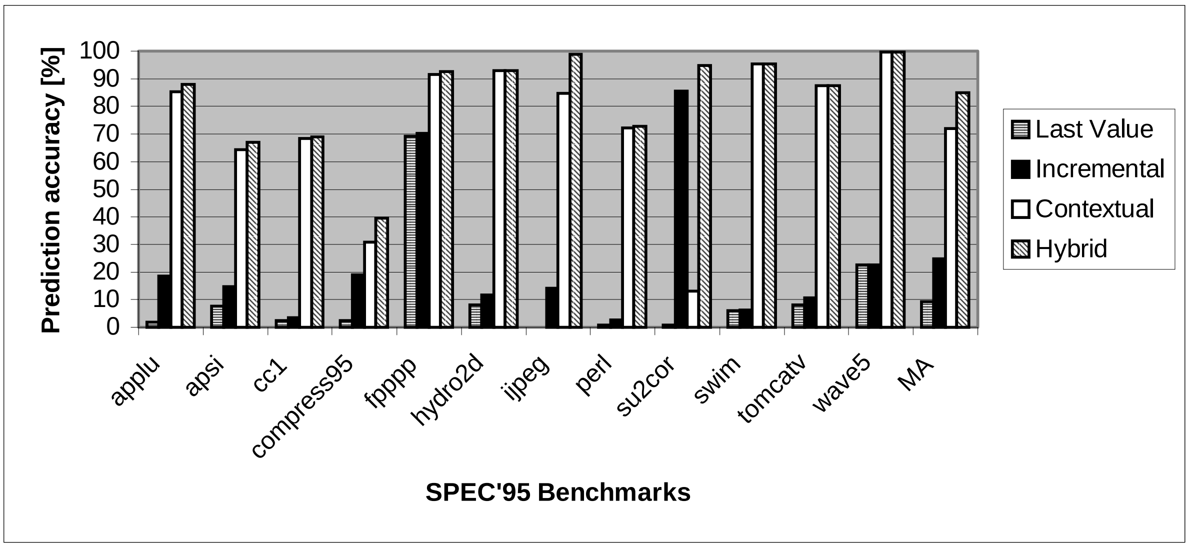

Figure 4. Register Value Prediction Accuracies using 4 distinct Predictors: Last Value, Incremental, Contextual and Hybrid (Contextual-Incremental).

Finally we present some results obtaine in our indirect jumps/calls target prediction research, a very difficult problem, too. Figures 5 and 6 present the obtained indirect jumps target prediction accuracies for a PPM predictor implemented with a complete associative Jump Value Prediction Table (JVPT) structure having 256 entries. Each entry in the JVPT stores 32 respectively 256 targets values of the corresponding indirect branch. The search pattern length is varied from 1 to 4 (figure 5) respectively from 1 to 12 (figure 6). Analyzing the average means obtained results we observed an optimal prediction accuracy of about 90.58% (for a search pattern length of 3 values) respectively of about 90.46% (for a search pattern length of 4 values) – when history length is 32, respectively 91.69% (for a search pattern length of 4 values and a history length of 256). These results are better than that reported by other researchers that used more simplified context predictors.

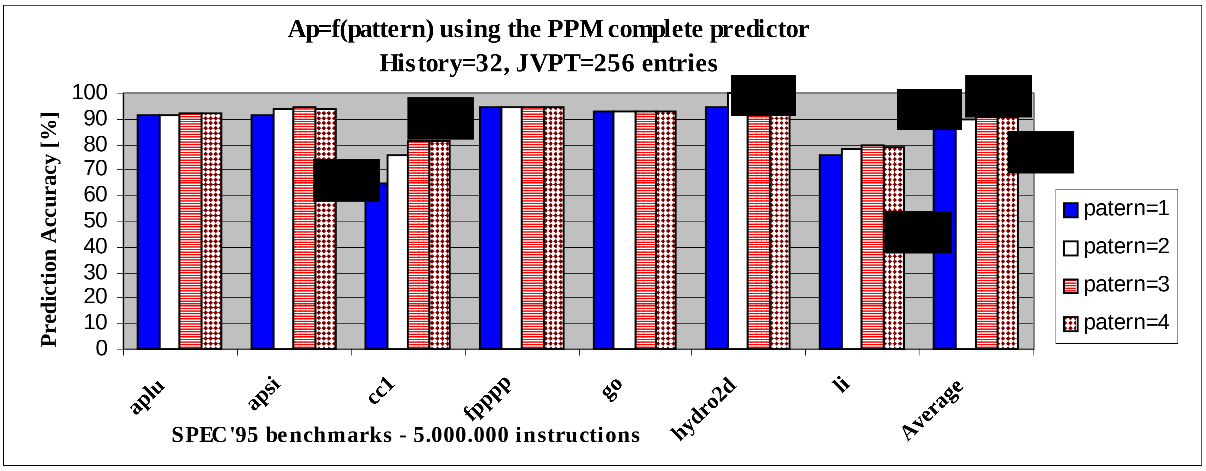

Figure 5. Indirect branch prediction accuracy using a complete PPM, varying the pattern size.

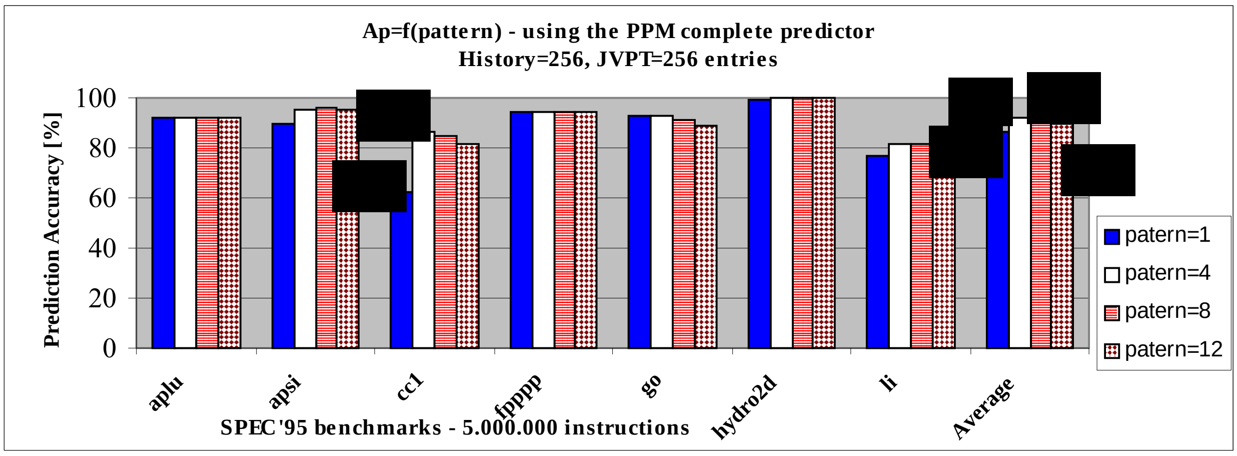

Figure 6. Indirect branch prediction accuracy using a complete PPM, varying the pattern size.

4. CONCLUSIONS AND FURTHER WORK

A lot of works in value prediction domain show clearly that there are significant amounts of value locality, both from memory-centric (message passing) and producer-centric (program structure) points of view. The simulation results show that all of the instructions within a single application have a relatively small overall result space during certain real-time periods. That is, while the potential range of values that could be produced is enormous, the actual range of values tends to be much more constrained. Increasing the history depth determines the improvement of value locality. The results are very useful because they express if and in which condition the value prediction is feasible (e.g. the history depth could be an useful indicator in developing the attached predictor). Through this paper we proposed a value prediction register-centric approach instead of a memory-centric or instruction-centric one, as all the other researchers proposed. An original evaluation is presented in figure 2, where it is evidenced the value locality concept associated with MIPS's general-purpose registers. The results obtained on some special registers ($at, $sp, $ra) of MIPS architecture are quite remarkable (≈90% value locality degree) leading to conclusion that value prediction might be successful applied at least on these registers. Also we demonstrated that an important challenge is to find some efficient confidence mechanisms. In this last sense, we are working now to a completely new hybrid value predictor, based on a neural network confidence approach. This approach continues our well-known work in neural branch prediction [Vin00, Ega03]. More precisely, a neural network (NN) structure is used for implementing a dynamic selection mechanism in order to select the best predictor at a certain moment. Each predictor will inform the NN about its last actions. Based on this information, the NN will select the proper predictor, optimal at a given time.

Also, encouraged by our already obtained results, we'll try to examine the value locality of all register writing instructions. We intend to exploit store value locality evaluating the potential reduction in multiprocessor data and address traffic. For improving the value prediction accuracy we'll try to develop some new hybrid prediction mechanisms.

ACKNOWLEDGEMENTS

This short - paper is based on a scientific presentation that I made up for the Information Science and Technology (IST) Section belonging to The Romanian Academy, in January 2004. Therefore I like to thank to IST Section members, especially to President Acad. Mihai Drăgănescu, for kindly inviting me to present a part of my research, for their useful suggestions and honorable appreciation.

REFERENCES

1. BURGER D., AUSTIN T., *The SimpleScalar Tool Set, Version 2.0*, University of Wisconsin Madison, USA, CSD TR #1342, June, 1997.
2. BUTTS J.A., SOHI G., *Characterizing and Predicting Value Degree of Use*, Proceedings of the 35th IEEE/ACM International Symposium on Microarchitecture, Istanbul, Turkey, 18-22 Nov. 2002.
3. CALDER B., REINMAN G., TULLSEN D., *Selective Value Prediction*, International Symposia of Computer Architecture (ISCA), Atlanta, SUA, May 1999.
4. CALDER B., FELLER P., EUSTACE A., *Value Profiling and Optimization*, Journal of Instruction-Level Parallelism 1, SUA, 1999.
5. CHANG, P., HAO, E. and PATT, Y. N., *Alternative Implementations of Hybrid Branch Predictors*, Micro-29, Ann Arbor, Michigan, November 1995, pp 252-257.
6. DESWET V., GOEMAN B., BOSSCHERE K., *Independent hashing as confidence mechanism for value predictors in microprocessors*, International Conf. EuroPar, Augsburg, Germany, 2002.
7. EGAN C., STEVEN G., VINȚAN L., *Cached Two-Level Adaptive Branch Predictors with Multiple Stages,* pg. 179-191, “Lecture Notes in Computer Science”, vol. 2299, Springer-Verlag, ISSN 0302-9743, ISBN 3-540-43409-7, Berlin Heidelberg, 2002.
8. EGAN C., VINTAN L. et al, *Two-Level Branch Prediction using Neural Networks*, Journal of Systems Architecture, vol. 49, issues 12-15, pg.557-570, ISSN: 1383-7621, Elsevier, December 2003.
9. FIELDS B., RUBIN S., BODIK R., *Focusing Processor Policies via Critical-Path Prediction*, The 28th Annual International Symp. On Comp. Archit., Göteborg, Sweden, July, 2001.
10. FLOREA A., VINȚAN L., SIMA D., *Understanding Value Prediction through Complex Simulations,* Proceedings of The 5th International Conf. on Technical Informatics (CONTI ’2002), University "Politehnica" of Timisoara, Romania, October 2002.
11. GABBAY F., MENDELSOHN A., *Using Value Prediction To Increase The Power Of Speculative Execution Hardware*, ACM Transactions on Computer Systems, vol 16, nr. 3, 1998.
12. HENNESSY, J. L. and PATTERSON, D. A., *Computer Architecture: A Quantitative Approach*, Morgan Kaufmann, 3rd Edition, 2002.
13. HUANG J., LILJA D., *Exploiting Basic Block Value Locality with Block Reuse*, University of Minnesota, Technical Report No. HPPC-98-09, 1998.
14. LEPAK K., LIPASTI M., *On The Value Locality of Store Instructions*, International Symposia on Computer Architecture, Vancouver, Canada, 2000.
15. LIPASTI M., WILKERSON C., SHEN P., *Value Locality and Load Value Prediction*, 17th ASPLOS International Conference VII, pg. 138-147, MA, SUA, October 1996.
16. LIPASTI M., SHEN J.P., *Exceeding The Dataflow Limit Via Value Prediction*, Proceedings of the 29th ACM/IEEE International Symposium on Microarchitecture, December, 1996.
17. MARCUELLO P., TUBELLA J., GONZALES A., *Value Prediction for Speculative Mutithreaded Architectures*, 1072-4451/IEEE, 1999.
18. MUDGE, T.N., CHEN, I., Coffey, J., *Limits of Branch Prediction,* Technical Report, Electrical Engineering and Computer Science Department, The University of Michigan, Ann Arbor, Michigan, USA, January 1996.
19. PETZOLD J., BAGCI F., TRUMLER W., and UNGERER T., *Context Prediction Based on Branch Prediction Methods*, Technical Report, University of Augsburg, 2003.
20. RABINER L.R., *A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition*, Proceedings of the IEEE, vol. 77, no. 2, February 1989.

21. SAZEIDES Y., SMITH J.E., *The Predictability of Data Values*, Proceedings of The 30th Annual International Symposium on Microarchitecture, December, 1997.
22. SAZEIDES Y., SMITH J.E., *Modeling Program Predictability*, 1063-6897/IEEE, 25th ISCA, 1998.
23. SAZEIDES Y.T., *An Analysis of Value Predictibality and its Application to a Superscalar Processor,* PhD Thesis, University of Wisconsin-Madison, SUA, 1999.
24. SILC J., ROBIC B., UNGERER T., *Processor Architecture, from Dataflow to Superscalar and beyond,* Springer-Verlag, 1999.
25. SODANI A., SOHI G., *Dynamic Instruction Reuse*, International Symposia on Computer Architecture (ISCA), Denver, SUA, 1997.
26. SODANI A., SOHI G., *Understanding the Differences Between Value Prediction And Instruction Reuse*, 0-8186-8609-X/IEEE, 1998.
27. *The SPEC benchmark programs*, http://www.spec.org.
28. STEVEN G., EGAN C., ANGUERA R., VINȚAN L., *Dynamic Branch Prediction using Neural Networks*, Proceedings of International Euromicro Conference DSD '2001, ISBN 0-7695-1239-9, Warsaw, Poland, September, 2001 (pg.178-185).
29. VINȚAN L., *Towards a Powerful Dynamic Branch Predictor*, Romanian Journal of Information Science and Technology (ROMJIST), vol.3, nr.3, pg.287-301, ISSN: 1453-8245, Romanian Academy, Bucharest, 2000.
30. VINȚAN L., *Prediction and Speculation in Advanced Microprocessors*, MatrixRom Publishing House, Bucharest, 2002 (in Romanian).
31. WANG K, FRANKLIN M., *Highly Accurate Data Value Precision Using Hybrid Predictors*, Proceedings of The 30th International Symposium on Microarchitecture, 1997.
32. WU Y., CHEN D.Y., FANG J., *Better Exploration of Region-Level Value Locality with Integrated Computation Reuse and Value Prediction*, The 28th Annual International Symp. On Comp. Archit., Göteborg, Sweden, July, 2001.
33. YEH, T. and PATT Y., *Alternative Implementations of Two-Level Adaptive Branch Prediction*, ISCA -19, Gold Coast, Australia, May 1992, pp124-134.
34. ZHOU H., FLANAGAN J., CONTE T., *Detecting Global Stride Locality in Value Streams*, The 30th Annual International Symp. on Comp. Archit., San Diego, California, June, 2003.

Received March 1, 2004