Value Prediction Focalized on CPU Registers

Lucian Vintan, Arpad Gellert, Adrian Florea

University "Lucian Blaga" of Sibiu, Computer Science Department,
E. Cioran 4, 550025 Sibiu, Romania

ABSTRACT

Value Prediction is a relatively new technique that increases performance by eliminating true data dependencies. Value prediction architectures allow data dependent instructions to issue and execute speculatively using the predicted value. This technique is built on the concept of value locality, which describes the likelihood of the recurrence of a previously seen value within a storage location. This paper extends dynamic value prediction by introducing the concept of register centric prediction instead of instruction centric prediction.

KEYWORDS: Dynamic value prediction; speculative execution; simulation

1 Introduction

The main aim of this paper consists in focalizing dynamic value prediction to CPU context by attaching value predictors to CPU registers. The value localities obtained on some registers of MIPS architecture were quite remarkable leading to conclusion that value prediction might be successfully applied, at least on these favorable registers. The register value prediction technique consists in predicting the next values of registers based on the previously seen values. It executes the subsequent data dependent instructions using the predicted values and the speculative execution will be validated when the correct values are known. If the value was correctly predicted the critical path is reduced, otherwise the instructions executed with wrong entries must be executed again.

2 Register Value Predictors

The main benefit of the proposed value prediction technique consists in unlocking the subsequent dependent instructions. In register-centric prediction the tables are indexed only in the second part of the decode stage (figure 1).

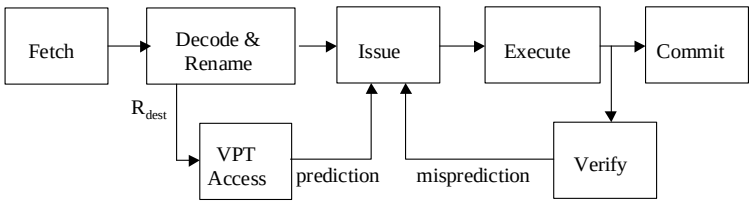

Figure 1: Pipeline with register value prediction

"Last Value" Predictors. The *last value* predictors (figure 2) predict the new value as the same with the last value stored in the corresponding register [Lip96]. Exploiting the correlation between register names and the values stored in those registers, will decrease instruction latencies. Each entry in the prediction table has its own confidence automaton, which is incremented when the prediction is correct and it is decremented otherwise. Obviously, the utility of Value Prediction techniques is emphasized only in the case of a correct prediction otherwise it determines structural hazards and a higher instruction execution latency. Based on the dynamic behavior of the register content there is developed the following classification: unpredictable and predictable registers. By treating separately each group of registers it can be avoided the costs of mispredictions. It is necessary the verification of the value generated, and the automata must be updated.

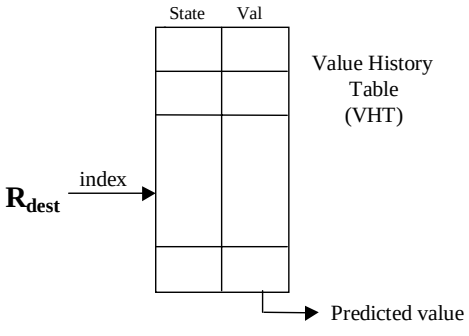

Figure 2: The "last value" predictor

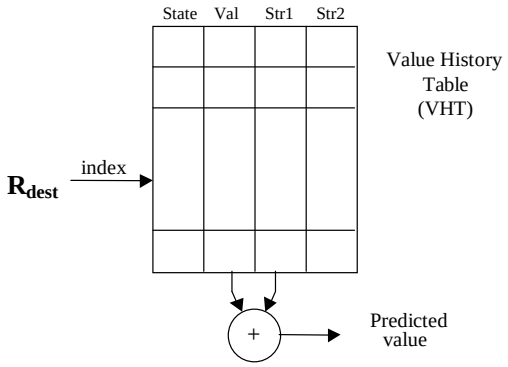

Figure 3: The stride predictor

Stride Predictors. In the case of stride predictors (figure 3), considering that v_{n-1} and v_{n-2} are the most recently values, the new value v_n will be calculated using the recurrence formula: $v_n = v_{n-1} + (v_{n-1} - v_{n-2})$, where $(v_{n-1} - v_{n-2})$ is the stride of the sequence. In the *Val* field of the prediction table, for each register is stored the last value. In the *Str1* and *Str2* fields there are stored the last two strides computed for each register. If the register is used again by an instruction as destination and $Str_1 = Str_2$, it is calculated the predicted value adding the stride to the value stored in the VHT (Val). If the automaton is in the predictable state, the prediction is generated. The confidence automaton is updated.

Context-Based Predictors. The context-based predictors predict the value that will be stored in a register based on the last values stored in that register. A context is a finite sequence of values with repeated apparition like in a Markov chain. The predictors that implement the *Prediction by Partial Matching* algorithm (PPM) represent an important class of context-based predictors. This predictor contains a set of simple Markov predictors (figure 4). It is predicted the value that followed the context with the highest frequency. The predicted value depends on the context, therefore, a longer context frequently drives to higher prediction accuracy, but sometimes it can behave as noise. In the considered sample only the 3rd order Markov predictor makes a "correct" prediction. A complete PPM predictor contains N simple *Markov* predictors, from the 0th order to the (N-1)th order. If the (N-1)th *Markov* predictor produces a prediction the process is finished, otherwise the (N-2)th order *Markov* predictor will be activated, and so on. In figure 5 is presented the structure of the context-based predictor. Each entry from the VHT has associated an automaton that is updated after each prediction. The fields V_1, …, V_4 stored the last four

values associated with each register (for a history of four values). If the automaton is in the predictable state, it predicts the value that follows the context with the highest frequency.

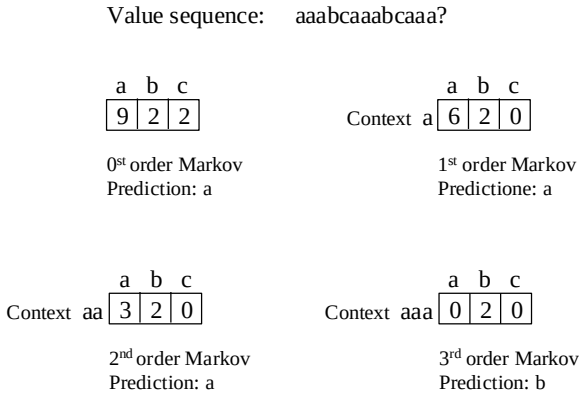

Figure 4: A PPM predictor

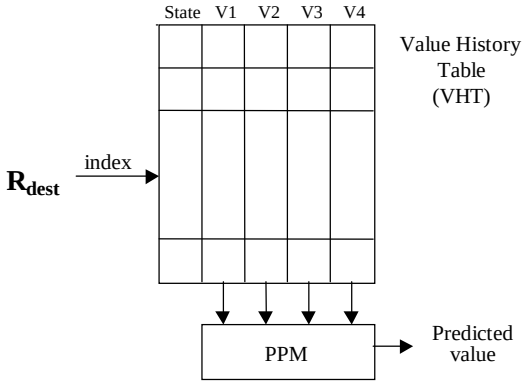

Figure 5: The context-based predictor

Hybrid Predictors. Researches show that a single type of predictor does not offer the best results. Some types of value sequences generated in programs are better predicted with a certain predictor, and others, with another type of a particular predictor [Wan97]. Therefore, it naturally appears the idea of hybrid prediction; two or more value predictors dynamically working together in the prediction process. In figure 6 it is presented a hybrid predictor composed by a context-based predictor and respectively a stride predictor. The context-based predictor has always priority; in this way the value generated by the stride predictor is used only if the context-based predictor cannot generate a prediction. This fixed prioritization seems to be not optimal; a dynamic prioritization based on confidences should be better.

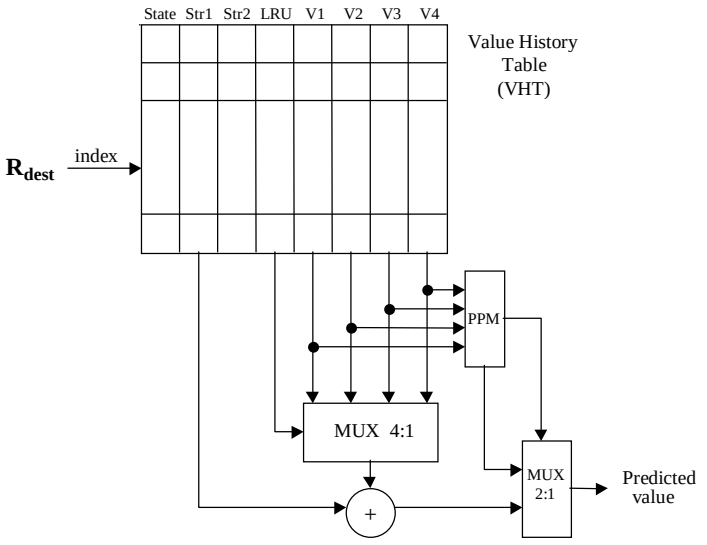

Figure 6: The hybrid predictor (context-based & stride)

3 Experimental Results

In our investigations, we are focusing only on the predictable registers, having prediction accuracy higher than a certain threshold, measured using the hybrid predictor on the SPEC benchmarks. The registers with prediction accuracy higher than 80% are: 1, 10÷12,

18, 29÷31 on SPEC'95, and respectively 1, 8, 11÷15, 20÷25, 29÷31 on SPEC2000. The global using rate of these registers is 10.58% on SPEC'95 benchmarks and 9.01% on SPEC2000.

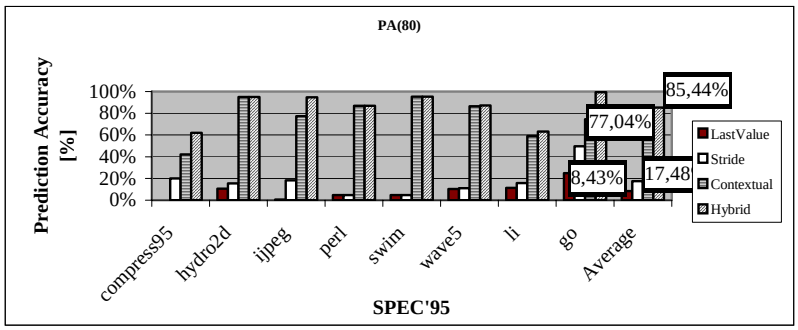

Figure 7: Prediction accuracy using 8 favorable registers (SPEC'95)

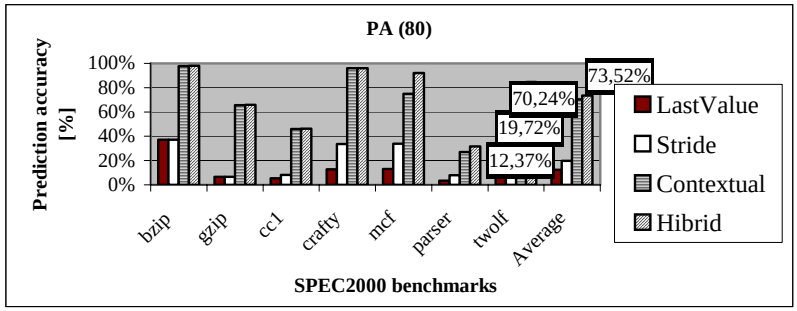

Figure 8: Prediction accuracy using 16 favorable registers (SPEC 2000)

Figures 7 and 8 emphasize for each benchmark the global prediction accuracy obtained with the implemented predictors using 8 respectively 16 selected registers. In these figures, each bar represents the register value prediction for a certain benchmark, measured by counting the number of times when prediction is accurate for any of the favorable registers and dividing to the total number when these registers are written.

4 Conclusions

Our results show that there is a time-correlation between the names of the destination registers and the values stored in these registers. The simulations exhibit that the hybrid predictor optimally exploits this correlation with an average prediction accuracy of 85.44%. Considering an 8-issue out-of-order superscalar processor we showed that register centric value prediction produce average speedups of 17.30% for the SPECint95 benchmarks, respectively of 13.58% for the SPECint2000 benchmarks.

References

[Lip96] Lipasti, M. H., Wilkerson, C. B., Shen, J. P. *Value Locality and Load Value Prediction*, The 7th International Conference on Architectural Support for Programming Languages and Operating Systems, 1996.

[Wan97] Wang K., Franklin M. *Highly Accurate Data Value Prediction using Hybrid Predictors*, The 30th Annual ACM/IEEE International Symposium on Microarchitecture, 1997.