University of Rhode Island
Dept. of Electrical and Computer Engineering
Kelley Hall
4 East Alumni Ave.
Kingston, RI 02881, USA

Technical Report No. 0496-0001

Data References for: "Reducing the Ill Effects of Branches"

Augustus K. Uht, University of Rhode Island
Vijay Sindagi, Texas Instruments
Sajee Somanathan, ADE Corporation

Email: uht@ele.uri.edu

April 3, 1996

1 Introduction

This document gives the sources for the data in the paper: "Reducing the Ill Effects of Branches",[1] in particular the data for Table 2, in which the BERTs' characteristics are given and compared. The references are not specified in the original paper for two reasons: there is a hard limit of 12 references for the paper (COMPUTER magazine specification); and including them would have made the table too busy (a style consideration).

In the version of the table herein, each datum has a numeric citation (superscript) indicating the published source of the number. The original footnotes for the table are not modified.

Many of these data do not appear directly in the cited works, but are gleaned from these works, normally by only looking at the data obtained from simulations of the SPECint92 benchmark suite, or approximations thereof.

There is other data available in some cases for blank table entries, but they were not included in the table as the benchmarks used could not reasonably be compared to the SPECint92's, or the experimental setup was in other ways found not to be relevant.

2 Data References for the Table

The table with citations follows on the next page.

2.1 Notes on Certain ILP Data

The following list contains notes (caveats) on some of the data in the table. The item heading format is: {"BERT name", "ILP Speedup":}.

semi-static, 1.5: This assumed a branch penalty of 8 cycles. A more conservative, and perhaps more common penalty of 14 cycles would change the ILP speedup to: 1.15.

dynamic (2-bit), 2.9: With the same consideration as with the semi-static prediction datum, the ILP speedup would go to: 1.9.

dynamic (adaptive), 3.6: With the same consideration as with the semi-static prediction datum, the ILP speedup would go to: 2.65.

VLIW-based, 3.6: This covers several VLIW BERT's, from the Multiflow Trace (ILP Speedup of about 1.55),[2] to Moon and Ebcioğlu's model (ILP Speedup of 3.6).[3]

2.2 Other Notes

Ref.: Recompilation w.r.t. Multiscalar and VLIW-based BERT's

Recent studies indicate that it may be possible to avoid recompilation for these BERT's. However, it is our view that the methods proposed are highly-tailored for the current software/hardware interfaces, and the interfaces might very well change from original machine to upgraded machine. For example, if the compiler desires to transmit new types of information to the hardware, for a modified execution algorithm to improve performance, the interface will change and hence recompilation will be necessary. Therefore we have indicated in the table that recompilation is necessary for the above BERT's.

	Characteristics[a]									
	Performance			Hardware			Software			
BERT	ILP speedup realized[b,e]	BPA (research)[b] (%)	BPA (real)[b] (%)	real machine hardware[b,d]	instruction set modified?	hardware cost[c]	code size increase?[c]	profiling necessary?	recompilation necessary?[f]	software complexity[c]
Single-Path with the specified branch predictor, given in the order appearing in the text:										
none	1.6[5,6]	-	-	none	no	0	no	no	no	0
always Not Taken	NA	NA	40[4]	little	no	1	no	no	no	0
always Taken	NA	61-75[7]	60[4]	little	no	1	no	no	no	0
BTFN	NA	NA	65[4]	little	no	2	no	no	no	0
semi-static (profiling)	1.5[8]	83.4[9]	75[4]	1 OP-Code bit	yes	3	no	yes	no	2
dynamic (1-bit)	NA	90[7]	77-79[4]	1K-2K entries[4]	no	4	no	no	no	0
dynamic (2-bit)	2.9[8]	90[10]-92.2[7]	78-89[4]	256-4K entries[4]	no	5	no	no	no	0
dynamic (adaptive)	3.6[8]	90-96[6,11]	90-94[11]	≥2K bits[4]	no	6	no	no	no	0
hybrid	NA	88[9]	-	1 OP-Code bit	yes	4	yes; 2	yes	no	3
Other BERT's, using various branch predictors, in increasing ILP speedup order:										
predication (alone)	1.6[12]	-	-	sev. OP-Code bits	yes	7	no	no	no	0
MCD (hard.-based)[g]	2.1[13]	-	-	-	no	7	yes; 1	no	no	1
multiscalar	2.6[14]	-	-	sev. OP-Code bits	yes	8	yes; 1	no	yes	3
predication (w/soft.)	3.0[15,16]	-	-	sev. OP-Code bits	yes	7	yes; 1	yes	no	4
VLIW-based	3.6[3; also: 2,17,18]	-	-	none	yes	0	yes; 3	yes	yes	4
Eager Execution (EE)	7.6[10]	-	-	1 branch bypassed[4]	no	9	no	no	no	0
Disjoint EE (DEE)[h]	8.5[10]	-	-	-	no	7	no	no	no	0
DEE-MCD[g,h]	31.9[10]	-	-	-	no	8	yes; 1	no	no	1

[a] Space and style constraints prohibit citing all of the sources of the BERT data in this Table. However, these references may be found in Technical Report No. 0496-0001 of the Dept. of Electrical and Computer Engineering, the University of Rhode Island; this is available on the World Wide Web at URL: `http://www.ele.uri.edu/faculty/uht/papers.html`. "NA" is: Not Available. "-" is: not relevant, or has not been used in a real machine.

[b] These data are from various sources, most included in the references. Many are from Microprocessor Report newsletters.[4] The BPA (research) figures are from studies looking at predictors by themselves, whereas the BPA (real) numbers include hardware and code limitations; most are based on real computers. When possible, the BPA and ILP data assume the SPECint92 benchmark suite or a portion thereof.

[c] The numbers are **order** rankings, worse numbers are higher; they do not indicate anything in absolute terms, nor can numbers from different columns be compared by their values in "better" or "worse" terms. The numbers are the authors' subjective opinions only.

[d] OP-Code bit(s): The BERT requires one or more ("sev.") additional bits in either branch or assignment instructions. An "entry" is the state (memory bits) for one automata.

[e] The numbers are speedup factors (i.e., in 100's of percent). These data are from research machine simulations. Each datum was computed as the harmonic mean of the individual benchmarks' speedups. Only the Single-Path "none" data assume unrestricted resources. Note that a BERT's actual operation and its simulated operation differ in varying degrees from BERT to BERT, and from study to study.

[f] ...upon machine upgrade or downgrade.

[g] These BERT's work best with some binary-to-binary code translation, via a *filter* program run once for an application program. The main action performed is to do limited loop unrolling. The filter is not required for correct code execution.

[h] The data given assume hardware-based implementations; 2-bit dynamic branch prediction is used. Software-based implementations are also possible.

Table 2: *Comparison of the BERT's.* The BERT's are listed in the leftmost column, with their absolute and relative characteristics given in the columns to the right. The characteristics are grouped into categories of performance realized, hardware attributes and software attributes.

References

[1] A. K. Uht, V. Sindagi, and S. Somanathan, "Reducing the Ill Effects of Branches," 1996. Submitted to IEEE COMPUTER Magazine, 1995; accepted, under revision.

[2] M. A. Schuette and J. P. Shen, "Instruction-Level Experimental Evaluation of the Multiflow TRACE 14/300 VLIW Computer," *The Journal of Supercomputing*, vol. 7, no. 1/2, pp. 249–271, 1993.

[3] S.-M. Moon and K. Ebcioğlu, "An Efficient Resource-Constrained Global Scheduling Technique for Superscalar and VLIW Processors," in *Proceedings of the Twenty-Fifth International Symposium on Microarchitecture (MICRO-25)*, pp. 55–70, ACM-IEEE, December 1992.

[4] L. Gwennap, "New Algorithm Improves Branch Prediction," *Microprocessor Report*, vol. 9, pp. 17–21, March 27, 1995.

[5] E. M. Riseman and C. C. Foster, "The Inhibition of Potential Parallelism by Conditional Jumps," *IEEE Transactions on Computers*, vol. C-21, pp. 1405–1411, December 1972.

[6] D. Wall, "Limits of Instruction-Level Parallelism," Tech. Rep. 93/6, Digital Western Research Laboratory, Palo Alto, California, November 1993. WRL Research Report; to obtain, send: `Subject: help` to: `WRL-Techreports@pa.dec.com`.

[7] J. E. Smith, "A Study of Branch Prediction Strategies," in *Proceedings of the 8th Annual Symposium on Computer Architecture*, pp. 135–148, IEEE and ACM, 1981.

[8] T.-Y. Yeh and Y. N. Patt, "A Comprehensive Instruction Fetch Mechanism for a Processor Supporting Speculative Execution," in *Proceedings of the Twenty-Fifth International Symposium on Microarchitecture (MICRO-25)*, pp. 129–139, ACM-IEEE, December 1992.

[9] C. Young and M. D. Smith, "Improving the Accuracy of Static Branch Prediction Using Branch Correlation," in *ASPLOS VI: Proceedings of the Sixth Annual International Conference on Architectural Support for Programming Languages and Operating Systems*, pp. 232–241, ACM, October 1994.

[10] A. K. Uht and V. Sindagi, "Disjoint Eager Execution: An Optimal Form of Speculative Execution," in *Proceedings of the 28th International Symposium on Microarchitecture*, pp. 313–325, IEEE and ACM, November/December 1995.

[11] T.-Y. Yeh and Y. N. Patt, "A Comparison of Dynamic Branch Predictors that use Two Levels of Branch History," in *Proceedings of the 20th Annual International Symposium on Computer Architecture*, pp. 257–266, IEEE and ACM, May 1993.

[12] M. Srinivas, A. Nicolau, and V. H. Allan, "An Approach to Combine Predicated/Speculative Execution for Programs with Unpredictable Branches," in *Proceedings of the International Conference on Parallel Architectures and Compilation Techniques*, IFIP, North-Holland, May 1994.

[13] A. K. Uht, "Concurrency Extraction via Hardware Methods Executing the Static Instruction Stream," *IEEE Transactions on Computers*, vol. 41, pp. 826–841, July 1992.

[14] G. S. Sohi, April 2, 1996. Personal communication.

[15] W. W. Hwu and D. I. August, "Predicated Execution Architectures: New Research Results and Directions," October 1995. Talk given at the First Intel Microprocessor Research Forum, Santa Clara, CA, USA.

[16] S. A. Mahlke, R. E. Hank, J. E. McCormick, D. I. August, and W. W. Hwu, "A Comparison of Full and Partial Predicated Execution Support for ILP Processors," in *Proceedings of the 22nd Annual International Symposium on Computer Architecture*, pp. 138–149, IEEE and ACM, May 1995.

[17] S. Melvin and Y. Patt, "Exploiting Fine-Grained Parallelism Through a Combination of Hardware and Software Techniques," in *Proceedings of the 18th Annual International Symposium on Computer Architecture, Toronto, Canada*, pp. 287–296, IEEE and ACM, May 1991.

[18] P. P. Chang, N. J. Warter, S. A. Mahlke, W. Y. Chen, and W. W. Hwu, "Three Architectural Models for Compiler-Controlled Speculative Execution," *IEEE Transactions on Computers*, vol. 44, pp. 481–494, April 1995.