University of Rhode Island
Dept. of Electrical and Computer Engineering
Kelley Hall
4 East Alumni Ave.
Kingston, RI 02881-0805, USA

Technical Report No. 0497-0002

Data References for: "Branch Effect Reduction Techniques"

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

Email: uht@ele.uri.edu
WWW: http://www.ele.uri.edu/~uht

April 25, 1997

1 Introduction

This document gives the sources for the data in the paper: "Branch Effect Reduction Techniques".[1] That paper was originally entitled: "Reducing the Ill Effects of Branches".[2] Additional information on BERTs and related topics is also given.

The main data considered is the data for Table 3, in which the BERTs' characteristics are given and compared. The complete set of references are not specified in the original paper for two reasons: the number of references allowable for the paper is limited (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.

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. Harmonic means were always used to obtain summary numbers.

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. As an example, there is a published estimate of a branch prediction accuracy (BPA) of 95% for the selector predictor in the DEC Alpha 21264,[3] but it is for all of the SPEC95's, not just the SPECint's, and it is an estimate, not measured, so it is not included in Table 3.

2 Data References for the Table

The table with citations follows on page 2.

	Characteristics[a]									
	Performance			Hardware			Software			
BERT	ILP speedup realized	BPA (research) (%)	BPA (real) (%)	real machine hardware	instruction set modified?	hardware cost	code size increase?	profiling necessary?	recompilation necessary?	software complexity
Single-Path with the specified branch predictor, given in the order appearing in the text:										
none	1.6[4,5]	-	-	none	no	0	no	no	no	0
always Not Taken	NA	NA	40[6]	little	no	1	no	no	no	0
always Taken	NA	61-75[7]	60[6]	little	no	1	no	no	no	0
BTFN	NA	NA	65[6]	little	no	2	no	no	no	0
semi-static (profiling)	1.5[8]	83.4[9]	75[6]	1 OP-Code bit	yes	3	no	yes	no	2
hybrid	NA	88[9]	-	1 OP-Code bit	yes	4	yes; 2	yes	no	3
dynamic (1-bit)	NA	90[7]	77-79[6]	1K-2K entries[6]	no	4	no	no	no	0
dynamic (2-bit)	2.9[8]	90[10]-92.2[7]	78-89[6]	256-4K entries[6]	no	5	no	no	no	0
dynamic (adaptive)	3.6[8]	90-96[5,11]	90-94[11]	≥2K bits[6]	no	6	no	no	no	0
selector	NA	93-97[12,13]	NA	≥28K bits[3]	no	7	no	no	no	0
Other BERT's, using various branch predictors, in increasing ILP speedup order:										
predication (alone)	1.6[14]	-	-	sev. OP-Code bits	yes	7	no	no	no	0
MCD (hard.-based)	2.1[15]	-	-	-	no	7	yes; 1	no	no	1
multiscalar	2.6[16]	-	-	sev. OP-Code bits	yes	8	yes; 1	no	yes	3
predication (w/soft.)	3.0[17,18]	-	-	sev. OP-Code bits	yes	7	yes; 1	yes	no	4
VLIW-based	3.6[19; also: 20–22]	-	-	none	yes	0	yes; 3	yes	yes	4
Eager Execution (EE)	7.6[10]	-	-	1 branch bypassed[6]	no	9	no	no	no	0
Disjoint EE (DEE)	8.5[10]	-	-	-	no	7	no	no	no	0
DEE-MCD	31.9[10]	-	-	-	no	8	yes; 1	no	no	1

[a] "NA" is: Not Available. "-" is: not relevant, or has not been used in a real machine.

Table 3: *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.

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, in the future, 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.

Multiscalar, 2.6: Recent data suggests that the realized speedup may be less.[23]

VLIW-based, 3.6: This covers several VLIW BERTs, from the Multiflow Trace (ILP Speedup of about 1.55),[21] to Moon and Ebcioğlu's model (ILP Speedup of 3.6).[19]

3 Sources and References for Other Mentioned Topics

3.1 Data Effect Reduction Techniques (DERTs)

In the "How Dependencies Limit Instruction-Level Parallelism" sidebar, there is a reference to recent research on data prediction and speculation. The base *data locality* was first noticed by Citron and Rudolph.[24] Lipasti et al published the first measurements of data locality.[25,26] They also described microarchitectural features to take advantage of the locality, but the performance gains were small (about 3.5% for SPECint(92?) benchmarks,[26] without espresso).

Smith et al made measurements indicating some of the potential of such approaches.[27] With perfect data load speculation, ILP speedups of 25 times are possible, although other measurements of their's suggest somewhat lower speedups are likely with realistic assumptions. It may be possible that this will be the DERT version of the original Riseman and Foster[4] BERT study.

3.2 Two-level Adaptive Predictor with Profiling

The two can be combined to improve performance over that of an adaptive predictor alone.[28]

3.3 Multiple Branch Prediction

Our work is available from the cited source[10] in the paper. Emma's work appears in a patent.[29] Recently a group at UC Irvine has also made progress in this area.[30]

A related subject is the speculative updating of branch predictor state, and the accompanying speculative branch prediction.[31] This is necessary since at a given time dynamic instances of a static branch may not have resolved, therefore the predictor's state is not fully updated, and thus a new prediction of the given static branch must necessarily be speculative, i.e., based on conditional (not resolved) branch signs.

An interesting question is whether a confidence predictor (next section) could somehow be used to advantage with speculative prediction.

3.4 Confidence Prediction

A branch confidence predictor gives an indication of how likely a prediction of a branch was to be correct. This may be useful in variants of Disjoint Eager Execution (DEE), as an aid to or substitute for the static tree heuristic, indicating when to eagerly execute a branch. However, confidence predictors currently do not indicate the desired order of assignment, so the disjoint assignment of resources must somehow be incorporated to get the benefits of DEE.

To our knowledge, the first confidence predictor was proposed in one of the first branch prediction papers,[7] but was effectively forgotten (although the paper most certainly was not) until Smith et al returned to the topic in 1996.[32] In the recent study, both 2-bit saturating counters and 2-level adaptive predictors were tested for their ability to measure branch prediction confidences.

3.5 Multiscalar Machines

Task prediction has been recently addressed.[23] Such methods could perhaps also be adapted for multiple branch prediction in other processors.

3.6 VLIW Variants

These include software pipelining[33] and boosting.[34]

3.7 Other Data

Other data present in the paper was obtained from the manufacturers' WWWeb sites in March and April of 1997. This includes the type of branch predictors used in the AMD-K5, AMD-K6, Cyrix 6x86, Cyrix M2 and DEC Alpha 21264, and their branch prediction accuracies.

4 Related Work and Issues, Not Mentioned in the Paper

4.1 Ref.: Recompilation w.r.t. Multiscalar and VLIW-based BERTs

Recent studies[35,36] indicate that it may be possible to avoid recompilation for these BERTs, on machine upgrade or downgrade. 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 BERTs.

Another study by Ebcioglu and Altman[37] proposes very fast dynamic recompilation to allow a special VLIW architecture to emulate arbitrary (especially existing) instruction sets. With this method, large code explosion can occur, affecting main memory requirements (a much larger memory system may be needed to avoid thrashing), but not the static object code size (disk space). Although the relative runtime compilation overhead may be relatively low in most of the studied benchmarks, it is prohibitively high for `cc1`, often considered to be the most realistic of the SPEC benchmarks. The ILP realized by the DAISY architecture is consistent with that of other VLIW methods (around 3-4), although if a finite cache is used, the ILP for `cc1` drops to less than 1. Only two SPECint95 benchmarks were simulated, so it is hard to determine summary numbers for the machine.

4.2 Other Branch Predictors

New types and variants of branch predictors continue to proliferate (see the relevant sections of recent proceedings of the conferences given in Section 5). The subject has also been approached from a theoretical point of view[38] using Markov processes. Even genetic algorithms have been applied to the development of new predictors.[39]

4.3 Other Predictor Considerations

It has recently been demonstrated[12,40] that context switching can have a deleterious effect on BPA. Other aspects of predictors have also been studied.[41,42]

4.4 Other Predictor Effects

Accurate branch predictors are essential for superpipelined processors. In the latter, it is not uncommon to have pipelines with more than 10 stages.[43] With such deep pipelines, the negative effects of mispredictions are great, giving branch penalties (idle time) of several cycles (e.g., 7-10 cycle penalty on the Pentium Pro[6,44]). Since in ILP machines each cycle corresponds to the execution of many instructions, performance is severely reduced by even infrequent mispredictions. Thus, deeply pipelined machines, such as the Pentium Pro and the DEC Alpha, benefit greatly from highly-accurate branch predictors.

5 Other Sources of BERT and ILP Information

In addition to the cited tutorial[45] and the Microarchitecture Symposium (MICRO), papers on these topics commonly appear in the Computer Architecture Symposium (ISCA), the Architectural Support for Programming Languages and Operating Systems conference (ASPLOS), and the High Performance Computer Architecture conference (HPCA). The proceedings for these gatherings are published by either the ACM or the IEEE Computer Society; it varies from year to year.

References

[1] A. K. Uht, V. Sindagi, and S. Somanathan, "Branch Effect Reduction Techniques," *COMPUTER*, May 1997.

[2] A. K. Uht, V. Sindagi, and S. Somanathan, "Reducing the Ill Effects of Branches," *COMPUTER*, 1996. Accepted, to appear. Available via URL: `http://www.ele.uri.edu/~uht`.

[3] L. Gwennap, "Digital 21264 Sets New Standard," *Microprocessor Report*, vol. 10, pp. 11–16, October 28, 1996.

[4] 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.

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

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

[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 and IEEE, December 1992.

[9] C. Young and M. D. Smith, "Improving the Accuracy of Static Branch Prediction Using Branch Correlation," in *Proceedings of the Sixth Annual International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS-VI)*, 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 (MICRO-28)*, 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. Evers, P.-Y. Chang, and Y. Patt, "Using Hybrid Branch Predictors to Improve Branch Prediction Accuracy in the Presence of Context Switches," in *Proceedings of the 23nd Annual International Symposium on Computer Architecture*, pp. 3–11, IEEE and ACM, May 1996.

[13] S. McFarling, "Combining Branch Predictors," WRL Technical Note TN-36, Digital Western Research Laboratory, Palo Alto, California, June 1993.

[14] 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 (PACT94)*, IFIP, North-Holland, May 1994.

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

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

[17] 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.

[18] 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.

[19] 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 and IEEE, December 1992.

[20] 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.

[21] 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.

[22] 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.

[23] Q. Jacobsen, S. Bennett, N. Sharma, and J. Smith, "Control Flow Speculation in Multiscalar Processors," in *Proceedings of the Third High-Performance Computer Architecture Conference (HPCA-3)*, pp. 218–229, IEEE, February 1997.

[24] Citron, D. and Rudolph, L., "Creating a Wider Bus Using Caching Techniques," in *Proceedings of the First High-Performance Computer Architecture Conference*, pp. 90–99, IEEE, January 1995.

[25] M. H. Lipasti, C. B. Wilkerson, and J. P. Shen, "Value Locality and Load Value Prediction," in *Proceedings of the Seventh Annual International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS-VII)*, pp. 138–147, ACM, October 1996.

[26] M. Lipasti and J. Shen, "Exceeding the Data Flow Limit via Value Prediction," in *Proceedings of the 29th International Symposium on Microarchitecture (MICRO-29)*, pp. 226–237, IEEE and ACM, December 1996.

[27] Y. Sazeides, S. Vassiliadis, and J. E. Smith, "The Performance Potential of Data Dependence Speculation & Collapsing," in *Proceedings of the 29th International Symposium on Microarchitecture (MICRO-29)*, pp. 238–247, IEEE and ACM, December 1996.

[28] P.-Y. Chang, E. Hao, T.-Y. Yeh, and Y. Patt, "Branch Classification: a New Mechanism for Improving Branch Predictor Performance," in *Proceedings of the 27th International Symposium on Microarchitecture (MICRO-27)*, pp. 22–31, IEEE and ACM, November/December 1994.

[29] P. G. Emma, J. W. Knight, J. H. Pomerene, and T. R. Puzak, "U.S. Patent No. 5,434,985: Simultaneous Prediction of Multiple Branches for Superscalar Processing," July 1995.

[30] Wallace, S. and Bagherzadeh, N., "Multiple Branch and Block Prediction," in *Proceedings of the Third High-Performance Computer Architecture Conference (HPCA-3)*, pp. 94–103, IEEE, February 1997.

[31] E. Hao, P.-Y. Chang, and Y. Patt, "The Effect of Speculatively Updating Branch History on Branch Prediction Accuracy, Revisited," in *Proceedings of the 27th International Symposium on Microarchitecture (MICRO-27)*, pp. 228–232, IEEE and ACM, November/December 1994.

[32] E. Jacobsen, E. Rotenberg, and J. E. Smith, "Assigning Confidence to Conditional Branch Prediction," in *Proceedings of the 29th International Symposium on Microarchitecture (MICRO-29)*, pp. 142–152, IEEE and ACM, December 1996.

[33] K. Ebcioğlu, "A Compilation Technique for Software Pipelining of Loops with Conditional Jumps," in *Proceedings of the Twentieth Annual Workshop on Microprogramming (MICRO-20)*, pp. 69–79, Association of Computing Machinery, December 1987.

[34] M. D. Smith, M. S. Lam, and M. A. Horowitz, "Boosting Beyond Static Scheduling in a Superscalar Processor," in *Proceedings of the 17th Annual International Symposium on Computer Architecture*, pp. 344–354, IEEE and ACM, May 1990.

[35] T. M. Conte and S. W. Sathaye, "Dynamic Rescheduling: A Technique for Object Code Compatibility in VLIW Architectures," in *Proceedings of the 28th International Symposium on Microarchitecture (MICRO-28)*, pp. 208–218, IEEE and ACM, November/December 1995.

[36] T. M. Conte, S. W. Sathaye, and S. Banerjia, "A Persistent Rescheduled-Page Cache for Low Overhead Object Code Compatibility in VLIW Architectures," in *Proceedings of the 29th International Symposium on Microarchitecture (MICRO-29)*, pp. 4–13, IEEE and ACM, December 1996.

[37] K. Ebcioğlu and E. R. Altman, "DAISY: Dynamic Compilation for 100% Architectural Compatibility," IBM Research Report RC 20538, IBM Research Division, August 5, 1996.

[38] I.-C. Chen, J. Coffey, and T. Mudge, "Analysis of Branch Prediction via Data Compression," in *Proceedings of the Seventh Annual International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS-VII)*, pp. 128–137, ACM, October 1996.

[39] J. Emer and N. Gloy, "A Language for Describing Predictors and Its Application to Automatic Synthesis," in *Proceedings of the 24nd Annual International Symposium on Computer Architecture*, IEEE and ACM, June 1997. To appear.

[40] N. Gloy, C. Young, J. Chen, and M. Smith, "An Analysis of Dynamic Branch Prediction Schemes on System Workloads," in *Proceedings of the 23nd Annual International Symposium on Computer Architecture*, pp. 12–21, IEEE and ACM, May 1996.

[41] S. Sechrest, C.-C. Lee, and T. Mudge, "The Role of Adaptivity in Two-Level Adaptive Branch Prediction," in *Proceedings of the 28th International Symposium on Microarchitecture (MICRO-28)*, pp. 264–269, IEEE and ACM, November/December 1995.

[42] S. Sechrest, C.-C. Lee, and T. Mudge, "Correlation and Aliasing in Dynamic Branch Predictors," in *Proceedings of the 23nd Annual International Symposium on Computer Architecture*, pp. 22–32, IEEE and ACM, May 1996.

[43] D. B. Papworth, "Tuning the Pentium Pro Microarchitecture," *MICRO*, vol. 16, pp. 8–15, April 1996.

[44] L. Gwennap, "Intel's P6 Uses Decoupled Superscalar Design," *Microprocessor Report*, vol. 9, pp. 9–15, February 16, 1995.

[45] H. C. Torng and S. Vassiliadis, Eds., *Instruction-Level Parallel Processors*. Los Alamitos, California: IEEE Computer Society Press, 1995.