Effects of Branch Predictors on Execution Time

Jakob Engblom*
Dept. of Information Technology, Uppsala University
P.O. Box 337, SE-751 05 Uppsala, Sweden
jakob@it.uu.se / http://www.docs.uu.se/~jakob

April 17, 2002

Abstract

This technical report describes the result of a short experimental investigation into the effect of branch predictors on the execution times of tight loops on modern aggressive microprocessors. The same code was tried on Pentium III, Athlon, UltraSparc II, and UltraSparc III processors. For reference, the same experiment was carried out on a simple V850E processor to determine the behavior without branch prediction.

The results indicate that advanced branch predictors give a very high and hard-to-understand variation in the execution time of loops, and that this effect can be very big relative to the execution time of other instructions.

1. Introduction

Branch Prediction is a technique intended to reduce the penalty of branches in deep pipelines [3]. The simplest form is to perform static prediction, i.e. continue fetching instructions from the taken or not-taken path while fetching instructions [6]. The simplest form of dynamic predictors are the local predictors that use a branch history for each branch that is used to predict its outcome [1]. Even more advanced approaches also employ a global predictor [5], which tracks the global history of recently taken and not taken branches and helps detects patterns across branches.

Dynamic branch predictors reduce execution time predictability by introducing a history state in the processor. In this report, we quantify the effects of branch predictors using measurements, in order to demonstrate just how annoyingly complex the behavior can be.

We have run the same experiment, described in Section 2, on a number of modern high-end processors, and observed some interesting behaviors.

This report is not an attempt to perform a thorough investigation of branch prediction and its relation to worst-case execution time (WCET) analysis, but rather a short summary of some interesting behaviors stumbled on during a graduate course in advanced computer architecture. We were using microbenchmarks to measure the memory system latencies of some uniprocessor systems. When doing these experiments, it became apparent that the results were almost chaotic on the Pentium III, which could be traced to the branch predictor. This prompted the present investigation.

2. Experimental Setup

The setup of the experiment is very simple:

```
for{inner_iterations=1; inner_iterations<32; inner_iterations++)
  {
    starttimer();
    for(n=0; n < 1000000; n++)               // OUTER LOOP
      {
        for(i=0; i < inner_iterations; i++)  // INNER LOOP
          {
             __nop();   // Some compiler-dependent way to get a nop
          }
      }
    stoptimer();
    recordtime();
  }
```

The result of compiling this code is typically an inner loop of about four instructions, with an outer loop containing about four instructions before and after the inner loop. The entire loop comfortably fits in the cache of any cached machine, so we can assume that the memory system does not influence the results.

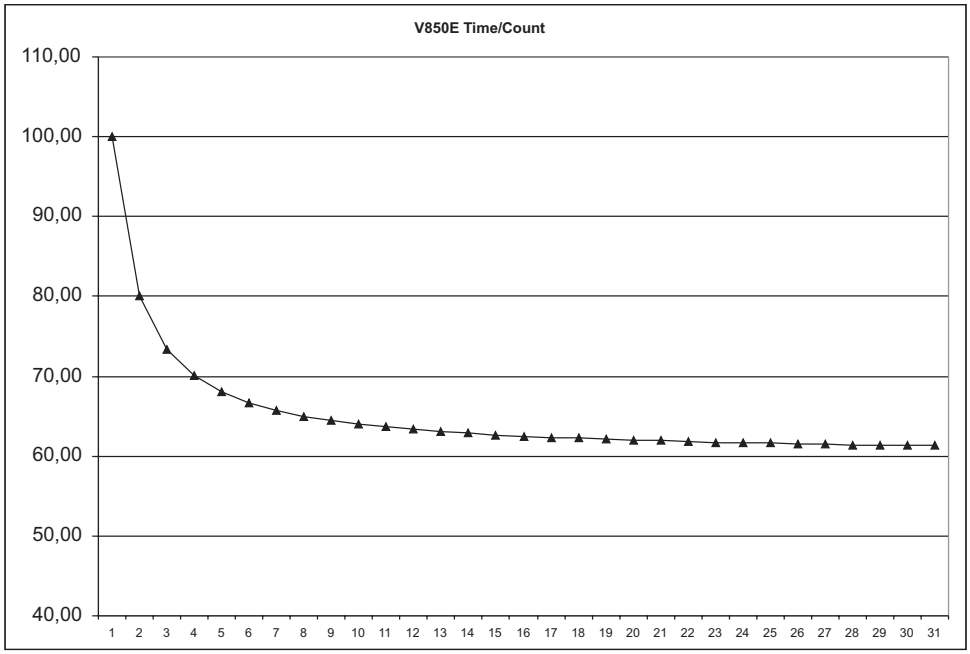
It is clear that the expected result, in the absence of branch prediction, is that the total time for the outer loop should be the greatest for `inner_iterations` = 31, and the least for `inner_iterations` = 1. If we divide the total execution time by `inner_iterations`, we should get a progressively lower value, as the overhead of the outer loop is spread over more executions of the inner loop. However, with branch prediction, stranger things happen, as will be seen in the next few sections.

3. V850E

The NEC V850E is a very simple processor that does not use any kind of branch prediction, apart from the fact that it keeps fetching instructions from the not-taken direction of the branch, making the fall-through case quite cheap. However, there is no dynamic branch prediction, and it behaves like expected with a nice smooth curve.

On this processor, we get the following execution times for the loop (note that we have scaled the "Time/Count" column by a factor 100, to make it more comparable to later measurements):

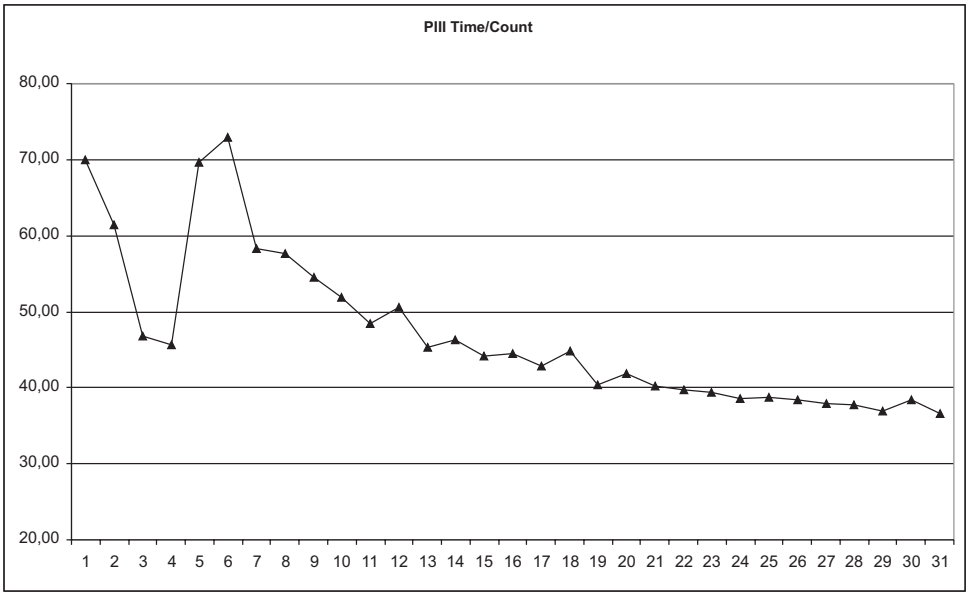
Iteration Count	Time	Time/Count
1	10006	100,06
2	16006	80,03
3	22006	73,35
4	28006	70,02
5	34006	68,01
6	40006	66,68
7	46006	65,72
8	52006	65,01
9	58006	64,45
10	64006	64,01
11	70006	63,64
12	76006	63,34
13	82006	63,08
14	88006	62,86
15	94006	62,67
16	100006	62,50
17	106006	62,36
18	112006	62,23
19	118006	62,11
20	124006	62,00
21	130006	61,91
22	136006	61,82
23	142006	61,74
24	148006	61,67
25	154006	61,60
26	160006	61,54
27	166006	61,48
28	172006	61,43
29	178006	61,38
30	184006	61,34
31	190006	61,29



4. Pentium III

The Pentium III employs an advanced global branch predictor, since it needs to keep a very deep pipeline filled. On this processor, the execution time for the inner loop varies quite significantly, and the pattern for the "Time/Count" value is quite strange, as seen in the following table and graph:

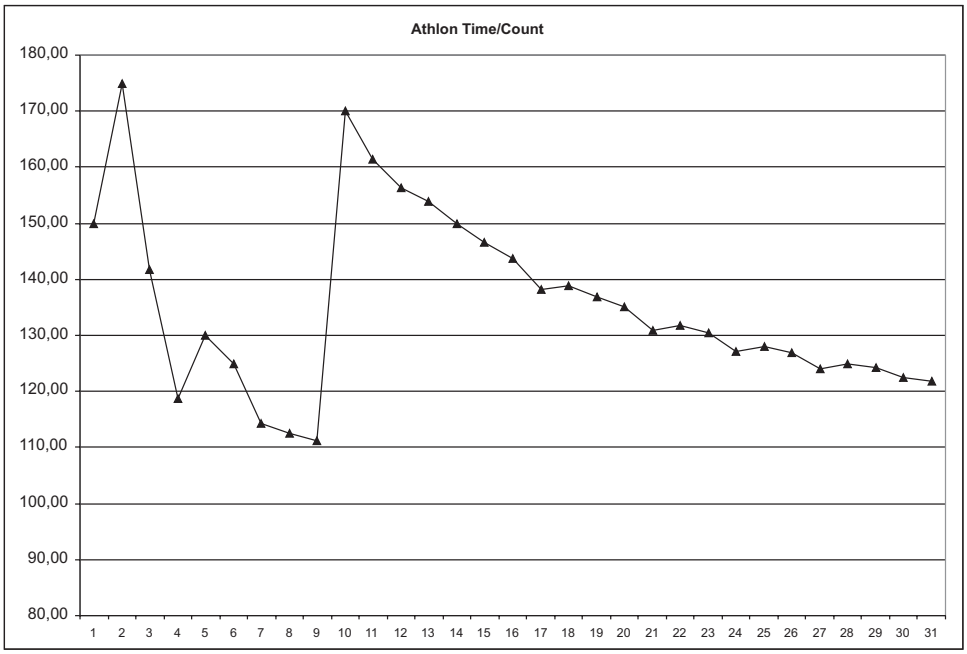
Iteration Count	Time	Time/Count
1	70,00	70,00
2	122,75	61,38
3	140,25	46,75
4	182,75	45,69
5	348,00	69,60
6	438,00	73,00
7	408,25	58,32
8	460,50	57,56
9	490,75	54,53
10	518,25	51,83
11	533,25	48,48
12	606,00	50,50
13	588,25	45,25
14	648,50	46,32
15	663,25	44,22
16	711,00	44,44
17	728,75	42,87
18	806,25	44,79
19	766,00	40,32
20	838,75	41,94
21	846,25	40,30
22	873,50	39,70
23	906,50	39,41
24	926,25	38,59
25	968,75	38,75
26	999,25	38,43
27	1023,75	37,92
28	1059,25	37,83
29	1071,25	36,94
30	1154,25	38,48
31	1136,75	36,67



5. Athlon

The Athlon employs an even more advanced global branch predictor than the Pentium III, and exhibits even more complex behavior. Especially noticeable is the strange jump at iteration count ten, which seems to indicate some kind of state transition in the branch prediction mechanism. But the execution time per iteration is actually lower before ten, which is very strange. Is it more expensive to jump back in a loop that runs 10 iterations?

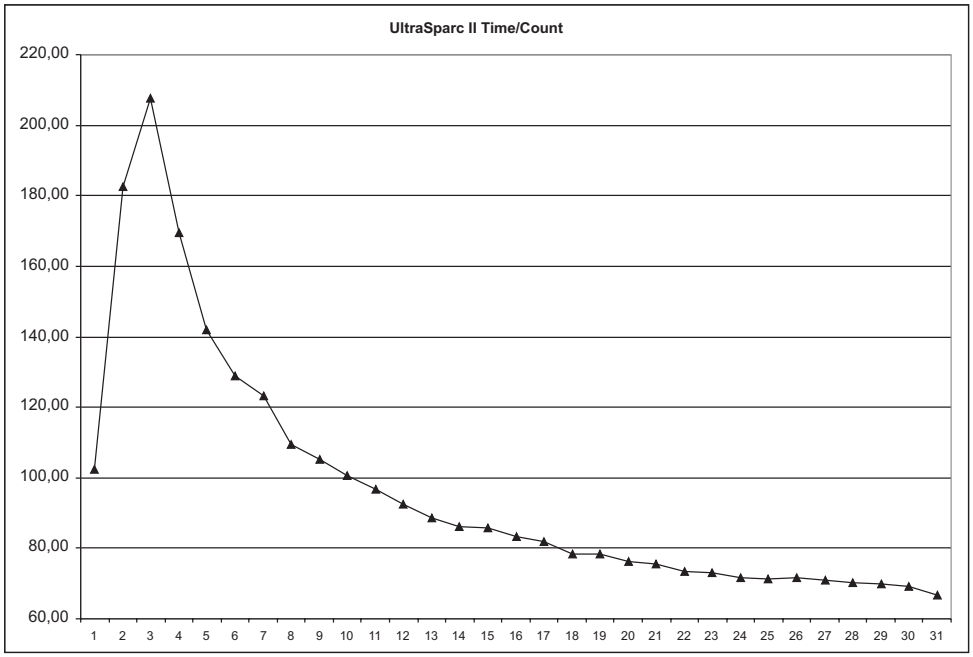
Iteration Count	Time	Time/Count
1	150,00	150,00
2	350,00	175,00
3	425,00	141,67
4	475,00	118,75
5	650,00	130,00
6	750,00	125,00
7	800,00	114,29
8	900,00	112,50
9	1000,00	111,11
10	1700,00	170,00
11	1775,00	161,36
12	1875,00	156,25
13	2000,00	153,85
14	2100,00	150,00
15	2200,00	146,67
16	2300,00	143,75
17	2350,00	138,24
18	2500,00	138,89
19	2600,00	136,84
20	2700,00	135,00
21	2750,00	130,95
22	2900,00	131,82
23	3000,00	130,43
24	3050,00	127,08
25	3200,00	128,00
26	3300,00	126,92
27	3350,00	124,07
28	3500,00	125,00
29	3600,00	124,14
30	3675,00	122,50
31	3775,00	121,77



6. UltraSparc II

The UltraSparc II is a much older and simpler design, but still uses a branch predictor. From the look of the graph, this appears to be purely local, since it initially mispredicts the loop when it iterates a few times, but learns well once the iteration count increases beyond three. The times for "Time/Count" are scaled by 1000. This type of local predictor was analyzed in [1].

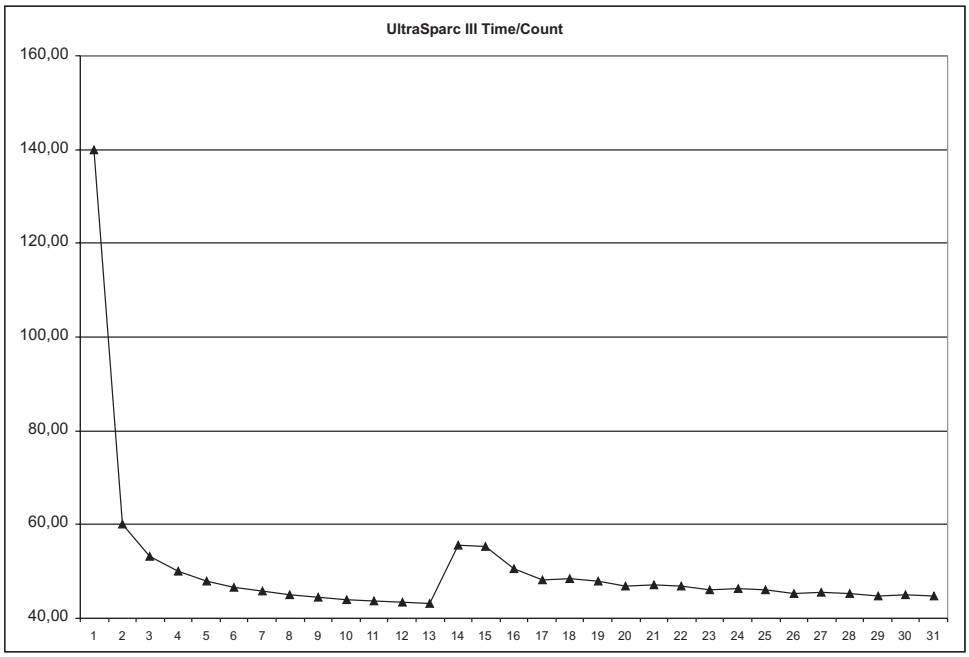
Iteration Count	Time	Time/Count
1	102500	102,50
2	365000	182,50
3	622500	207,50
4	677500	169,38
5	710000	142,00
6	772500	128,75
7	862500	123,21
8	875000	109,38
9	947500	105,28
10	1005000	100,50
11	1065000	96,82
12	1110000	92,50
13	1152500	88,65
14	1205000	86,07
15	1285000	85,67
16	1335000	83,44
17	1390000	81,76
18	1407500	78,19
19	1487500	78,29
20	1527500	76,38
21	1587500	75,60
22	1615000	73,41
23	1682500	73,15
24	1722500	71,77
25	1782500	71,30
26	1862500	71,63
27	1912500	70,83
28	1967500	70,27
29	2025000	69,83
30	2077500	69,25
31	2067500	66,69



7. UltraSparc III

The UltraSparc III seems to employ a very good branch prediction mechanism that quickly learns the behavior of our tight loop nest. However, for some reason there is a hickup at iteration count 14, which just like the Athlon could indicate a switch between different states in the branch predictor.

Iteration Count	Time	Time/Count
1	140000	140,00
2	120000	60,00
3	160000	53,33
4	200000	50,00
5	240000	48,00
6	280000	46,67
7	320000	45,71
8	360000	45,00
9	400000	44,44
10	440000	44,00
11	480000	43,64
12	520000	43,33
13	560000	43,08
14	780000	55,71
15	830000	55,33
16	810000	50,63
17	820000	48,24
18	870000	48,33
19	910000	47,89
20	940000	47,00
21	990000	47,14
22	1030000	46,82
23	1060000	46,09
24	1110000	46,25
25	1150000	46,00
26	1180000	45,38
27	1230000	45,56
28	1270000	45,36
29	1300000	44,83
30	1350000	45,00
31	1390000	44,84



8. Discussion

From the measurements in the previous sections, it is easy to see that the effect of branch predictors on the execution time of a program is very large.

The global predictors on the Pentium III and Athlon have a behavior that seems almost random, even though the underlying mechanisms are quite simple when described. Accounting for this type of predictors in WCET analysis requires global analysis, since the outcome of each and every branch can affect how the next branch is taken. Even for the very simple loop structure used in this experiment, we can see that for low loop counts, the branch predictor has a very poor prediction rate.

On the Pentium III, it also seems that the small size of the inner loop is a problem, since the processor does not have the time to update the branch predictor tables as the loop iterates. This makes the learning slower, and indeed Intel recommends that loops should contain at least six or seven instructions in order to give the branch predictor time to update. Analyzing this kind of microarchitectural behavior is not a promising proposition.

A processor without branch prediction, like the V850E, has a very regular behavior where more instructions executed means a greater total execution time, while on the Pentium III, increasing the number of loop iterations can actually decrease the execution time, as observed between iteration counts six and seven. This makes WCET analysis much more complex, since we cannot assume that iterating a loop for the maximal number of iterations comprise the worst case, and is a case analogous to timing anomalies [4, 2], requiring WCET analysis to consider very many different scenarios in order to find the worst case. Note that despite the flailing behavior of the Athlon, there is no case where increasing the number of loop iterations in the inner loop decreases the execution time.

References

[1] A. Colin and I. Puaut. Worst Case Execution Time Analysis for a Processor with Branch Prediction. *Journal of Real-Time Systems*, 18(2/3):249–274, May 2000.

[2] Jakob Engblom. *Processor Pipelines and Static Worst-Case Execution Time Analysis*. PhD thesis, Dept. of Information Technology, Uppsala University, April 2002. Acta Universitatis Upsaliensis, Dissertations from the Faculty of Science and Technology 36, `http://publications.uu.se/theses/`.

[3] J. L. Hennessy and D. A. Patterson. *Computer Architecture A Quantitative Approach.* Morgan Kaufmann Publishers Inc., 2^{nd} edition, 1996. ISBN 1-55860-329-8.

[4] Thomas Lundqvist and Per Stenström. Timing Anomalies in Dynamically Scheduled Microprocessors. In *Proc. 20^{th} IEEE Real-Time Systems Symposium (RTSS'99)*, December 1999.

[5] Tulika Mitra and Abhik Roychoudhury. Effects of Branch Prediction on Worst Case Execution Time of Programs. Technical Report 11-01, National University of Singapore (NUS), November 2001.

[6] NEC Corporation. *V850E/MS1 32/16-bit Single Chip Microcontroller: Architecture*, 3^{rd} edition, January 1999. Document no. U12197EJ3V0UM00.