NC State University

Department of Electrical and Computer Engineering

ECE 521: Fall 2015 (Rotenberg)

Project #2: Branch Prediction

By

Laxman Sole

NCSU Honor Pledge: "I have neither given nor received unauthorized aid on this test or assignment."

Student's electronic signature: Laxman Sole

Course number: ECE521

Table of Contents

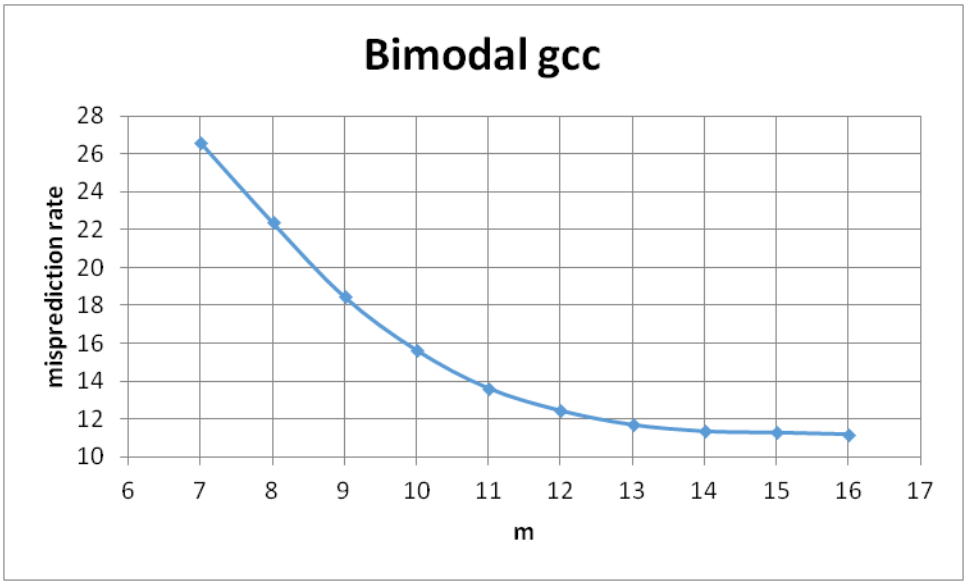
Bimodal Predictor

1) *gcc* Benchmark

Data:

m	GCC
7	26.65
8	22.43
9	18.49
10	15.67
11	13.65
12	12.47
13	11.72
14	11.37
15	11.3
16	11.21

Graph:

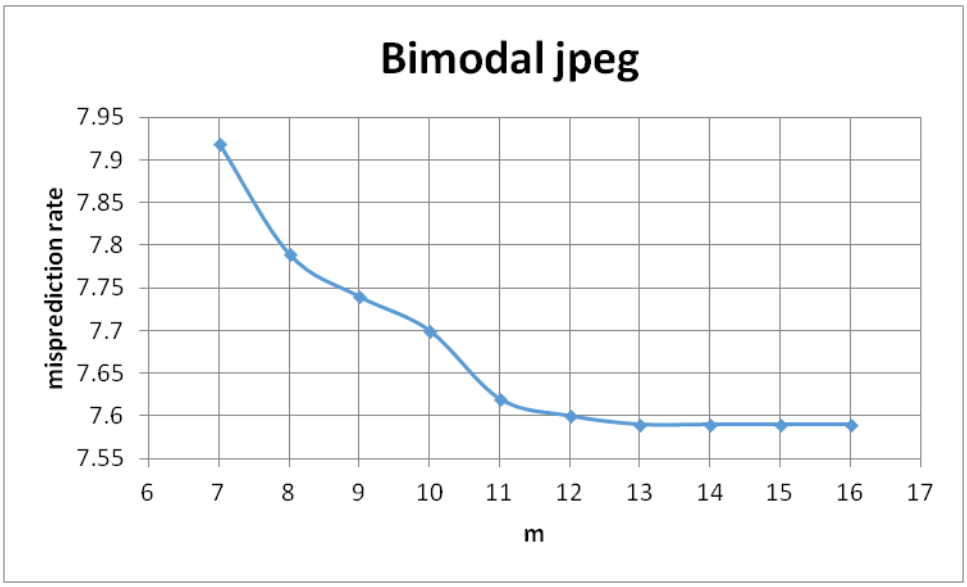


2) *jpeg* Benchmark

Data:

m	jpeg
7	7.92
8	7.79
9	7.74
10	7.7
11	7.62
12	7.6
13	7.59
14	7.59
15	7.59
16	7.59

Graph:

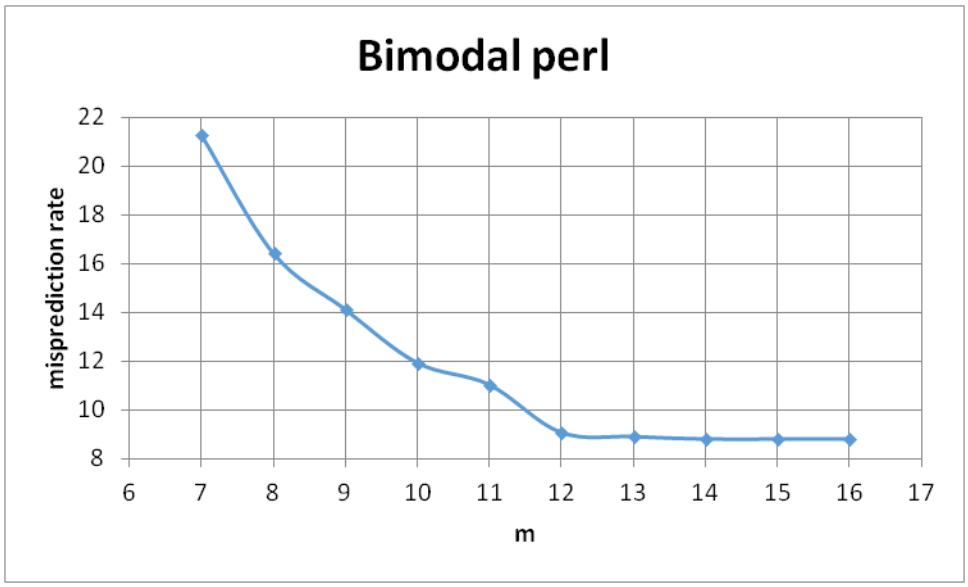


3) *perl* Benchmark

Data

m	perl
7	21.31
8	16.45
9	14.14
10	11.95
11	11.05
12	9.09
13	8.92
14	8.82
15	8.82
16	8.83

Grpah



4) Analysis:

From the collected data, we can see that misprediction rate decreases with increase in the value of m i.e. no. of pc bits used to index the bimodal table. The rate of decrease is high for lower value of m and it saturates after particular value of m. This behavior is same for all three benchmarks.

Small value of m indicates multiple branches shares the same counter and because of this for small value of m, misprediction rate is higher. This is similar to capacity and conflict misses in terms of cache hierarchy. Also for higher values of m, misprediction rate saturates. Saturated value is the lowest possible value we can achieve. This value doesn't change even if we increase m value further. This is analogous to compulsory misses in terms of cache hierarchy.

Better misprediction rate is achieved at the cost of higher memory and other resources.

For different benchmarks, misprediction rate is different even if value of m is same. gcc benchmark shows variation from 26.65% to 11.21% when m ranges from 7 to 16. jpeg benchmark do not show much variation over the range of m and varies very less %. perl benchmark varies comparable to gcc benchmark but misprediction rate is lower than gcc benchmark for same value of m.

5) Design:

m	GCC	jpeg	perl
7	26.65	7.92	21.31
8	22.43	7.79	16.45
9	18.49	7.74	14.14
10	15.67	7.7	11.95
11	13.65	7.62	11.05
12	12.47	7.6	9.09
13	11.72	7.59	8.92
14	11.37	7.59	8.82
15	11.3	7.59	8.82
16	11.21	7.59	8.83

With the 16 kilobyte budget constraint, we can achieve the minimum misprediction rate and minimum predictor cost with following configurations:

i) *gcc* benchmark : m = 12 and misprediction rate = 12.47%
ii) *jpeg* benchmark : m= 11 and misprediction rate = 7.62%
iii) *perl* benchmark : m = 12 and misprediction rate = 9.09%

These values can be considered as knee points of the graph.

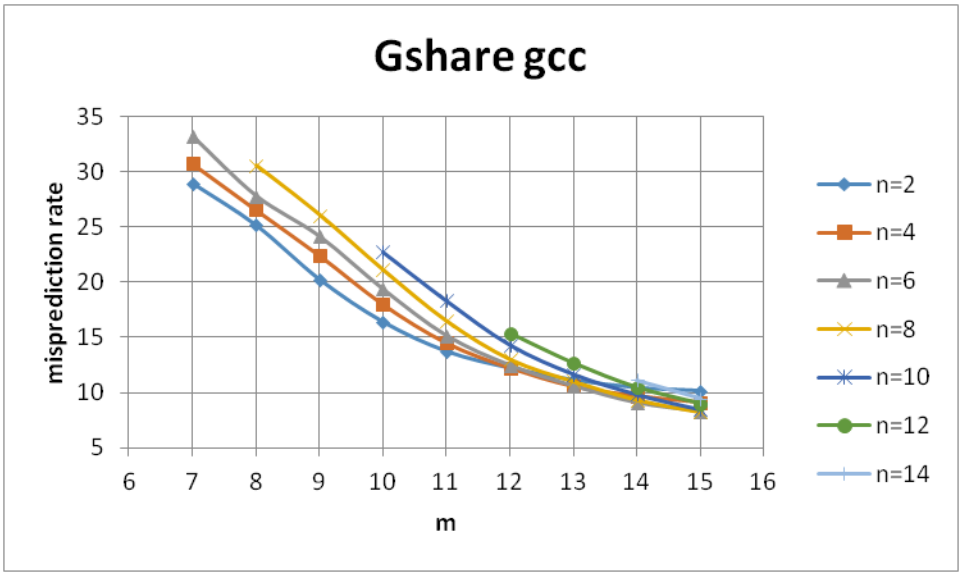
Gshare Predictor

1) *gcc* Benchmark

Data

m	2	4	6	8	10	12	14
7	28.98	30.76	33.22				
8	25.18	26.57	27.82	30.56			
9	20.25	22.43	24.14	26.08			
10	16.39	17.99	19.36	21.1	22.77		
11	13.71	14.49	15.14	16.47	18.34		
12	12.2	12.23	12.46	13	14.33	15.4	
13	11.11	10.57	10.59	11	11.68	12.68	
14	10.42	9.69	9.08	9.34	9.83	10.48	11.13
15	10.13	9.13	8.3	8.22	8.46	9.01	9.48

Graph

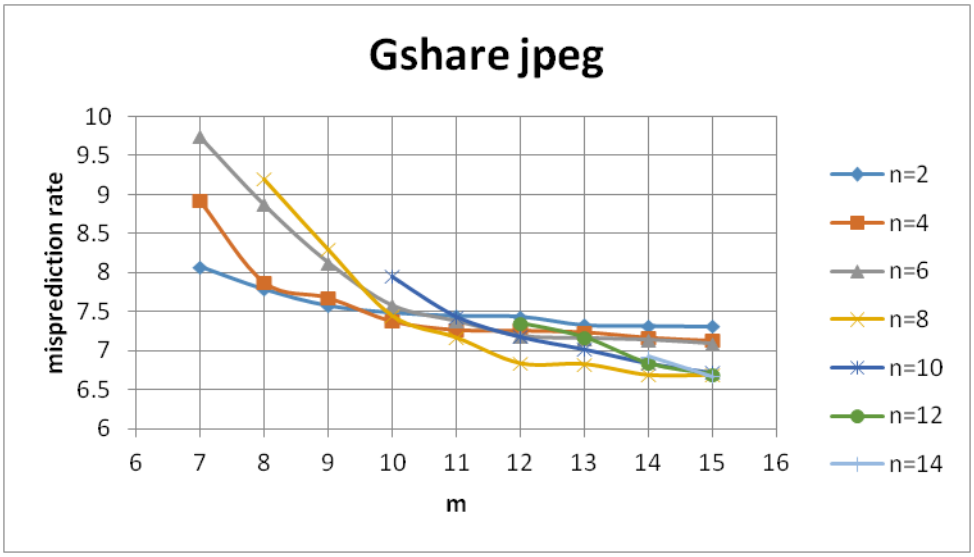


2) *jpeg* Benchmark

Data

m	2	4	6	8	10	12	14
7	8.08	8.92	9.74				
8	7.79	7.88	8.87	9.2			
9	7.58	7.68	8.13	8.3			
10	7.49	7.38	7.58	7.45	7.95		
11	7.45	7.27	7.38	7.17	7.44		
12	7.44	7.26	7.19	6.84	7.18	7.35	
13	7.33	7.24	7.16	6.83	7.02	7.17	
14	7.32	7.17	7.14	6.69	6.84	6.84	6.93
15	7.31	7.13	7.09	6.69	6.72	6.7	6.67

Graph

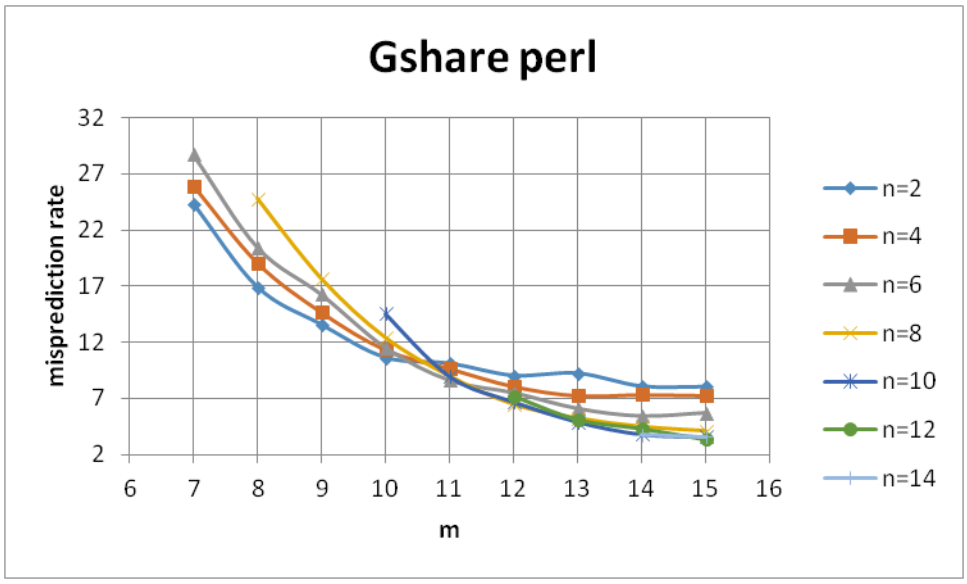


3) *perl* Benchmark

Data

m	2	4	6	8	10	12	14
7	24.34	25.96	28.71				
8	16.92	19.09	20.45	24.79			
9	13.57	14.68	16.25	17.66			
10	10.63	11.35	11.52	12.42	14.57		
11	10.11	9.68	8.6	9	8.98		
12	9.03	8.09	7.5	6.49	6.71	7.16	
13	9.23	7.27	6.09	5.26	4.92	5.09	
14	8.07	7.35	5.43	4.51	3.8	4.3	3.75
15	8.02	7.28	5.71	4.13	3.58	3.35	3.58

Grpah



4) Analysis:

From the data we can see that, for all three benchmarks misprediction rate decreases with increase in value of m. Also for the same smaller values of m misprediction rate increases with increased value of n. This behavior is different for higher values of m, where misprediction rate initially high for lower value of n and it decrease with increased n and after particular value of n it again increases.
This behavior is same for all the three benchmarks.
For gcc, till m= 12, n = 2 gives the best misprediction rate. After 12, it gives the best results for n=4, 6, 8 for m = 13, 14, 15 respectively.
For jpeg, till m= 9, n = 2 gives the best misprediction rate. After 9, it gives the best results for n=4, 4, 8, 8, 8, 14 for m = 10, 11, 12, 13, 14, 15 respectively.
For perl, till m= 10, n = 2 gives best misprediction rate. After 10, it gives the best results for n= 6, 8, 10, 10, 12 for m = 11, 12, 13, 14, 15 respectively.

This behavior is mainly because for smaller values of m, many branches indexed to the same entry in gshare table and added global history to that make the prediction worst. But for higher values of m, relatively very less branches are mapped to each index and in this case global history register value helps it to predict better.

5) Design:

NOTE: I have assumed the predictor cost >= $n+2*2^m$ bits.
So $16*2^{10}*2^3>=n+2*2^{15}$ bits. So my max value of m is 15.

gcc:

m	2	4	6	8	10	12	14
7	28.98	30.76	33.22				
8	25.18	26.57	27.82	30.56			
9	20.25	22.43	24.14	26.08			
10	16.39	17.99	19.36	21.1	22.77		
11	13.71	14.49	15.14	16.47	18.34		
12	12.2	12.23	12.46	13	14.33	15.4	
13	11.11	10.57	10.59	11	11.68	12.68	
14	10.42	9.69	9.08	9.34	9.83	10.48	11.13
15	10.13	9.13	8.3	8.22	8.46	9.01	9.48

In my opinion, for gcc, m=13 and n=4 gives the best tradeoff between minimum misprediction rate and minimum predictor cost in bits.

jpeg:

m	2	4	6	8	10	12	14
7	8.08	8.92	9.74				
8	7.79	7.88	8.87	9.2			
9	7.58	7.68	8.13	8.3			
10	7.49	7.38	7.58	7.45	7.95		
11	7.45	7.27	7.38	7.17	7.44		
12	7.44	7.26	7.19	6.84	7.18	7.35	
13	7.33	7.24	7.16	6.83	7.02	7.17	
14	7.32	7.17	7.14	6.69	6.84	6.84	6.93
15	7.31	7.13	7.09	6.69	6.72	6.7	6.67

In my opinion, for jpeg, m=12 and n=8 gives the best tradeoff between minimum misprediction rate and minimum predictor cost in bits.

perl:

m	2	4	6	8	10	12	14
7	24.34	25.96	28.71				
8	16.92	19.09	20.45	24.79			
9	13.57	14.68	16.25	17.66			
10	10.63	11.35	11.52	12.42	14.57		
11	10.11	9.68	8.6	9	8.98		
12	9.03	8.09	7.5	6.49	6.71	7.16	
13	9.23	7.27	6.09	5.26	4.92	5.09	
14	8.07	7.35	5.43	4.51	3.8	4.3	3.75
15	8.02	7.28	5.71	4.13	3.58	3.35	3.58

In my opinion, for perl, m=13 and n=10 gives the best tradeoff between minimum misprediction rate and minimum predictor cost in bits.