NC State University

Department of Electrical and Computer Engineering

ECE 463/521: Fall 2015 (Rotenberg)

Project #2: Branch Prediction

by

ARJUN AUGUSTINE

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

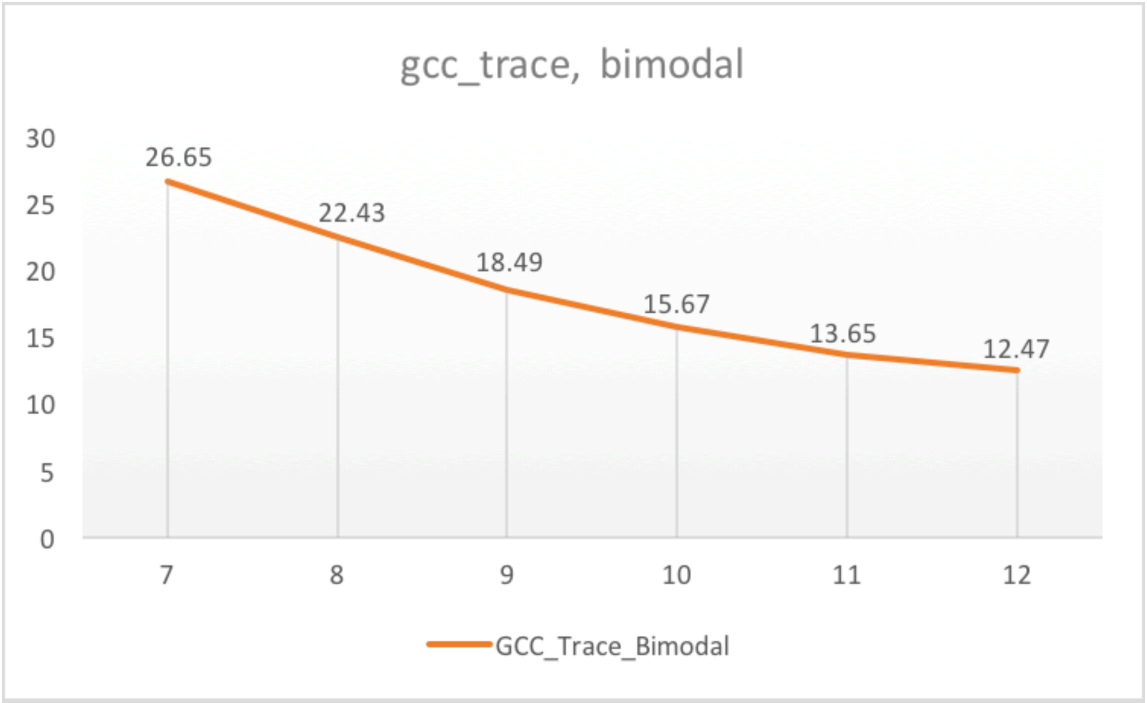
Student's electronic signature: ______Arjun Augustine______
(sign by typing your name)

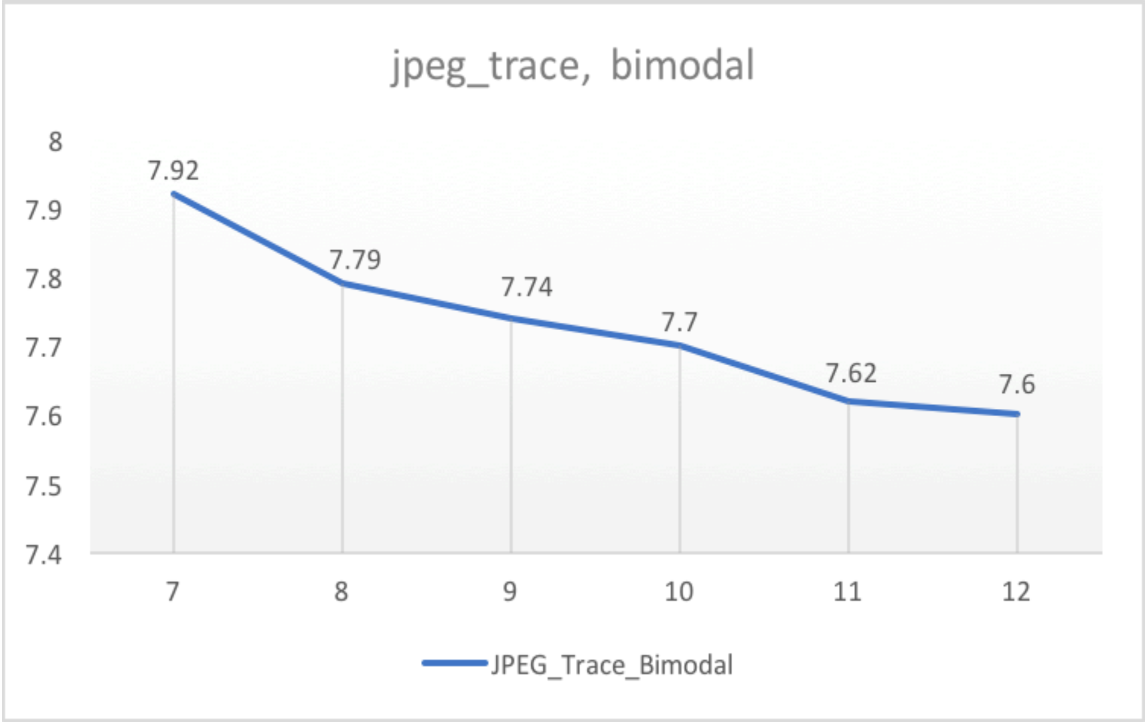
Course number: _______521_________
(463 or 521 ?)

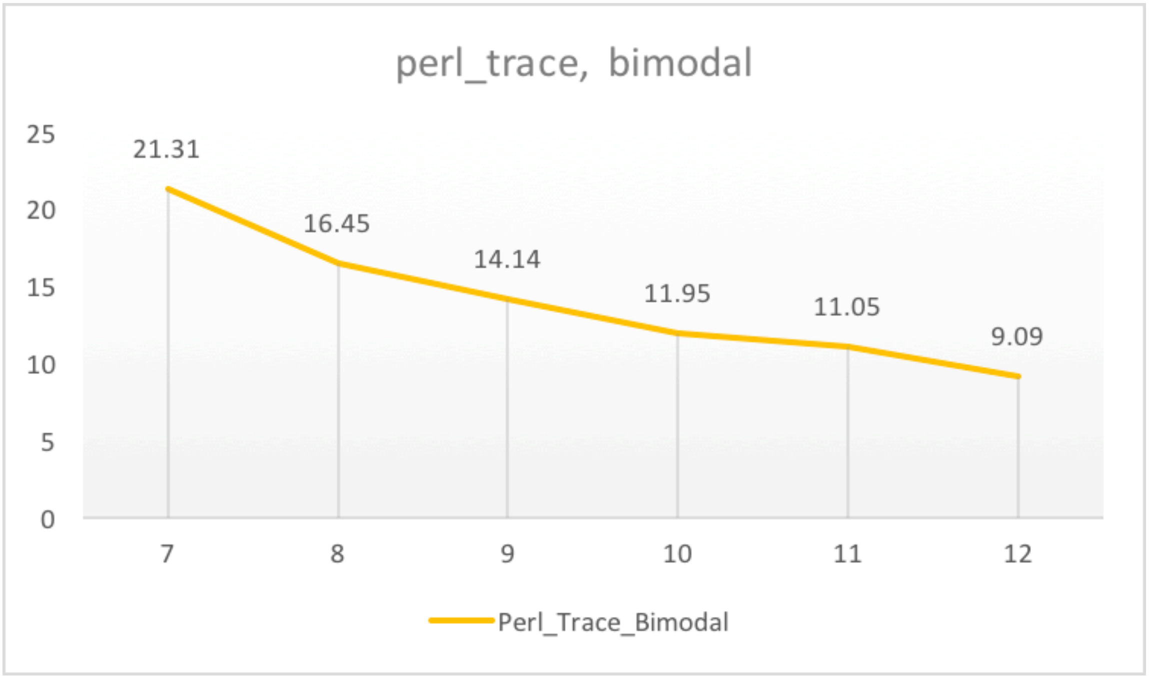
1. BIMODAL PREDICTOR

Simulate BIMODAL PREDICTOR configurations for 7 ≤ m ≤ 12. Use the traces gcc, jpeg, and perl in the trace directory.

1.1. [15 or 10 points] Graphs: Produce one graph for each benchmark. Graph title: "<benchmark>, bimodal". Y-axis: branch misprediction rate. X-axis: m. Per graph, there should be only 1 curve consisting of 6 data-points (connect the data-points with a line).







1.2. [5 points] Analysis: Draw conclusions and discuss trends. Discuss similarities/differences among benchmarks.

As indicated by the three graphs shown, misprediction rate goes down with increasing M value (size of predictor table). But the U-shape of the graphs hints that the decrease in misprediction rate bottoms out beyond a certain M value. Jpeg trace has the least misprediction rate among the three implying that the trace forms better predictable patterns. However, Gcc and Perl traces behave somewhat similar.

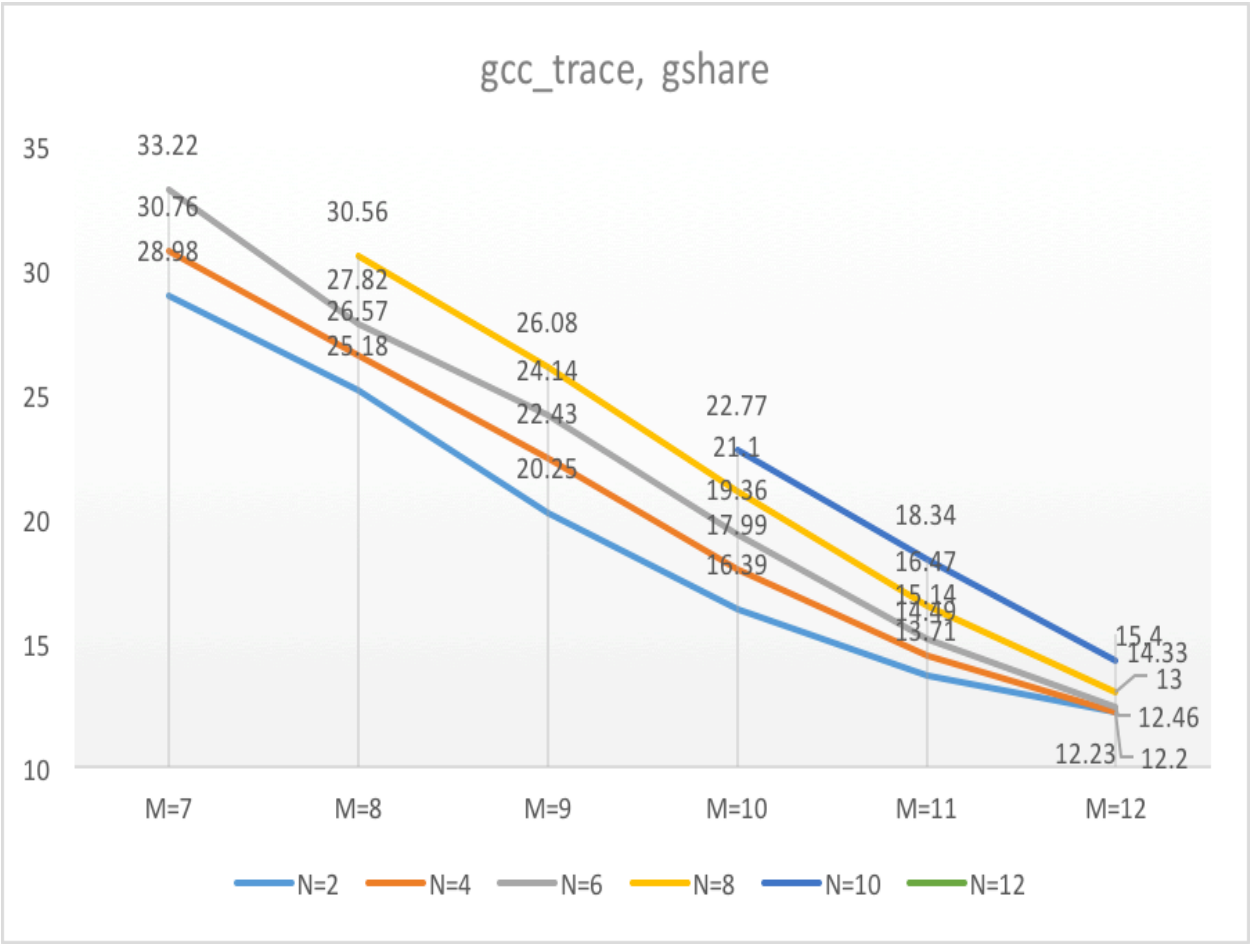
1.3. [5 points] Design: For each trace, choose a bimodal predictor design that minimizes misprediction rate AND minimizes predictor cost in bits. You have a maximum budget of 16 kilobytes of storage. When "minimizing" misprediction rate, look for diminishing returns, i.e., point where misprediction rate levels off and additional hardware provides only minor improvement. Bottom line: use good sense to provide high performance and reasonable cost. Justify your designs with supporting data and explanations.

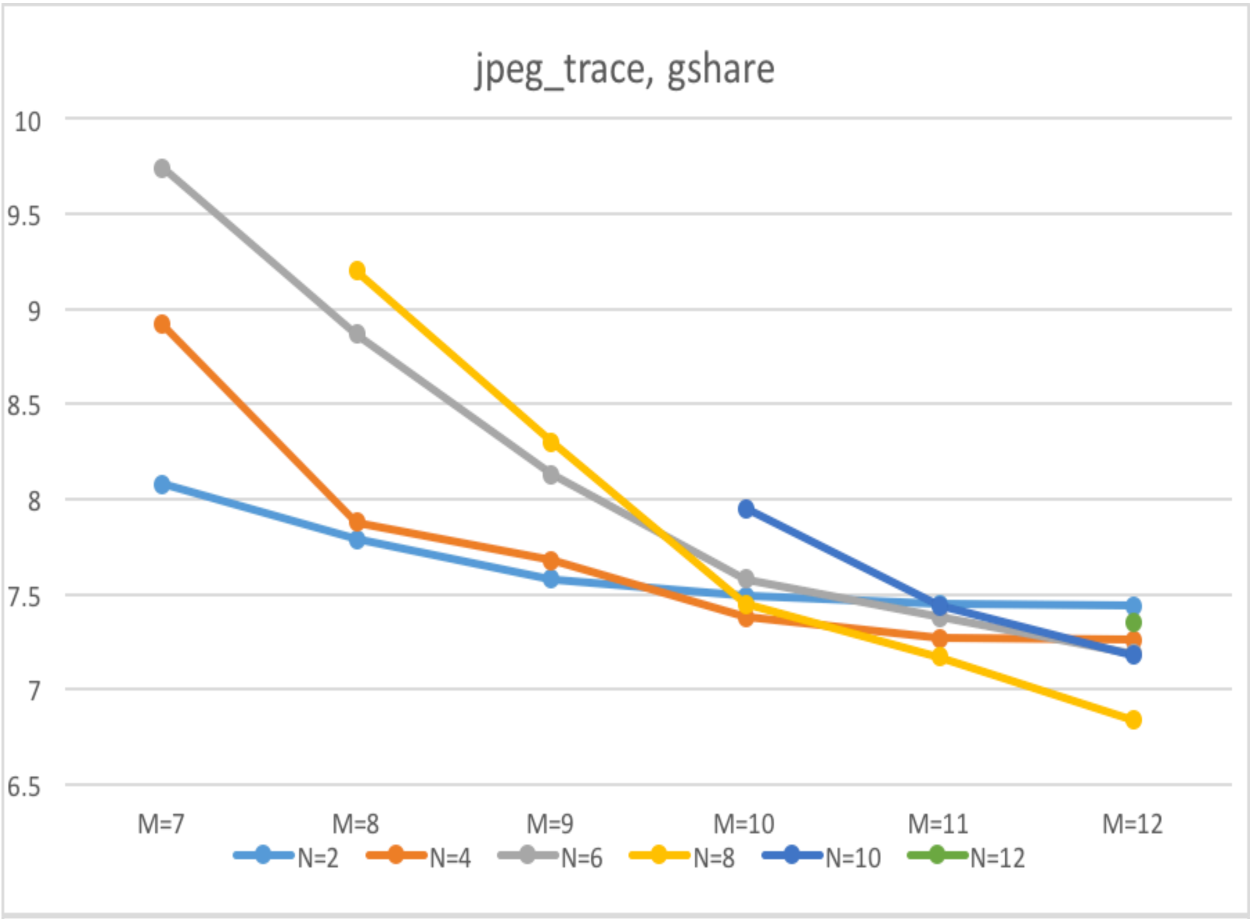
Maximum memory size required for M=12 would be: 2^{12}*2(2bits per counter)/8 = 10KB.
Gcc_trace.txt: There is a near steady improvement in misprediction rate with increasing predictor table size. An M value of 11 would be optimum here since beyond this point, the percentage improvement falls below 10%.
Jpeg_trace.txt: An M value of 7 gives best performance Vs cost trade-off in this case. Increasing the M to 8 or 9 gives improvement of a paltry 1.5% for every doubling in predictor table size.
Perl_trace.txt: An M value of 10 would be best in case of the perl trace graph as returns diminish to less than 10% after this point.

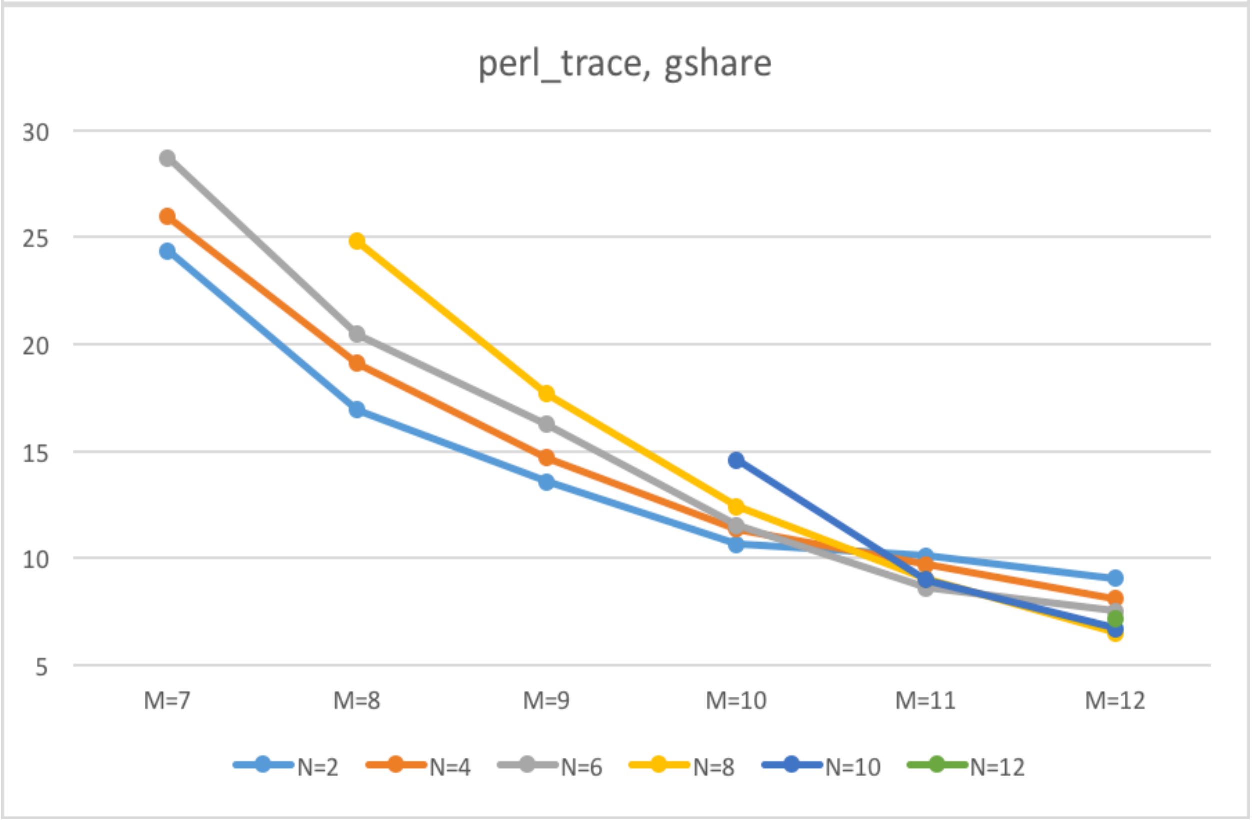
2. GSHARE PREDICTOR

Simulate GSHARE PREDICTOR configurations for 7 ≤ m ≤ 12, 2 ≤ n ≤ m, n is even. Use the traces gcc, jpeg, perl in the trace directory.

2.1 [15 or 10 points] Graphs: Produce one graph for each benchmark. Graph title: "<benchmark>, gshare". Y-axis: branch misprediction rate. X-axis: m. Per graph, there should be a total of 27 data-points plotted as a family of 6 curves. Data-points having the same value of n are connected with a line, i.e., one curve for each value of n. Note that not all curves have the same number of data-points.




jpeg_trace, gshare
10
9.5
9
8.5
8
7.5
7
6.5
M=7
M=8
M=9
M=10
M=11
M=12
N=2
N=4
N=6
N=8
N=10
N=12


perl_trace, gshare
30
25
20
15
10
5
M=7
M=8
M=9
M=10
M=11
M=12
N=2
N=4
N=6
N=8
N=10
N=12

2.2 [5 points] Analysis: Draw conclusions and discuss trends. Discuss similarities/differences among benchmarks.

Similar to the graphs obtained for bimodal predictor, misprediction rates in gshare graphs also tend to go down with increasing M value (size of predictor table). Also, the graphs have a similar U shape indicating diminishing returns for increasing memory in predictor table. Surprisingly, a higher N value for the same M value for smaller M-s (M=7, M=8, M=9 …) seems to increase the misprediction rate even though a bigger history is available. This gets corrected as M value increases, especially in the "perl trace, gshare" graph where a higher N value gives lower misprediction rates for M=11, M=12 etc. This trend is not as evident in the "gcc trace, gshare" graph. Even so, we can conclude that a larger global branch history register gives returns only when the size of branch predictors is sufficiently high. This could mean that the larger branch history captured with larger values of N stood for branches that got evicted from predictor table (when M value was low) and therefore required larger M values (and thus the corresponding branches not getting evicted when required) to contribute to better performance.

2.3 [5 points] Design: For each trace, choose a gshare predictor design that minimizes misprediction rate AND minimizes predictor cost in bits. You have misprediction rate AND minimizes predictor cost in bits. You have a maximum budget of 16 kilobytes of storage. When "minimizing" misprediction rate, look for diminishing returns, i.e., the point where misprediction rate starts leveling off and additional hardware provides only minor improvement. Bottom line: use good sense to provide high performance and reasonable cost. Justify your designs with supporting data and explanations.

In the case of Gshare branch predictors, increasing N value, or the size of global branch history register is not as expensive as increasing M value, or the size of the predictor table. Increasing M has an exponential effect whereas increasing N only increases one bit in the history register. So we can try to use maximum N value for best performance if required.
In the gcc trace graph, an M value of 11 is recommended since returns fall below 10% beyond this point. And for an M value of 11, N=2 gives the best performance.
In the case of the jpeg trace graph, M=7 and N=2 gives lesser performance than M=12, N=8 by about 20% but saves memory by about 2^5 or 32 times. Also, the overall performance of both configurations are not far off from each other, and it seems imprudent to spend as much on so little. So, M=7 and N=2 is recommended.
Finally, for the perl trace graph, increasing M initially gives a lot of performance improvement until M reaches 10. An N value of 2 gives maximum performance at this point.