NC State University

Department of Electrical and Computer Engineering

ECE 463/521: Spring 2010 (Rotenberg)

Project #2: Branch Prediction

By

Mistry Deep Dilipkumar

Student ID – 000962873

ddmistry@ncsu.edu

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

Student's electronic signature: Mistry Deep Dilipkumar

Course number: 521

1. Introduction

The main aim of this project is to simulate different types of branch predictors. In this project we simulate three different types of branch predictors

- bimodal
- gshare
- hybrid

The **bimodal** predictor predicts the branch outcome on the basis of local branch history recorded by a two bit counter.
The **gshare** predictor predicts the branch outcome on the basis of local branch history recorded by a two bit counter and in addition to that, on the basis of a global branch history register.
The **hybrid** predictor has both the predictors and then selects the type of predictor dynamically on the basis of the choose table.

2. Implementation

I have implemented the simulator called “sim” in JAVA. It takes in the command line arguments as follows:

```
sim <type> <m> <tracefile>                                for bimodal,
sim <type> <m> <n> <tracefile>                            for gshare,
sim <type> <k> <m1> <n> <m2> <tracefile>                  for hybrid,
```

The **bimodal** predictor generates a prediction table of size 2^m and then implements a 2 bit counter in that table. It is initialized to value 2 i.e. weakly taken. On every branch, it indexes into the table by taking index as m+1 through 2 of the PC. It checks the value of the counter there and if the value is less than 2, then it predicts that branch is “not taken” else it predicts that branch is taken.

The **gshare** predictor also generates a prediction table of size 2^m and then implements a 2 bit counter in that table. Its operation is exactly similar to that of the bimodal predictor except for the fact that it uses a different index. It keeps a global branch history register which has values 0 or 1 depending on past history of branches. It generates an index by XORing the current global branch history register with the uppermost n bits of the m PC bits. It predicts whether the branches are taken or not taken in a similar way to bimodal predictor.

The **hybrid** predictor contains both the predictors. It first obtains both the predictions. Then it uses a chooser table to choose which prediction to choose. If the chooser counter value is greater than or equal to 2, then it chooses the prediction that was obtained from the gshare predictor, otherwise it uses the prediction that was obtained from the bimodal predictor.

In each mode, the simulator generates results in which it prints the contents of the prediction table and in case of a hybrid predictor also a chooser table. Also the simulator takes in actual branch outcomes given in the tracefile and counts the number of mispredictions and misprediction rate.

3. Analysis of bimodal predictor

3.1 Bimodal - gcc trace

Here we simulate the bimodal predictor with different values of “m” ranging from 7 to 12. We obtain the following results.

m	misprediction rate
7	26.65 %
8	22.43 %
9	18.49 %
10	15.67 %
11	13.65 %
12	12.47 %

Using the above values we plot the graph of misprediction rate VS the values of m.

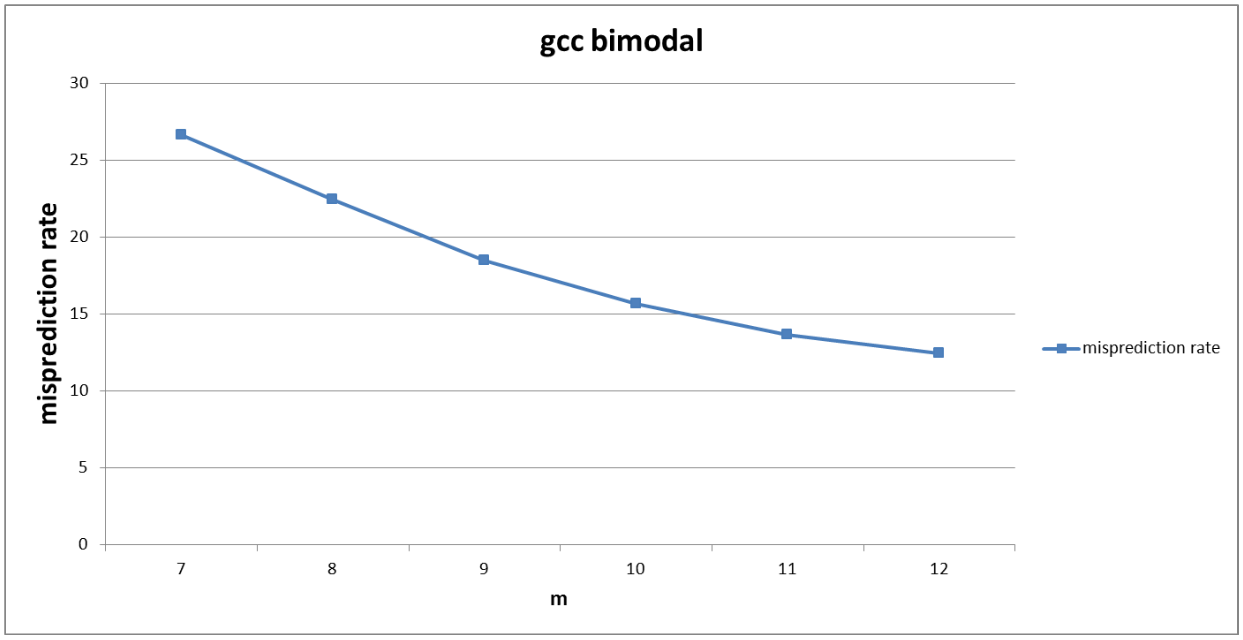

Fig 1

3.2 Bimodal - jpeg trace

Here we simulate the bimodal predictor with different values of "m" ranging from 7 to 12. We obtain the following results.

m	misprediction rate
7	7.92 %
8	7.79 %
9	7.74 %
10	7.7 %
11	7.62 %
12	7.6 %

Using the above values we plot the graph of misprediction rate VS the values of m.

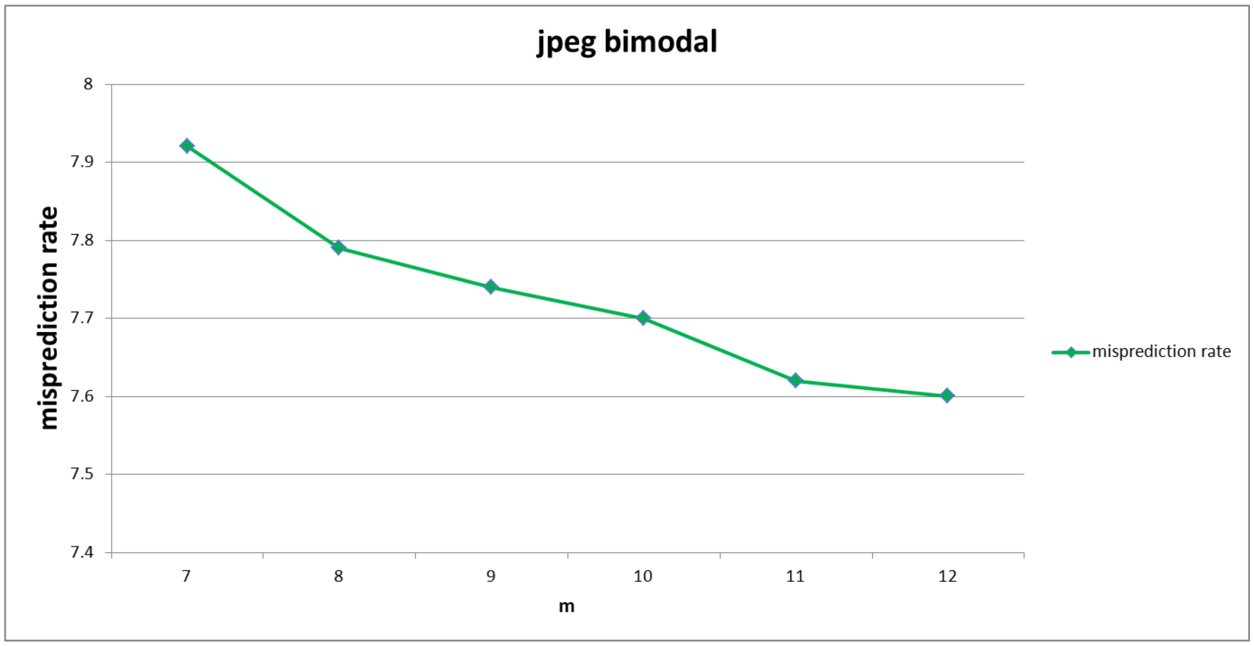

Fig 2

3.3 Bimodal - perl trace

Here we simulate the bimodal predictor with different values of “m” ranging from 7 to 12. We obtain the following results.

m	misprediction rate
7	21.31 %
8	16.45 %
9	14.14 %
10	11.95 %
11	11.05 %
12	9.09 %

Using the above values we plot the graph of misprediction rate VS the values of m.

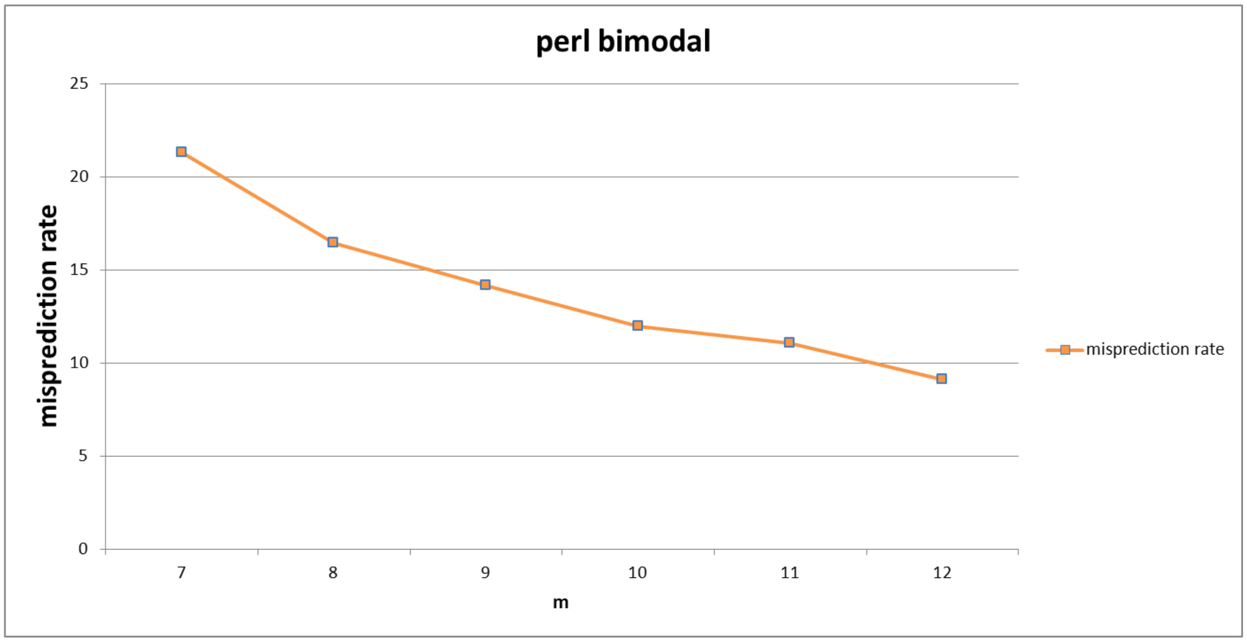

Fig 3

3.4 Comparison and analysis

From sections 3.1, 3.2 and 3.3 we observe the following:

- The jpeg trace has the least misprediction rate among all the tracefiles.
- This indicates that jpeg trace is the easiest to predict and it has predictable patterns as compared to gcc and perl traces.
- In all the graphs, the misprediction rate tends to decrease as the value of m increases, this is due to the fact that as m increases, the size of the prediction table increases thereby allowing more entries in it.
- This results in better recognitions of more patterns thereby reducing the misprediction rate.

3.5 Optimum Design

Now, the optimum design for the bimodal predictor would be,

It requires 2×2^m bits for storage. Now maximum allowable storage is given as 16 KB. To minimize the misprediction rate, we can intuitively predict that the least rate would be for the predictor with largest m.

Thus,

$$16KB = 2 \times 2^m$$

$$2^4 \times 2^{10} \times 2^3 = 2 \times 2^m$$

$$2^m = 2^{16}$$

$$m = 16$$

So the best design for all the three cases is for m = 16.

I simulated the bimodal predictor for the value of m = 16 and got following results.

file	misprediction rate
gcc_trace	11.21 %
jpeg_trace	7.59 %
perl_trace	8.83 %

4. Analysis of gshare predictor

4.1 gshare - gcc trace

Here we simulate the gshare predictor with different values of “m” ranging from 7 to 12.
And for each such value we take even values of n from 0 to m.
We obtain the following results.

m	n=2	n=4	n=6	n=8	n=10	n=12
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 %

Using the above values we plot the graph of misprediction rate VS the values of n for each m.

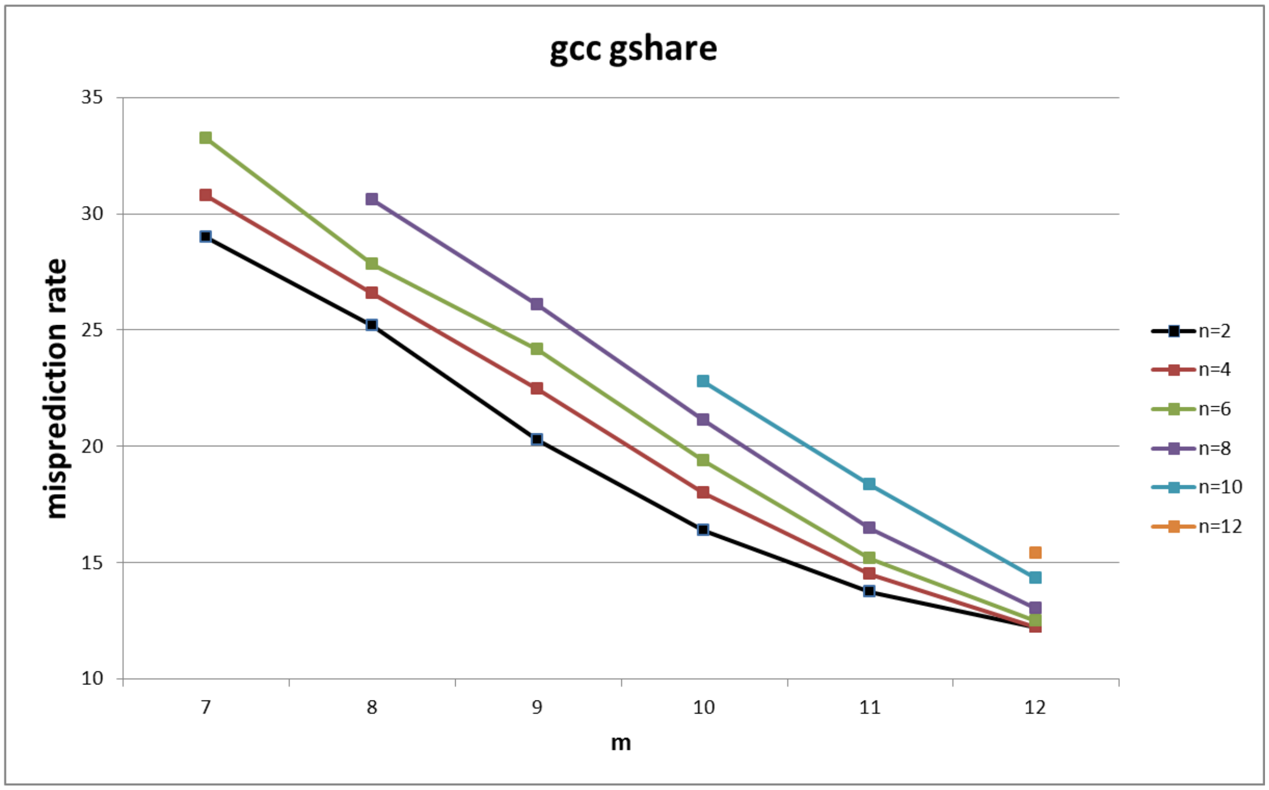

Fig 4

4.2 gshare - jpeg trace

Here we simulate the gshare predictor with different values of “m” ranging from 7 to 12.
And for each such value we take even values of n from 0 to m.
We obtain the following results.

m	n=2	n=4	n=6	n=8	n=10	n=12
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 %

Using the above values we plot the graph of misprediction rate VS the values of n for each m.

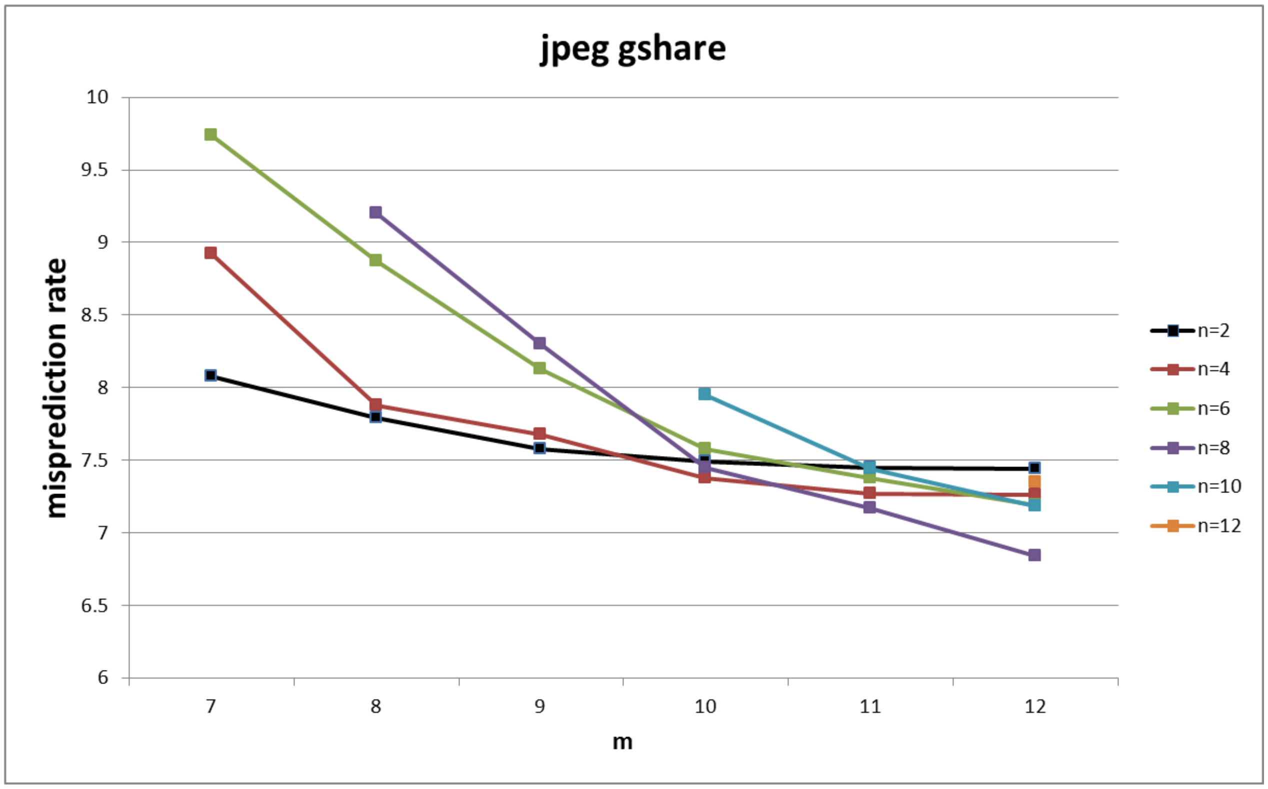

Fig 5

4.3 gshare - perl trace

Here we simulate the gshare predictor with different values of "m" ranging from 7 to 12.
And for each such value we take even values of n from 0 to m.
We obtain the following results.

m	n=2	n=4	n=6	n=8	n=10	n=12
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 %

Using the above values we plot the graph of misprediction rate VS the values of n for each m.

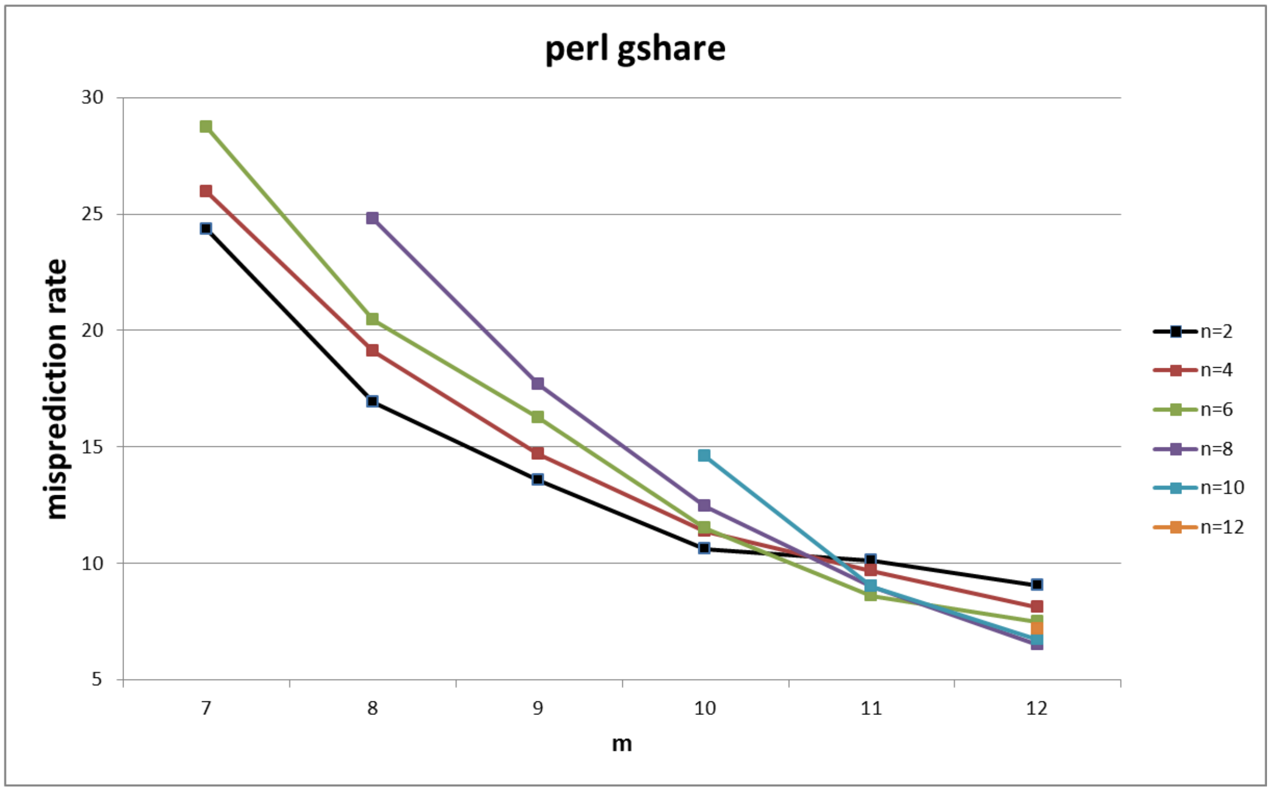

Fig 6

4.4 Comparison and analysis

From sections 4.1, 4.2 and 4.3 we observe the following:

- The jpeg trace has the least misprediction rate among all the tracefiles when we consider overall values otherwise, considering the absolute values; we find that the least value is for the perl trace.
- This indicates that jpeg trace is the easiest to predict and it has predictable patterns as compared to gcc and perl traces.
- In all the graphs, the misprediction rate tends to decrease as the value of m increases, this is due to the fact that as m increases, the size of the prediction table increases thereby allowing more entries in it.
- This results in better recognitions of more patterns thereby reducing the misprediction rate.
- However, we observe that the misprediction rate tends to be lowest with value of n = 8 whenever m is greater than or equal 10 and then increases for higher values of n.
- This shows us that as the prediction table becomes more and more larger, more values are stored in it and more patterns are identified thereby reducing misprediction rate overall.
- But side by side, the global history register has to also store the cumulative history of all the branches.
- Thus there is a tradeoff, as the value of m increases, for increasing value of n, mispredictions increase only till a certain point which is n=8 for our simulator. Thereafter misprediction rate increases.

4.5 Optimum Design

Now, the optimum design for the gshare predictor would be,

It requires $2 \times 2^m + n$ bits for storage. Now maximum allowable storage is given as 16 KB. To minimize the misprediction rate, we can intuitively predict that the least rate would be for the predictor with largest m and optimal n.

Thus, for gcc trace, for all traces, we can choose max m and min n = 8, (assuming that we still put the constraint for n to be only even)

$$16KB = (2 \times 2^m) + 8$$

$$2^4 \times 2^{10} \times 2^3 = 2 \times 2^m + 8$$

$$2^m \approx 65535$$

$$m = 16$$

So the best design for all traces is for m = 16 and n =8.

I simulated the gshare predictor for the value of m = 16 and n = 8 and got following results.

file	**misprediction rate**
gcc_trace	7.57 %
jpeg_trace	6.65 %
perl_trace	4.12 %

5 Conclusions

From the above experiments and simulations, we observe that the branches exhibit a predictable manner of occurrence. We can predict almost any type of branch patterns with an accuracy of close to 90 % since we are using dynamic adaptive branch predictors. We also observe that as we increase the size and complexity of hardware and storage for the prediction tables, the misprediction rate decreases. However there is a certain limit to which this is true after which there are no further improvements in the prediction rate. Thus there is a kind of tradeoff between the storage space and complexity versus the accuracy of predictions.