Experiment Project 2

Introduction

In this part of the project, we are supposed to find the best accuracy of predictors for different workload. To find the optimal configuration, we have to use the tricks the same way we did for experiment 1.

Process

TO start, I choose the basic configuration provided by the constructor. The accuracy that I got was about 79% which petty high for the >1Kb bit budget (10 + 2^ 11 = 2058 bits). So, second thing was to use this and see if increasing the bit budget for GShare improves the accuracy or not. After careful calculation I realize that I can only increase the GShare bit budget to max of 4KB(it goes little over if I went 8KB) which is G= 14, and it improve the accuracy greatly. The average accuracy that I got was about 93%.

```
SIMULATION CONFIGURATION
G:          14
P:          10
N:          5
Predictor:  Gshare
SETUP COMPLETE - STARTING SIMULATION

SIMULATION OUTPUT
Total Instructions:                48574320
Total Branch Instructions:         10000000
Branches Correctly Predicted:      8940210
Branches Miss Predicted:           1059790
Misses Per Kilo Instructions:      21
Number of Tag conflicts            1593192
Fraction Branch Instructions:      0.20587010
Branch Prediction accuracy:        0.89402100
Total Stages in the Pipeline:      5
Branch Miss Prediction Stalls:     2
Average CPI:                       1.04363581
```

```
SIMULATION CONFIGURATION
G:          10
P:          10
N:          5
Predictor:  Gshare
SETUP COMPLETE - STARTING SIMULATION

SIMULATION OUTPUT
Total Instructions:                48574320
Total Branch Instructions:         10000000
Branches Correctly Predicted:      7512226
Branches Miss Predicted:           2487774
Misses Per Kilo Instructions:      51
Number of Tag conflicts            4827359
Fraction Branch Instructions:      0.20587010
Branch Prediction accuracy:        0.75122260
Total Stages in the Pipeline:      5
Branch Miss Prediction Stalls:     2
Average CPI:                       1.10243166
```

Then I tried similar pattern for yeh pat predictor. So, I tried the biggest possible configuration available to us. G = 12 and P= 12 (bits = (P*2^G) +(2^P+1)), which adds up to total of 57344 bits or 7KB which improved accuracy litter bit. On average I got 94% as my accuracy for the workloads. Which is not so much, considering the amount of bits that we are using.

```
SIMULATION CONFIGURATION
G:          12
P:          12
N:          5
Predictor:  Yeh-Patt
SETUP COMPLETE - STARTING SIMULATION

SIMULATION OUTPUT
Total Instructions:                97307048
Total Branch Instructions:         10000000
Branches Correctly Predicted:      9211433
Branches Miss Predicted:           788567
Misses Per Kilo Instructions:      8
Number of Tag conflicts            177077
Fraction Branch Instructions:      0.10276748
Branch Prediction accuracy:        0.92114330
Total Stages in the Pipeline:      5
Branch Miss Prediction Stalls:     2
Average CPI:                       1.01620781
```

```
SIMULATION CONFIGURATION
G:          10
P:          10
N:          5
Predictor:  Yeh-Patt
SETUP COMPLETE - STARTING SIMULATION

SIMULATION OUTPUT
Total Instructions:                97307048
Total Branch Instructions:         10000000
Branches Correctly Predicted:      8957023
Branches Miss Predicted:           1042977
Misses Per Kilo Instructions:      10
Number of Tag conflicts            717217
Fraction Branch Instructions:      0.10276748
Branch Prediction accuracy:        0.89570230
Total Stages in the Pipeline:      5
Branch Miss Prediction Stalls:     2
Average CPI:                       1.02143682
```

Similarly, I tried maximum capacity configuration for perceptron, G=59 P=13 (total bits = 7*2^13 + GHR) which is approximately 7 kb. This is the maximum size available in all three predictors. The average accuracy that I got form these huge weights are about 97% which is way higher than the yeh pat. On similar note, perceptron works way better than any predictor on basic settings.

```
SIMULATION CONFIGURATION
G:          59
P:          13
N:          5
Predictor:  PERCEPTRON
SETUP COMPLETE - STARTING SIMULATION

SIMULATION OUTPUT
Total Instructions:                48854371
Total Branch Instructions:         10000000
Branches Correctly Predicted:      9851438
Branches Miss Predicted:           148562
Misses Per Kilo Instructions:      3
Number of Tag conflicts            485072
Fraction Branch Instructions:      0.20468998
Branch Prediction accuracy:        0.98514380
Total Stages in the Pipeline:      5
Branch Miss Prediction Stalls:     2
Average CPI:                       1.00608183
```

```
SIMULATION CONFIGURATION
G:          10
P:          10
N:          5
Predictor:  PERCEPTRON
SETUP COMPLETE - STARTING SIMULATION

SIMULATION OUTPUT
Total Instructions:                48854371
Total Branch Instructions:         10000000
Branches Correctly Predicted:      9654388
Branches Miss Predicted:           345612
Misses Per Kilo Instructions:      7
Number of Tag conflicts            2564941
Fraction Branch Instructions:      0.20468998
Branch Prediction accuracy:        0.96543880
Total Stages in the Pipeline:      5
Branch Miss Prediction Stalls:     2
Average CPI:                       1.01414866
```

Results

As suspected, bigger the predictor better its performance. Also. The perceptron performance way better than other two algorithm in a small budget. So, I choose this configuration for the given workloads. Since we have 8KB of bit budget.

G = 59 , P=13, N=5, predictor = perceptron