Computer architecture - HOMEWORK 4 - ECGR-5181

Observed and Documented by: Sumukh Raghuram Bhat – 801131997

Problem 1:

Branch prediction is the process of guessing the next instruction to fetch. It is a type of dynamic info fetch and the prediction is based on dynamic changes in the branch behavior.

The misprediction rates for the different branch predictors are tabled:

Branch Predictor type	Misprediction Rate
One Level Branch Predictor	15.91%
Two Level Branch Predictor	22.52%
Two Level Gshare Branch Predictor	21.58%
Two Level Local Branch Predictor	23.09%

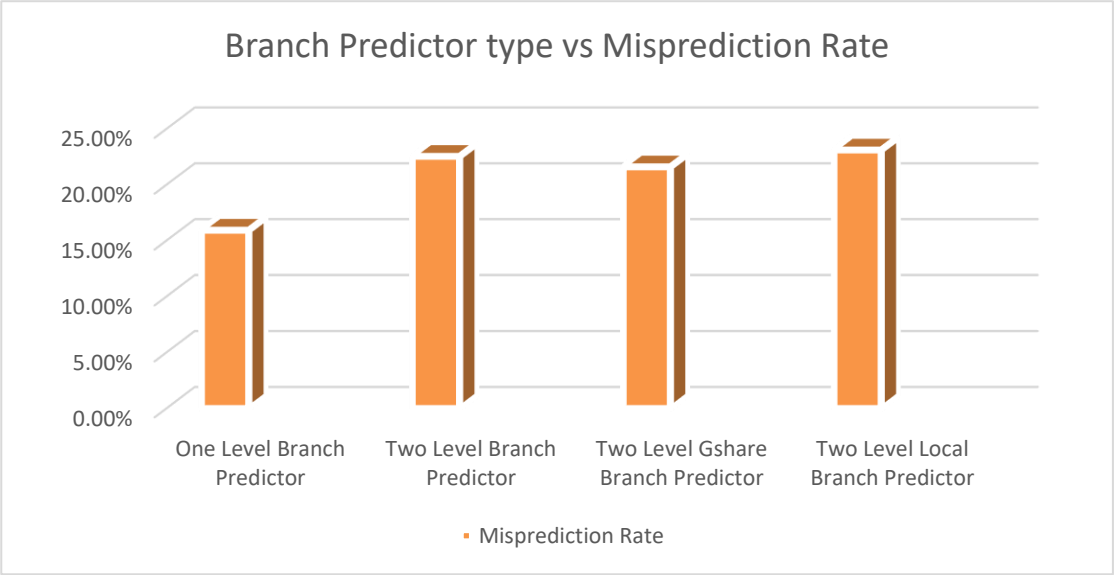
In One-Level Branch Predictor, when a branch is evaluated, the corresponding state machine is updated. Branches evaluated as not taken change the state toward strongly not taken, and branches evaluated as taken change the state toward strongly taken.

The Two-Level Branch Predictor, also referred to as Correlation-Based Branch Predictor, uses a two-dimensional table of counters, also called "Pattern History Table".

A local branch predictor has a separate history buffer for each conditional jump instruction. It may use a two-level adaptive predictor. The history buffer is separate for each conditional jump instruction, while the pattern history table may be separate as well or it may be shared between all conditional jumps.

A global branch predictor does not keep a separate history record for each conditional jump. Instead it keeps a shared history of all conditional jumps. The advantage of a shared history is that any correlation between different conditional jumps is part of making the predictions. The disadvantage is that the history is diluted by irrelevant information if the different conditional jumps are uncorrelated, and that the history buffer may not include any bits from the same branch if there are many other branches in between.

The bar chart for it as follows:



From the simulation, we have:

For the one level branch predictor, the program counter is stored in the Pattern History Table. This one level branch predictor gave us a misprediction rate of 15.94%.

For the two-level branch predictor, the history of the last 10 branches is used for predicting the current branch. We get a misprediction rate of 22.56%.

For two level Gshare branch predictor, the history of last 10 branches and the PC are XOR'ed in this predictor. The resulting pattern is used to refer to the history table. By this, we can prevent duplication. We get a misprediction rate of 21.52%.

For the two-level local branch predictor, a table is used for local branch histories. We get a misprediction rate of 23.11%.

Observation: One Level Branch Prediction gives the best performance followed by the Gshare Branch Predictor.

The Two-Level Branch Predictor, also referred to as Correlation-Based Branch Predictor, uses a two-dimensional table of counters, also called "Pattern History Table". The table entries are two-bit counters. The Two-Level Global Branch Predictor gives bad performance because conflicts in the pattern are possible.

The Two Level Gshare Predictor gives a slightly better performance compared to the Global Branch Predictor.

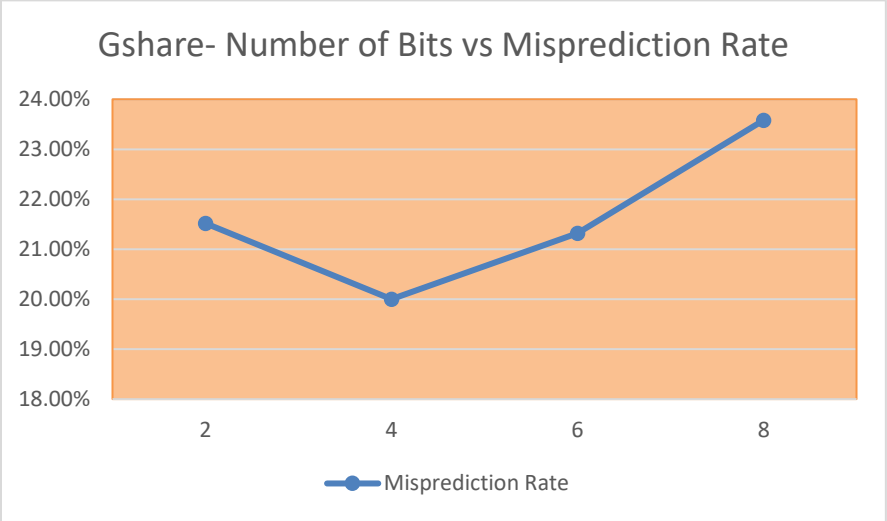
The Two-Level Local Branch Predictor also sees a comparatively poor performance as conflicts are possible.

Problem 2:

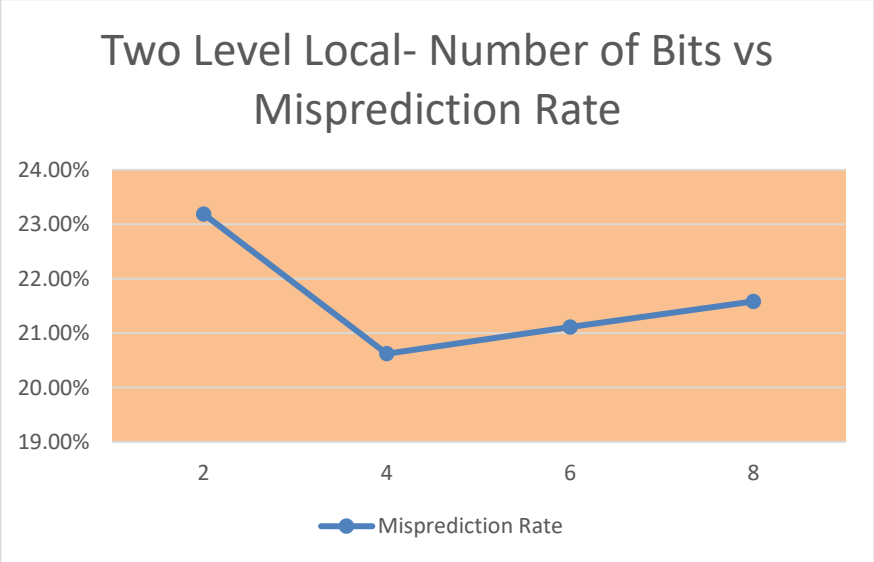
By changing the counter size to 2 bits, 4 bits, 6 bits and 8 bits we get the following outputs and graphs are plotted:

For Gshare Branch Predictor:

Number of Bits	Misprediction Rate
2	21.52%
4	20.00%
6	21.32%
8	23.58%



For Two Level Local Branch Prediction:



The misprediction rates for the different branch predictors are tabled:

Number of Bits	Misprediction Rate
2	23.19%
4	20.62%
6	21.11%
8	21.58%

Observation: From the graphs, we can see that the increase in counter size changes the misprediction rates. The misprediction reduces from 2 bits to 4 bits and increases up till 8 bits. While increasing the number of bits, the number of intermediate states will also increase between taken and not taken. This will increase misprediction like how it is in the trace file.

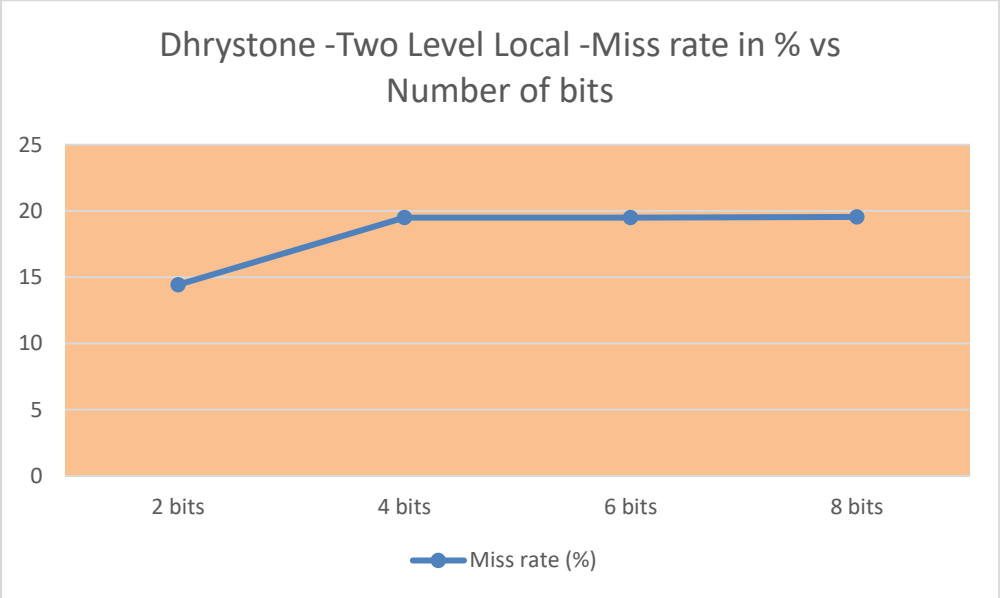
Hence having it 4 bits is the best for this branch predictor.

Problem 3:

Two trace files were generated, one for Linpack benchmark and the other for Dhrystone benchmark.

The trace file for Dhrystone was linked to the Two-Level Local Branch Predictor, we observe the following:

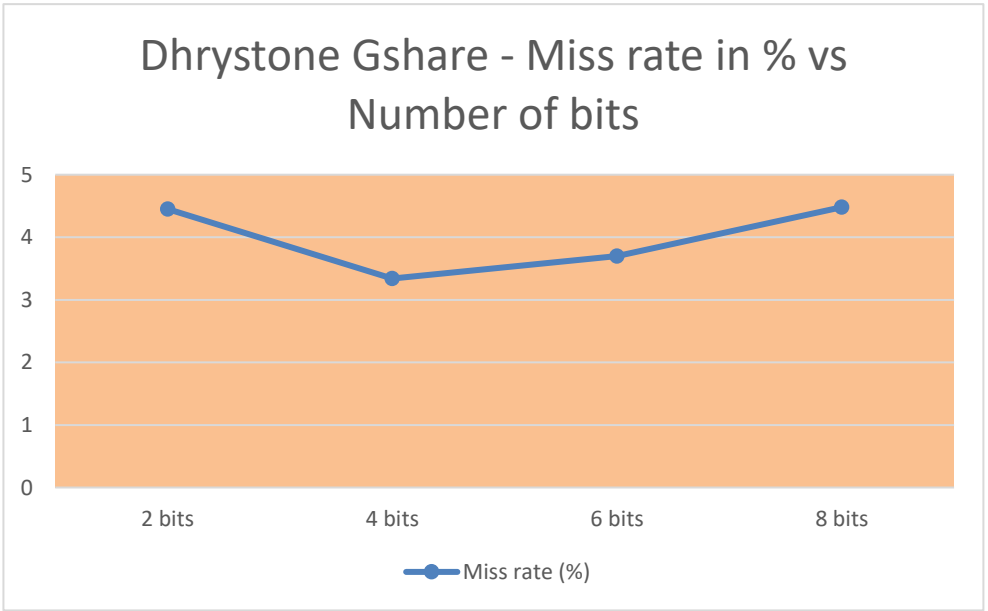
Number of bits	Mispredictions	Miss rate (%)
2 bits	114795	14.43
4 bits	152334	19.50
6 bits	152485	19.51
8 bits	152692	19.55



We can see the miss rate shoots up for 4 bits when compared to 2 bits. The miss rate increases between 4 bits, 6 bits and 8 bits is very minute and can be neglected.

When the trace file for Dhrystone is linked to the Gshare Predictor, we observe the following:

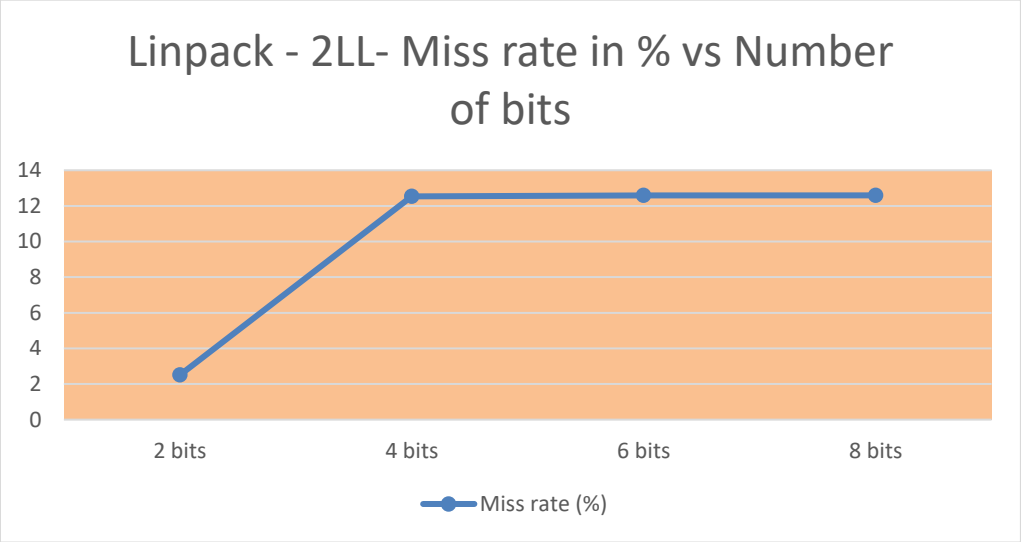
Number of bits	Mispredictions	Miss rate (%)
2 bits	34891	4.45
4 bits	26231	3.34
6 bits	30167	3.70
8 bits	34289	4.48



The miss rate reduces from 2 bits to 4 bits and increases up till 8 bits. The miss rate is the best for 4 bits.

The trace file for Linpack was linked to the Two-Level Local Branch Predictor, we observe the following:

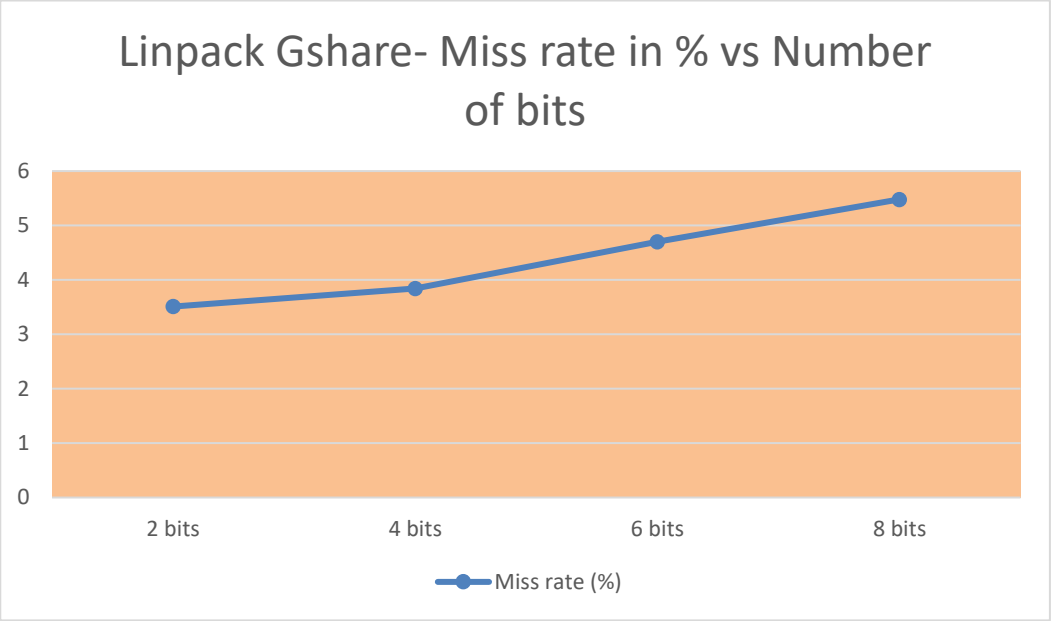
Number of bits	Mispredictions	Miss rate (%)
2 bits	43765	2.50
4 bits	214375	12.54
6 bits	214894	12.58
8 bits	215097	12.59



There is a significant increase in the miss rate from 2 bits to 4 bits. The miss rate almost becomes constant when we increase the number of bits henceforth.

When the trace file for Linpack is linked to the Gshare Predictor, we observe the following:

Number of bits	Mispredictions	Miss rate (%)
2 bits	60836	3.51
4 bits	65940	3.84
6 bits	81801	4.70
8 bits	98122	5.48



There is a steady increase in the miss rate when we increase the number of bits. We can infer that the miss rate is the best when the number of bits is 2. Highest correction rate is 87.76 for 4 bits with 1024 table entries. The graph is plotted below.

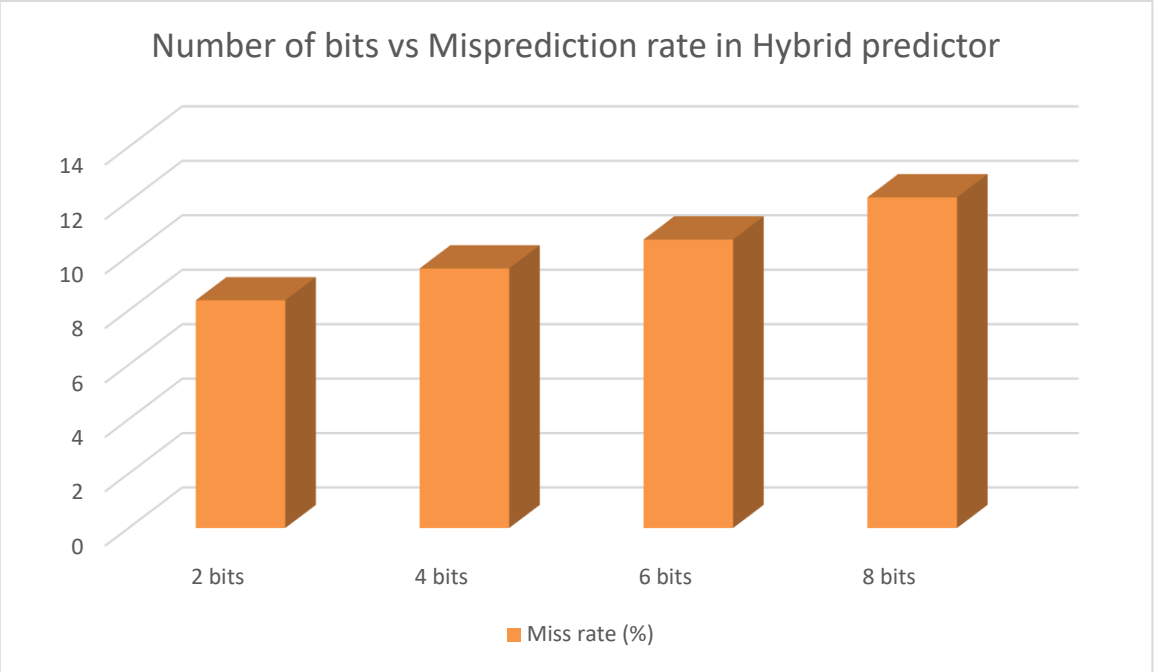
Problem 4:

The hybrid branch predictor is created using multiple type of predictors making sure to have a better prediction rate. It is also called combined predictor as it implements more than one prediction mechanism. The final prediction is based either on a meta-predictor that remembers which of the predictors has made the best predictions in the past, or a majority vote function based on an odd number of different predictors.

Predictors like Gshare use multiple table entries to track the behavior of any particular branch. This multiplication of entries makes it much more likely that two branches will map to the same table entry (a situation called aliasing), which in turn makes it much more likely that prediction accuracy will suffer for those branches. Once you have multiple predictors, it is beneficial to arrange that each predictor will have different aliasing patterns, so that it is more likely that at least one predictor will have no aliasing.

The results are tabulated below:

Number of bits	**Mispredictions**	**Miss rate (%)**	**Correct Prediction rate (%)**
2 bits	1374238	8.37	91.62
4 bits	1566229	9.54	90.45
6 bits	1740850	10.60	89.39
8 bits	1995131	12.15	87.84



Observation: It can be seen that in this case too, the number of bits in the bit counter if increased, increases the misprediction rate.

Hence in the Hybrid Predictor, by using more than one type of predictor (i.e., multiple algorithms), the best prediction can be selected.
The advantages of such a type of prediction is that it provides a better accuracy and also it reduces the warm-up time but has a longer access latency when compared to other type of predictors.