Branch Prediction Analysis Report

Group Members:

Jyoti	2022CSB1319
Ojas Jain	2022CSB1099
Siddharth Verma	2022CSB1126

Theory:

Branch prediction methods are crucial in modern computer architectures to improve performance by speculatively executing instructions. In RISC-V architecture, several branch prediction methods are commonly employed:

1. **Always Taken:** This method assumes that all branches are always taken. It is a simple strategy where the processor always predicts that a branch will be taken regardless of its outcome.
2. **Always Not Taken:** Conversely, this method predicts that branches are never taken. It's a straightforward approach where the processor always assumes that the branch will continue to the next sequential instruction.
3. **1-Bit Predictor:** This predictor maintains a single-bit saturating counter for each branch. The counter is in our case initially predicting "not taken" (0). If the branch is taken, the counter is incremented; if it's not taken, the counter is decremented. When the counter saturates, meaning it reaches its maximum value, further increments or decrements don't change its state.
4. **2-Bit Predictor:** This method improves upon the 1-bit predictor by using a two-bit saturating counter. The counter can be in one of four states, typically represented by the values 00, 01, 10, and 11. The counter transitions between these states based on the outcome of branch predictions. We have initially assumed the state to be ‘weakly taken’ or 01 state. The MSB denotes whether we are predicting a branch or not where 0 is not taken and 1 is taken.

Analysis:

BubbleSort Code Prediction Analysis:

1. Always Taken Predictor: 51.8149%
2. Always Not Taken Predictor: 48.1851%
3. 1-Bit Predictor: 96.1011%
4. 2-Bit Predictor: 97.1183%

Factorial Code Prediction Analysis:

1. Always Taken Predictor:65.506%
2. Always Not Taken Predictor: 34.494%
3. 1-Bit Predictor: 87.7293%
4. 2-Bit Predictor: 89.6991%

SquareRoot Code Prediction Analysis:

1. Always Taken Predictor:64.0606%
2. Always Not Taken Predictor: 35.9394%
3. 1-Bit Predictor: 94.9849%
4. 2-Bit Predictor: 95.8192%

Inferences:

Always Taken Predictor

By looking at the accuracy of the always taken predictor across the 3 codes,it averages out to about 60% from which we can infer that the ‘taken’ statements generally outweigh the ‘not taken’ statements.This discrepancy in the uniformity is likely due to the way loops are written in these programs ie the loop iterates when the branch instruction is taken and terminates when it is not taken.

Always Not Taken Predictor

By looking at the accuracy of the always not taken predictor across the 3 codes,it averages out to about 40% from which we can infer that the ‘taken’ statements generally outweigh the ‘not taken’ statements which ideally in a large set of inputs should be about 50/50..This discrepancy has been discussed in the above section.

1 bit Predictor

By looking at the accuracy of the 1 bit predictor across the 3 codes,it averages out to about 92-93%.The reason for this may be that a general large code consisting of branch instructions will have large amounts of periods where consecutive ‘taken’ or ‘not taken’ statements are coming and the 1 bit predictor handles this perfectly.One problem with the 1 bit predictor is its inability to handle alternatively spread ‘taken’ and ‘not taken’ where its accuracy would be 0%.

2 bit Predictor

By looking at the accuracy of the 2 bit predictor across the 3 codes,it averages out to about 94-95%.This also handles the case of consecutive ‘taken’ or ‘not taken’ statements perfectly just like 1 bit predictor.The improvement upon the 1 bit predictor is in the case of alternatively spread ‘taken’ and ‘not taken’ where its accuracy is 50%.Also it will handle random sequences of ‘taken’ and ‘not taken’ statements generally better than the 1 bit predictor.