Problem 1
Design and describe (all the decisions have to be motivated) a 1-BHT and a 2-BHT able to execute the following assembly code.

```
          ADDI R1 R1 100
LOOP2:    MULTD F2 F2 F1
          ADDD F1 F3 F2
          SUBI R1 R1 1
          BNEZ R1 LOOP2
```

The obtained result, in terms of miss predictions, is inline with theoretical characteristics of the two predictors? Please effectively support your answer.

Answer

The branch history table's primary design parameter are the size (2^k) and number of prediction bits (b).
The impact of the table size here cannot be evaluated, since we only have a single branch. However, it is limited to imposing a misprediction on the first branch.
The table is indexed with the lower k bits of the branch instruction address (PC) or a k bit hash (more effective, but more costly).
Assuming R1 is originally 0, then the BNEZ instruction is executed 100 times, and the branch is taken always, except the last time.
Consequently, with the 1 bit predictor, the prediction is successful every time, except the first and last (98% success or 2% misprediction).
The 2-bit predictor does not fail on the first branch, but fails on the last (99% success or 1% misprediction).

Problem 3
Design and describe (all the decisions have to be motivated) a 1-BHT and a 2-BHT able to execute the following assembly code.
(R0 is set to 1, R1 is set to 0)

```
LOOP:   LD F3 0 R0
        ADDD F1 F3 F3
        MULTD F2 F3 F1
        ADDI R1 R1 1000
LOOP2:  MULTD F2 F2 F3
        SUBI R1 R1 1
        BNEZ R1 LOOP2
        SUBI R0 R0 1
        BNE R0 R1 LOOP
```

The obtained result, in terms of miss predictions, is inline with theoretical characteristics of the two predictors?
If no, please, change the value of R0 and R1 to make it. Please effectively support your answer.

Answer

The branch history table's primary design parameter are the size (2^k) and number of prediction bits (b). The impact of the table size is limited to imposing a misprediction on the first branch in case the two instruction collide.
The table is indexed with the lower k bits of the branch instruction address (PC) or a k-bit hash (more effective, but more costly).
Here, assuming a fixed length 32-bit ISA, the two instructions collide if k=1 if no hash is used. With any choice of k>1, the addresses do not collide (although they could collide with other branches in the rest of the code).

The BNEZ instruction is executed 1000 times, and the branch is taken always, except the last time.
The BNE instruction is executed only once, and the branch is not taken.

Consequently, with the 1-bit predictor, and assuming misprediction when no history is available, the prediction is successful every time, except the first and last (99.8% success or 0.2% misprediction) for the inner loop, and fails for the outer loop (100% misprediction).
Modifying the initial value of R0 to run the outer loop multiple times brings the prediction in line with the theoretical characteristics. To get the expected <18% misprediction, we need to run the loop at least 12 times.
However, considering the overall success rate, we have only 3 mispredictions on 1001 branches, which is well within the usual rates.

The 2-bit predictor does not fail on the first branch, but fails on the last (99.9% success or 0.1% misprediction), when considering the innermost loop only.
For the outer loop, the prediction fails since the branch is not taken at the first time. Once more, we need at least 12 outer iterations to fall into the usual range of prediction for the specific branch.