10 Branch Prediction [60 points]

A processor implements an *in-order* pipeline with *15 stages*. Each stage completes in a single cycle. The pipeline stalls on a conditional branch instruction until the condition of the branch is evaluated. However, you *do not* know at which stage the branch condition is evaluated. Please answer the following questions.

(a) [10 points] A program with 2500 dynamic instructions completes in 4514 cycles. If 500 of those instructions are conditional branches, at the end of which pipeline stage are the branch instructions resolved? (Assume that the pipeline does not stall for any other reason than conditional branches, e.g., data dependencies, during the execution of that program.)

At the end of the 5th stage.

Explanation: $Total\ cycles = 15 + 2500 + 500 * X - 1$
$4514 = 2514 + 500 * X$
$2000 = 500 * X$
$X = 4$
Each branch causes 4 idle cycles (bubbles), thus branches are resolved at the end of 5th stage.

(b) [2+3 points] In a new, higher-performance version of the previous processor, the architects implement a *mysterious* branch prediction mechanism to improve the performance of the processor. They keep the rest of the design exactly the same as before. The new design with the mysterious branch predictor completes the execution of the following piece of code in 136 cycles.

Please note that the number of pipeline stages and the stage at which the branch condition is evaluated are same as the previous question. Also, assume that the pipeline never stalls due to any other reasons than conditional branches.

```
MOV R1, #0 // R1 = 0

LOOP_1:
    BEQ R1, #5, LAST // Branch to LAST if R1 == 5
    ADD R1, R1, #1   // R1 = R1 + 1
    MOV R2, #0       // R2 = 0
LOOP_2:
    BEQ R2, #5, LOOP_1 // Branch to LOOP_1 if R2 == 5.
    ADD R2, R2, #1     // R2 = R2 + 1
    B LOOP_2           // Unconditional branch to LOOP_2

LAST:
    MOV R1, #1         // R1 = 0
```

How many instructions will be executed when running this piece of code? Show your work.

Total instructions executed = 98;

How many of them are CONDITIONAL branch instructions? Show your work.

Conditional branch instructions = 36;

(c) Based on the given information, determine which of the following branch prediction mechanisms could be the *mysterious* branch predictor implemented in the new version of the processor. For each branch prediction mechanism below, you should circle the configuration parameters that makes it match the performance of the mysterious branch predictor.

(I) [10 points] **Static Branch Predictor**

Could this be the mysterious branch predictor?

YES NO

If YES, for which configuration below is the answer *YES*? Pick an option for each configuration parameter.

i. Static Prediction Direction

Always taken Always not taken

Explain clearly to receive points.

YES, if the static prediction direction is *always not taken*.

Explanation: 98 instructions (36 of them are conditional branches) finishes execution in 136 cycles. This means there are 6 branch mispredictions. So, any predictor that produces 6 mispredictions can be our mysterious predictor.
A static predictor with always not taken predicton generates 6 mispredictions. Hence, YES.

(II) [15 points] **Last Time Branch Predictor**

Could this be the mysterious branch predictor?

YES NO

If YES, for which configuration is the answer *YES*? Pick an option for each configuration parameter.

i. Initial Prediction Direction

Taken Not taken

ii. Local for each branch instruction (i.e., PC-based) or global (i.e., shared among all branches) history?

Local Global

Explain clearly to receive points.

NO.

Explanation: There is no configuration for this branch predictor that results in 6 mispredictions for the above program.

Local-taken: 12 mispredictions, Local-NotTaken: 10 mispredictions,
Global-taken: 10 mispredictions, Global-NotTaken: 9 mispredictions.

(III) [5 points] **Backward taken, Forward not taken (BTFN)**

Please recollect, a conditional branch is said to be backward if its target address is lower than the branch PC, and vice-versa.

Could this be the mysterious branch predictor?

YES NO

Explain clearly to receive points.

NO.

Explanation: BTFN predictor makes 26 mispredictions. Hence it cannot be our mysterious branch predictor.

(IV) [15 points] **Two-bit Counter Based Prediction** (using saturating arithmetic)

Could this be the mysterious branch predictor?

YES NO

If YES, for which configuration is the answer *YES*? Pick an option for each configuration parameter.

i. Initial Prediction Direction

`00 (Strongly not taken)` `01 (Weakly not taken)`
`10 (Weakly taken)` `11 (Strongly taken)`

ii. Local for each branch instruction (i.e., PC-based, without any interference between different branches) or global (i.e., a single counter shared among all branches) history?

Local Global

Explain clearly to receive points.

YES, if *local* or *global* history registers with *00* or *01* initial values are used.

Explanation: Such a configuration yields exactly 6 mispredictions, which results in 136 cycles execution time for the above program.