5 Branch Prediction [70 points]

A processor implements an *in-order* pipeline with *12 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) [15 points] A program with 1000 dynamic instructions completes in 2211 cycles. If 200 of those instructions are conditional branches, at the end of which pipeline stage the branch instructions are resolved? (Assume that the pipeline does not stall for any other reason than the conditional branches (e.g., data dependencies) during the execution of that program.)

At the end of the 7th stage.

Explanation: $Total\ cycles = 12 + 1000 + 200 * X - 1$
$2211 = 1011 + 200 * X$
$1400 = 200 * X$
$X = 6$
Each branch causes 6 idle cycles (bubbles), thus branches are resolved at the end of 7th stage.

(b) In a new, higher-performance version of the 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 code in 115 cycles.

```
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, #3, LOOP_1 // Branch to LOOP_1 if R2==3.
    ADD R2, R2, #1     // R2 = R2 + 1
    B LOOP_2           // Unconditional branch to LOOP_2

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

Assume that the pipeline never stalls due to a data dependency. 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) [15 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:

> *YES*, if the static prediction direction is *always not taken*.
>
> **Explanation:** Such a predictor makes 6 mispredictions, which is the number resulting in 115 cycles execution time for the above program.

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 (PC-based) or global (shared among all branches) history?

Local Global

Explain:

> *NO*.
>
> **Explanation:** There is not a configuration for this branch predictor that results in 6 mispredictions for the above program.

iii) [10 points] **Backward taken, Forward not taken (BTFN)**

Could this be the mysterious branch predictor?

YES NO

Explain:

> *NO*.
>
> **Explanation:** BTFN predictor does not make exactly 6 mispredictions for the above program.

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:

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

Explanation: Such a configuration yields 6 mispredictions, which results in 115 cycles execution time for the above program.