Noel Ball
HW3

Question 1:
A.)

Branch	Taken?	a_0	a_1	a_2
B1	Yes	1	0	
B1	No	2	1	
B2	Yes			2
B3	Yes	3		
B1	Yes	5	1	
B2	No	6		0
B3	Yes	9		
B1	Yes	9	1	
B1	No	10	0	
B2	No	10		0
B3	Yes	13		
B1	Yes	13	1	
B1	No	14	0	
B2	Yes	15		2
B3	Yes	16		
B1	Yes	17	1	
B1	No	18	0	
B2	No	18		0
B3	No	19		

B.)

Branch	Prediction	BHT After
B1	No	1
B1	Yes	0
B2	No	1
B3	No	1
B1	No	1
B2	Yes	0
B3	Yes	1
B1	No	1
B1	Yes	0
B2	No	0
B3	Yes	1
B1	No	1
B1	Yes	0
B2	No	1
B3	Yes	1
B1	No	1
B1	Yes	0
B2	No	0
B3	Yes	1

C.)

Branch	Prediction	BHT After
B1	No	01
B1	No	00
B2	No	01
B3	No	01
B1	No	01
B2	No	00
B3	No	10
B1	No	01
B1	No	00
B2	No	01
B3	Yes	11
B1	No	01
B1	No	00
B2	No	00
B3	Yes	11
B1	No	01
B1	No	00
B2	No	01
B3	Yes	11

D.)

Branch	1B BHT	2B BHT
B1	0/18	9/18
B2	1/9	9/9
B3	4/5	4/5
Overall	5/32	22/32

Question 2:

A.) The alternating pattern T NT T NT T NT… and so on, will start with BHT as 0 mispredicting NT. This updates the BHT to 1 mispredicting T when the next branch instruction is not taken. This pattern keeps flipping the predictors bit every cycle leading to a 0% prediction accuracy.

B.) In specific cases, such as T T NT NT repeating, it is possible for a 1B predictor to perform as well as a 2B predictor but never better. 2B predictors are designed to achieve better performance due to forgiving one off deviations from a consistent pattern.

C.) The average penalty per instruction from adding a branch predictor can be calculated as $0.20 \times (1 - p) \times 3$ due to 20% branch instructions, 1 - p incorrect predictions, and a 3-cycle penalty from branching. The overall CPI with the branch predictor becomes $1 + (0.20 \times (1 - p) \times 3) = 1 + 0.60 \times (1 - p) = 1.6 - 0.6p$.
With $Speedup = \frac{Original\ CPI}{Predictor\ CPI} = \frac{1.6}{1.6 - 0.6p}$ from adding a branch predictor and BHB.

Question 3:

A.) Implementing precise exceptions in a consistent location within the pipeline is preferred for many reasons; the program order is preserved when the instructions before the point of exception are fully executed and those after it are not executed, the state recovery is simplified, and the complexity for the processors control logic is reduced

Bi.) For the first code snippet, a two-wide in-order processor might perform better because it can utilize dual execution to handle the loads and stores simultaneously, which is the bulk of the execution.

Bii.) For the second code snippet, a one-wide out-of-order processor is likely more beneficial because it can rearrange instruction execution dynamically to mitigate the long-latency operations and handle the multiple dependencies present.