Computer Architecture HW4 Handwritten part
Name: Leung Ko Tsun(梁高浚) SID: T10902303

4.28 The importance of having a good branch predictor depends on how often conditional branches are executed. Together with branch predictor accuracy, this will determine how much time is spent stalling due to mispredicted branches. In this exercise, assume that the breakdown of dynamic instructions into various instruction categories is as follows:

R-type	beqz/bnez	jal	ld	sd
40%	25%	5%	25%	5%

Also, assume the following branch predictor accuracies:

Always-Taken	Always-Not-Taken	2-Bit
45%	55%	85%

4.28.1 [10] <§4.8> Stall cycles due to mispredicted branches increase the CPI. What is the extra CPI due to mispredicted branches with the always-taken predictor? Assume that branch outcomes are determined in the ID stage and applied in the EX stage that there are no data hazards, and that no delay slots are used.

4.28.1: The extra CPI due to mis-predicted branches with the Always-taken predictor is 0.4125.
If a branch is incorrectly predicted, the following three instructions are flushed: IF, ID, and EX stages. So, 55% of the branches will result in the flushing of three instructions, CPI = 1 + (1 - 0.45)*0.25*3 = 1.4125.

4.28.2 [10] <§4.8> Repeat 4.28.1 for the "always-not-taken" predictor.

4.28.2: The extra CPI is 0.3375.
(1 + 0.25*(1-0.55)) = 1.1125

4.28.3 [10] <§4.8> Repeat 4.28.1 for the 2-bit predictor.

4.28.3: The extra CPI is 0.1125
(1 + 0.25*(1-0.85)) = 1.0375

4.28.4 [10] <§4.8> With the 2-bit predictor, what speedup would be achieved if we could convert half of the branch instructions to some ALU instruction? Assume that correctly and incorrectly predicted instructions have the same chance of being replaced.

The speedup is 1.0184.
If half of the branch instructions are changed, the percentage of instructions that are branches will reduce from 25% to 12.5%. Since predicted and mispredicted branches are replaced equally, the misprediction rate keeps the same at 15%. Therefore, the new CPU is 1 + 0.125(1-0.85) = 1.01875. So the speedup is 1.0375 / 1.01875 = 1.0184

4.28.5 [10] <§4.8> With the 2-bit predictor, what speedup would be achieved if we could convert half of the branch instructions in a way that replaced each branch instruction with two ALU instructions? Assume that correctly and incorrectly predicted instructions have the same chance of being replaced.

The speedup is 0.91.

We can treat two ADD instructions as a branch with an extra cycle. Therefore, half of the branches will have 1 extra cycle, and 15% of the other half will have 1 extra cycle, and the remaining branches have 0 extra cycles. CPI: 1 + 0.5*0.25*1 + 0.5*0.25*0.15*1 = 1.14375 and the speedup is 1.0375/1.14375 = 0.91

4.28.6 [10] <§4.8> Some branch instructions are much more predictable than others. If we know that 80% of all executed branch instructions are easy-to-predict loop-back branches that are always predicted correctly, what is the accuracy of the 2-bit predictor on the remaining 20% of the branch instructions?

The accuracy is 25%. Since 80% of branches are always predicted correctly, and overall accuracy is 0.85, so 0.8*1 + 0.2*x = 0.85, solving x we have x = 0.25.

4.29 This exercise examines the accuracy of various branch predictors for the following repeating pattern (e.g., in a loop) of branch outcomes: T, NT, T, T, NT.

4.29.1 [5] <§4.8> What is the accuracy of always-taken and always-not-taken predictors for this sequence of branch outcomes?

4.29.1
accuracy for always-taken: ⅗ = 60%
accuracy for always-not-takenL ⅖ = 40%

4.29.2 [5] <§4.8> What is the accuracy of the 2-bit predictor for the first four branches in this pattern, assuming that the predictor starts off in the bottom left state from Figure 4.61 (predict not taken)?

4.29.2

Outcomes	Predictor value	correct of incorrect	accuracy
T,NT,T,T	0,1,0,1	I,C,I,I	25%

4.29.3 [10] <§4.8> What is the accuracy of the 2-bit predictor if this pattern is repeated forever?

4.29.3

Outcomes	Predictor value	Correct of incorrect	accuracy
T,NT,T,T,NT	1st occurence: 0,1,0,1,2 2nd: 1,2,1,2,3 3rd: 2,3,2,3,3 4th: 2,3,2,3,3	C,I,C,C,I	60%

4.29.4 [30] <§4.8> Design a predictor that would achieve a perfect accuracy if this pattern is repeated forever. You predictor should be a sequential circuit with one output that provides a prediction (1 for taken, 0 for not taken) and no inputs other than the clock and the control signal that indicates that the instruction is a conditional branch.

4.29.4 The predictor should be an N-bit shift register, here N is the number of branch outcomes in the target pattern. Initially, the shift register should be with the pattern itself(0 for not taken, 1 for taken), and the prediction is always the value in the leftmost bit of the shift register. The register should be shifted after each predicted branch.

4.29.5 [10] <§4.8> What is the accuracy of your predictor from 4.29.4 if it is given a repeating pattern that is the exact opposite of this one?

4.29.5
Accuracy: 0 (Output from predictors is always opposite of the actual outcome of the branch instruction)

4.29.6 [20] <§4.8> Repeat 4.29.4, but now your predictor should be able to eventually (after a warm-up period during which it can make wrong predictions) start perfectly predicting both this pattern and its opposite. Your predictor should have an input that tells it what the real outcome was. Hint: this input lets your predictor determine which of the two repeating patterns it is given.

The predictor is the same as in 4.29.4, except that it should compare its prediction to the actual outcome and use logical NOT to invert all the bits in the shift register if the prediction is incorrect. This predictor can still always predict the given pattern perfectly. As for the opposite pattern, the first prediction will be wrong, so the predictor’s state is inverted and then the predictions are always correct. As for the warm-up period, there is no warm-up period for the given pattern and the warm-up period for the opposite pattern is only 1 branch.