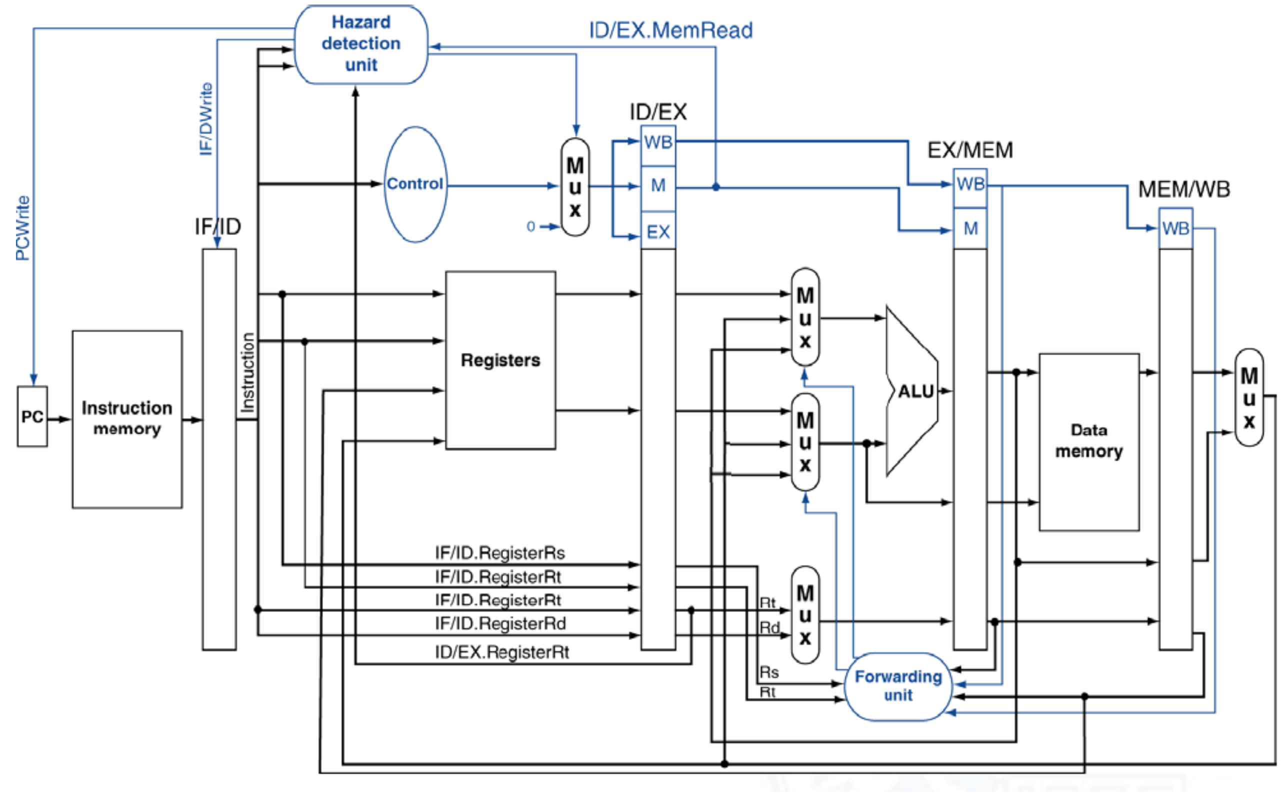
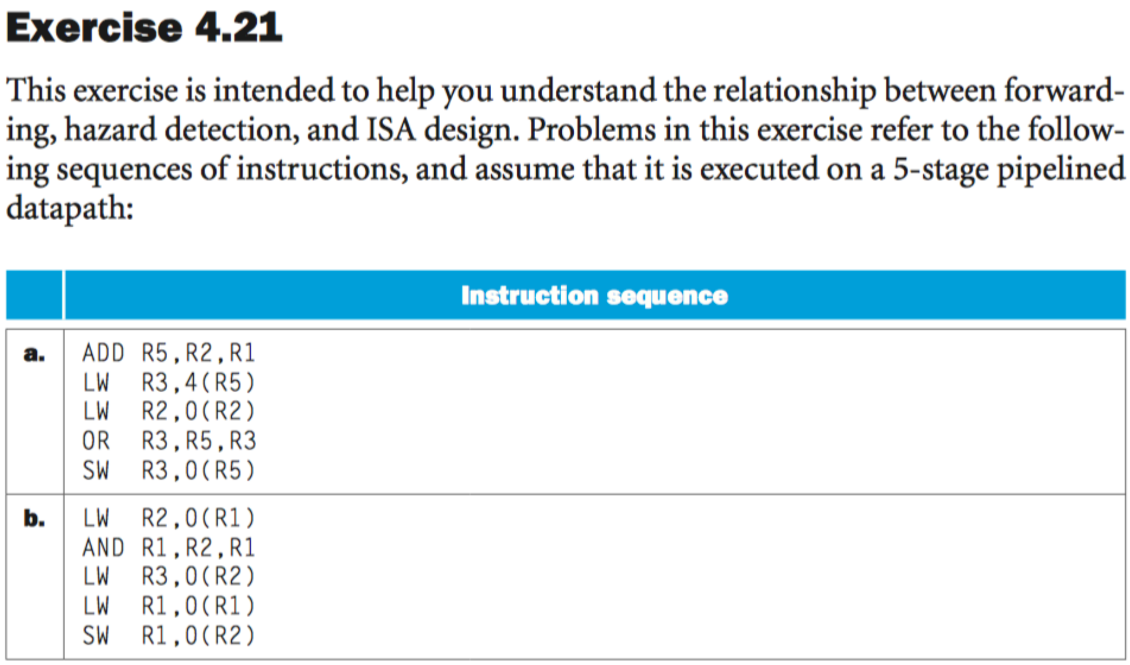
JOINT INSTITUTE
交大密西根学院

Ve370 Introduction to Computer Organization

Homework 6 with problems

1. Exercise 4.21.4 (b), refer to the following figure instead of Figure 4.60. (20 points)
2. Exercise 4.21.5 (b), refer to the following figure instead of Figure 4.60. (10 points)

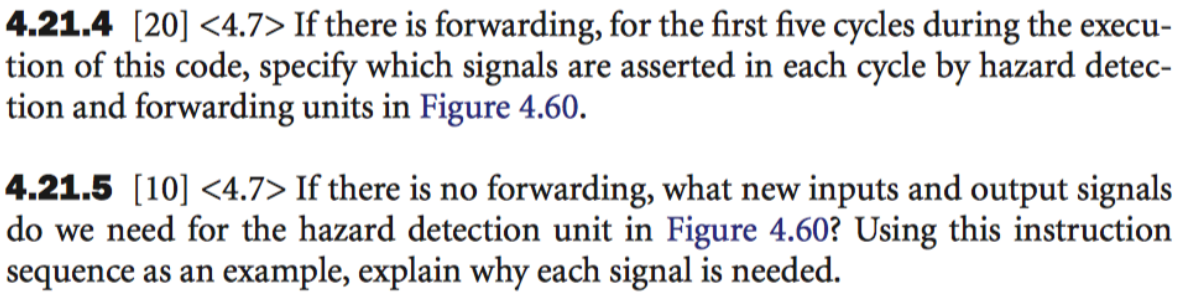




Exercise 4.21

This exercise is intended to help you understand the relationship between forwarding, hazard detection, and ISA design. Problems in this exercise refer to the following sequences of instructions, and assume that it is executed on a 5-stage pipelined datapath:

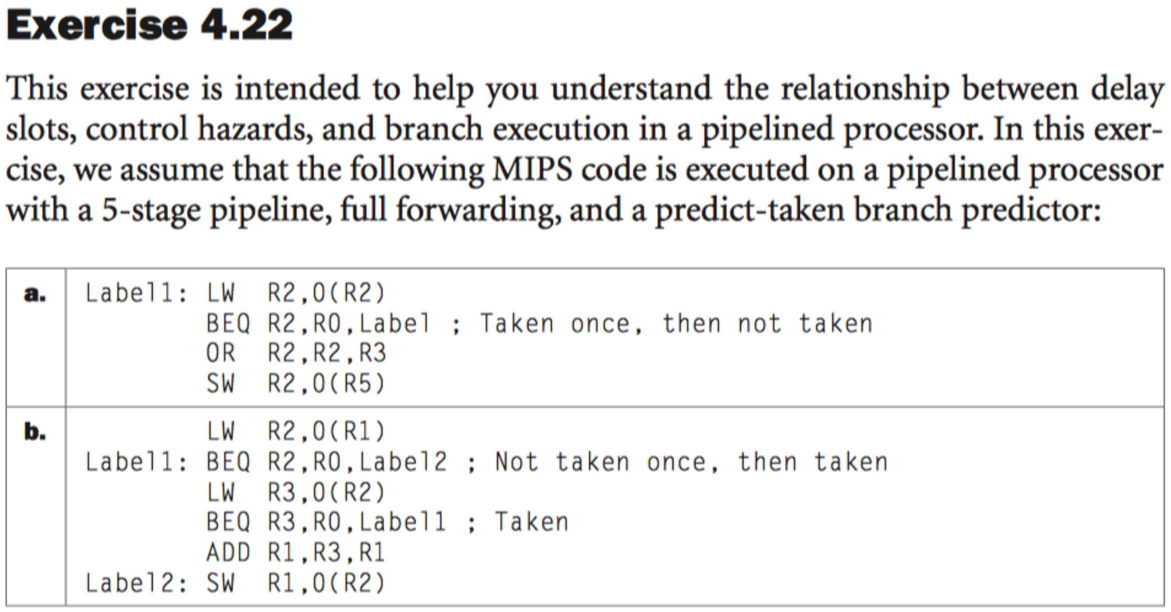
	Instruction sequence
a.	ADD R5,R2,R1 LW R3,4(R5) LW R2,0(R2) OR R3,R5,R3 SW R3,0(R5)
b.	LW R2,0(R1) AND R1,R2,R1 LW R3,0(R2) LW R1,0(R1) SW R1,0(R2)



4.21.4 [20] <4.7> If there is forwarding, for the first five cycles during the execution of this code, specify which signals are asserted in each cycle by hazard detection and forwarding units in Figure 4.60.

4.21.5 [10] <4.7> If there is no forwarding, what new inputs and output signals do we need for the hazard detection unit in Figure 4.60? Using this instruction sequence as an example, explain why each signal is needed.

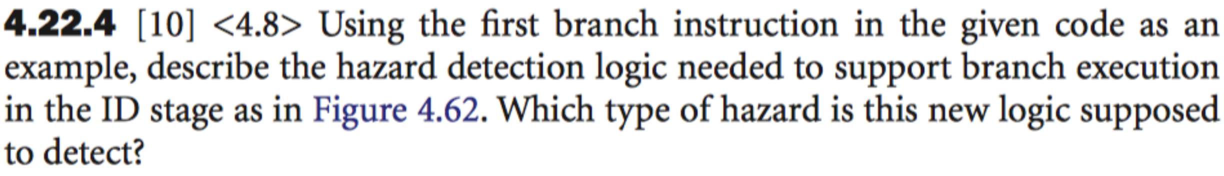
3. Exercise 4.22.4 (b) (10 points)
4. Exercise 4.22.5 (b) (10 points)



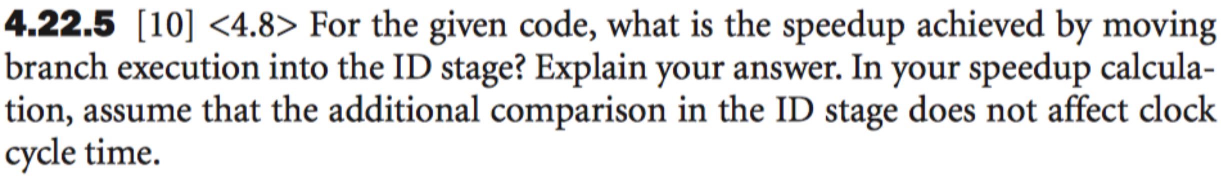
Exercise 4.22

This exercise is intended to help you understand the relationship between delay slots, control hazards, and branch execution in a pipelined processor. In this exercise, we assume that the following MIPS code is executed on a pipelined processor with a 5-stage pipeline, full forwarding, and a predict-taken branch predictor:

a.	Label1: LW R2,0(R2) BEQ R2,R0,Label ; Taken once, then not taken OR R2,R2,R3 SW R2,0(R5)
b.	LW R2,0(R1) Label1: BEQ R2,R0,Label2 ; Not taken once, then taken LW R3,0(R2) BEQ R3,R0,Label1 ; Taken ADD R1,R3,R1 Label2: SW R1,0(R2)

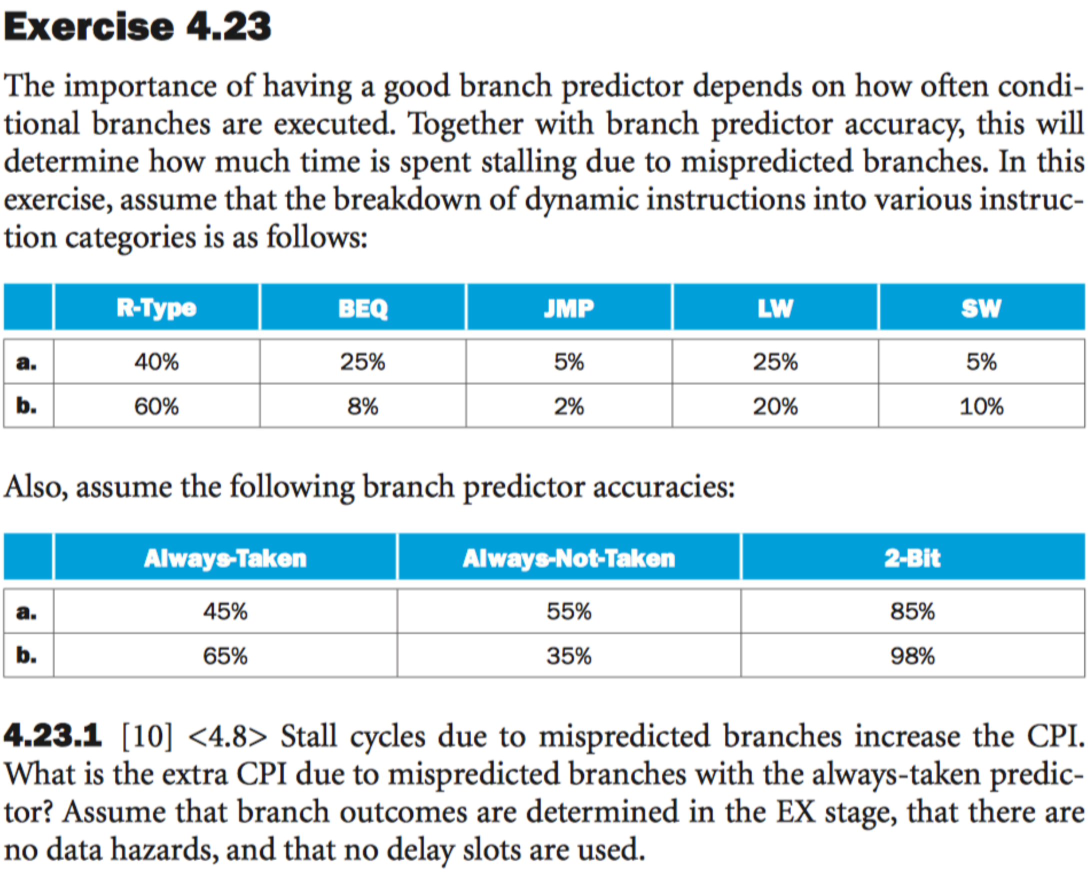


4.22.4 [10] <4.8> Using the first branch instruction in the given code as an example, describe the hazard detection logic needed to support branch execution in the ID stage as in Figure 4.62. Which type of hazard is this new logic supposed to detect?



4.22.5 [10] <4.8> For the given code, what is the speedup achieved by moving branch execution into the ID stage? Explain your answer. In your speedup calculation, assume that the additional comparison in the ID stage does not affect clock cycle time.

5. Exercise 4.23.1 (a) (10 points)



Exercise 4.23

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	BEQ	JMP	LW	SW
a.	40%	25%	5%	25%	5%
b.	60%	8%	2%	20%	10%

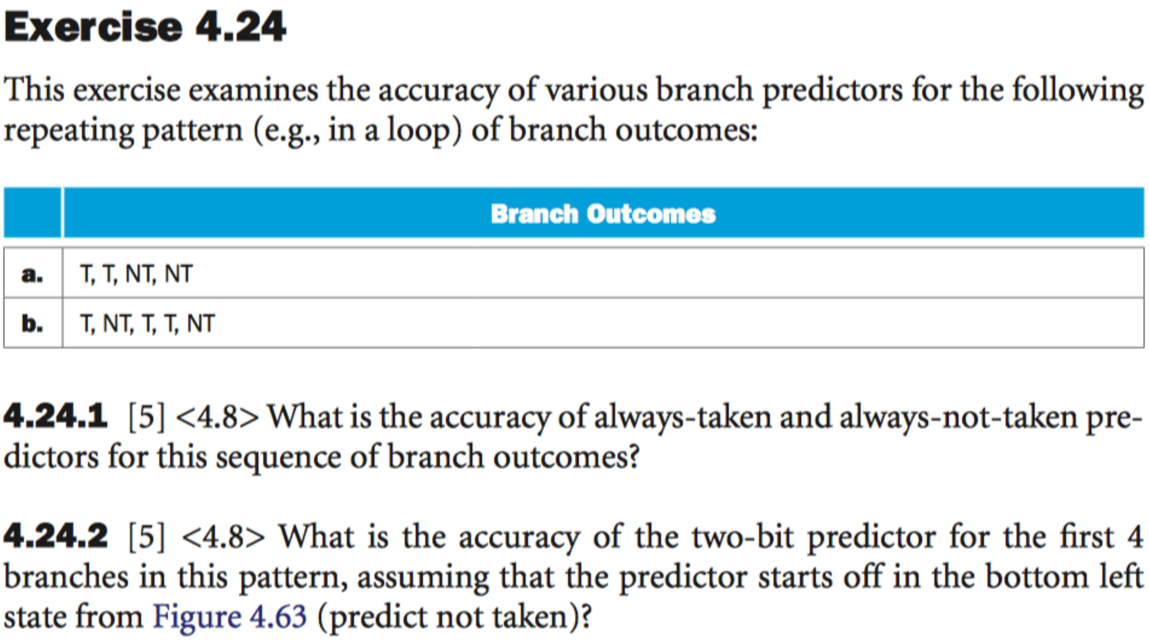
Also, assume the following branch predictor accuracies:

	Always-Taken	Always-Not-Taken	2-Bit
a.	45%	55%	85%
b.	65%	35%	98%

4.23.1 [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 EX stage, that there are no data hazards, and that no delay slots are used.

6. Exercise 4.24.1 (b) (5 points)

7. Exercise 4.24.2 (b) (5 points)



Exercise 4.24

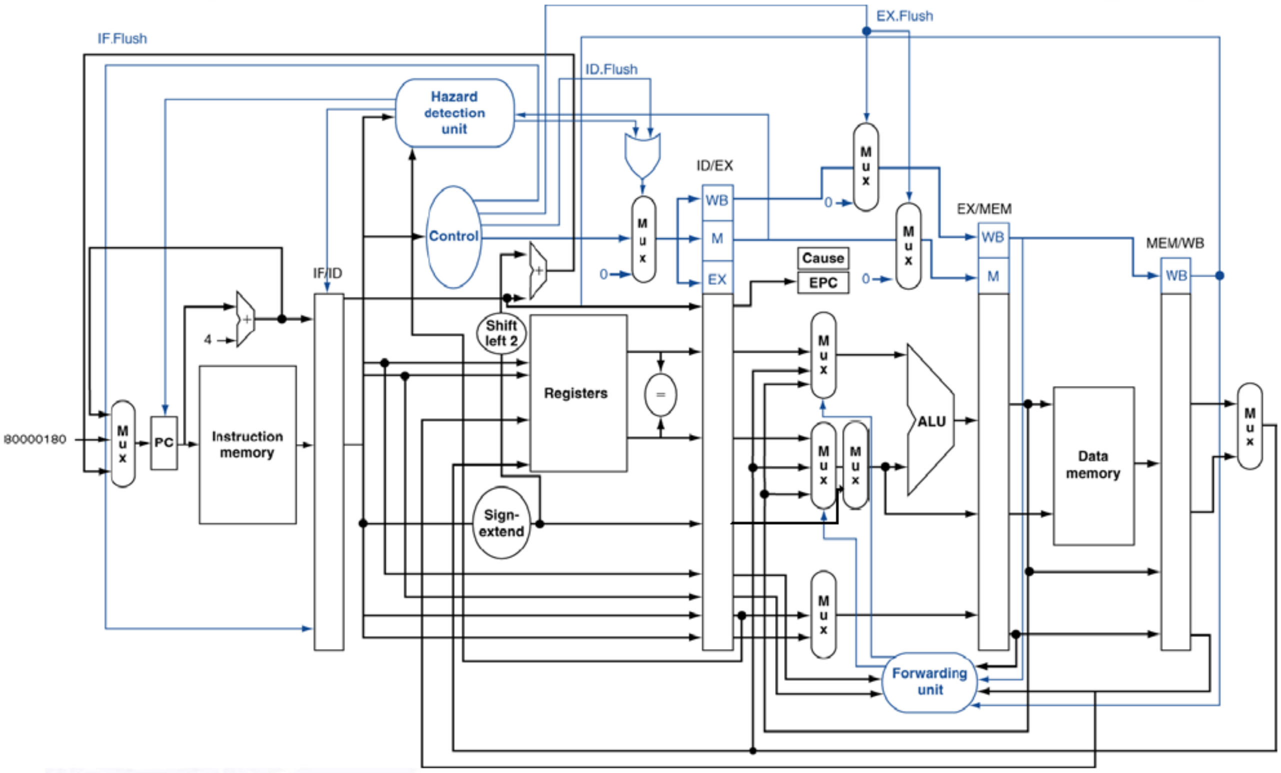
This exercise examines the accuracy of various branch predictors for the following repeating pattern (e.g., in a loop) of branch outcomes:

	Branch Outcomes
a.	T, T, NT, NT
b.	T, NT, T, T, NT

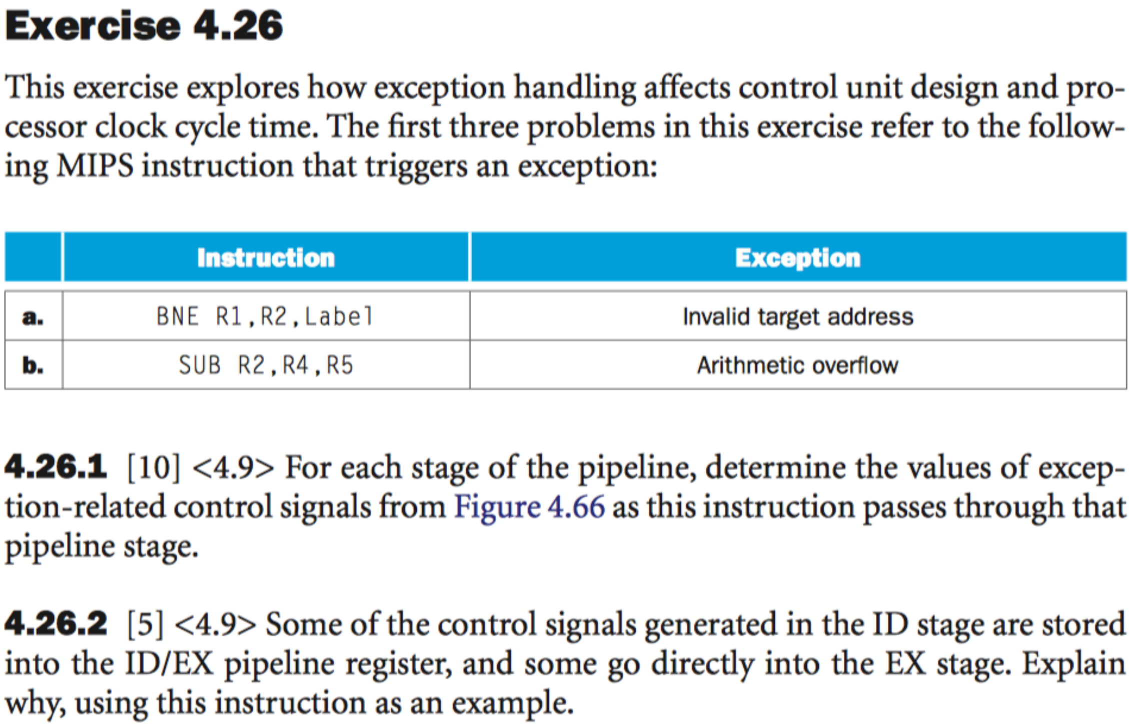
4.24.1 [5] <4.8> What is the accuracy of always-taken and always-not-taken predictors for this sequence of branch outcomes?

4.24.2 [5] <4.8> What is the accuracy of the two-bit predictor for the first 4 branches in this pattern, assuming that the predictor starts off in the bottom left state from Figure 4.63 (predict not taken)?

8. Exercise 4.26.1 (b), refer to the following figure instead of Figure 4.66. (5 points)



9. Exercise 4.26.2 (a) (5 points)



Exercise 4.26

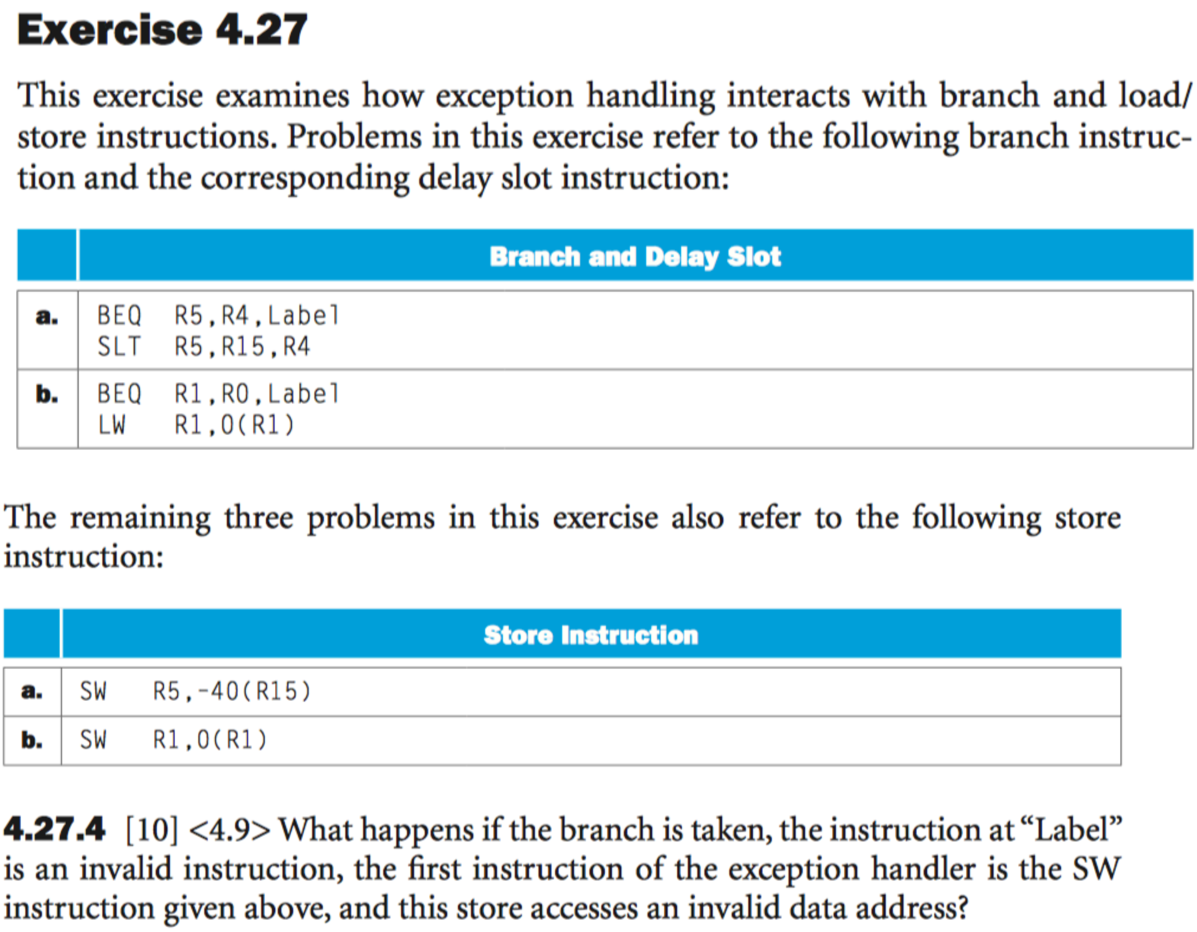
This exercise explores how exception handling affects control unit design and processor clock cycle time. The first three problems in this exercise refer to the following MIPS instruction that triggers an exception:

	Instruction	Exception
a.	BNE R1,R2,Label	Invalid target address
b.	SUB R2,R4,R5	Arithmetic overflow

4.26.1 [10] <4.9> For each stage of the pipeline, determine the values of exception-related control signals from Figure 4.66 as this instruction passes through that pipeline stage.

4.26.2 [5] <4.9> Some of the control signals generated in the ID stage are stored into the ID/EX pipeline register, and some go directly into the EX stage. Explain why, using this instruction as an example.

10. Excercise 4.27.4 (b) (20 points)



Exercise 4.27

This exercise examines how exception handling interacts with branch and load/store instructions. Problems in this exercise refer to the following branch instruction and the corresponding delay slot instruction:

	Branch and Delay Slot

a.	`BEQ  R5,R4,Label`
	`SLT  R5,R15,R4`

b.	`BEQ  R1,R0,Label`
	`LW   R1,0(R1)`

The remaining three problems in this exercise also refer to the following store instruction:

	Store Instruction
a.	`SW   R5,-40(R15)`
b.	`SW   R1,0(R1)`

4.27.4 [10] <4.9> What happens if the branch is taken, the instruction at "Label" is an invalid instruction, the first instruction of the exception handler is the SW instruction given above, and this store accesses an invalid data address?