

Homework 2 (Due May 22 before final exam)

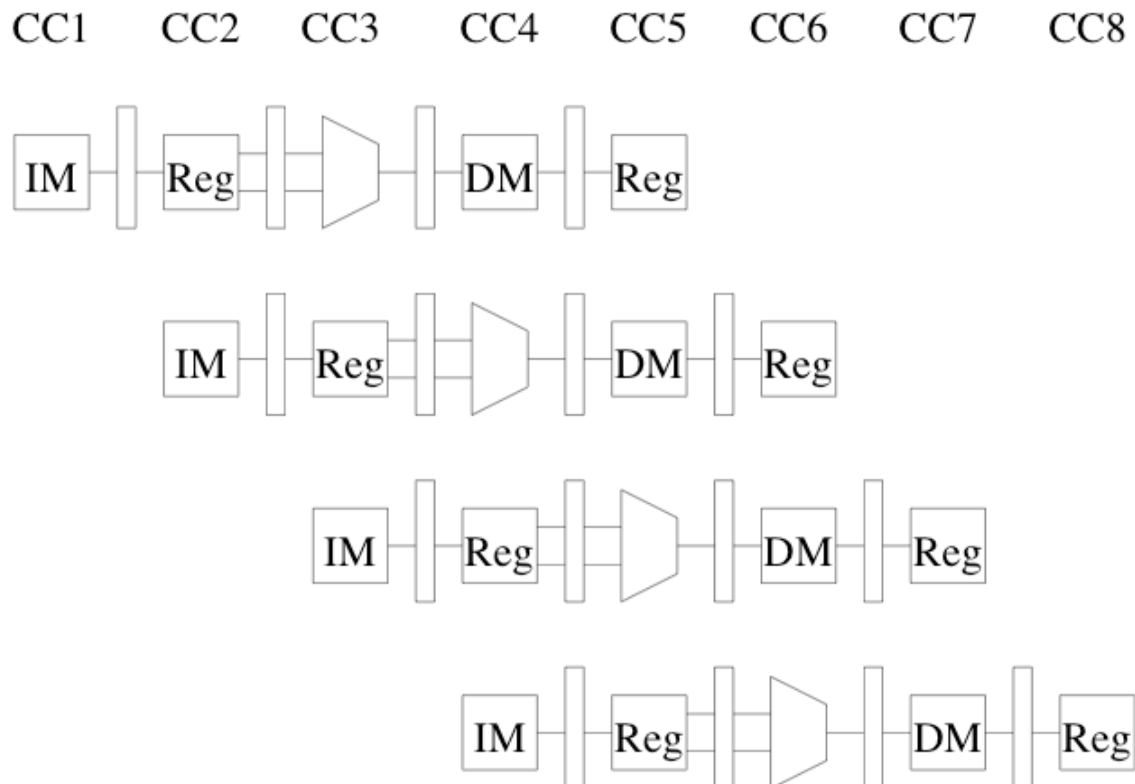
Problem 1:

Consider the following MIPS code sequence:

```

4 add $2, $3, $4
8 lw $1, 100($2)
12 add $3, $1, $2
16 sw $2, 20($1)
    
```

(a) In the diagram below, label each row with an executed instruction, assuming that the code starts executing at address 4. Mark all data dependencies by drawing straight lines (similar those in text Figure 6.28) between when the result is in the register file and when it needs to be taken from the register file. Assume that the code is to run on the pipelined datapath in text Figure 6.27 (this datapath implements neither stalling nor forwarding, but registers write in the first half and read in the second half of the clock cycle). For each data dependency, label it either as a hazard or as a non-hazard.



(b) Modify the code to remove the data hazards by inserting a minimum number of NOPs. In your solution, give both new and old instruction addresses.

New Address	Code

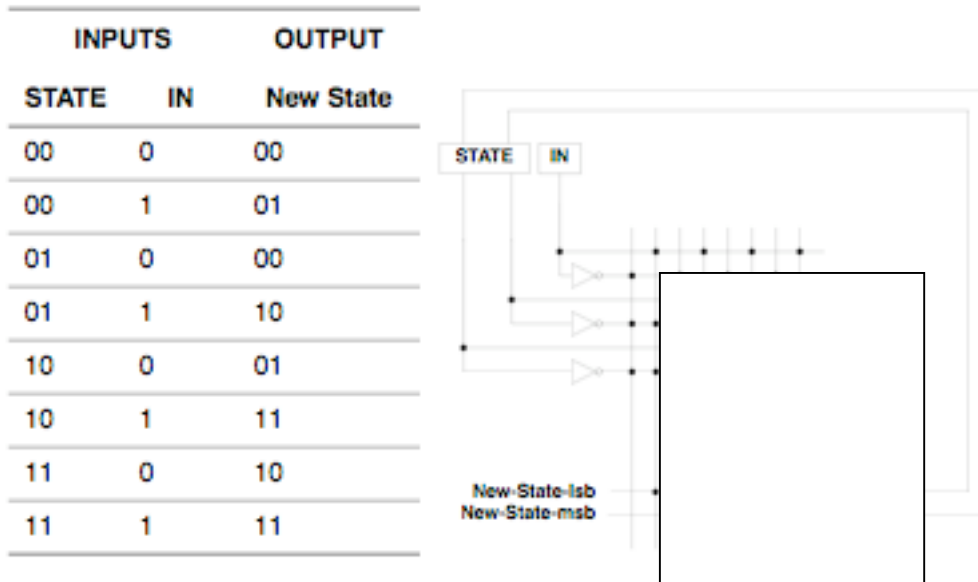
Problem 3:

Consider the single-cycle datapath in Figure 5.17 in the textbook. It is possible that hardware fault would occur making a control signal get stuck in a certain value, either 0 or 1. If each of the control signals listed below would suffer an individual stuck-at-value fault, certain instructions would not work properly. For each of these cases, identify at least one instruction or class of instructions that would fail and explain why.

1. RegWrite = 1
2. ALUop0 = 1
3. Branch = 0
4. MemRead = 0
5. MemWrite = 1

Problem 4:

(a) Construct a finite state machine that has four states and two possible input values (a single input that is either **0** or **1**). The state machine should implement a *saturating counter*: i.e. every input value of **1** should cause the state machine to go to the next highest state number (from **00** to **01**, from **01** to **10**, from **10** to **11**), and for every input value of **0** it should go to the next lowest state number (from **11** to **10**, from **10** to **01**, from **01** to **00**). The state machine should **NOT** wrap around from **11** to **00** or from **00** to **11**—this is what it means to be a saturating counter: the values *saturate* at **3** and **0**. Show a truth table and resulting FSM. You can use the same PLA notation as in the diagram below if you like. Finish the and-or gate array to implement the state machine. The first row of the and-gate array and the first two columns of the or-gate array have been given.



(b) Construct a truth table that has three inputs, and outputs the control signals for data forwarding. The three inputs are the preceding instruction, the second instruction that data depends on the first instruction, and the number of cycles between the two instructions. Assume only one operand need to be forwarded from the first instruction to the second instruction, and the instructions in between neither read nor write to the registers or memory units used by these two. Complete the truth table, indicating which data forwarding signals should be asserted.

EX/MEM: The ALU operand is forwarded from the prior ALU result.

MEM/WB: The ALU operand is forwarded from data memory.

1 st Inst	2 nd Inst	Cycles in between	EX/MEM	MEM/WB
lw	add	0		
lw	add	1		
lw	add	2		
lw	add	3		
sw \$1, 0(\$2)	lw \$3, 0(\$2)	0		
sw \$1, 0(\$2)	lw \$3, 0(\$2)	1		
sw \$1, 0(\$2)	lw \$3, 0(\$2)	2		
sw \$1, 0(\$2)	lw \$3, 0(\$2)	2		