4.10 In this exercise, we examine how resource hazards, control hazards, and Instruction Set Architecture (ISA) design can affect pipelined execution. Problems in this exercise refer to the following fragment of MIPS code:

```
sw r16,12(r6)
lw r16,8(r6)
beq r5,r4,Label # Assume r5!=r4
add r5,r1,r4
slt r5,r15,r4
```

Assume that individual pipeline stages have the following latencies:

IF	ID	EX	MEM	WB
200ps	120ps	150ps	190ps	100ps

4.10.1 [10] <§4.5> For this problem, assume that all branches are perfectly predicted (this eliminates all control hazards) and that no delay slots are used. If we only have one memory (for both instructions and data), there is a structural hazard every time we need to fetch an instruction in the same cycle in which another instruction accesses data. To guarantee forward progress, this hazard must always be resolved in favor of the instruction that accesses data. What is the total execution time of this instruction sequence in the 5-stage pipeline that only has one memory? We have seen that data hazards can be eliminated by adding nops to the code. Can you do the same with this structural hazard? Why?

Here is the pipeline stages:

sw r16,12(r6)	IF	ID	EX	M	WB						
lw r16,8(r6)		IF	ID	EX	M	WB					
beq r5,r4,L l			IF	ID	EX	M	WB				
add r5,r1,r4						IF	ID	EX	M	WB	
slt r5,r15,r4							IF	ID	EX	M	WB

The total execution time is 11 clock cycles.

Nops cannot be added to the stages to eliminate the hazard. This hazard must be addressed with a hardware hazard detection unit in the processor. The nops need to be fetched like other instructions. In the chart, pipelines stages has no structural hazards because the add and slt instructions are delayed for 2 clock cycles. Thus the structural hazards are eliminated.

4.13 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 sequence of instructions, and assume that it is executed on a 5-stage pipelined datapath:

```
add r5,r2,r1
lw r3,4(r5)
lw r2,0(r2)
or r3,r5,r3
sw r3,0(r5)    .
```

4.13.1 [5] <§4.7> If there is no forwarding or hazard detection, insert nops to ensure correct execution.

1	IF	ID	EX	M	WB									
2		IF	ID	Nop	Nop	EX	M	WB						
3			IF	Nop	Nop	ID	EX	M	WB					
4						IF	ID	Nop	EX	M	WB			
5							IF	Nop	ID	Nop	Nop	EX	M	WB

4.13.4 [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.

The PCwrite is always 1 because there is no stalling. IF/ID read signals is always 1 too. The control signal goes to the ID/EX mux is 0.

For the forwarding unit ForwardA and ForwardB, they are all X(don’t care). Because the first two cycles have no EX.
During the third cycle, the first instruction enters its EX. Its operands come from Regs, so ForwardA and ForwardB are both 00.
In the fourth cycle, the second instruction enters its EX. The first operand(r5) comes from forwading the prior ALU result, so ForwardA = 10. The second operand is the offset comes from the instruction code itself, so ForwardB = 00.
During the last cycle, the third instruction enters its EX. The operands come from Regs, so ForwardA and ForwardB are both 00.