Disclaimer:

1. The solution is just for your reference. They may contain some mistakes. DO TRY to solve the problems by yourself. Please also pay attentions to the course website for the updates.

2. Try not to use pseudoinstructions for any exercises that ask you to produce MIPS code. Your goal should be to learn the real MIPS instruction set, and if you are asked to count instructions, your count should reflect the actual instructions that will be executed and not the pseudoinstructions.

Selected exercise for Chapter 4: 4.8.1~4.8.5, 4.9,

4.8.1 Solution: Pipeline: 350 ps. Single-cycle: 250+ 350+150+300+200=1250ps

4.8.2 Solution: Pipeline: 350*5= 1750ps. Single-cycle: 1250ps

4.8.3 Solution: Stage to split is ID because it has the largest latency.

IF	ID1	ID2	EX	MEM	WB
250ps	175ps	175ps	150ps	300ps	200ps

New clock cycle time is 300 ps

4.8.4

Answer: 35%

LW and SW instructions use the data memory. As a result, the utilization of the data memory is 15% + 20% = 35%.

4.8.5

Answer:65%

Similarly, ALU and LW instructions use the register block's write port. As a result, the utilization of the register block's write port is 40% + 25% = 65%

4.9.1

Instruction sequence	Dependences
I1: OR R1,R2,R3 I2: OR R2,R1,R4 I3: OR R1,R1,R2	RAW on R1 from I1 to I2 and from I1 to I3 RAW on R2 from I2 to I3 WAR on R2 from I1 to I2 WAR on R1 from I2 to I3 WAW on R1 from I1 to I3

Hint:

	Write	Read	Read
I1	R1	R2	R3
I2	R2	R1	R4
I3	R1	R1	R2

4.9.2 In the basic five-stage pipeline WAR and WAW dependences do not cause any hazards. Without forwarding, any RAW dependence between an instruction and the next two instructions (if register read happens in the second half of the clock cycle and the register write happens in the first half). The code that eliminates these hazards by inserting NOP instructions is:

Instruction sequence	
OR R1,R2,R3	
NOP	Delay I2 to avoid RAW hazard on R1 from I1
NOP	
OR R2,R1,R4	
NOP	Delay I3 to avoid RAW hazard on R2 from I2
NOP	
OR R1,R1,R2	

4.9.3 With full forwarding, an ALU instruction can forward a value to EX stage of the next instruction without a hazard. However, a load cannot forward to the EX stage of the next instruction (by can to the instruction after that). The code that eliminates these hazards by inserting NOP instructions is:

Instruction sequence	
OR R1,R2,R3	
OR R2,R1,R4	No RAW hazard on R1 from I1 (forwarded)
OR R1,R1,R2	No RAW hazard on R2 from I2 (forwarded)

4.9.4 The total execution time is the clock cycle time times the number of cycles. Without any stalls, a three-instruction sequence executes in 7 cycles (5 to complete the first instruction, then one per instruction). The execution without forwarding must add a stall for every NOP we had in 4.9.2, and execution forwarding must add a stall cycle for every NOP we had in 4.9.3. Overall, we get:

No forwarding: (7 + 4)*250 ps = 2750 ps

With forwarding: 7*300 ps = 2100 ps. Speedup due to forwarding= 1.31

4.9.5 With ALU-ALU-only forwarding, an ALU instruction can forward to the next instruction, but not to the second-next instruction (because that would be forwarding from MEM to EX). A load cannot forward at all, because it determines the data value in MEM stage, when it is too late for ALU-ALU forwarding. We have:

Instruction sequence	
OR R1,R2,R3	
OR R2,R1,R4	ALU-ALU forwarding of R1 from I1
Nop	Because no MEM to EX forwarding
Nop	
OR R1,R1,R2	

4.9.6 solution

No forwarding: (7 + 4)*250 ps = 2750 ps

With ALU-ALU forwarding only: (5+4)*290=2610ps

Speedup with ALU-ALU forwarding: 2750/2610=1.053