Homework 4 Solution

Exercise 4.12

In this exercise, we examine how pipelining affects the clock cycle time of the processor. Problems in this exercise assume that individual stages of the datapath have the following latencies:

	IF	ID	EX	MEM	WB
a.	250ps	350ps	150ps	300ps	200ps
b.	200ps	170ps	220ps	210ps	150ps

4.12.1 [5] <4.5> What is the clock cycle time in a pipelined and non-pipelined processor?

4.12.2 [10] <4.5> What is the total latency of an `LW` instruction in a pipelined and non-pipelined processor?

4.12.3 [10] <4.5> If we can split one stage of the pipelined datapath into two new stages, each with half the latency of the original stage, which stage would you split and what is the new clock cycle time of the processor?

4.12.1

	Pipelined	Single-Cycle
a.	350ps	1250ps
b.	220ps	950ps

4.12.2

	Pipelined	Single-Cycle
a.	1750ps	1250ps
b.	1100ps	950ps

4.12.3

	Stage to Split	New Clock Cycle Time
a.	ID	300ps
b.	EX	210ps

Pipeline's clock cycle follows the biggest stage.

4.12.2:
Pipeline = 5*cycles,
Single-cyclye = 1*cycle=addition of every stage

Exercise 4.13

In this exercise, we examine how data dependences affect execution in the basic 5-stage pipeline described in Section 4.5. Problems in this exercise refer to the following sequence of instructions:

	Instruction Sequence

a.	SW R16,-100(R6)
	LW R4,8(R16)
	ADD R5,R4,R4

b.	OR R1,R2,R3 OR R2,R1,R4 OR R1,R1,R2

4.13.1 [10] <4.5> Indicate dependences and their type.

4.13.2 [10] <4.5> Assume there is no forwarding in this pipelined processor. Indicate hazards and add NOP instructions to eliminate them.

4.13.3 [10] <4.5> Assume there is full forwarding. Indicate hazards and add NOP instructions to eliminate them.

RAW: read after write
WAR: write after read
WAW: write after write

4.13.1

	Instruction Sequence	Dependences
a.	I1: SW R16,-100(R6) I2: LW R4,8(R16) I3: ADD R5,R4,R4	RAW on R4 from I2 to I3
b.	I1: OR R1,R2,R3 I2: OR R2,R1,R4 I3: OR R1,R1,R2	RAW on R1 from I1 to I2 and 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

	Instruction Sequence	

a.	SW R16,-100(R6)
	LW R4,8(R16)
	NOP
	NOP
	ADD R5,R4,R4
		Delay I3 to avoid RAW hazard on R4 from I2

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

4.13.2
WAR and WAW won't cause hazard.
RAW will cause hazard if no forwarding.

4.13.3 [10] <4.5> Assume there is full forwarding. Indicate hazards and add NOP instructions to eliminate them.

	Instruction Sequence	

a.	SW R16,-100(R6)
	LW R4,8(R16)
	NOP
	ADD R5,R4,R4
		
		Delay I3 to avoid RAW hazard on R4 from I2
		Value for R4 is forwarded from I2 now

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

Full forwarding means that an ALU result can forward to the EX stage of next instruction. However, load can't forward to the EX stage of next instruction.

	Without Forwarding	With Full Forwarding	With ALU-ALU Forwarding Only
a.	250ps	300ps	290ps
b.	180ps	240ps	210ps

4.13.4 [10] <4.5> What is the total execution time of this instruction sequence without forwarding and with full forwarding? What is the speedup achieved by adding full forwarding to a pipeline that had no forwarding?

4.13.5 [10] <4.5> Add `NOP` instructions to this code to eliminate hazards if there is ALU-ALU forwarding only (no forwarding from the MEM to the EX stage).

4.13.6 [10] <4.5> What is the total execution time of this instruction sequence with only ALU-ALU forwarding? What is the speedup over a no-forwarding pipeline?

	No Forwarding	With Forwarding	Speedup Due to Forwarding
a.	(7 + 2) × 250ps = 2250ps	(7 + 1) × 300ps = 2400ps	0.94 (This is really a slowdown)
b.	(7 + 4) × 180ps = 1980ps	7 × 240ps = 1680ps	1.18

5 cycles for the first instruction, then add one for each extra instruction.

4.13.5 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	

a.	`SW R16,-100(R6)`
	`LW R4,8(R16)`
	`ADD R5,R4,R4`
		ALU-ALU forwarding of R4 from I2

b.	`OR R1,R2,R3`
	`OR R2,R1,R4`
	`OR R1,R1,R2`
		ALU-ALU forwarding of R1 from I1
		ALU-ALU forwarding of R2 from I2

4.13.6

	No Forwarding	With ALU-ALU Forwarding Only	Speedup with ALU-ALU Forwarding
a.	(7 + 2) × 250ps = 2250ps	7 × 290ps = 2030ps	1.11
b.	(7 + 4) × 180ps = 1980ps	7 × 210ps = 1470ps	1.35

Exercise 4.15

In this exercise, we examine how the ISA affects pipeline design. Problems in this exercise refer to the following new instruction:

a.	`ADDM Rd,Rt+Offs(Rs)`	Rd=Rt+Mem[Offs+Rs]
b.	`BEQM Rd,Rt,Offs(Rs)`	If Rt=Mem[Offs+Rs] then PC = Rd

4.15.1 [20] <4.5> What must be changed in the pipelined datapath to add this instruction to the MIPS ISA?

4.15.2 [10] <4.5> Which new control signals must be added to your pipeline from 4.15.1?

4.15.3 [20] <4.5, 4.13> Does support for this instruction introduce any new hazards? Are stalls due to existing hazards made worse?

4.15.1

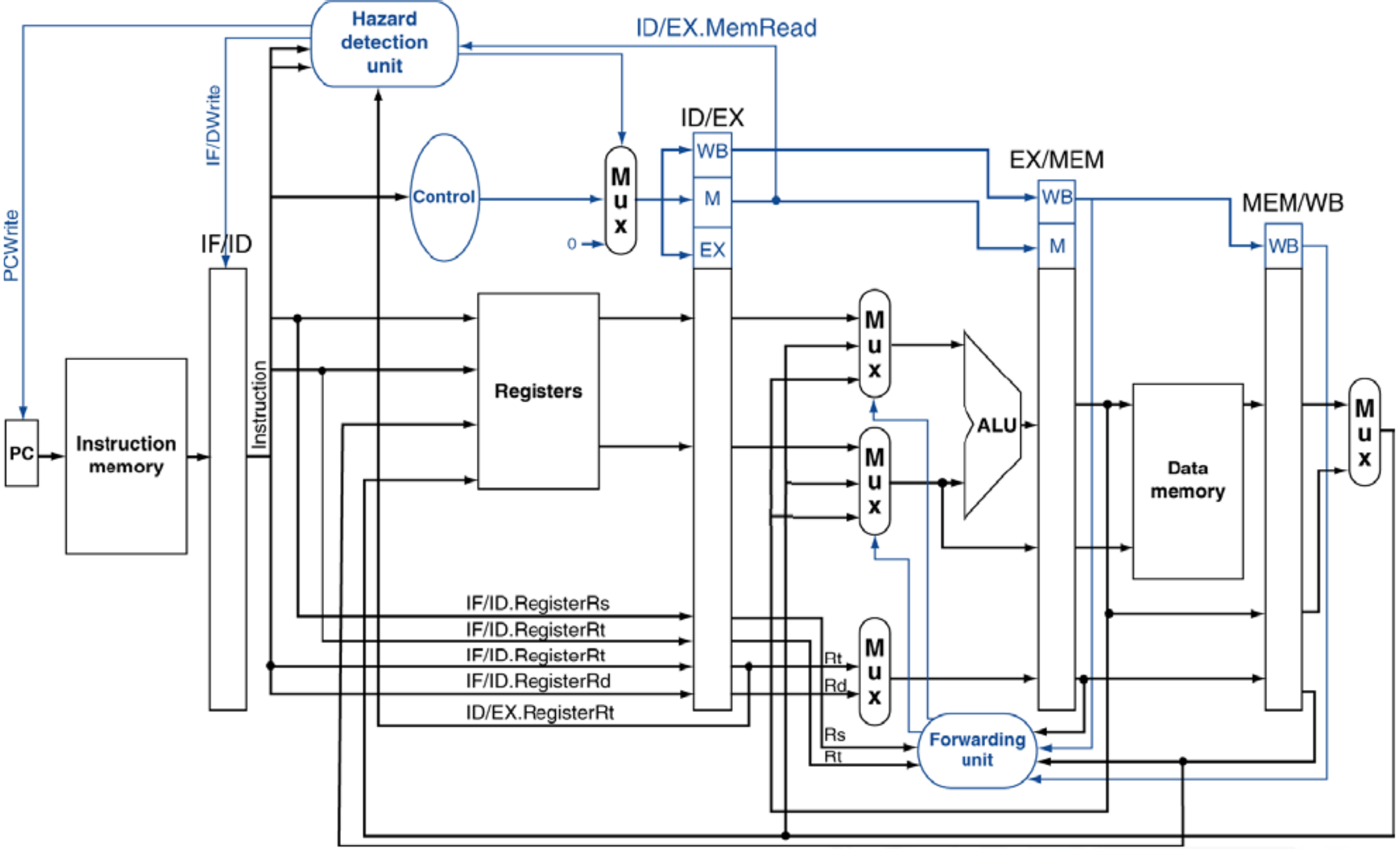
a) We need to get Mem first, so the stage before MEM is same. Then we'll need to compute before WB, so we need a stage to compute, and a wire to conduct Rt.
b) We will change PC after MEM stage, so a extra PC changing input is required form Rd, and therefore, a new Mux. Also, we need to change register file, because it's the first time we do three read.

4.15.2

a) A signal that selects whether to compute or just pass the memory-read-out.
b) A signal to decide whether is BEQM, like we do support BNE in last homework.

4.15.3

a) Since we need to do calculation after MEM, there might need some forwarding or add stall to due with data hazard. However, if ADDM is after a instruction to write, even with the forwarding, two cycles are necessary.
b) The instruction may cause control hazard, since we can only know address after MEM, so it may be 2 cycles longer than BEQ.


Hazard detection unit
ID/EX.MemRead
IF/DWrite
PCWrite
Control
M u x
0
ID/EX
WB
M
EX
EX/MEM
WB
M
MEM/WB
WB
IF/ID
PC
Instruction memory
Instruction
Registers
M u x
ALU
Data memory
M u x
M u x
M u x
IF/ID.RegisterRs
IF/ID.RegisterRt
IF/ID.RegisterRt
IF/ID.RegisterRd
ID/EX.RegisterRt
Rt
Rd
Rs
Rt
Forwarding unit

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)

4.21.1 [5] <4.7> If there is no forwarding or hazard detection, insert NOPs to ensure correct execution.

4.21.2 [10] <4.7> Repeat 4.21.1 but now use NOPs only when a hazard cannot be avoided by changing or rearranging these instructions. You can assume register R7 can be used to hold temporary values in your modified code.

4.21.1

a.	ADD R5,R2,R1
	NOP
	NOP
	LW R3,4(R5)
	LW R2,0(R2)
	NOP
	OR R3,R5,R3
	NOP
	NOP
	SW R3,0(R5)

b.	LW R2,0(R1)
	NOP
	NOP
	AND R1,R2,R1
	LW R3,0(R2)
	NOP
	LW R1,0(R1)
	NOP
	NOP
	SW R1,0(R2)

a.	I1: ADD R5,R2,R1 I3: LW R2,0(R2) NOP I2: LW R3,4(R5) NOP NOP I4: OR R3,R5,R3 NOP NOP I5: SW R3,0(R5)	Moved up to fill NOP slot Had to add another NOP here, so there is no performance gain
b.	I1: LW R2,0(R1) NOP NOP I2: AND R1,R2,R1 I3: LW R3,0(R2) NOP I4: LW R1,0(R1) NOP NOP I5: SW R1,0(R2)	No improvement is possible. There is a chain of RAW dependences from I1 to I2 to I4 to I5, and each step in the chain has to be separated by two instructions.

1. Try to avoid dependence.
2. Use R7 to deal with WAW and WAR.

4.21.3 [10] <4.7> If the processor has forwarding, but we forgot to implement the hazard detection unit, what happens when this code executes?

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.

4.21.3 If RAW, and the write instruction is load, there need a stall.

a) Correct.

b) In I2, R2 is wrong. In I5, R1 is wrong.

	Instruction Sequence	First Five Cycles 1 2 3 4 5	Signals

a.	ADD R5,R2,R1
	LW R3,4(R5)
	LW R2,0(R2)
	OR R3,R5,R3
	SW R3,0(R5)
		IF ID EX MEM WB
		IF ID EX MEM
		IF ID EX
		IF ID
		IF
			1: PCWrite = 1, ALUin1 = X, ALUin2 = X
			2: PCWrite = 1, ALUin1 = X, ALUin2 = X
			3: PCWrite = 1, ALUin1 = 0, ALUin2 = 0
			4: PCWrite = 1, ALUin1 = 1, ALUin2 = 0
			5: PCWrite = 1, ALUin1 = 0, ALUin2 = 0

b.	LW R2,0(R1)
	AND R1,R2,R1
	LW R3,0(R2)
	LW R1,0(R1)
	SW R1,0(R2)
		IF ID EX MEM WB
		IF ID *** EX
		IF *** ID
		IF
			1: PCWrite = 1, ALUin1 = X, ALUin2 = X
			2: PCWrite = 1, ALUin1 = X, ALUin2 = X
			3: PCWrite = 1, ALUin1 = 0, ALUin2 = 0
			4: PCWrite = 0, ALUin1 = X, ALUin2 = X
			5: PCWrite = 1, ALUin1 = 2, ALUin2 = 0

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.

4.21.6 [20] <4.7> For the new hazard detection unit from 4.21.5, specify which output signals it asserts in each of the first five cycles during the execution of this code.

4.21.5 Check Rd for R-type and load, to deal with ID stage needed MEM and EX stage.
Check the destination register (the bottommost output of EX/MEM register.
Rd from ID/EX register and the output number of the output register from EX/MEM.
Rt field from ID/EX. (already in)
No extra output.

4.21.6 As explained for 4.21.5, we only need to specify the value of the PCWrite signal, because IF/IDWrite is equal to PCWrite and the ID/EXzero signal is its opposite. We have:

	Instruction Sequence	First Five Cycles 1	2	3	4	5	Signals

a.	ADD R5,R2,R1
	LW R3,4(R5)
	LW R2,0(R2)
	OR R3,R5,R3
	SW R3,0(R5)
		IF	ID
			IF
				EX
				ID
				IF
					MEM
					***
					***
						WB
						***
						***
						***
							1: PCWrite = 1
							2: PCWrite = 1
							3: PCWrite = 1
							4: PCWrite = 0
							5: PCWrite = 0

b.	LW R2,0(R1)
	AND R1,R2,R1
	LW R3,0(R2)
	LW R1,0(R1)
	SW R1,0(R2)
		IF	ID
			IF
				EX
				ID
				IF
					MEM
					***
					***
						WB
						***
						***
						***
							1: PCWrite = 1
							2: PCWrite = 1
							3: PCWrite = 1
							4: PCWrite = 0
							5: PCWrite = 0