4.9. (1)

指令	相关
I1: or R1, R2, R3	RAW on R1 from I1 to I2 and I3
I2: or R2, R1, R4	RAW on R2 from I2 to I3
I3: or R1, R1, R2	WAR on R2 from I1 to I2
	WAR on R1 from I2 to I3
	WAW on R1 from I1 to I3

(2)

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	

(3)

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) No forwarding: (7+4)·180 ps = 1980 ps

with forwarding: (7+240) ps = 1680 ps

speed up: $\frac{1980}{1680} = 1.18$

(5)

or R1, R2, R3	
or R2, R1, R4	ALU-ALU forwarding of R1 from I1
or R1, R1, R2	ALU-ALU forwarding of R2 from I2

(6) No forwarding: (7+4)·180 ps = 1980 ps

with forwarding: 7·210 ps = 1470 ps

speed up: $\frac{1980}{1470} = 1.35$

4.10. (1)

Instruction	Pipeline Stage	Cycles
SW R16, 12(R6)	IF ID EX MEM WB	11
LW R16, 8(R6)	IF ED EX MEM WB	
BEQ R5, R4, L61	IF ID EX MEM WB	
ADD R3, R1, R4	IF ID EX MEM WB	
SLT R5, R15, R4	IF ID EX MEM WB	

(2) Instructions Executed: 5

Cycles with 5 stages: 4+5 = 9

Cycles with 4 stages: 3+5 = 8

speedup: 9/8 = 1.13

(3) Instructions Executed: 5

Branches Executed: 1

Cycles with branch in EXE: 4+5+1×2 = 11

Cycles with branch in ID: 4+5+1×1 = 10

speedup: 11/10 = 1.10

(4) Cycle time with 5 stages: 200ps (IF)

Cycle time with 4 stages: 210ps (MEM + 20ps)

Speedup: $(9 \cdot 200)/(8 \cdot 210) = 1.07$

(5) New ID latency: 180ps

New EX Latency: 140ps

New cycle time: 200ps (IF)

Old cycle time: 200ps (IF)

Speedup: $\frac{11 \cdot 200}{10 \cdot 200} = 1.10$

(6) Cycles with branch in EX: $4+5+1\times2 = 11$

Execution time (branch in EX): $11 \cdot 200\text{ps} = 2200\text{ps}$

cycles with branch in MEM: $4+5+1\times3 = 12$

Execution time (branch in MEM): $12 \cdot 200\text{ps} = 2400\text{ps}$

Speedup: $\frac{2200}{2400} = 0.92$

4.11 (1)

LW R1, 0(R1)	WB
LW R1, 0(R1)	EX MEM WB
BEQ R1, R0, Loop	ID xxx EX MEM WB
LW R1, 0(R1)	IF xxx ID EX MEM WB
AND R1, R1, R2	IF ID xxx EX MEM WB
LW R1, 0(R1)	IF xxx ID EX MEM
LW R1, 0(R1)	IF ID xxx
BEQ R1, R0, Loop	IF xxx

(2) Cycles per loop iteration: 8

Cycles in which all stages do useful work: 0

% of cycles in which all stages do useful work: 0%

4.12.(1) CPI: $1 + 0.35 \times 2 + 0.15 \times 1 = 1.85$

Stall Cycles: 46% (0.85/1.85)

(2) CPI: $1 + 0.2 = 1.2$

Stall Cycles: 17% (0.20/1.20)

(3) EX / MEM : 0.2 + 0.05 + 0.1 + 0.1 = 0.45

MEM / WB : 0.05 + 0.2 + 0.1 = 0.35

Fewer stall cycles with: MEM / WB

(a) Without forwarding: 1.85 · 150ps = 277.5 ps

With forwarding: 1.20 · 150 ps = 180 ps

Speedup : 1.54

(5) With full forwarding: 1.20 · 150 ps = 180 ps

Time-travel forwarding = 1.250ps = 250ps

Speedup: 0.72

(6) EX / MEM : 1.45 · 150 ps = 217.5

MEM / WB : 1.35 · 150ps = 202.5 ps

Shorter time per instruction with: MEM / WB

4.13: (1) 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)

(2) I1: ADD R5, R2, R1
I3: LW R2, 0(R2) — Moved up to fill NOP slot
NOP
I2: LW R3, 4(R5)
NOP — Had to add another NOP here,
NOP — so there is no performance gain
I4: OR R3, R5, R3
NOP
NOP
I5: SW R3, 0(R5)

(3) Code executes correctly (for both loads, there is no RAW dependence between the load and the next instruction).

(4)

Instruction sequence	First five cycles 1 2 3 4 5	signals

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

(5) we need to check Rd for R-type instructions and Rd for loads.

No additional outputs are needed. We can stall the pipeline using the three output signals that we already have.

(6)

Instruction sequence	First five cycles 1 2 3 4 5	signals

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 . . . . . .
	IF . . . . . .
	. . .
		1: PCWrite = 1
		2: PCWrite = 1
		3: PCWrite = 1
		4: PCWrite = 0
		5: PCWrite = 0

4.15 (1) Extra CPI: $3 \times (1-0.45) \times 0.25 = 0.41$

(2) Extra CPI: $3 \times (1-0.55) \times 0.25 = 0.34$

(3) Extra CPI: $3 \times (1-0.85) \times 0.25 = 0.113$

(4) CPI without conversion: $1 + 3 \times (1-0.85) \cdot 0.25 = 1.113$

CPI with conversion: $1 + 3 \times (1-0.85) \times 0.25 \times 0.5 = 1.056$

(5) CPI without conversion: 1.113

Cycles per original instruction with conversion: $1 + (1 + 3 \times (1-0.85)) \times 0.25 \times 0.5 = 1.181$

speedup from conversion: $1.113 / 1.181 = 0.94$

(6) Correctly predicted: B · 0.85

Correctly predicted non-loop-back: B · 0.05

Accuracy on non-loop-back branches: $(B \times 0.05)/(B \times 0.20) = 0.25$ (25%)