Lecture 19: Pipelining

- Today's topics:
  - Data hazards and instruction scheduling
  - Control hazards

11am

Example 2 – Bypassing

- Show the instruction occupying each stage in each cycle (with bypassing) if I1 is R1+R2→R3 and I2 is R3+R4→R5 and I3 is R3+R8→R9. Identify the input latch for each input operand.

	CYC-1	CYC-2	CYC-3	CYC-4	CYC-5	CYC-6	CYC-7	CYC-8
L1	IF I1	IF I2	IF I3	IF I4	IF I5	IF	IF	IF
L2	D/R	D/R I1	D/R I2	D/R I3	D/R I4	D/R	D/R	D/R
L3	ALU	ALU	**L3 L3** ALU I1	**L4 L3** ALU I2	**L5 L3** ALU I3	ALU	ALU	ALU
L4	DM	DM	DM	DM I1	DM I2	DM I3	DM	DM
L5	RW	RW	RW	RW	RW I1	RW I2	RW I3	RW

RR

Decode + Reg Rd

L5 L4 L3

mux

RW

Problem 0

DR → PoC

add $1, $2, $3
add $5, $1, $4

RAW dependence

● Point of Production

● Point of Consumption

middle of 5th stage

Without bypassing:

PoP

add $1, $2, $3:	IF	DR	AL	DM	RW			
add $5, $1, $4:		IF	DR	DR	DR	AL	DM	RW

PoC

2 more DR stages than ideal
⇒ 2 stalls

With bypassing:

PoP

add $1, $2, $3:	IF	DR	AL	DM	RW	
add $5, $1, $4:		IF	DR	AL	DM	RW

0 stalls

Used to be here

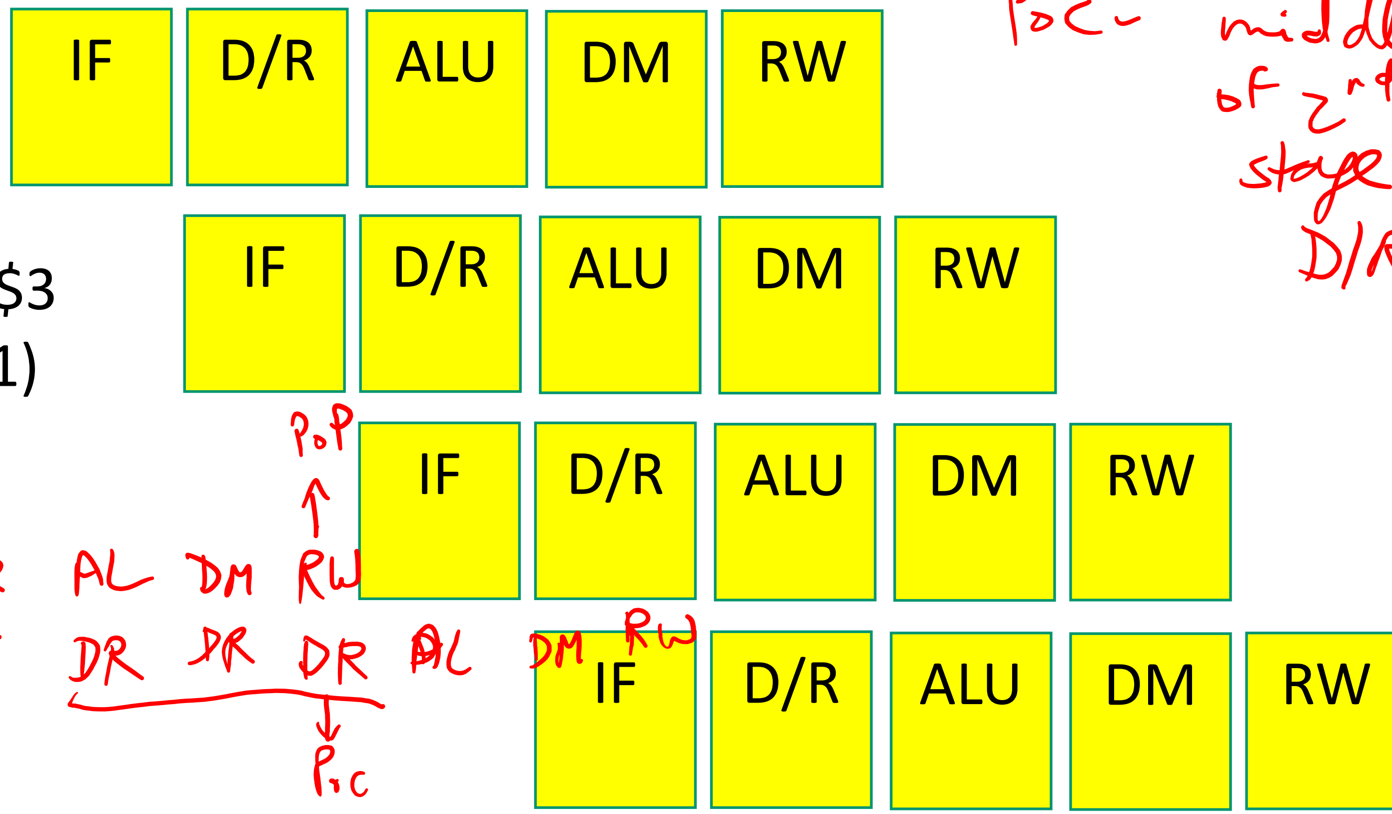
PoC

Problem 1 — without bypassing

PoP - middle of 5th stage (RW)

Data dep - RAW - Prod - Cons

PoC - middle of 2nd stage D/R



add $1, $2, $3
lw $4, 8($1)

ADD: IF DR AL DM RW

PoP

LW: IF DR DR DR AL DM RW

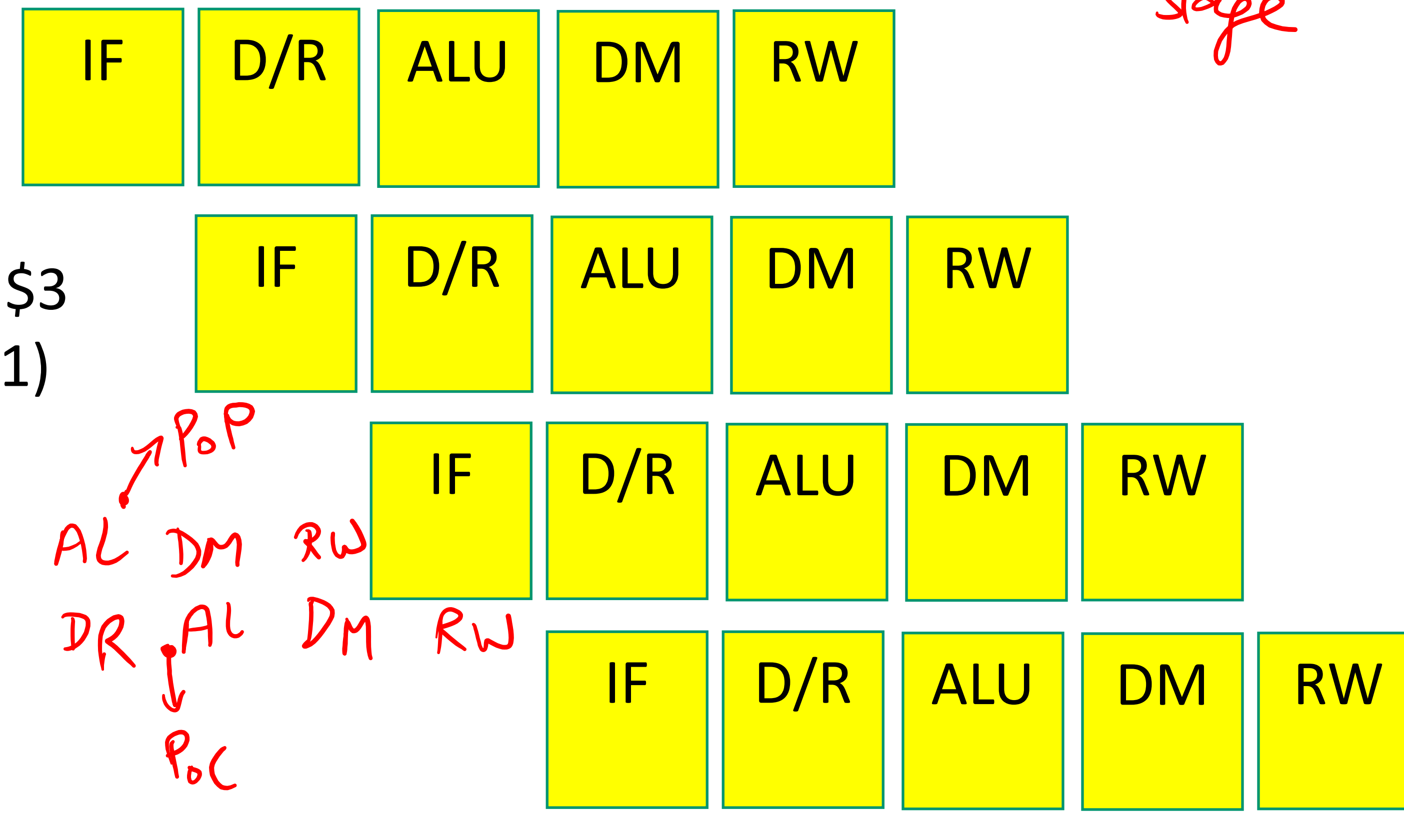
PoC

2 stalls

Problem 1 — with bypassing

PoP — end of ALU stage

PoC — start of ALU stage



add $1, $2, $3

lw $4, 8($1)

PoP

ADD: IF DR AL DM RW

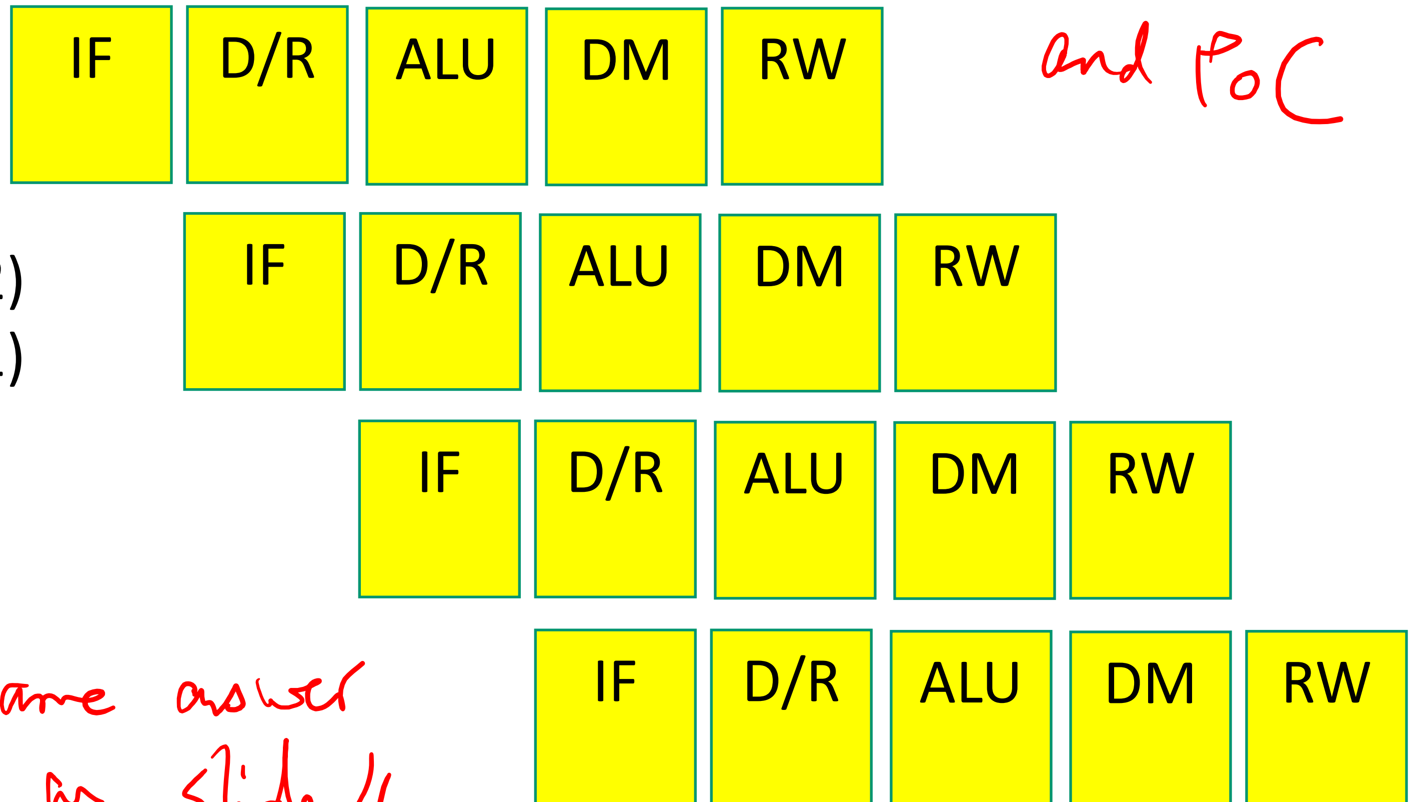
LW: IF DR AL DM RW

PoC

0 stall cycles

Problem 2 – without bypassing

same PoP
and PoC



lw $1, 8($2)
lw $4, 8($1)

same answer
as slide 4

2 stalls

Problem 2 – with bypassing

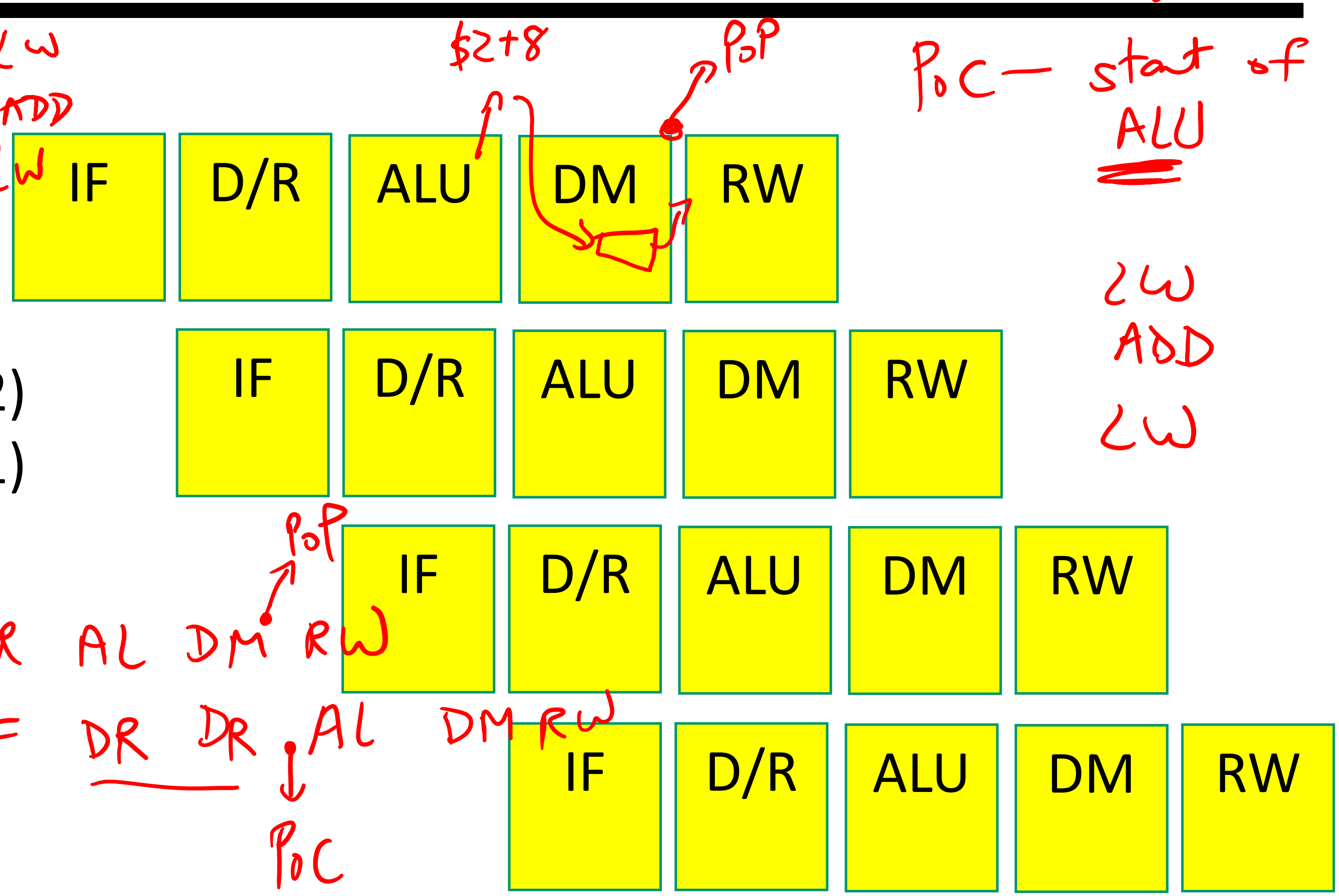
PoP – end of DM stage

PoC – start of ALU

LLW
LW
ADD

Reorder

LW
ADD
LW



lw $1, 8($2)
lw $4, 8($1)

LW1: IF DR AL DM RW

LW2: IF DR DR AL DM RW

PoC

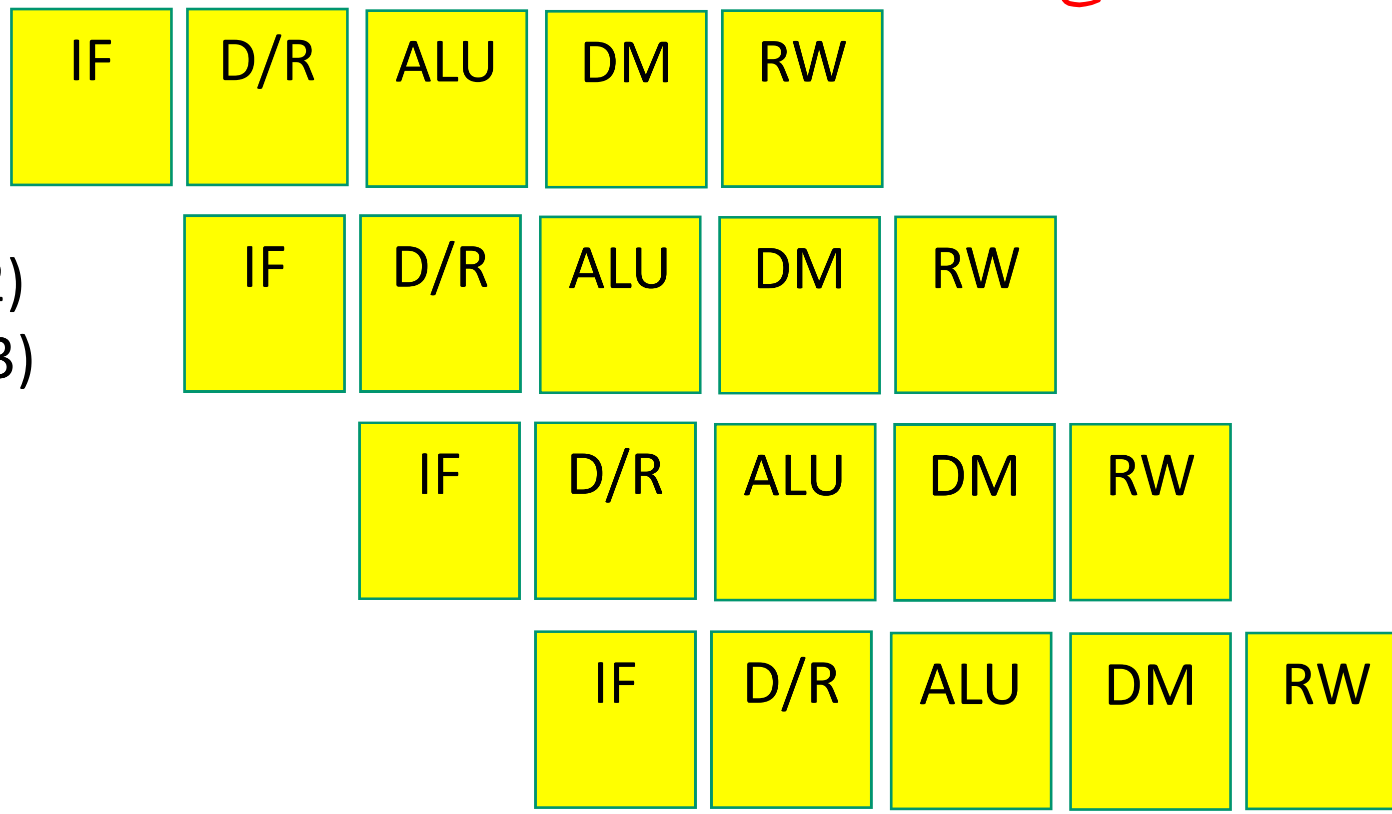
2 DR stages ⇒ 1 stall cycle

Problem 3

without byp – see slide 4

2 stalls

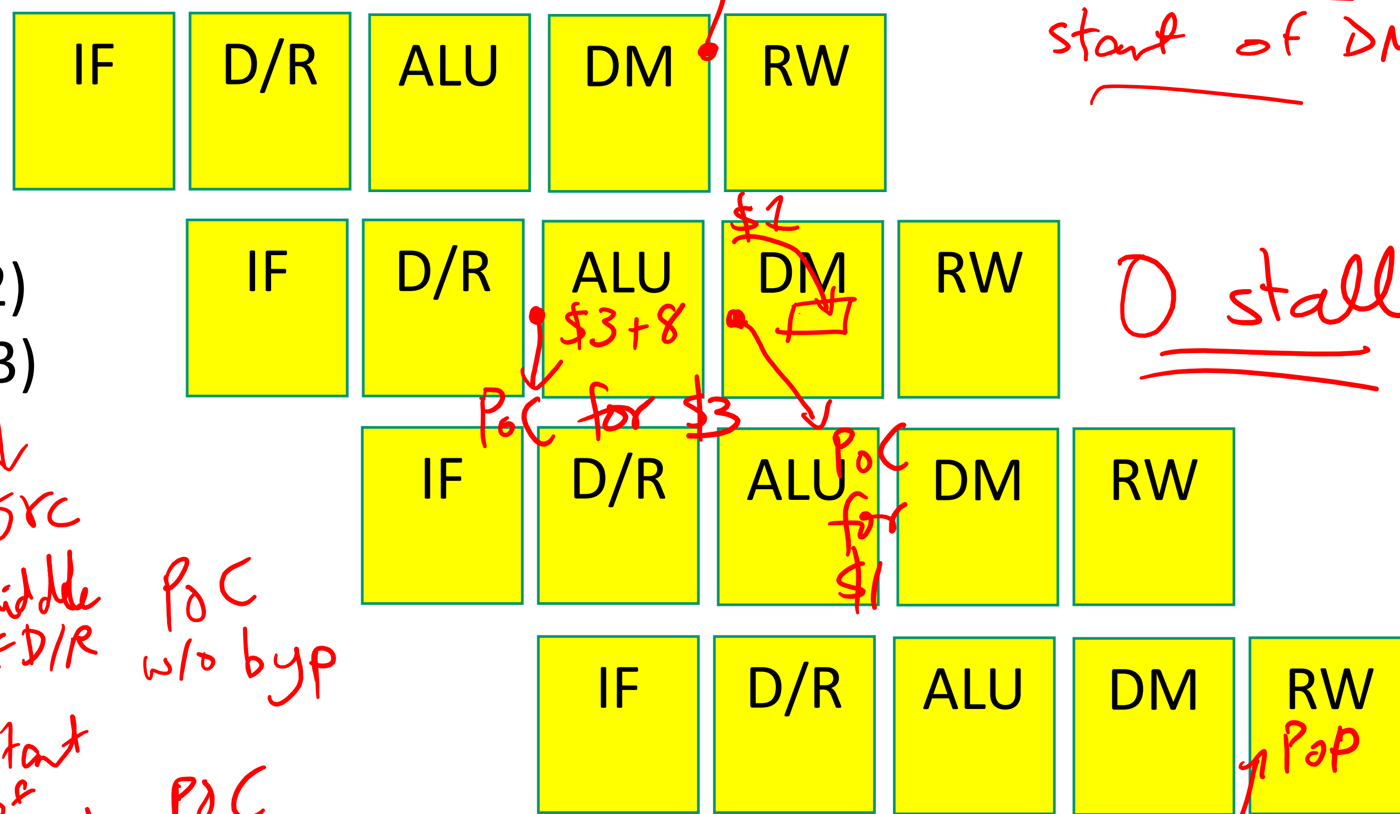
lw $1, 8($2)
sw $1, 8($3)



Problem 3 with byp

PoP – end of DM stage

PoP PoC – (focus on $1) start of DM



lw $1, 8($2)
sw $1, 8($3)

lw	sw
src middle of D/R	src middle of D/R
start of DM	start of ALU

PoC w/o byp

PoC with byp

0 stalls

LW : IF DR AL DM RW
SW : IF DR AL DM

PoC

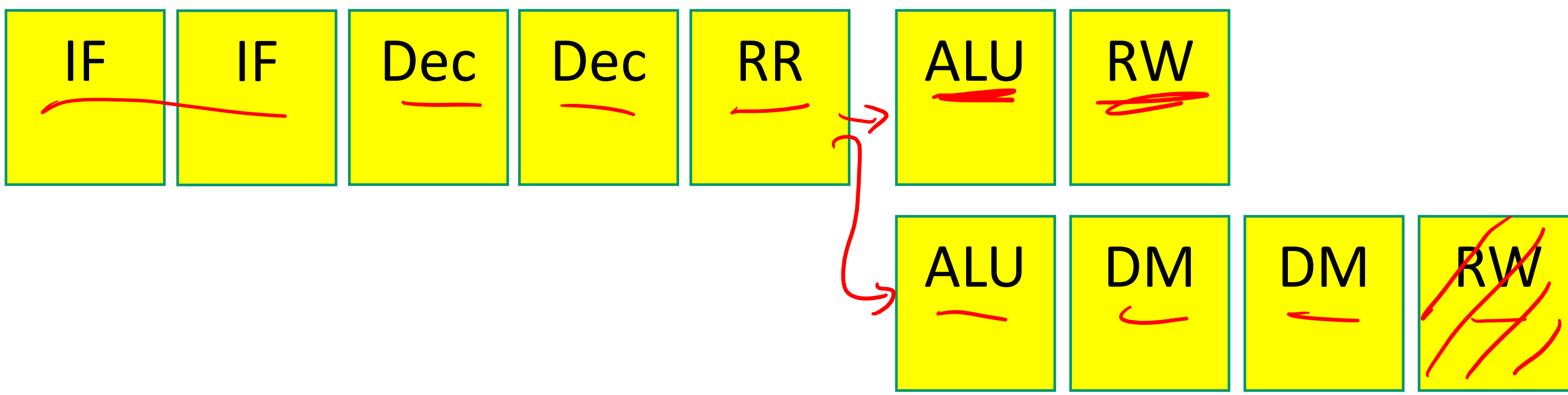
Problem 4 — w/o byp

PoP – end of gn stage

ADD LW/SW

PoC – start of RR

A 7 or 9 stage pipeline, RR and RW take an entire stage



lw $1, 8($2) IF IF DE DE RR AL DM DM RW — PoP

add $4, $1, $3 IF IF DE DE DE DE DE DE RR AL W — PoC

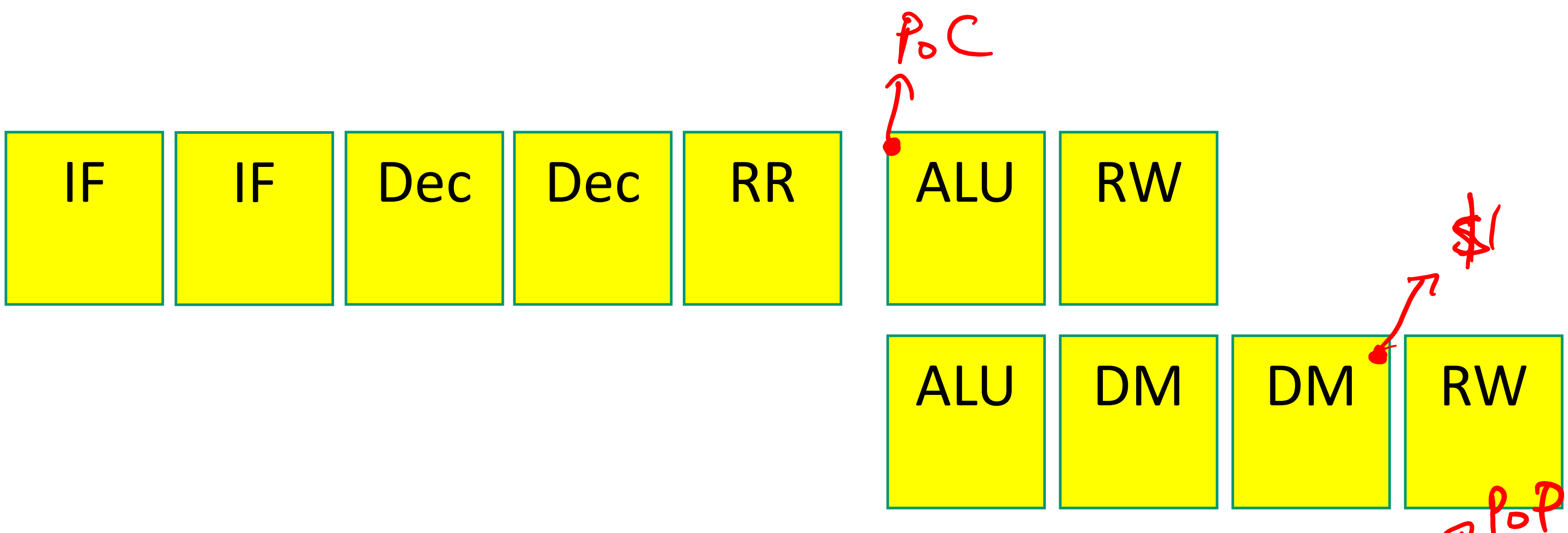
4 stalls

Problem 4 with byp

PoP – end of 2nd DM stage

PoC – start of ALU

A 7 or 9 stage pipeline, RR and RW take an entire stage



lw $1, 8($2)   IF IF DE DE RR AL DM DM RW

add $4, $1, $3   IF IF $\underbrace{\text{DE DE DE DE}}_{\text{2 stalls}}$ RR AL

PoC

Problem 4

Without bypassing: 4 stalls

IF:IF:DE:DE:RR:AL:DM:DM:RW

IF: IF :DE:DE:DE:DE: DE :DE:RR:AL:RW

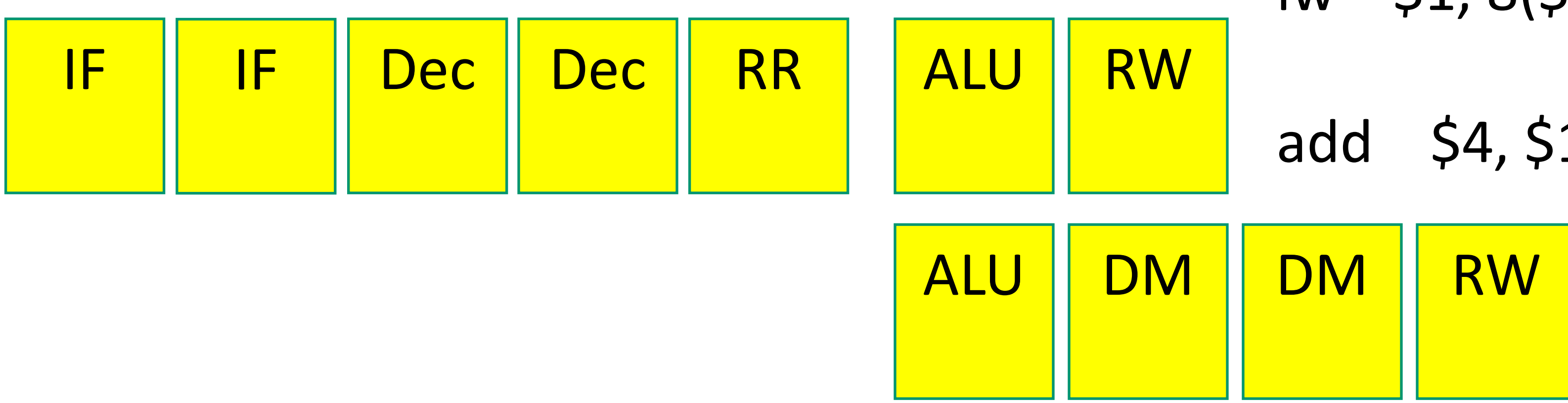
With bypassing: 2 stalls

IF:IF:DE:DE:RR:AL:DM:DM:RW

IF: IF :DE:DE:DE:DE: RR :AL:RW

lw $1, 8($2)

add $4, $1, $3



Control Hazards

BEQ $1, $2, 100 => Result known at end of 2nd cycle

PC+4 (NOT-TAKEN)

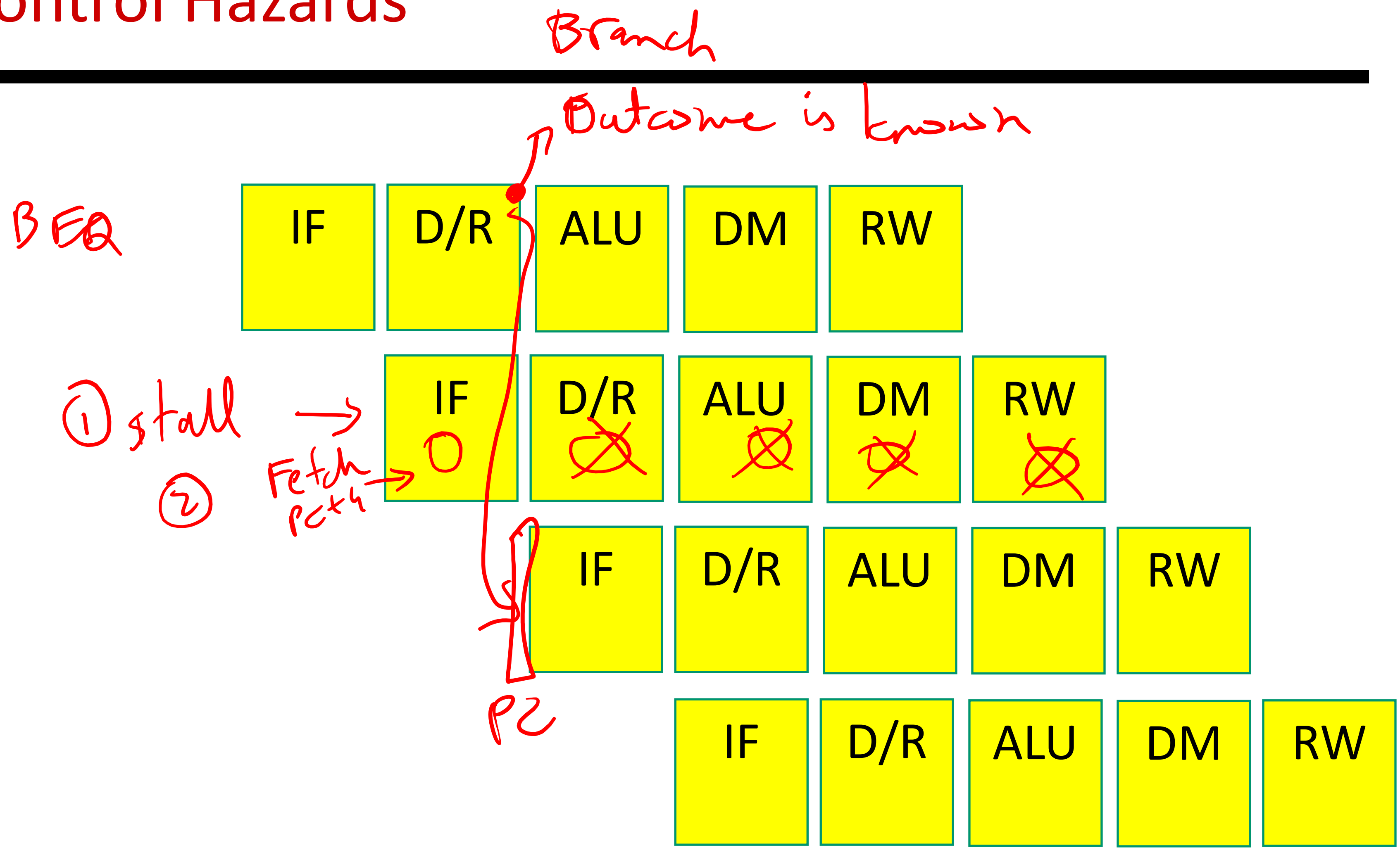
PC+4+100 (TAKEN)

- Simple techniques to handle control hazard stalls:
  - for every branch, introduce a stall cycle (note: every 6th instruction is a branch!)
  - assume the branch is not taken and start fetching the next instruction – if the branch is taken, need hardware to cancel the effect of the wrong-path instruction
  - fetch the next instruction (branch delay slot) and execute it anyway – if the instruction turns out to be on the correct path, useful work was done – if the instruction turns out to be on the wrong path, hopefully program state is not lost
  - make a smarter guess and fetch instructions from the expected target

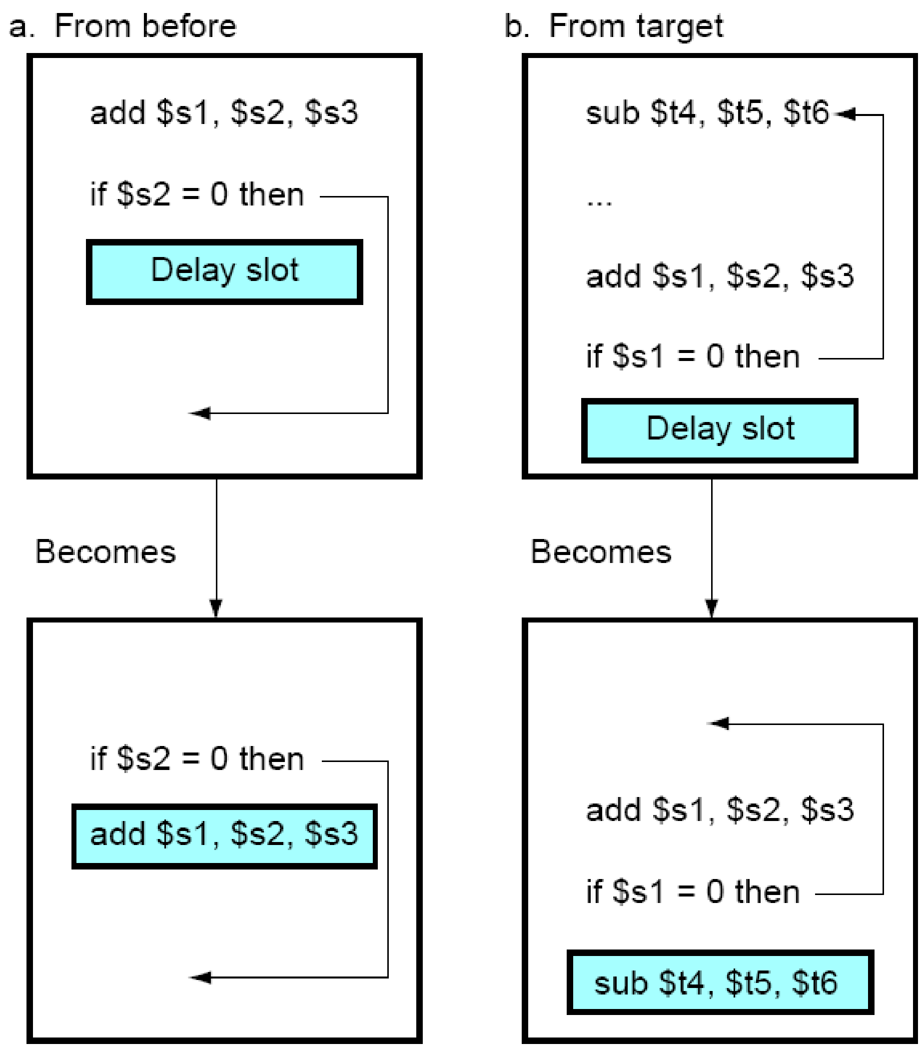
100 instrs
100 cycles
16 br
116 cycles

60% T 40% NT
100 instrs
16 br
9.6 br T → cost of cycle
109.6 cyc

Control Hazards



Branch Delay Slots



Source: H&P textbook