

Lecture 19: Pipelining

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

Example 2 – Bypassing

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

CYC-1	CYC-2	CYC-3	CYC-4	CYC-5	CYC-6	CYC-7	CYC-8
IF I1	IF I2	IF I3	IF I4	IF I5	IF	IF	IF
D/R	D/R I1	D/R I2	D/R I3	D/R I4	D/R	D/R	D/R
ALU	ALU	ALU I3 I3	ALU I4 I3	ALU I5 I3	ALU	ALU	ALU
DM	DM	DM	DM I1	DM I2	DM I3	DM	DM
RW	RW	RW	RW	RW I1	RW I2	RW I3	RW

Problem 0

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

● Point of Production

● Point of Consumption

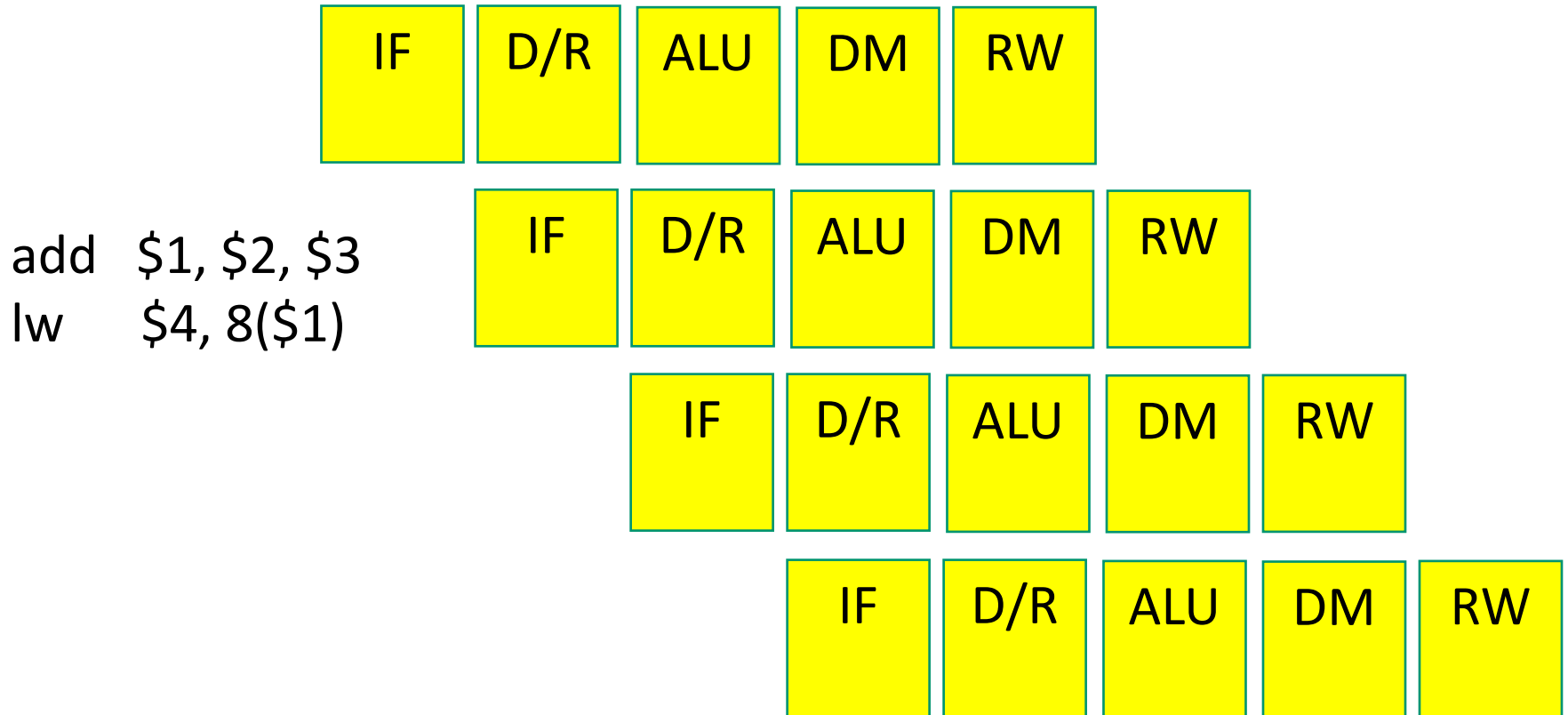
Without bypassing:

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

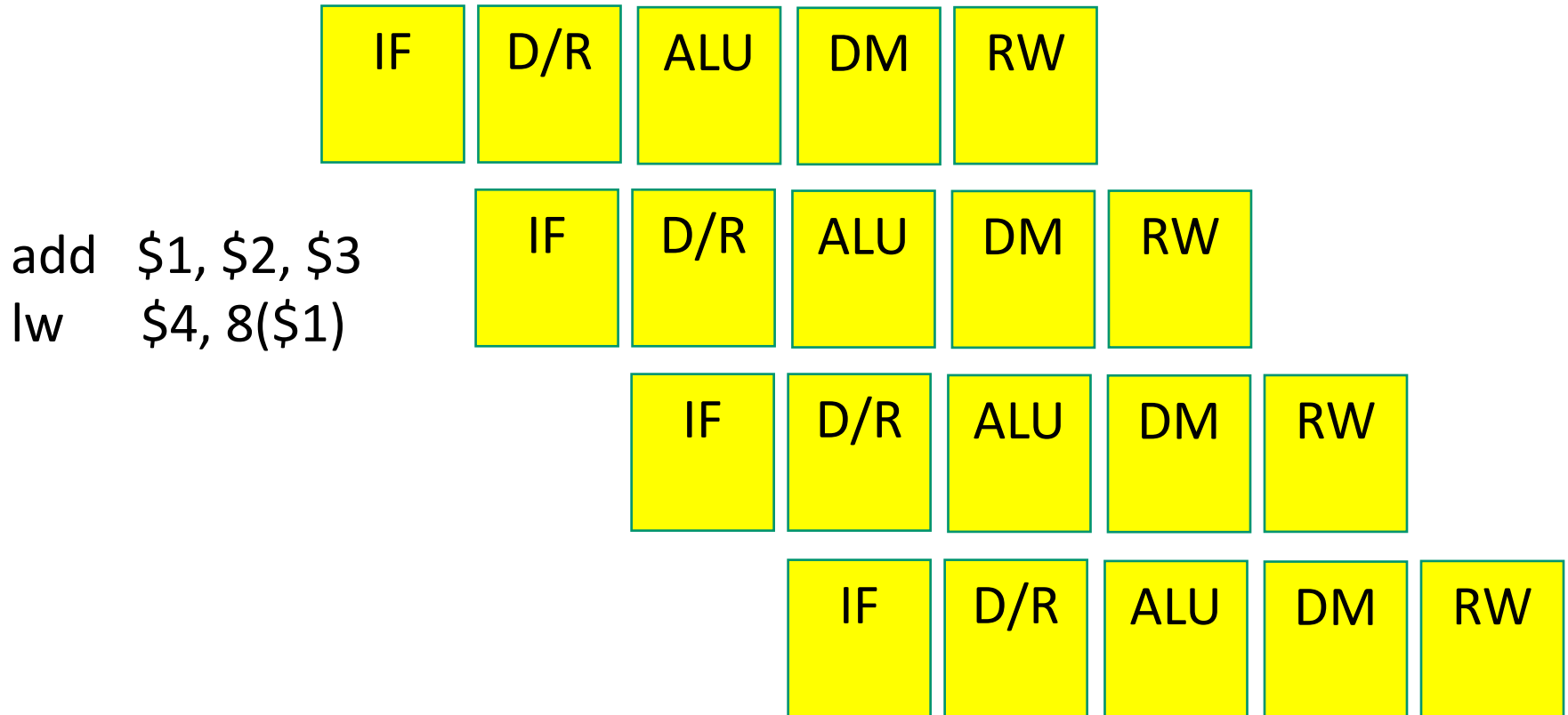
With bypassing:

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

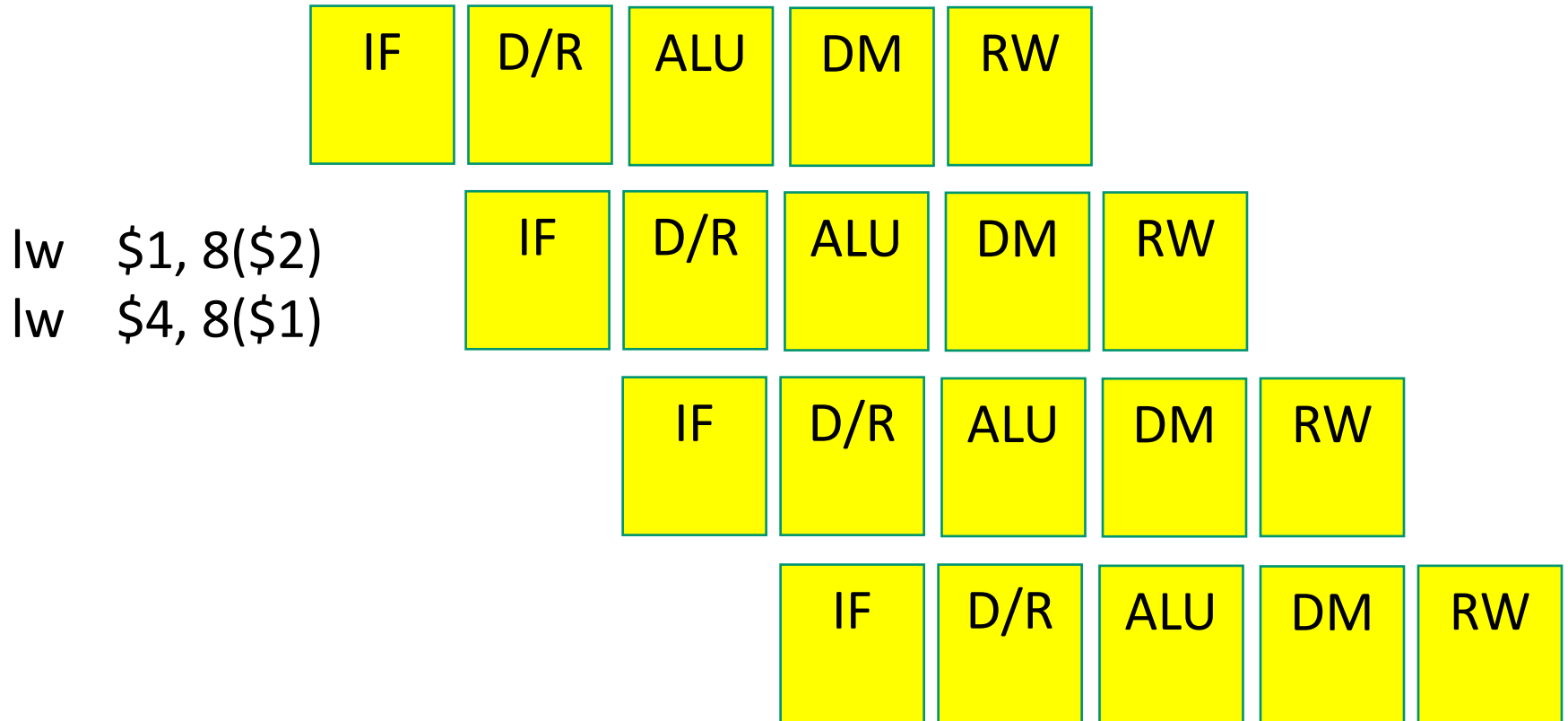
Problem 1 – No Byp



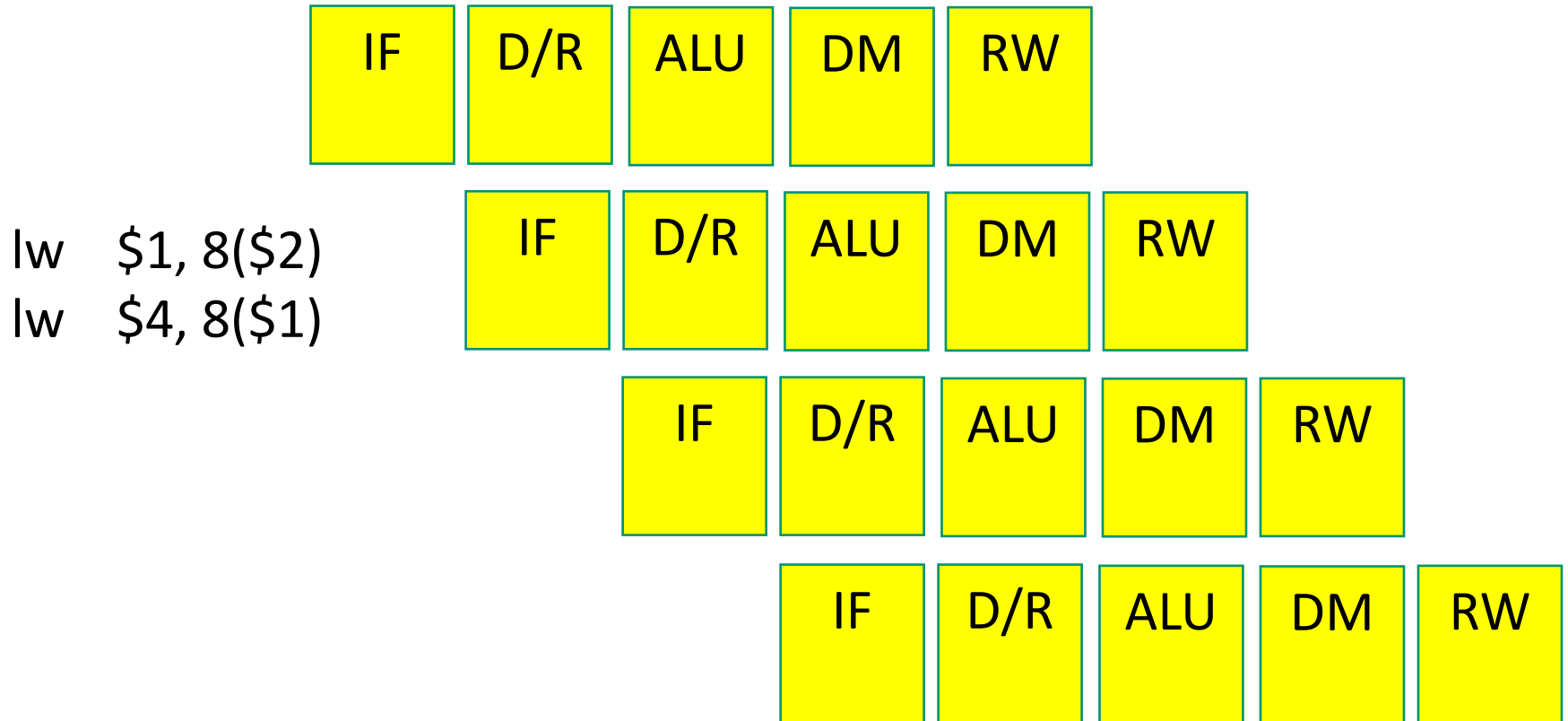
Problem 1 – with Byp



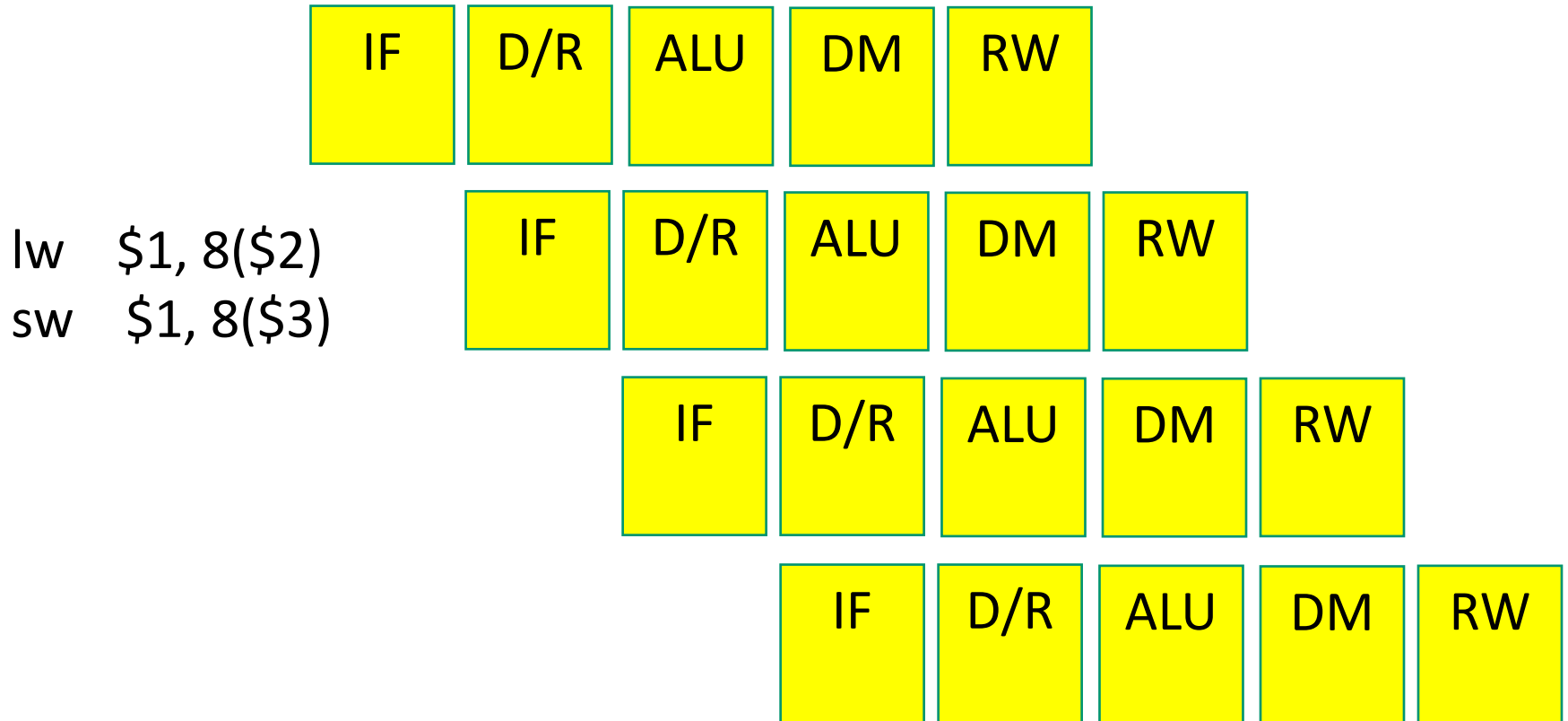
Problem 2 – no Byp



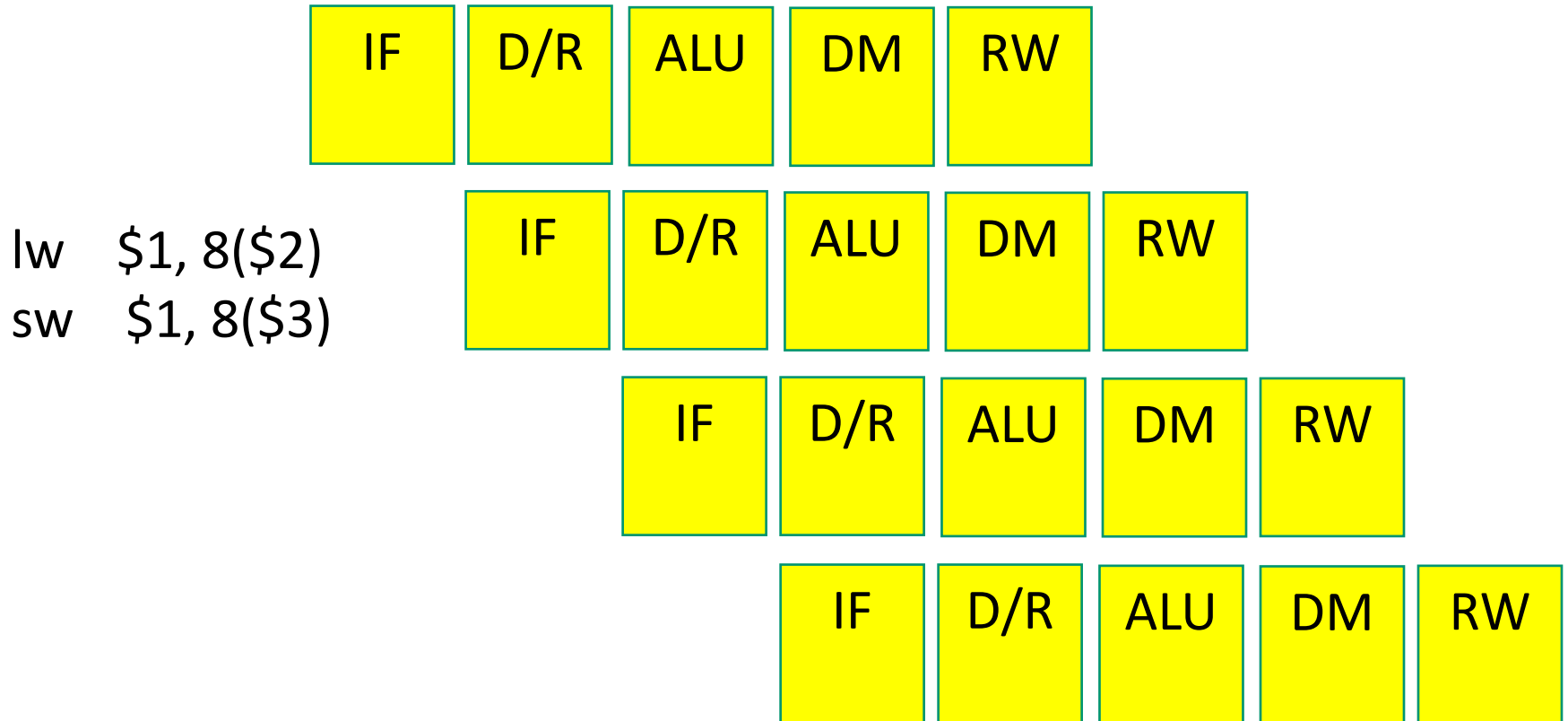
Problem 2 – with Byp



Problem 3 – no Byp

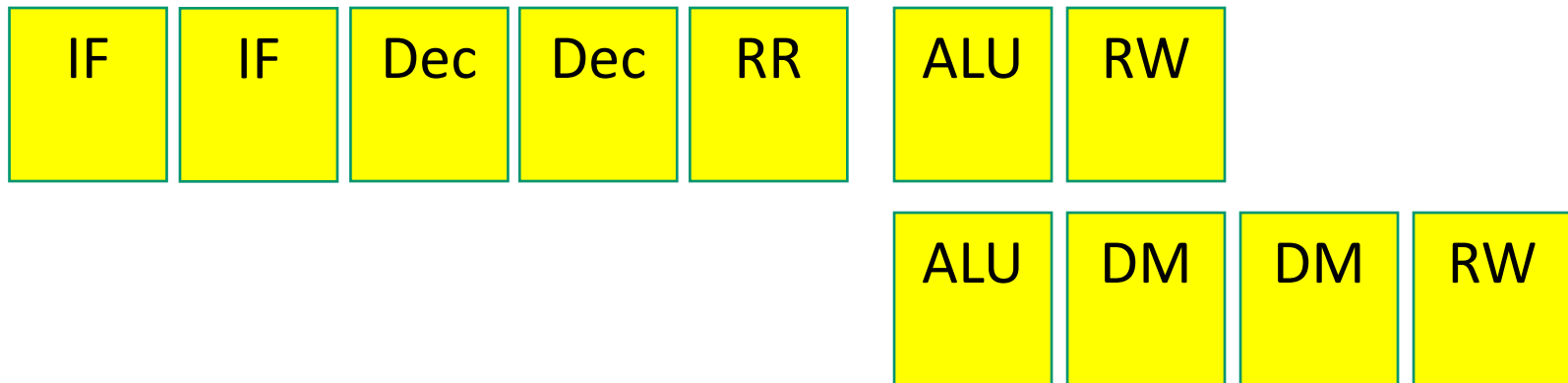


Problem 3 – with Byp



Problem 4 – no Byp

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

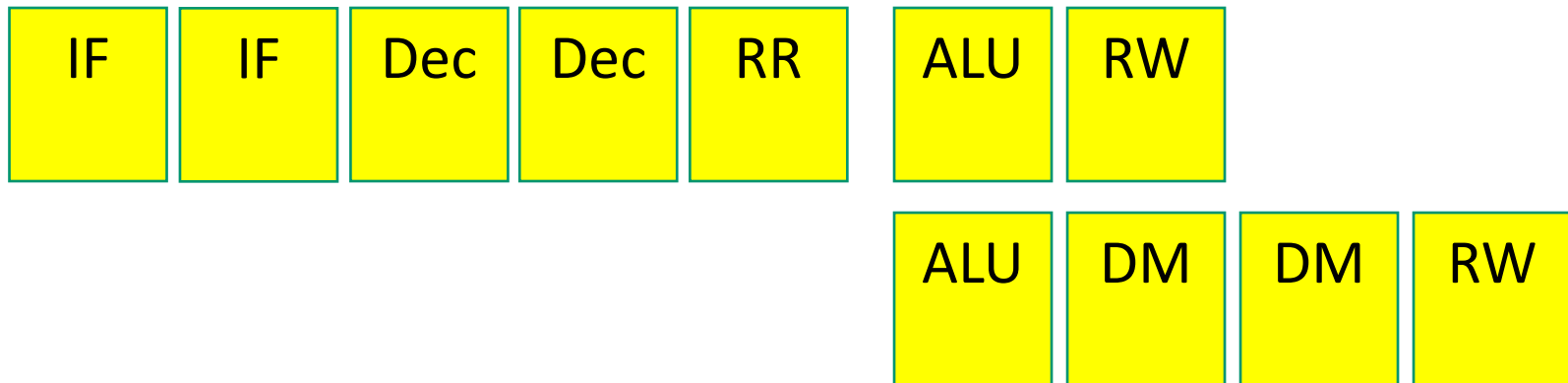


lw \$1, 8(\$2)

add \$4, \$1, \$3

Problem 4 – with Byp

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



lw \$1, 8(\$2)

add \$4, \$1, \$3

Problem 4

Without bypassing: 4 stalls

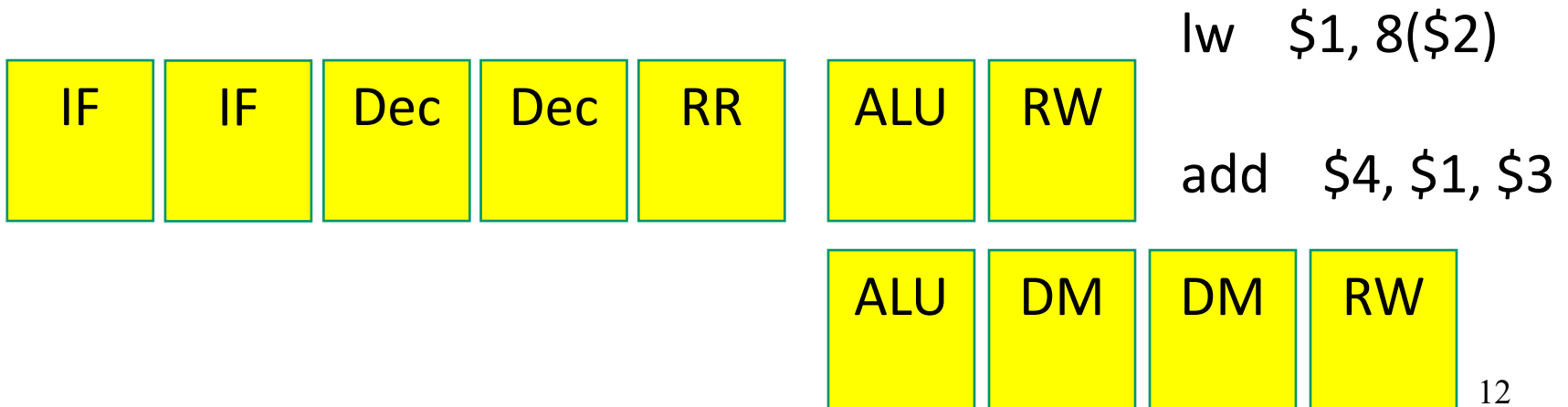
IF:IF:DE:DE:RR:AL:DM:DM:RW

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

With bypassing: 2 stalls

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

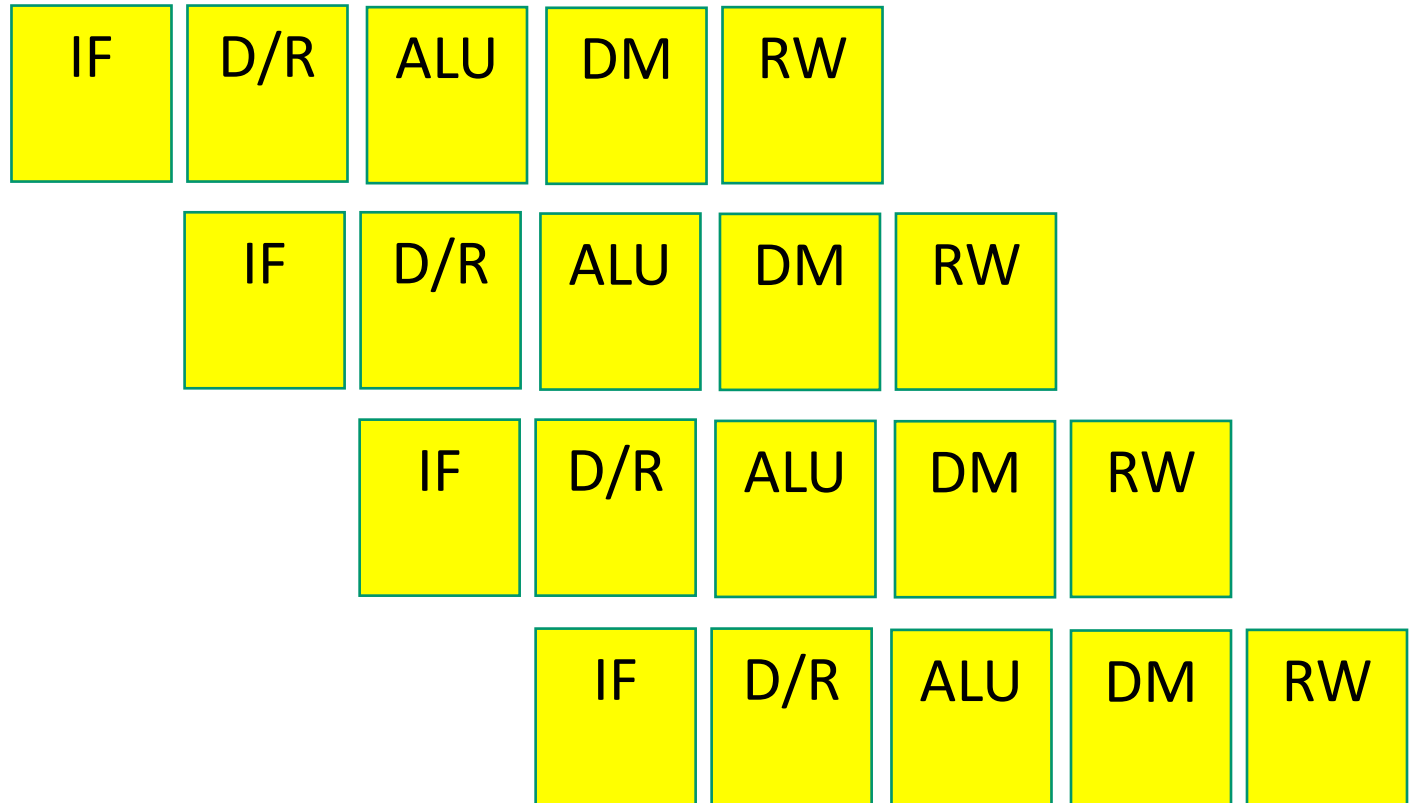
IF: IF :DE:DE:DE:DE: RR :AL:RW



Control Hazards

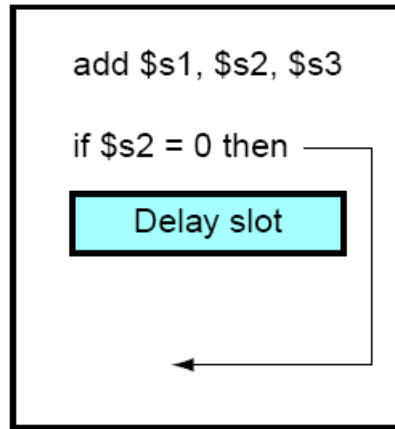
- Simple techniques to handle control hazard stalls:
 - for every branch, introduce a stall cycle (note: every 6th instruction is a branch!)
 - assume the branch is not taken and start fetching the next instruction – if the branch is taken, need hardware to cancel the effect of the wrong-path instruction
 - 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

Control Hazards

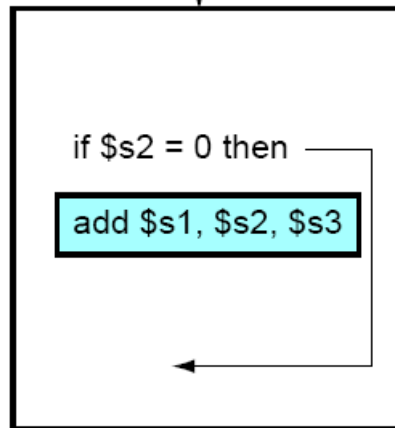


Branch Delay Slots

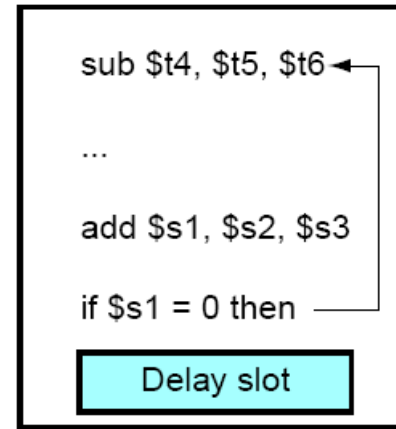
a. From before



Becomes



b. From target



Becomes

