Lecture 19: Pipelining

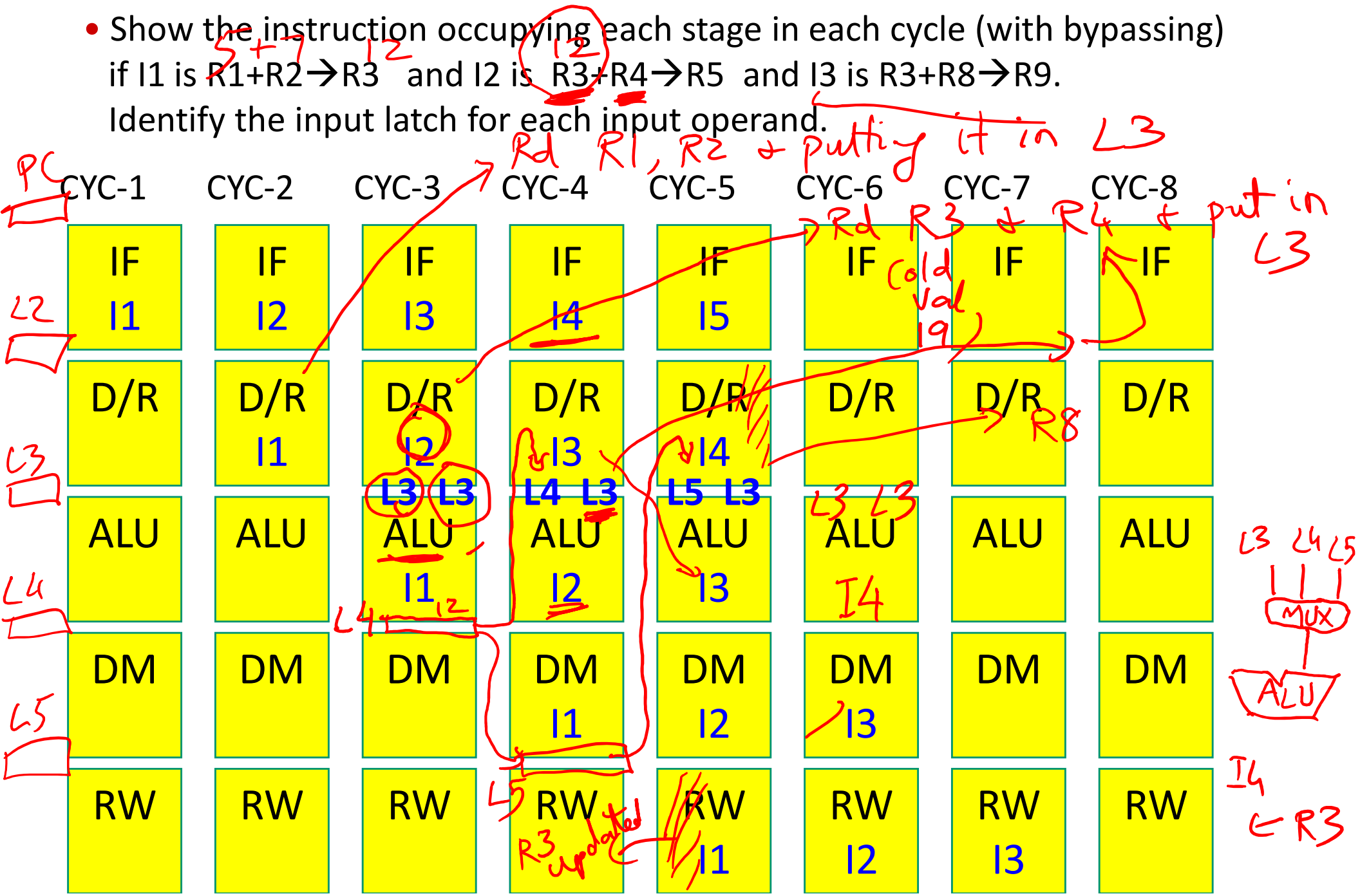
- Today's topics:
  - Data hazards and instruction scheduling
  - Control hazards

HW 7 posted
(due on Tue/Wed)

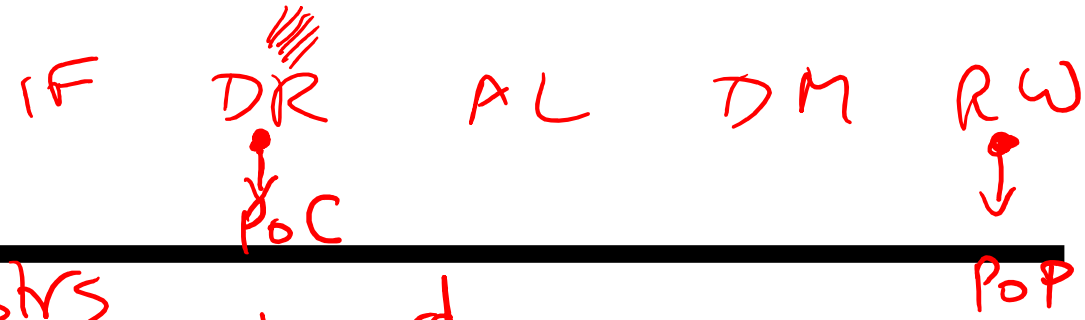
Example 2 – Bypassing

(HW 7)

- 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.



Problem 0



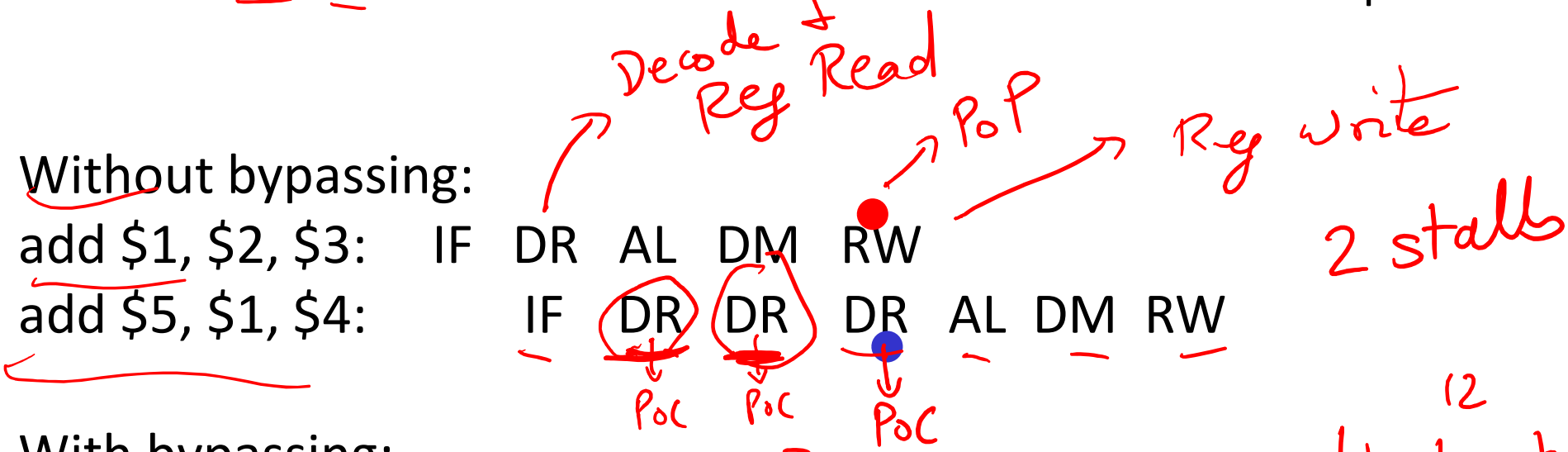
2 back-to-back instrs

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

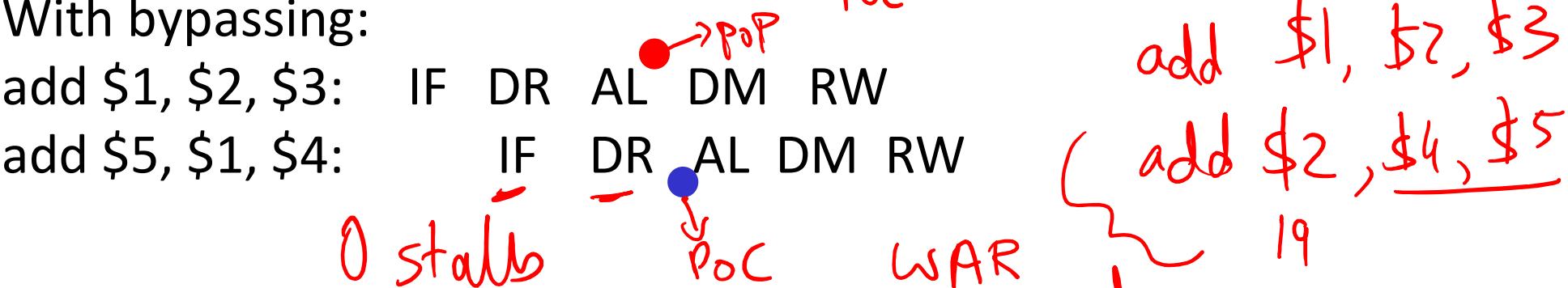
Prod } RAW hazard
Cons

- Point of Production
- Point of Consumption

Without bypassing:



With bypassing:



12

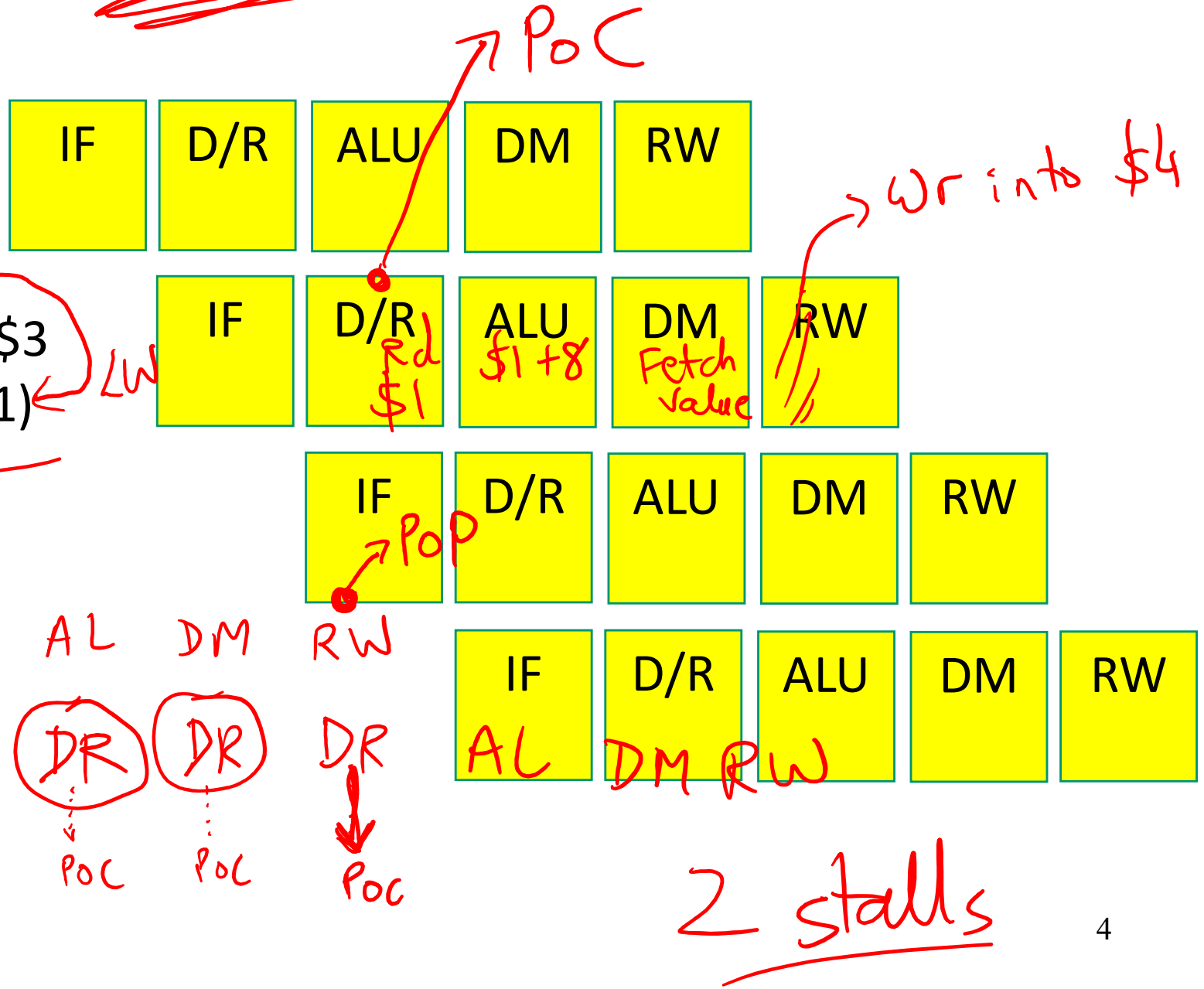
add $1, $2, $3

19

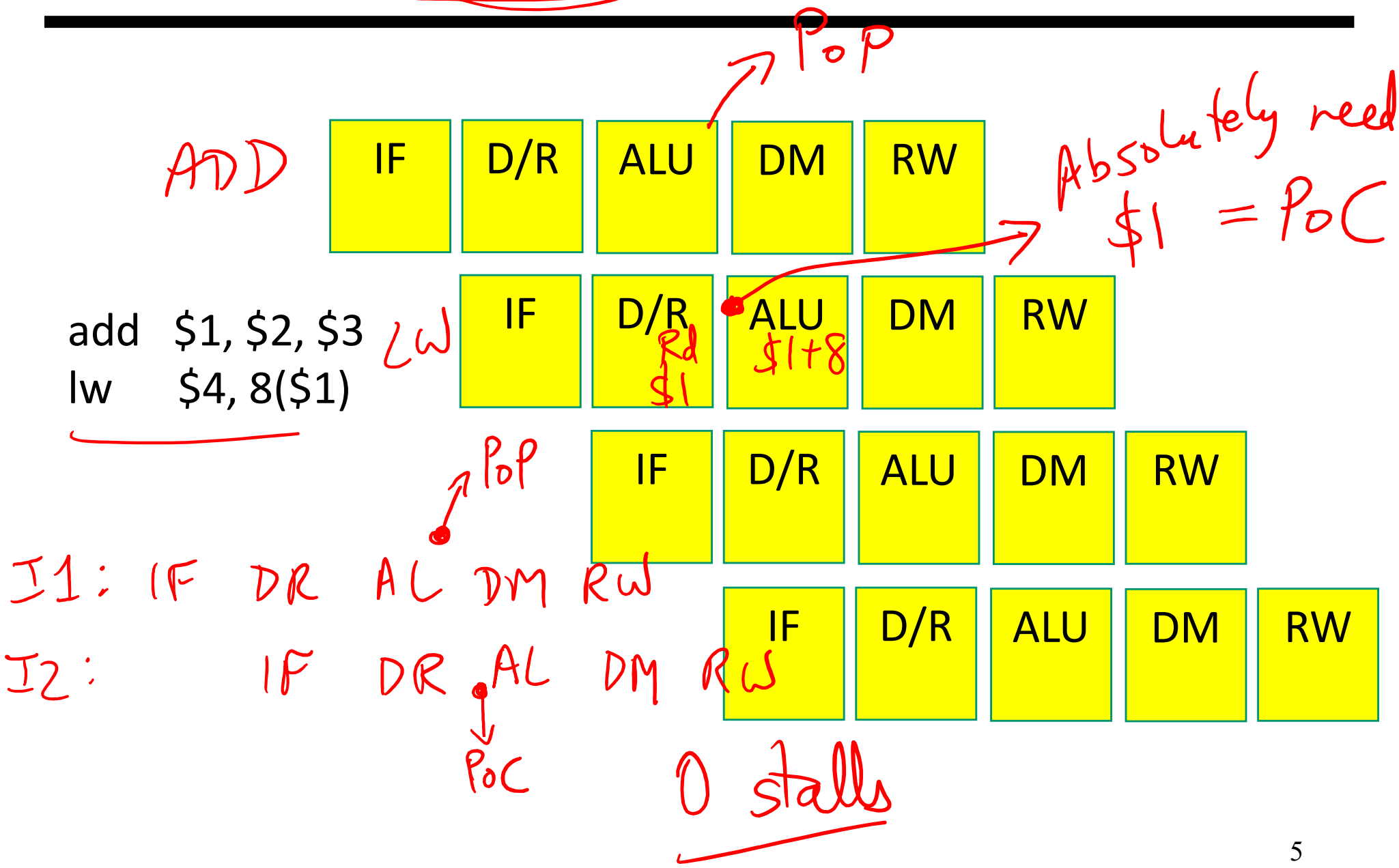
add $2, $4, $5

Problem 1 – No Byp

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

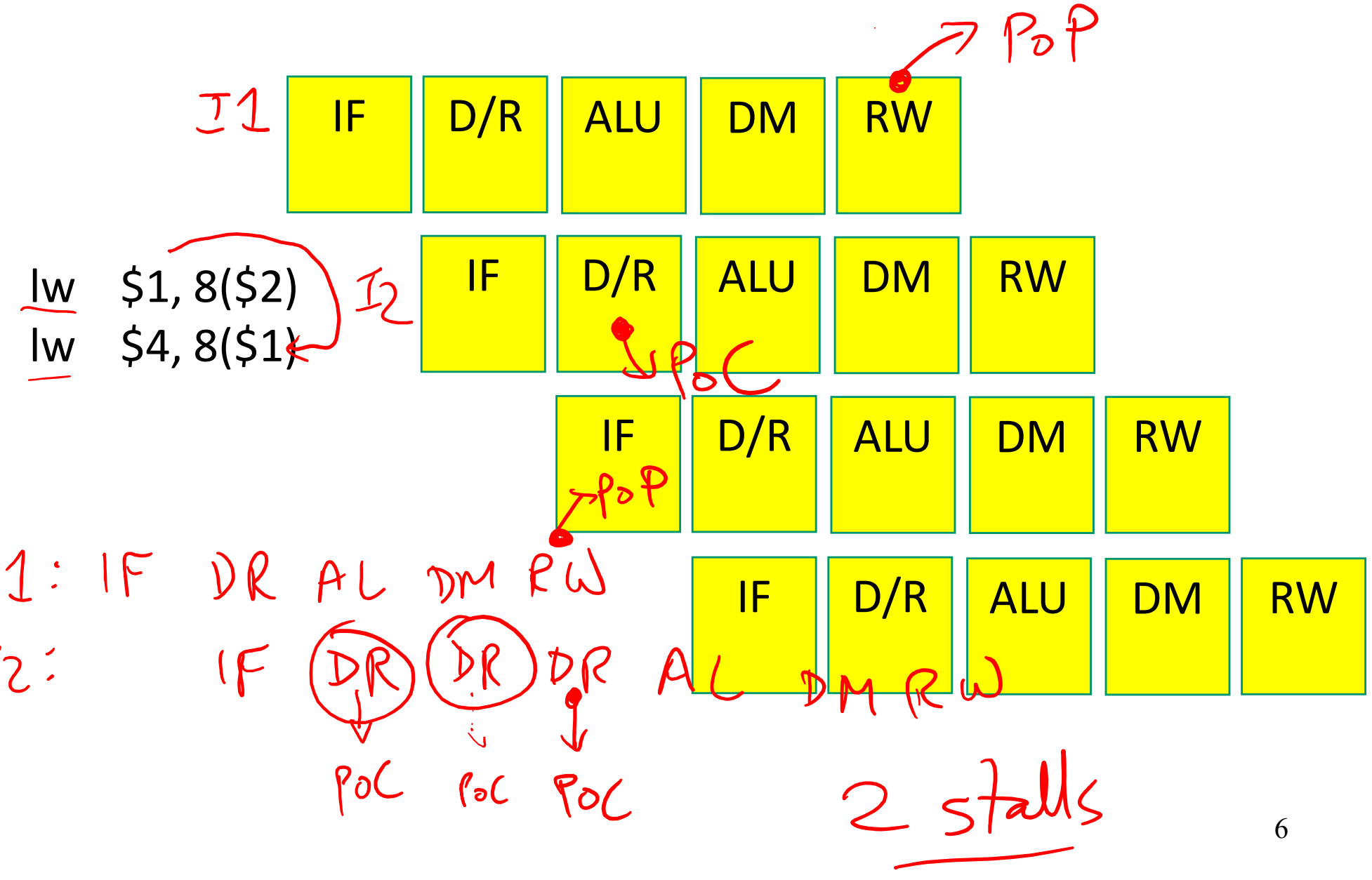


Problem 1 – with Byp

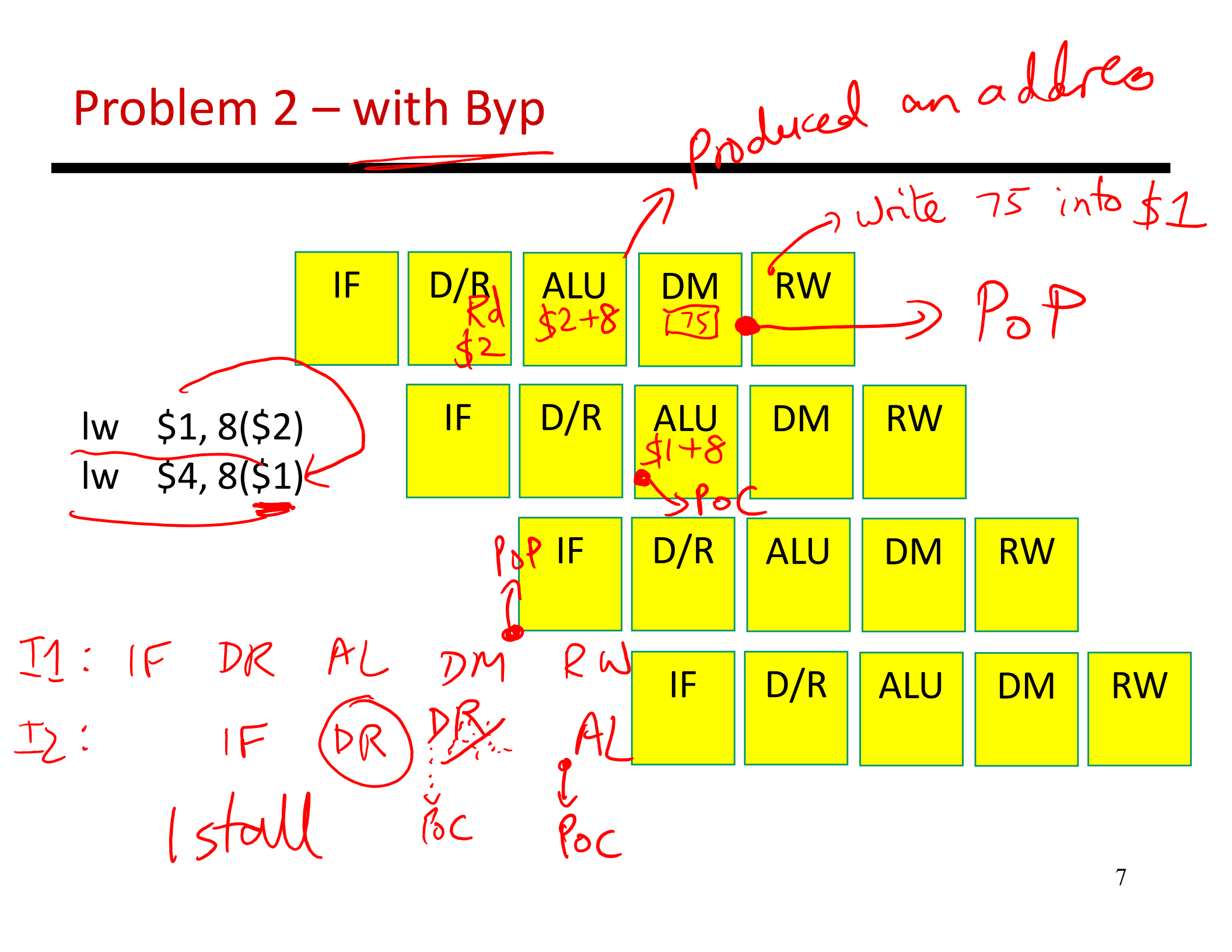


Problem 2 – no Byp

Wr into RF is PoP
Rd from RF is PoC

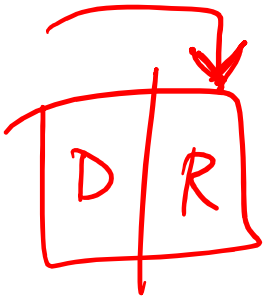


Problem 2 – with Byp



Problem 3 – no Byp

Only place to consume is

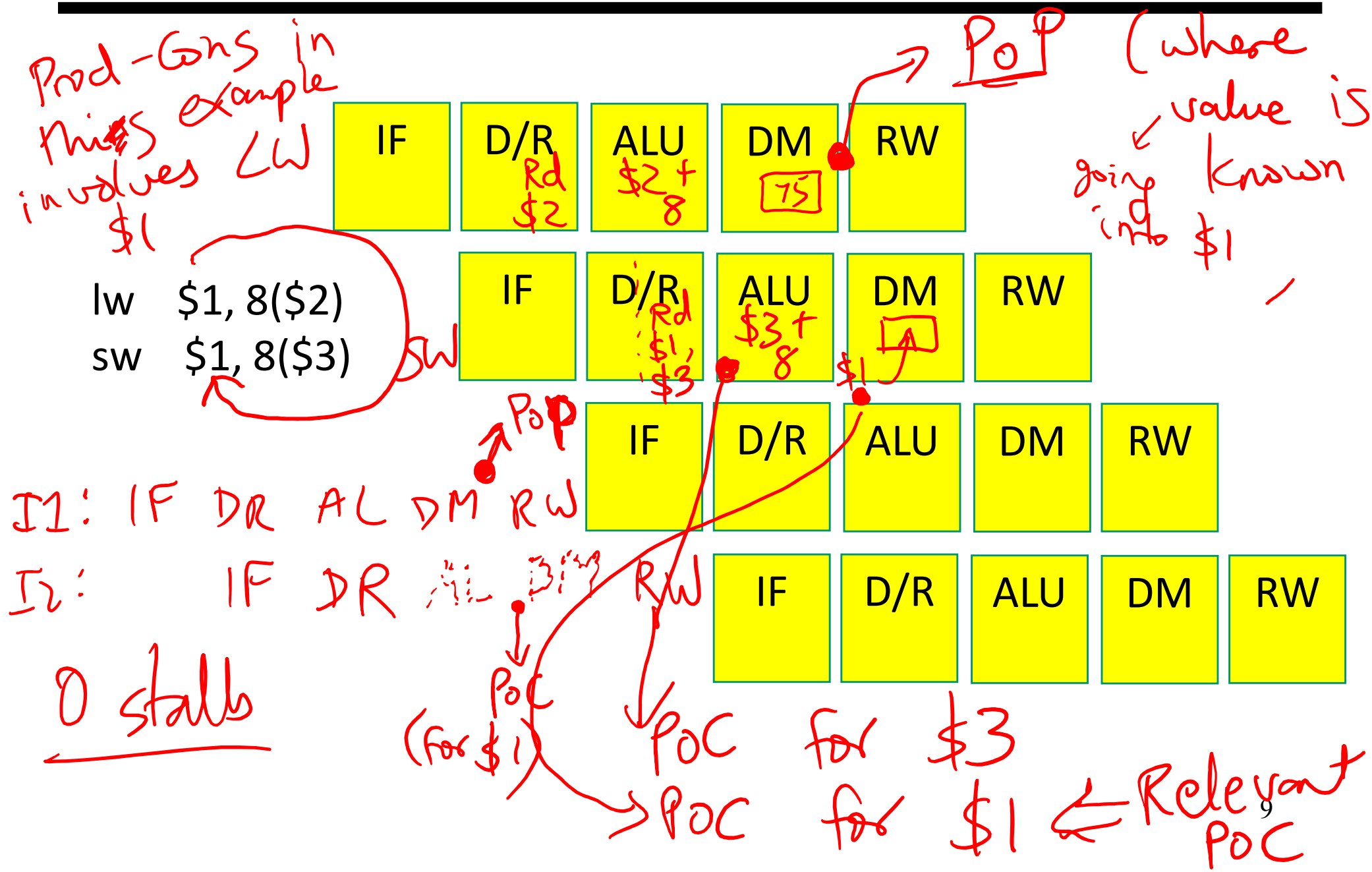


lw $1, 8($2)
sw $1, 8($3)

Exactly same as before

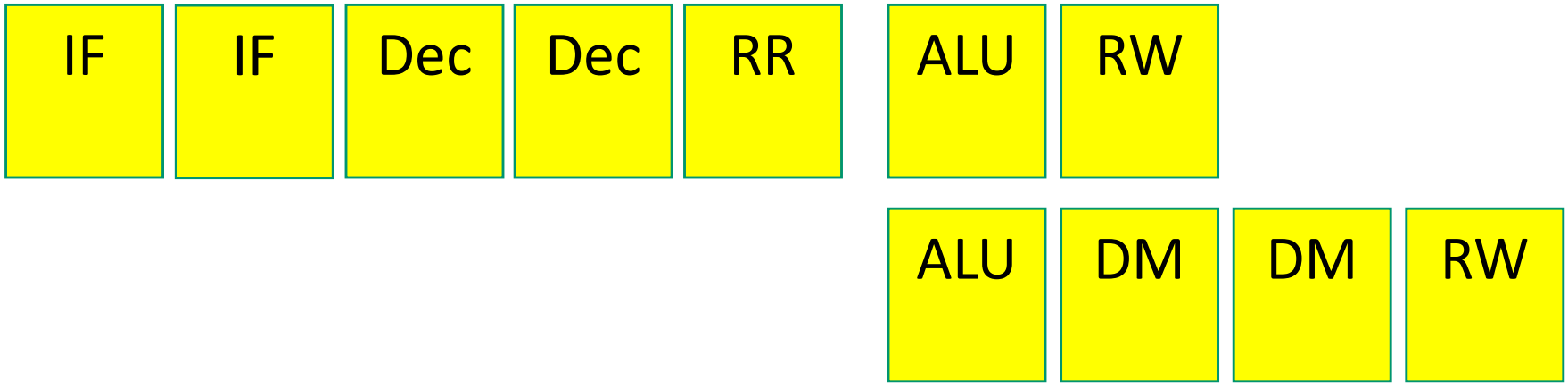
2 stalls

Problem 3 – with Byp



Problem 4 – no Byp

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

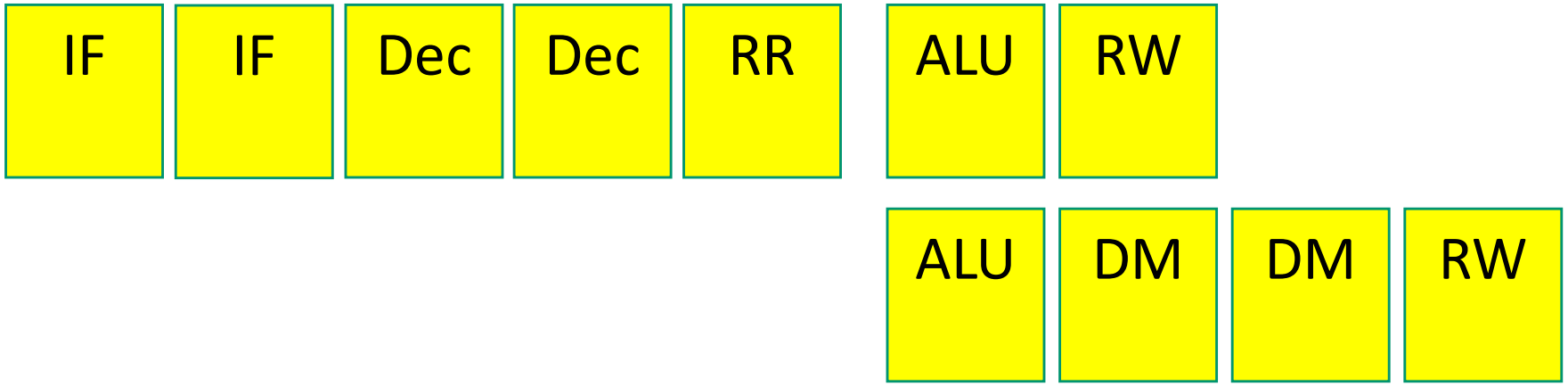


lw $1, 8($2)

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

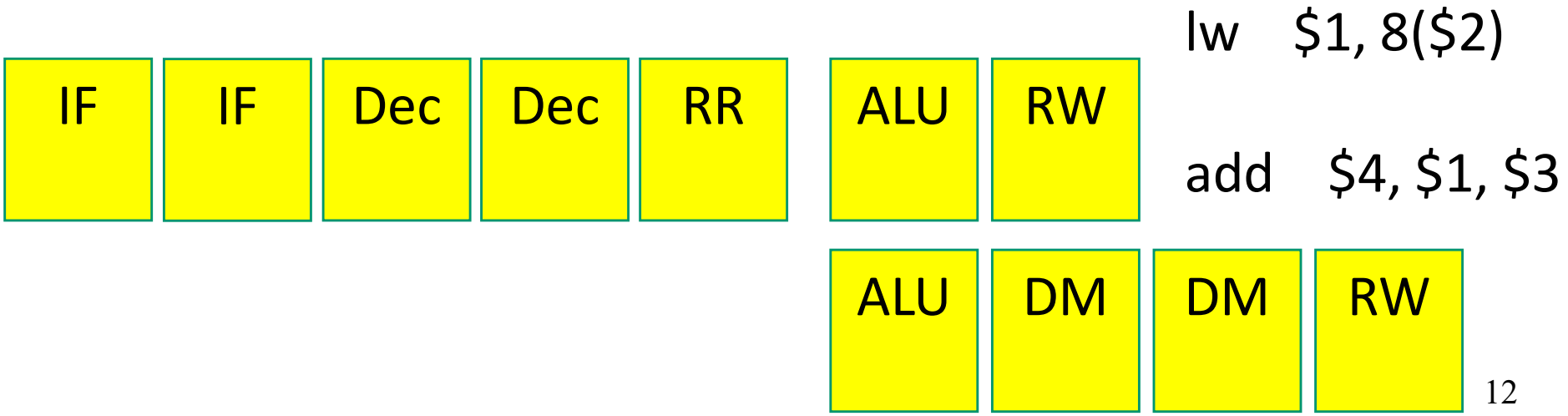
IF:IF:DE:DE:RR:AL:DM:DM:RW

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

With bypassing: 2 stalls

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

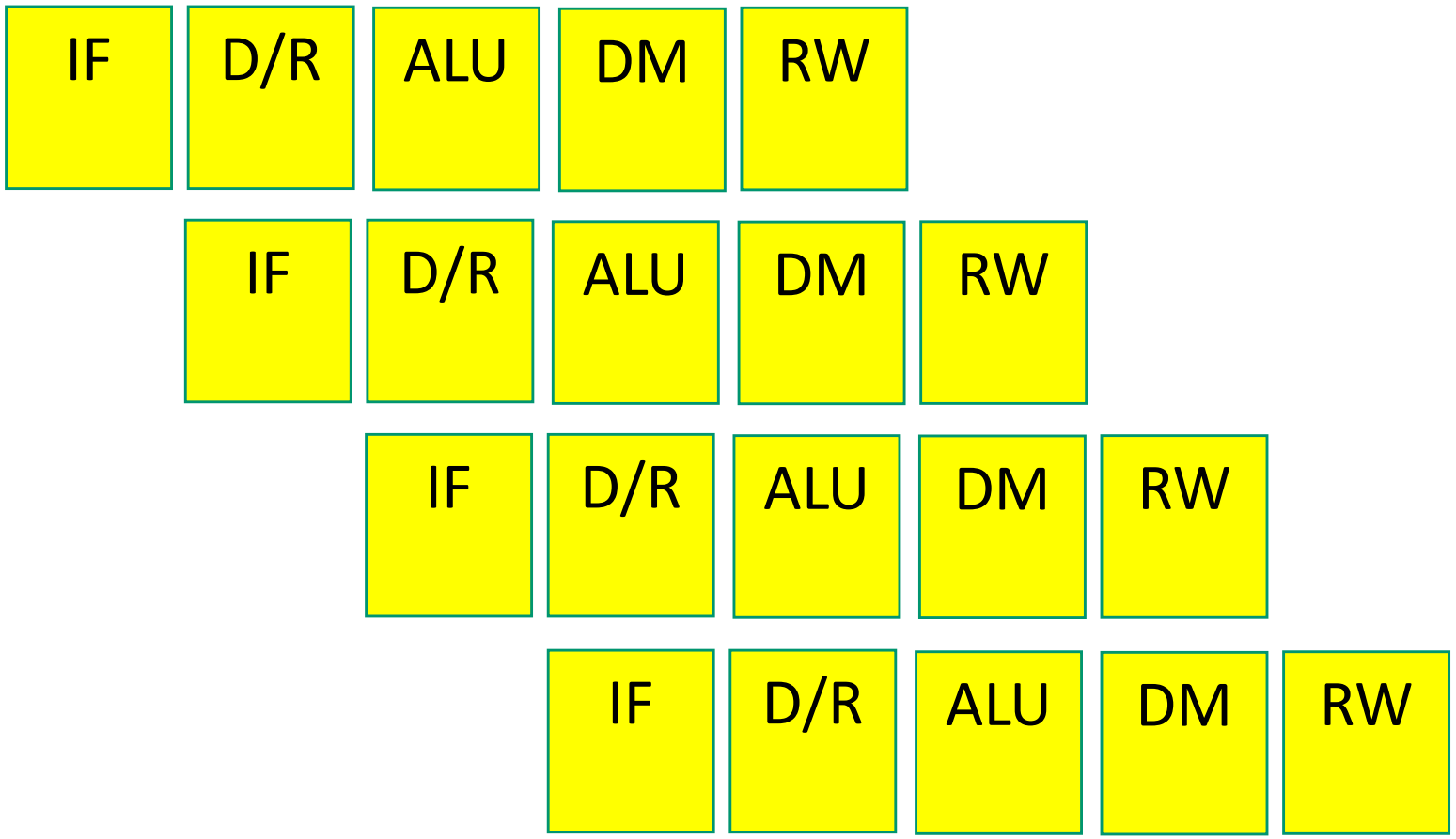
IF: IF :DE:DE:DE:DE: RR :AL:RW



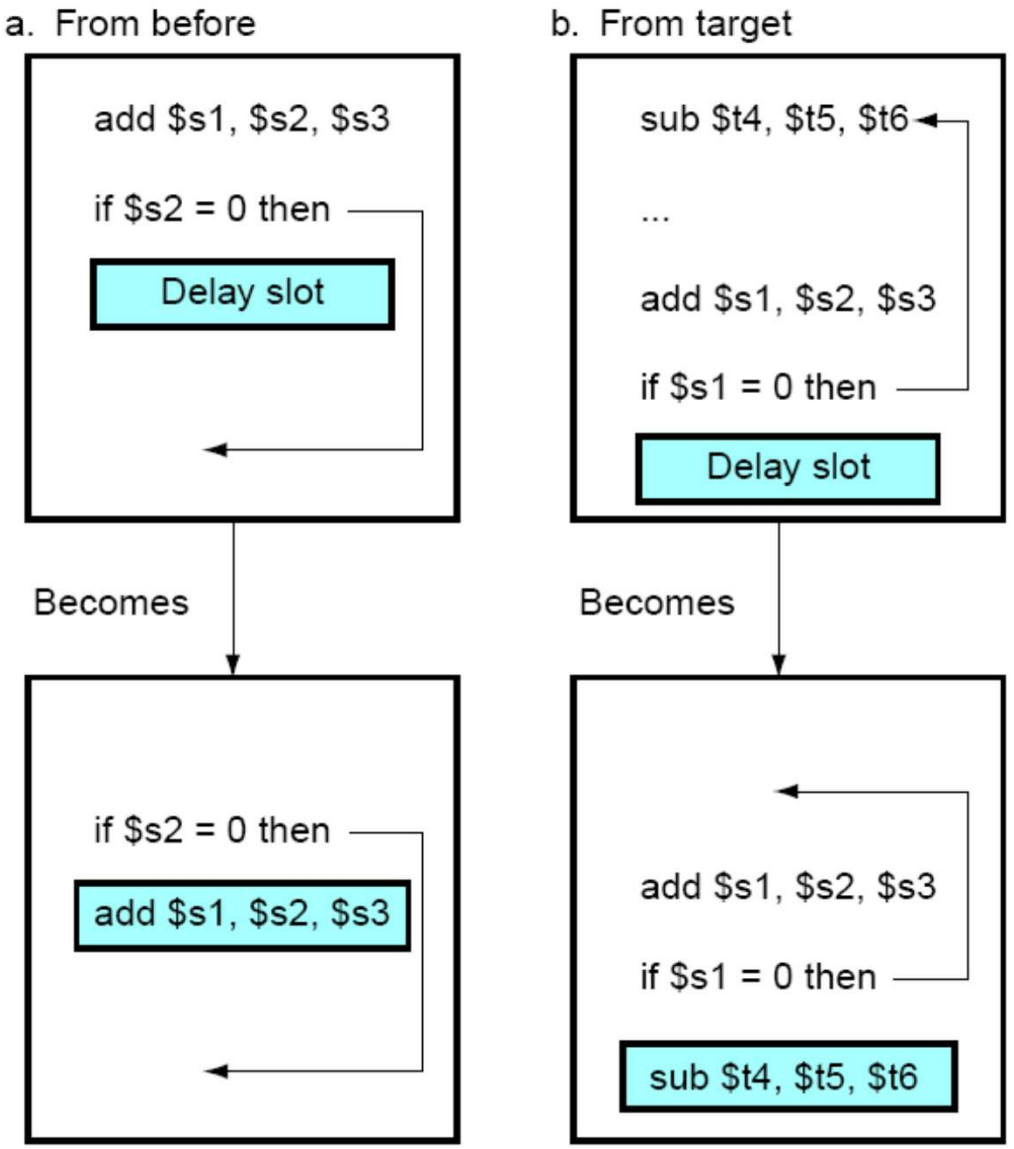
Control Hazards

- 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



Source: H&P textbook