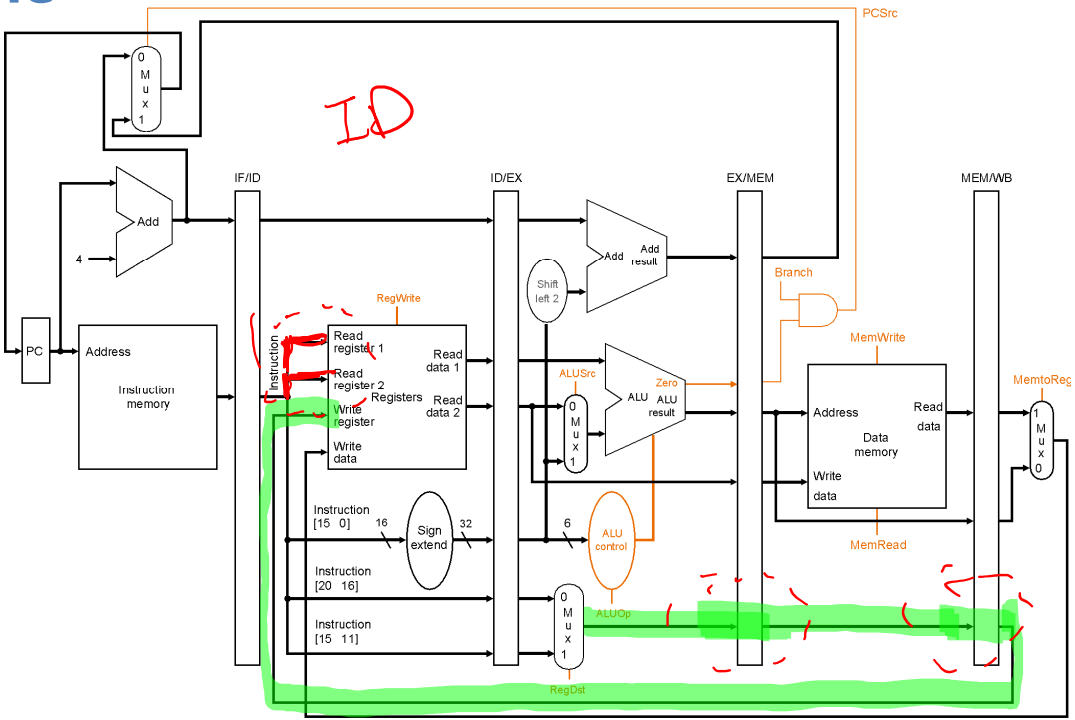
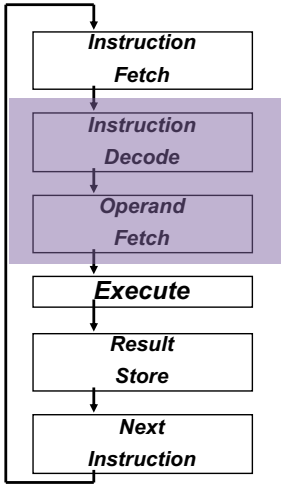
The Pipeline



- What comparisons tell us when to stall?

Stalling the Pipeline

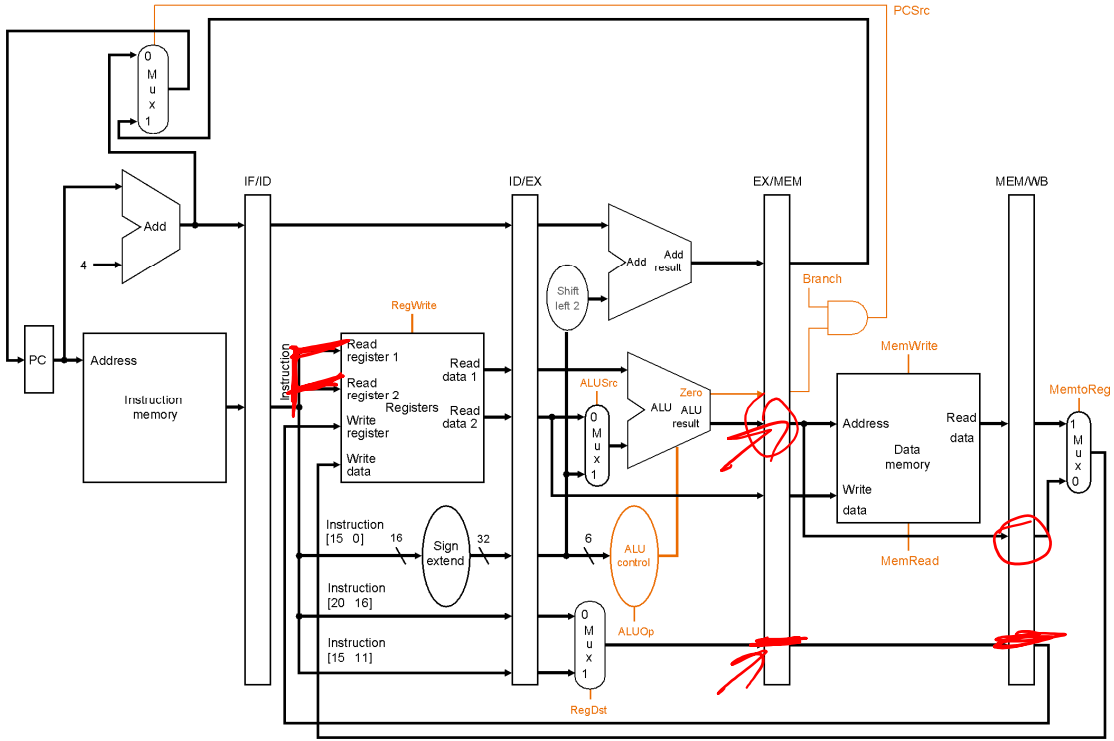
- Once we detect a hazard, then we have to be able to stall the pipeline (insert a *bubble*).
- Stalling the pipeline is accomplished by
  - (1) preventing the IF and ID stages from making progress
    - the ID stage because it cannot proceed until the dependent instruction completes
    - the IF stage because we do not want to lose any instructions.
  - (2) essentially, inserting "nops" in hardware



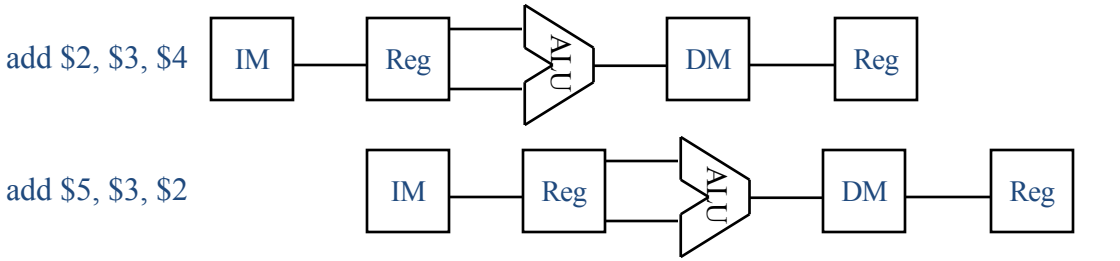
Stalling the Pipeline

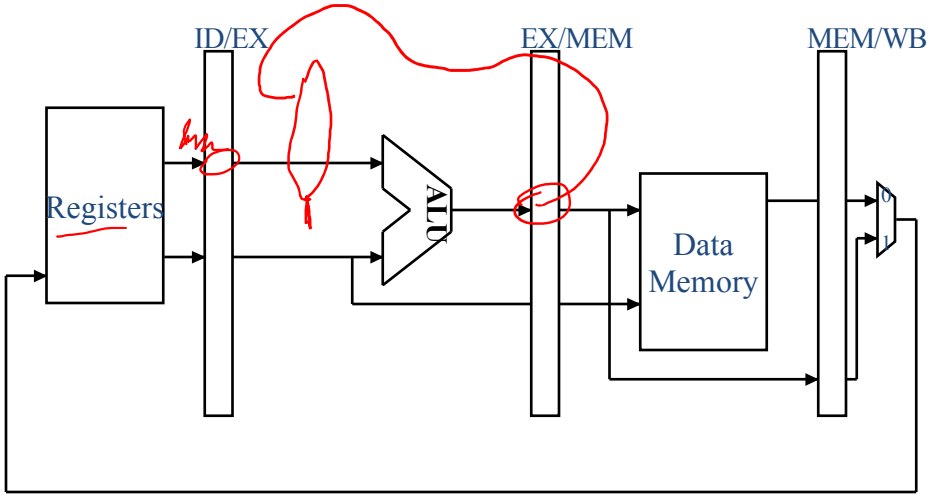
- Preventing the IF and ID stages from proceeding
  - don't write the PC (PCWrite = 0)
  - don't rewrite IF/ID register (IF/IDWrite = 0)
- Inserting "nops"
  - set all control signals propagating to EX/MEM/WB to zero

Can we do better? How else might we deal with (some?) data hazards?

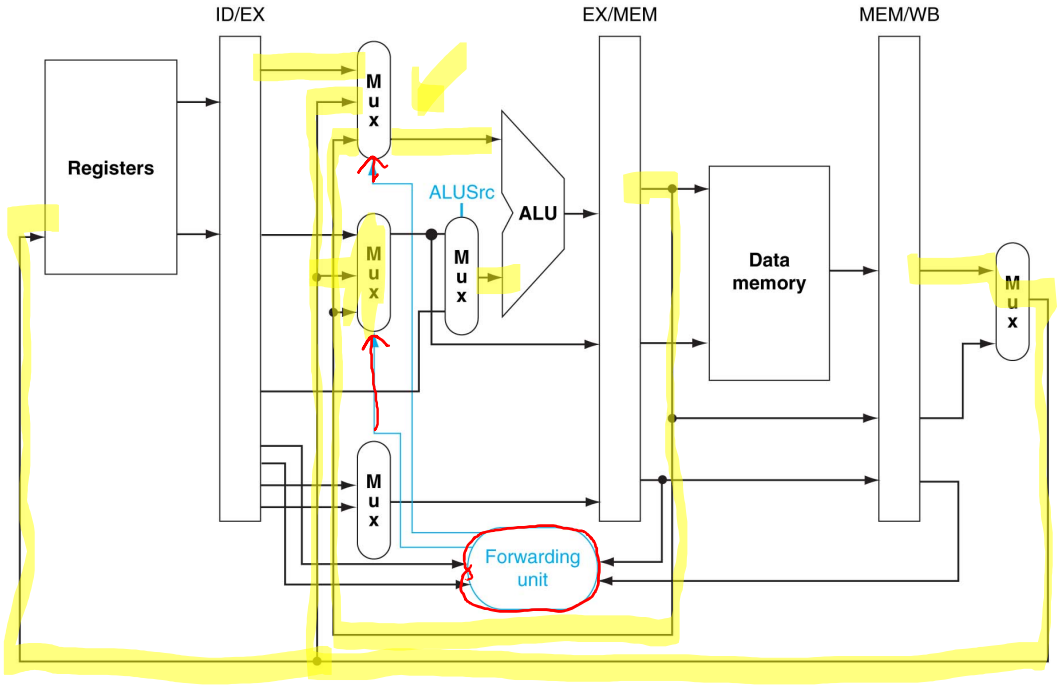


Reducing Data Hazards Through Forwarding





Reducing Data Hazards Through Forwarding

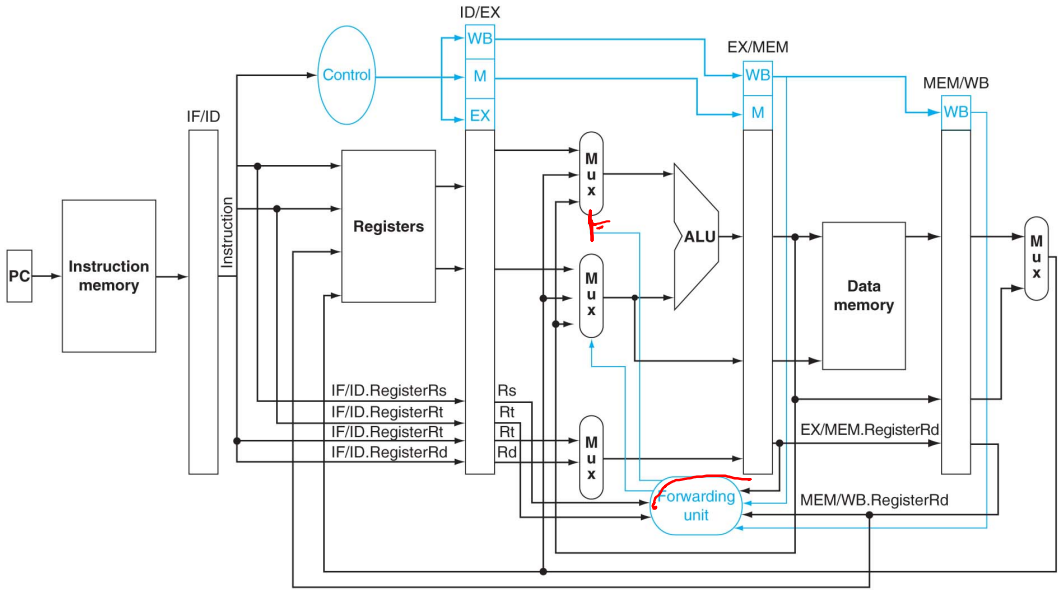


Reducing Data Hazards Through Forwarding

EX Hazard: *(similar for the MEM stage)*

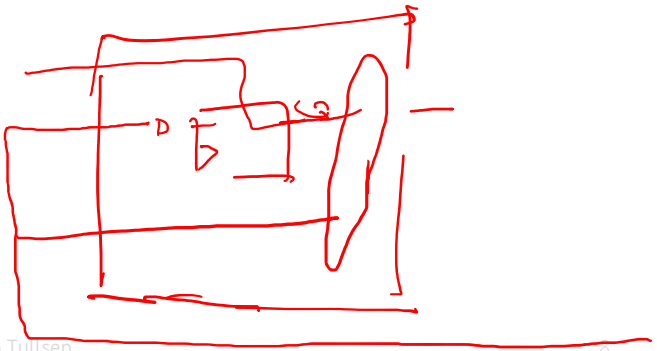
if (EX/MEM.RegWrite
and (EX/MEM.RegisterRd != 0)
and (EX/MEM.RegisterRd = ID/EX.RegisterRs)) ForwardA = 10

if (EX/MEM.RegWrite
and (EX/MEM.RegisterRd != 0)
and (EX/MEM.RegisterRd = ID/EX.RegisterRt)) ForwardB = 10

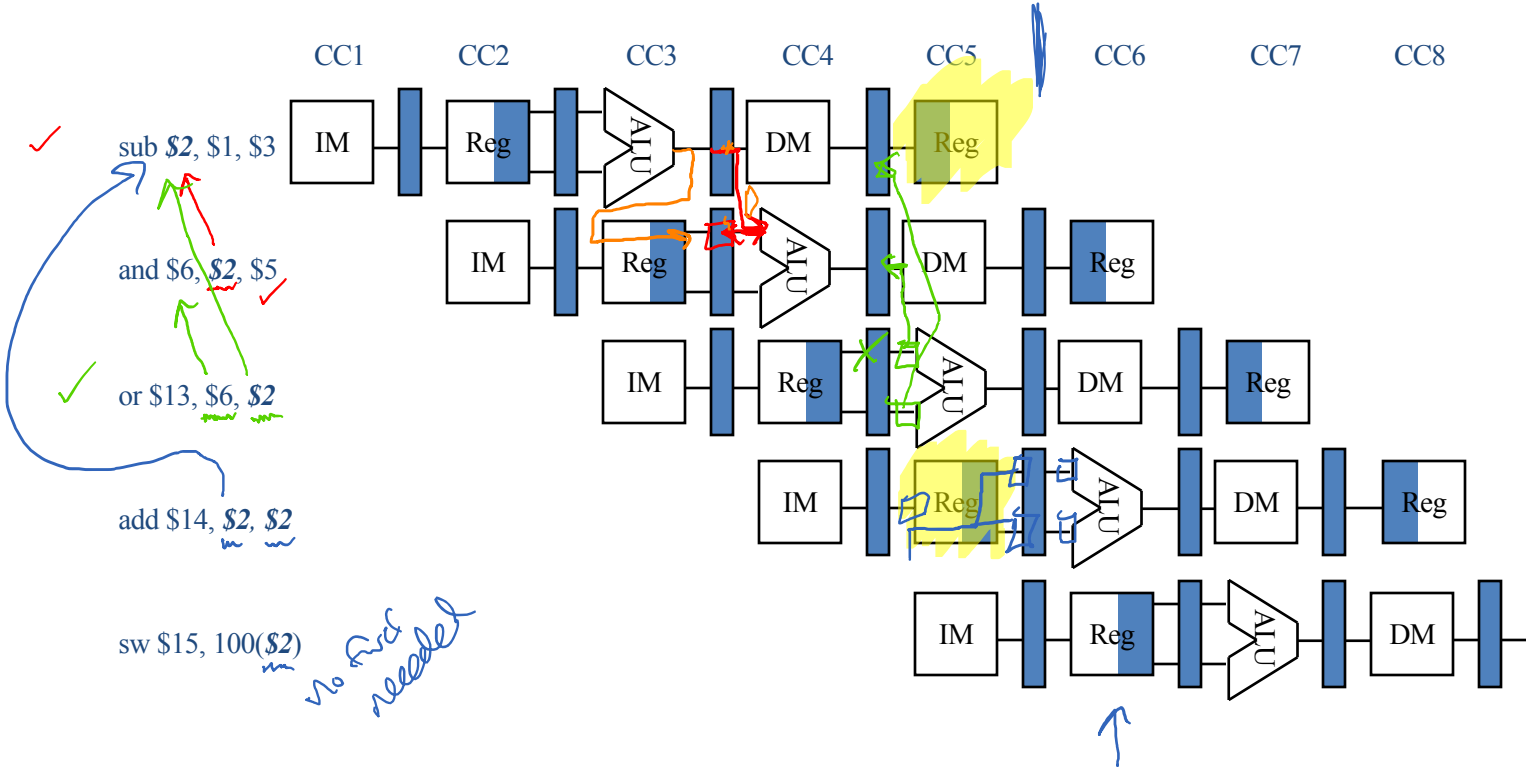


Data Forwarding

- The Previous Data Path handles two types of data hazards
  - EX hazard
  - MEM hazard
- We assume the register file handles the third (WB hazard)
  - if the register file is asked to read and write the same register in the same cycle, we assume that the reg file allows the write data to be forwarded to the output
  - We're still going to call that forwarding.

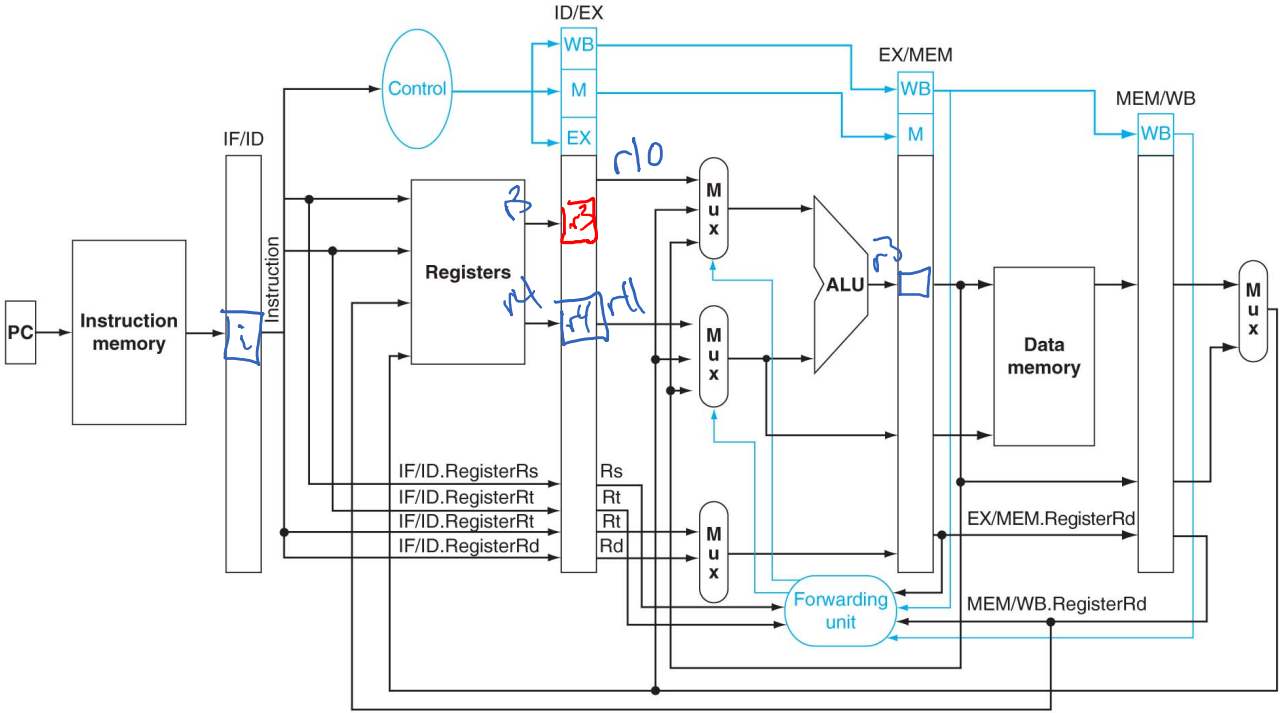


Eliminating Data Hazards via Forwarding



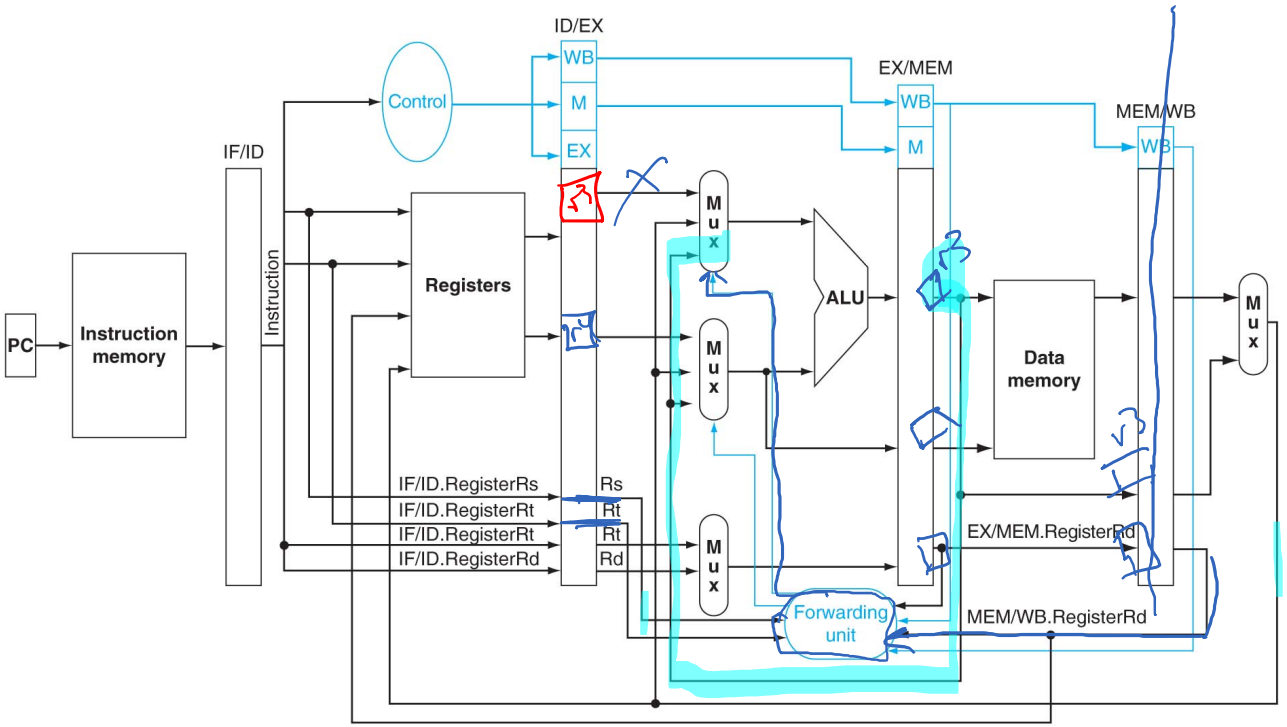
Forwarding in Action

add $1, $12, $3 **sub $12, $3, $4** **add $3, $10, $11** Memory Access Write Back



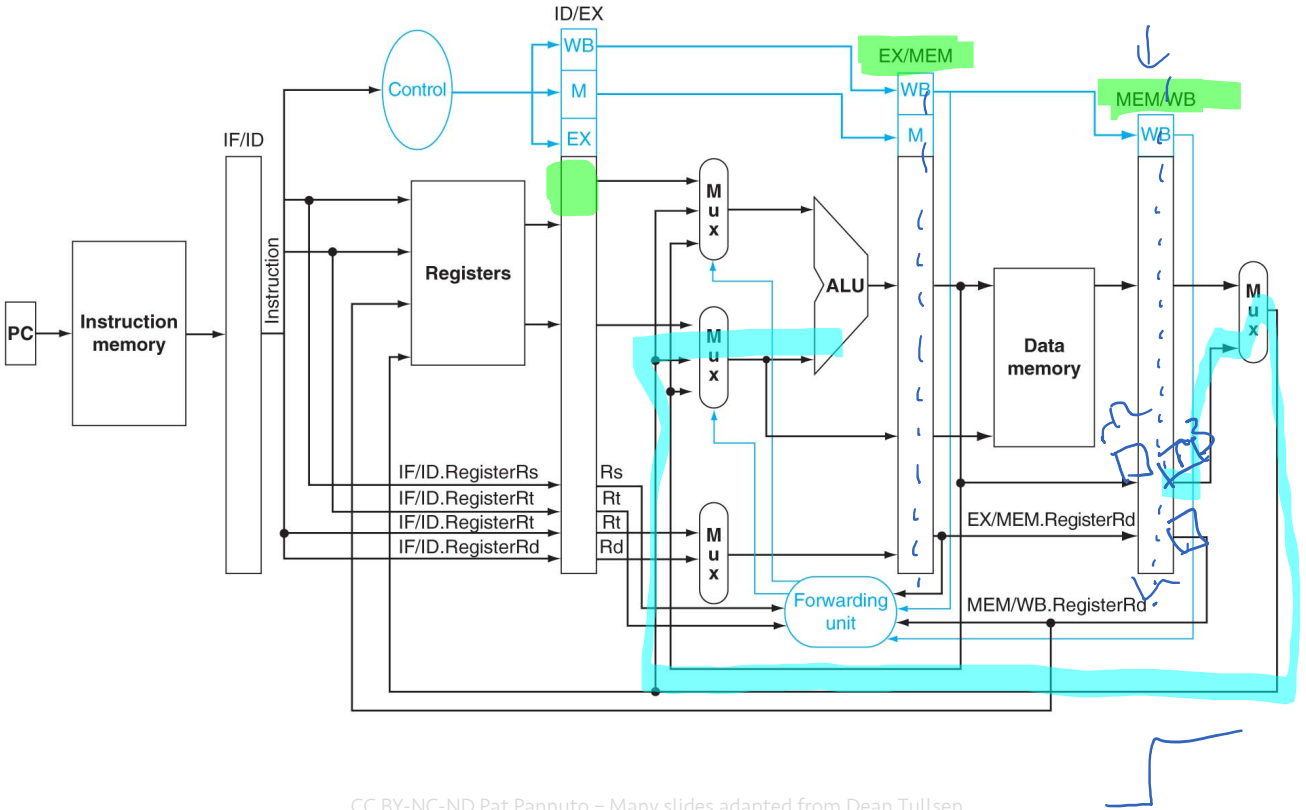
Forwarding in Action

Instruction Fetch | **add $1, $12, $3** | **sub $12, $3, $4** | **add $3, $10, $11** | Write Back

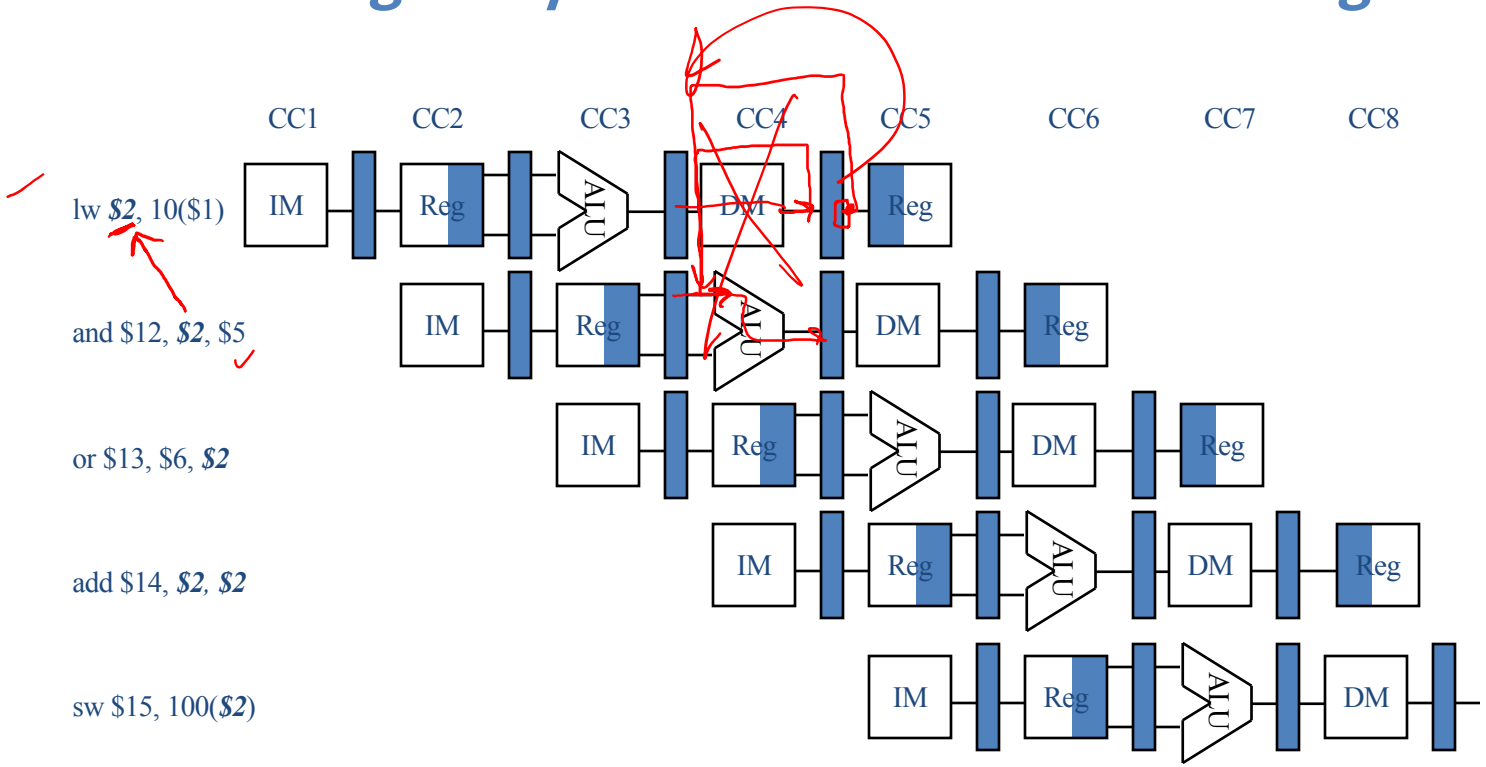


Forwarding in Action

Instruction Fetch | Instruction Decode | **add $1, $12, $3** | **sub $12, $3, $4** | **add $3, $10, $11**



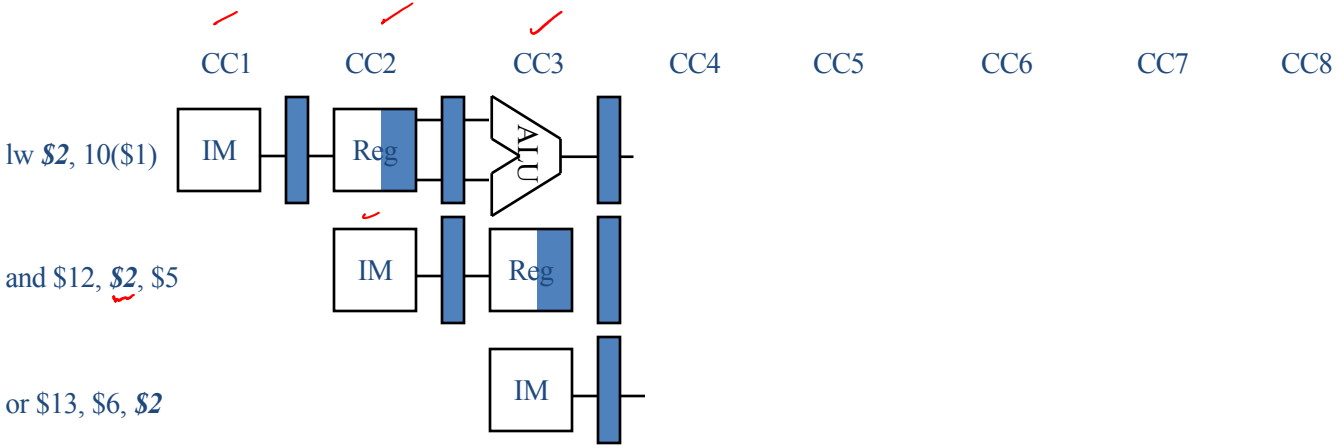
Eliminating Every Data Hazard via Forwarding?



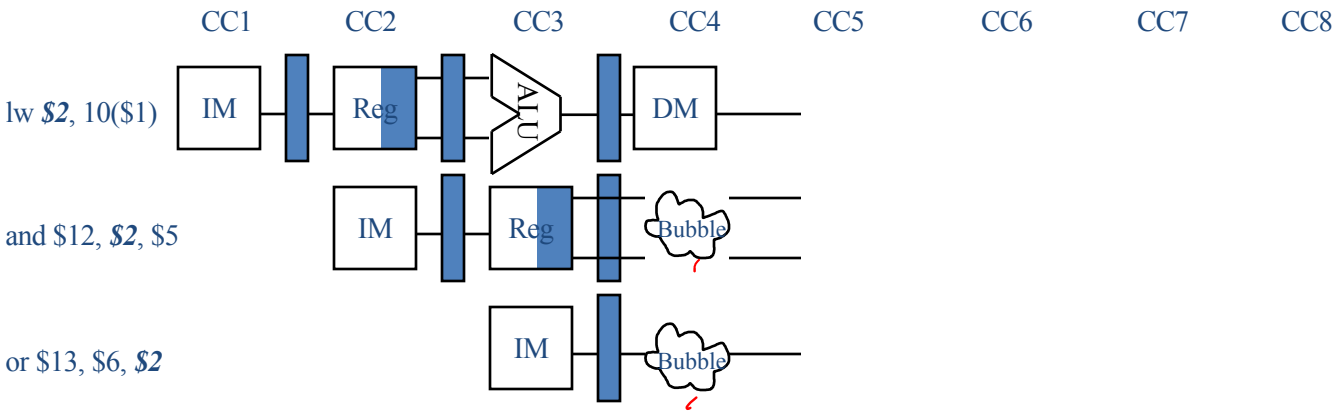
Eliminating Data Hazards via Forwarding and stalling

	CC1	CC2	CC3	CC4	CC5	CC6	CC7	CC8
lw ***$2***, 10($1)								
and $12, ***$2***, $5								
or $13, $6, ***$2***								
add $14, ***$2***, ***$2***								
sw $15, 100(***$2***)								

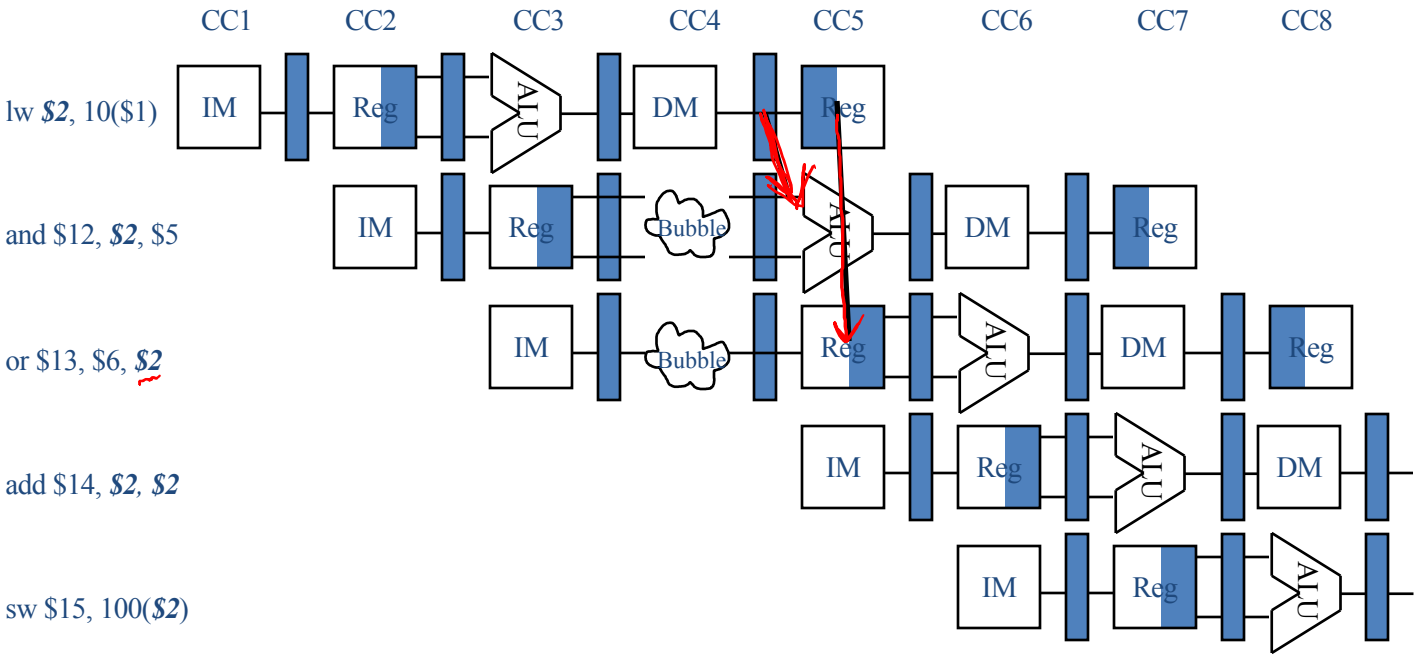
Eliminating Data Hazards via Forwarding and stalling



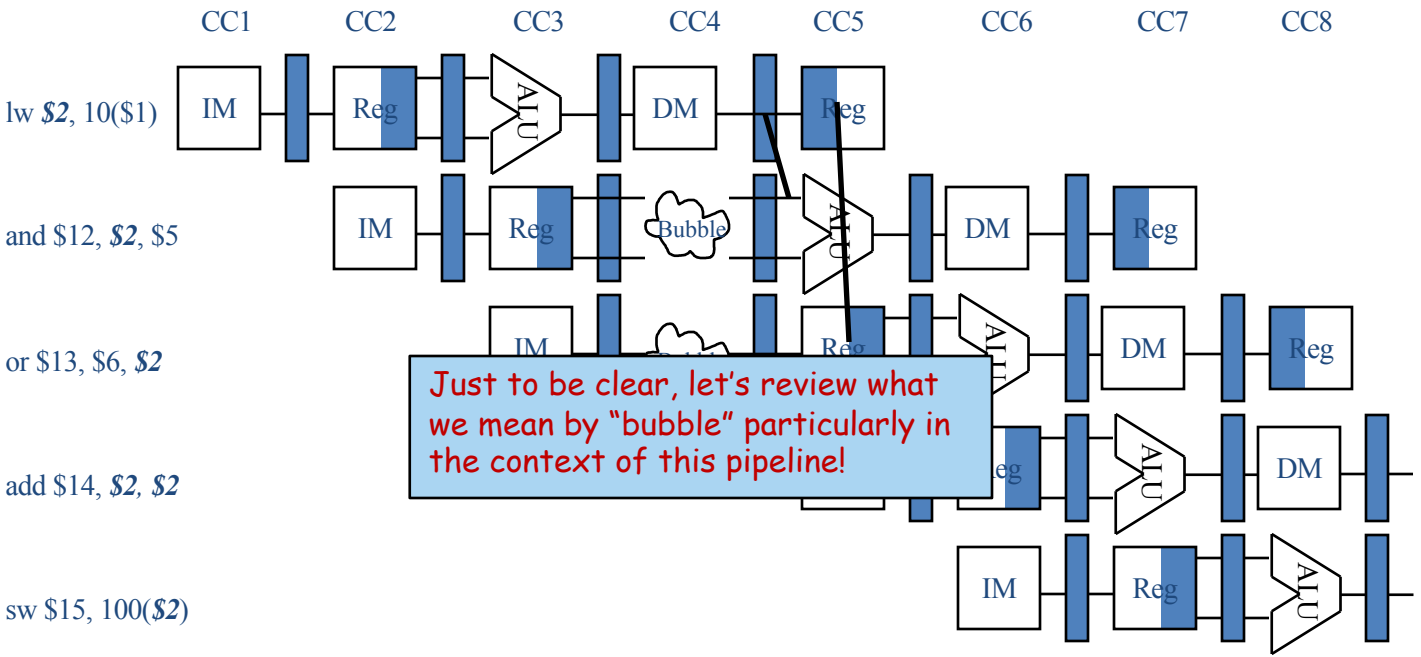
Eliminating Data Hazards via Forwarding and stalling



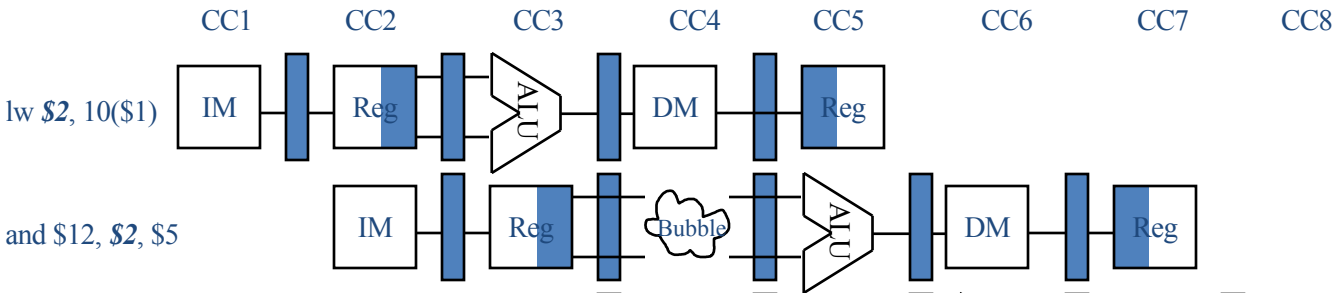
Eliminating Data Hazards via Forwarding and stalling



Eliminating Data Hazards via Forwarding and stalling

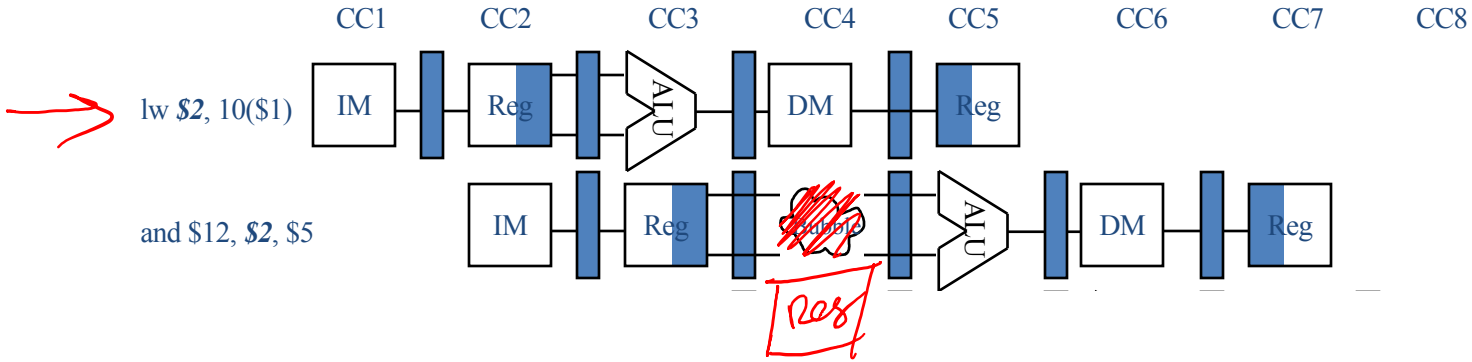


Eliminating Data Hazards via Forwarding and stalling



What is really happening during the bubble (for this particular pipeline)?

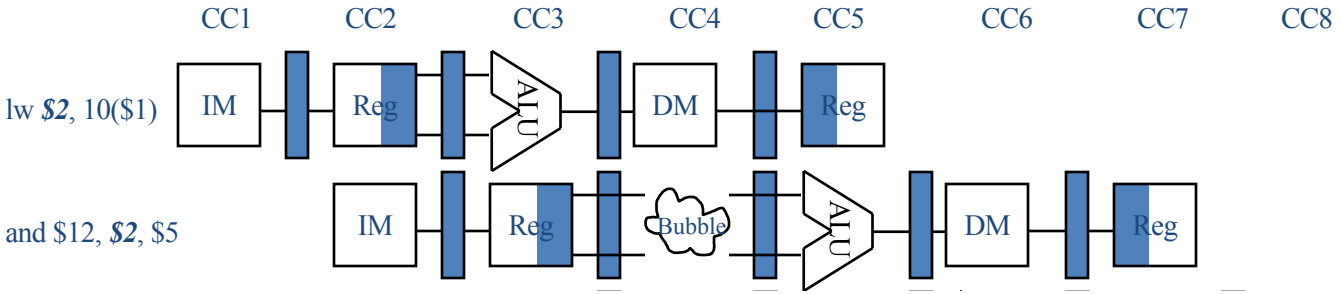
Eliminating Data Hazards via Forwarding and stalling



What is really happening during the bubble (for this particular pipeline)?

- While ***lw*** moves to the Mem stage in CC4, the ***and*** instruction repeats the ID stage (important because the values the ***and*** reads in CC4 are the ones it will carry forward).

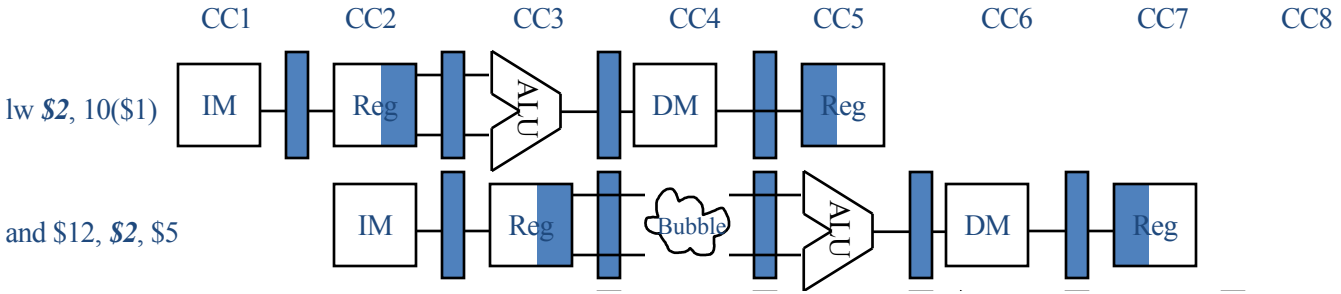
Eliminating Data Hazards via Forwarding and stalling



What is really happening during the bubble (for this particular pipeline)?

- While ***lw*** moves to the Mem stage in CC4, the ***and*** instruction repeats the ID stage (important because the values the ***and*** reads in CC4 are the ones it will carry forward).
- There is now *no instruction* in the EX stage. So we better make sure that whatever is in the EX stage is safe.

Eliminating Data Hazards via Forwarding and stalling



What is really happening during the bubble (for this particular pipeline)?

- While ***lw*** moves to the Mem stage in CC4, the ***and*** instruction repeats the ID stage (important because the values the ***and*** reads in CC4 are the ones it will carry forward).
- There is now *no instruction* in the EX stage. So we better make sure that whatever is in the EX stage is safe.
  - Safe = no state changes (PC, reg, memory), now or as it moves through the pipeline.

Poll Q: Stalls & Forwards

- How many stalls occur and how many values require hardware forwarding support to avoid stalling for our MIPS 5-stage pipeline?

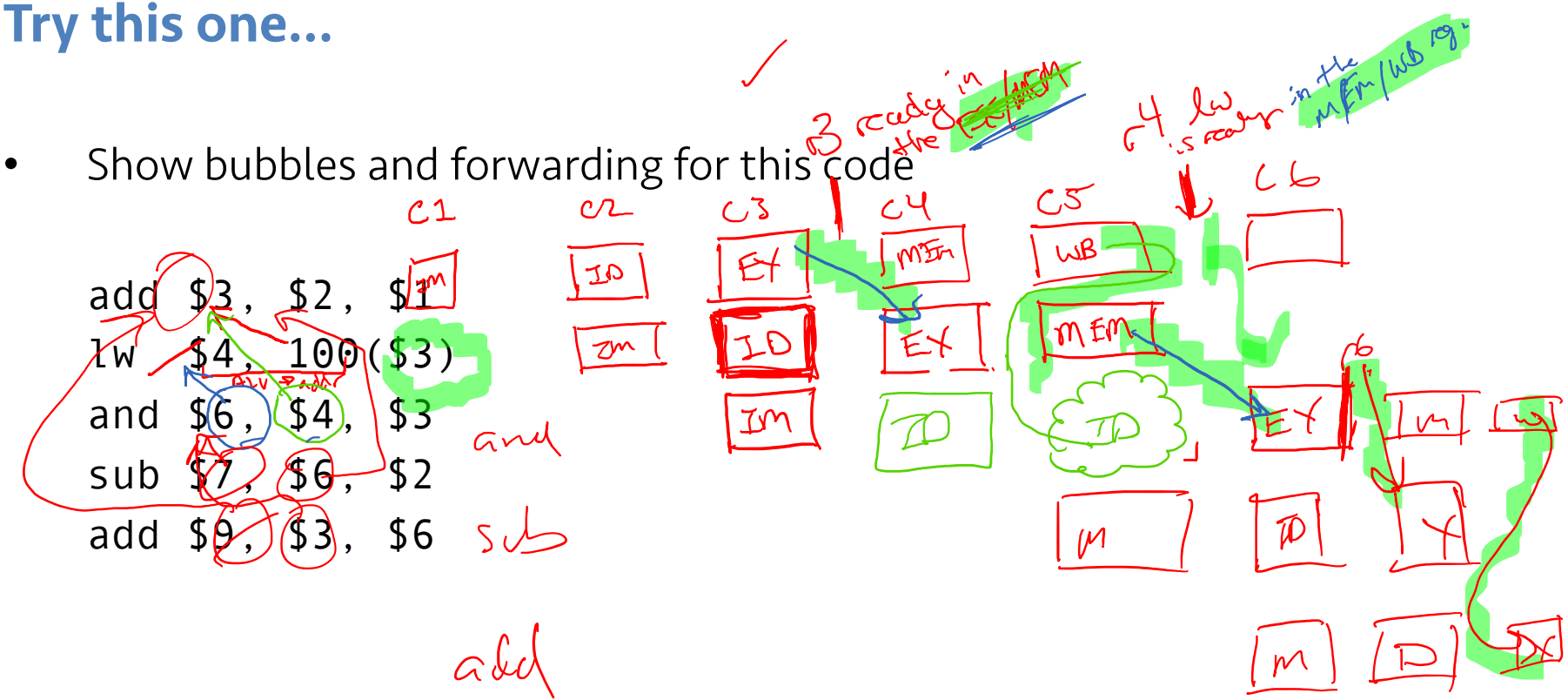
```
add $3, $2, $1
lw  $4, 100($3)
and $6, $4, $3
sub $7, $6, $2
add $9, $3, $6
```

Selection	Stalls	Forwarded values
A	1	3
B	2	4
C	2	3
D	1	5
E	None of the above	

Try this one…

- Show bubbles and forwarding for this code

```
add $3, $2, $1
lw  $4, 100($3)
and $6, $4, $3
sub $7, $6, $2
add $9, $3, $6
```



Another one...

To Do in Discussion on Wed

- Show bubbles and forwarding for this code

```
lw   $9, 100($6)      IF   ID   EX   M   WB
addi $6, $9, #26
sub  $7, $6, $9
add  $6, $3, $6
add  $3, $2, $6
```