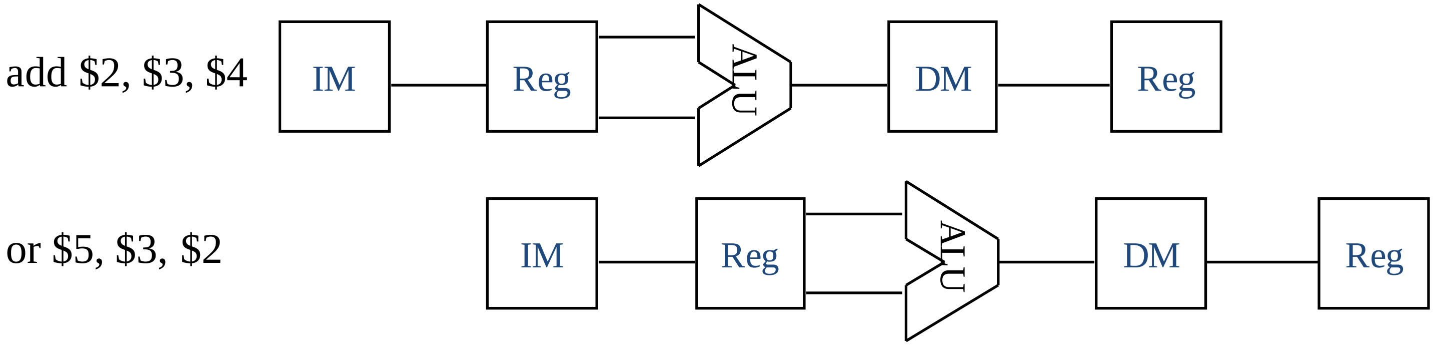
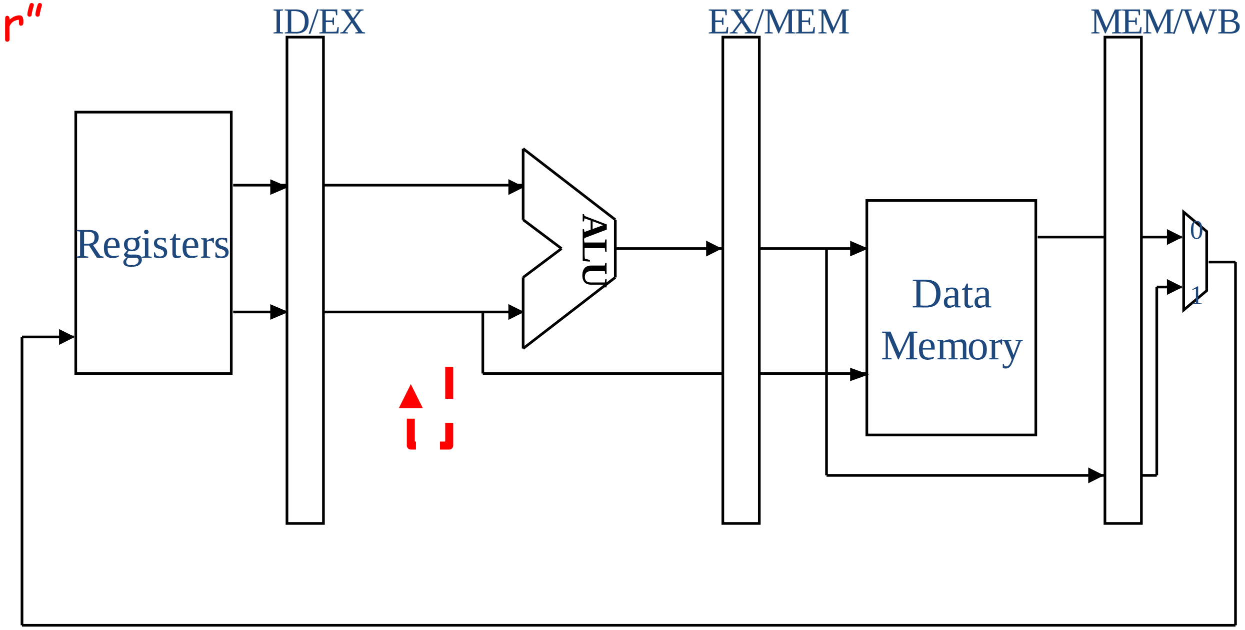
Pipelining

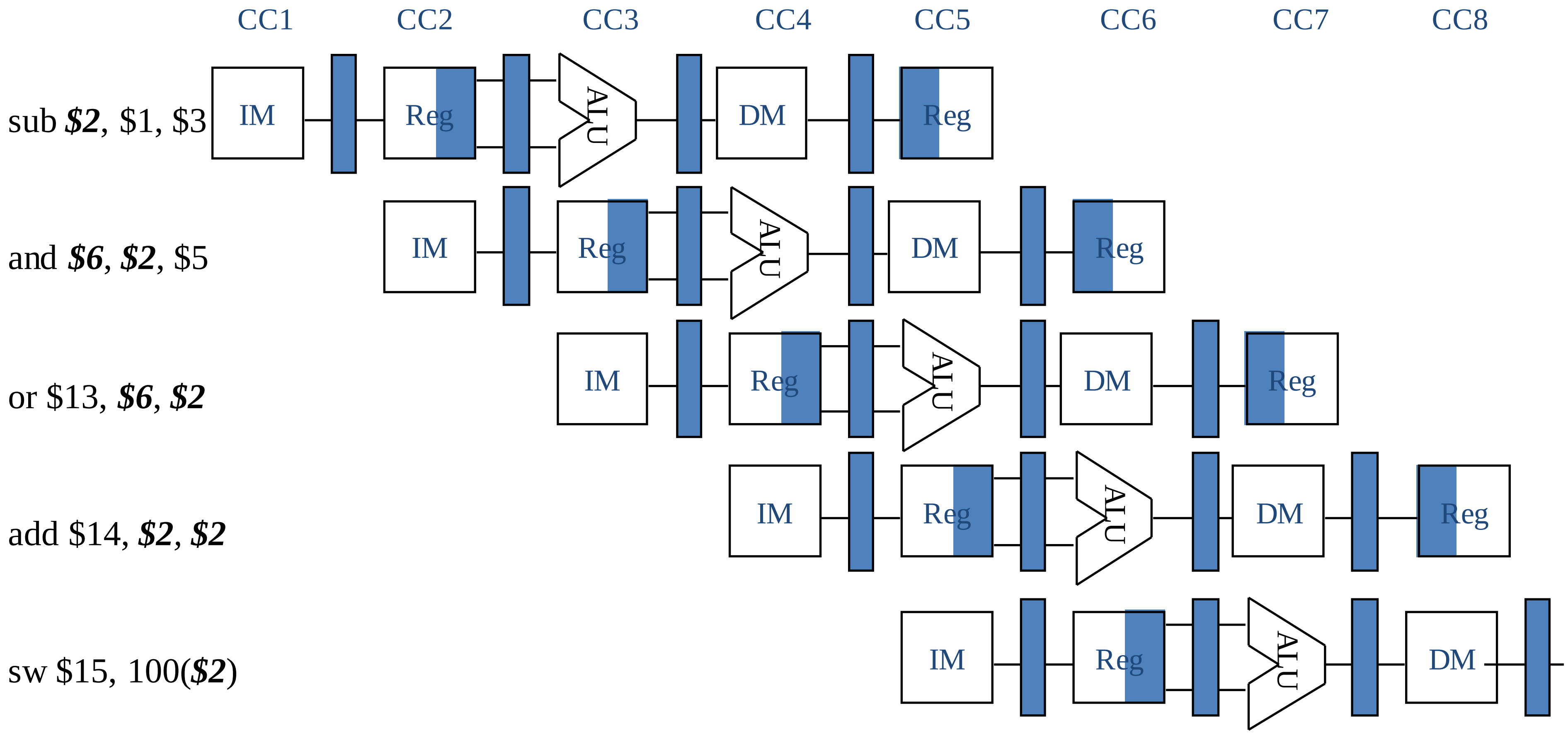
Reducing Data Hazards Through Forwarding



We could avoid stalling if we could get the ALU output from "add" to ALU input for the "or"



Eliminating Data Hazards via Forwarding



sub ***$2***, $1, $3

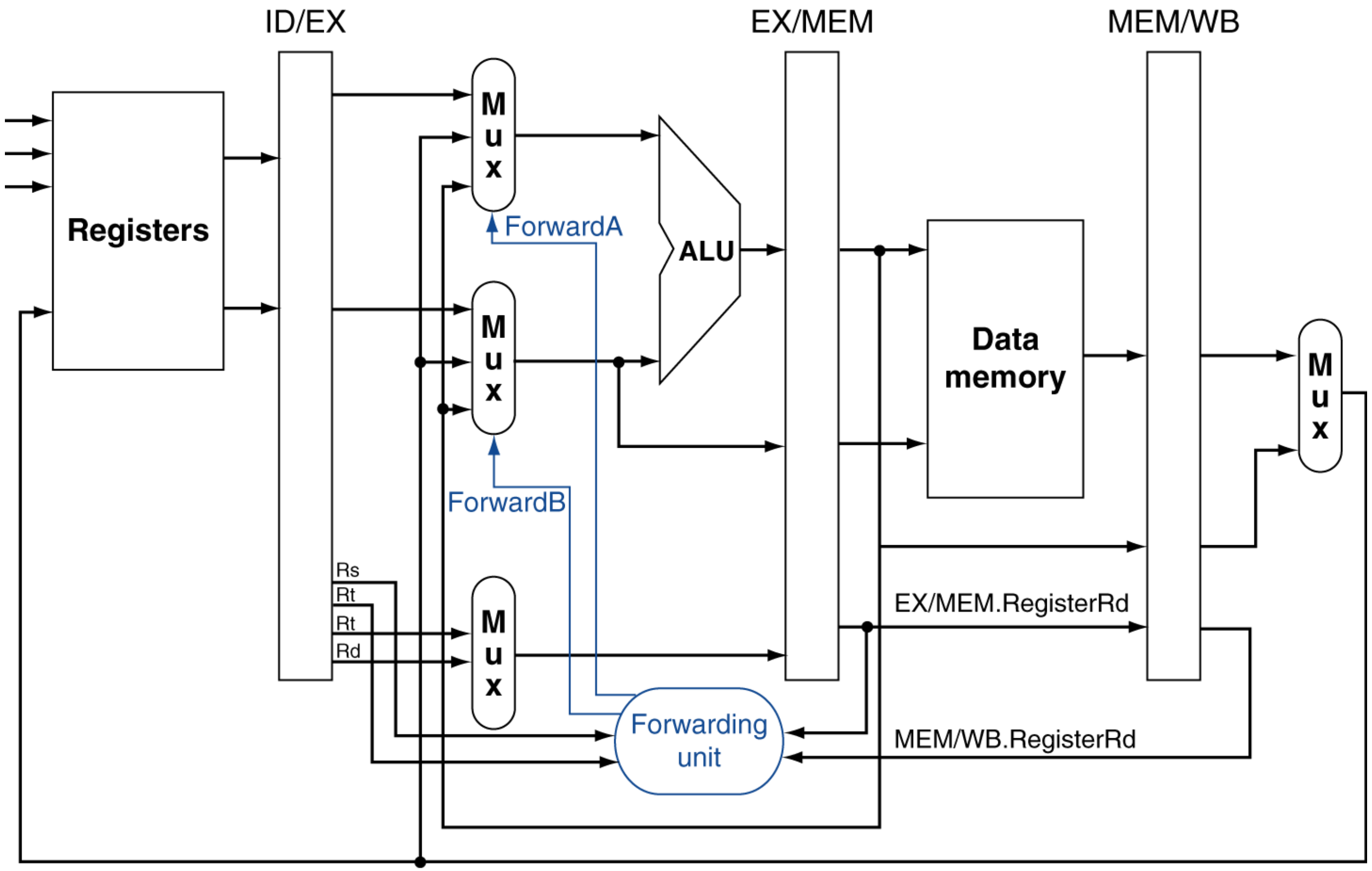
and ***$6***, ***$2***, $5

or $13, ***$6***, ***$2***

add $14, ***$2***, ***$2***

sw $15, 100(***$2***)

Forwarding Paths



b. With forwarding

Forwarding Conditions

- EX hazard
  - if (EX/MEM.RegWrite
    and (EX/MEM.RegisterRd = ID/EX.RegisterRs))
    ForwardA = 10
  - if (EX/MEM.RegWrite
    and (EX/MEM.RegisterRd = ID/EX.RegisterRt))
    ForwardB = 10
- MEM hazard
  - if (MEM/WB.RegWrite
    and (MEM/WB.RegisterRd = ID/EX.RegisterRs))
    ForwardA = 01
  - if (MEM/WB.RegWrite
    and (MEM/WB.RegisterRd = ID/EX.RegisterRt))
    ForwardB = 01

Double Data Hazard

- Consider the sequence:

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

- Both hazards occur
  - Want to use the most recent
- Revise MEM hazard condition
  - Only fwd if EX hazard condition isn’t true

Revised Forwarding Condition

- MEM hazard

  -
    ```
    if (MEM/WB.RegWrite
      and not (EX/MEM.RegWrite
              and (EX/MEM.RegisterRd = ID/EX.RegisterRs))
      and (MEM/WB.RegisterRd = ID/EX.RegisterRs))
     ForwardA = 01
    ```

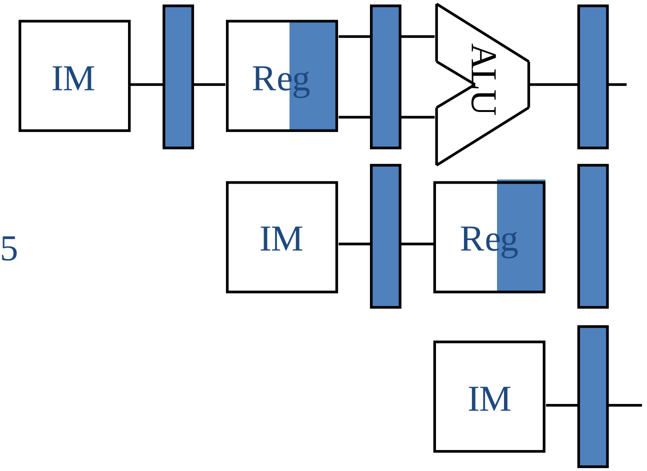
  -
    ```
    if (MEM/WB.RegWrite
      and not (EX/MEM.RegWrite
              and (EX/MEM.RegisterRd = ID/EX.RegisterRt))
      and (MEM/WB.RegisterRd = ID/EX.RegisterRt))
     ForwardB = 01
    ```

Does Forwarding Eliminate All Data Hazards

	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***)								

Does Forwarding Eliminate All Data Hazards

CC1 CC2 CC3 CC4 CC5 CC6 CC7 CC8



lw ***$2***, 10($1)

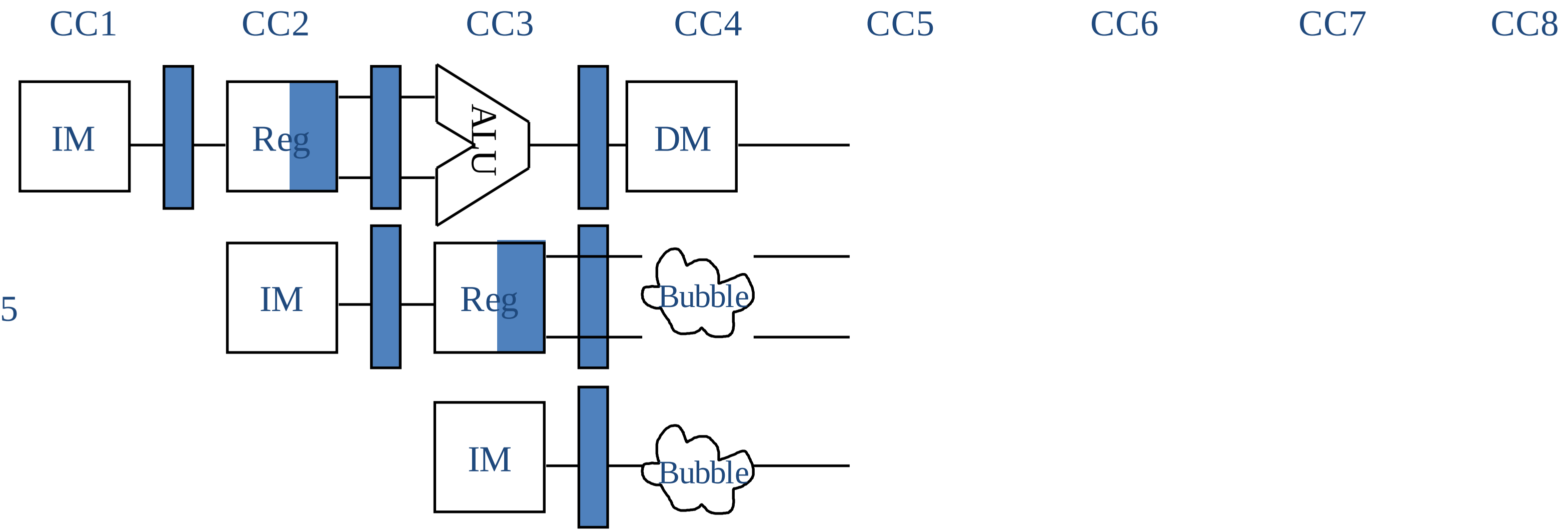
and $12, ***$2***, $5

or $13, $6, ***$2***

add $14, ***$2***, ***$2***

sw $15, 100(***$2***)

Does Forwarding Eliminate All Data Hazards



lw ***$2***, 10($1)

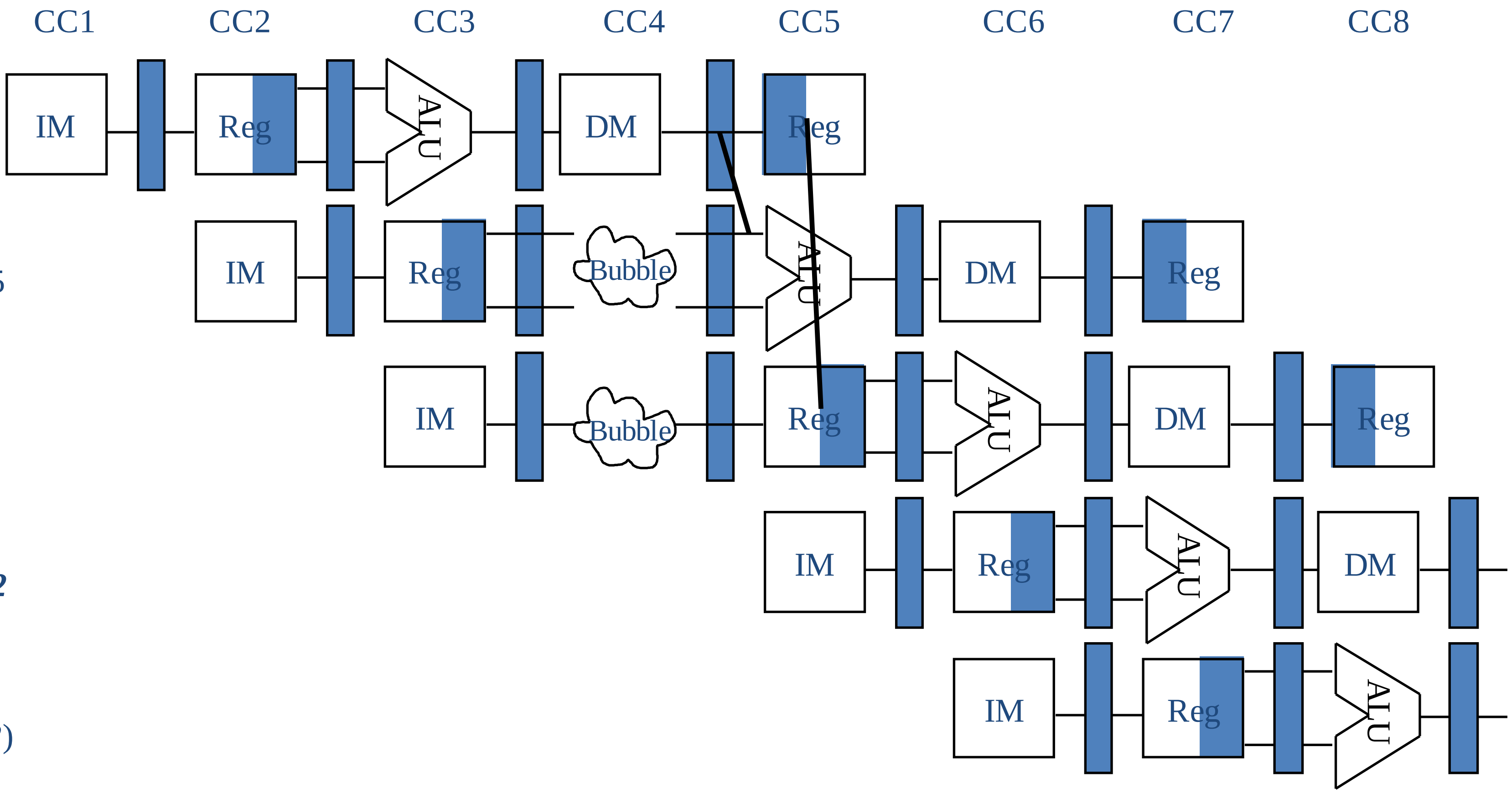
and $12, ***$2***, $5

or $13, $6, ***$2***

add $14, ***$2***, ***$2***

sw $15, 100(***$2***)

Does Forwarding Eliminate All Data Hazards



lw ***$2***, 10($1)

and $12, ***$2***, $5

or $13, $6, ***$2***

add $14, ***$2***, ***$2***

sw $15, 100(***$2***)

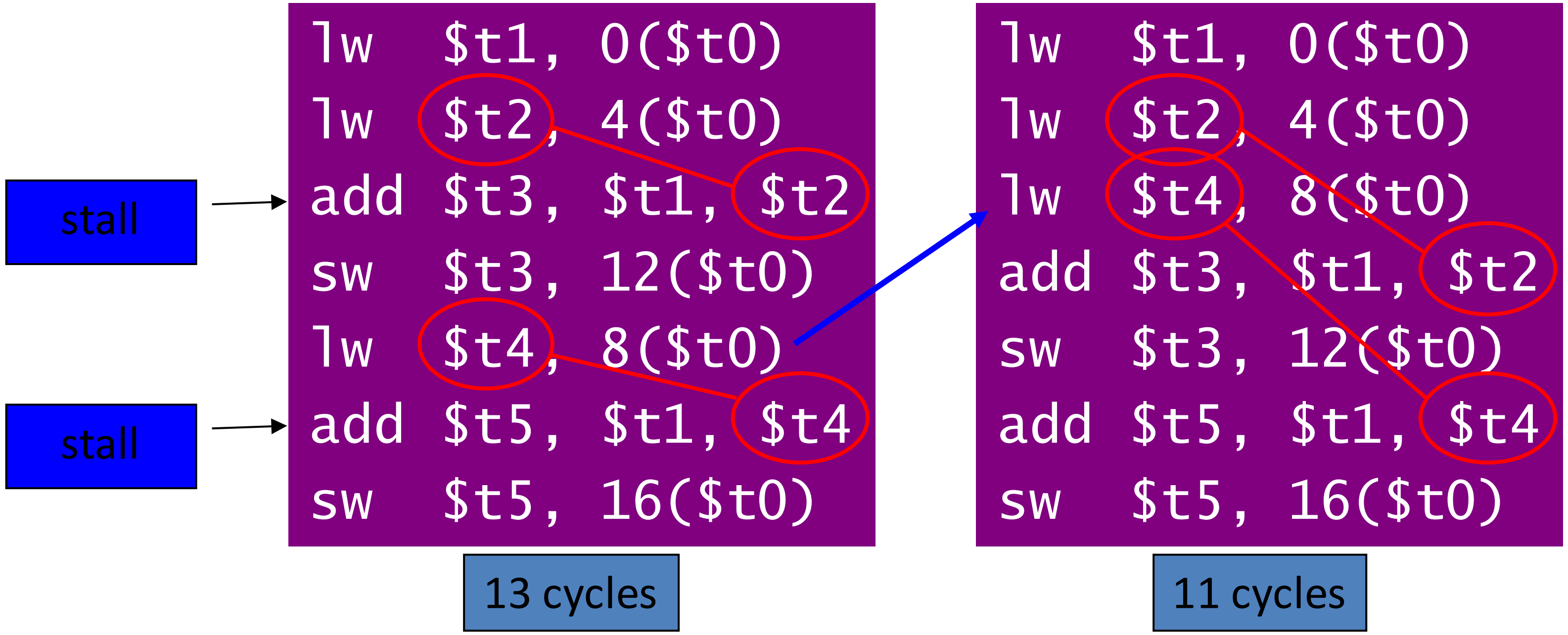
Try this one...

Show stalls and forwarding for this code

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

Code Scheduling to Avoid Stalls

- Reorder code to avoid use of load result in the next instruction
- C code for `A = B + E; C = B + F;`



stall

stall

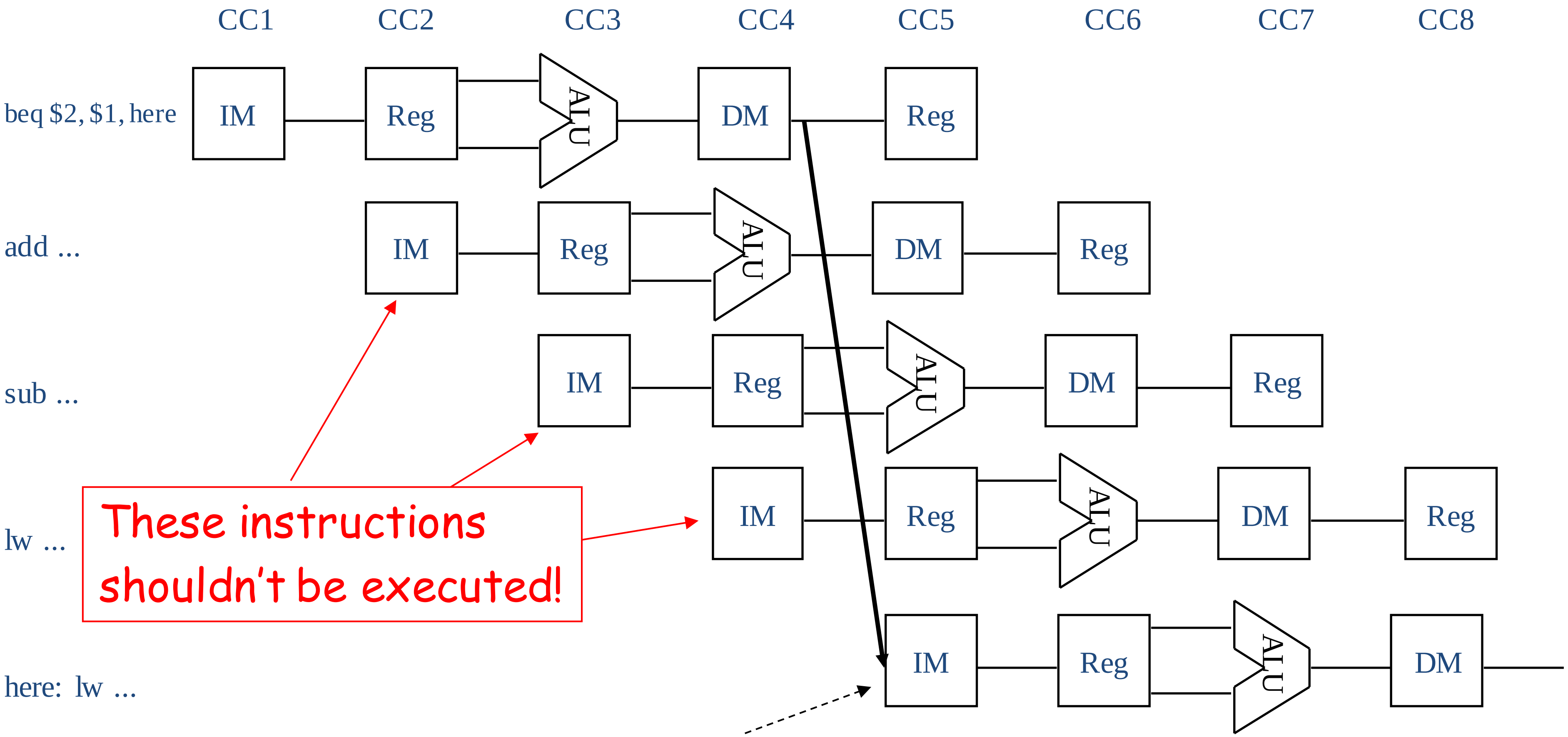
```
lw   $t1, 0($t0)
lw   $t2, 4($t0)
add  $t3, $t1, $t2
sw   $t3, 12($t0)
lw   $t4, 8($t0)
add  $t5, $t1, $t4
sw   $t5, 16($t0)
```

13 cycles

```
lw   $t1, 0($t0)
lw   $t2, 4($t0)
lw   $t4, 8($t0)
add  $t3, $t1, $t2
sw   $t3, 12($t0)
add  $t5, $t1, $t4
sw   $t5, 16($t0)
```

11 cycles

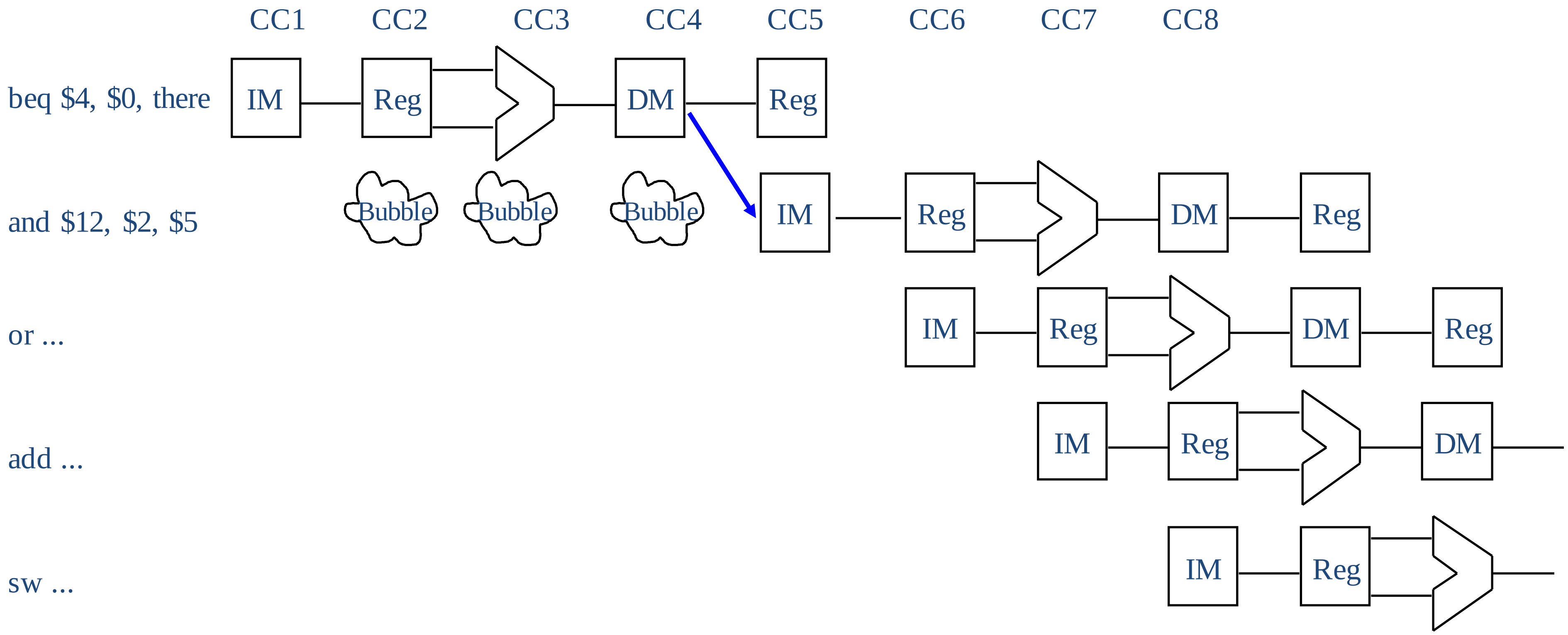
Branch Hazards



These instructions shouldn't be executed!

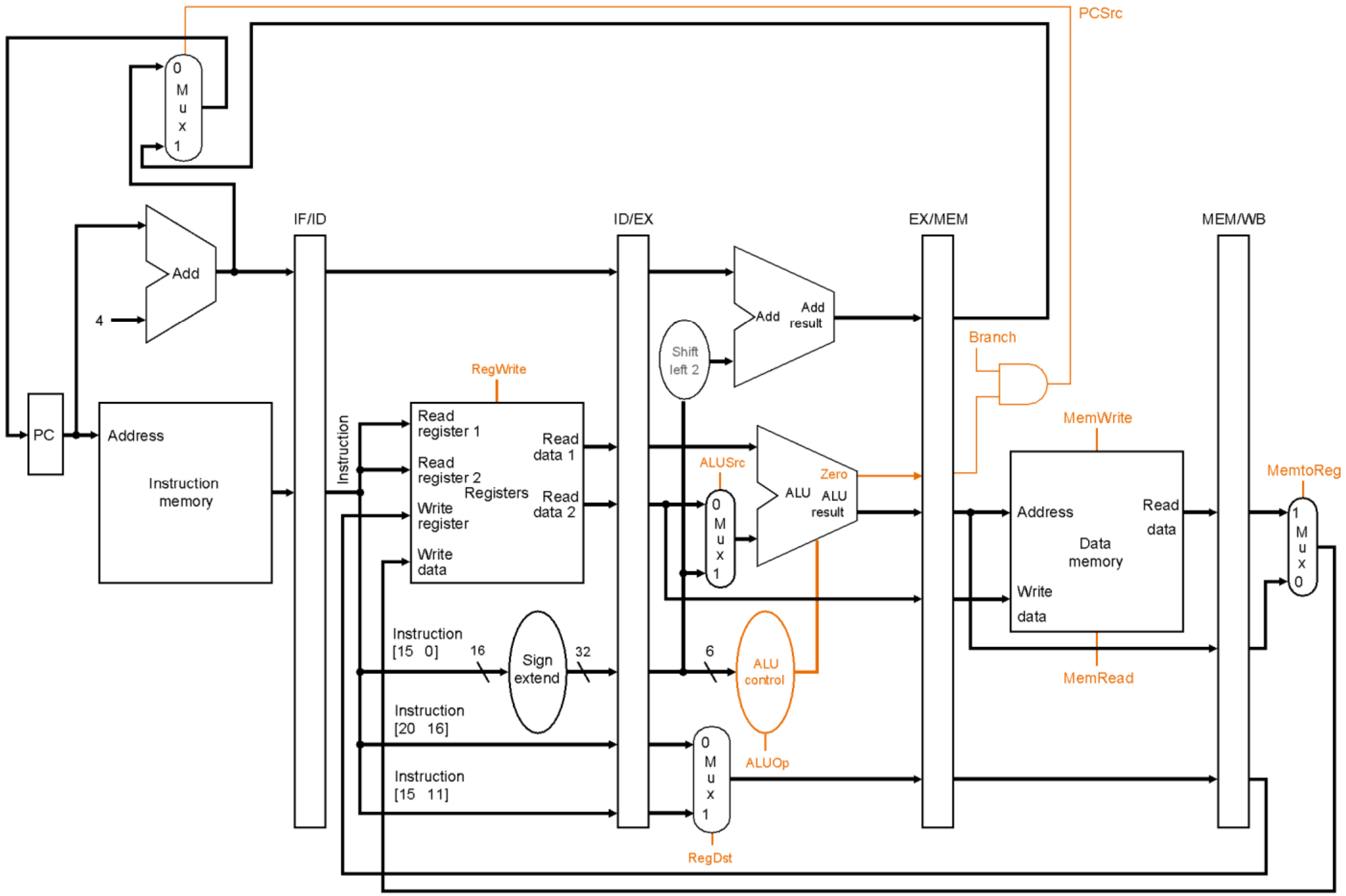
Finally, the right instruction

Stalling for Branch Hazards



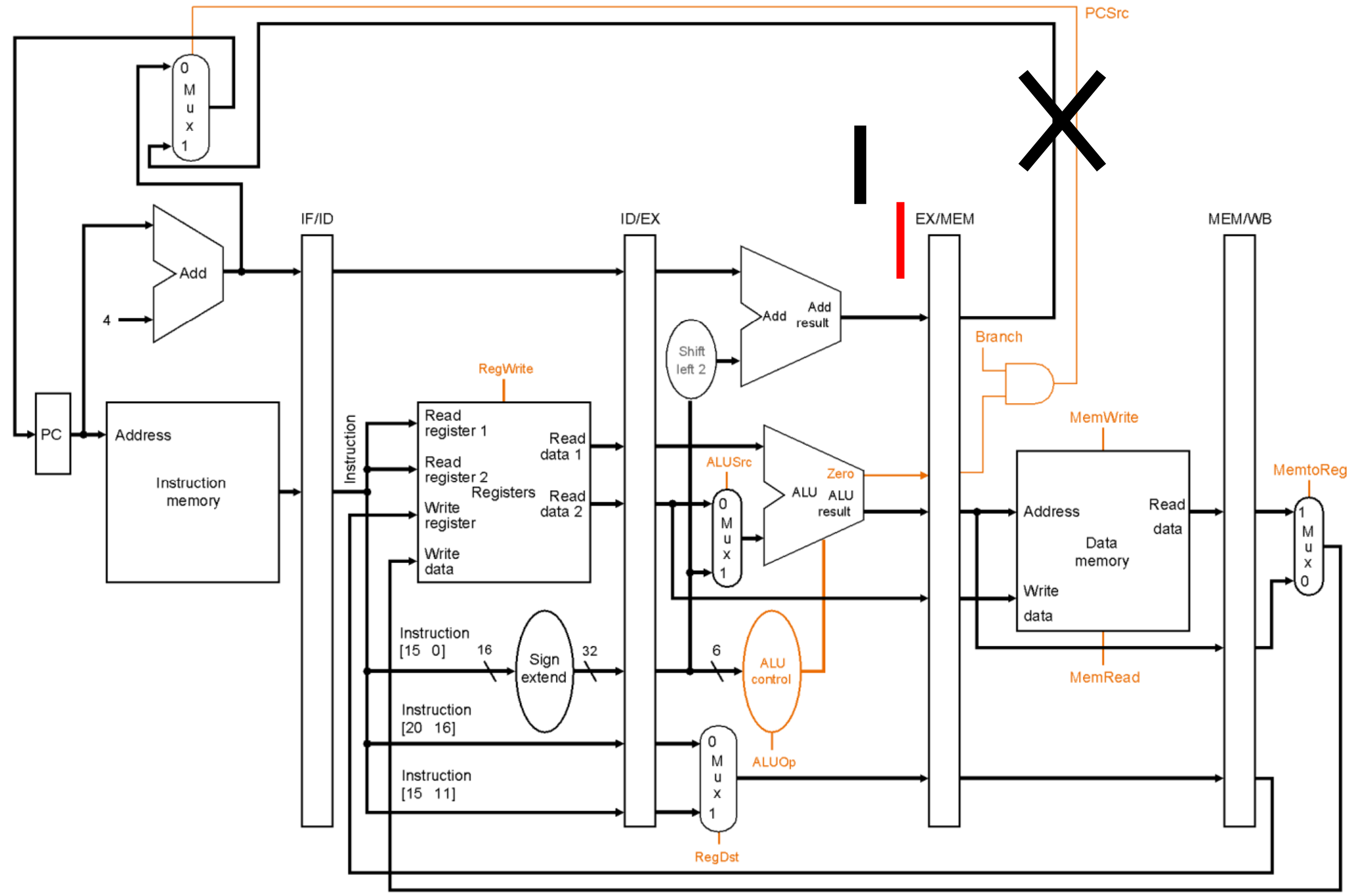
Reducing the Branch Delay

it's easy to reduce stall to 2-cycles

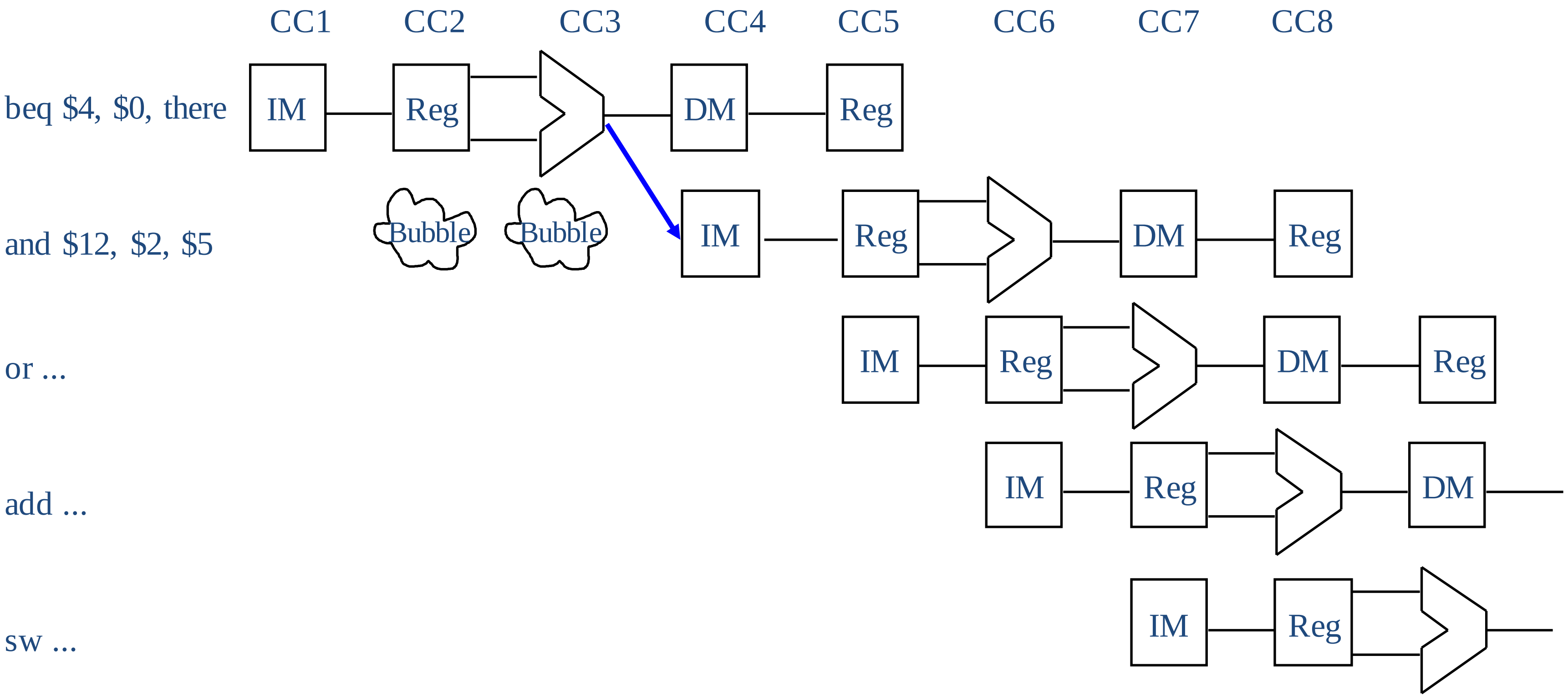


Reducing the Branch Delay

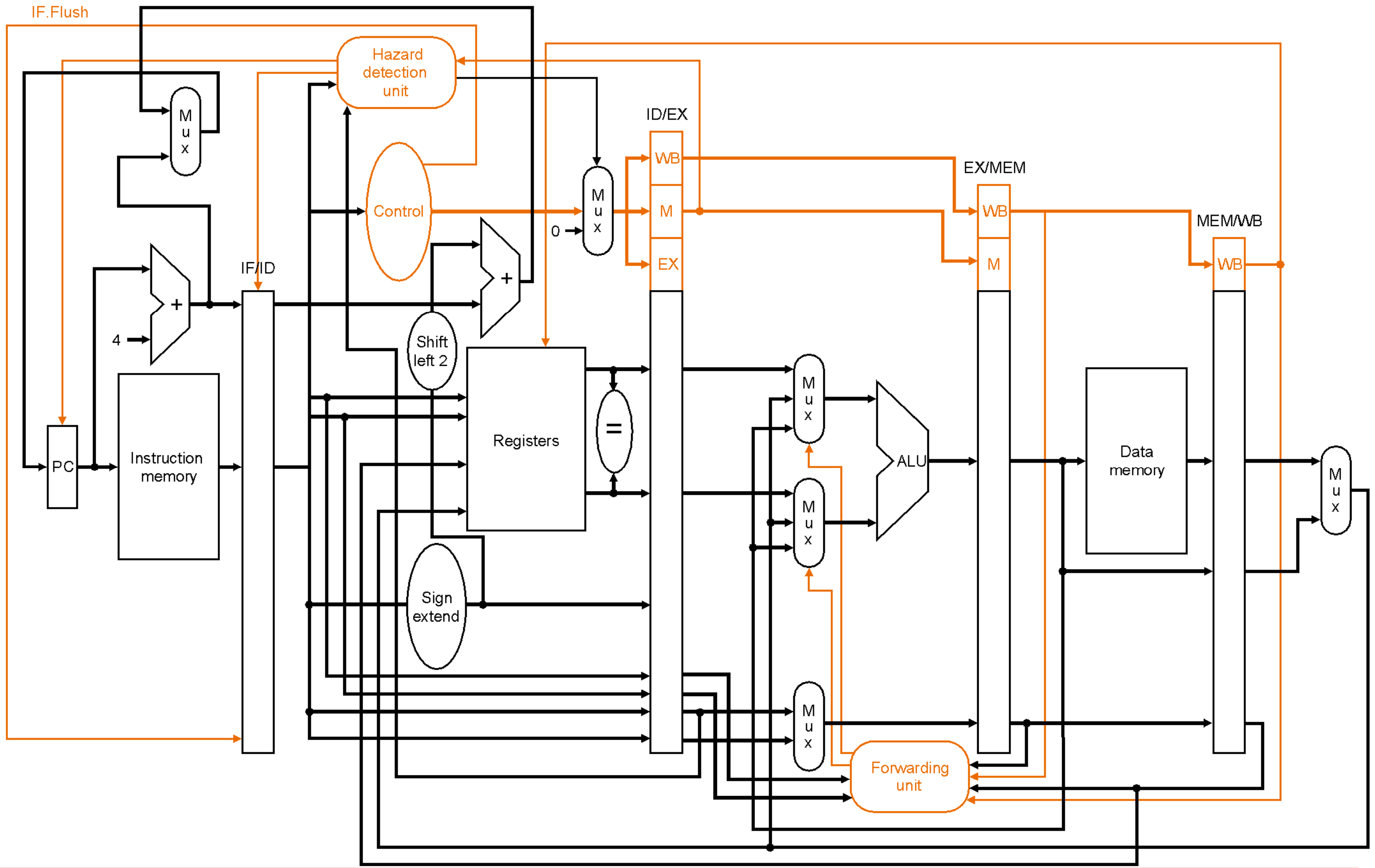
it's easy to reduce stall to 2-cycles



Stalling for Branch Hazards

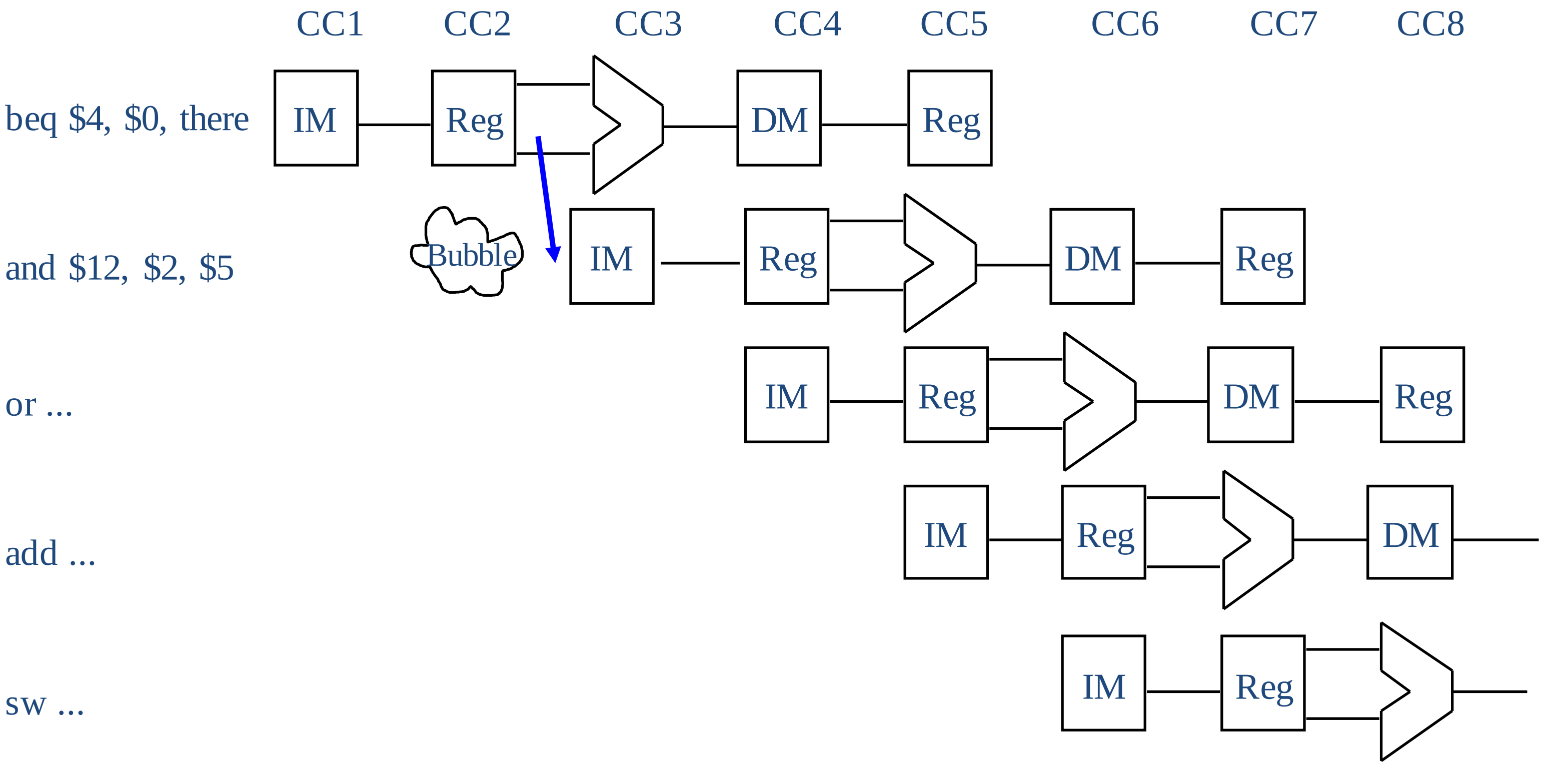


One-cycle branch misprediction penalty



- Target computation & equality check in ID phase.
  - This figure also shows flushing hardware (will talk about flushing later).

Stalling for Branch Hazards


CC1
CC2
CC3
CC4
CC5
CC6
CC7
CC8
beq $4, $0, there
IM
Reg
DM
Reg
and $12, $2, $5
Bubble
IM
Reg
DM
Reg
or ...
IM
Reg
DM
Reg
add ...
IM
Reg
DM
sw ...
IM
Reg