Chapter 4

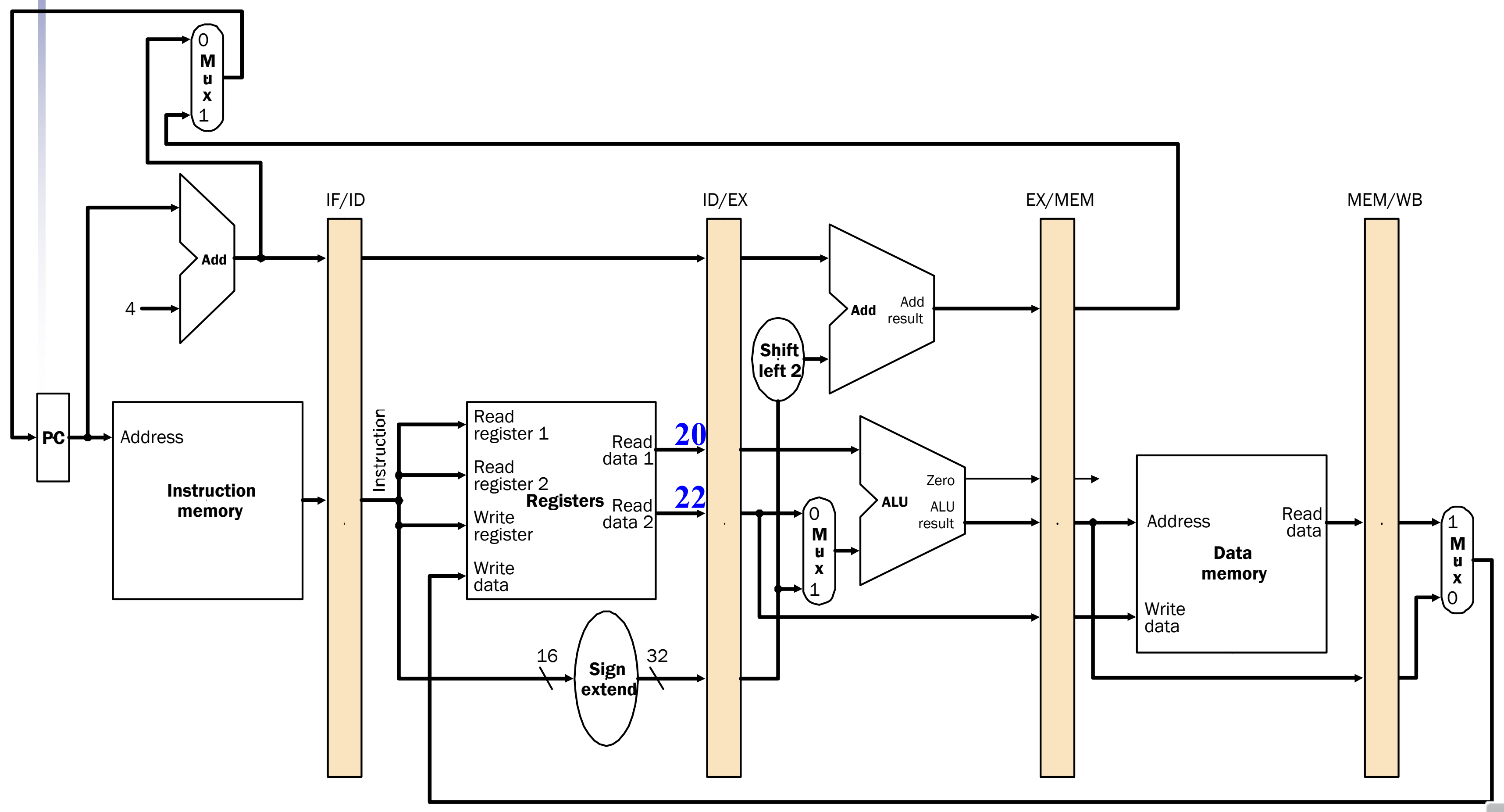
The Processor

Hazards

- Suppose initially, register $i holds the number 2i
- What happens when we see the following dynamic instruction sequence:
  - **add $3, $10, $11**
    - this should add 20 + 22, putting result 42 into $3
  - **lw $8, 50($3)**
    - this should load memory location 92 (42+50) into $8
  - **sub $11, $8, $7**
    - this should subtract 14 from that just-loaded val

The Pipeline in Execution

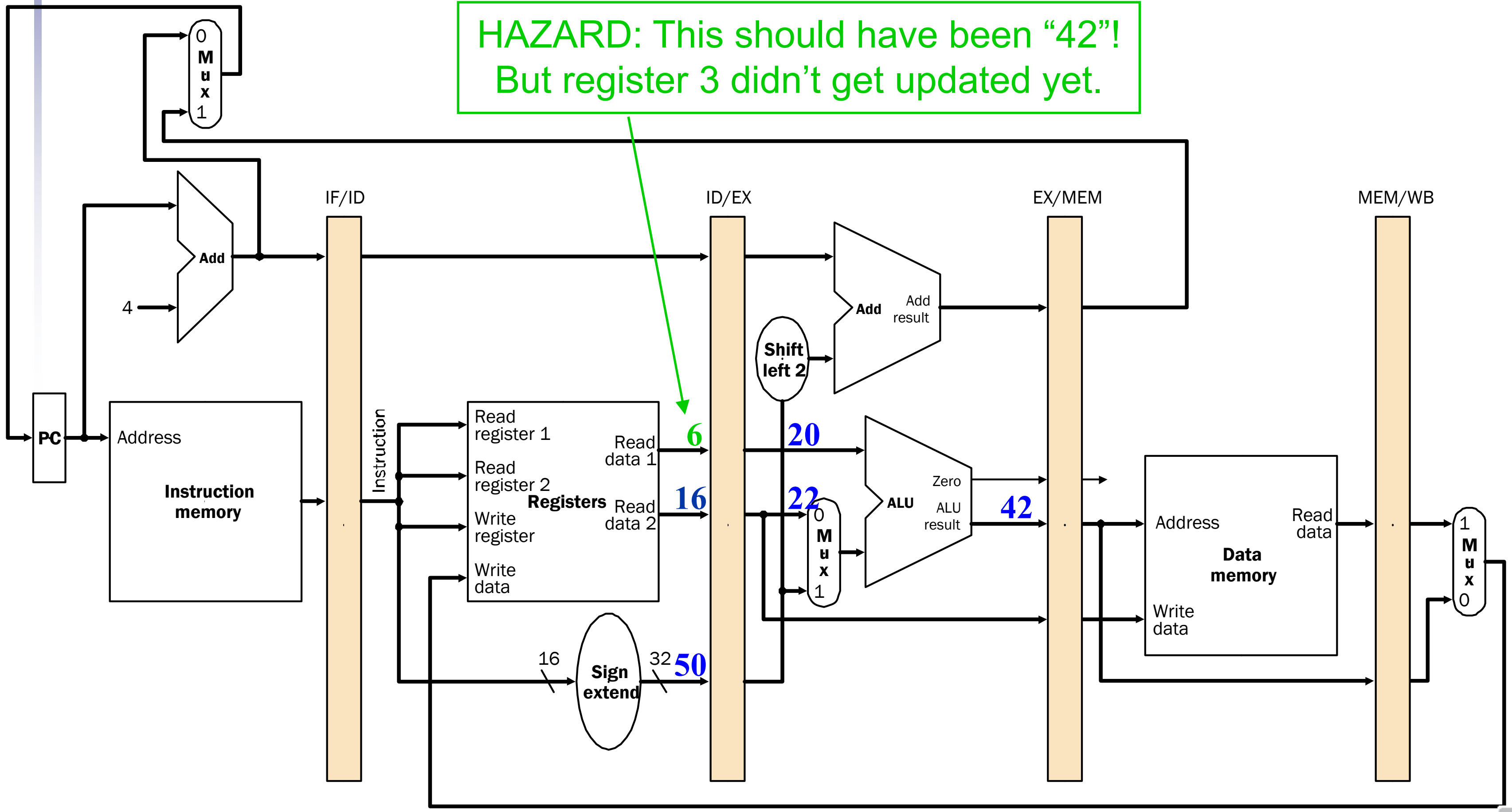
lw $8, 50($3) **add $3, $10, $11** Execute/ Address Calculation Memory Access Write Back



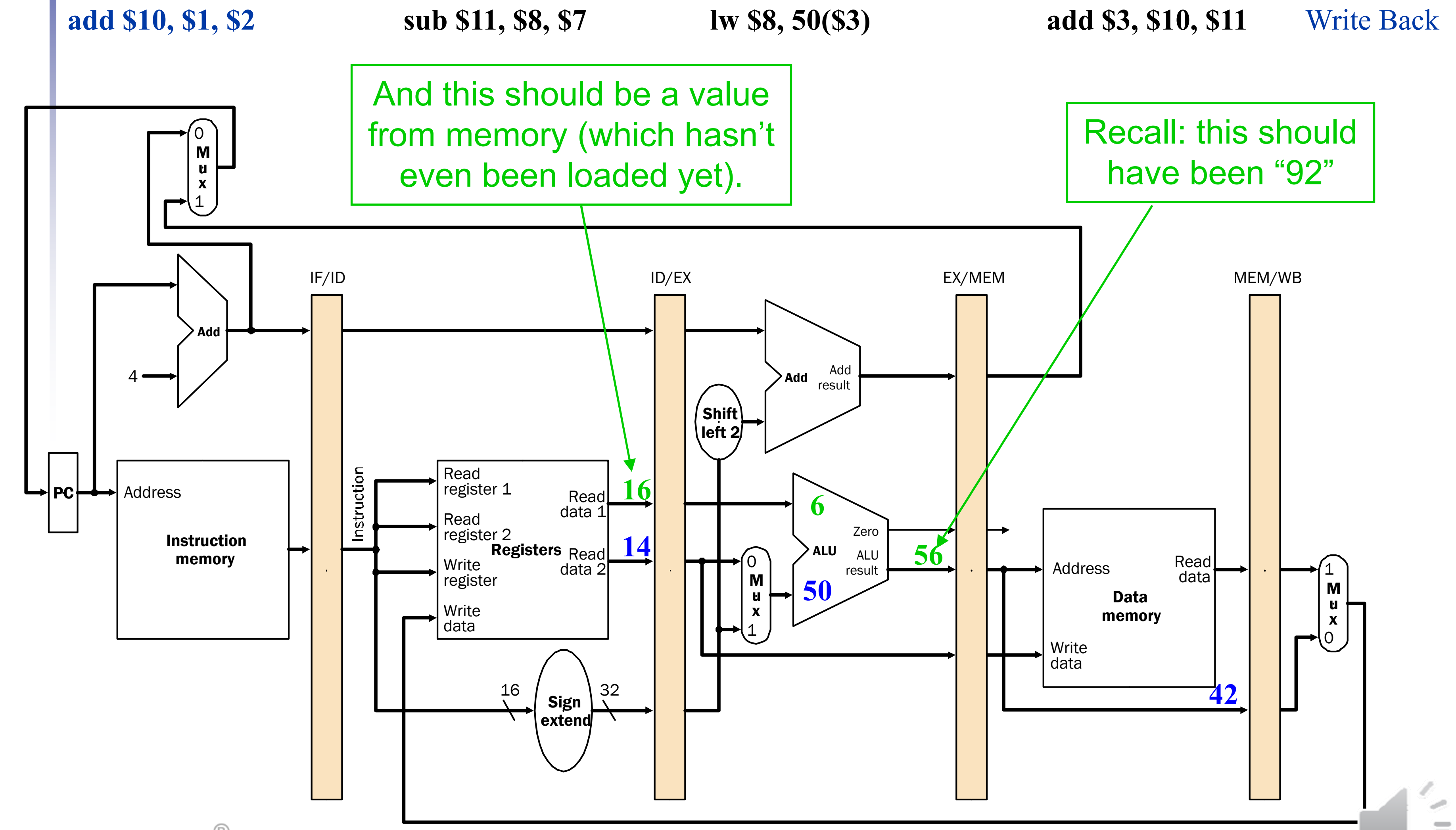
The Pipeline in Execution

sub \$11, \$8, \$7 lw \$8, 50(\$3) add \$3, \$10, \$11 Memory Access Write Back

HAZARD: This should have been "42"!
But register 3 didn't get updated yet.



The Pipeline in Execution



Hazards

- Situations that prevent starting the next instruction in the next cycle
- Structure hazards
  - A required resource is busy
- Data hazard
  - Need to wait for previous instruction to complete its data read/write
- Control hazard
  - Deciding on control action depends on previous instruction

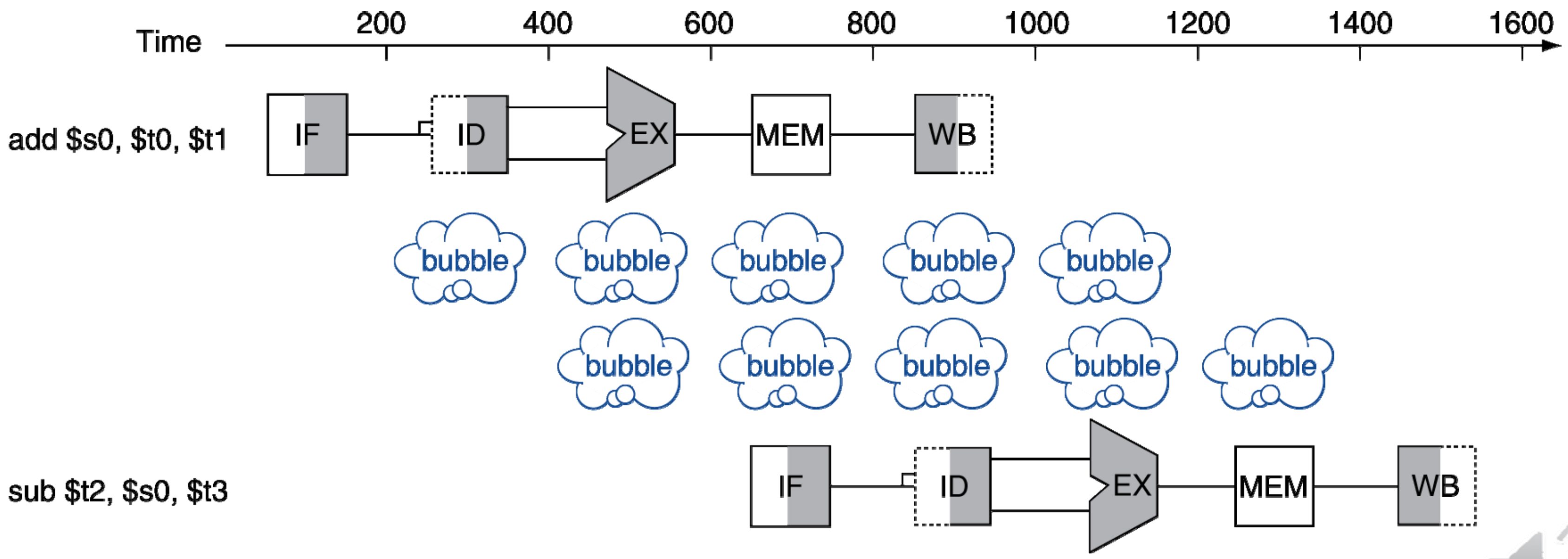
Structure Hazards

- Conflict for use of a resource
- In MIPS pipeline with a single memory
  - Load/store requires data access
  - Instruction fetch would have to *stall* for that cycle
    - Would cause a pipeline “bubble”
- Hence, pipelined datapaths require separate instruction/data memories
  - Or separate instruction/data caches

Data Hazards

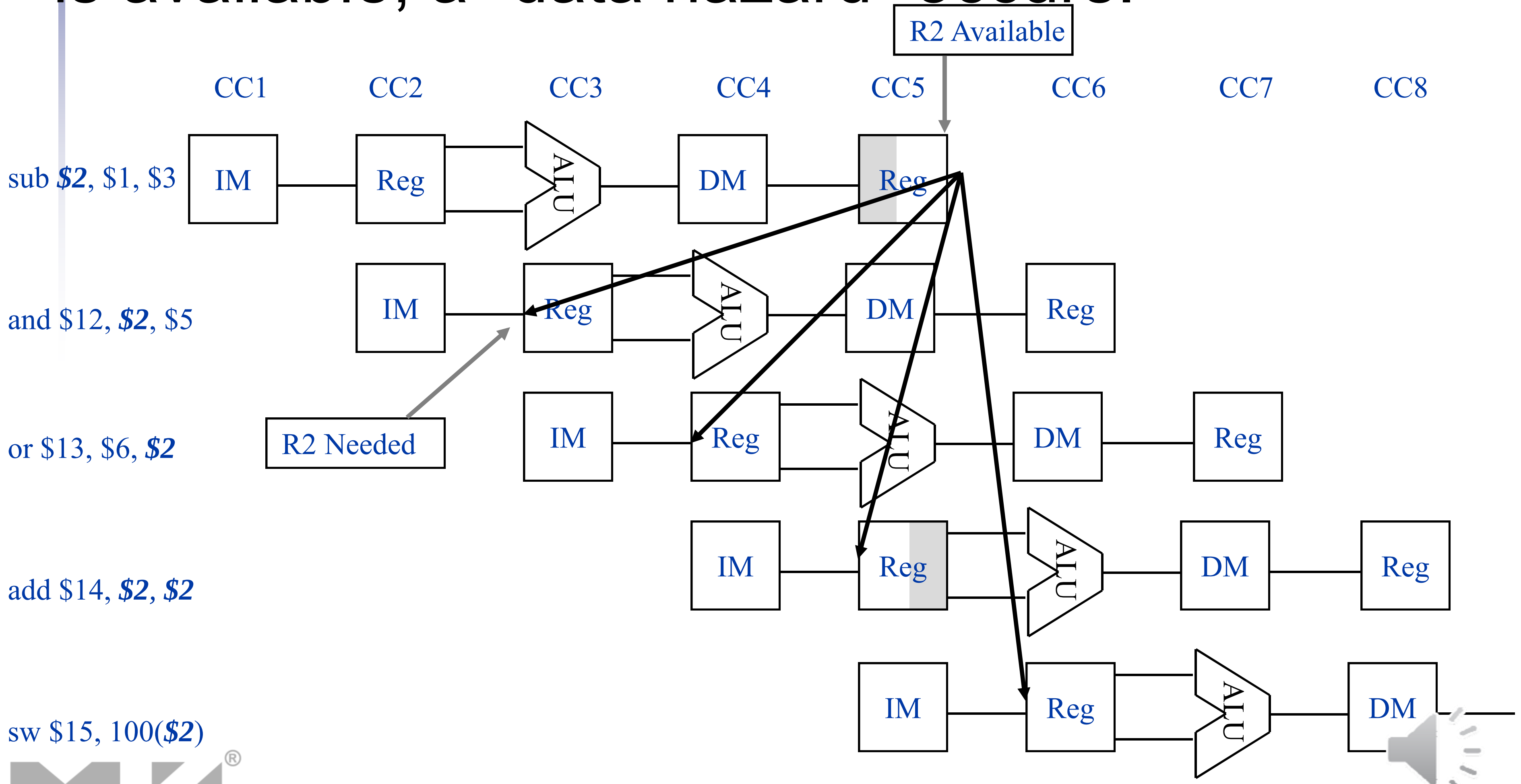
- An instruction depends on completion of data access by a previous instruction

  -
    ```
    add  $s0, $t0, $t1
    sub  $t2, $s0, $t3
    ```



Data Hazards

- When a result is needed in the pipeline before it is available, a "data hazard" occurs.



R2 Available
CC1
CC2
CC3
CC4
CC5
CC6
CC7
CC8
sub $2, $1, $3
and $12, $2, $5
or $13, $6, $2
add $14, $2, $2
sw $15, 100($2)
R2 Needed
IM
Reg
ALU
DM