

# Lecture: Pipelining Hazards

---

- Topics: Basic pipelining implementation, hazards, bypassing
- HW2 posted, due Wednesday

# Problem 3

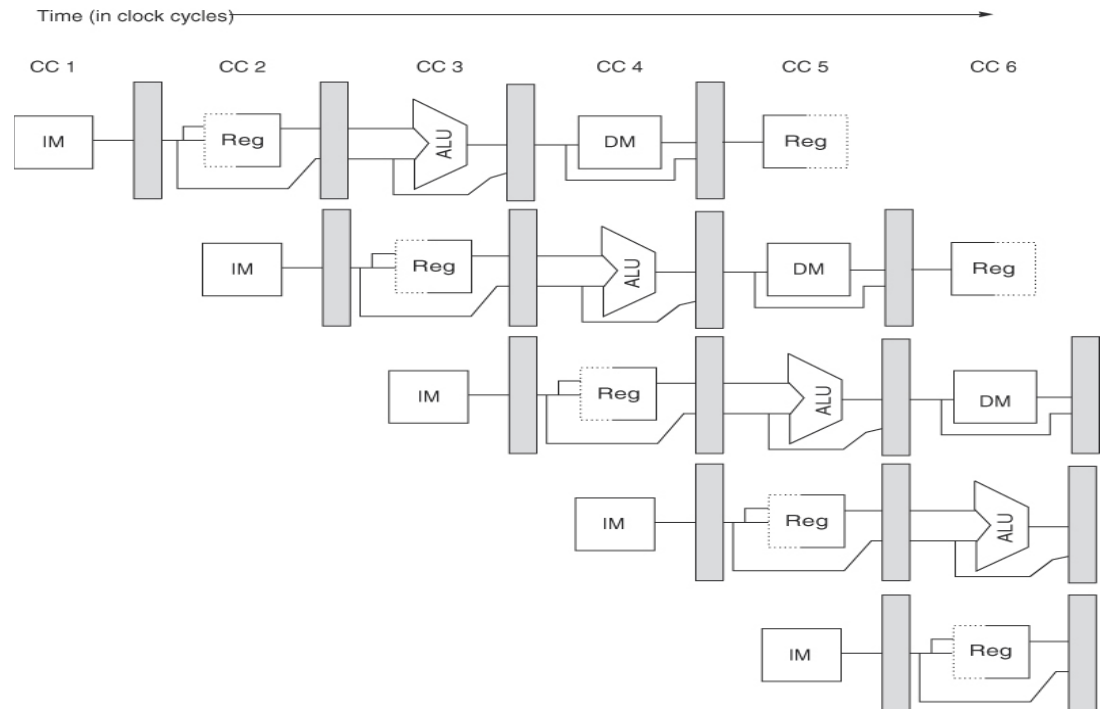
- For the following code sequence, show how the instrs flow through the pipeline:

ADD R1, R2, → R3

BEZ R4, [R5]

LD [R6] → R7

ST [R8] ← R9



# Pipeline Summary

---

	RR	ALU	DM	RW
ADD R1, R2, → R3	Rd R1,R2	R1+R2	--	Wr R3
BEZ R1, [R5]	Rd R1, R5	--	--	--
	Compare, Set PC			
LD 8[R3] → R6	Rd R3	R3+8	Get data	Wr R6
ST 8[R3] ← R6	Rd R3,R6	R3+8	Wr data	--

# Problem 4

---

- Convert this C code into equivalent RISC assembly instructions

`a[i] = b[i] + c[i];`

# Problem 4

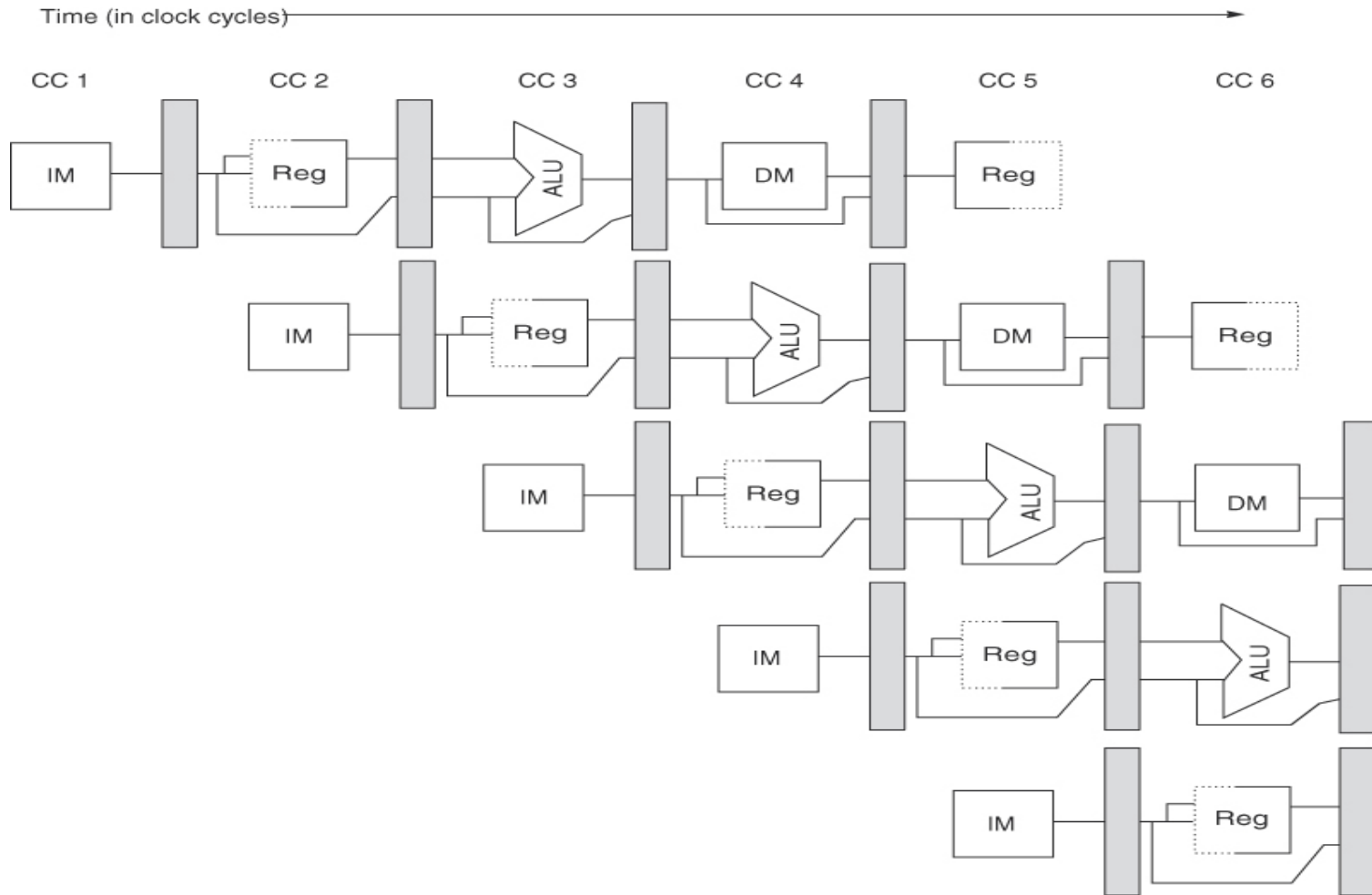
---

- Convert this C code into equivalent RISC assembly instructions

$a[i] = b[i] + c[i];$

```
LD [R1], R2 # R1 has the address for variable i
MUL R2, 8, R3 # the offset from the start of the array
ADD R4, R3, R7 # R4 has the address of a[0]
ADD R5, R3, R8 # R5 has the address of b[0]
ADD R6, R3, R9 # R6 has the address of c[0]
LD [R8], R10 # Bringing b[i]
LD [R9], R11 # Bringing c[i]
ADD R10, R11, R12 # Sum is in R12
ST [R7], R12 # Putting result in a[i]
```

# A 5-Stage Pipeline



# Hazards

---

- Structural hazards: different instructions in different stages (or the same stage) conflicting for the same resource
- Data hazards: an instruction cannot continue because it needs a value that has not yet been generated by an earlier instruction
- Control hazard: fetch cannot continue because it does not know the outcome of an earlier branch – special case of a data hazard – separate category because they are treated in different ways

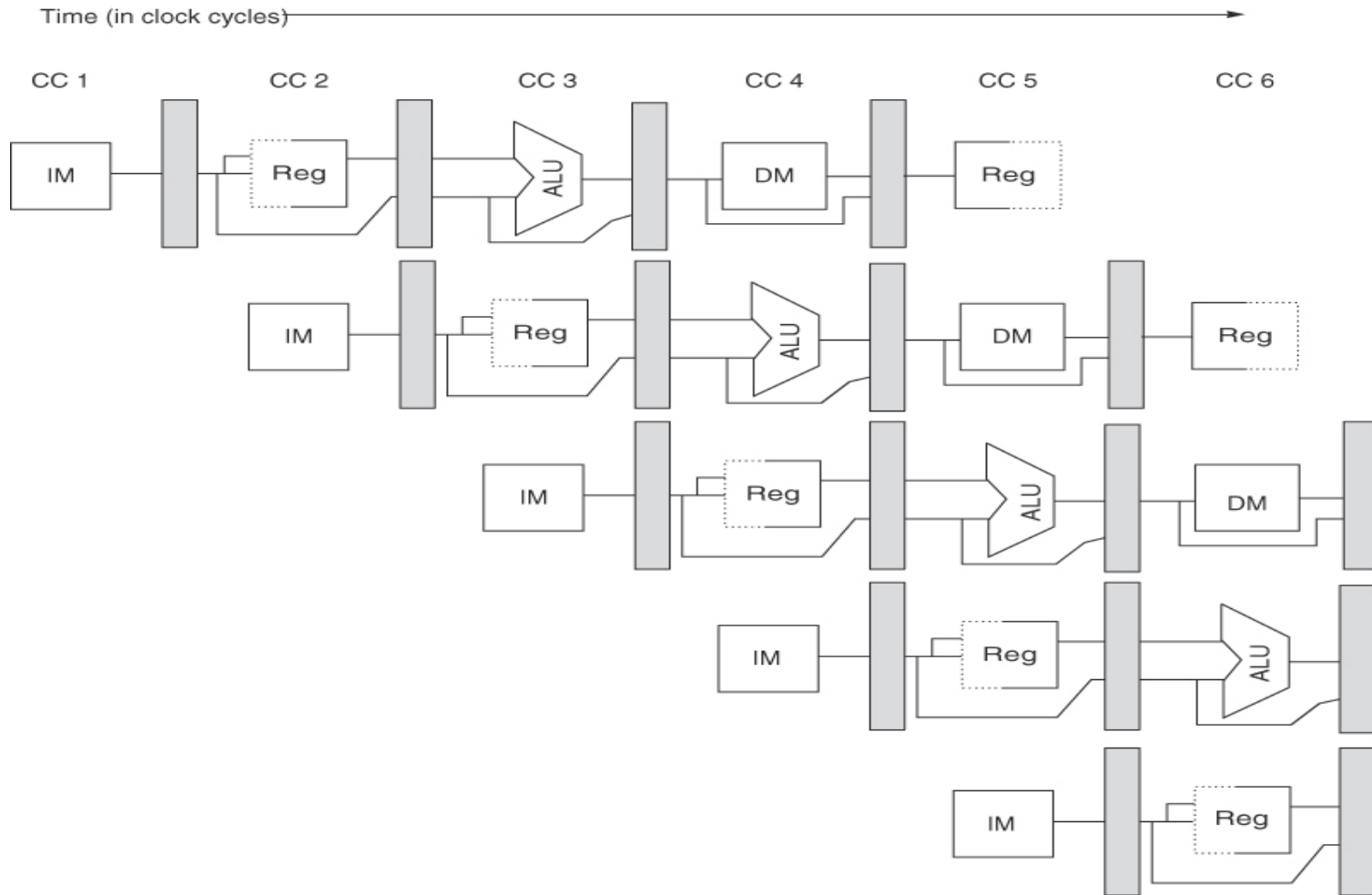
# Structural Hazards

---

- Example: a unified instruction and data cache → stage 4 (MEM) and stage 1 (IF) can never coincide
- The later instruction and all its successors are delayed until a cycle is found when the resource is free → these are pipeline bubbles
- Structural hazards are easy to eliminate – increase the number of resources (for example, implement a separate instruction and data cache)



# A 5-Stage Pipeline



# Problem 5

- Show the instruction occupying each stage in each cycle (no bypassing) if I1 is  $R1+R2 \rightarrow R3$  and I2 is  $R3+R4 \rightarrow R5$  and I3 is  $R7+R8 \rightarrow R9$

CYC-1    CYC-2    CYC-3    CYC-4    CYC-5    CYC-6    CYC-7    CYC-8

IF	IF	IF	IF	IF	IF	IF	IF
D/R	D/R	D/R	D/R	D/R	D/R	D/R	D/R
ALU	ALU	ALU	ALU	ALU	ALU	ALU	ALU
DM	DM	DM	DM	DM	DM	DM	DM
RW	RW	RW	RW	RW	RW	RW	RW

# Problem 5

- Show the instruction occupying each stage in each cycle (no bypassing) if I1 is  $R1+R2 \rightarrow R3$  and I2 is  $R3+R4 \rightarrow R5$  and I3 is  $R7+R8 \rightarrow R9$

CYC-1    CYC-2    CYC-3    CYC-4    CYC-5    CYC-6    CYC-7    CYC-8

IF I1	IF I2	IF I3	IF I3	IF I3	IF I4	IF I5	IF
D/R	D/R I1	D/R I2	D/R I2	D/R I2	D/R I3	D/R I4	D/R
ALU	ALU	ALU I1	ALU	ALU	ALU I2	ALU I3	ALU
DM	DM	DM	DM I1	DM	DM	DM I2	DM I3
RW	RW	RW	RW	RW I1	RW	RW	RW I2

# Problem 6

- 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	IF	IF	IF	IF	IF	IF	IF
D/R	D/R	D/R	D/R	D/R	D/R	D/R	D/R
ALU	ALU	ALU	ALU	ALU	ALU	ALU	ALU
DM	DM	DM	DM	DM	DM	DM	DM
RW	RW	RW	RW	RW	RW	RW	RW

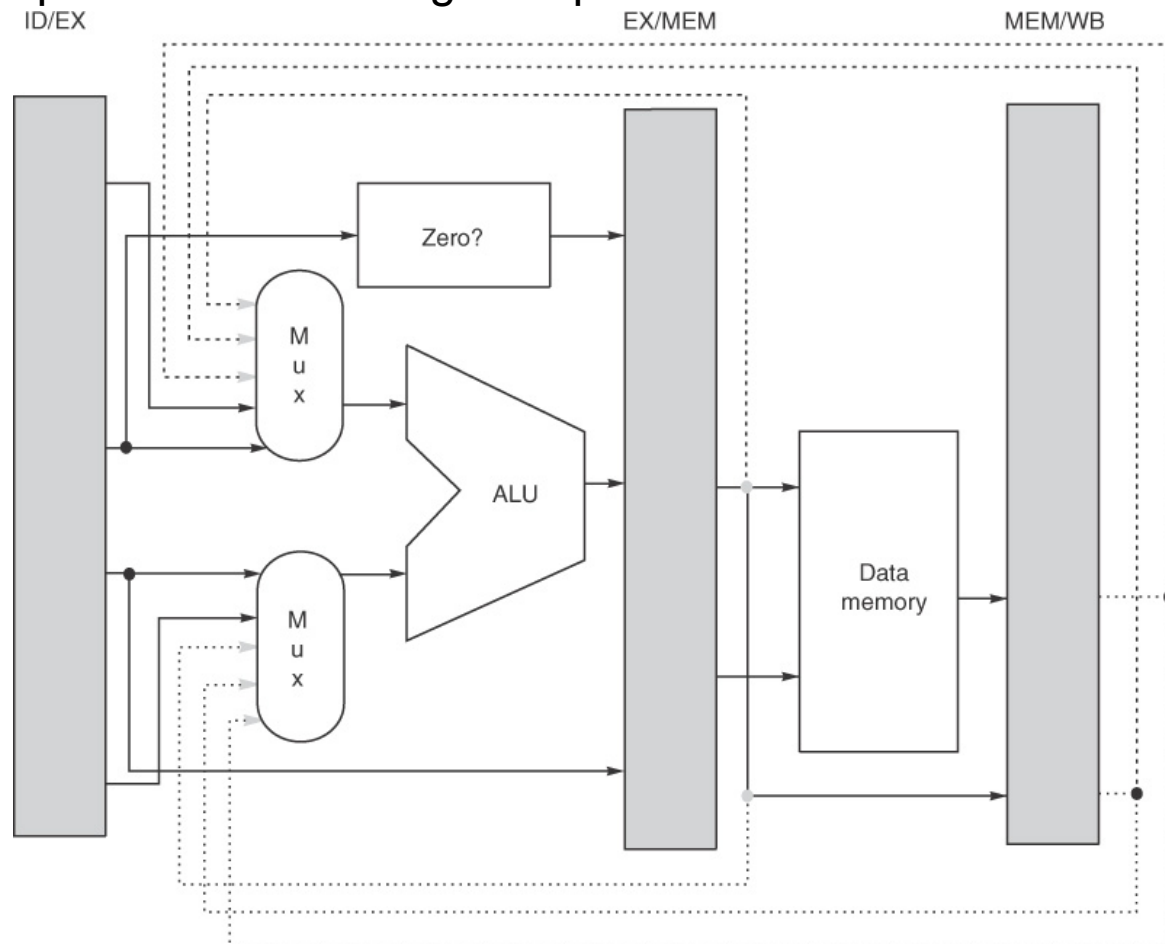
# Problem 6

- 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.

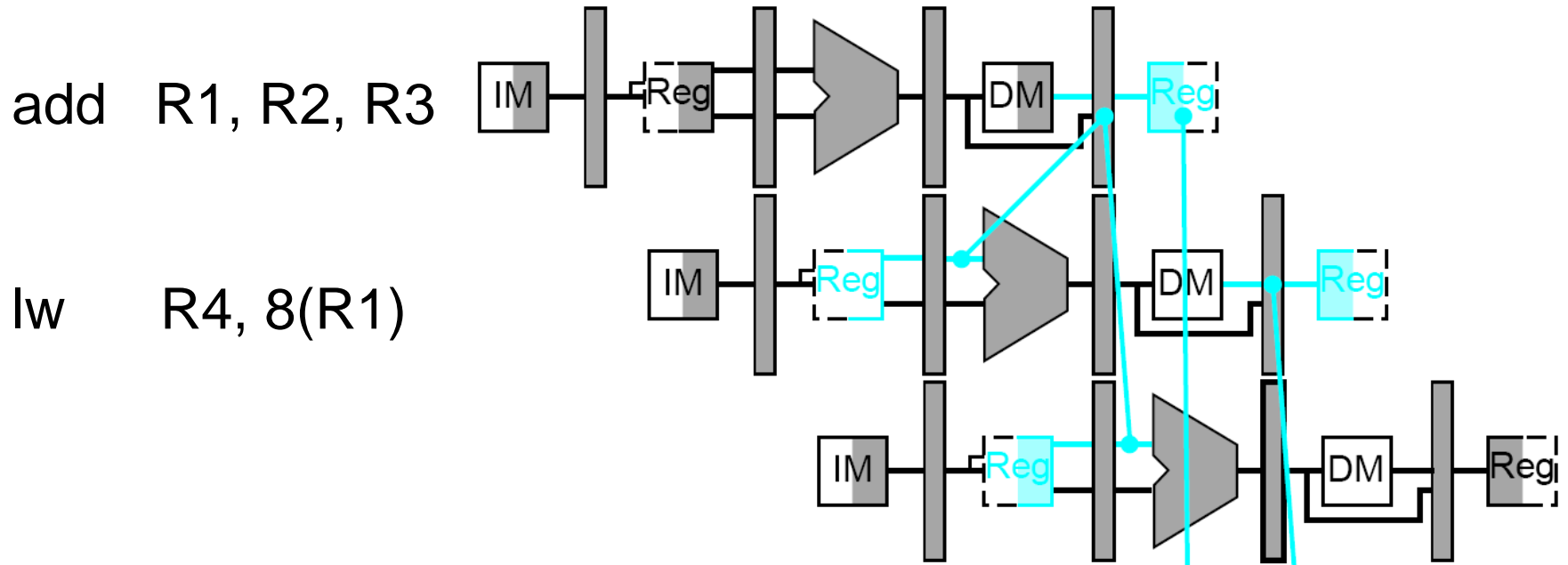
CYC-1	CYC-2	CYC-3	CYC-4	CYC-5	CYC-6	CYC-7	CYC-8
IF I1	IF I2	IF I3	IF I4	IF I5	IF	IF	IF
D/R	D/R I1	D/R I2	D/R I3	D/R I4	D/R	D/R	D/R
ALU	ALU	L3 L3 ALU I1	L4 L3 ALU I2	L5 L3 ALU I3	ALU	ALU	ALU
DM	DM	DM	DM I1	DM I2	DM I3	DM	DM
RW	RW	RW	RW	RW I1	RW I2	RW I3	RW

# Pipeline Implementation

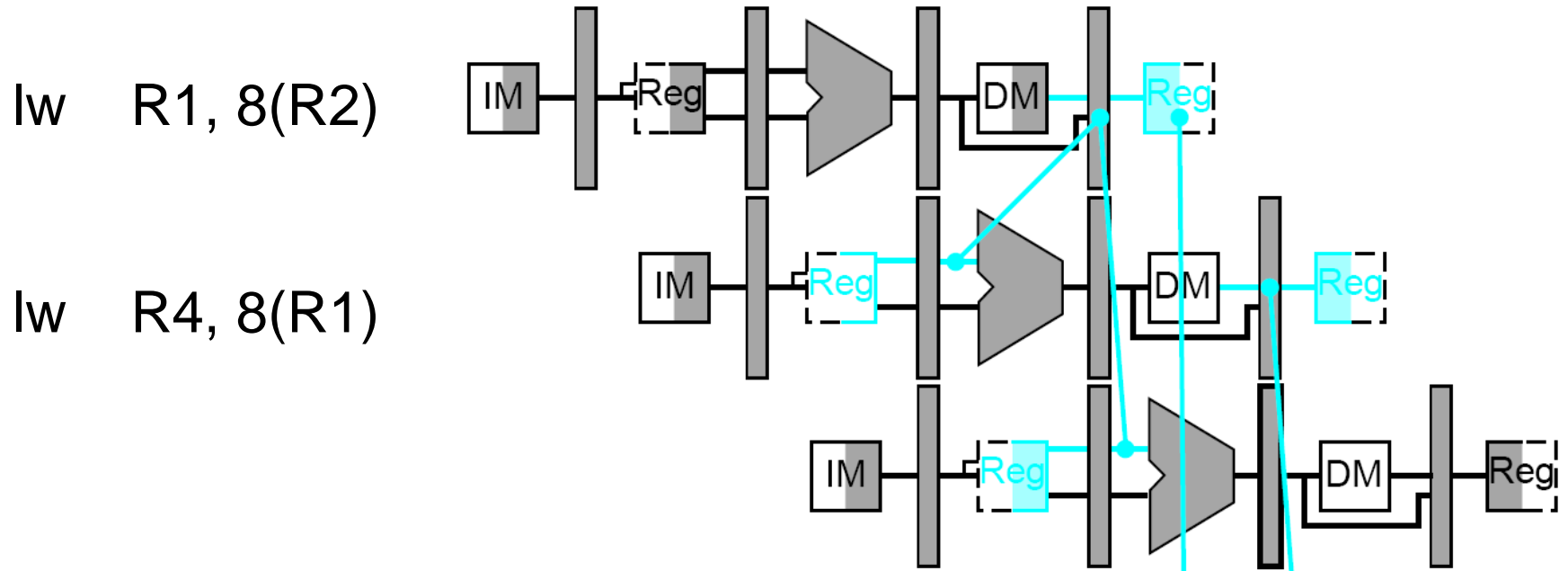
- Signals for the muxes have to be generated – some of this can happen during ID
- Need look-up tables to identify situations that merit bypassing/stalling – the number of inputs to the muxes goes up



# Example

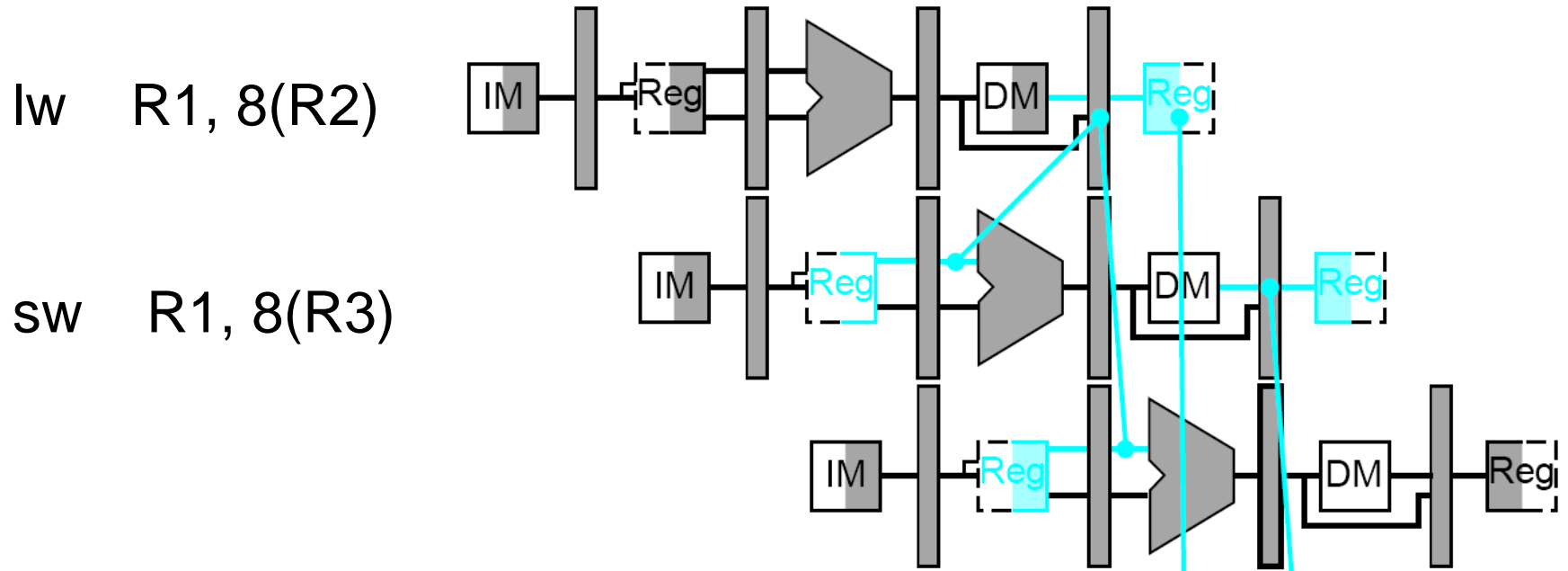


# Example





# Example



# Summary

---

- For the 5-stage pipeline, bypassing can eliminate delays between the following example pairs of instructions:

add/sub            R1, R2, R3  
add/sub/lw/sw    R4, R1, R5

lw            R1, 8(R2)  
sw            R1, 4(R3)

- The following pairs of instructions will have intermediate stalls:

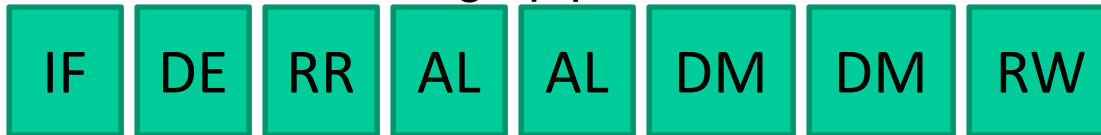
lw            R1, 8(R2)  
add/sub/lw    R3, R1, R4      or    sw    R3, 8(R1)

fmul        F1, F2, F3  
fadd        F5, F1, F4

# Problem 7

---

- Consider this 8-stage pipeline

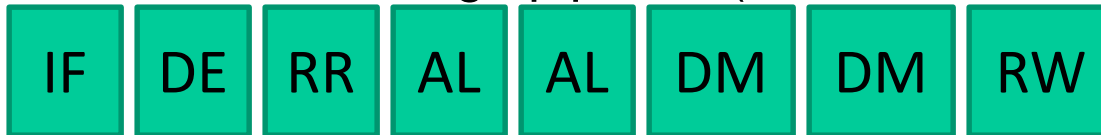


- For the following pairs of instructions, how many stalls will the 2<sup>nd</sup> instruction experience (with and without bypassing)?
  - ADD R1+R2→R3  
ADD R3+R4→R5
  - LD [R1]→R2  
ADD R2+R3→R4
  - LD [R1]→R2  
SD [R2]←R3
  - LD [R1]→R2  
SD [R3]←R2

# Problem 7

---

- Consider this 8-stage pipeline (RR and RW take a full cycle)



- For the following pairs of instructions, how many stalls will the 2<sup>nd</sup> instruction experience (with and without bypassing)?

- ADD R1+R2→R3  
ADD R3+R4→R5                      without: 5    with: 1
- LD [R1]→R2  
ADD R2+R3→R4                      without: 5    with: 3
- LD [R1]→R2  
SD [R2]←R3                          without: 5    with: 3
- LD [R1]→R2  
SD [R3]←R2                          without: 5    with: 1

# Title

---

- Bullet