Lecture: Pipelining Hazards

- Topics: structural and data hazards
- HW2 posted; due in a week

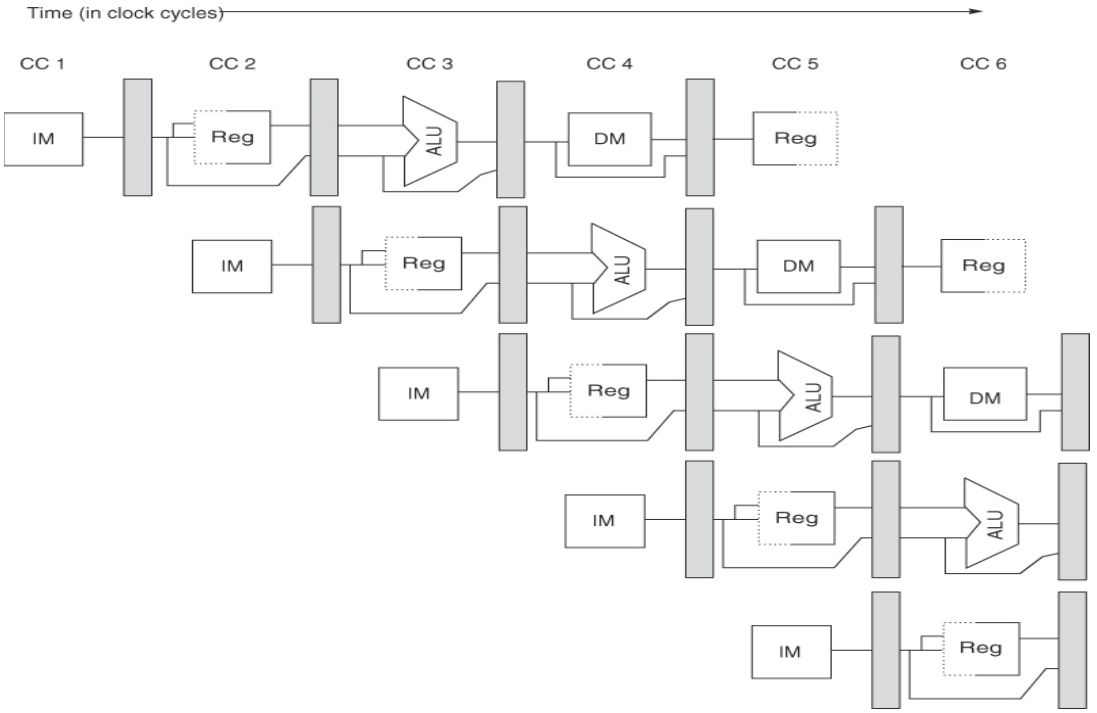
RISC/CISC Loads/Stores

Registers and memory
Complex and reduced instrs
Format of a load/store

Problem 3

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

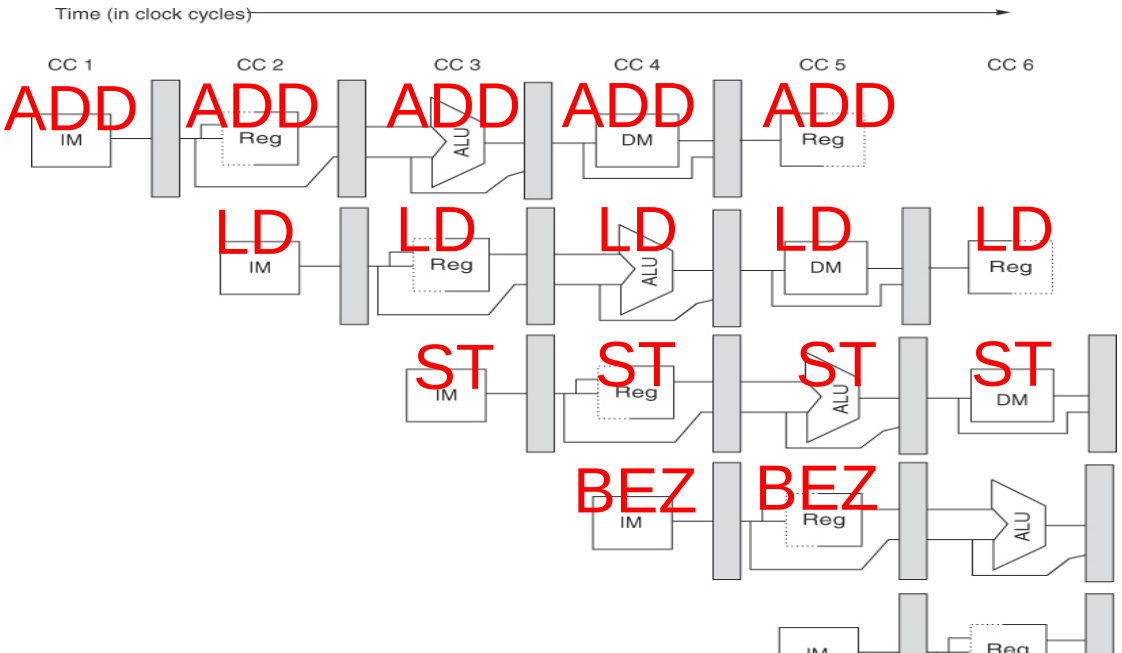
  ADD R3 ← R1, R2
  LD R7 ← 8[R6]
  ST R9 → 4[R8]
  BEZ R4, [R5]



Problem 3

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

  ADD R3 ← R1, R2
  LD R7 ← 8[R6]
  ST R9 → 4[R8]
  BEZ R4, [R5]



Pipeline Summary

	RR	ALU	DM	RW
ADD R3 ← R1, R2	Rd R1,R2	R1+R2	--	Wr R3
BEZ R1, [R5]	Rd R1, R5	--	--	--
	Compare, Set PC			
LD R6 ← 8[R3]	Rd R3	R3+8	Get data	Wr R6
ST R6 → 8[R3]	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  R2, [R1]   # R1 has the address for variable i
MUL  R3, R2, 8   # the offset from the start of the array
ADD  R7, R3, R4   # R4 has the address of a[0]
ADD  R8, R3, R5   # R5 has the address of b[0]
ADD  R9, R3, R6   # R6 has the address of c[0]
LD R10, [R8]        #  Bringing b[i]
LD R11, [R9]        #  Bringing c[i]
ADD R12, R11, R10   # Sum is in R12
ST  R12, [R7]       # Putting result in a[i]
```

Hazards

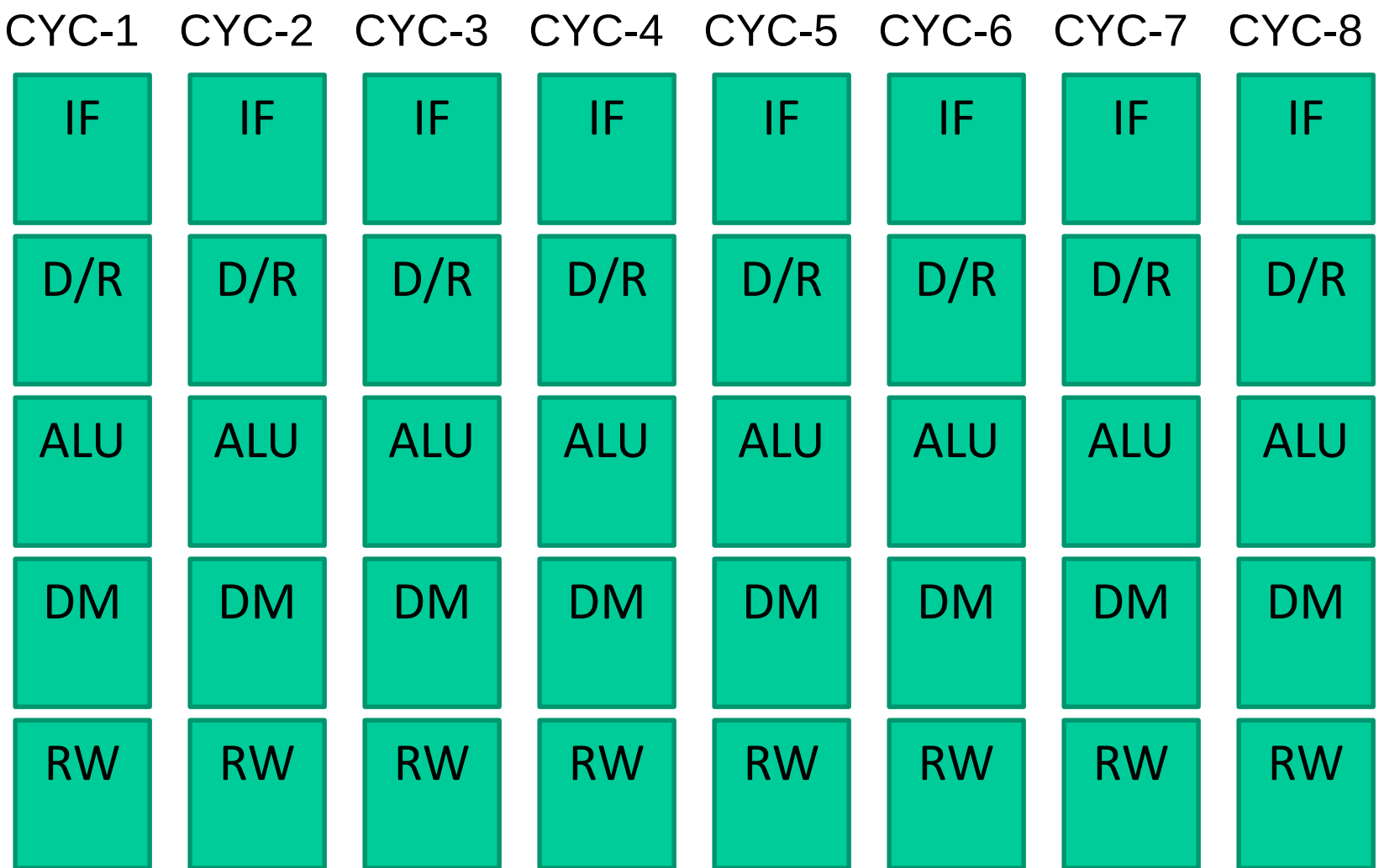
- Structural hazards: different instructions in different stages (or the same stage) conflicting for the same resource
- Data hazards: an instruction cannot continue because it needs a value that has not yet been generated by an earlier instruction
- Control hazard: fetch cannot continue because it does 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)

Problem 5

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

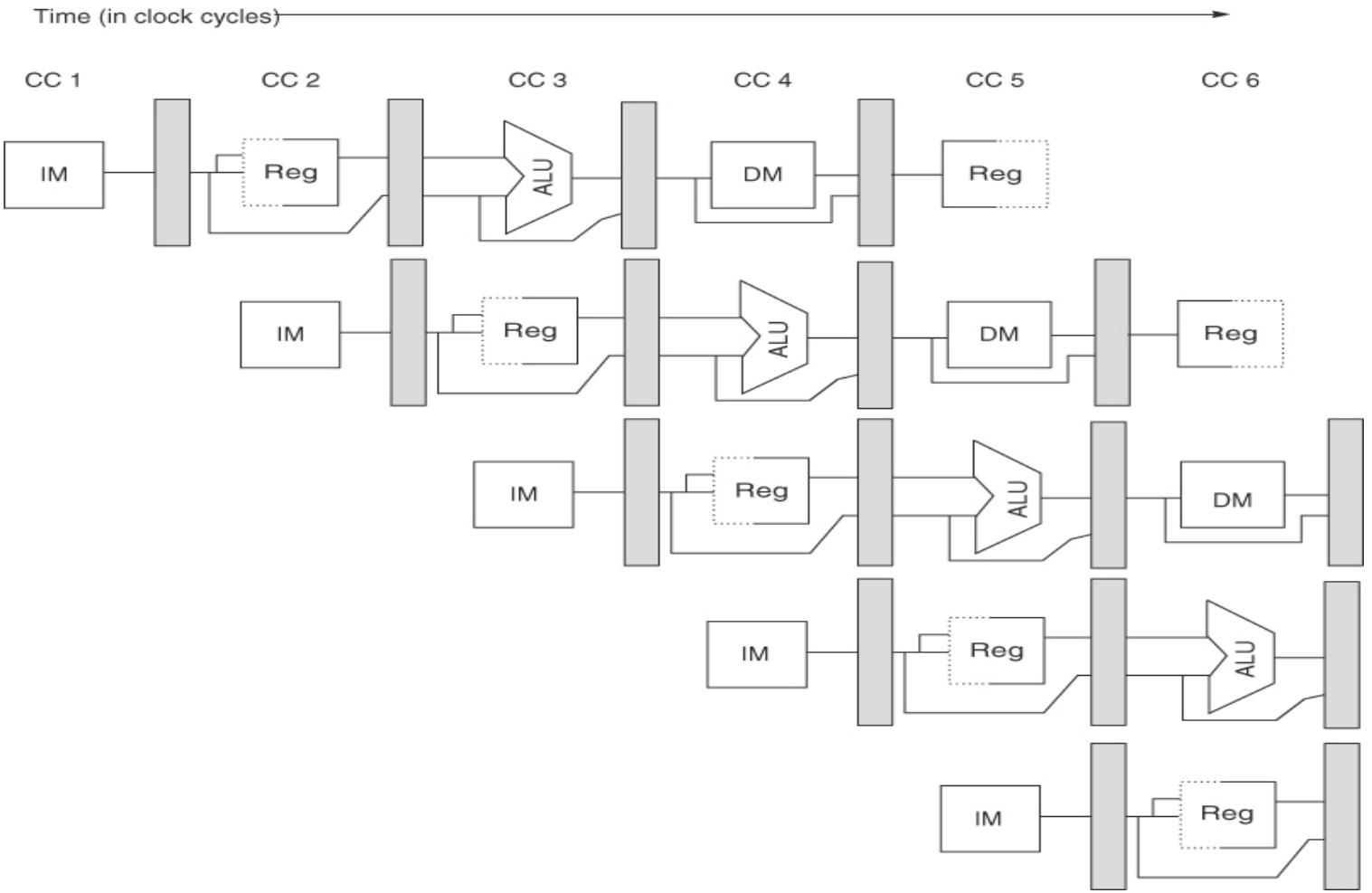


Problem 5

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

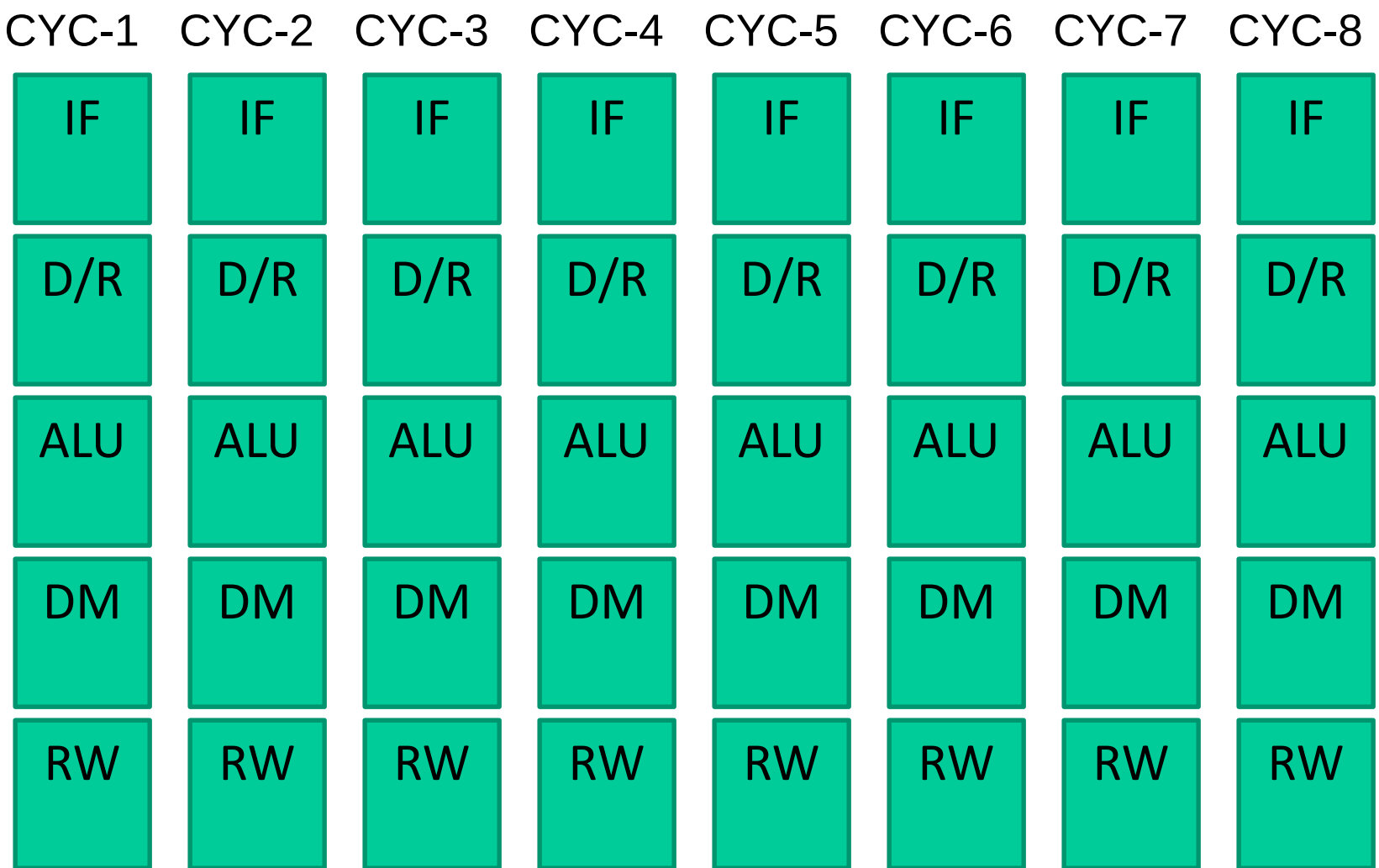
Bypassing: 5-Stage Pipeline



Source: H&P textbook

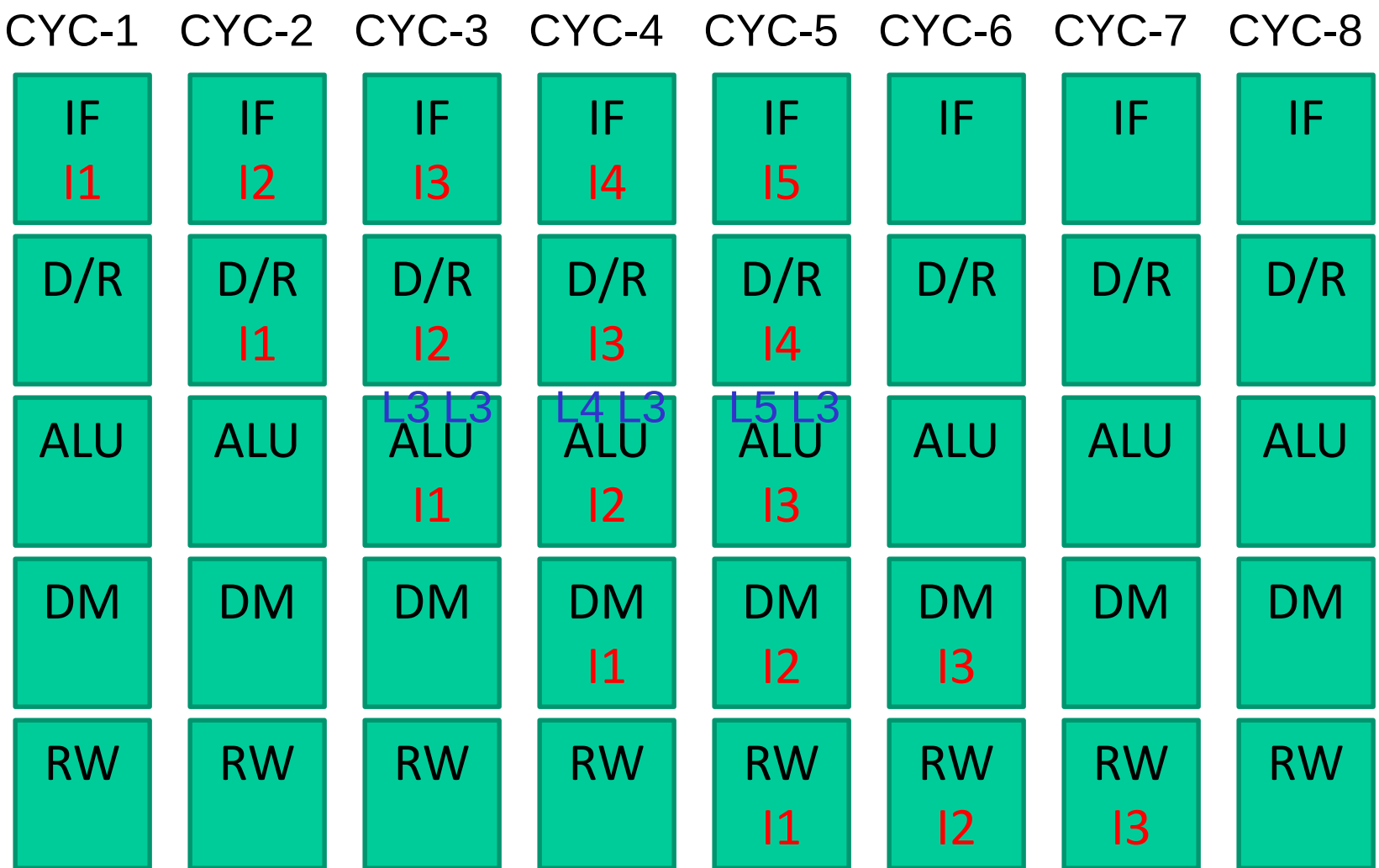
Problem 6

- 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.



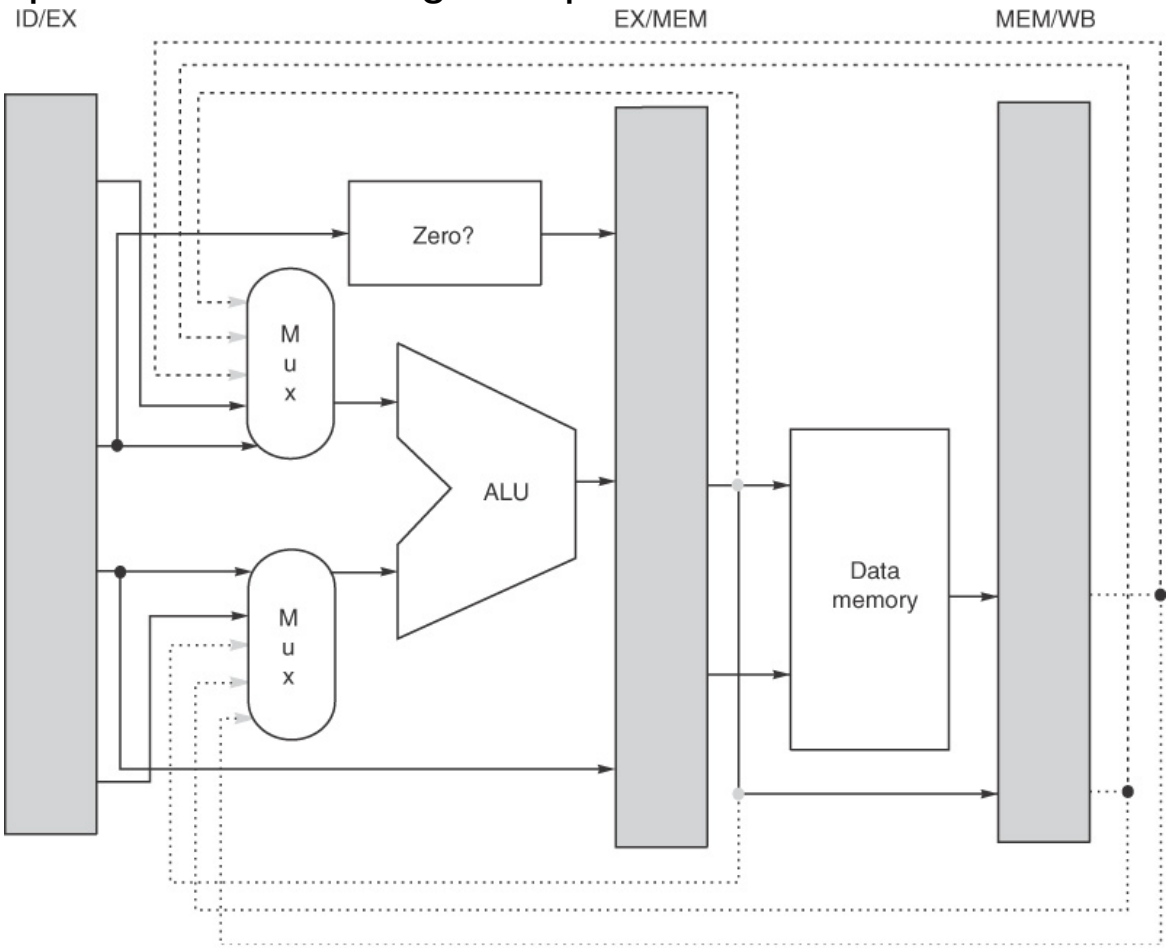
Problem 6

- 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.



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



Problem 7

- For the 5-stage pipeline (RR and RW take half a cycle)

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

  - ADD R3 ← R1+R2
    ADD R5 ← R3+R4
  - LD R2 ← [R1]
    ADD R4 ← R2+R3
  - LD R2 ← [R1]
    SD R3 → [R2]
  - LD R2 ← [R1]
    SD R2 → [R3]

Problem 7

- For the 5-stage pipeline (RR and RW take half a cycle)

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

  - ADD R3 ← R1+R2
    ADD R5 ← R3+R4 — without: 2 with: 0
  - LD R2 ← [R1]
    ADD R4 ← R2+R3 — without: 2 with: 1
  - LD R2 ← [R1]
    SD R3 → [R2] — without: 2 with: 1
  - LD R2 ← [R1]
    SD R2 → [R3] — without: 2 with: 0

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 8

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

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

  - ADD R3 ← R1+R2
    ADD R5 ← R3+R4
  - LD R2 ← [R1]
    ADD R4 ← R2+R3
  - LD R2 ← [R1]
    SD R3 → [R2]
  - LD R2 ← [R1]
    SD R2 → [R3]

Problem 8

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

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

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

Title

- Bullet