

Lecture 18: Pipelining

- Today's topics:
 - Hazards and instruction scheduling
 - Branch prediction
 - Out-of-order execution

Example 5

- 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.

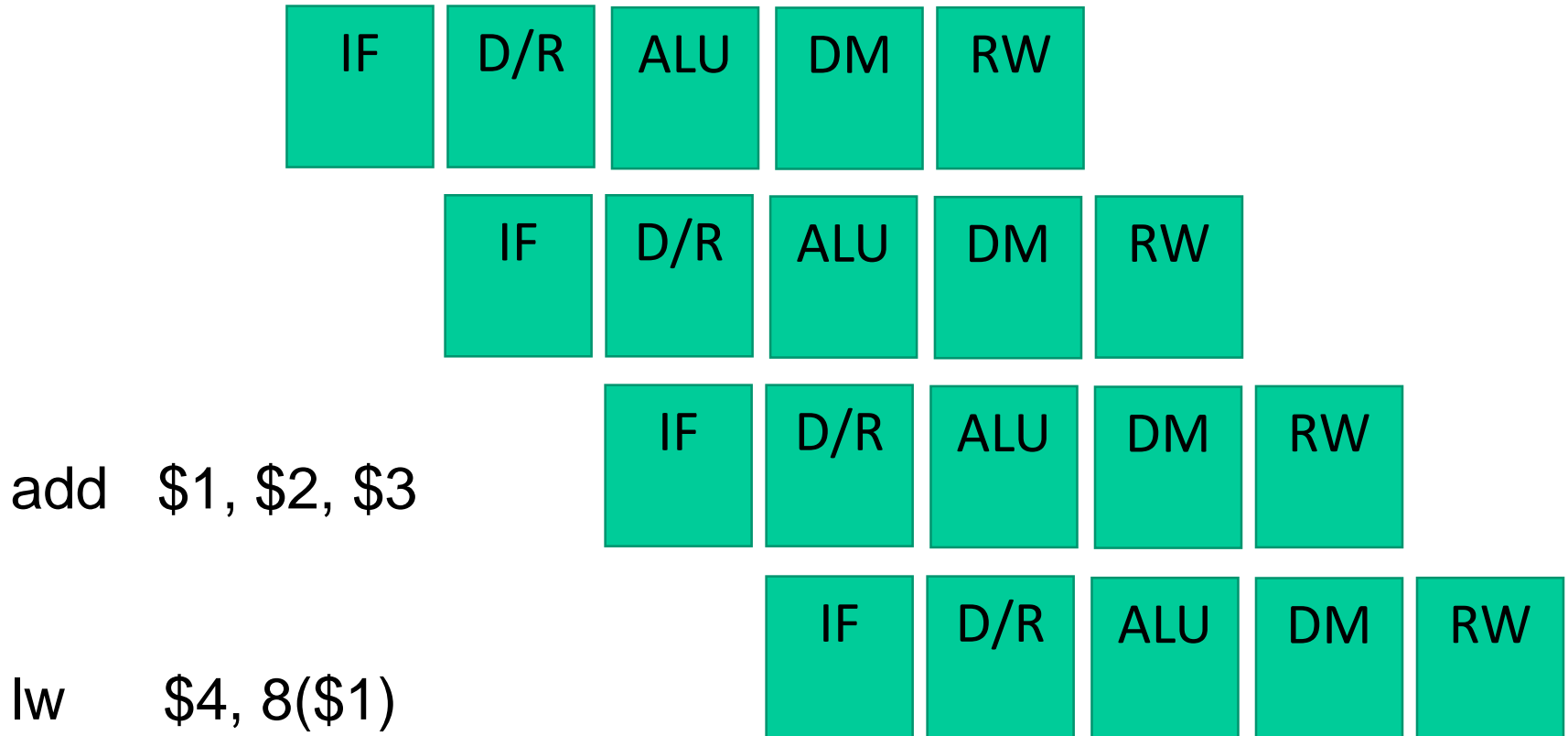
CYC-1	CYC-2	CYC-3	CYC-4	CYC-5	CYC-6	CYC-7	CYC-8
IF	IF	IF	IF	IF	IF	IF	IF
D/R	D/R	D/R	D/R	D/R	D/R	D/R	D/R
ALU	ALU	ALU	ALU	ALU	ALU	ALU	ALU
DM	DM	DM	DM	DM	DM	DM	DM
RW	RW	RW	RW	RW	RW	RW	RW

Example 5

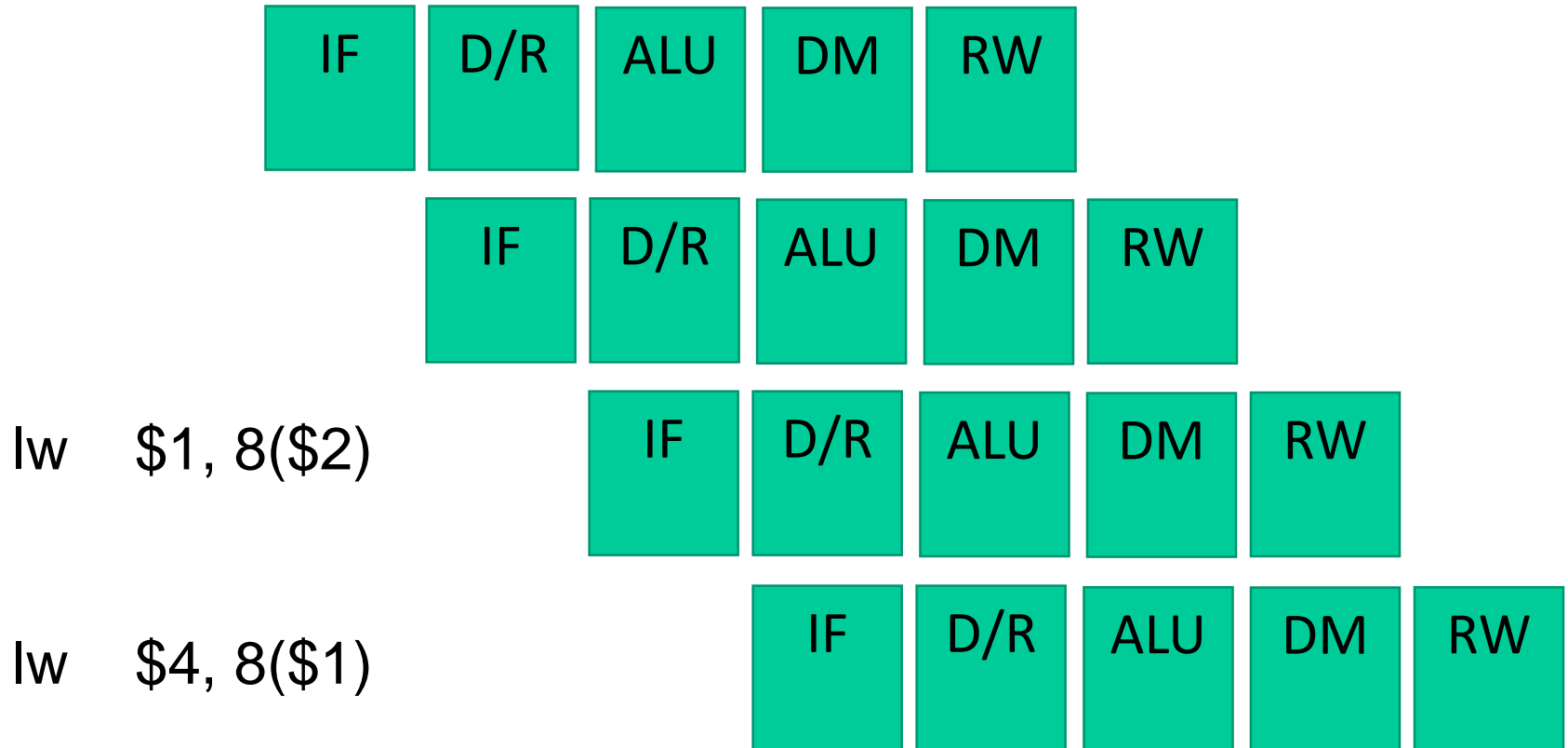
- Show the instruction occupying each stage in each cycle (with bypassing) if I1 is $R1+R2 \rightarrow R3$ and I2 is $R3+R4 \rightarrow R5$ and I3 is $R3+R8 \rightarrow R9$. Identify the input latch for each input operand.

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

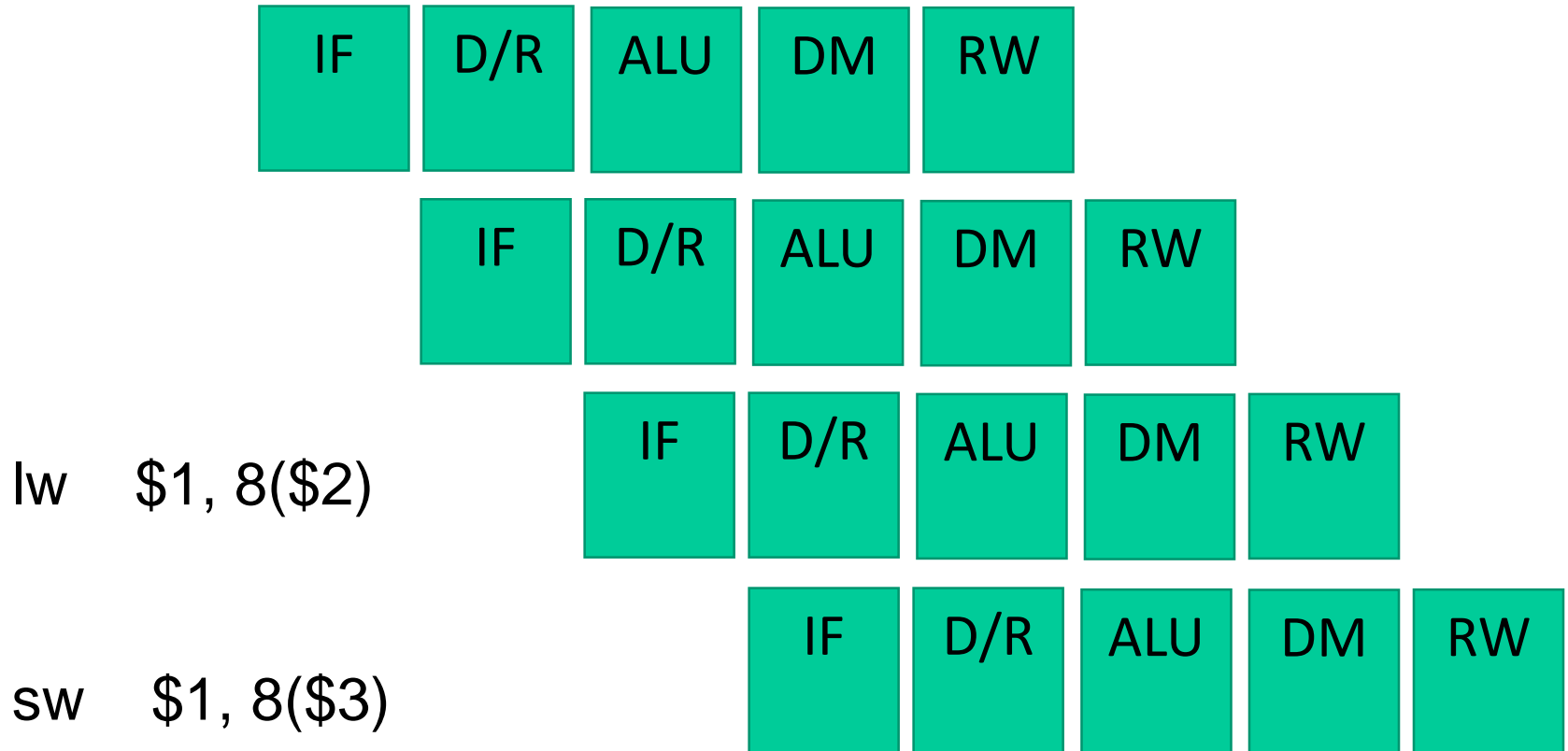
Example 1



Example 2



Example 3

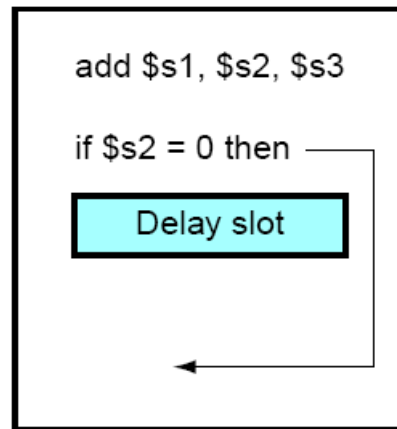


Control Hazards

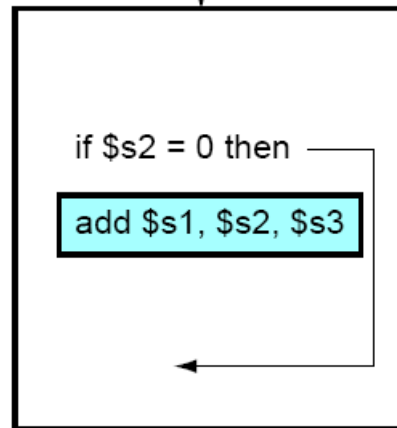
- Simple techniques to handle control hazard stalls:
 - for every branch, introduce a stall cycle (note: every 6th instruction is a branch!)
 - assume the branch is not taken and start fetching the next instruction – if the branch is taken, need hardware to cancel the effect of the wrong-path instruction
 - fetch the next instruction (branch delay slot) and execute it anyway – if the instruction turns out to be on the correct path, useful work was done – if the instruction turns out to be on the wrong path, hopefully program state is not lost
 - make a smarter guess and fetch instructions from the expected target

Branch Delay Slots

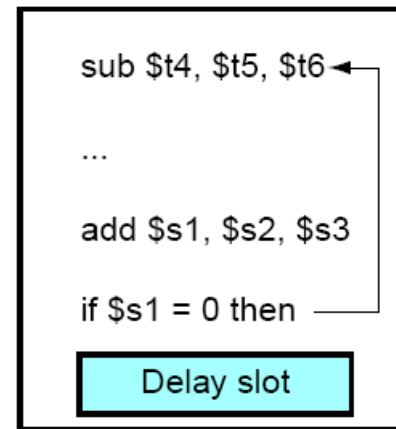
a. From before



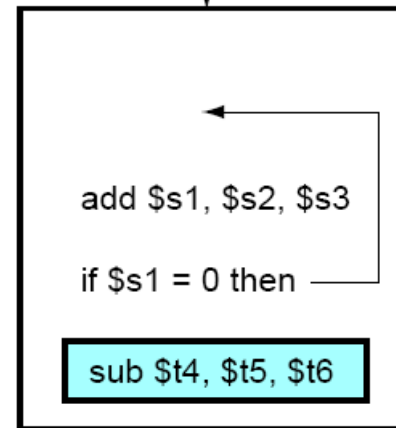
Becomes



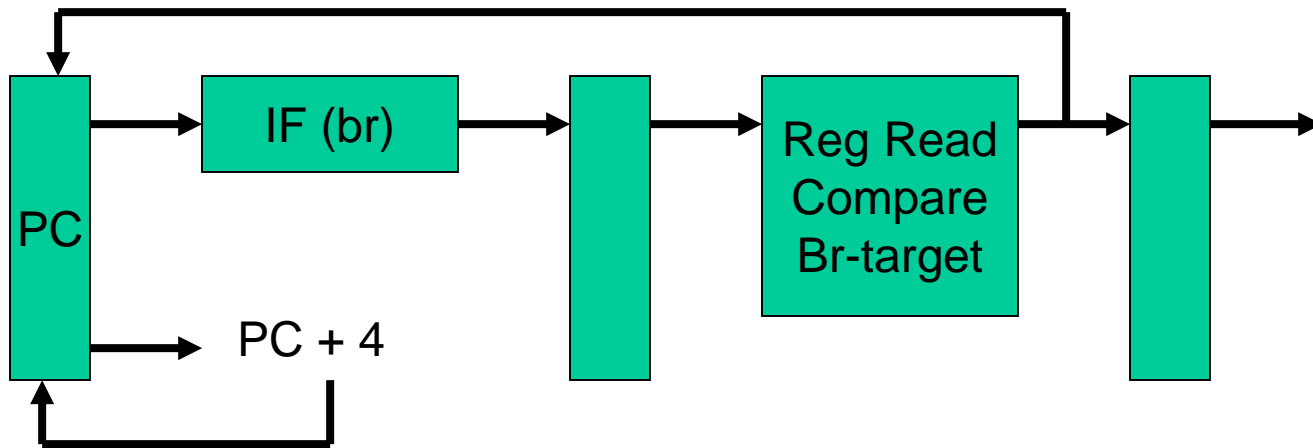
b. From target



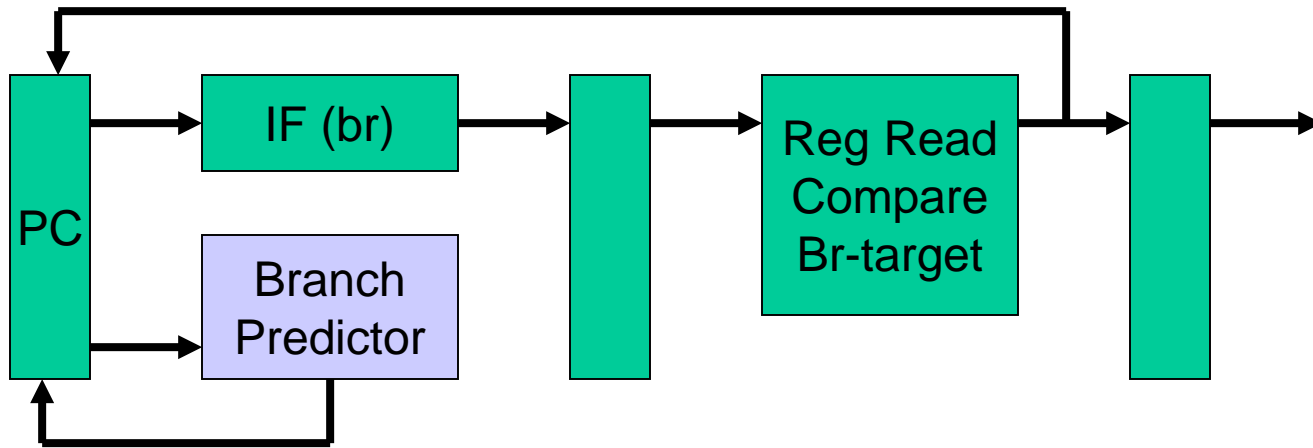
Becomes



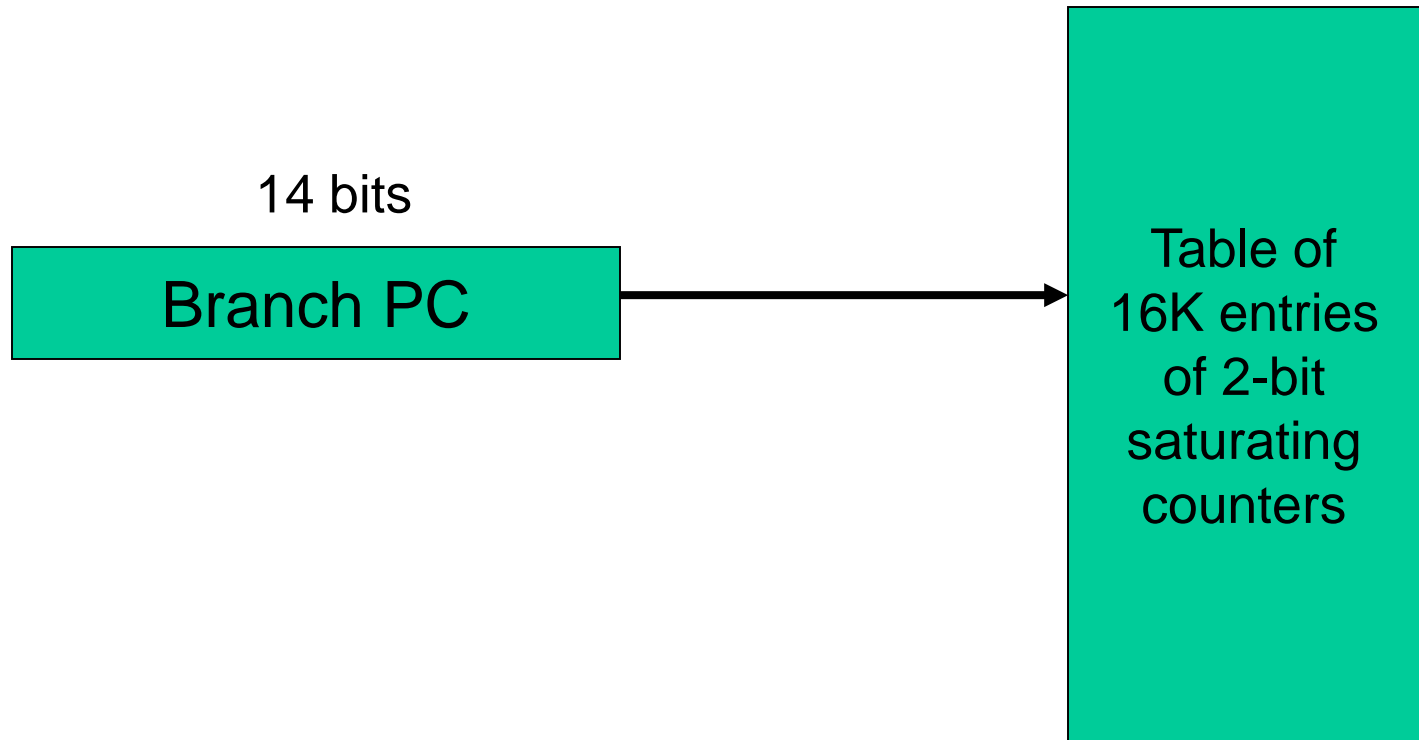
Pipeline without Branch Predictor



Pipeline with Branch Predictor



Bimodal Predictor



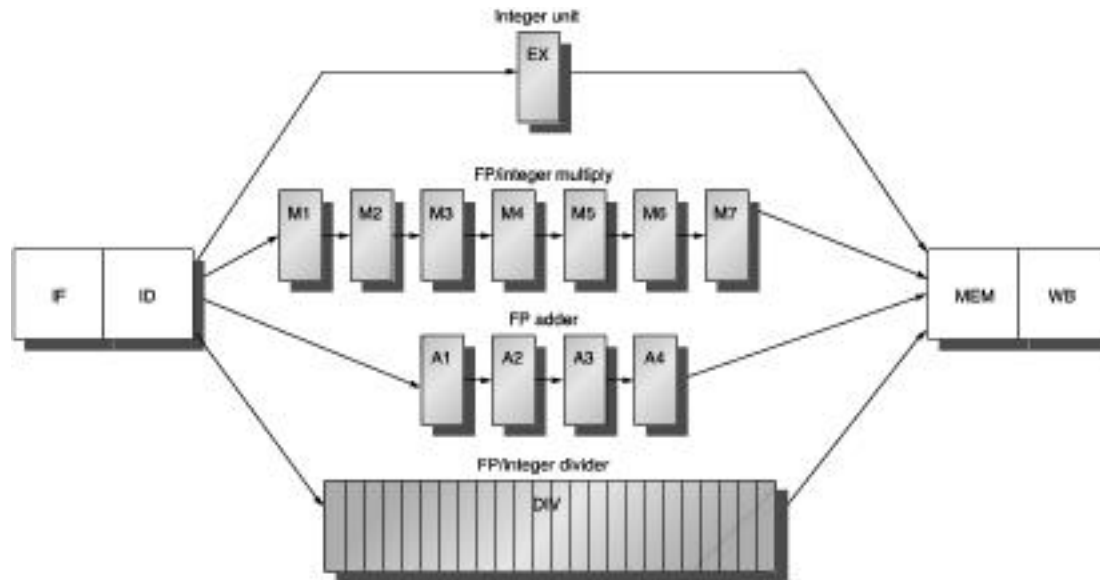
2-Bit Prediction

- For each branch, maintain a 2-bit saturating counter:
if the branch is taken: $\text{counter} = \min(3, \text{counter} + 1)$
if the branch is not taken: $\text{counter} = \max(0, \text{counter} - 1)$
... sound familiar?
- If $(\text{counter} \geq 2)$, predict taken, else predict not taken
- The counter attempts to capture the common case for each branch

Slowdowns from Stalls

- Perfect pipelining with no hazards \rightarrow an instruction completes every cycle (total cycles \sim num instructions)
 \rightarrow speedup = increase in clock speed = num pipeline stages
- With hazards and stalls, some cycles (= stall time) go by during which no instruction completes, and then the stalled instruction completes
- Total cycles = number of instructions + stall cycles

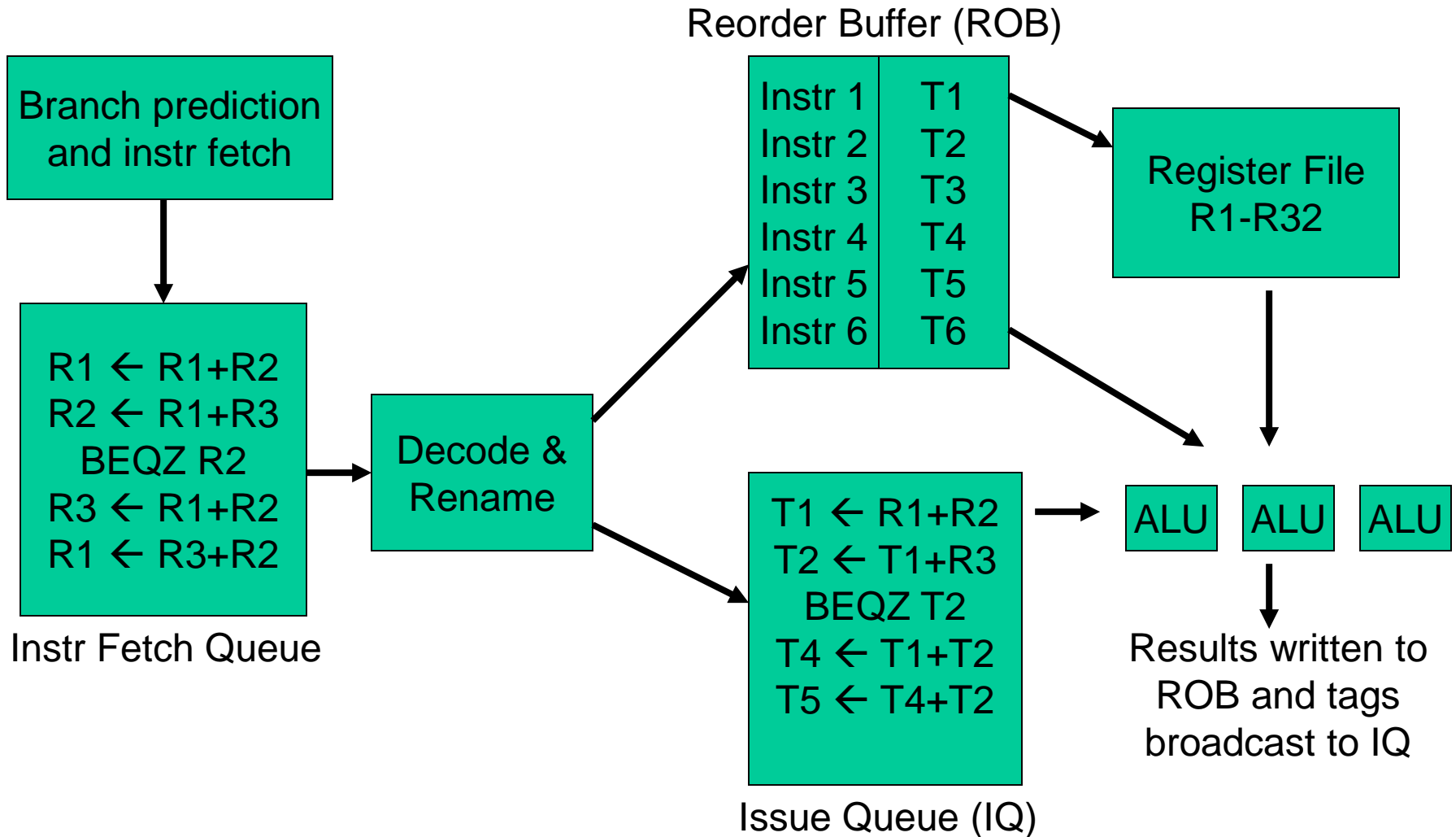
Multicycle Instructions



© 2003 Elsevier Science (USA). All rights reserved.

- Multiple parallel pipelines – each pipeline can have a different number of stages
- Instructions can now complete out of order – must make sure that writes to a register happen in the correct order

An Out-of-Order Processor Implementation



Example Code

Completion times	with in-order	with ooo
ADD R1, R2, R3	5	5
ADD R4, R1, R2	6	6
LW R5, 8(R4)	7	7
ADD R7, R6, R5	9	9
ADD R8, R7, R5	10	10
LW R9, 16(R4)	11	7
ADD R10, R6, R9	13	9
ADD R11, R10, R9	14	10

Title

- Bullet