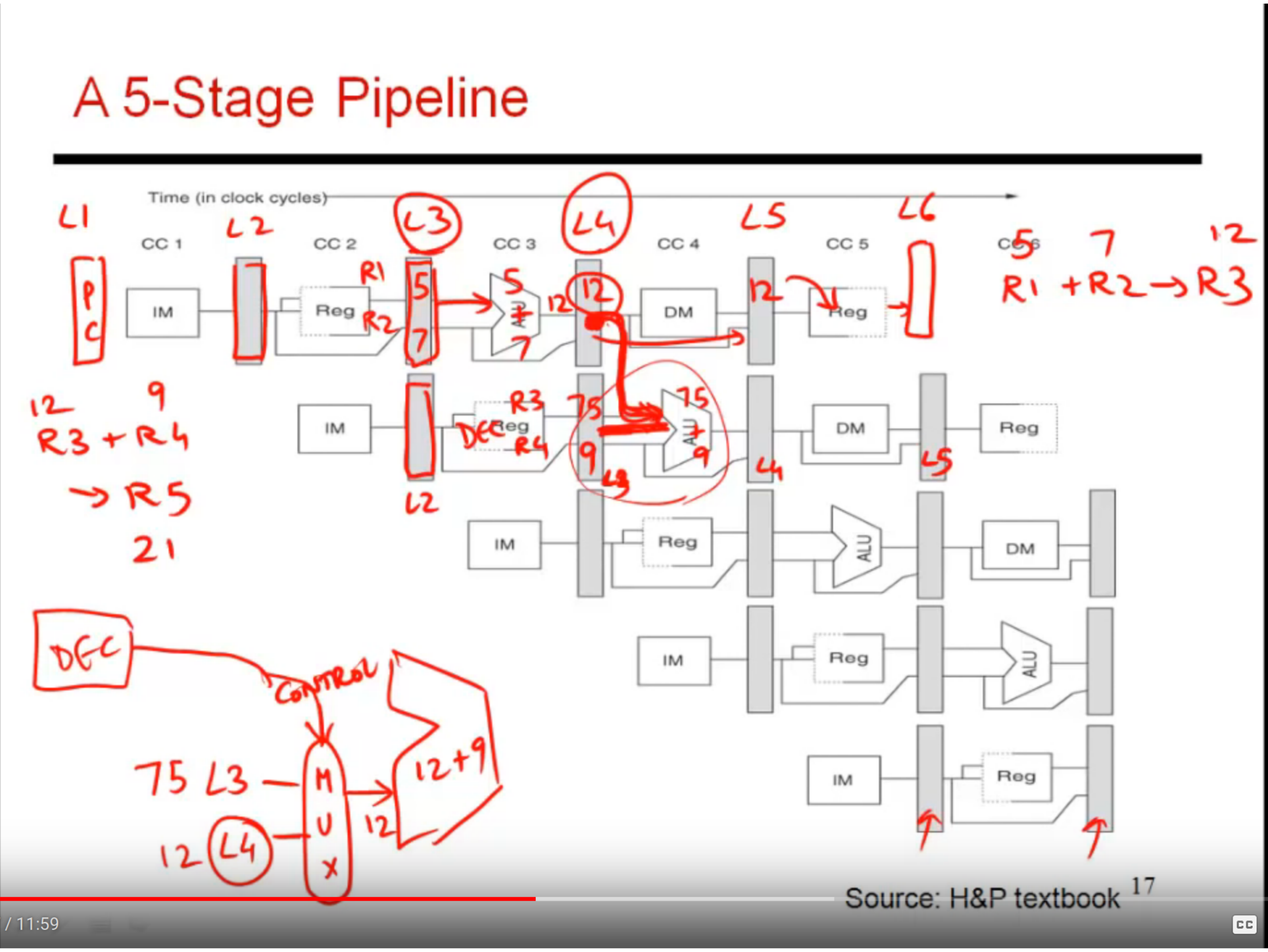
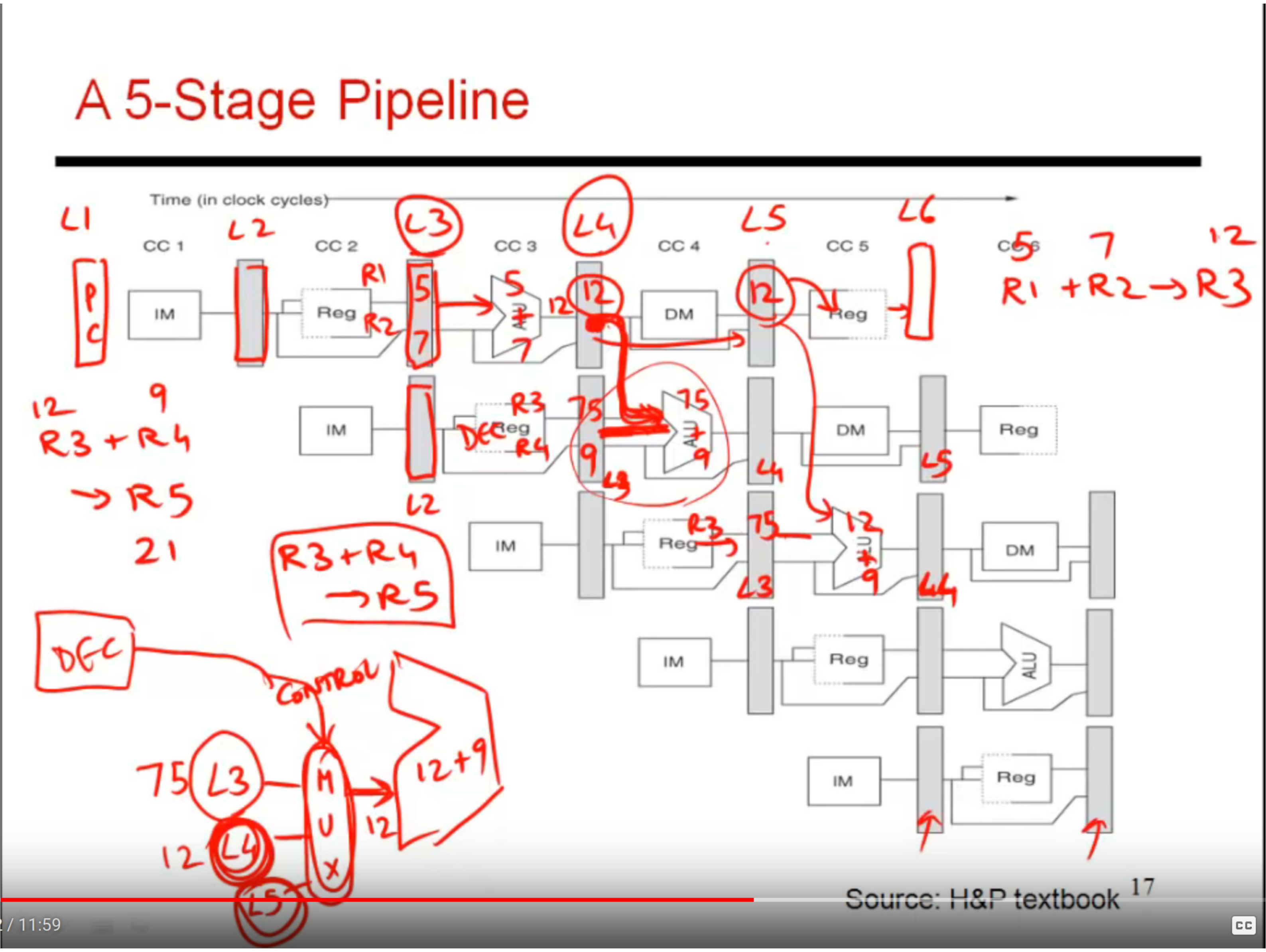


A 5-Stage Pipeline

Source: H&P textbook 17

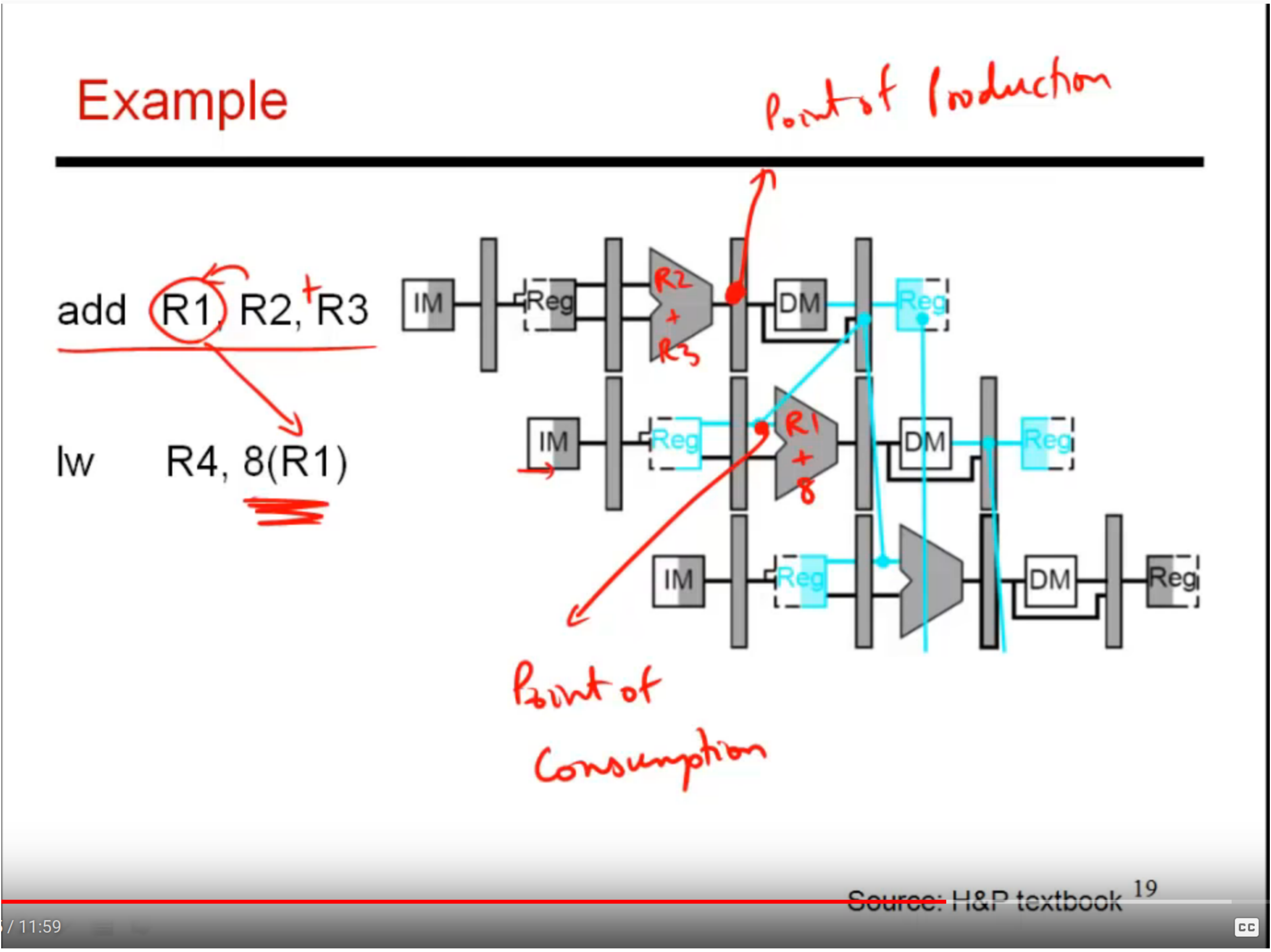
/ 11:59



A 5-Stage Pipeline

Source: H&P textbook 17

/ 11:59

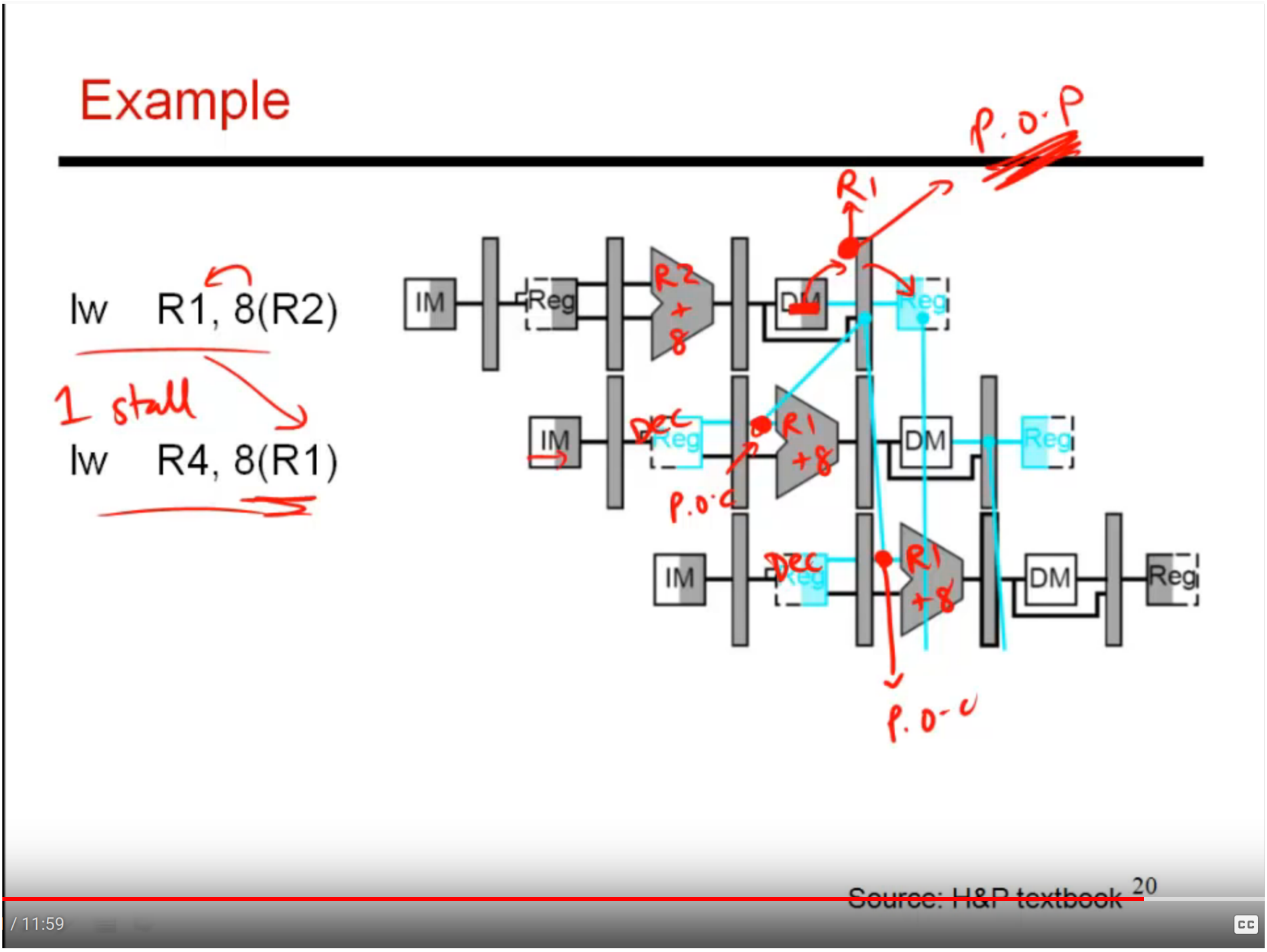


Example

add R1, R2, + R3

lw R4, 8(R1)

Source: H&P textbook 19



Example

lw R1, 8(R2)

1 stall

lw R4, 8(R1)

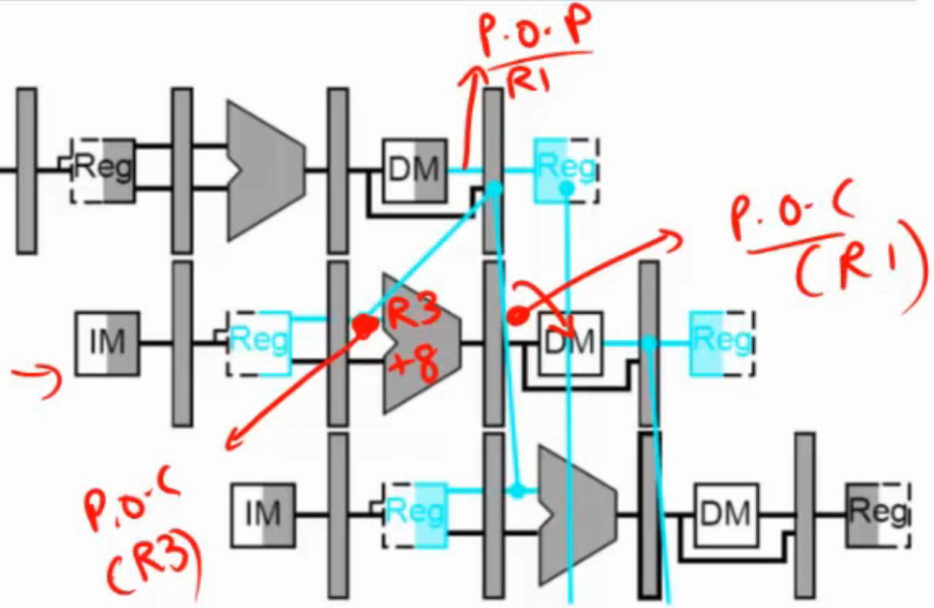
Source: H&P textbook

Example

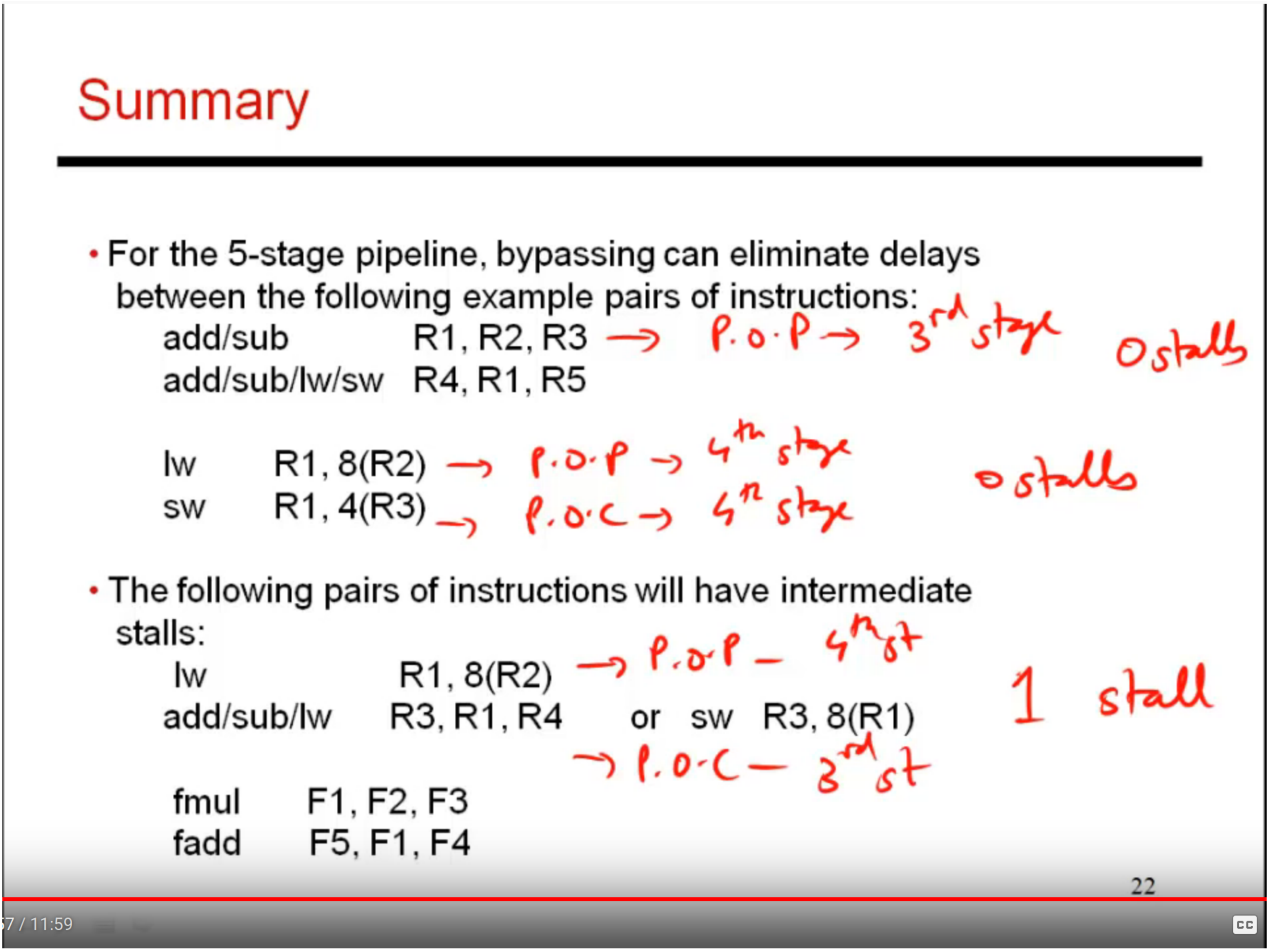
lw R1, 8(R2)

0 stalls

sw R1, 8(R3)



Source: H&P textbook



Summary

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

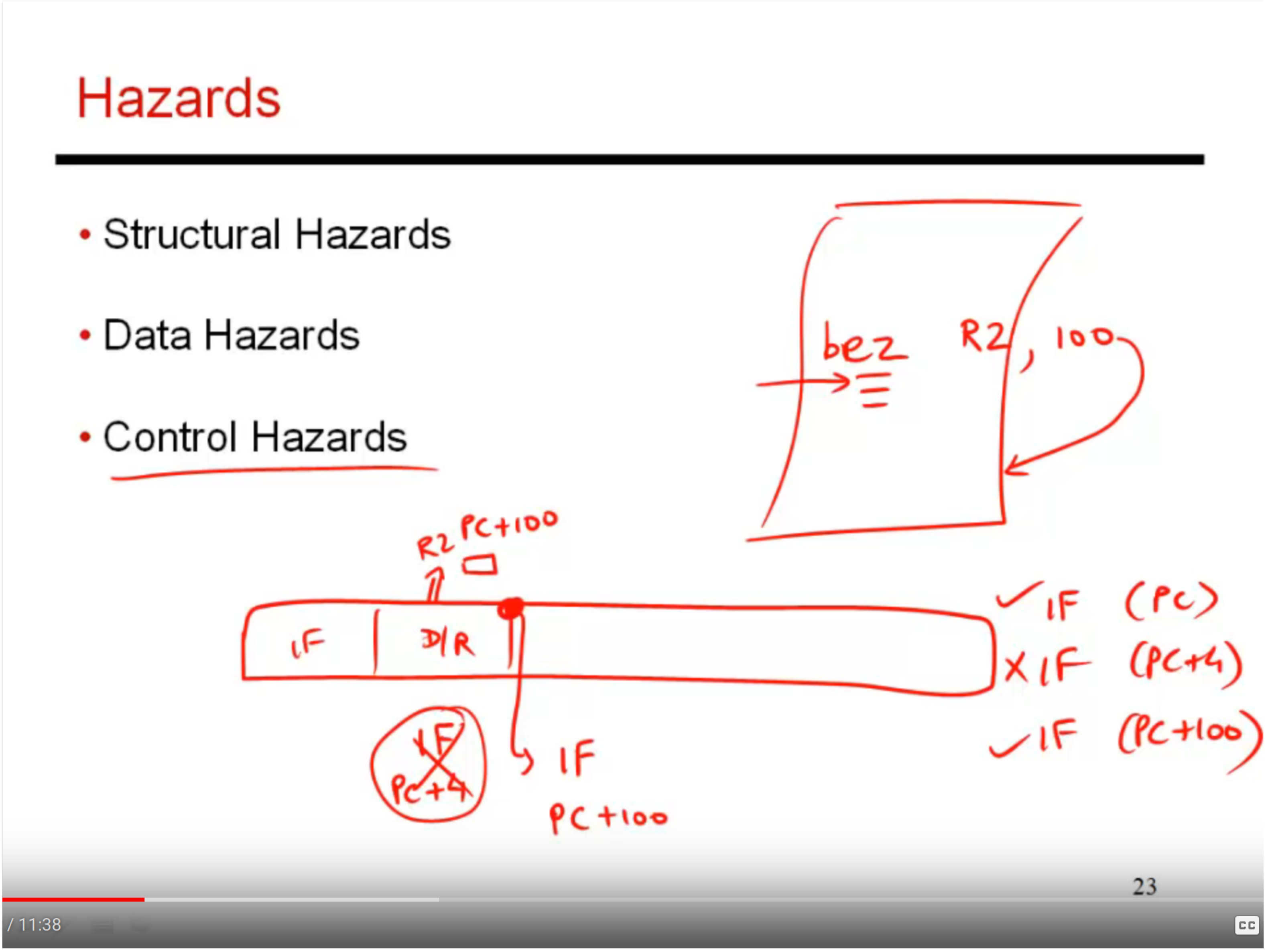
add/sub R1, R2, R3 → P.O.P → 3rd stage 0 stalls
add/sub/lw/sw R4, R1, R5

lw R1, 8(R2) → P.O.P → 4th stage 0 stalls
sw R1, 4(R3) → P.O.C → 4th stage

- The following pairs of instructions will have intermediate stalls:

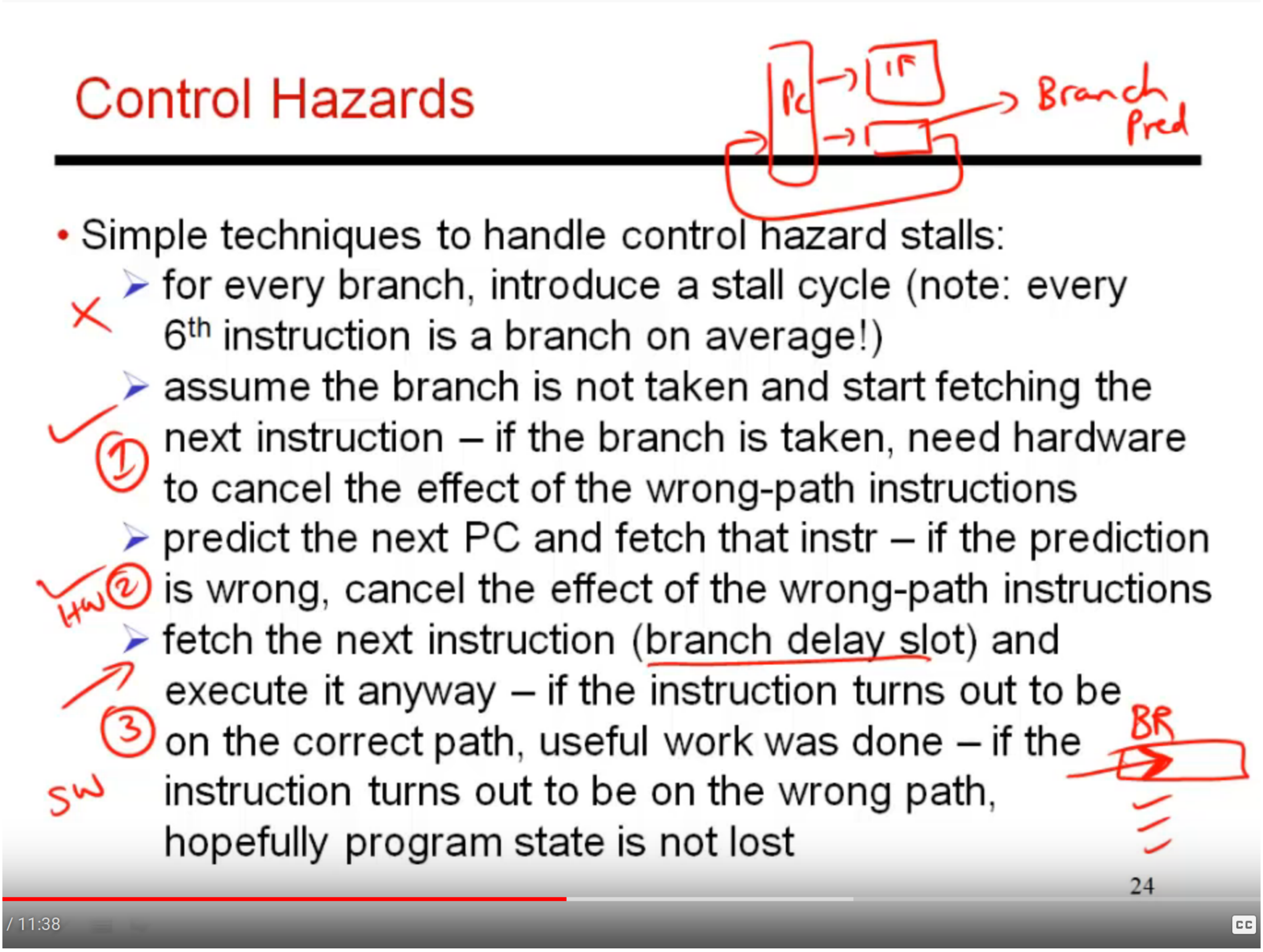
lw R1, 8(R2) → P.O.P – 4th st 1 stall
add/sub/lw R3, R1, R4 or sw R3, 8(R1)
→ P.O.C – 3rd st

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



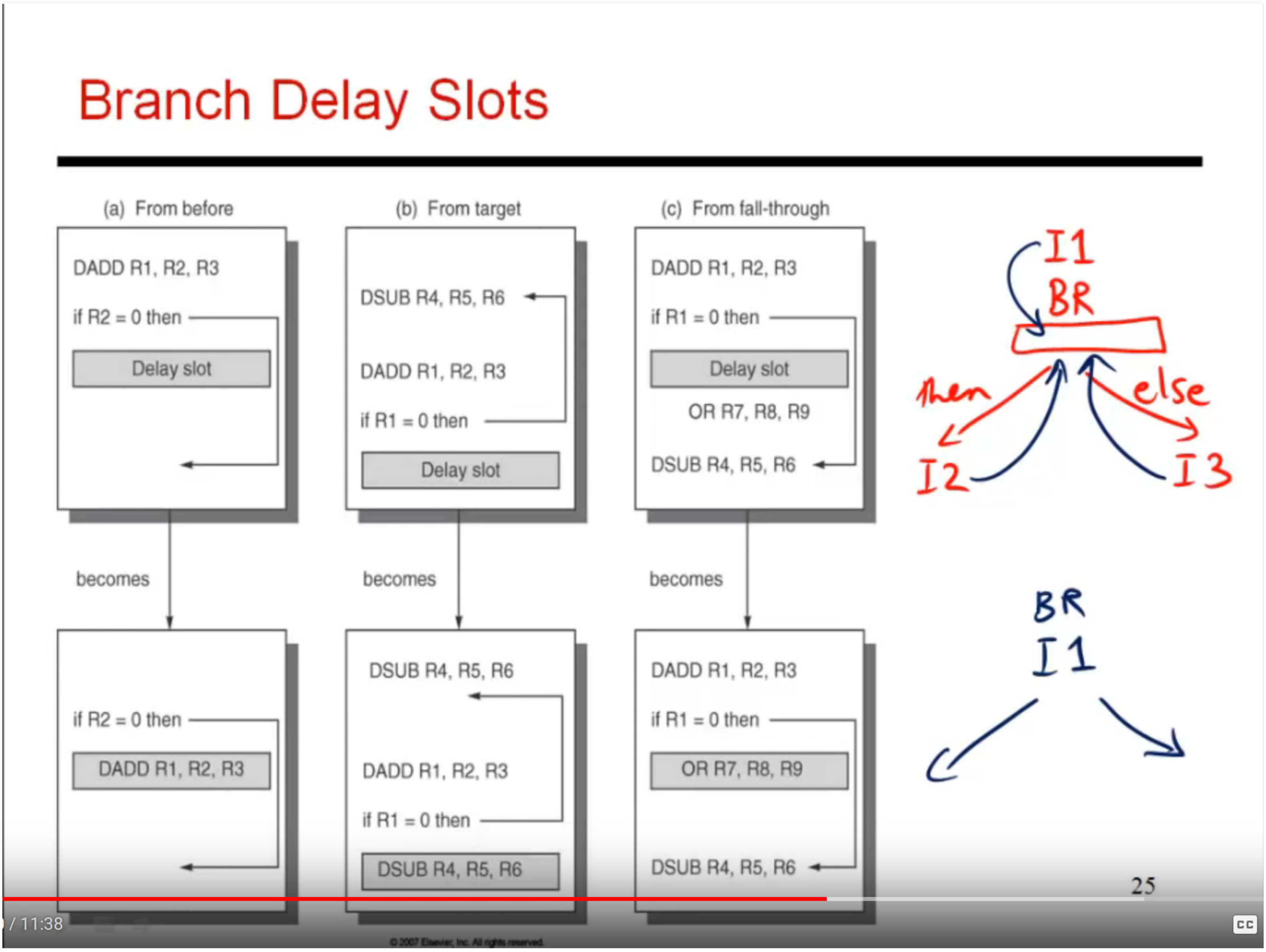
Hazards

- Structural Hazards
- Data Hazards
- Control Hazards



Control Hazards

- Simple techniques to handle control hazard stalls:
  - for every branch, introduce a stall cycle (note: every 6th instruction is a branch on average!)
  - 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 instructions
  - predict the next PC and fetch that instr – if the prediction is wrong, cancel the effect of the wrong-path instructions
  - 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



Branch Delay Slots

(a) From before

DADD R1, R2, R3

if R2 = 0 then

Delay slot

becomes

if R2 = 0 then

DADD R1, R2, R3

(b) From target

DSUB R4, R5, R6

DADD R1, R2, R3

if R1 = 0 then

Delay slot

becomes

DSUB R4, R5, R6

DADD R1, R2, R3

if R1 = 0 then

DSUB R4, R5, R6

(c) From fall-through

DADD R1, R2, R3

if R1 = 0 then

Delay slot

OR R7, R8, R9

DSUB R4, R5, R6

becomes

DADD R1, R2, R3

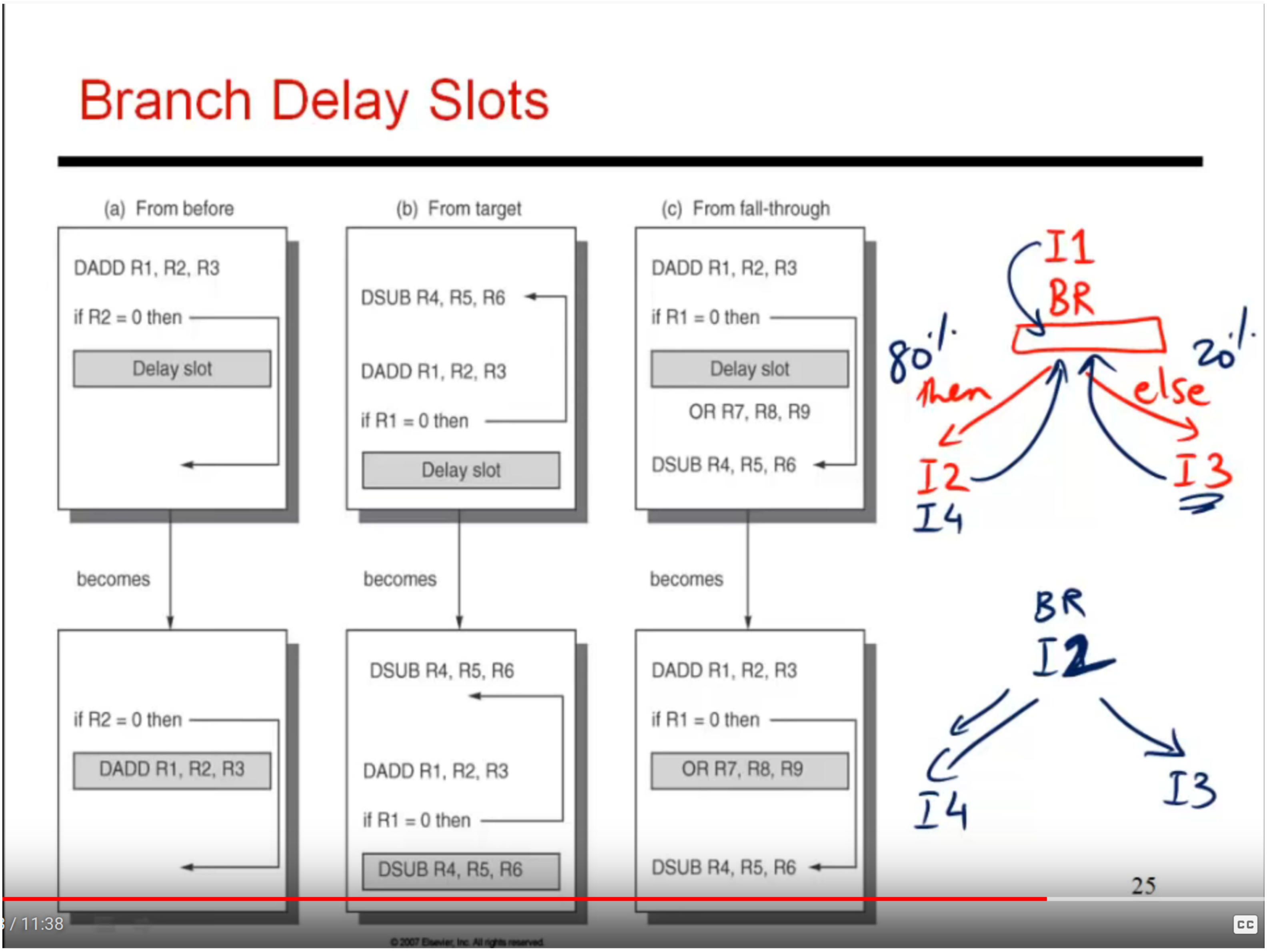
if R1 = 0 then

OR R7, R8, R9

DSUB R4, R5, R6

25

© 2007 Elsevier, Inc. All rights reserved.



Branch Delay Slots

(a) From before

```
DADD R1, R2, R3
if R2 = 0 then
Delay slot
```

becomes

```
if R2 = 0 then
DADD R1, R2, R3
```

(b) From target

```
DSUB R4, R5, R6
DADD R1, R2, R3
if R1 = 0 then
Delay slot
```

becomes

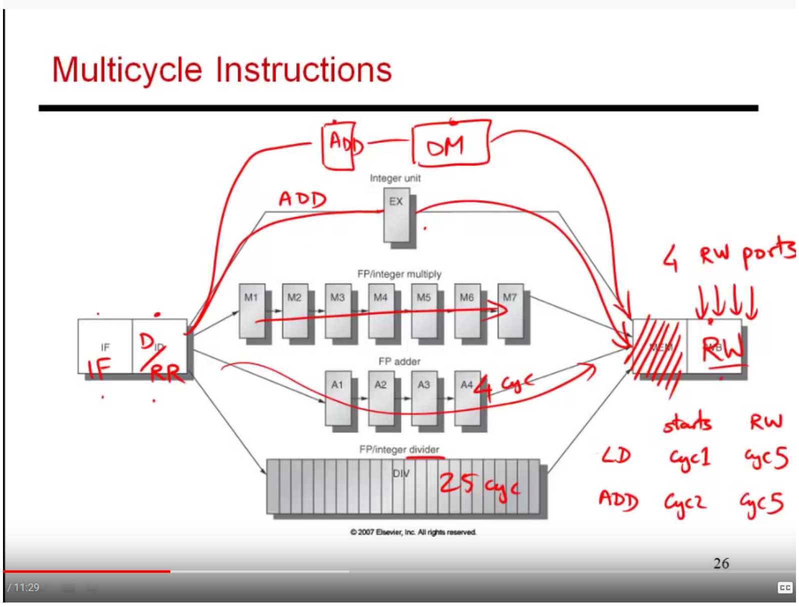
```
DSUB R4, R5, R6
DADD R1, R2, R3
if R1 = 0 then
DSUB R4, R5, R6
```

(c) From fall-through

```
DADD R1, R2, R3
if R1 = 0 then
Delay slot
OR R7, R8, R9
DSUB R4, R5, R6
```

becomes

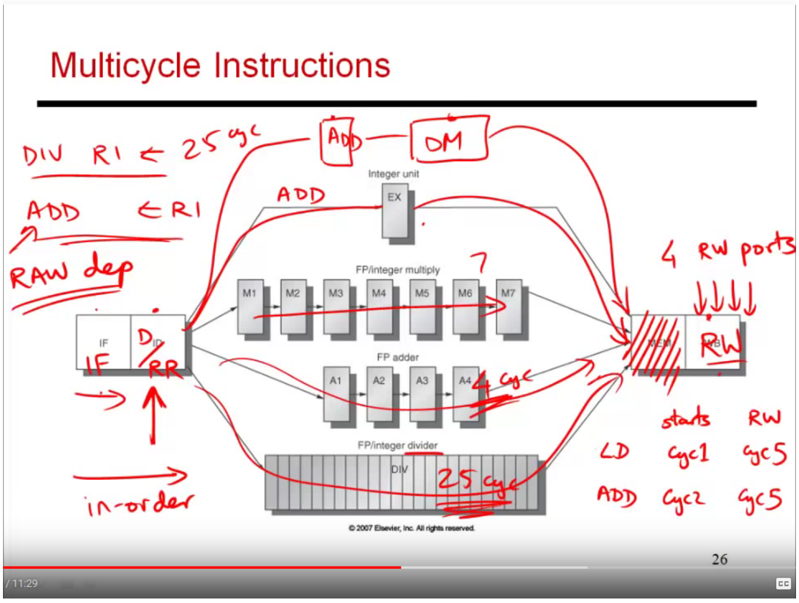
```
DADD R1, R2, R3
if R1 = 0 then
OR R7, R8, R9
DSUB R4, R5, R6
```



Multicycle Instructions

	starts	RW
LD	cyc1	cyc5
ADD	cyc2	cyc5

© 2007 Elsevier, Inc. All rights reserved.



Multicycle Instructions

DIV R1 ← 25 cyc

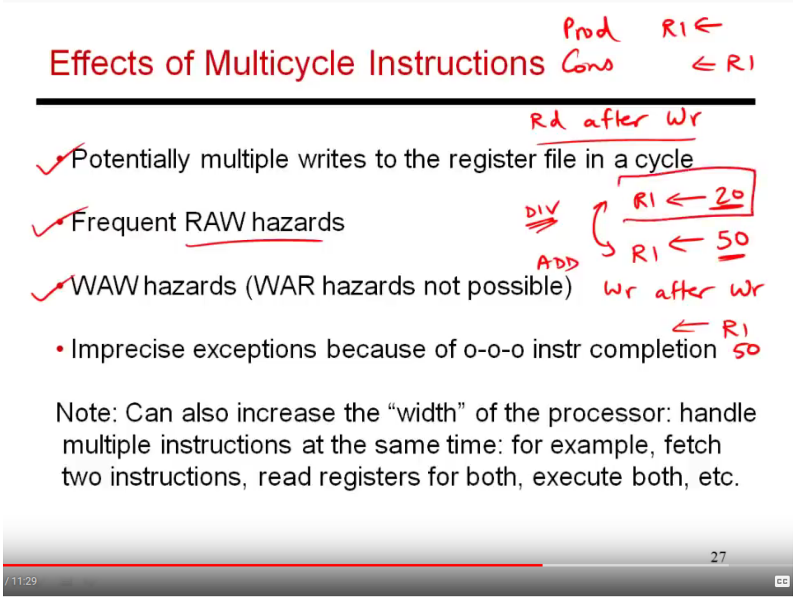
ADD ← R1

RAW dep

in-order

	starts	RW
LD	cyc1	cyc5
ADD	Cyc2	cyc5

© 2007 Elsevier, Inc. All rights reserved.



Effects of Multicycle Instructions

Prod R1 ←

Cons ← R1

Rd after Wr

- ✓ Potentially multiple writes to the register file in a cycle
- ✓ Frequent RAW hazards
- ✓ WAW hazards (WAR hazards not possible)
- Imprecise exceptions because of o-o-o instr completion

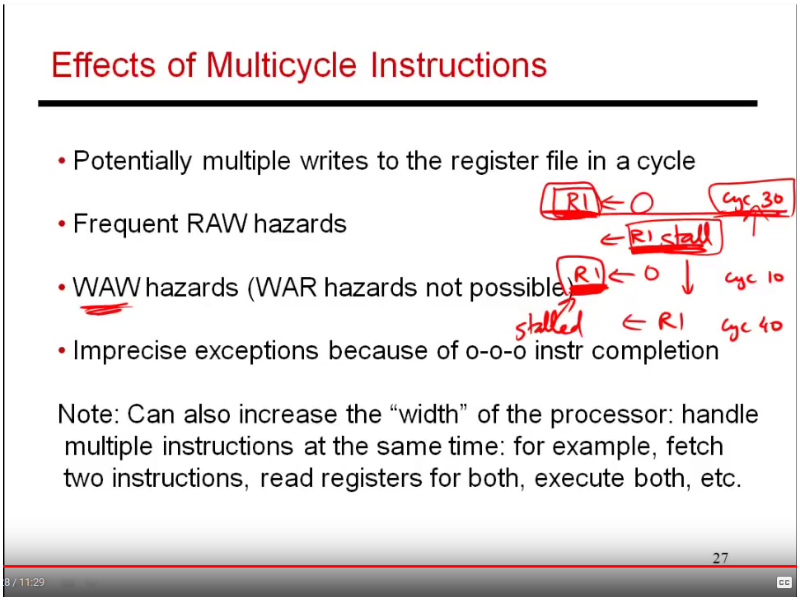
DIV R1 ← 20

ADD R1 ← 50

Wr after Wr

← R1 50

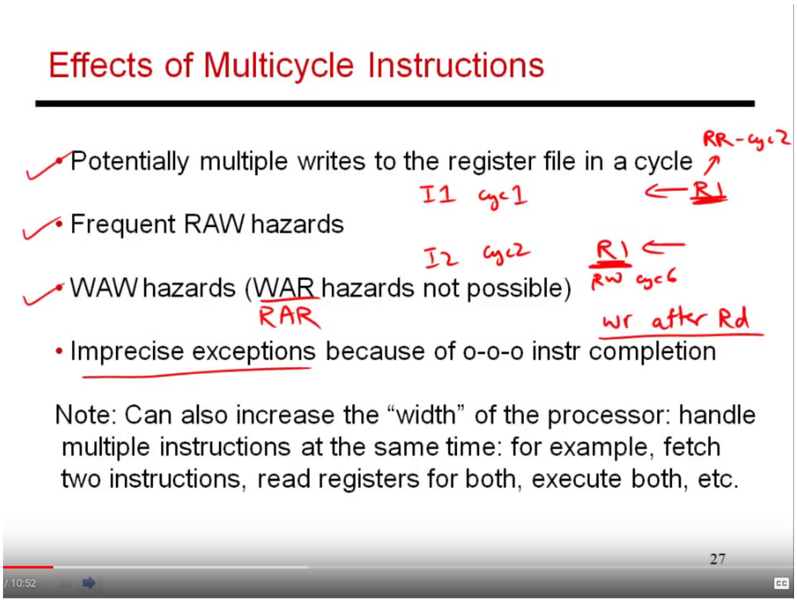
Note: Can also increase the “width” of the processor: handle multiple instructions at the same time: for example, fetch two instructions, read registers for both, execute both, etc.



Effects of Multicycle Instructions

- Potentially multiple writes to the register file in a cycle
- Frequent RAW hazards
- WAW hazards (WAR hazards not possible)
- Imprecise exceptions because of o-o-o instr completion

Note: Can also increase the "width" of the processor: handle multiple instructions at the same time: for example, fetch two instructions, read registers for both, execute both, etc.



Effects of Multicycle Instructions

- ✓ Potentially multiple writes to the register file in a cycle

RR - cyc2

I1 cyc1 ← R1

- ✓ Frequent RAW hazards

I2 cyc2 R1 ←

RW cyc6

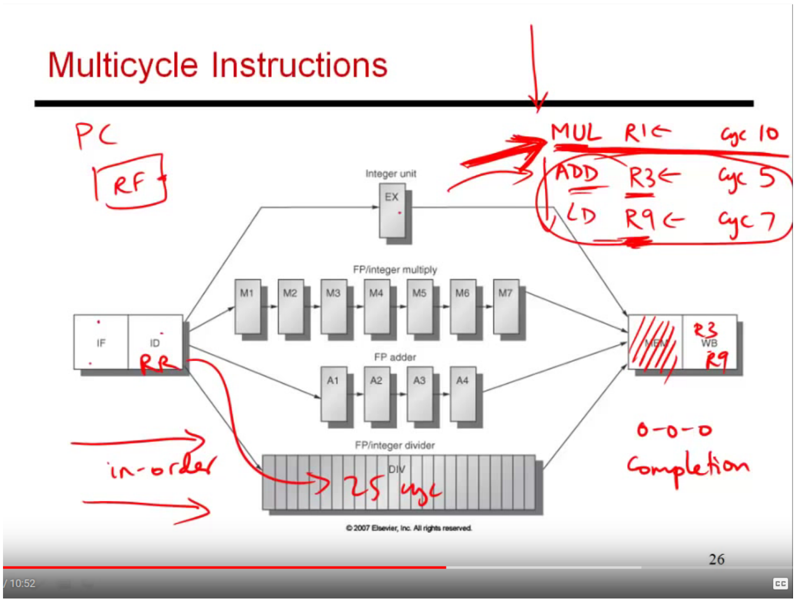
- ✓ WAW hazards (WAR hazards not possible)

RAR

wr after Rd

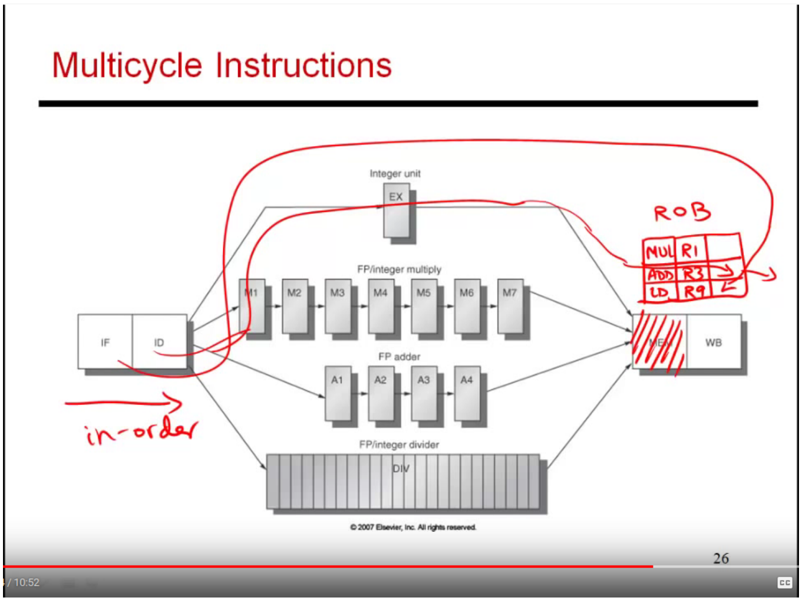
- Imprecise exceptions because of o-o-o instr completion

Note: Can also increase the “width” of the processor: handle multiple instructions at the same time: for example, fetch two instructions, read registers for both, execute both, etc.



Multicycle Instructions

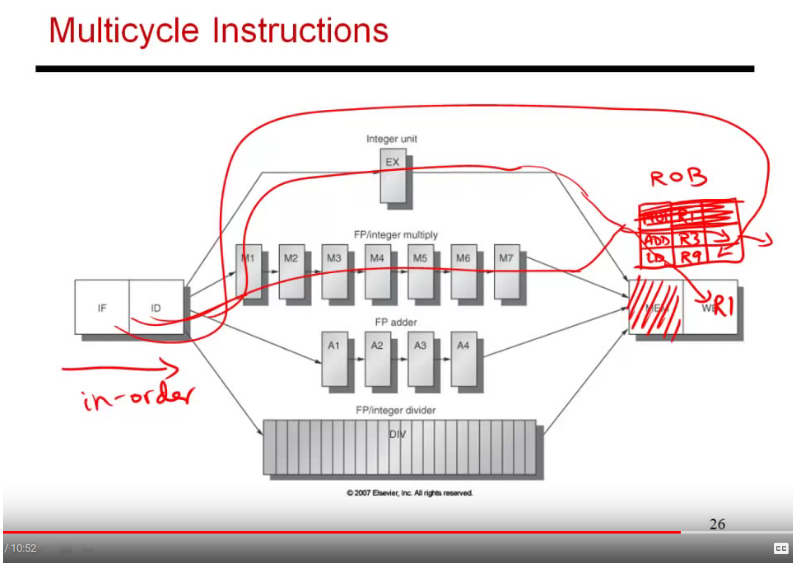
© 2007 Elsevier, Inc. All rights reserved.



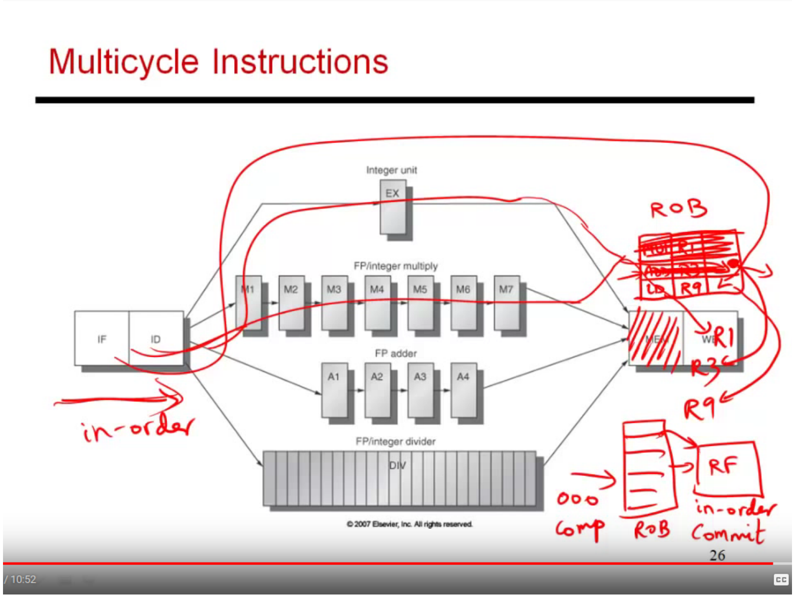
Multicycle Instructions

MUL	R1	
ADD	R3	
LD	R9	

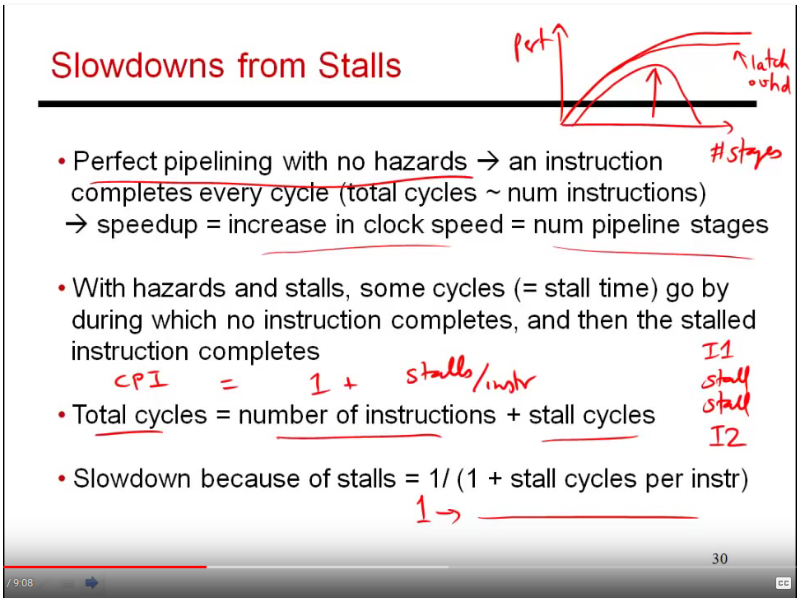
© 2007 Elsevier, Inc. All rights reserved.



Multicycle Instructions



Multicycle Instructions



Slowdowns from Stalls

- Perfect pipelining with no hazards → an instruction completes every cycle (total cycles ~ num instructions) → 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

I1
stall
stall
I2

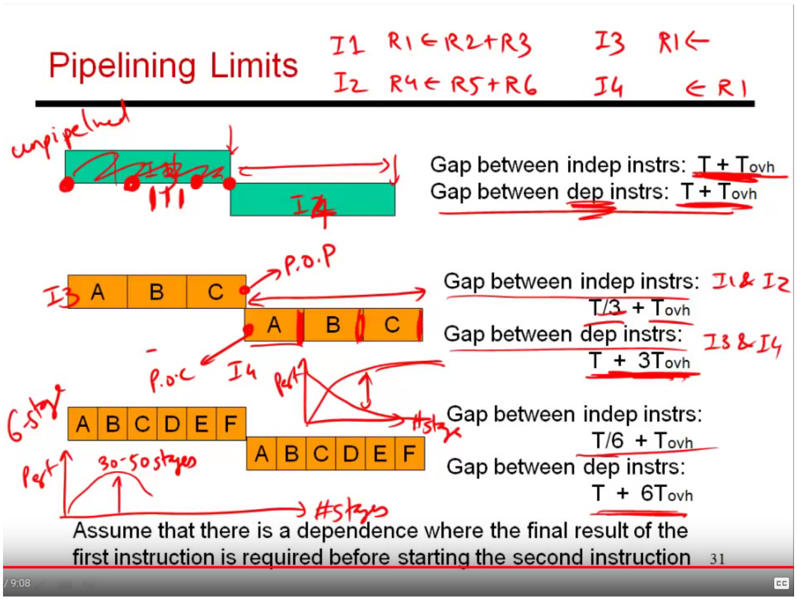
CPI = 1 + stalls/instr

- Total cycles = number of instructions + stall cycles

- Slowdown because of stalls = 1/ (1 + stall cycles per instr)

1 →

30



Pipelining Limits

I1 R1 ← R2 + R3 I3 R1 ←

I2 R4 ← R5 + R6 I4 ← R1

Gap between indep instrs: T + Tovh

Gap between dep instrs: T + Tovh

Gap between indep instrs: I1 & I2

T/3 + Tovh

Gap between dep instrs: I3 & I4

T + 3Tovh

Gap between indep instrs:

T/6 + Tovh

Gap between dep instrs:

T + 6Tovh

Assume that there is a dependence where the final result of the first instruction is required before starting the second instruction