Imperial College London

Lecture 8

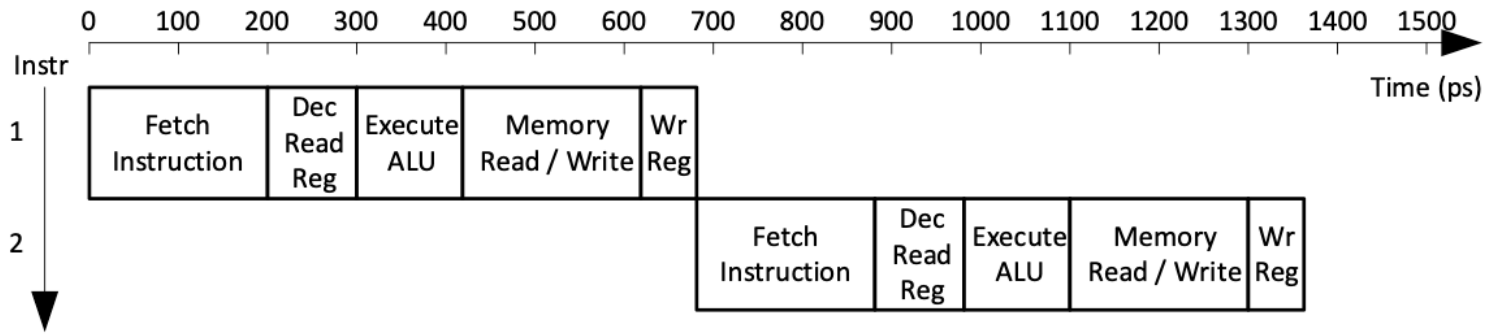
Pipelined Processor

Peter Cheung
Imperial College London

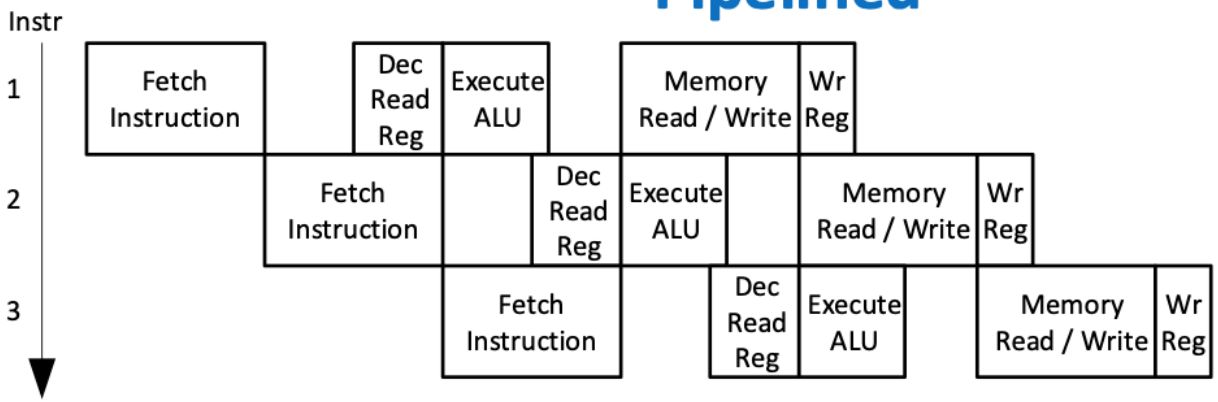
URL: www.ee.imperial.ac.uk/pcheung/teaching/EE2_CAS/
E-mail: p.cheung@imperial.ac.uk

Single-cyle vs Pipelined Processor

Single-Cycle

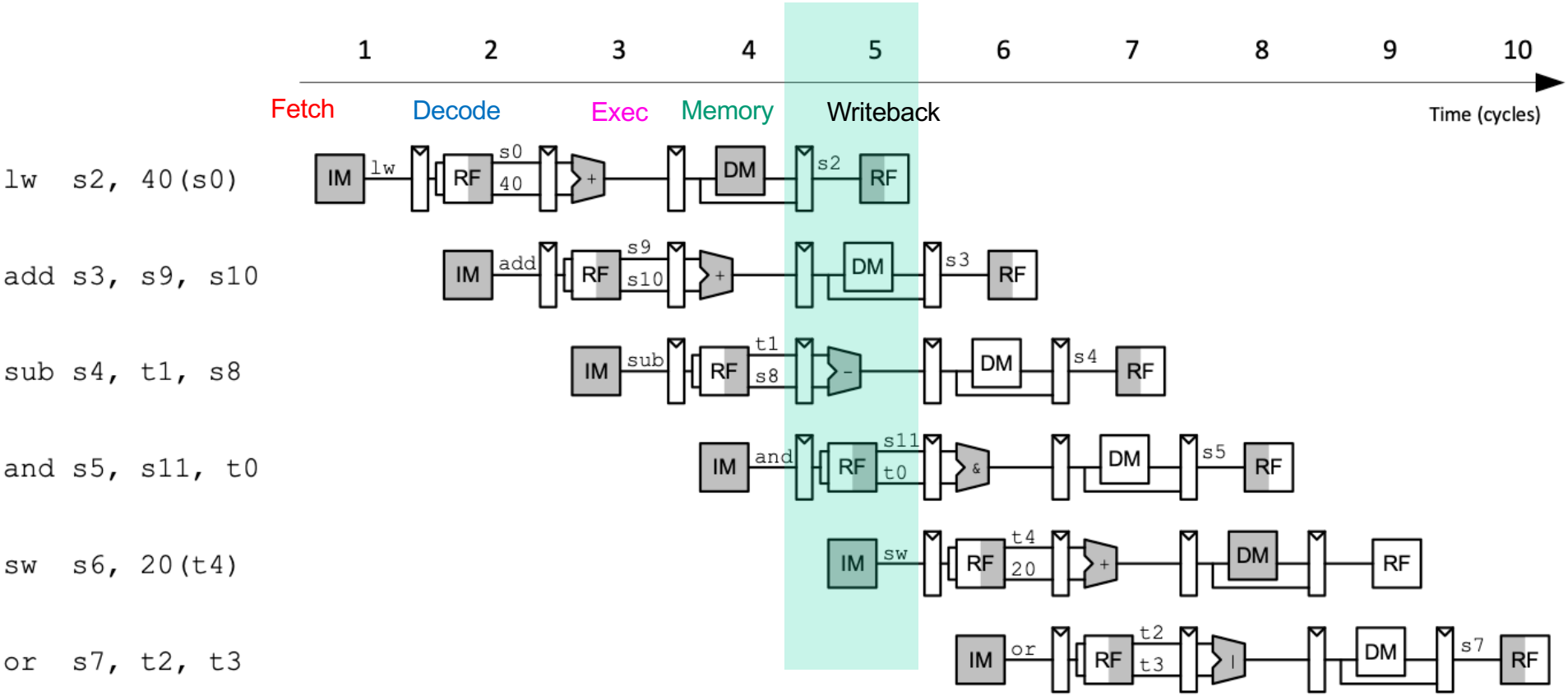


Pipelined



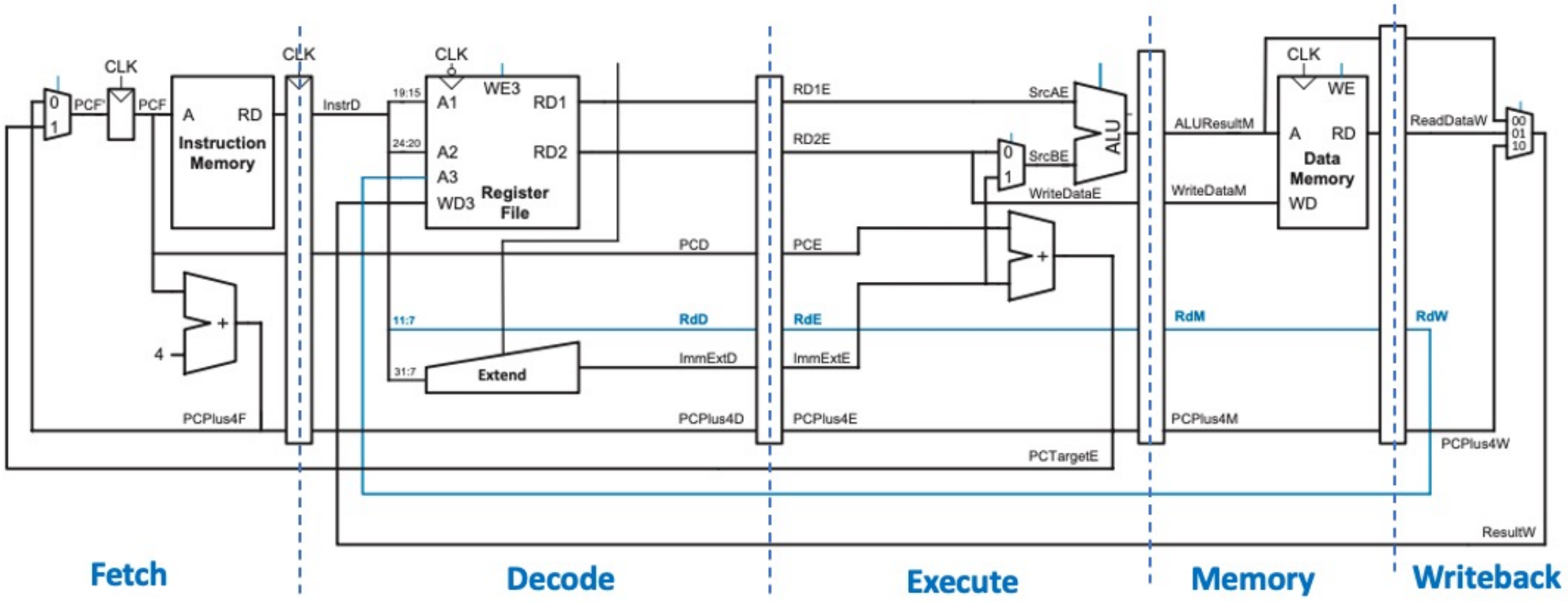
Based on: "*Digital Design and Computer Architecture (RISC-V Edition)*"
by Sarah Harris and David Harris (H&H),

Pipelined Processor Abstraction



Based on: "*Digital Design and Computer Architecture (RISC-V Edition)*" by Sarah Harris and David Harris (H&H),

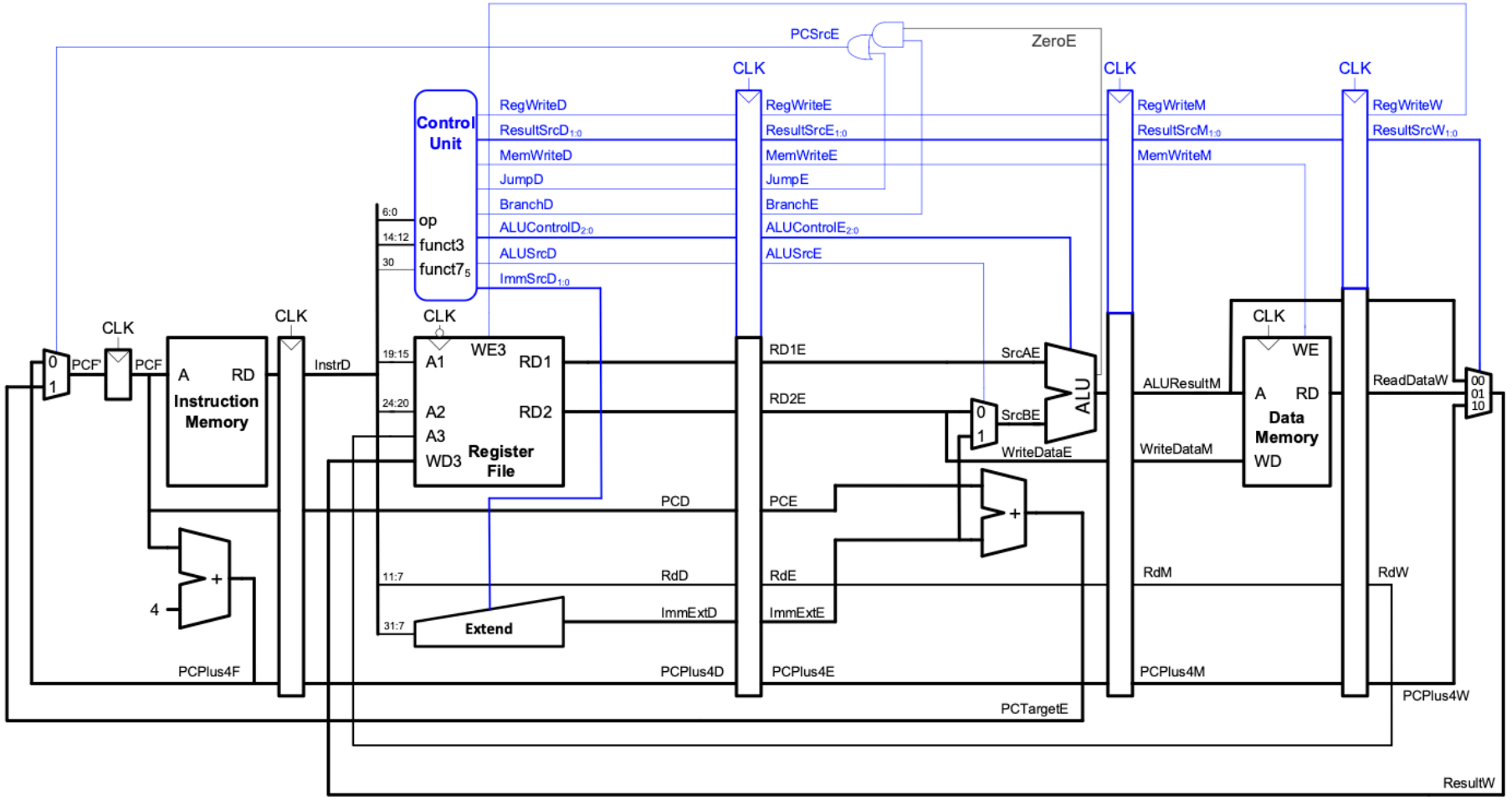
Adding Pipeline to Single-Cycle Processor



- Signals in Pipelined Processor are appended with first letter of stage (i.e., PC**F**, PC**D**, PC**E**).
- Register file written on **falling edge** of *CLK*

Based on: "*Digital Design and Computer Architecture (RISC-V Edition)*" by Sarah Harris and David Harris (H&H).

Adding Pipeline to Single-Cycle Processor



- **Same control unit** as single-cycle processor
- **Control signals travel with** the instruction (drop off when used)

Based on: "*Digital Design and Computer Architecture (RISC-V Edition)*" by Sarah Harris and David Harris (H&H).

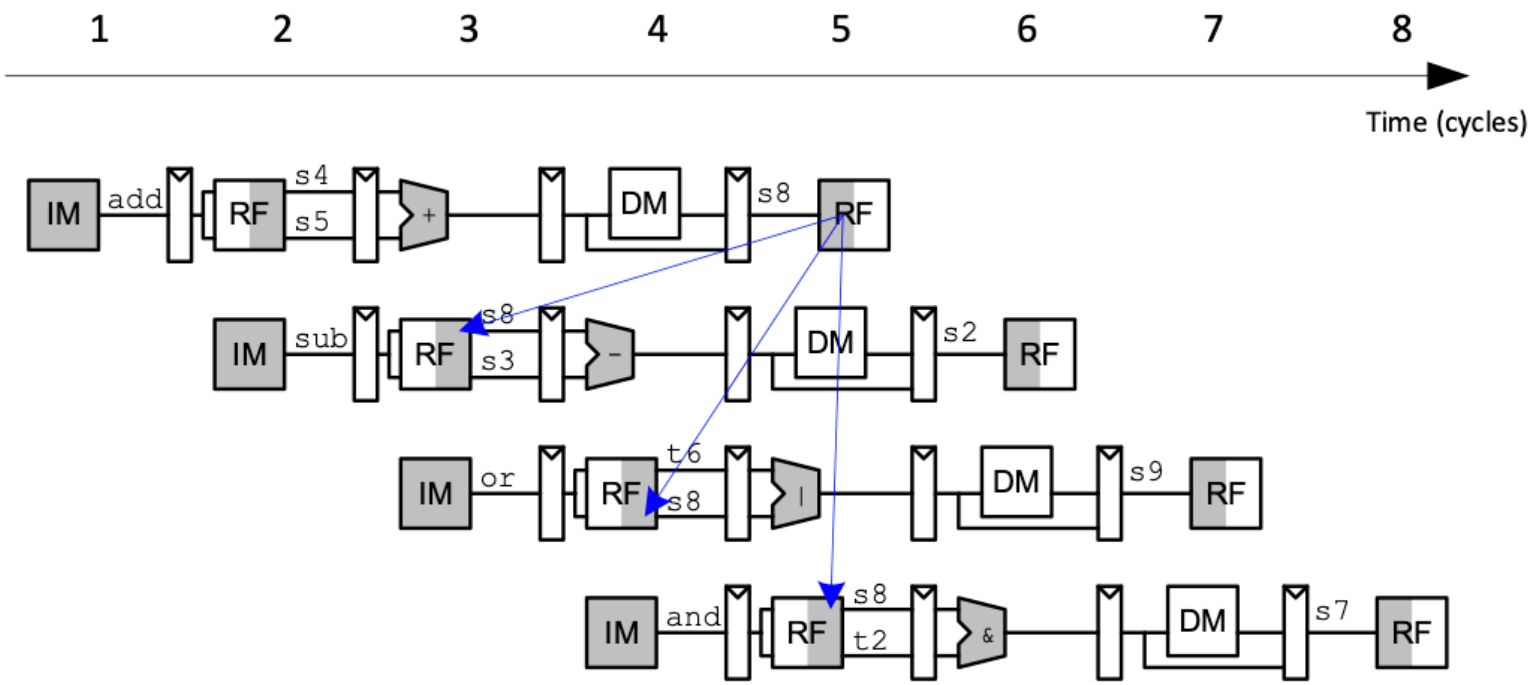
Pipeline Hazards

- When an instruction depends on result from instruction that hasn't completed
- Two types of hazards:
  - **Data hazard:** register value not yet written back to register file
  - **Control hazard:** next instruction not decided yet (caused by branch)

Based on: "*Digital Design and Computer Architecture (RISC-V Edition)*" by Sarah Harris and David Harris (H&H).

Data Hazard

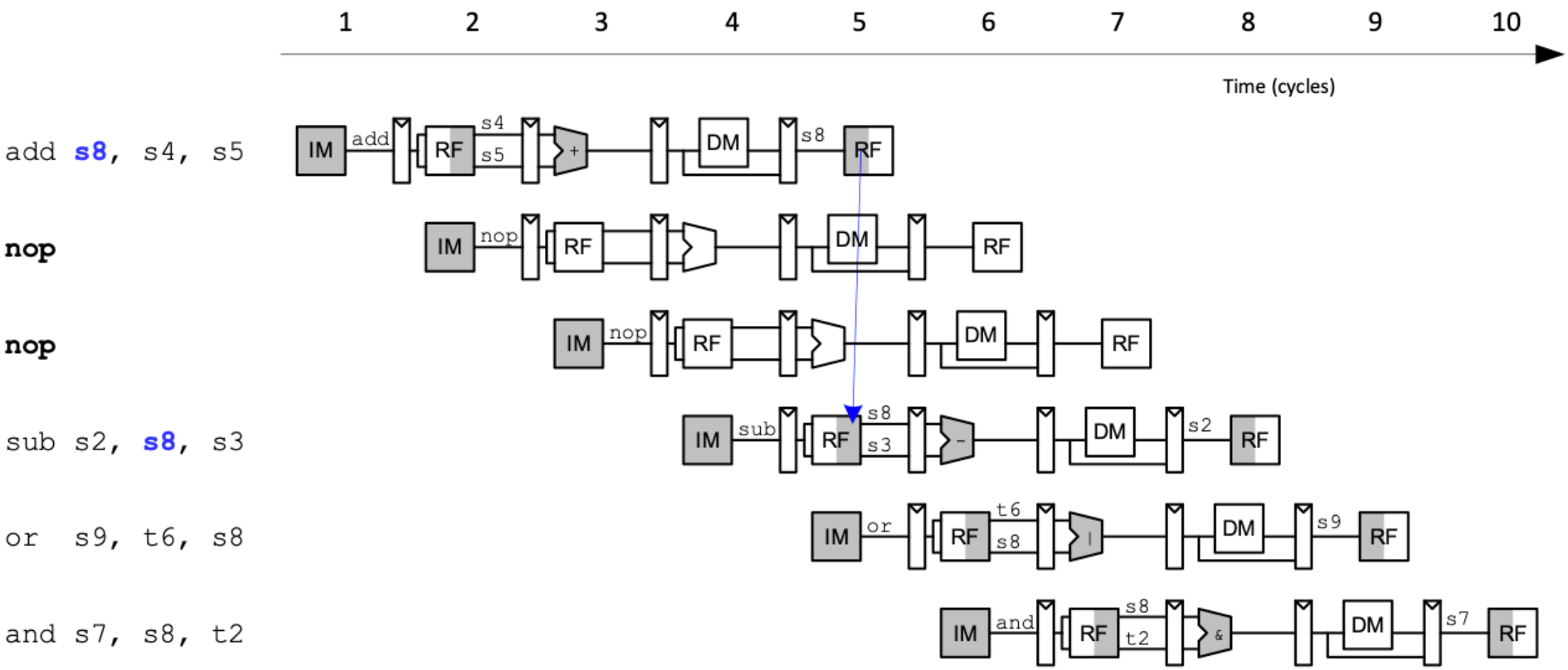
```
add s8, s4, s5
sub s2, s8, s3
or  s9, t6, s8
and s7, s8, t2
```



Based on: "*Digital Design and Computer Architecture (RISC-V Edition)*" by Sarah Harris and David Harris (H&H).

Avoid Data Hazard in code

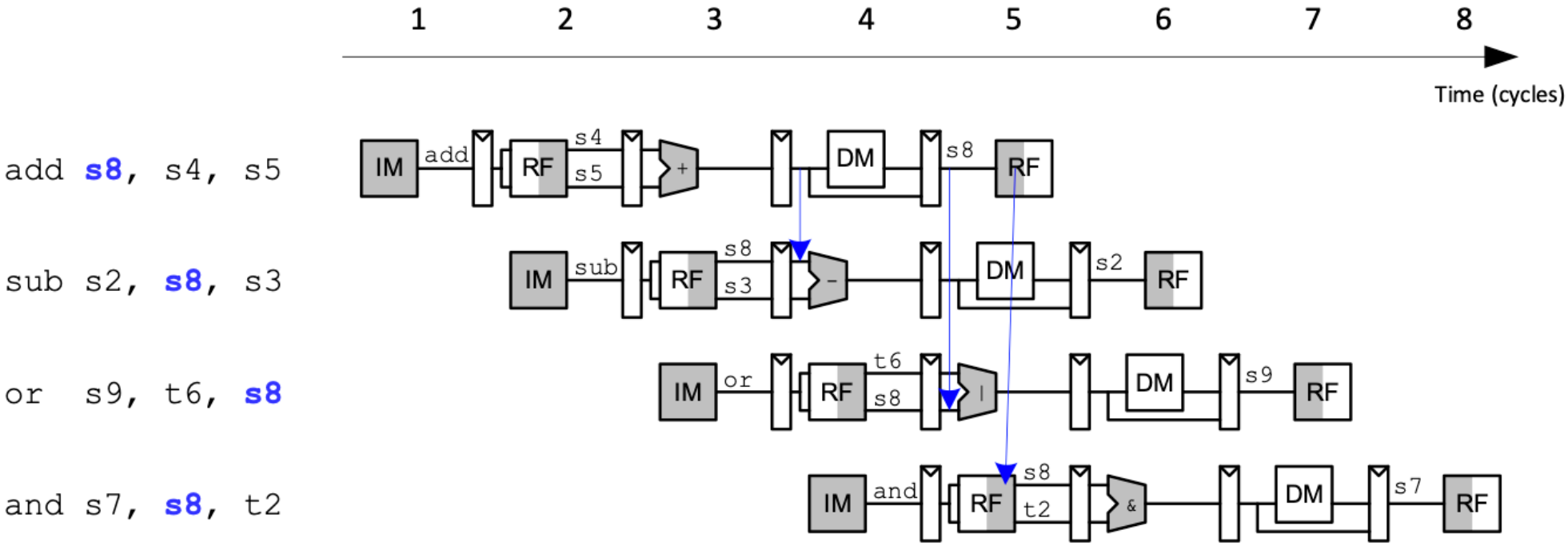
- **Insert** enough **nops** for result to be ready
- Or move independent useful instructions forward



Based on: "*Digital Design and Computer Architecture (RISC-V Edition)*" by Sarah Harris and David Harris (H&H).

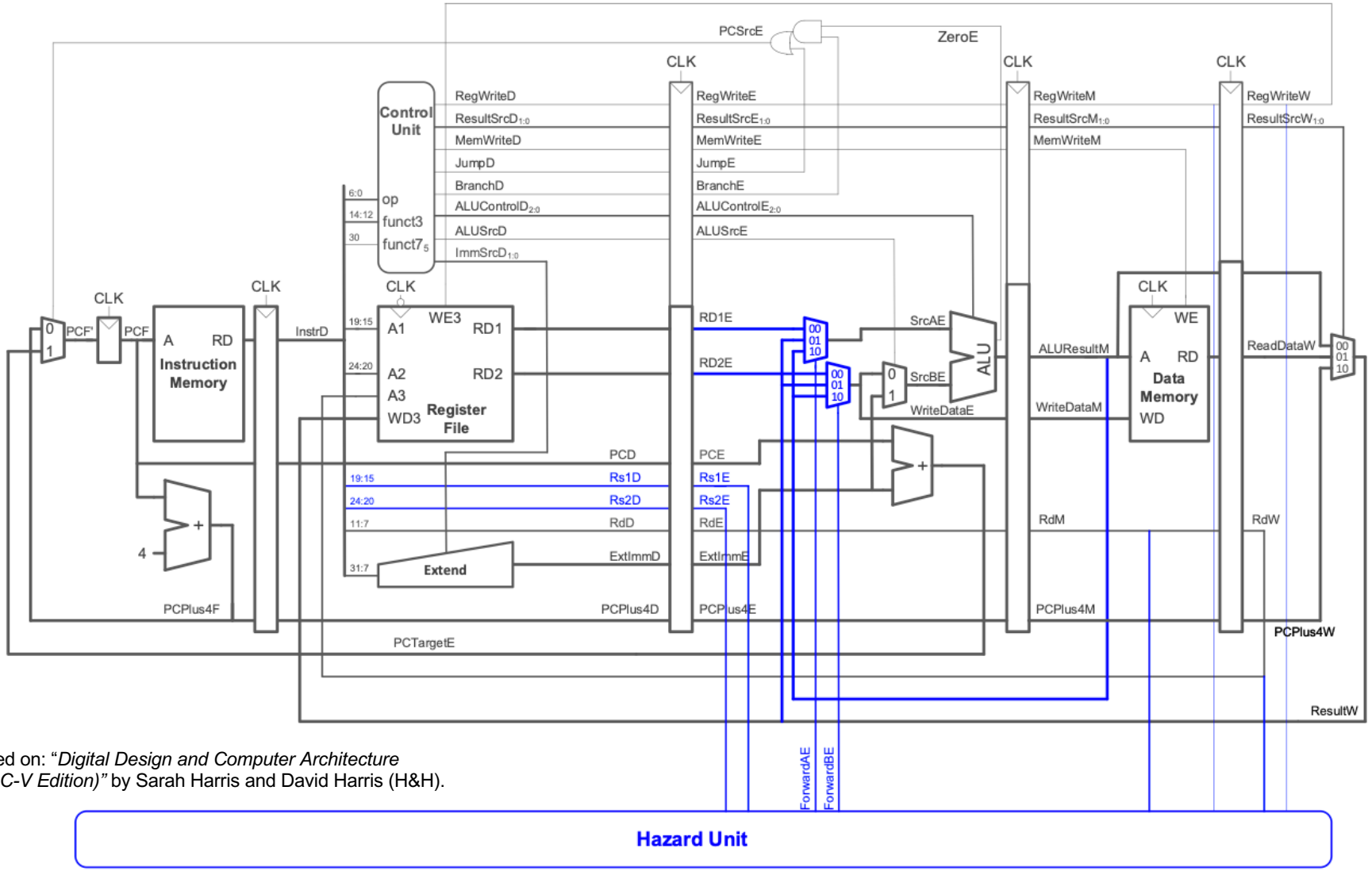
Handle Data Hazard by Forwarding

- Data is **available on internal busses** before it is written back to the register file (RF).
- **Forward data** from internal busses **to Execute stage**.
- Check if source register **in Execute stage matches** destination register of instruction **in Memory or Writeback stage**.
- If so, forward result.



Based on: "*Digital Design and Computer Architecture (RISC-V Edition)*" by Sarah Harris and David Harris (H&H).

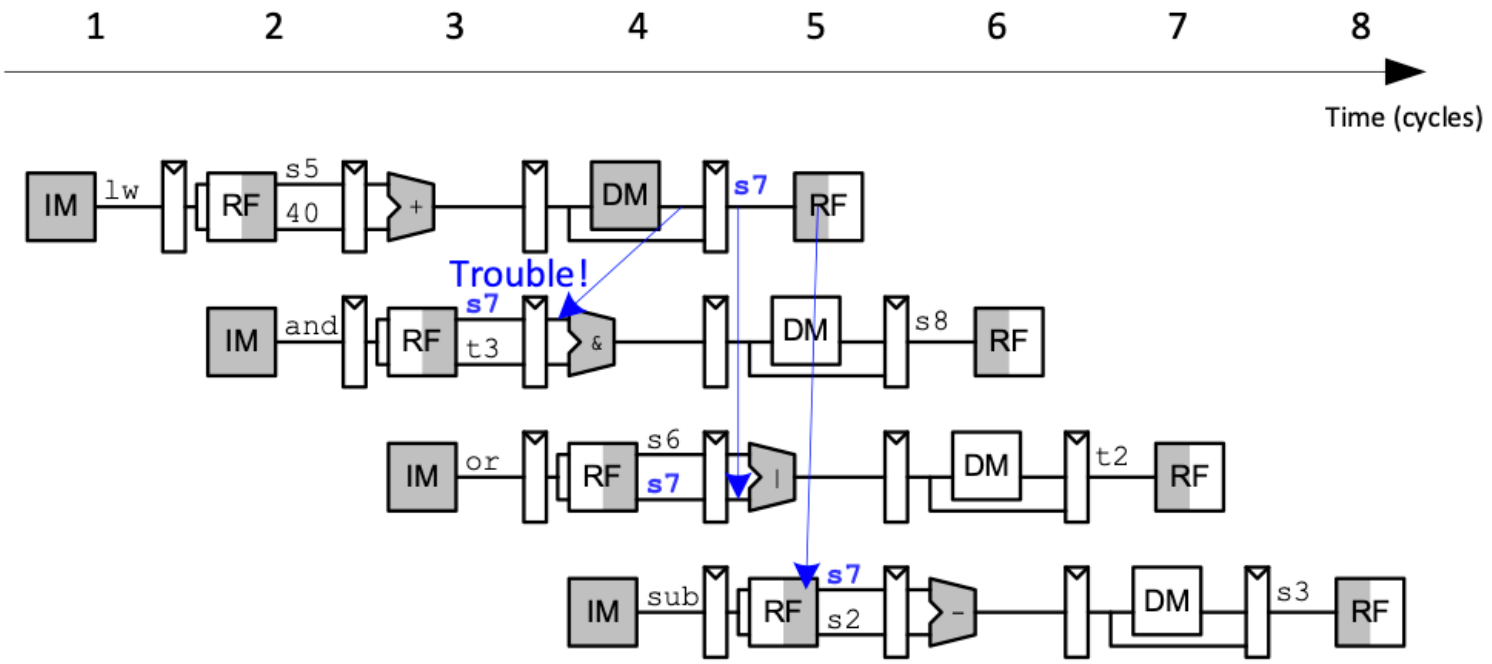
Adding Hazard Unit



Based on: "*Digital Design and Computer Architecture (RISC-V Edition)*" by Sarah Harris and David Harris (H&H).

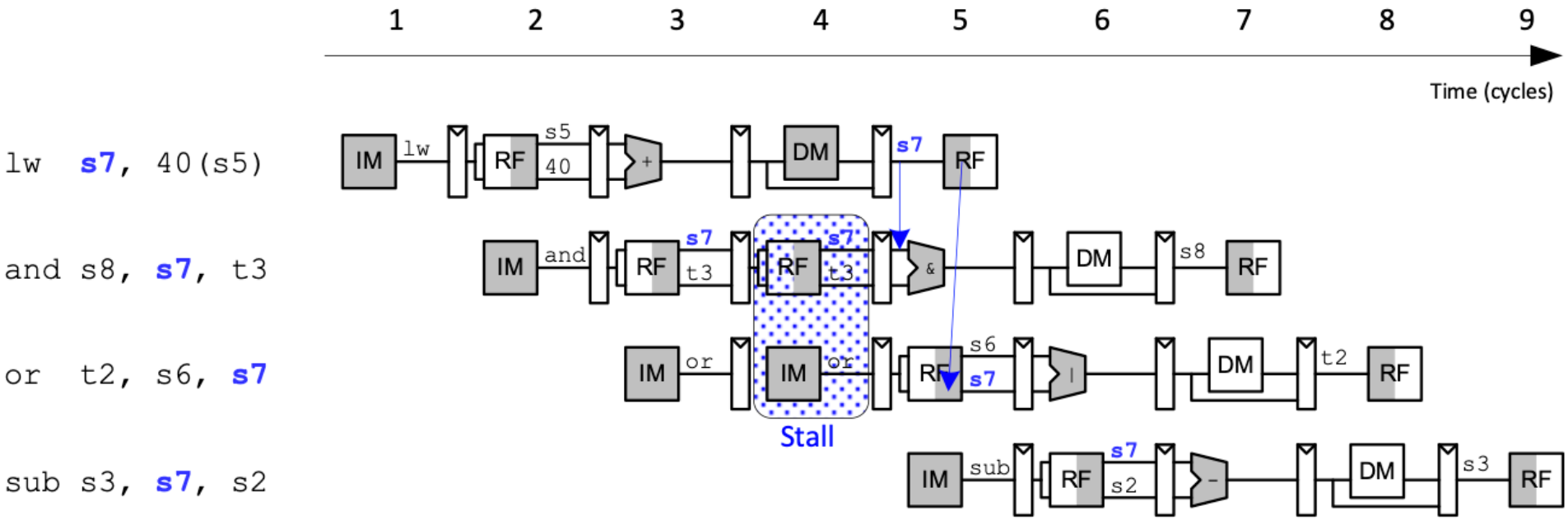
Data Hazard due to lw Data Dependency

```
lw   s7, 40(s5)
and s8, s7, t3
or   t2, s6, s7
sub s3, s7, s2
```



Based on: "*Digital Design and Computer Architecture (RISC-V Edition)*" by Sarah Harris and David Harris (H&H).

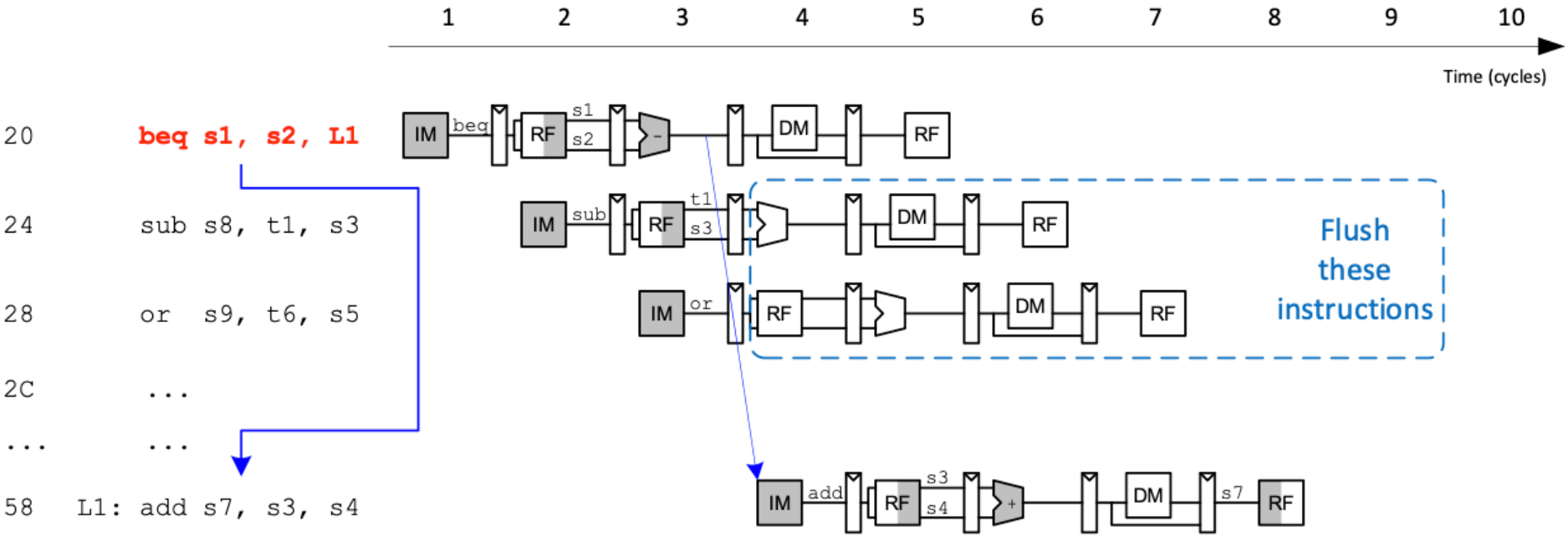
Stalling to solve lw Data Dependency



Based on: "*Digital Design and Computer Architecture (RISC-V Edition)*" by Sarah Harris and David Harris (H&H).

Control Hazard

- **beq, bne:**
  - Branch **not determined until the Execute stage** of pipeline
  - **Instructions** after branch **fetched** before branch occurs
  - These **2 instructions must be flushed** if branch happens



Based on: "*Digital Design and Computer Architecture (RISC-V Edition)*" by Sarah Harris and David Harris (H&H).

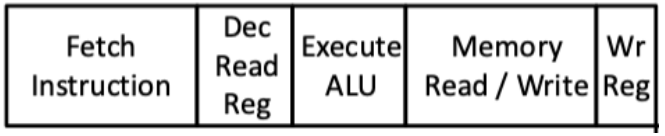
Single-Cycle Processor Performance

Program Execution Time

= (#instructions)(cycles/instruction)(seconds/cycle)

= # instructions x CPI x T_C

Element	Parameter	Delay (ps)
Register clk-to-Q	t_{pcq}	40
Register setup	t_{setup}	50
Multiplexer	t_{mux}	30
AND-OR gate	$t_{AND\text{-}OR}$	20
ALU	t_{ALU}	120
Decoder (control unit)	t_{dec}	25
Extend unit	t_{ext}	35
Memory read	t_{mem}	200
Register file read	t_{RFread}	100
Register file setup	$t_{RFsetup}$	60



Program with 100 billion instructions:

Exec Time = # instructions x CPI x T_C

= $(100 \times 10^9)(1)(750 \times 10^{-12}\ s)$

= **75 seconds**

750ps

$$T_{c\_single} = t_{pcq\_PC} + 2t_{\text{mem}} + t_{RF\text{read}} + t_{\text{ALU}} + t_{\text{mux}} + t_{RF\text{setup}}$$

Based on: "*Digital Design and Computer Architecture (RISC-V Edition)*" by Sarah Harris and David Harris (H&H).

Pipelined Processor Performance

Pipelined processor critical path:

$T_{c\_pipelined}$ = max of

$t_{pcq} + t_{mem} + t_{setup}$	Fetch
$2(t_{RFread} + t_{setup})$	Decode
$t_{pcq} + 4t_{mux} + t_{ALU} + t_{AND\text{-}OR} + t_{setup}$	Execute
$t_{pcq} + t_{mem} + t_{setup}$	Memory
$2(t_{pcq} + t_{mux} + t_{RFwrite})$	Writeback

350ps

Program with 100 billion instructions

Execution Time = (# instructions) × CPI × T_c

= $(100 \times 10^9)(1.23)(350 \times 10^{-12})$

= **43 seconds**

Based on: "*Digital Design and Computer Architecture (RISC-V Edition)*" by Sarah Harris and David Harris (H&H).