

Chapter 7: Microarchitecture

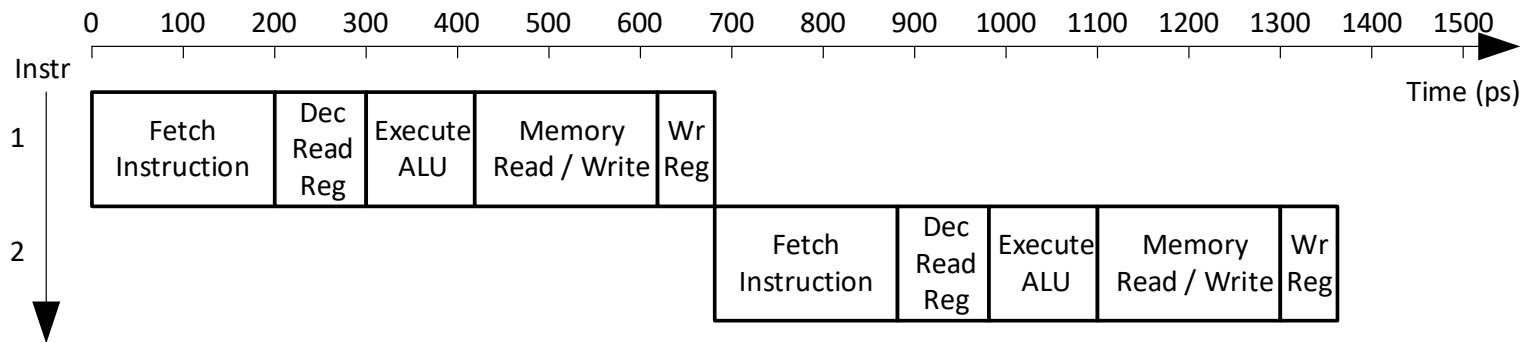
Pipelined RISC-V Processor

Pipelined RISC-V Processor

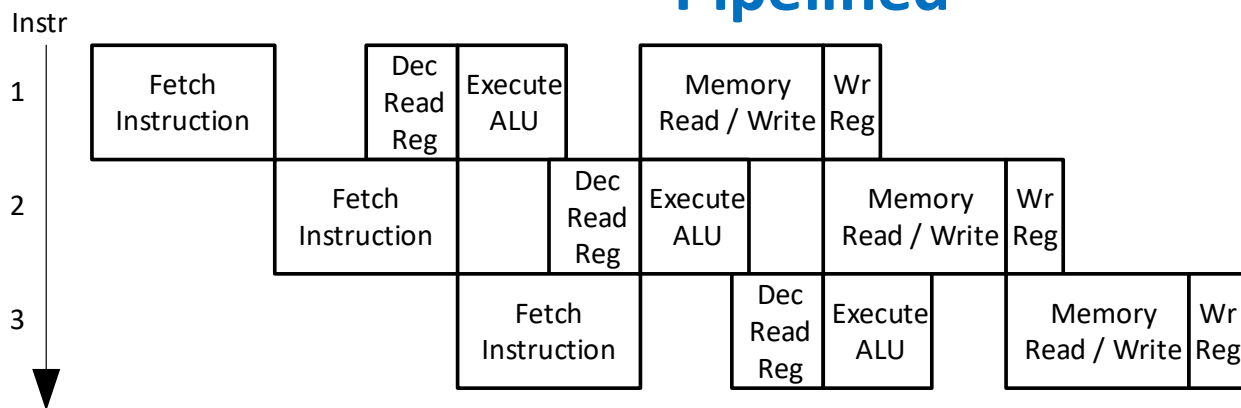
- **Temporal parallelism**
- Divide single-cycle processor into **5 stages**:
 - Fetch
 - Decode
 - Execute
 - Memory
 - Writeback
- Add **pipeline registers** between stages

Single-Cycle vs. Pipelined Processor

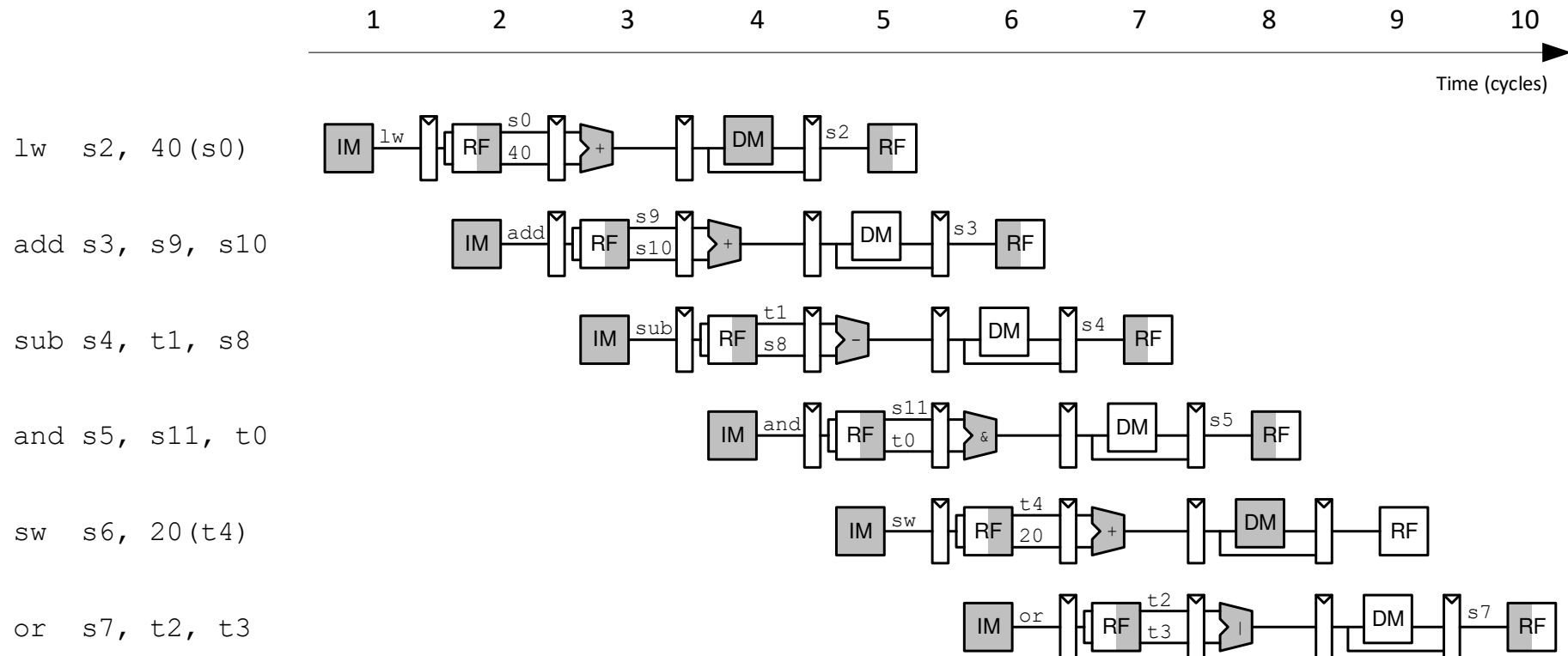
Single-Cycle



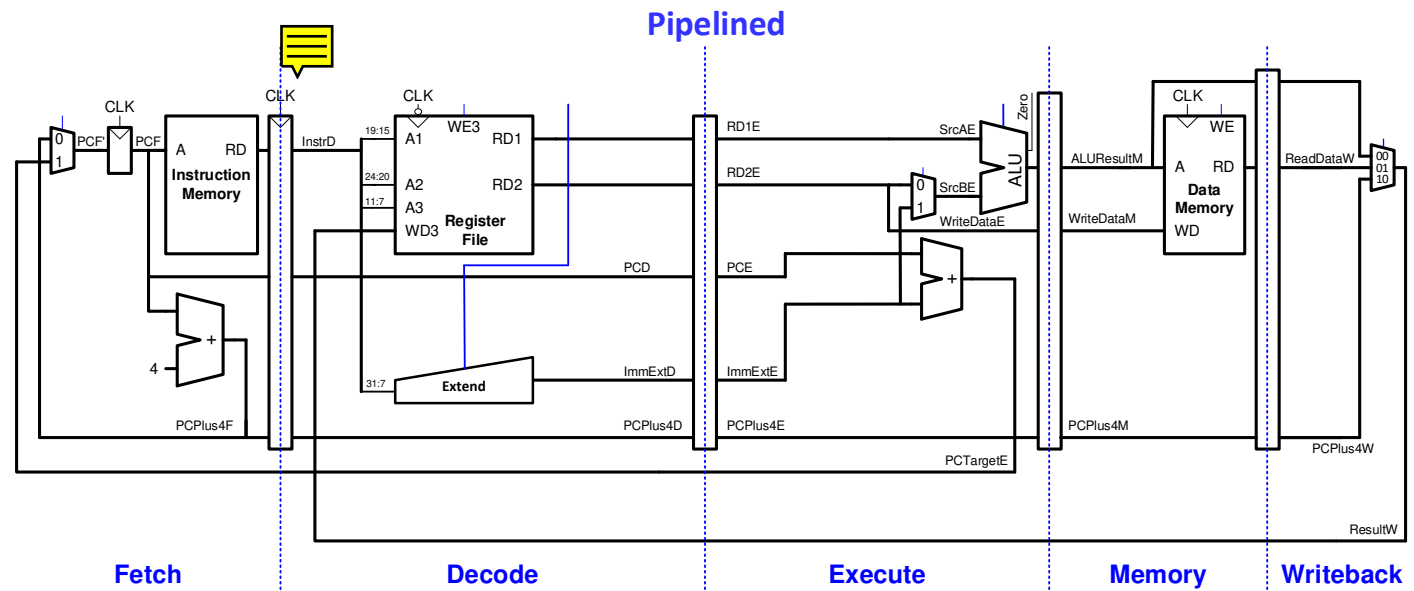
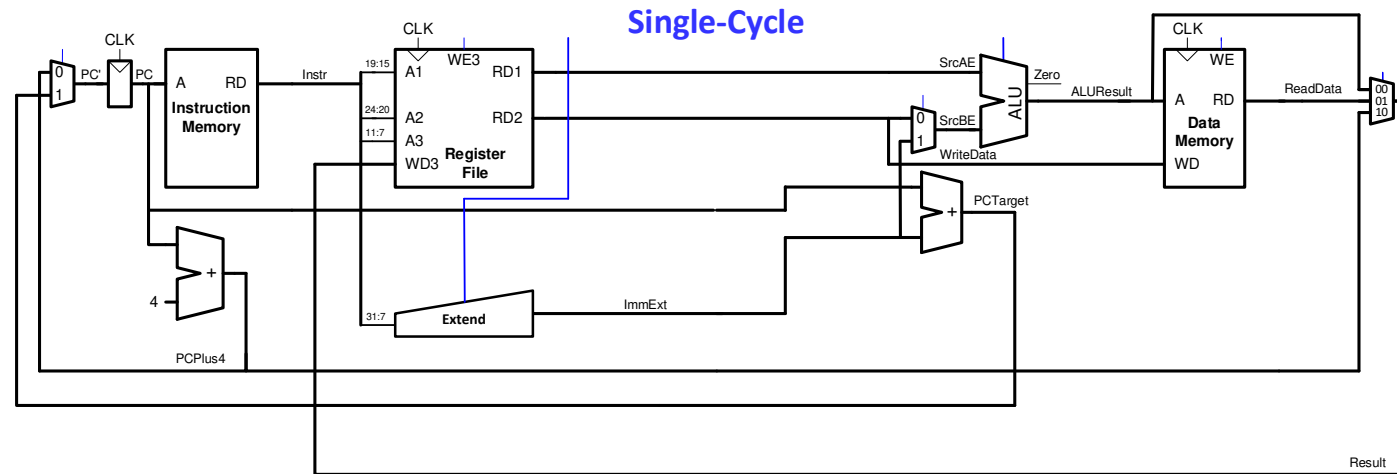
Pipelined



Pipelined Processor Abstraction

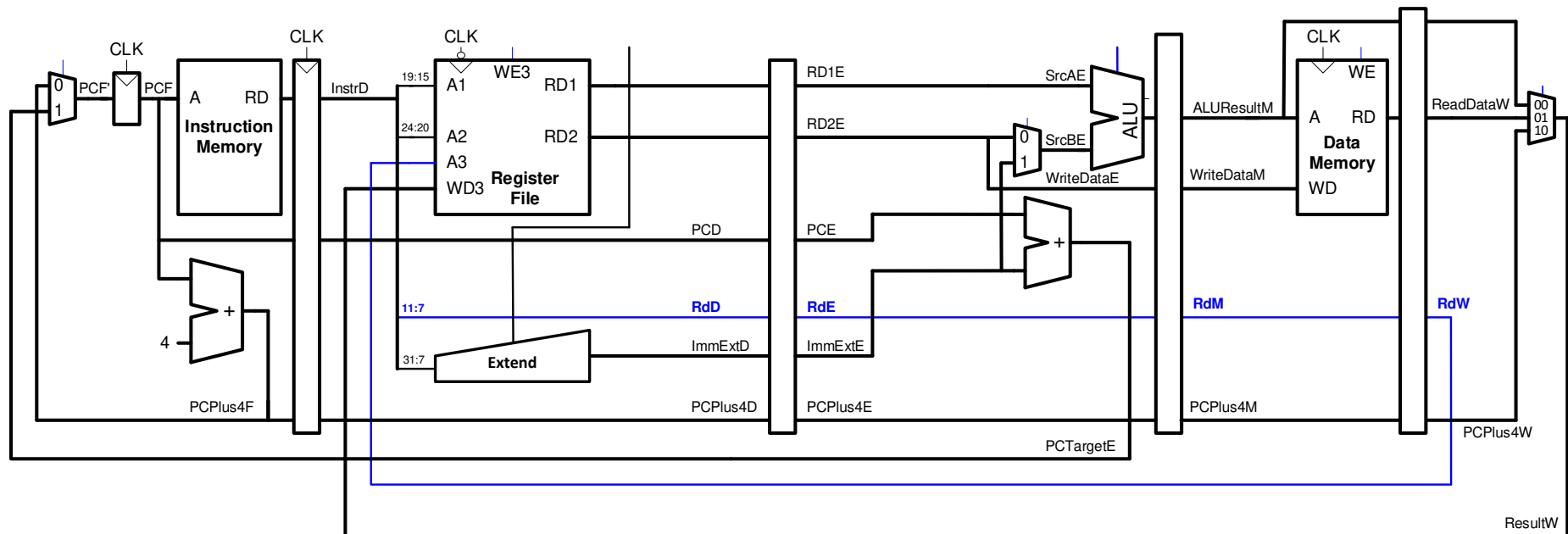


Single-Cycle & Pipelined Datapaths



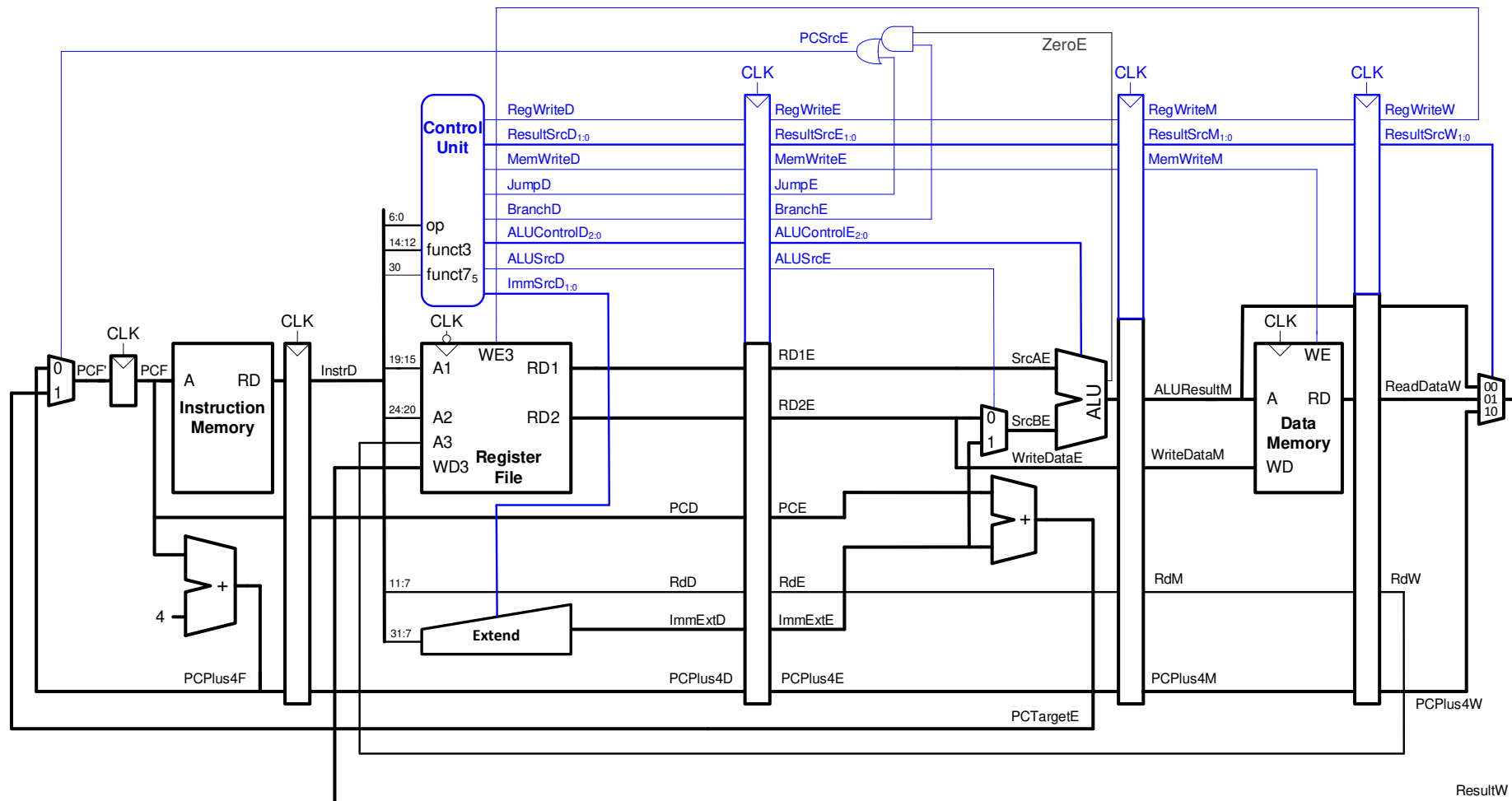
Signals in Pipelined Processor are appended with first letter of stage (i.e., PC**F**, PC**D**, PC**E**).

Corrected Pipelined Datapath



- **Rd** must arrive at same time as **Result**
- Register file written on **falling edge** of **CLK**

Pipelined Processor with Control



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

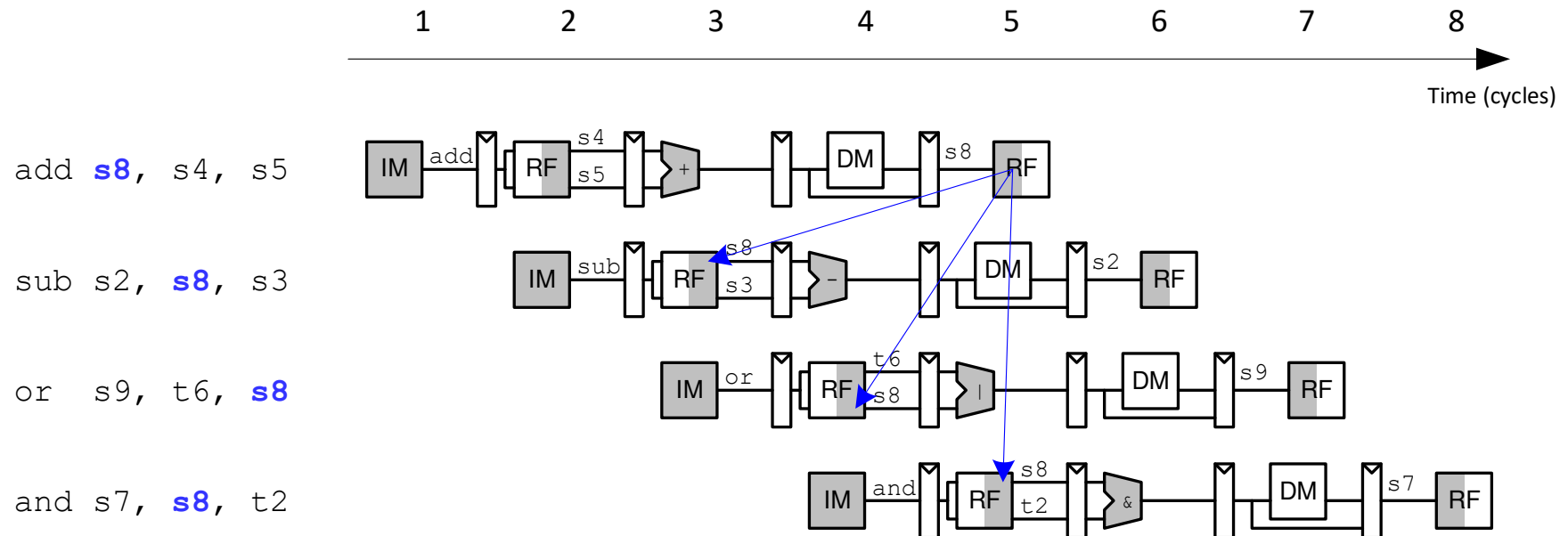
Chapter 7: Microarchitecture

Pipelined Processor Hazards

Pipelined Hazards

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

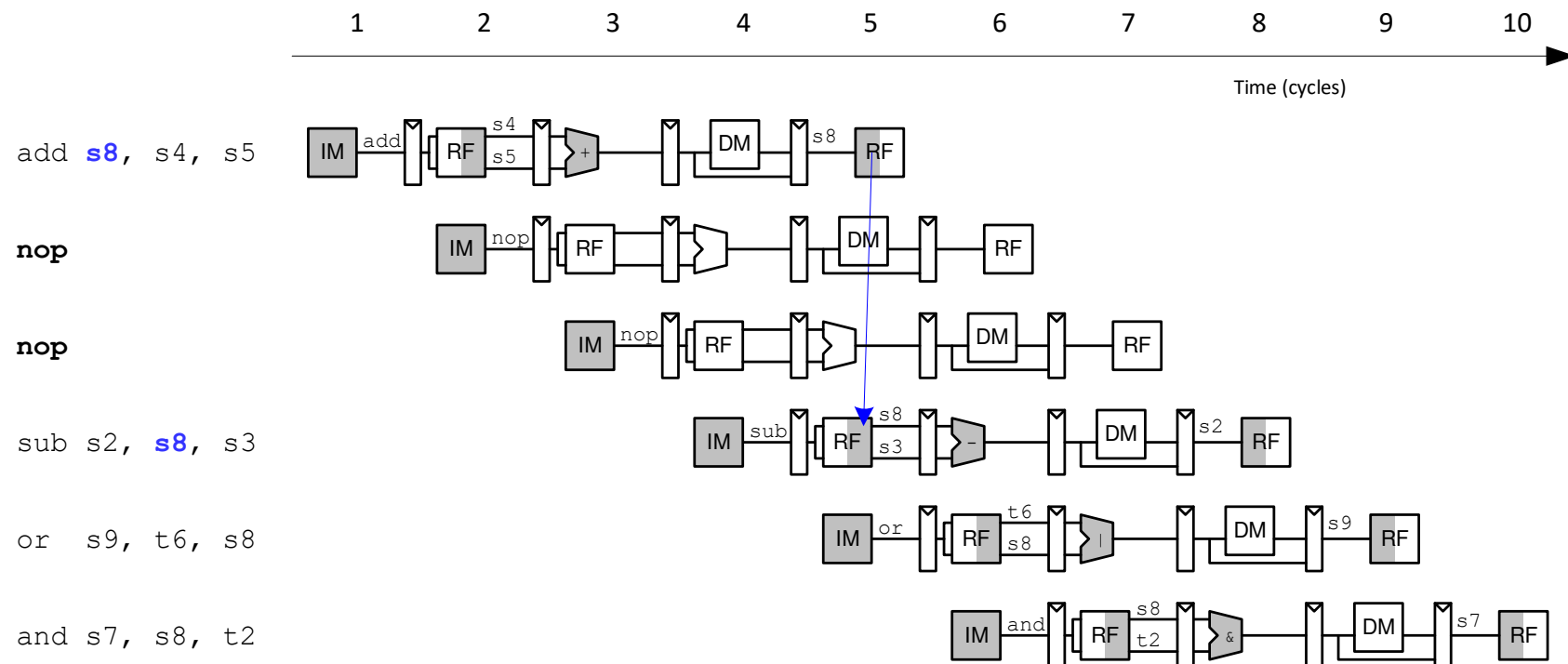
Data Hazard



Handling Data Hazards

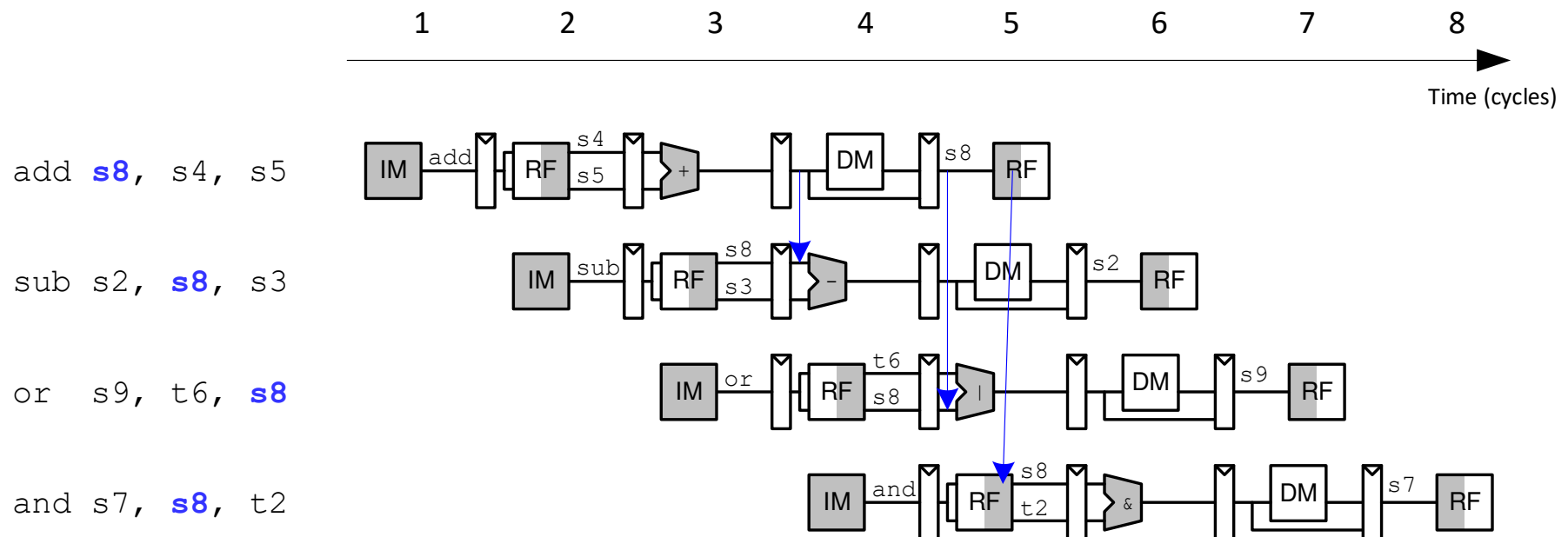
Handling Data Hazards

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



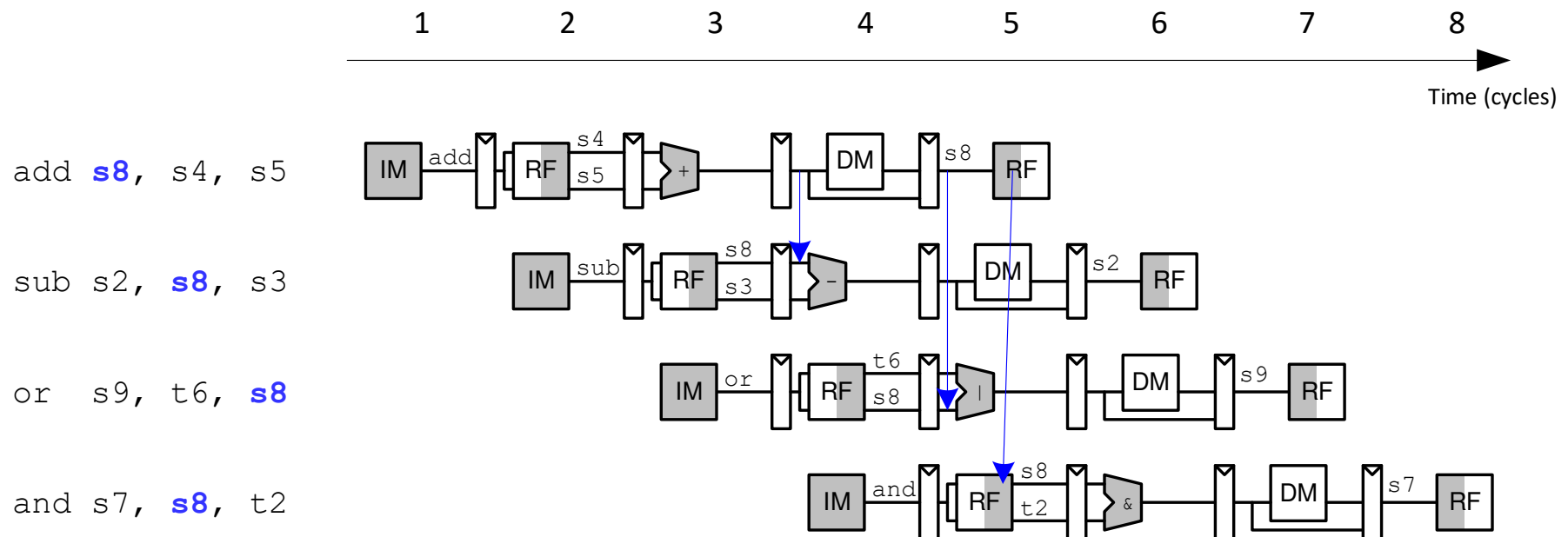
Data Forwarding

- Data is **available on internal busses** before it is written back to the register file (RF).
- **Forward data** from internal busses **to Execute stage**.

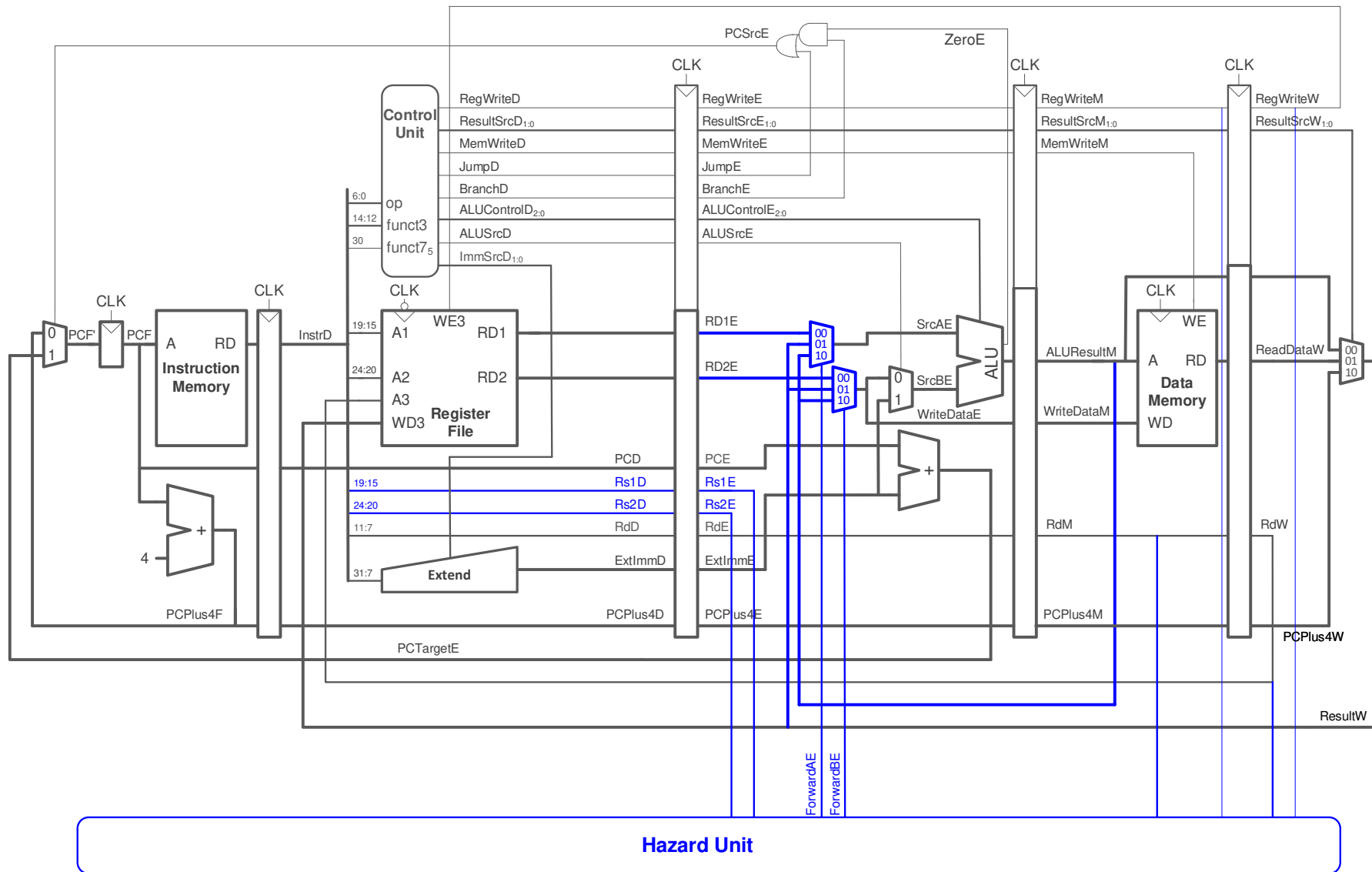


Data Forwarding

- Check if source register **in Execute stage matches** destination register of instruction **in Memory or Writeback stage**.
- If so, forward result.



Data Forwarding: Hazard Unit



Data Forwarding

- **Case 1: Execute** stage $Rs1$ or $Rs2$ matches **Memory** stage Rd ?
Forward from Memory stage
- **Case 2: Execute** stage $Rs1$ or $Rs2$ matches **Writeback** stage Rd ?
Forward from Writeback stage
- **Case 3:** Otherwise use value read from register file (as usual)

Equations for $Rs1$:

```
if      (( $Rs1E == RdM$ ) AND  $RegWriteM$ )           // Case 1
         $ForwardAE = 10$ 
else if (( $Rs1E == RdW$ ) AND  $RegWriteW$ )           // Case 2
         $ForwardAE = 01$ 
else     $ForwardAE = 00$                            // Case 3
```

***ForwardBE** equations are similar (replace $Rs1E$ with $Rs2E$)*

Data Forwarding

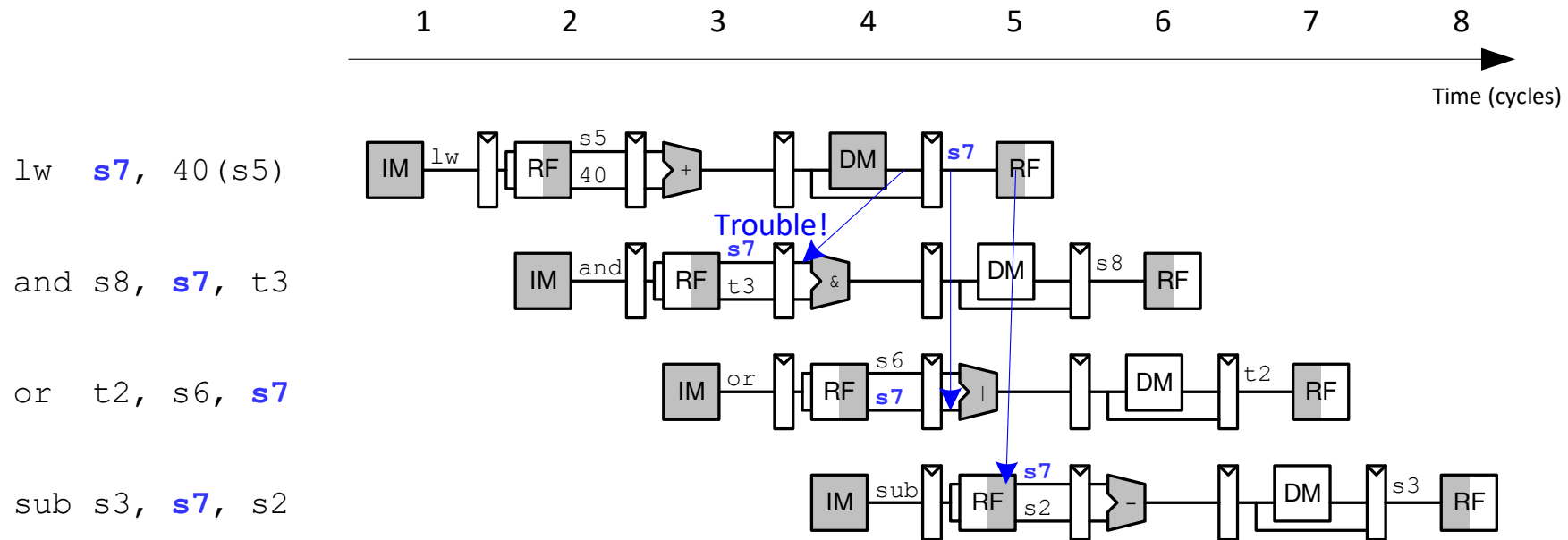
- **Case 1: Execute** stage $Rs1$ or $Rs2$ matches **Memory** stage Rd ?
Forward from Memory stage
- **Case 2: Execute** stage $Rs1$ or $Rs2$ matches **Writeback** stage Rd ?
Forward from Writeback stage
- **Case 3:** Otherwise use value read from register file (as usual)

Equations for $Rs1$:

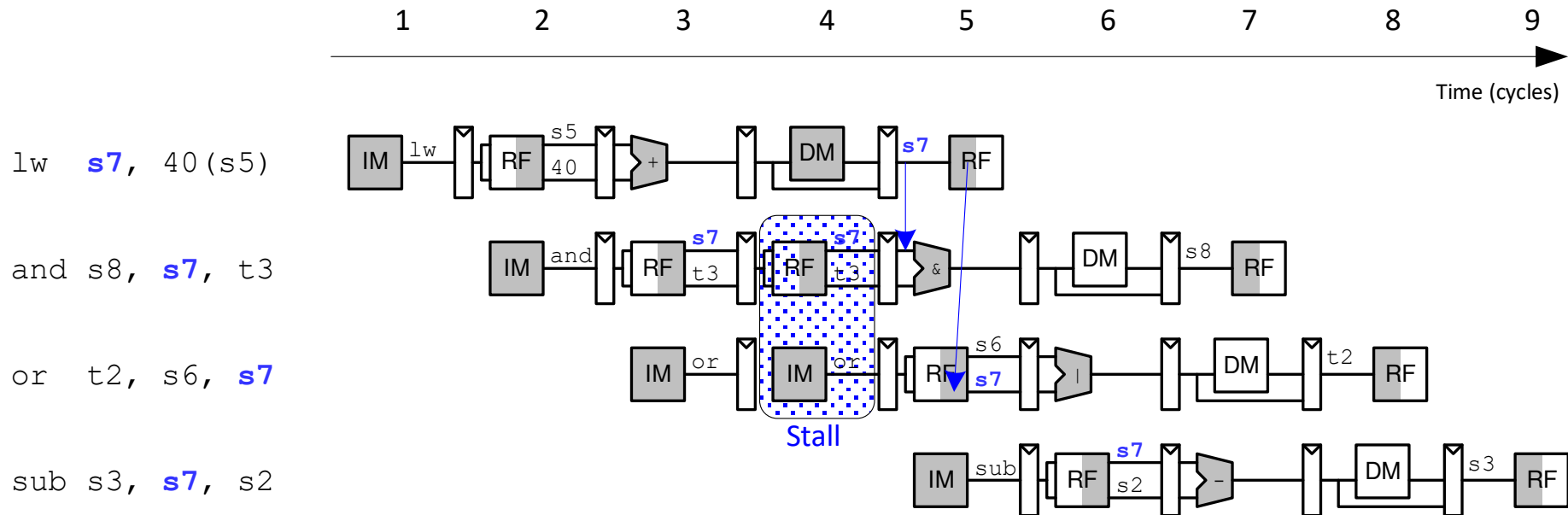
```
if      (( $Rs1E == RdM$ ) AND  $RegWriteM$ ) AND ( $Rs1E != 0$ ) // Case 1
        ForwardAE = 10
else if (( $Rs1E == RdW$ ) AND  $RegWriteW$ ) AND ( $Rs1E != 0$ ) // Case 2
        ForwardAE = 01
else    ForwardAE = 00                                     // Case 3
```

ForwardBE equations are similar (replace $Rs1E$ with $Rs2E$)

Data Hazard due to lw Dependency



Stalling to solve 1w Data Dependency



Stalling Logic

- Is either **source register in the Decode stage** the same as the **destination register in the Execute stage**?

AND

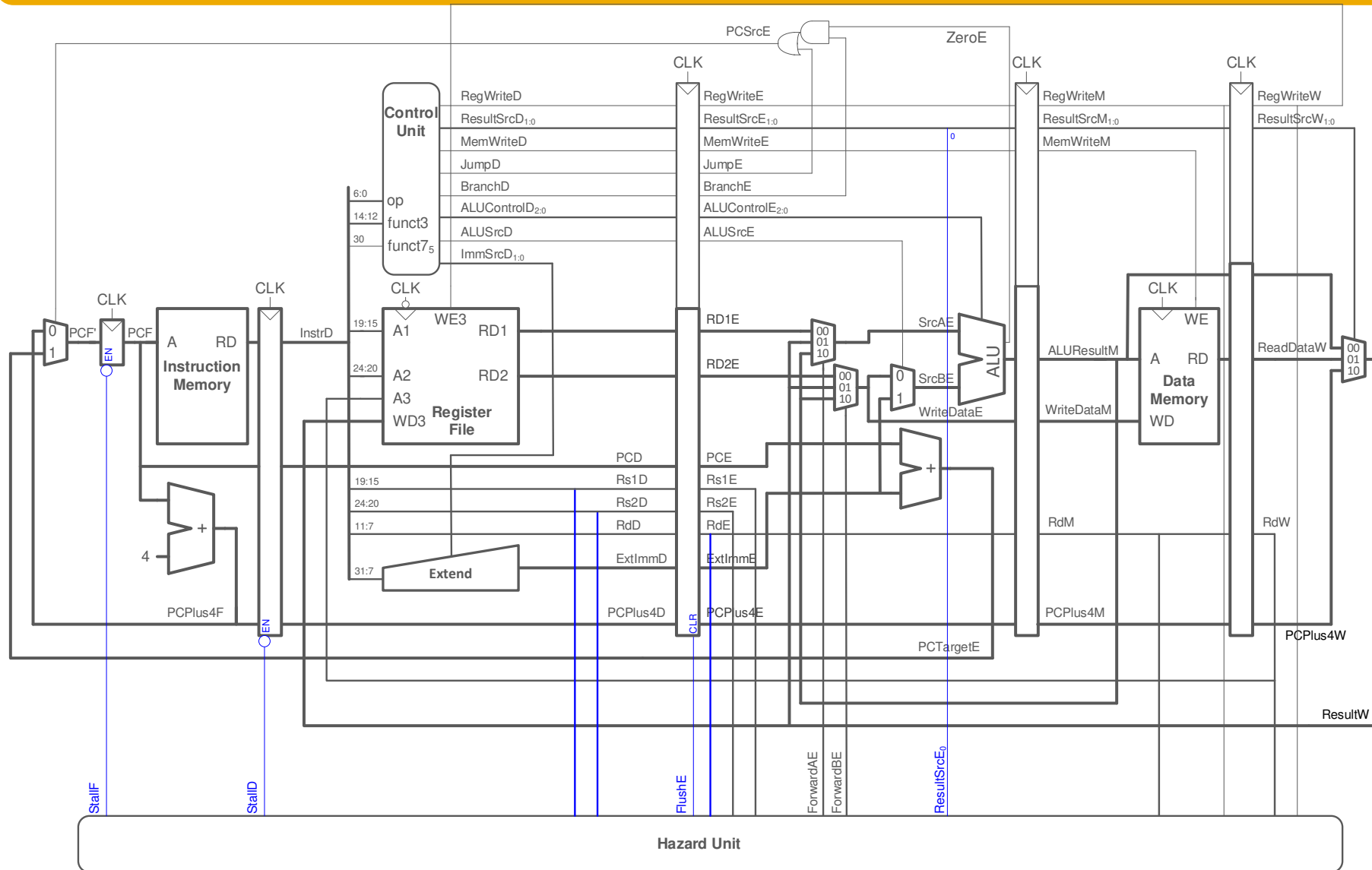
- Is the instruction in the **Execute stage a lw**?

$lwStall = ((Rs1D == RdE) \text{ OR } (Rs2D == RdE)) \text{ AND } ResultSrcE_1$

$StallF = StallD = FlushE = lwStall$

(Stall the Fetch and Decode stages, and flush the Execute stage.)

Stalling Hardware



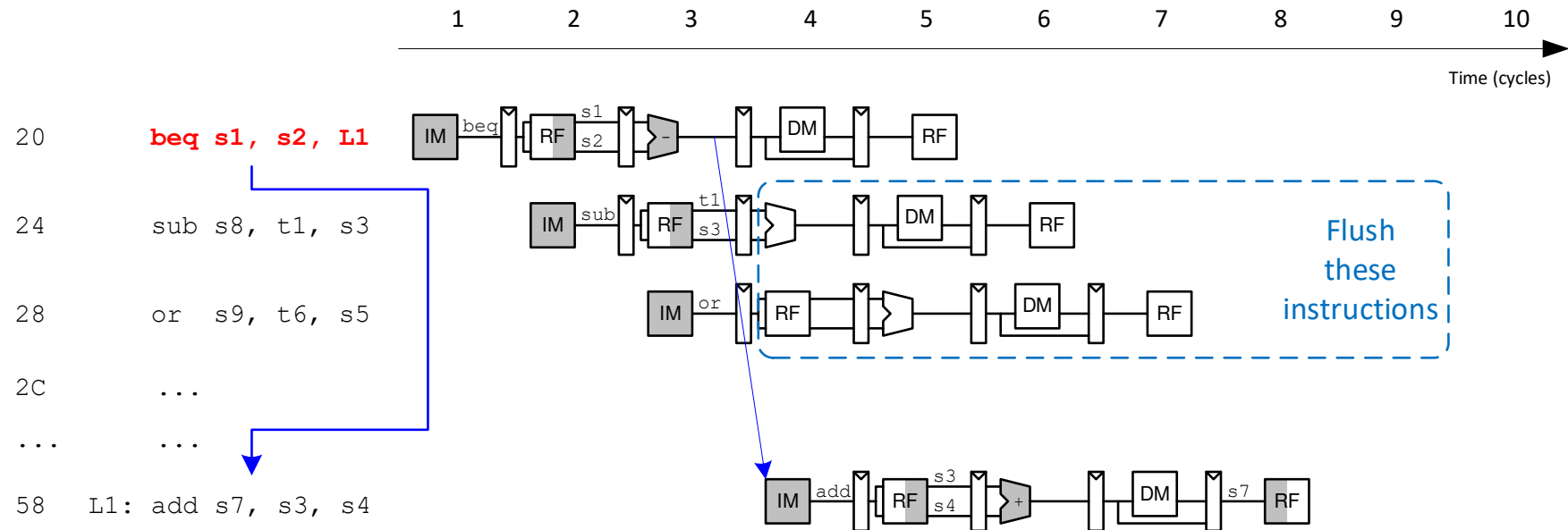
Chapter 7: Microarchitecture

Pipelined Processor Control Hazards

Control Hazards

- **beq:**
 - 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

Control Hazards



Branch misprediction penalty:

The number of instructions flushed when a branch is taken (in this case, 2 instructions)

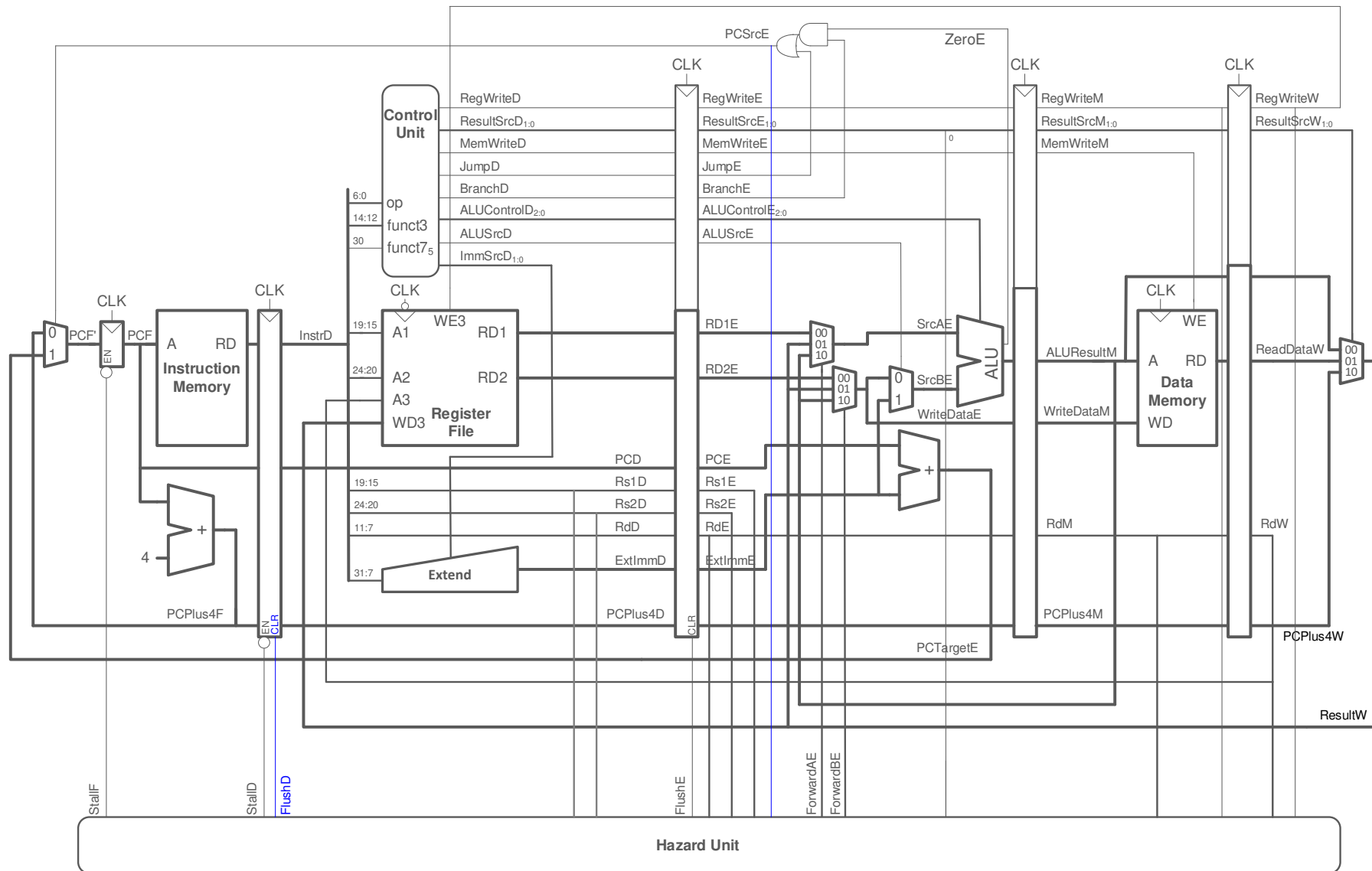
Control Hazards: Flushing Logic

- If branch is taken in execute stage, need to flush the instructions in the Fetch and Decode stages
 - Do this by clearing Decode and Execute Pipeline registers using *FlushD* and *FlushE*
- **Equations:**

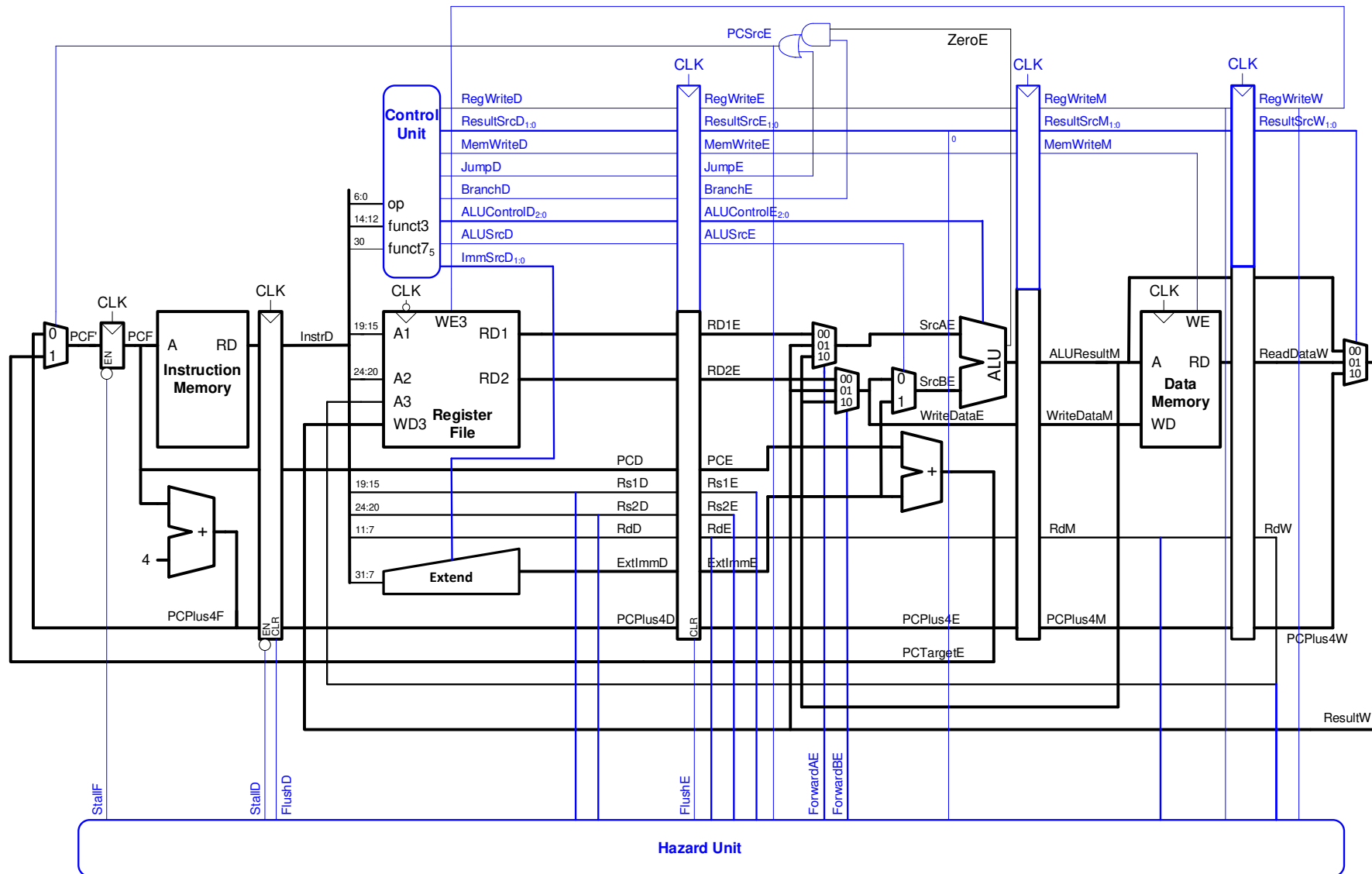
$$FlushD = PCSrcE$$

$$FlushE = lwStall \text{ OR } PCSrcE$$

Control Hazards: Flushing Hardware



RISC-V Pipelined Processor with Hazard Unit



Chapter 7: Microarchitecture

Pipelined Performance

Pipelined Processor Performance Example

- **SPECINT2000 benchmark:**
 - 25% loads
 - 10% stores
 - 13% branches
 - 52% R-type
- **Suppose:**
 - 40% of loads used by next instruction
 - 50% of branches mispredicted
- **What is the average CPI?** (Ideally it's 1, but...)

Pipelined Processor Performance Example

Pipelined processor critical path:

$T_{c_pipelined} = \max \text{ of}$

$$t_{pcq} + t_{mem} + t_{setup}$$

Fetch

$$2(t_{RFread} + t_{setup})$$

Decode

$$t_{pcq} + 4t_{mux} + t_{ALU} + t_{AND-OR} + t_{setup}$$

Execute

$$t_{pcq} + t_{mem} + t_{setup}$$

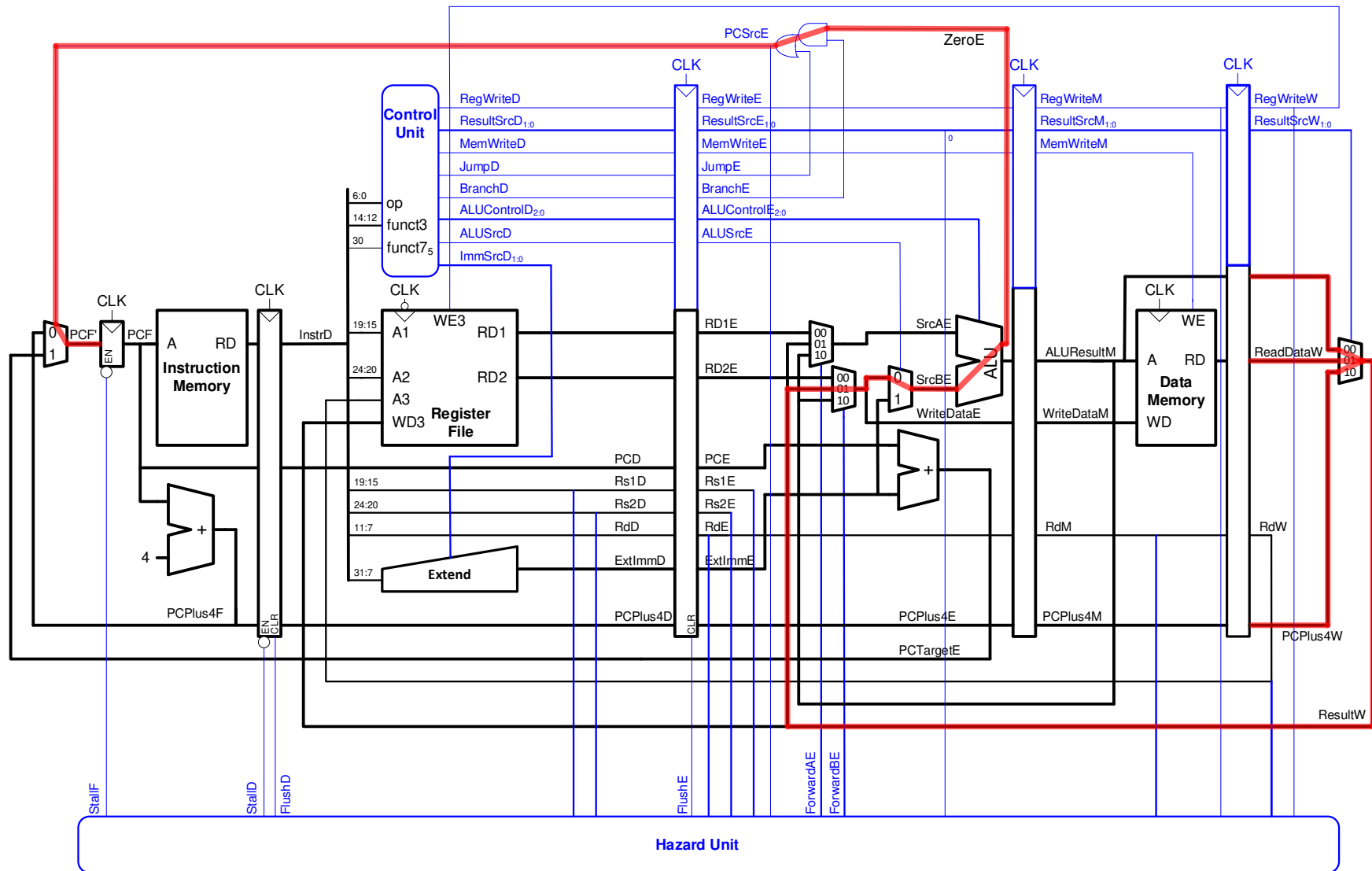
Memory

$$2(t_{pcq} + t_{mux} + t_{RFwrite})$$

Writeback

- Decode and Writeback stages **both use the register file** in each cycle
- So each stage gets half of the cycle time ($T_c/2$) to do their work
- Or, stated a different way, **2x of their work** must fit in a cycle (T_c)

Pipelined Critical Path: Execute Stage



Pipelined Performance Example

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

$$T_{c_pipelined} = t_{pcq} + 4t_{mux} + t_{ALU} + t_{AND-OR} + t_{setup}$$

$$=$$

Pipelined Performance Example

Program with 100 billion instructions

$$\begin{aligned}\text{Execution Time} &= (\# \text{ instructions}) \times \text{CPI} \times T_c \\ &= (100 \times 10^9)(1.23)(350 \times 10^{-12}) \\ &= \mathbf{43 \text{ seconds}}\end{aligned}$$

Processor Performance Comparison

Processor	Execution Time (seconds)	Speedup (single-cycle as baseline)
Single-cycle	75	1
Multicycle	155	0.5
Pipelined	43	1.7