

CSE 112: Computer Organization

Instructor: Sujay Deb

Lecture 18



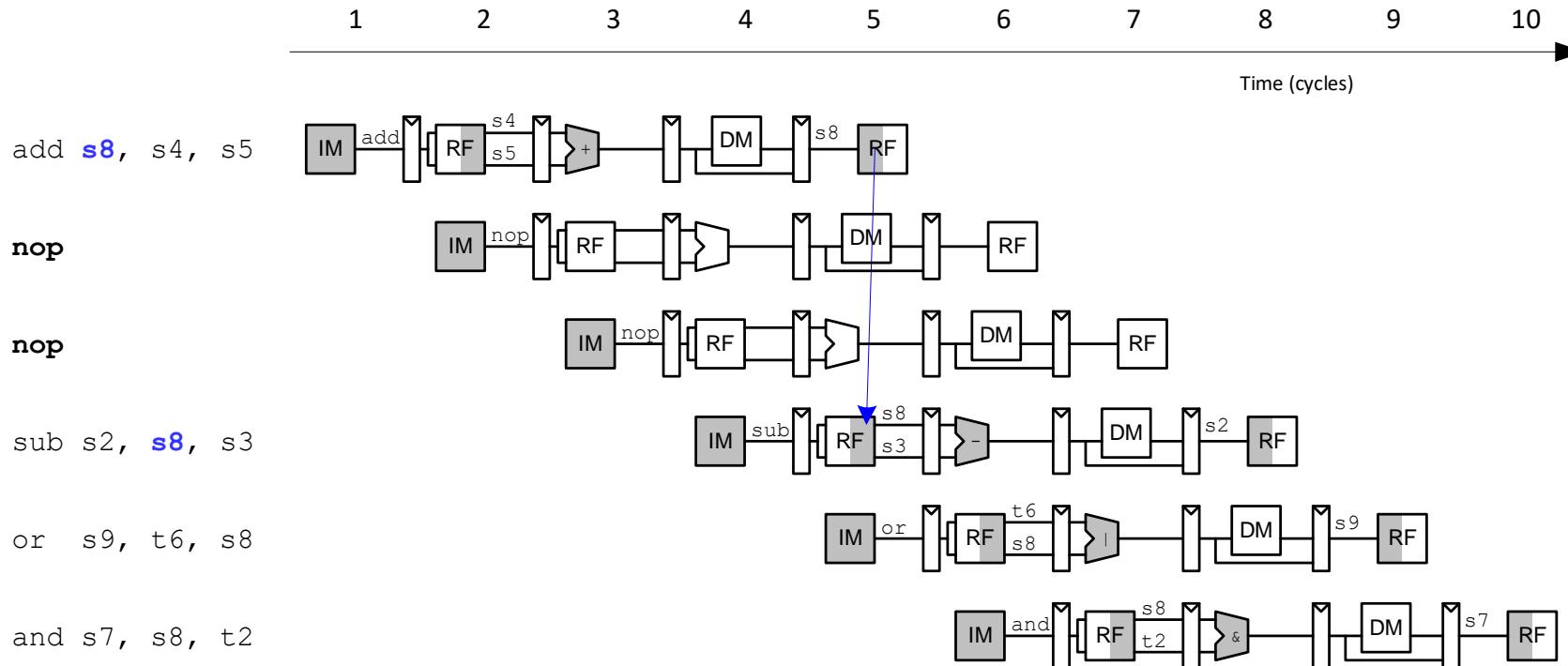
INDRAPRASTHA INSTITUTE of
INFORMATION TECHNOLOGY
DELHI



Pipelined Processor Hazards

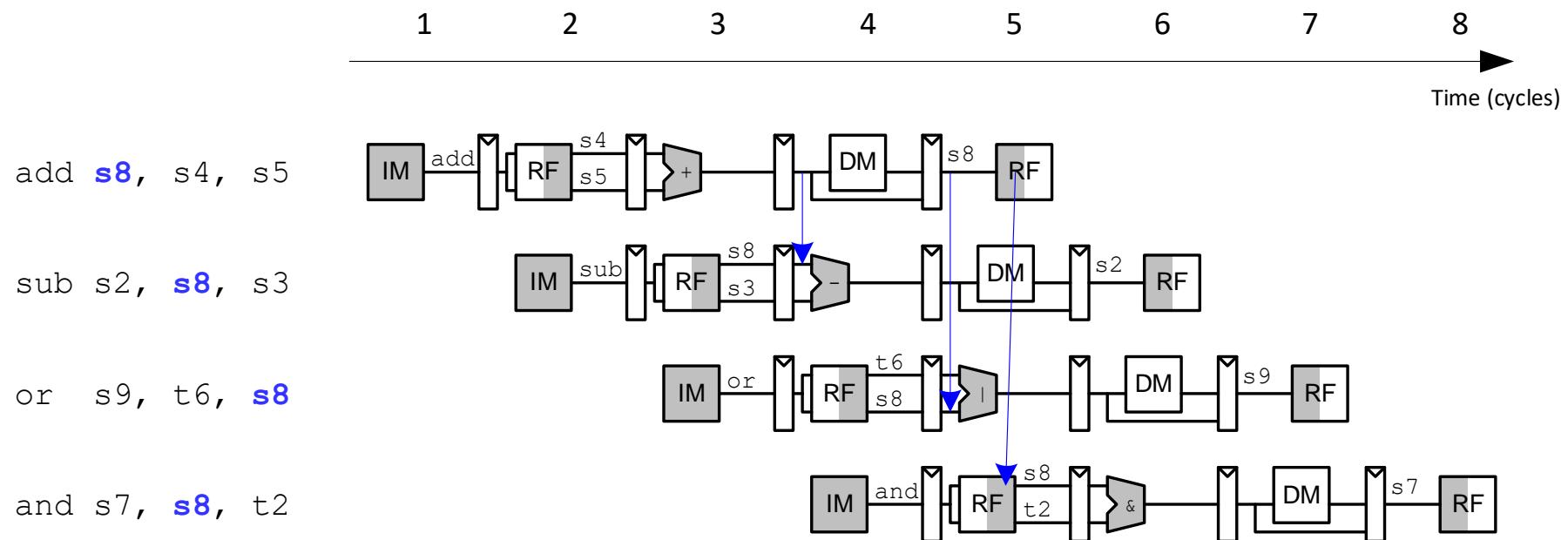
Handling Data Hazards

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



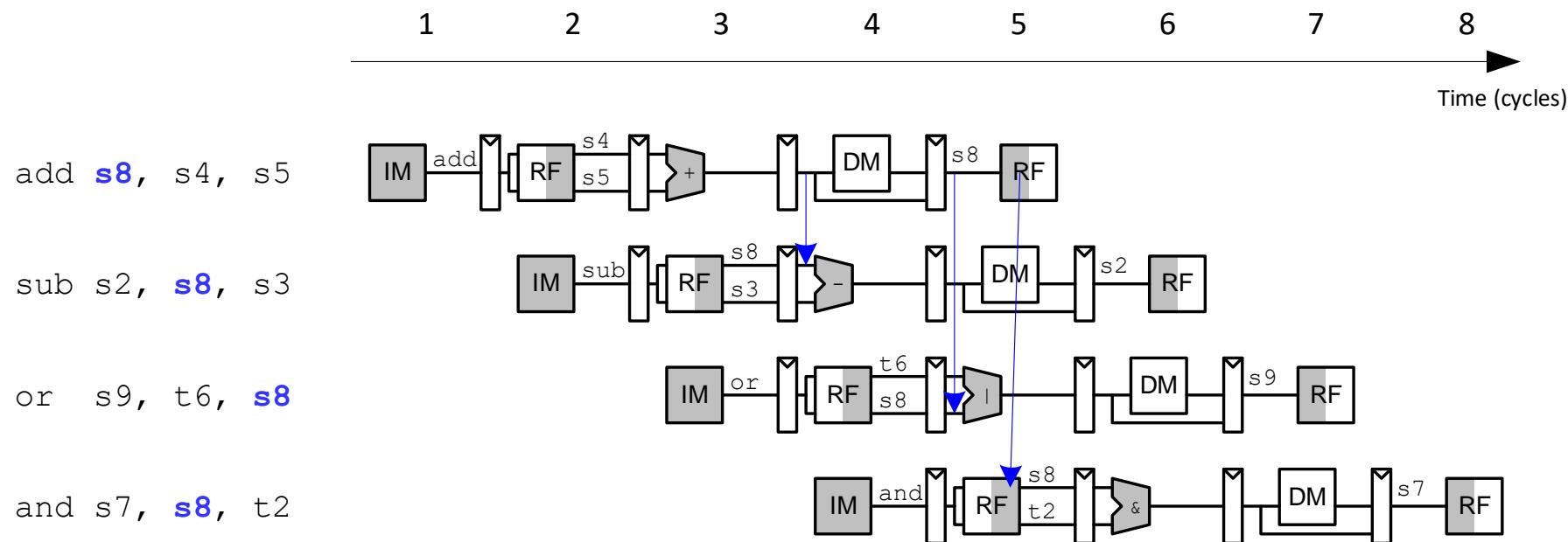
Data Forwarding

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

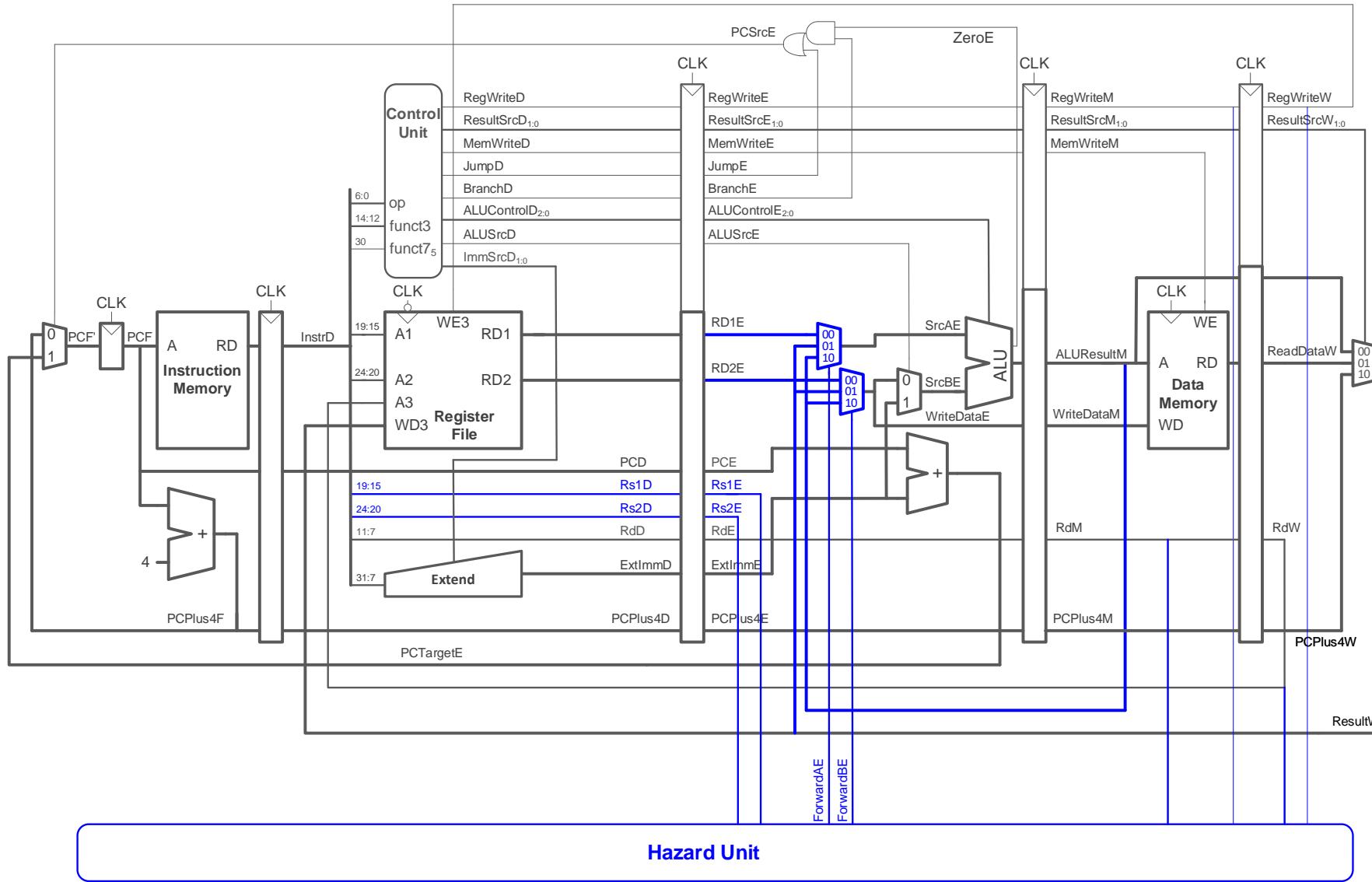


Data Forwarding

- 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

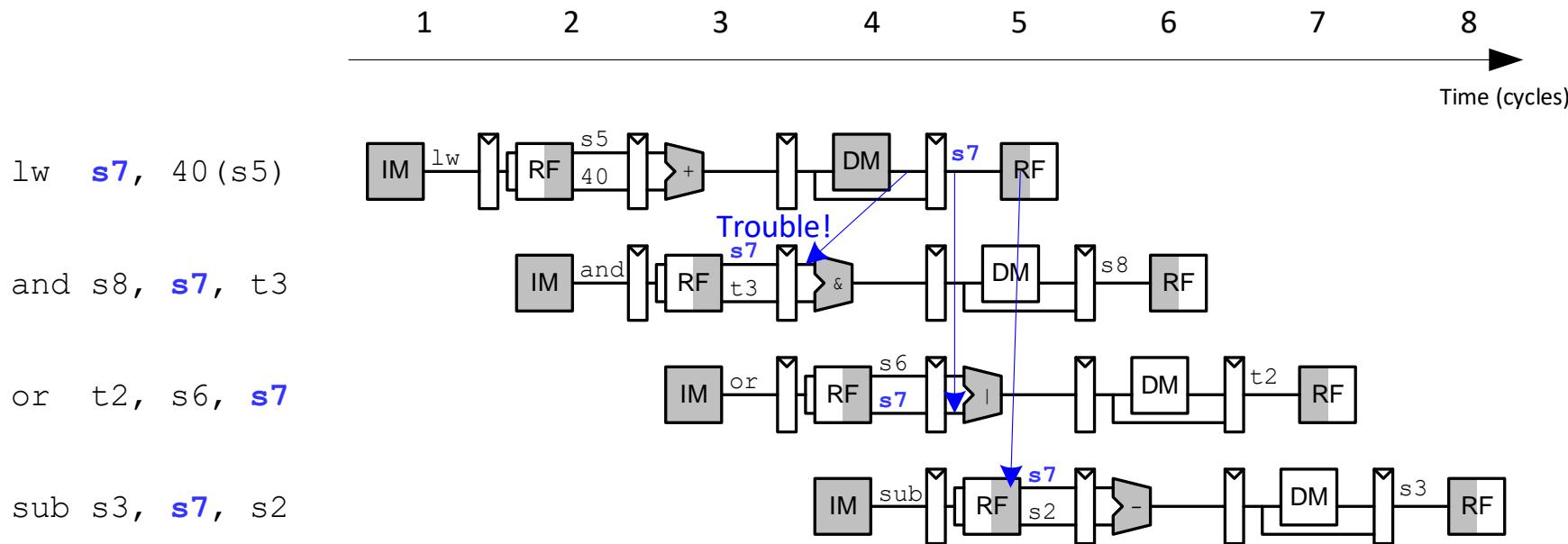
- **Case 1:** Execute stage $Rs1$ or $Rs2$ matches **Memory** stage Rd ?
Forward from Memory stage
- **Case 2:** Execute stage $Rs1$ or $Rs2$ matches **Writeback** stage Rd ?
Forward from Writeback stage
- **Case 3:** Otherwise use value read from register file (as usual)

Equations for $Rs1$:

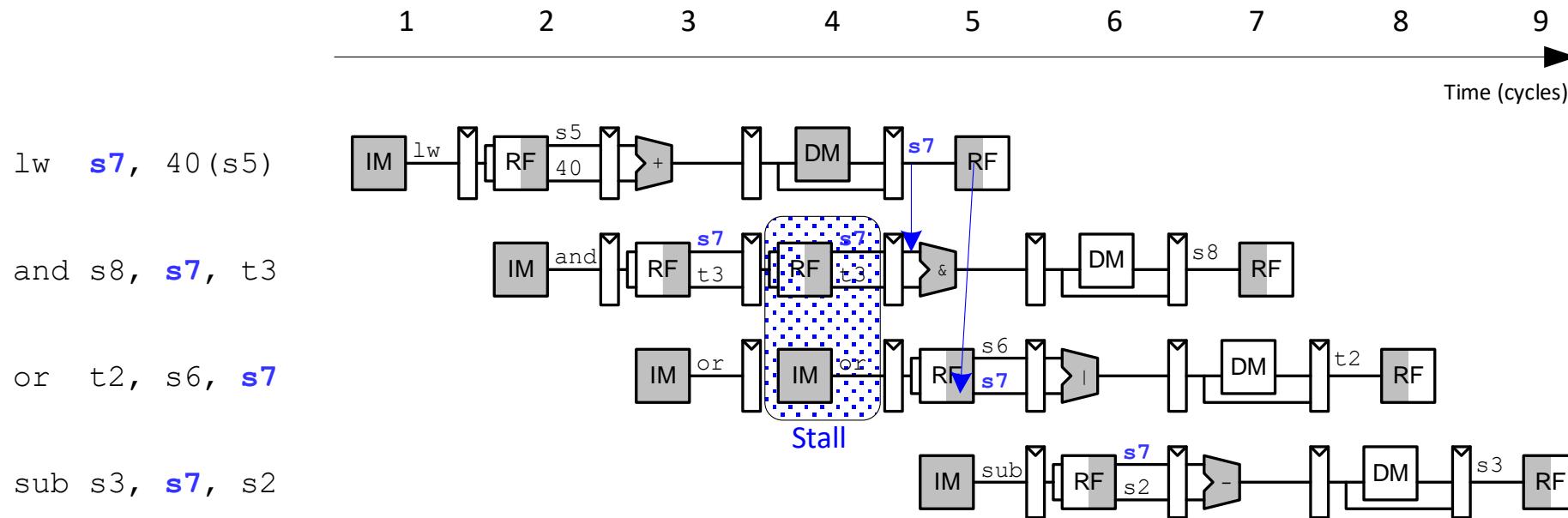
```
if      ((Rs1E == RdM) AND RegWriteM) AND (Rs1E != 0) // Case 1
        ForwardAE = 10
else if ((Rs1E == RdW) AND RegWriteW) AND (Rs1E != 0) // Case 2
        ForwardAE = 01
else      ForwardAE = 00                                // Case 3
```

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

Data Hazard due to 1w Dependency



Stalling to solve l_w Data Dependency



Stalling Logic

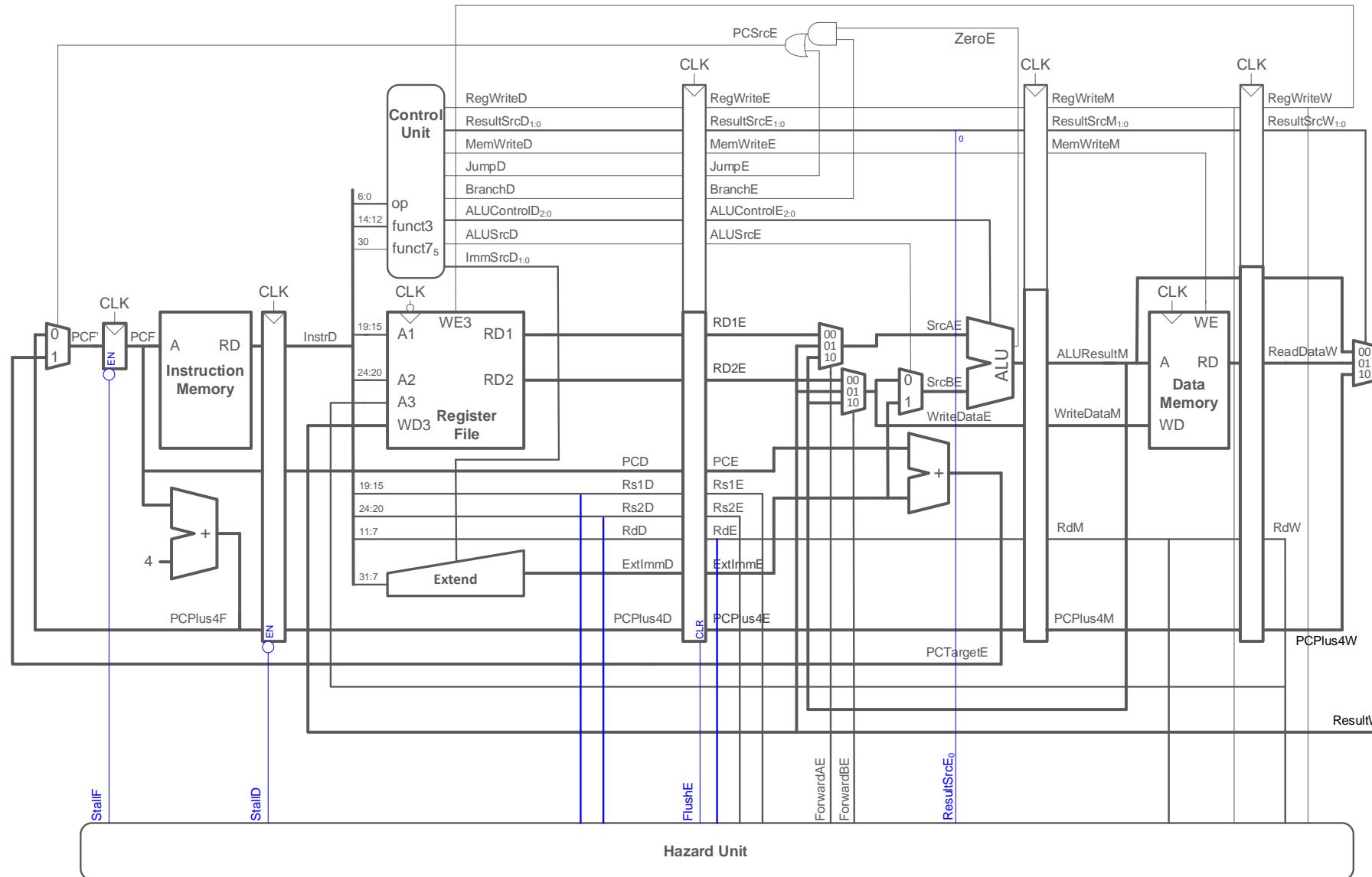
- Is either **source register in the Decode stage** the same as the **destination register in the Execute stage?**
AND
- Is the instruction in the **Execute stage a lw?**

$IwStall = ((Rs1D == RdE) \text{ OR } (Rs2D == RdE)) \text{ AND } ResultSrcE_0$

$StallF = StallD = FlushE = IwStall$

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

Stalling Hardware

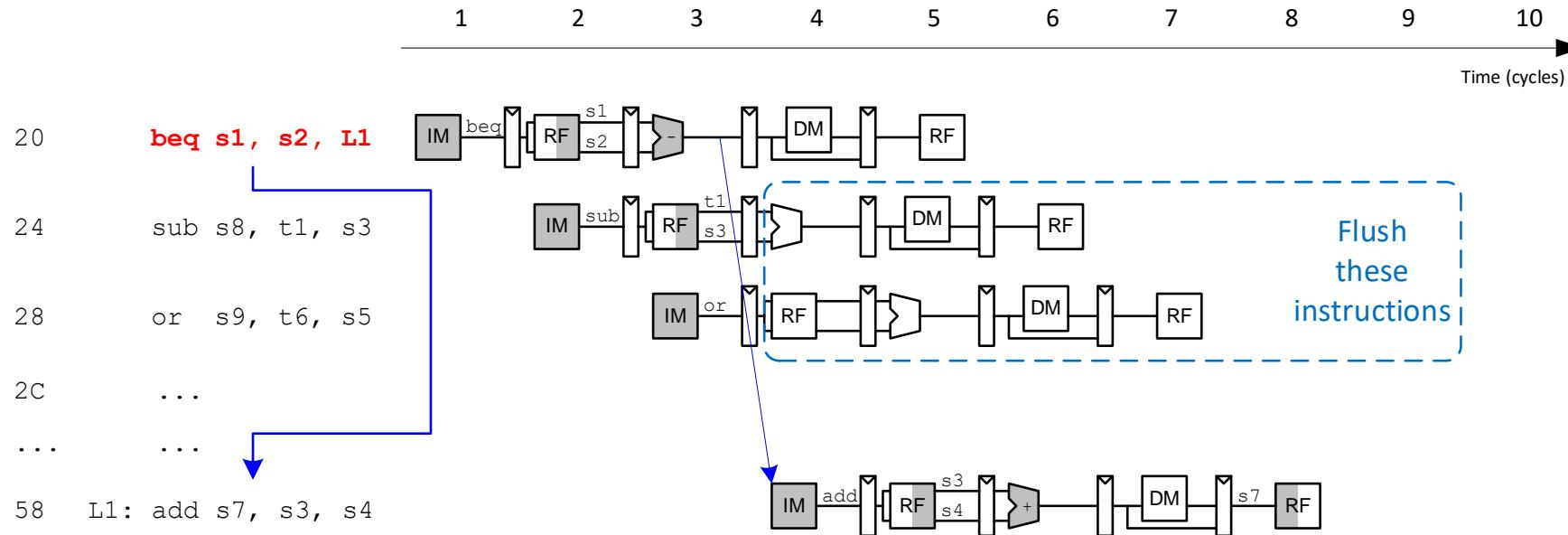


Pipelined Processor Control Hazards

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)

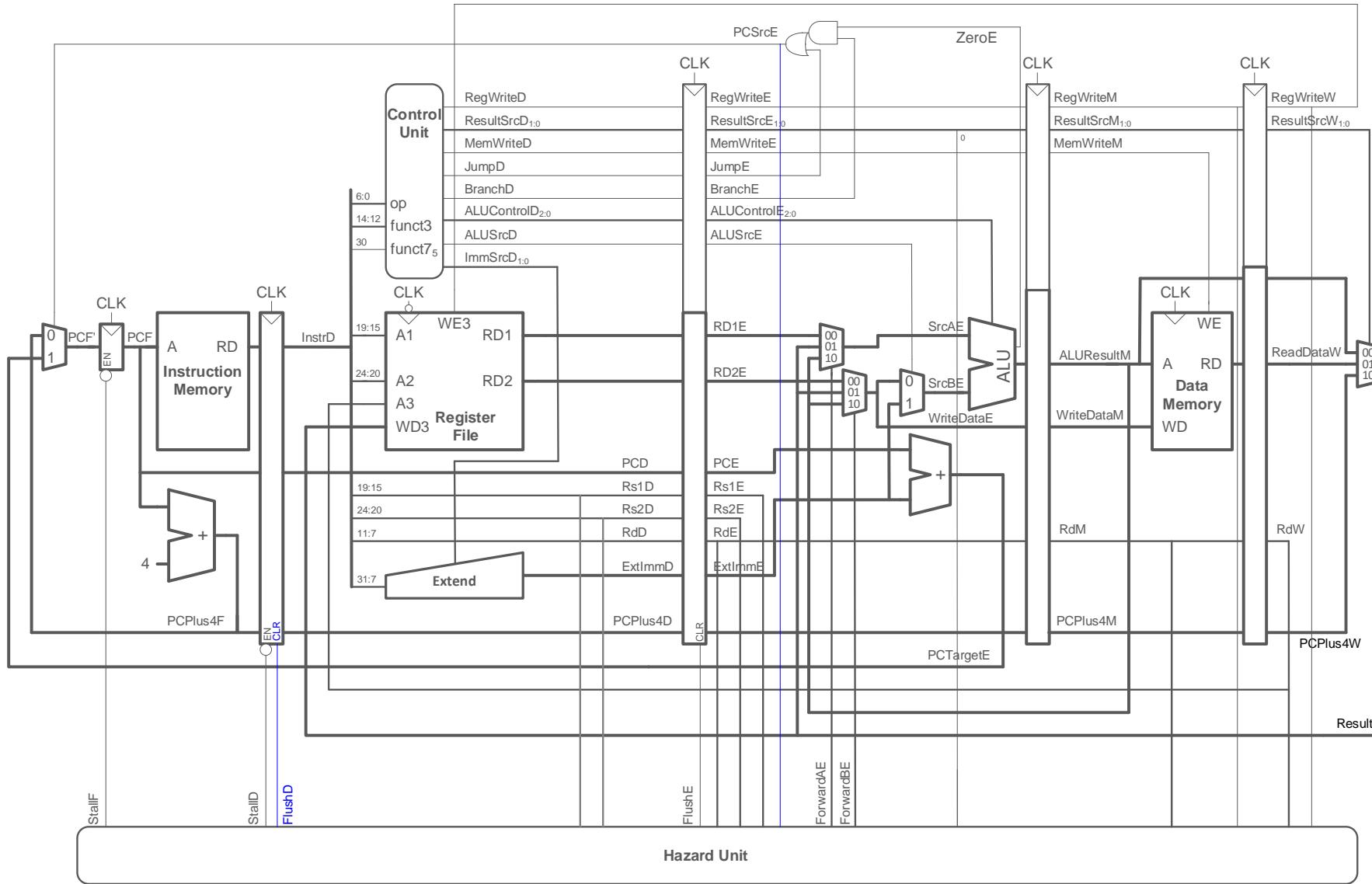
Control Hazards: Flushing Logic

- 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:**

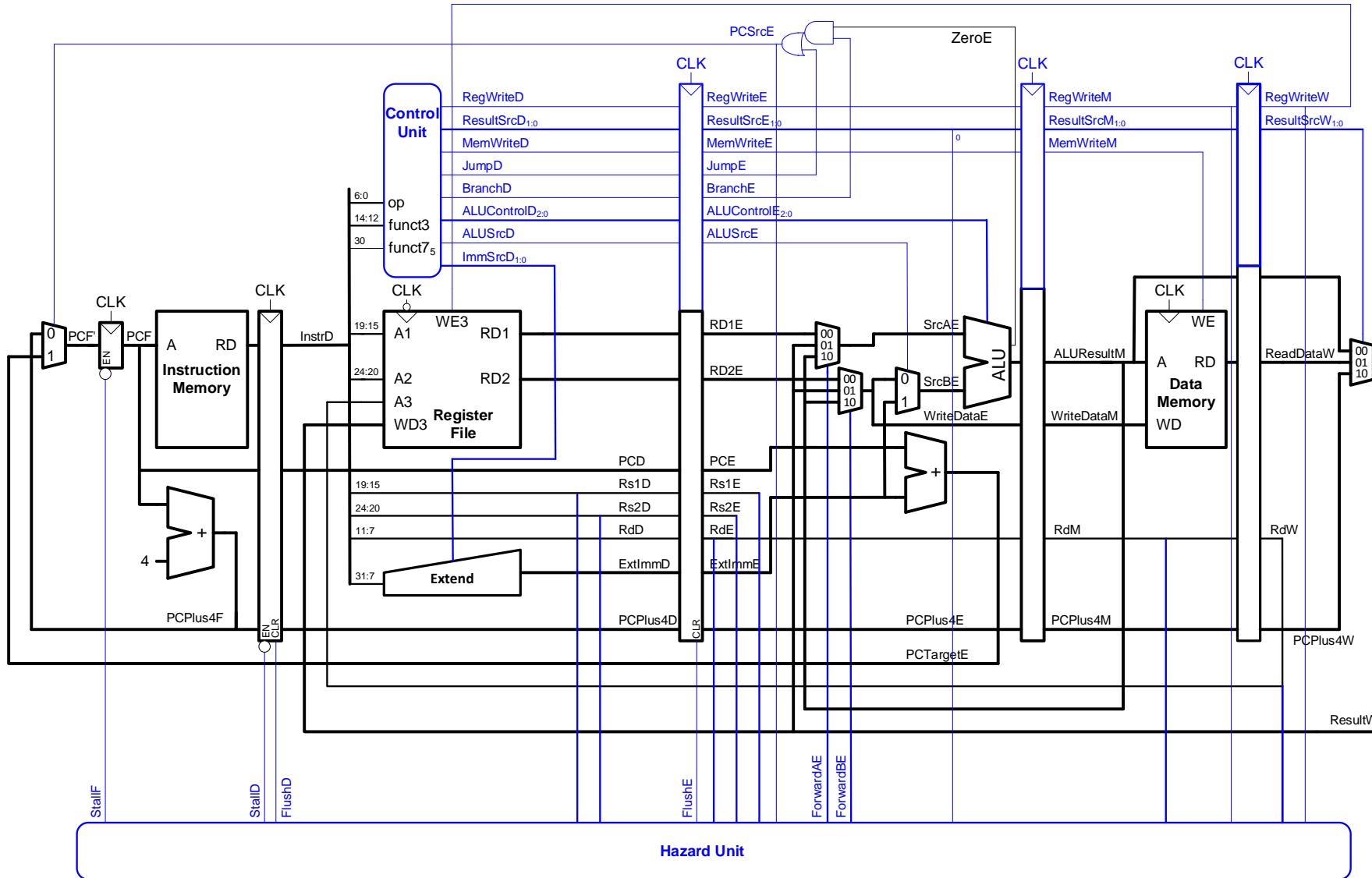
$$\textit{FlushD} = \textit{PCSrcE}$$

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

Control Hazards: Flushing Hardware



RISC-V Pipelined Processor with Hazard Unit



Summary of Hazard Logic

Data hazard logic (shown for SrcA of ALU):

```
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
```

Load word stall logic:

$lwStall = ((Rs1D == RdE) \text{ OR } (Rs2D == RdE)) \text{ AND } ResultSrcE_0$
 $StallF = StallD = lwStall$

Control hazard flush:

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

Class Interaction # 20

