CSE 112: Computer Organization

Instructor: Sujay Deb

Lecture 16

INDRAPRASTHA INSTITUTE *of* INFORMATION TECHNOLOGY DELHI

AMS Lab @IIITD

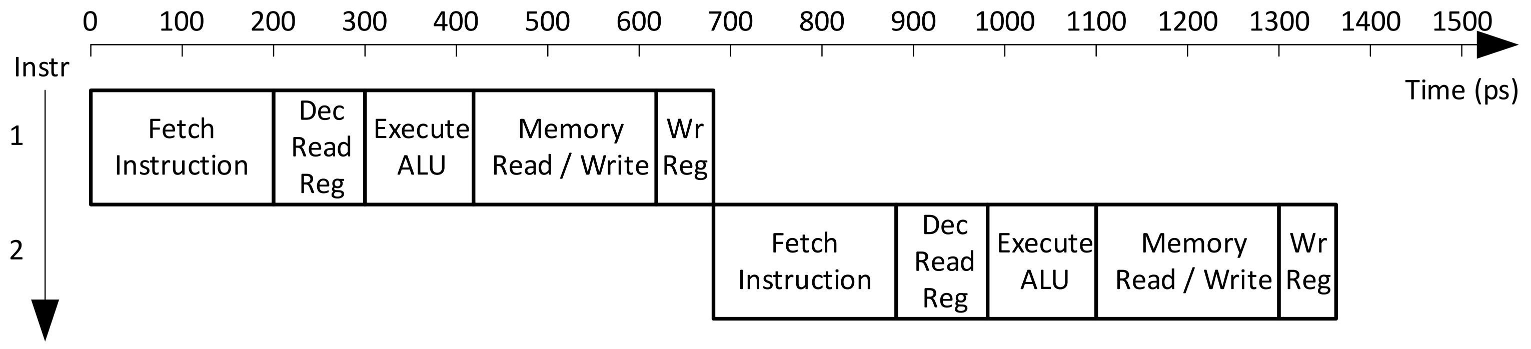
Pipelined RISC-V Processor

Pipelined RISC-V Processor

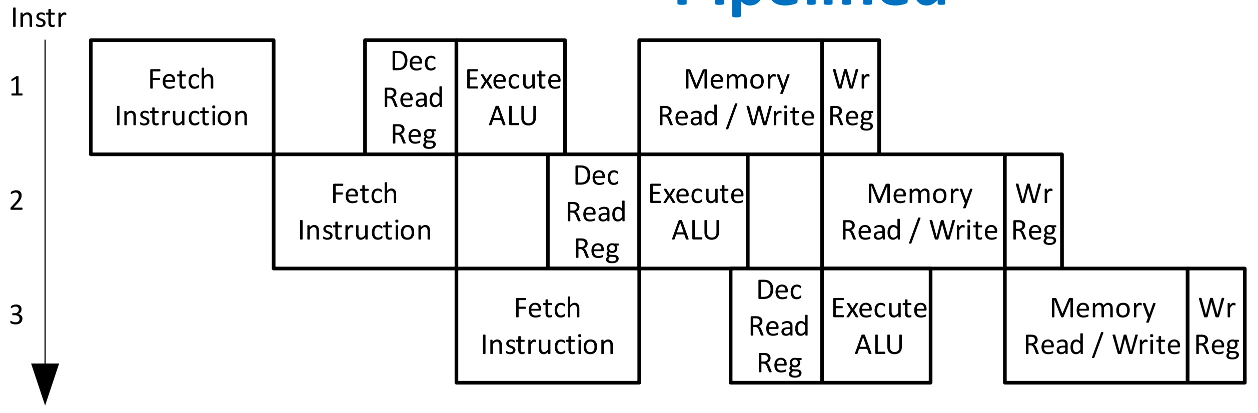
- **Temporal parallelism**
- Divide single-cycle processor into **5 stages**:
  - Fetch
  - Decode
  - Execute
  - Memory
  - Writeback
- Add **pipeline registers** between stages

Single-Cycle vs. Pipelined Processor

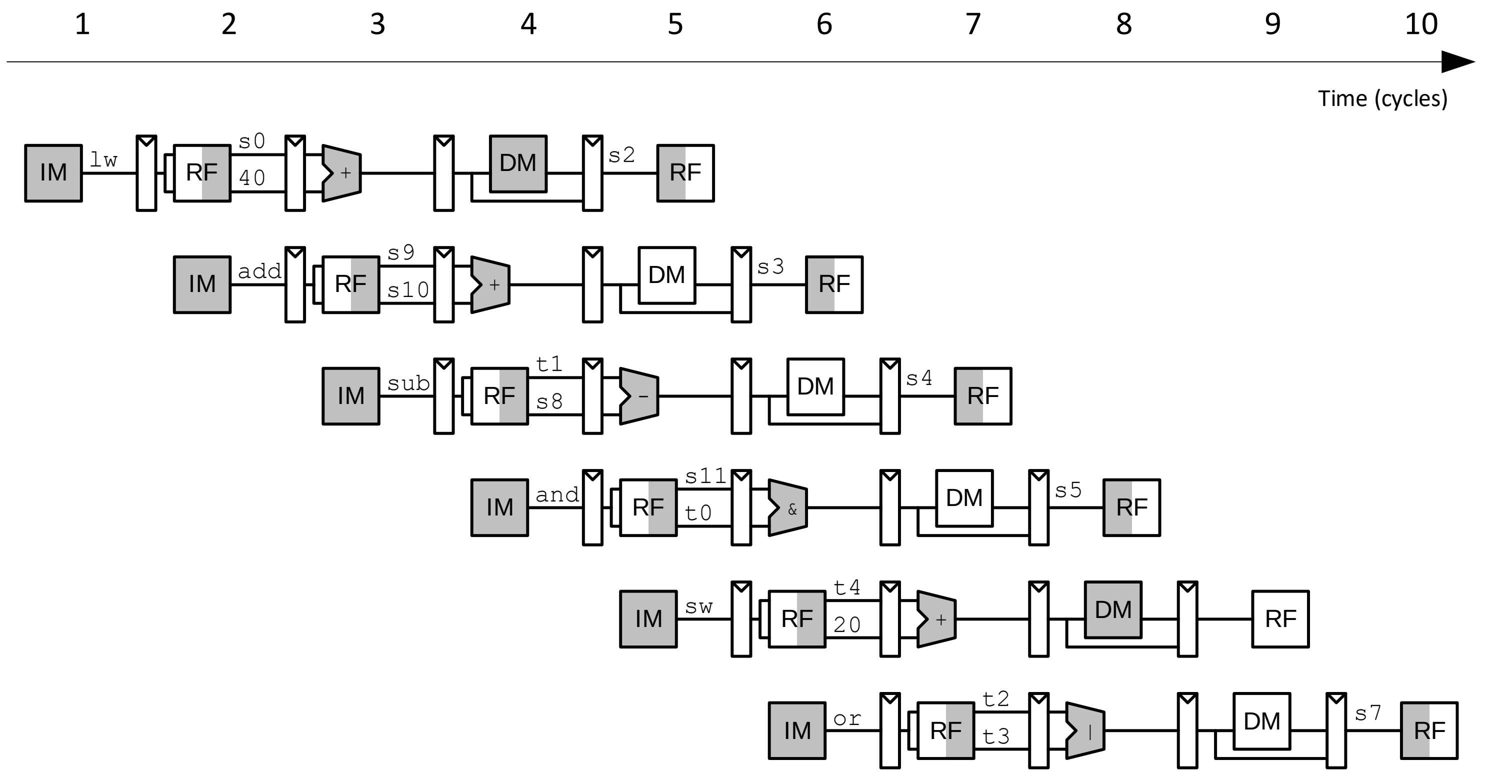
Single-Cycle



Pipelined

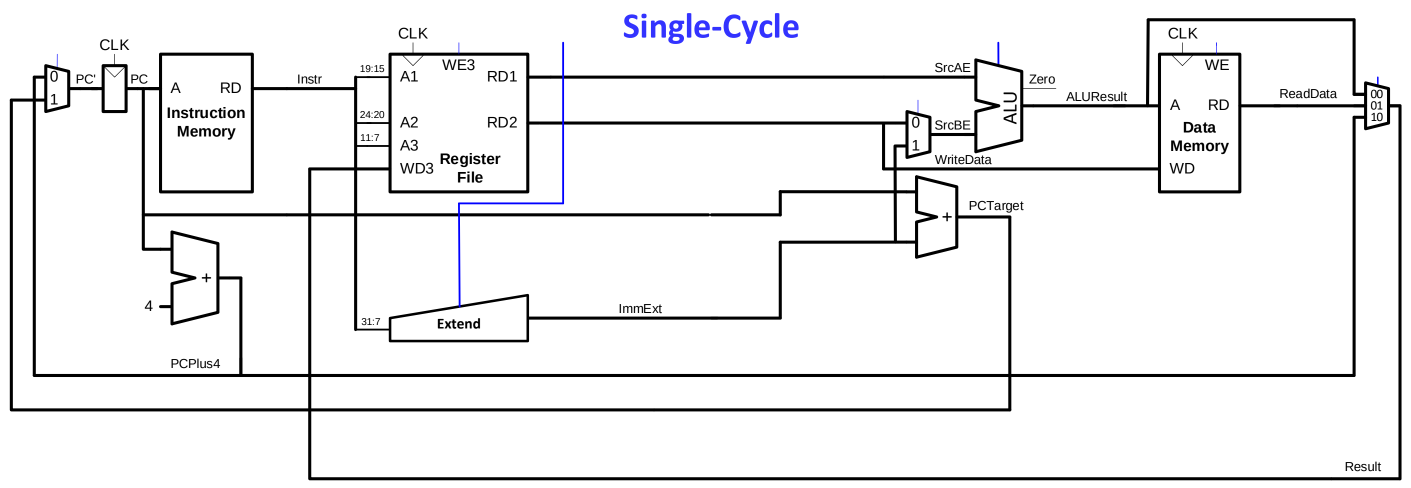


Pipelined Processor Abstraction

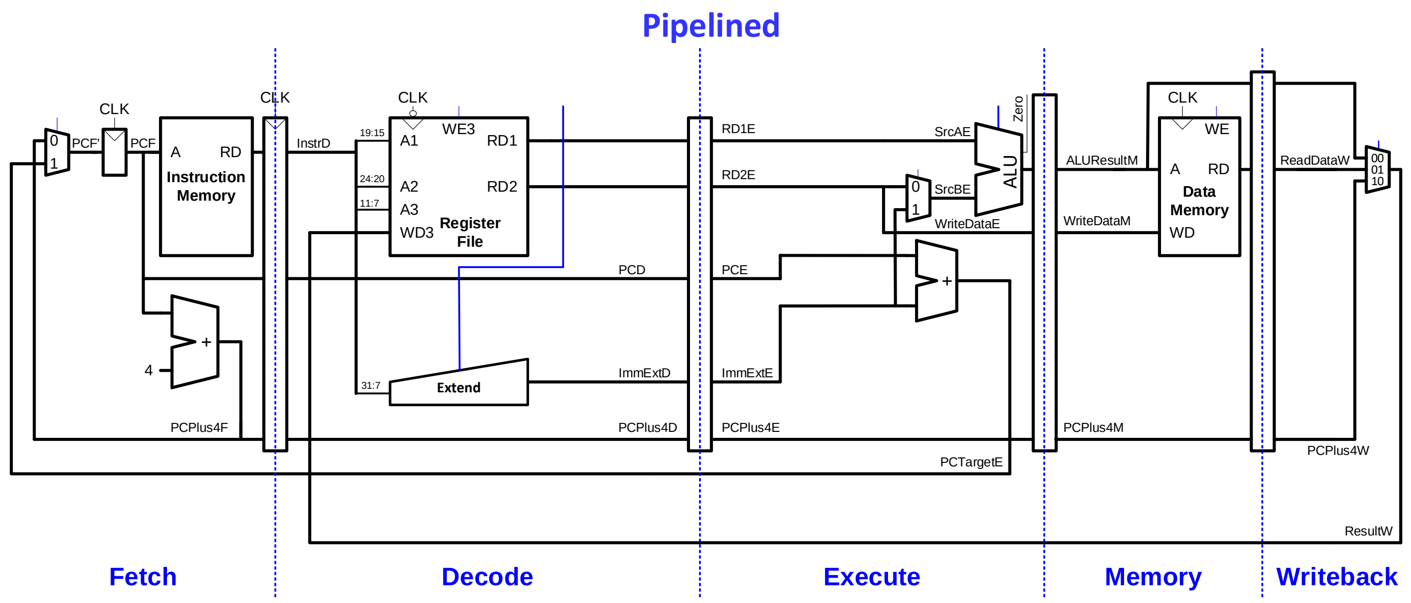


```
lw  s2, 40(s0)
add s3, s9, s10
sub s4, t1, s8
and s5, s11, t0
sw  s6, 20(t4)
or  s7, t2, t3
```

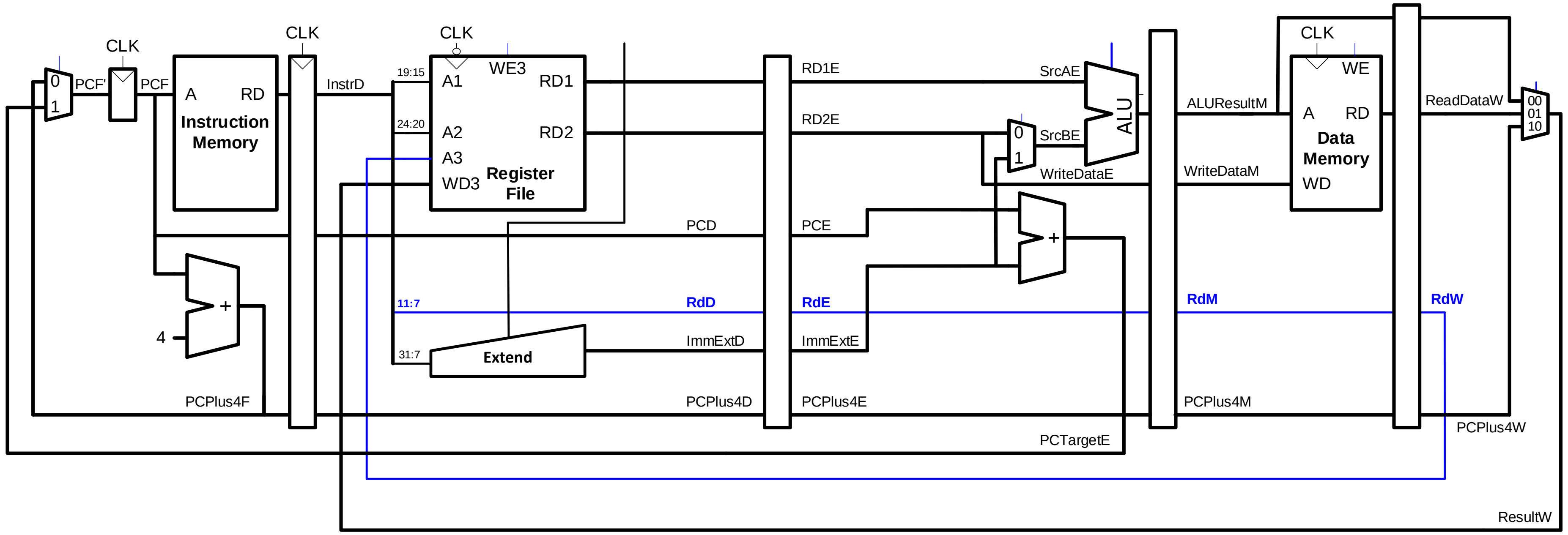
Single-Cycle & Pipelined Datapaths



Signals in Pipelined Processor are appended with first letter of stage (i.e., PCF, PCD, PCE).

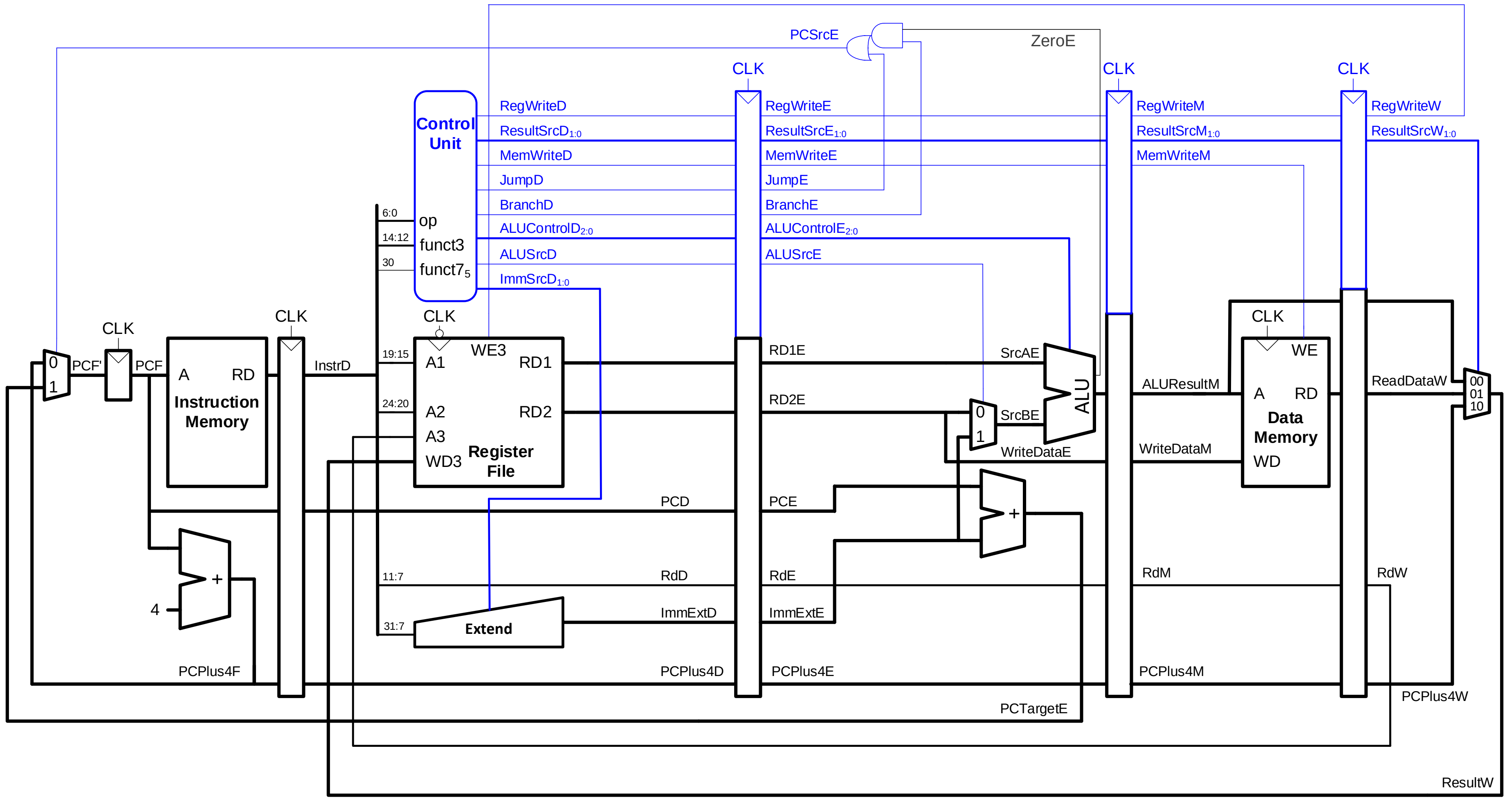


Corrected Pipelined Datapath



- *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)

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)

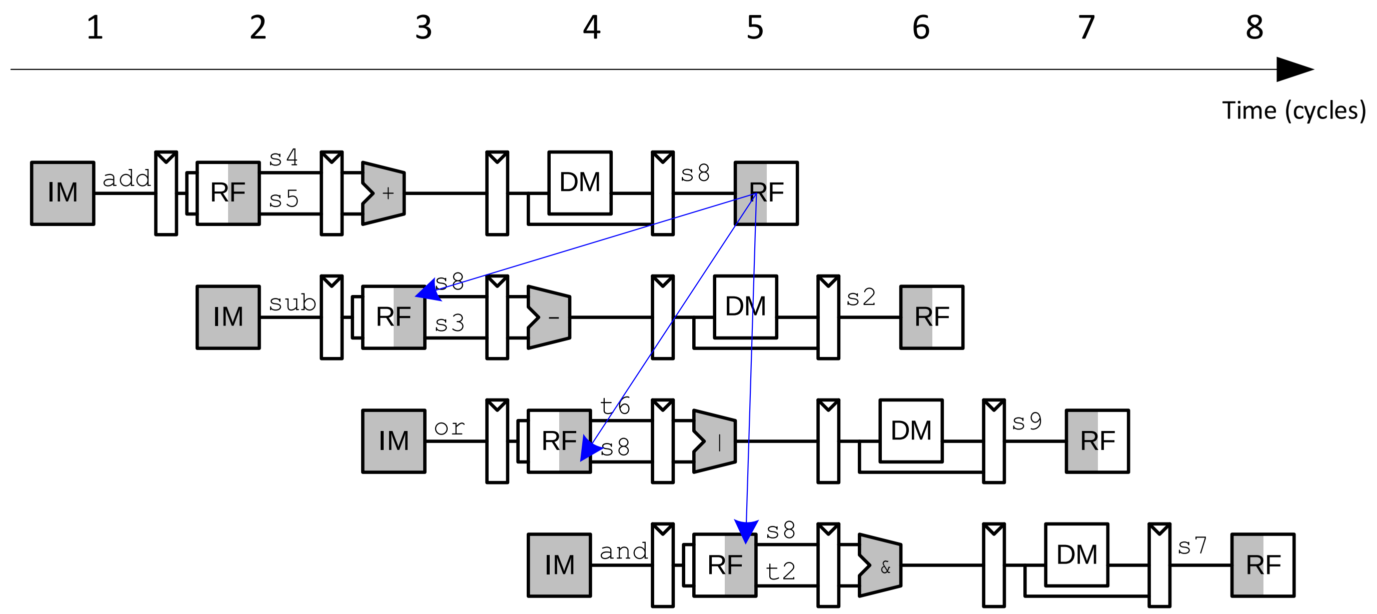
Data Hazard

```
add s8, s4, s5

sub s2, s8, s3

or  s9, t6, s8

and s7, s8, t2
```

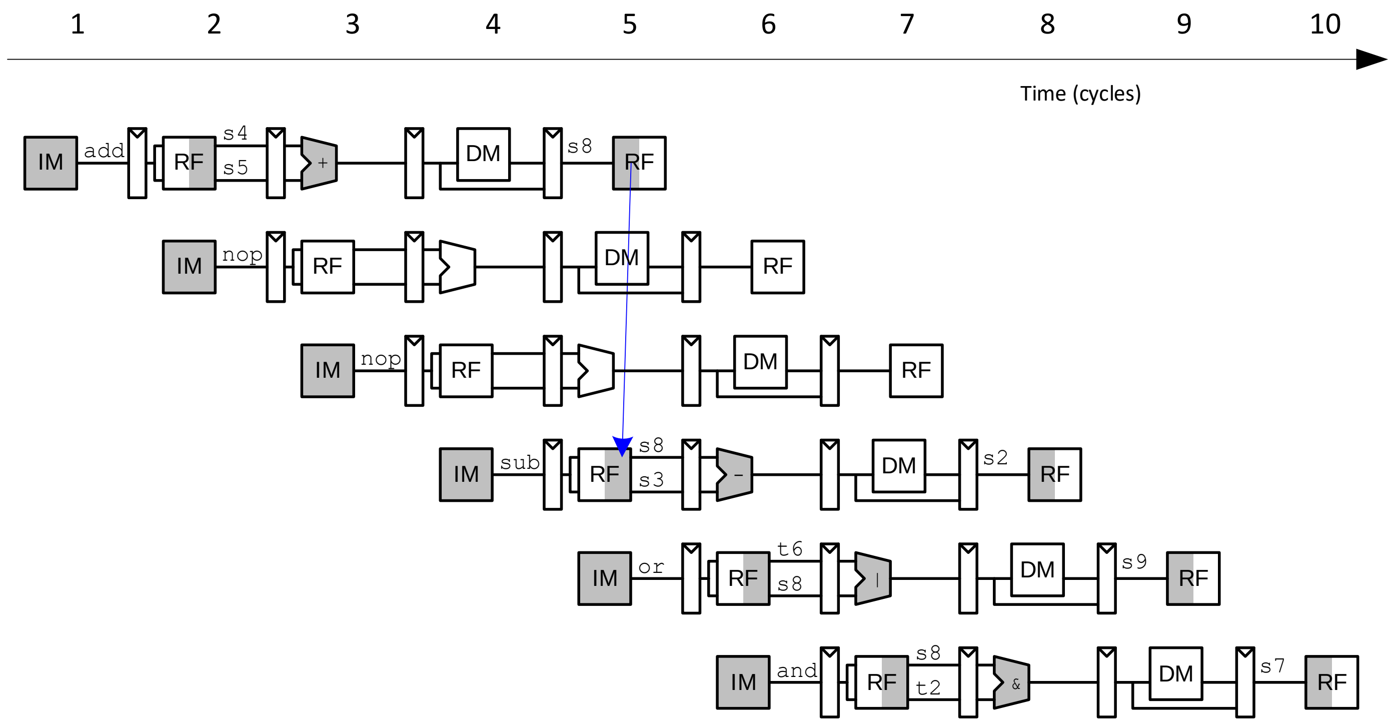


Handling Data Hazards

Handling Data Hazards

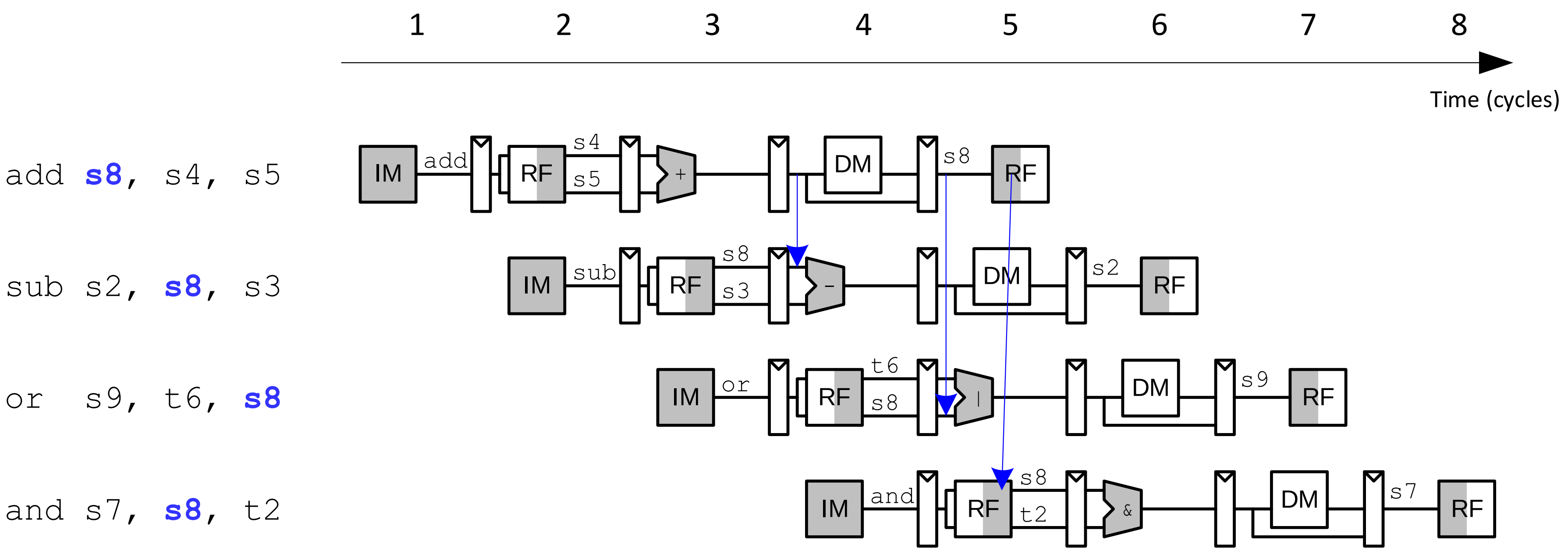
- **Insert** enough **nops** for result to be ready
- Or move independent useful instructions forward

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



Data Forwarding

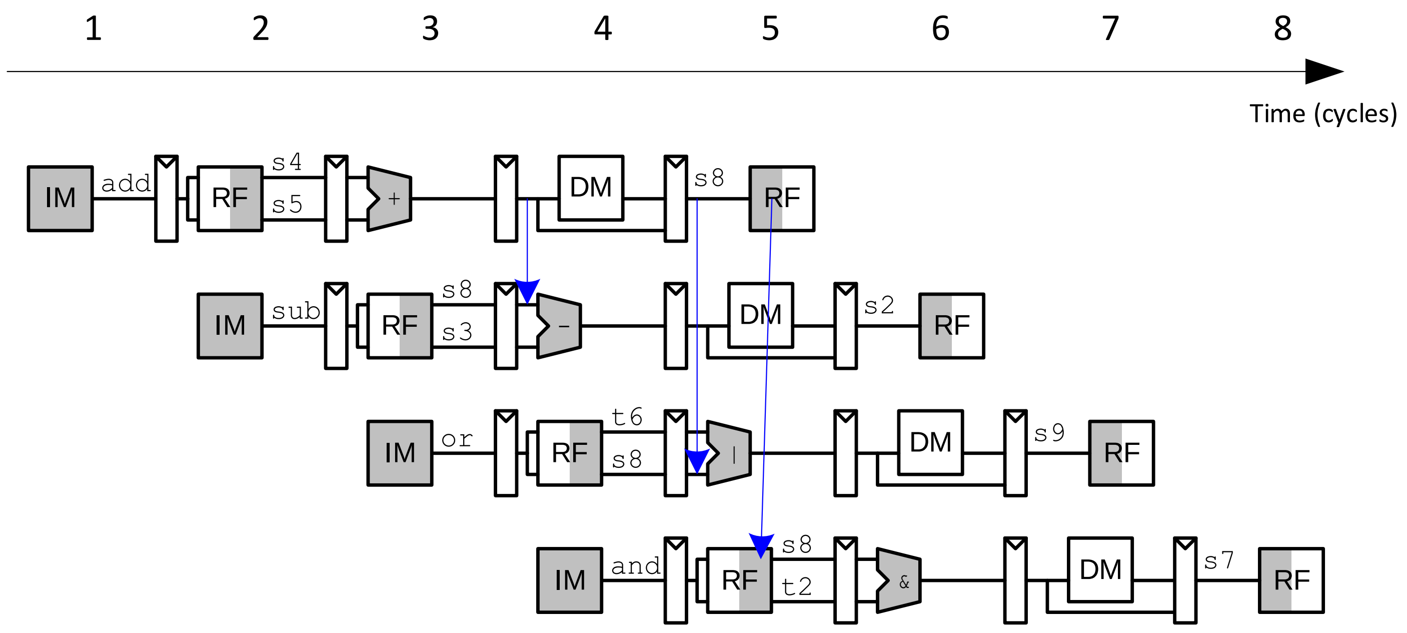
- Data is **available on internal busses** before it is written back to the register file (RF).
- **Forward data** from internal busses **to Execute stage**.



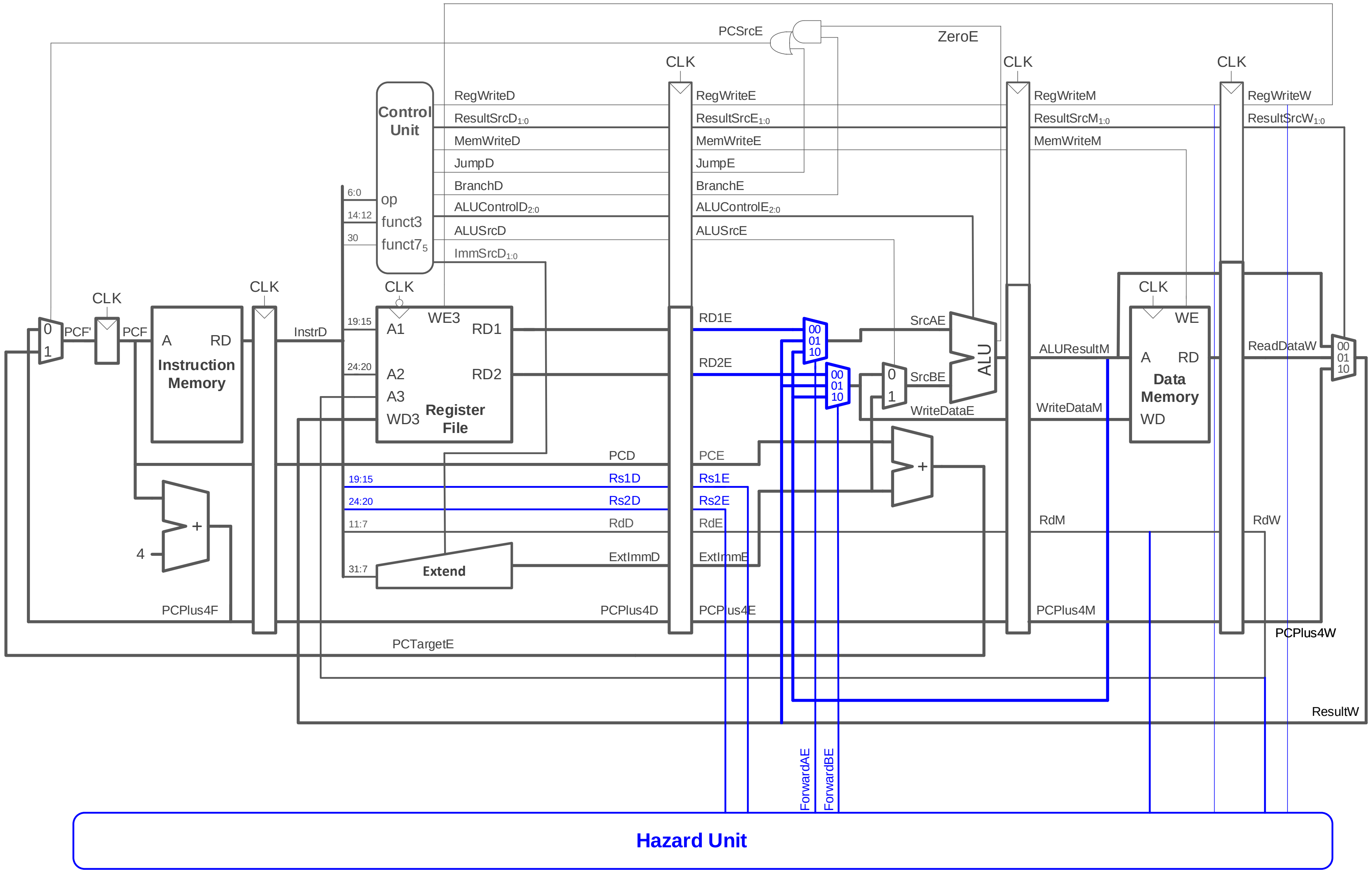
Data Forwarding

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

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



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**)*

Class Interaction #18