CSE 112: Computer Organization

Instructor: Sujay Deb

Lecture 19

INDRAPRASTHA INSTITUTE *of* INFORMATION TECHNOLOGY DELHI

AMS Lab @IIITD

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

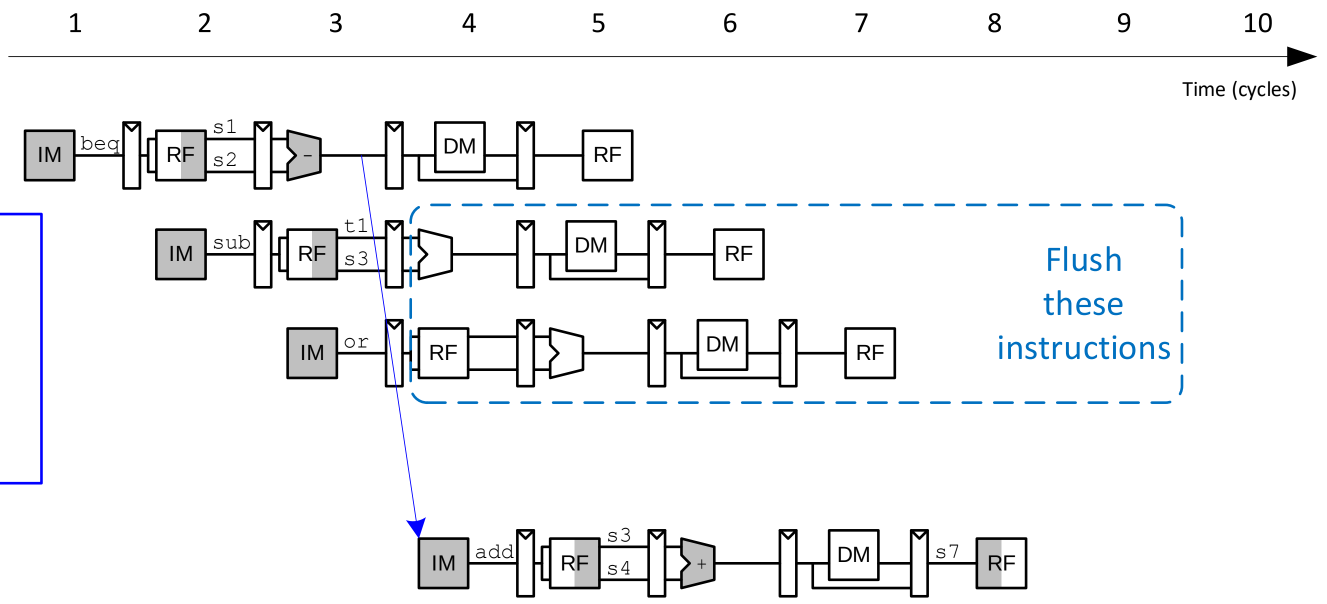
```
20       beq s1, s2, L1

24       sub s8, t1, s3

28       or  s9, t6, s5

2C       ...
...      ...

58   L1: add s7, s3, s4
```



Branch misprediction penalty:

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

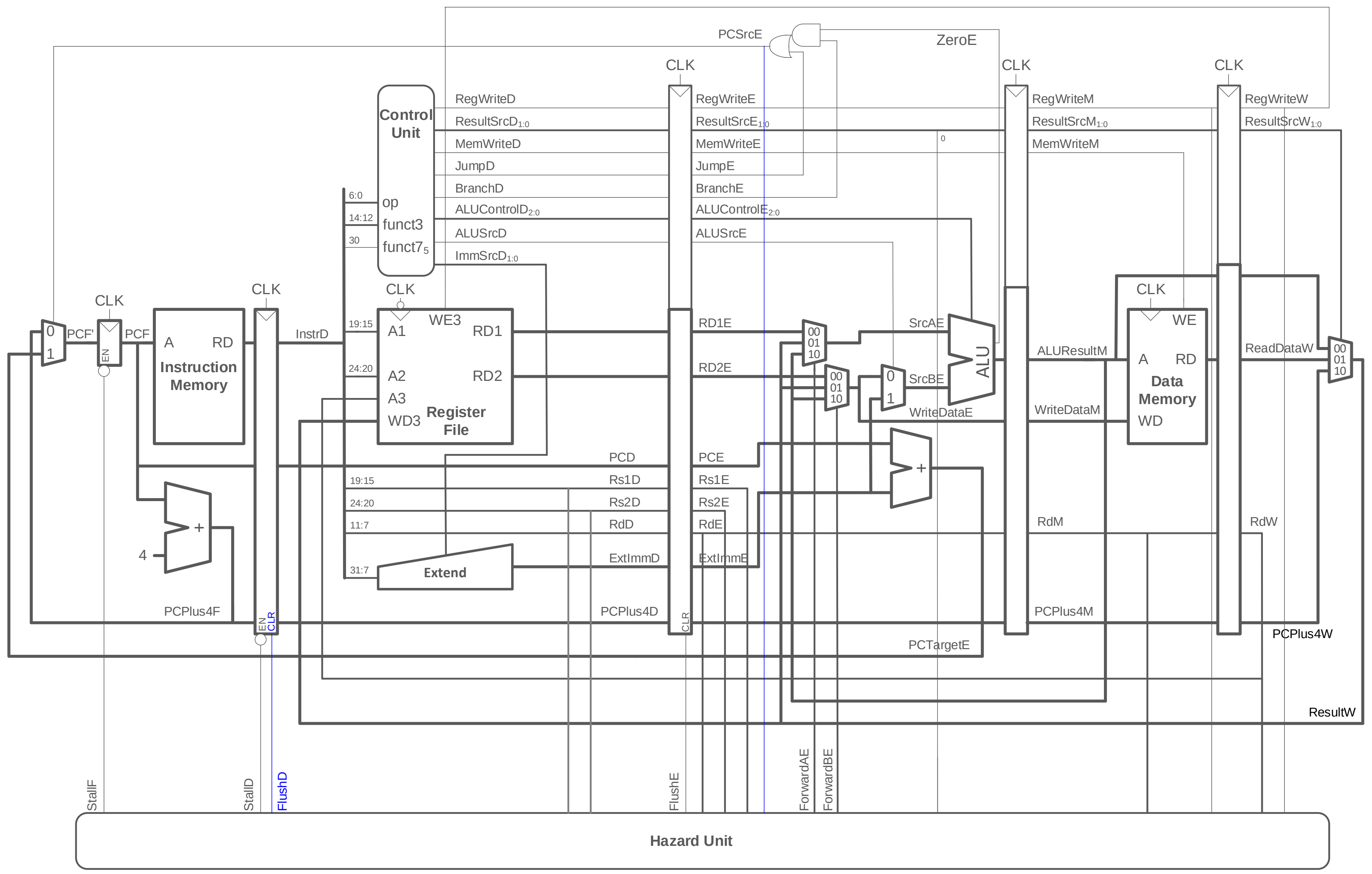
Control Hazards: Flushing Logic

- 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:**

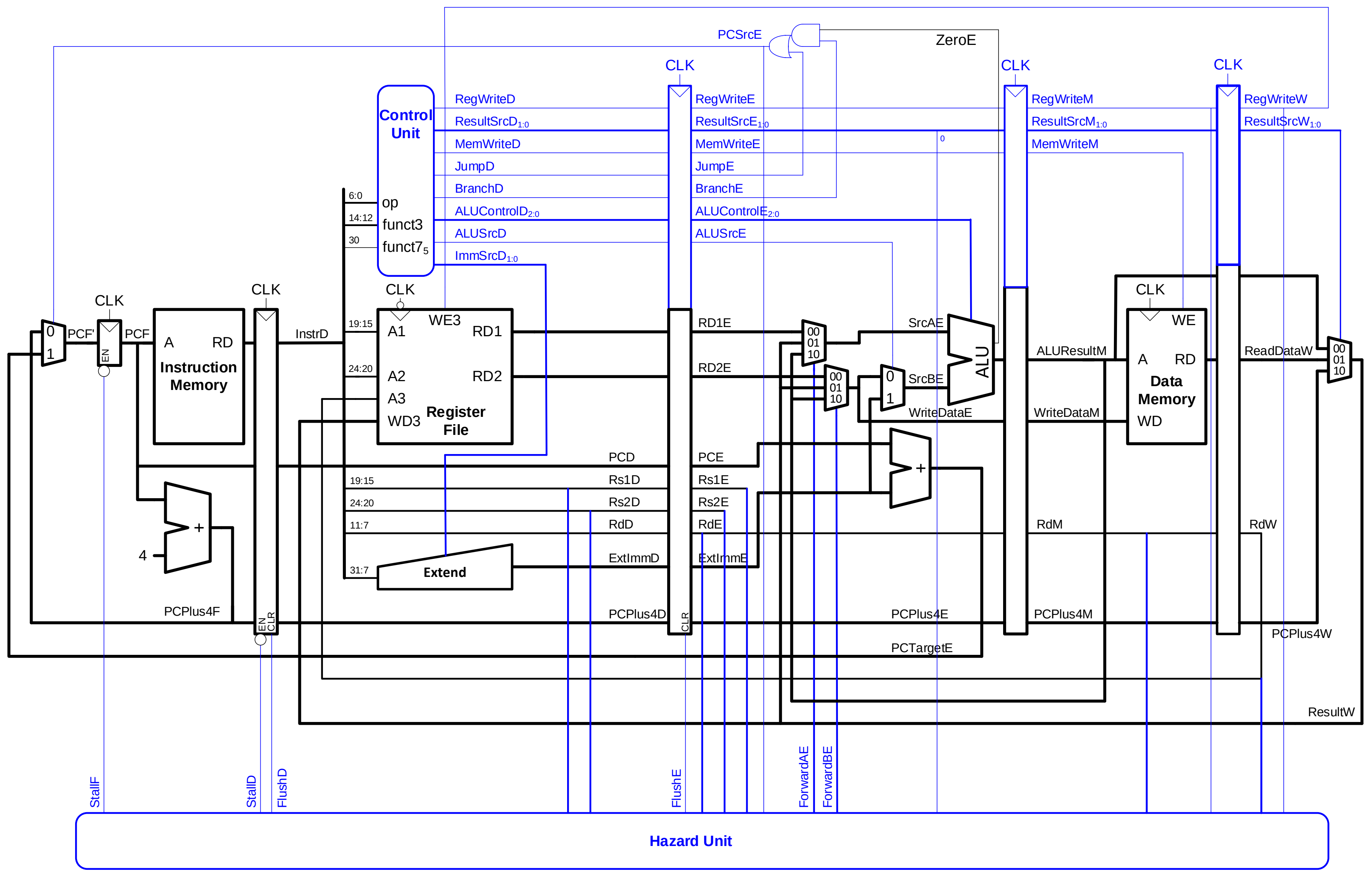
  FlushD = PCSrcE

  FlushE = lwStall OR *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*) OR (*Rs2D* == *RdE*)) AND $ResultSrcE_0$

StallF = ***StallD*** = *lwStall*

Control hazard flush:

FlushD = *PCSrcE*

FlushE = *lwStall* OR *PCSrcE*

Class Interaction # 21

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

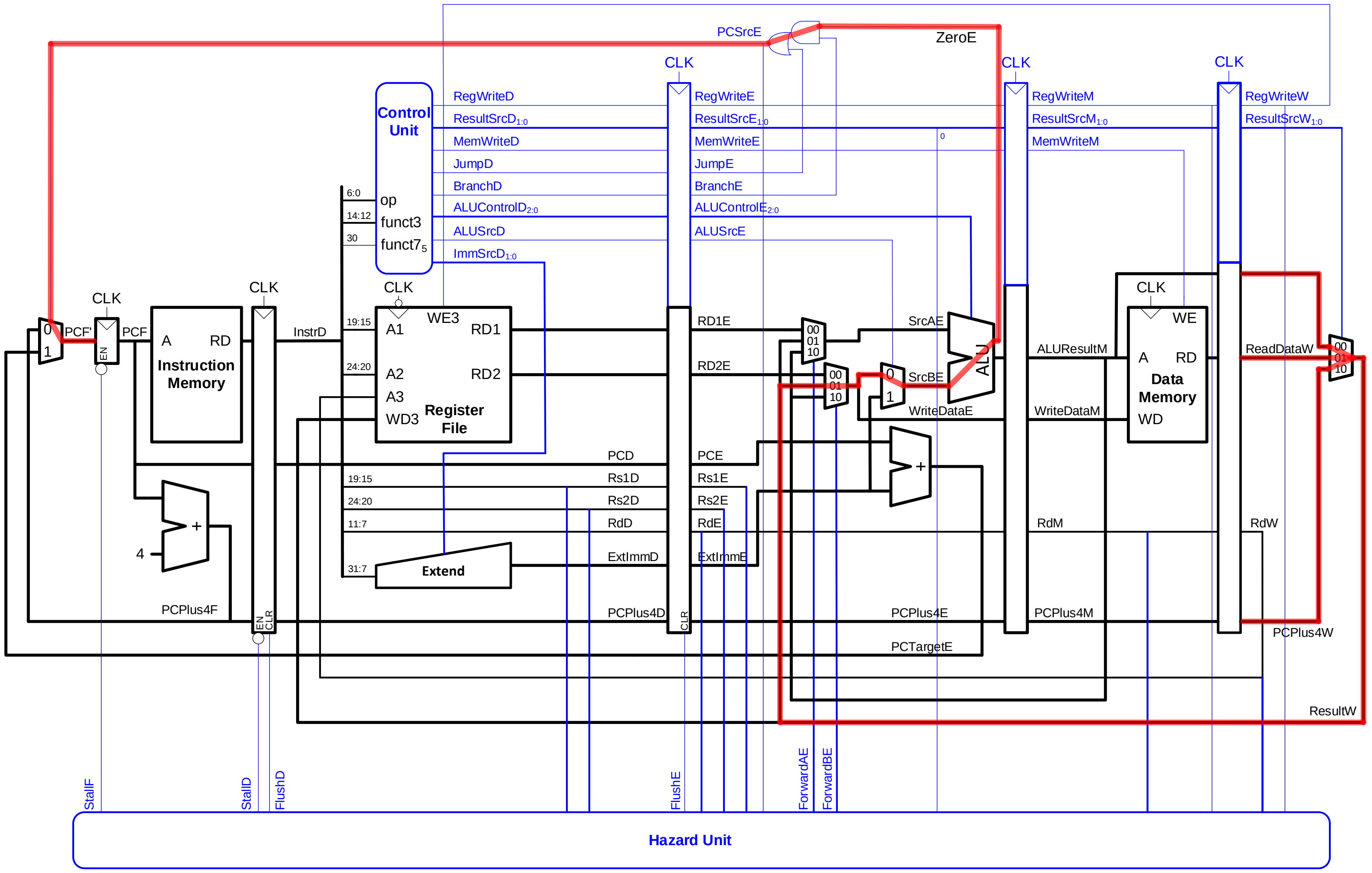
Pipelined processor critical path:

$T_{c\_pipelined}$ = max of

$t_{pcq} + t_{\text{mem}} + t_{\text{setup}}$	Fetch
$2(t_{\text{RFread}} + t_{\text{setup}})$	Decode
$t_{pcq} + 4t_{\text{mux}} + t_{\text{ALU}} + t_{\text{AND-OR}} + t_{\text{setup}}$	Execute
$t_{pcq} + t_{\text{mem}} + t_{\text{setup}}$	Memory
$2(t_{pcq} + t_{\text{mux}} + t_{\text{RFwrite}})$	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	$t_{\mathrm{AND\text{-}OR}}$	20
ALU	t_{ALU}	120
Decoder (Control Unit)	t_{dec}	25
Extend unit	t_{dec}	35
Memory read	t_{mem}	200
Register file read	$t_{RF\mathrm{read}}$	100
Register file setup	$t_{RF\mathrm{setup}}$	60

$$T_{c\_pipelined} = t_{pcq} + 4t_{\mathrm{mux}} + t_{\mathrm{ALU}} + t_{AND\text{-}OR} + t_{\mathrm{setup}}$$
$$=$$

Pipelined Performance Example

Program with 100 billion instructions

Execution Time $= (\#\text{ instructions}) \times \text{CPI} \times T_c$

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

= 43 seconds

Processor Performance Comparison

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

Class Interaction # 22