Pipelined Implementation (1)

Lecture 11
October 30th, 2018

Jae W. Lee (jaewlee@snu.ac.kr)
Computer Science and Engineering
Seoul National University

***Slide credits**: [CS:APP3e] slides from CMU; [COD5e] slides from Elsevier Inc.*

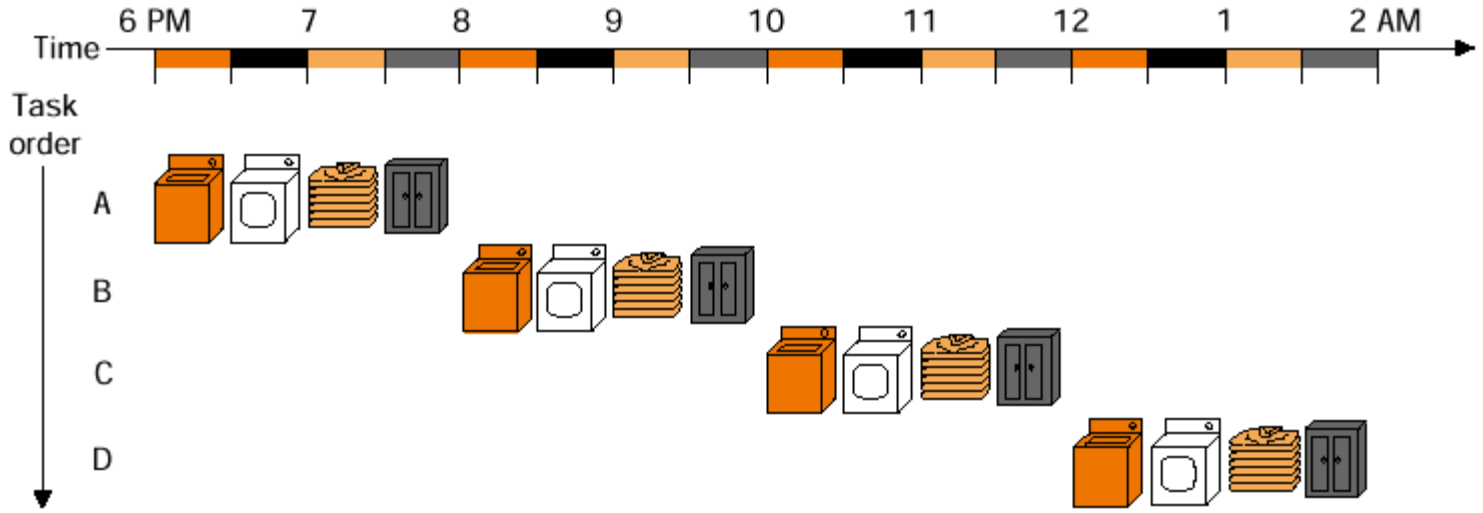
Today

Textbook: [CS:APP3e] 4.4

- **You Already Know Pipelining: Laundry Example**
- **Pipelining for Computation**
- **Pipelined Instruction Execution**
- **Major Hurdles of Pipelining: Hazards**

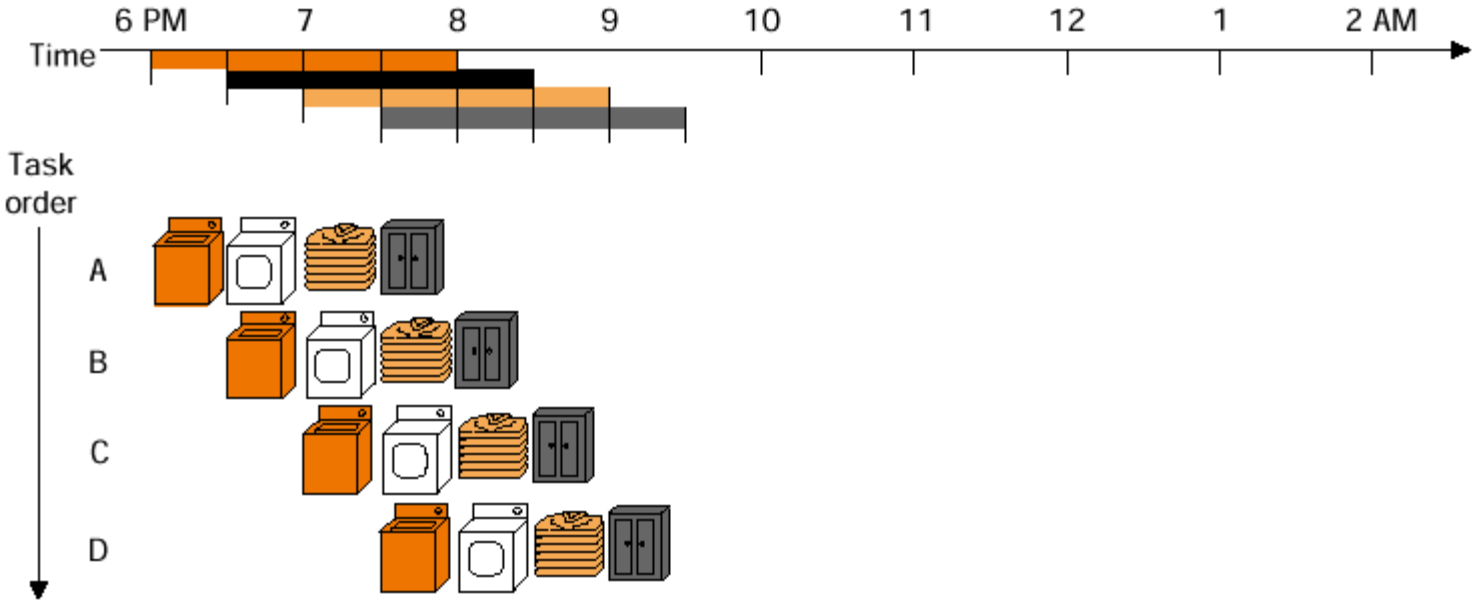
You Already Know Pipelining: Laundry Example

- **Sequential Processing: Wash-Dry-Fold-Store**

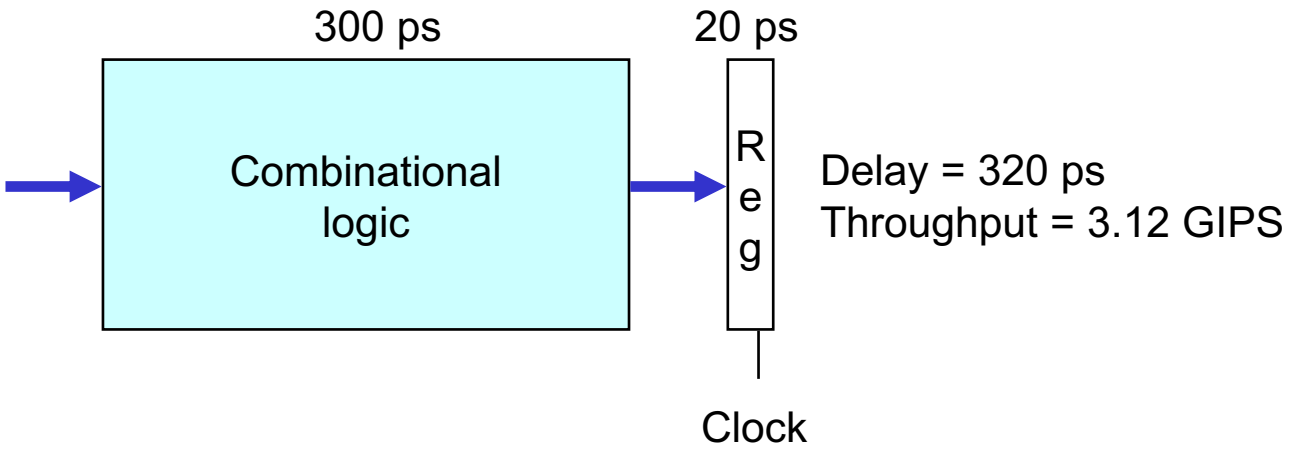


You Already Know Pipelining: Laundry Example

- **Pipelined Processing**

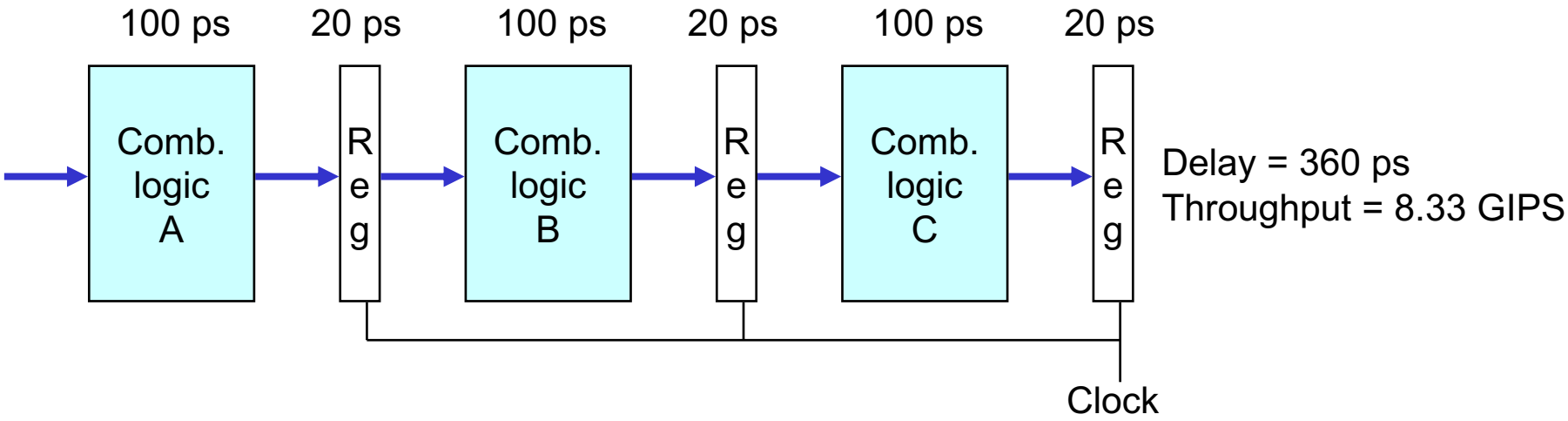


Pipelining for Computation



- **System**
  - Computation requires total of 300 picoseconds
  - Additional 20 picoseconds to save result in register
  - Must have clock cycle of at least 320 ps

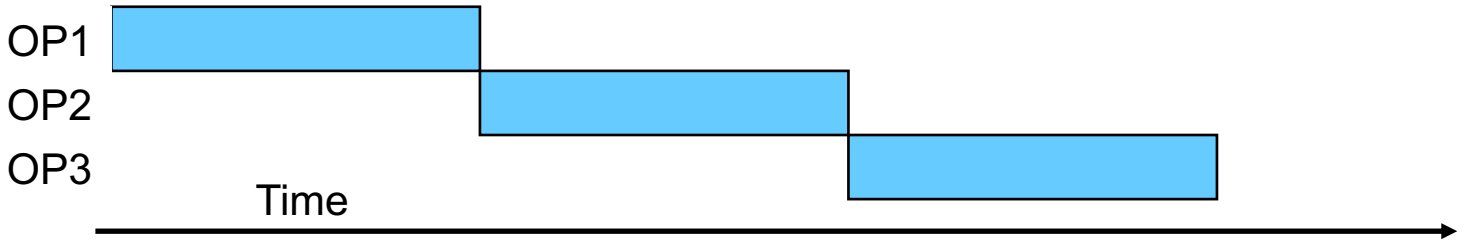
Pipelining for Computation



- **3-Way Pipelined Version**
  - Divide combinational logic into 3 blocks of 100 ps each
  - Can begin new operation as soon as previous one passes through stage A.
    - Begin new operation every 120 ps
  - Overall latency increases
    - 360 ps from start to finish

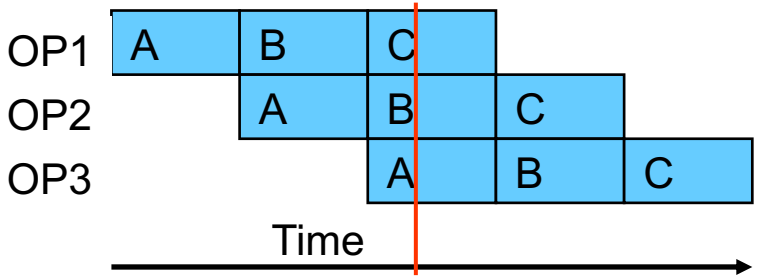
Pipelining for Computation: Pipeline Diagrams

- **Unpipelined**



  - Cannot start new operation until previous one completes

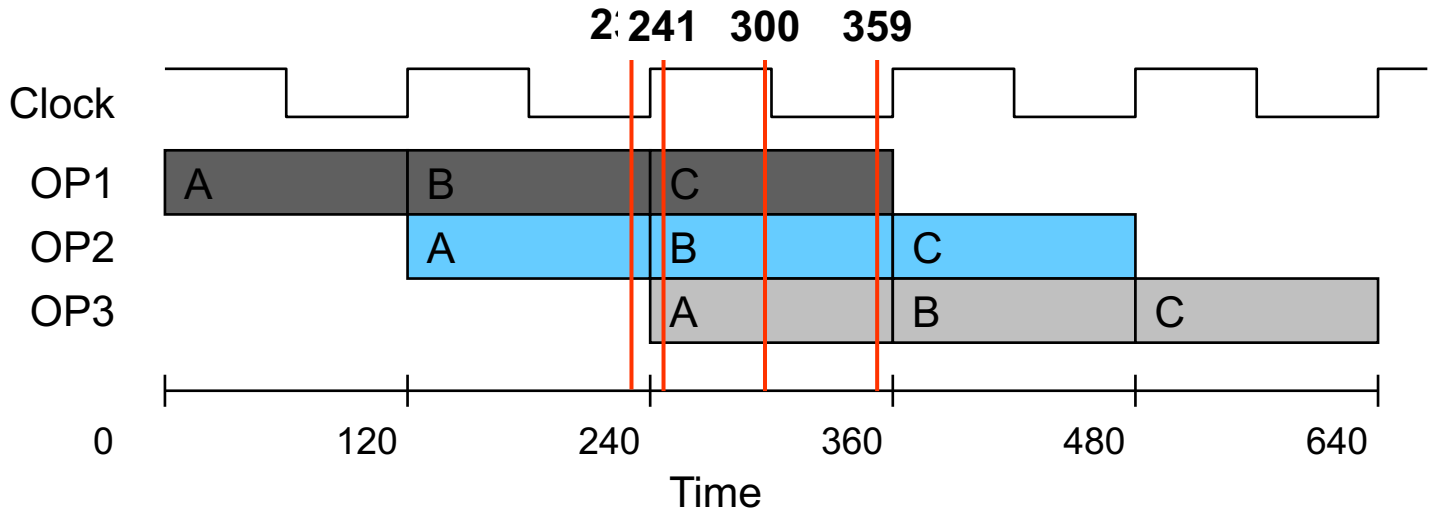
- **3-Way Pipelined**

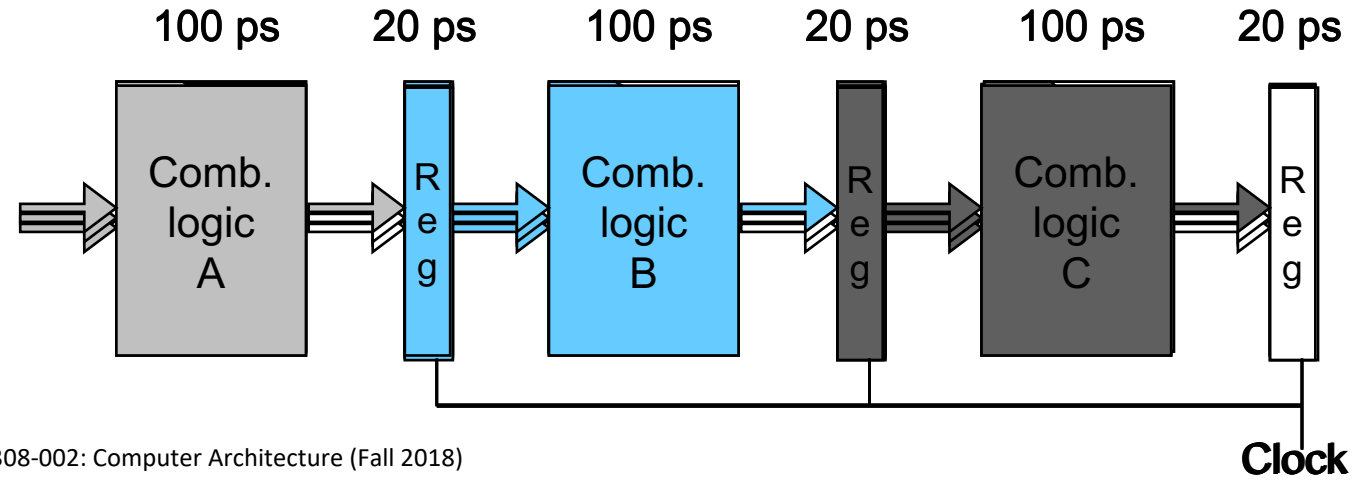


  - Up to 3 operations in process simultaneously

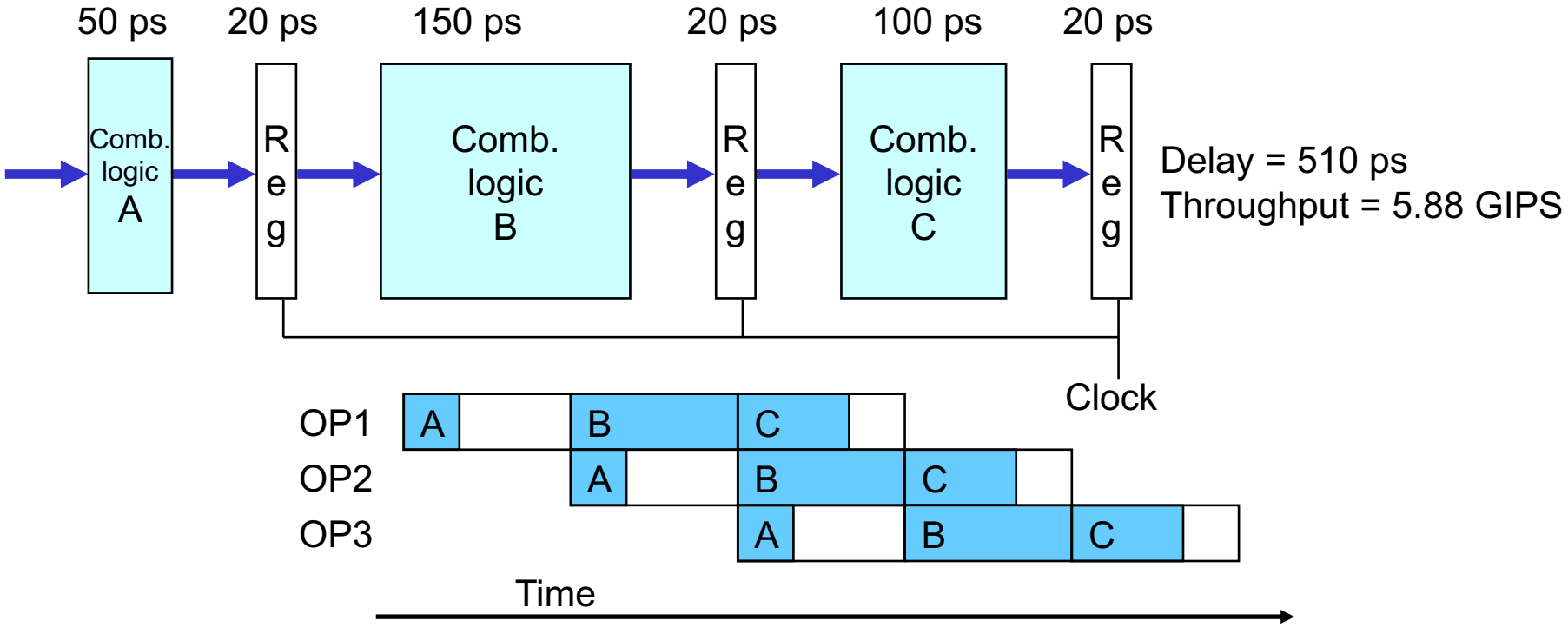
Pipelining for Computation

- **3-Way Pipelined Version: Operation**



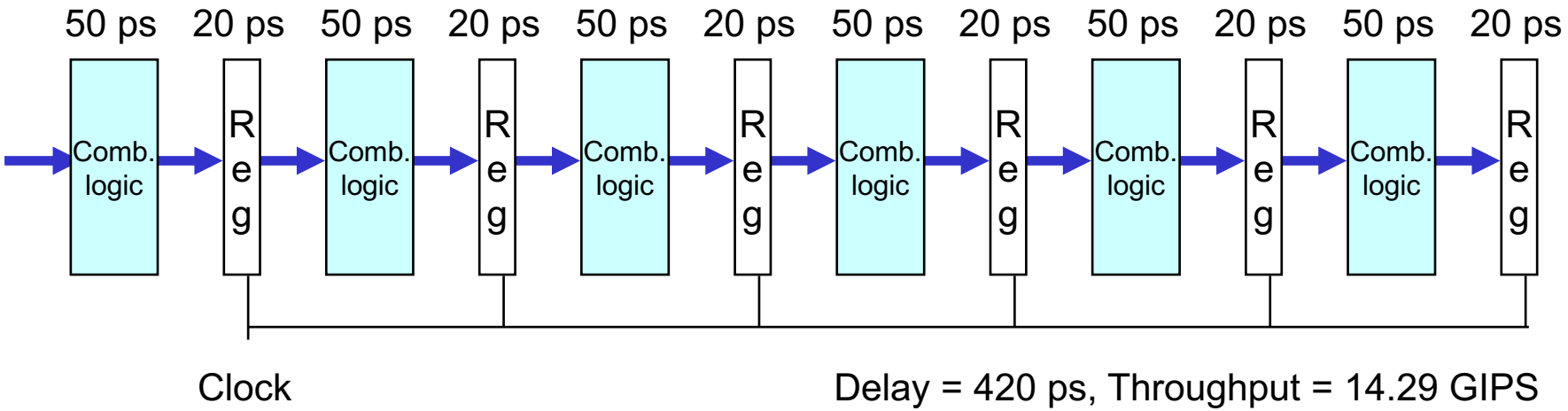


Limitations of Pipelining: Nonuniform Delays



- Throughput limited by slowest stage
- Other stages sit idle for much of the time
- Challenging to partition system into balanced stages

Limitations of Pipelining: Register Overhead



- As try to deepen pipeline, overhead of loading registers becomes more significant
- Percentage of clock cycle spent loading register:
  - 1-stage pipeline: 6.25%
  - 3-stage pipeline: 16.67%
  - 6-stage pipeline: 28.57%
- High speeds of modern processor designs obtained through very deep pipelining

Pipelined Instruction Execution

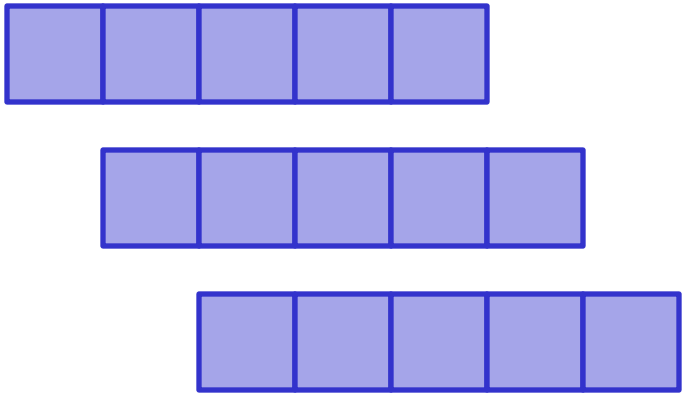
- **Basic steps of execution**

1. Instruction fetch step (F)
2. Instruction decode/register fetch step (D)
3. Execution/effective address step (E)
4. Memory access (M)
5. Register write-back step (W)

Pipelined Instruction Execution

- Sequential Execution

- Pipelined Execution



Pipelined Instruction Execution

- **Basic Pipeline**

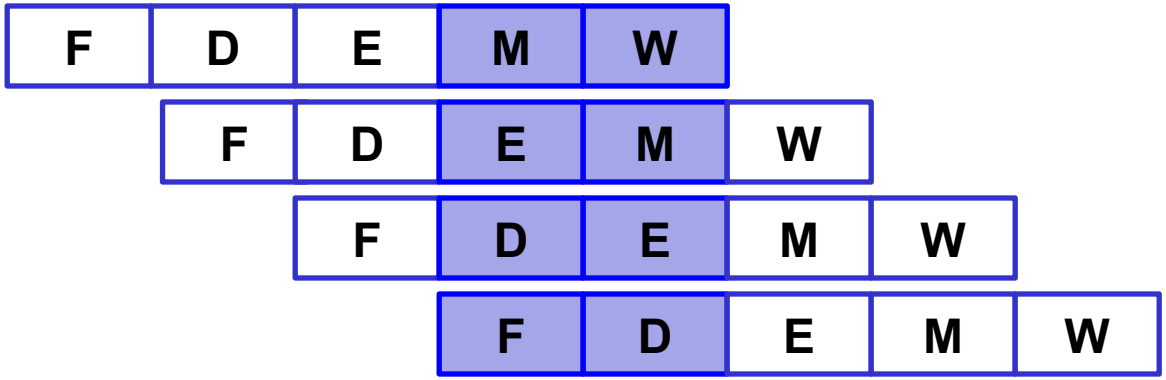
	Clock number								
Instruction number	1	2	3	4	5	6	7	8	9
Instruction *i*	F	D	E	M	W				
Instruction *i* + 1		F	D	E	M	W			
Instruction *i* + 2			F	D	E	M	W		
Instruction *i* + 3				F	D	E	M	W	
Instruction *i* + 4					F	D	E	M	W

Major Hurdles of Pipelining: Hazards

- Structural Hazard
- Data Hazard
- Control Hazard

Major Hurdles of Pipelining: Hazards

- **Structural Hazard**



	Clock number								
Instruction number	1	2	3	4	5	6	7	8	9
Load Instruction	F	D	E	M	W				
Instruction $i+1$		F	D	E	M	W			
Instruction $i+2$			F	D	E	M	W		
Instruction $i+3$				F	D	E	M	W	
Instruction $i+4$					F	D	E	M	W

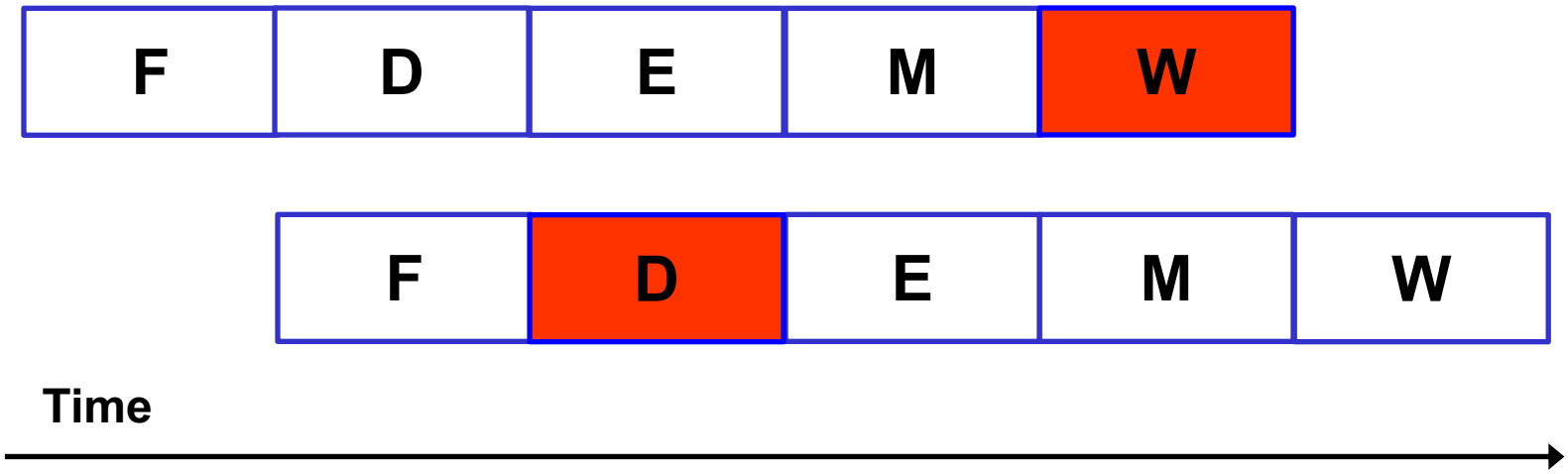
Major Hurdles of Pipelining: Hazards

- **Solutions to Structural Hazard: Resource Duplication**
  - example
    - Separate I and D caches for memory access conflict
    - Multi-port register file for register file access conflict

Major Hurdles of Pipelining: Hazards

- Data Hazard (RAW hazard)

```
addq %rdx, %rax
subq %rax, %rcx
```

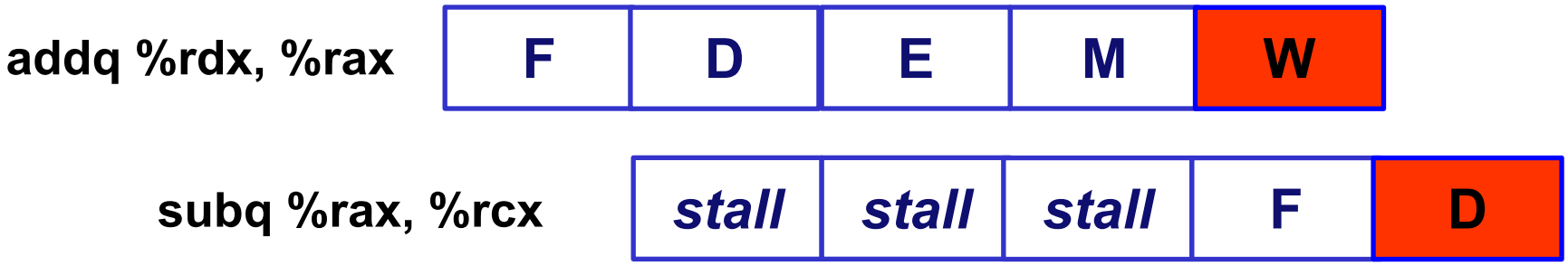


Major Hurdles of Pipelining: Hazards

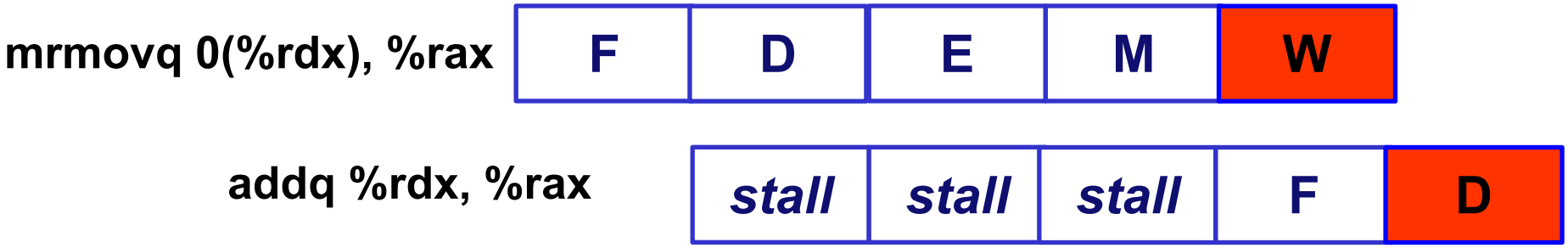
- **Solutions to Data Hazard**
  1. Freezing the pipeline
  2. (Internal) Forwarding
  3. Compiler scheduling

Major Hurdles of Pipelining: Hazards

- **Freezing The Pipeline**
  - ALU result to next instruction

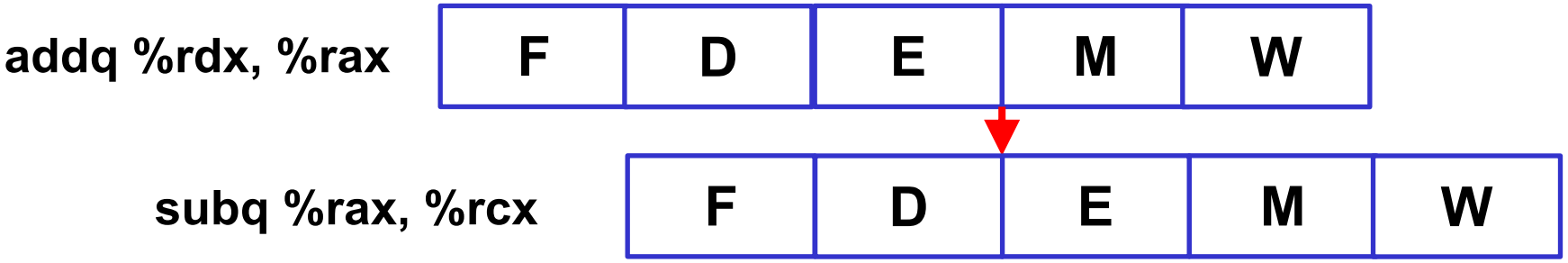


  - Load result to next instruction

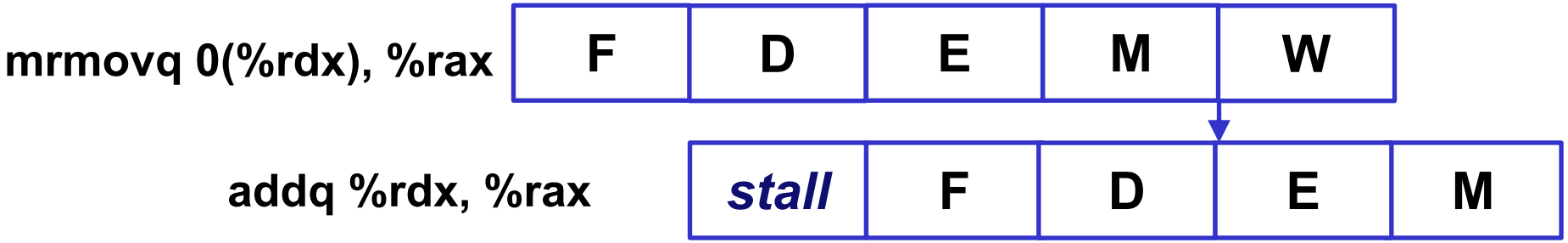


Major Hurdles of Pipelining: Hazards

- **(Internal) Forwarding**
  - ALU result to next instruction (Stall X)



  - Load result to next instruction (Stall 1)



Load interlock

Major Hurdles of Pipelining: Hazards

- **Control Hazards**
  - Caused by PC-changing instructions ((conditional/unconditional) Jump, Call/Return)

(Example)

Branch Instruction	F	D	E	M	W					
Branch successor		F	*stall*	*stall*	F	D	E	M	W	
Branch successor + 1						F	D	E	M	W
Branch successor + 2							F	D	E	M
Branch successor + 3								F	D	E
Branch successor + 4									F	D
Branch successor + 5										F

For 5-stage pipeline, 3 cycle penalty
15% branch frequency. CPI = 1.45

Major Hurdles of Pipelining: Hazards

- **Solution: Branch prediction!**
  - Example: Predict-taken

Taken branch instruction	F	D	E	M	W					
Branch target		F	D	E	M	W				
Branch target + 1			F	D	E	M	W			
Branch target + 2				F	D	E	M	W		
Branch target + 3					F	D	E	M	W	

Untaken branch instruction	F	D	E	M	W					
Branch target		F	D	E	*idle*	*idle*				
Branch target + 1			F	D	*idle*	*idle*	*idle*			
Branch target + 2				F	*idle*	*idle*	*idle*	*idle*		
Untaken branch instruction + 1					F	D	E	M	W	

~60% success rate