Serial versus Pipelined Execution

COE 301 / ICS 233

Computer Organization

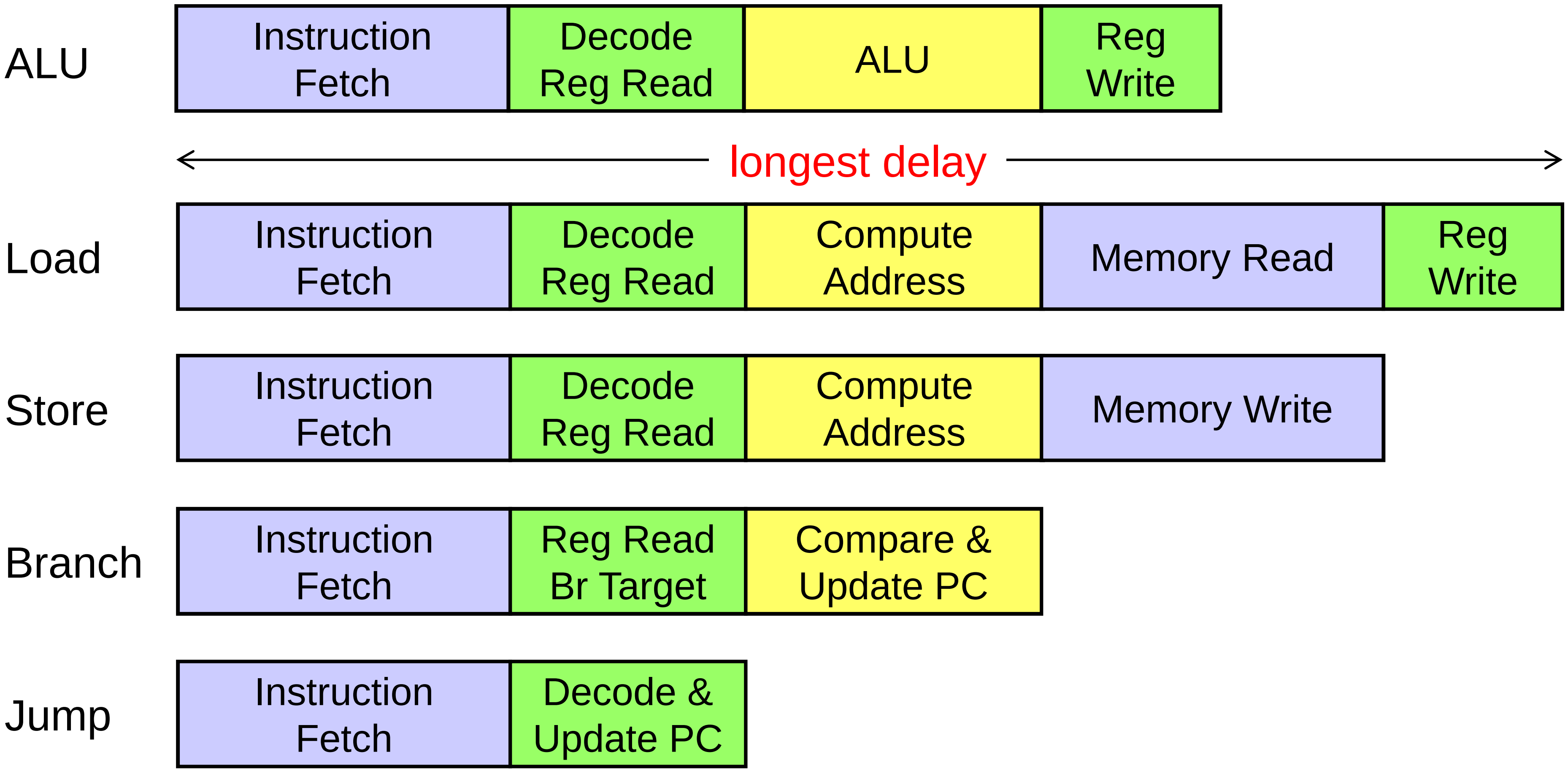
Dr. Muhamed Mudawar

College of Computer Sciences and Engineering

King Fahd University of Petroleum and Minerals

Drawback of Single Cycle Processor

- Drawback is the Long cycle time
  - All instructions take as much time as the slowest instruction



Alternative: Multicycle Implementation

- Break instruction execution into five steps
  - Instruction fetch
  - Instruction decode, register read, target address for jump/branch
  - Execution, memory address calculation, or branch outcome
  - Memory access or ALU instruction completion
  - Load instruction completion
- One clock cycle per step (clock cycle is reduced)
  - First 2 steps are the same for all instructions

Instruction	# cycles	Instruction	# cycles
ALU & Store	4	Branch	3
Load	5	Jump	2

Performance Example

- Assume the following operation times for components:
  - Access time for Instruction and data memories: 200 ps
  - Delay in ALU and adders: 180 ps
  - Delay in Decode and Register file access (read or write): 150 ps
  - Ignore the other delays in PC, mux, extender, and wires
- Which of the following would be faster and by how much?
  - Single-cycle implementation for all instructions
  - Multicycle implementation optimized for every class of instructions
- Assume the following instruction mix:
  - 40% ALU, 20% Loads, 10% stores, 20% branches, & 10% jumps

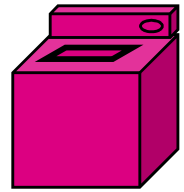
Solution

Instruction Class	Instruction Memory	Register Read	ALU Operation	Data Memory	Register Write	Total
ALU	200	150	180		150	680 ps
Load	200	150	180	200	150	880 ps
Store	200	150	180	200		730 ps
Branch	200	150	180 ←	Compare and update PC		530 ps
Jump	200	150 ←	Decode and update PC			350 ps

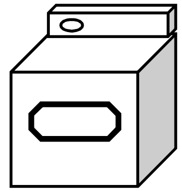
- For fixed single-cycle implementation:
  - Clock cycle = 880 ps determined by longest delay (load instruction)
- For multi-cycle implementation:
  - Clock cycle = max (200, 150, 180) = 200 ps (maximum delay at any step)
  - Average CPI = $0.4\times4 + 0.2\times5 + 0.1\times4 + 0.2\times3 + 0.1\times2 = 3.8$
- Speedup = 880 ps / (3.8 × 200 ps) = 880 / 760 = 1.16

Pipelining Example

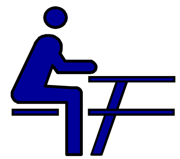
- Laundry Example: Three Stages



1. Wash dirty load of clothes

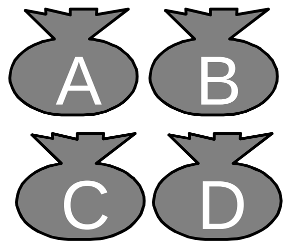


2. Dry wet clothes



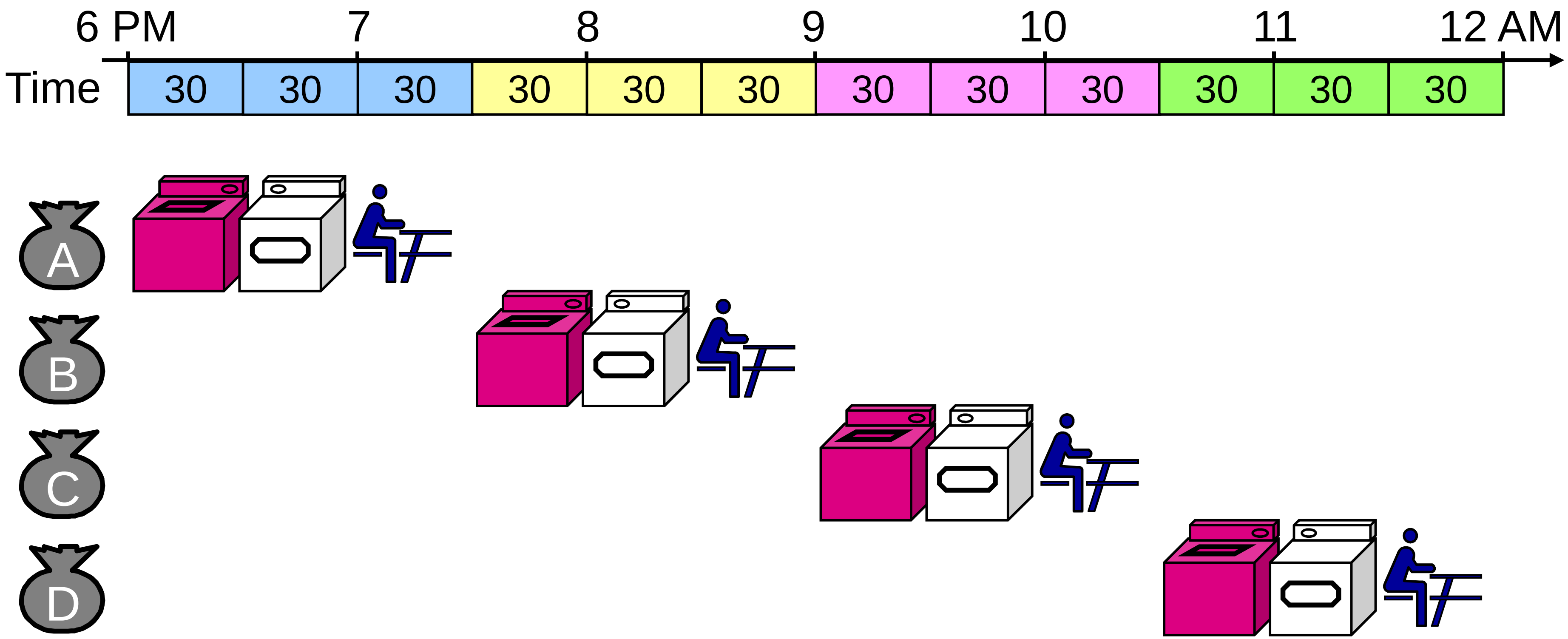
3. Fold and put clothes into drawers

- Each stage takes 30 minutes to complete



- Four loads of clothes to wash, dry, and fold

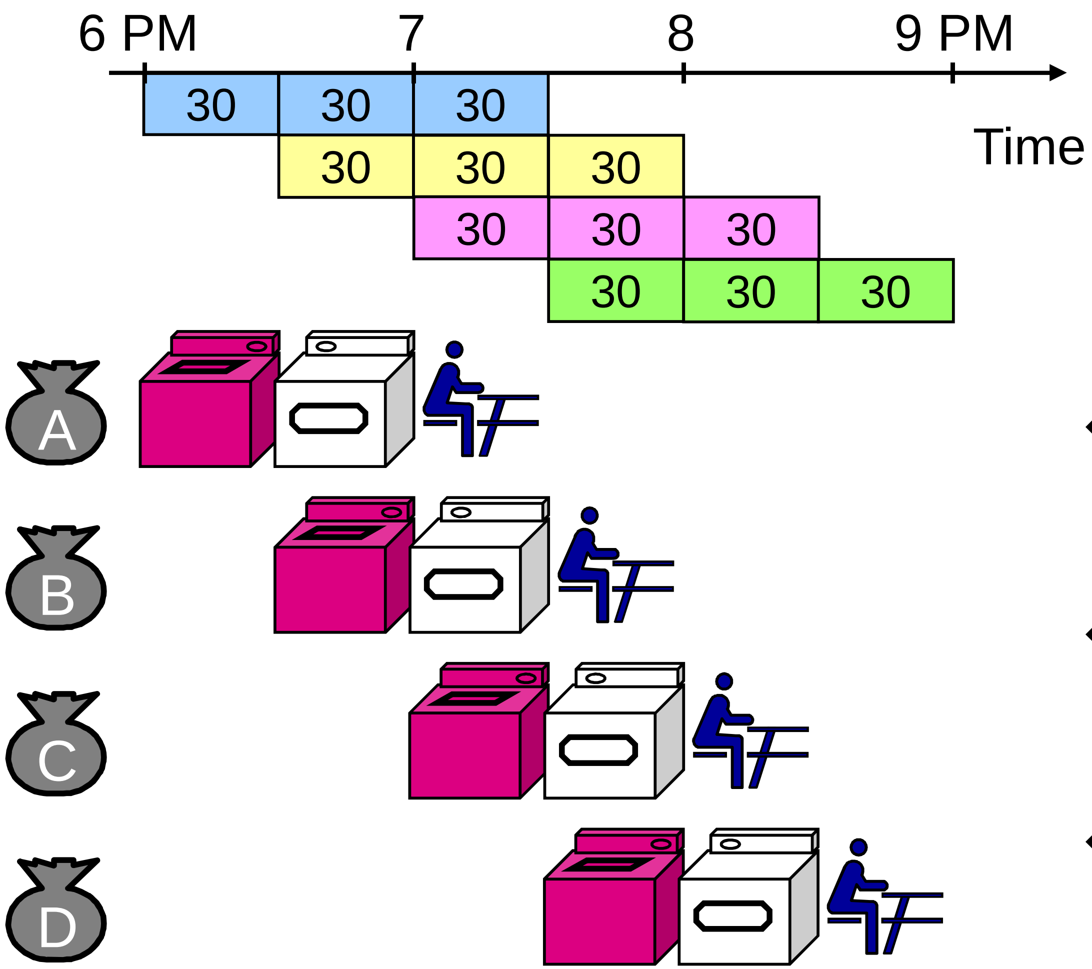
Sequential Laundry



❖ Sequential laundry takes 6 hours for 4 loads

❖ Intuitively, we can use pipelining to speed up laundry

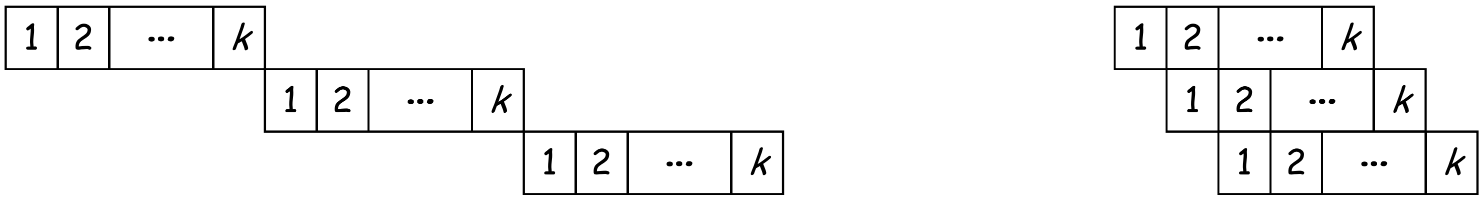
Pipelined Laundry: Start Load ASAP



- Pipelined laundry takes 3 hours for 4 loads
- Speedup factor is 2 for 4 loads
- Time to wash, dry, and fold one load is still the same (90 minutes)

Serial Execution versus Pipelining

- Consider a task that can be divided into *k* subtasks
  - The *k* subtasks are executed on *k* different stages
  - Each subtask requires one time unit
  - The total execution time of the task is *k* time units
- Pipelining is to overlap the execution
  - The *k* stages work in parallel on *k* different tasks
  - Tasks enter/leave pipeline at the rate of one task per time unit

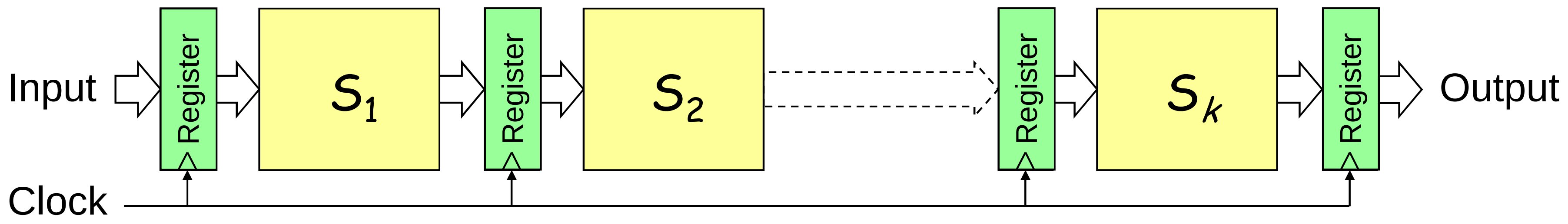


Without Pipelining
One completion every *k* time units

With Pipelining
One completion every 1 time unit

Synchronous Pipeline

- Uses clocked registers between stages
- Upon arrival of a clock edge …
  - All registers hold the results of previous stages simultaneously
- The pipeline stages are combinational logic circuits
- It is desirable to have balanced stages
  - Approximately equal delay in all stages
- Clock period is determined by the maximum stage delay



Pipeline Performance

- Let τ_i = time delay in stage S_i
- Clock cycle $\tau = \max(\tau_i)$ is the maximum stage delay
- Clock frequency $f = 1/\tau = 1/\max(\tau_i)$
- A pipeline can process n tasks in $k + n - 1$ cycles
  - k cycles are needed to complete the first task
  - $n - 1$ cycles are needed to complete the remaining $n - 1$ tasks
- Ideal speedup of a k-stage pipeline over serial execution

$$S_k = \frac{\text{Serial execution in cycles}}{\text{Pipelined execution in cycles}} = \frac{nk}{k + n - 1}$$

$S_k \rightarrow k$ for large n

MIPS Processor Pipeline

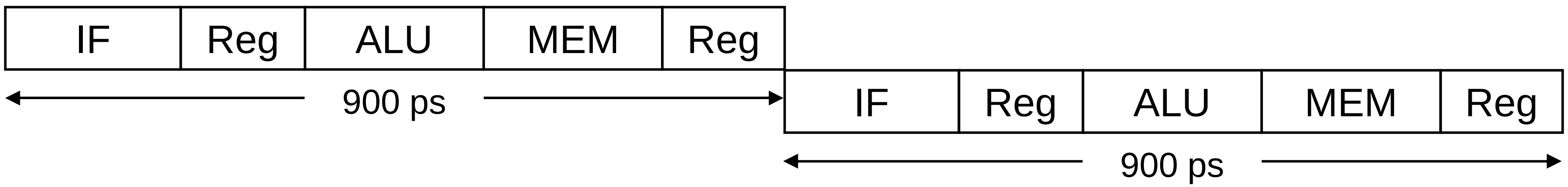
- Five stages, one cycle per stage

1. IF: Instruction Fetch from instruction memory
2. ID: Instruction Decode, register read, and J/Br address
3. EX: Execute operation or calculate load/store address
4. MEM: Memory access for load and store
5. WB: Write Back result to register

Single-Cycle vs Pipelined Performance

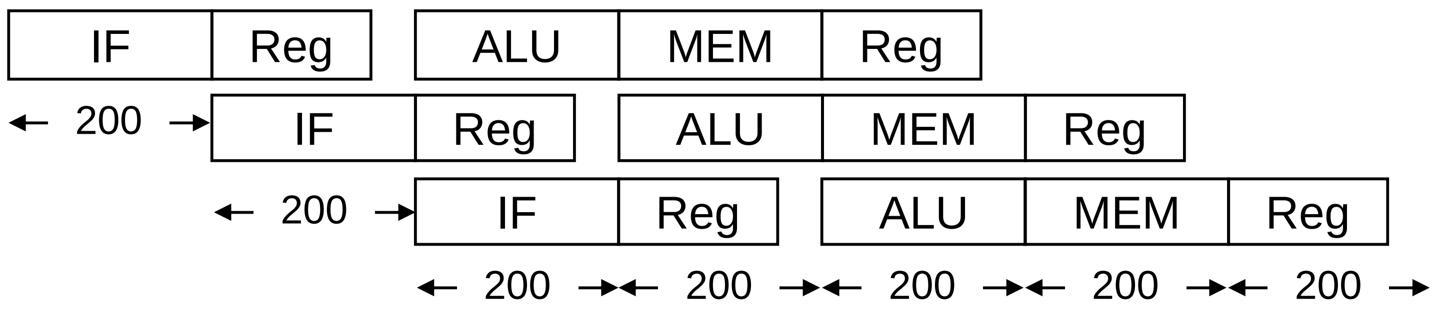
- Consider a 5-stage instruction execution in which …
  - Instruction fetch = ALU operation = Data memory access = 200 ps
  - Register read = register write = 150 ps
- What is the clock cycle of the single-cycle processor?
- What is the clock cycle of the pipelined processor?
- What is the speedup factor of pipelined execution?
- Solution

Single-Cycle Clock = 200+150+200+200+150 = 900 ps



Single-Cycle versus Pipelined – cont'd

- Pipelined clock cycle = max(200, 150) = 200 ps



- CPI for pipelined execution = 1
  - One instruction completes each cycle (ignoring pipeline fill)
- Speedup of pipelined execution = 900 ps / 200 ps = 4.5
  - Instruction count and CPI are equal in both cases
- Speedup factor is less than 5 (number of pipeline stage)
  - Because the pipeline stages are not balanced

Pipeline Performance Summary

- Pipelining doesn't improve latency of a single instruction
- However, it improves throughput of entire workload
  - Instructions are initiated and completed at a higher rate
- In a *k*-stage pipeline, *k* instructions operate in parallel
  - Overlapped execution using multiple hardware resources
  - Potential speedup = number of pipeline stages *k*
- Pipeline rate is limited by slowest pipeline stage
- Unbalanced lengths of pipeline stages reduces speedup
- Also, time to fill and drain pipeline reduces speedup