Parallel Computers

An Overview of Parallel Processing

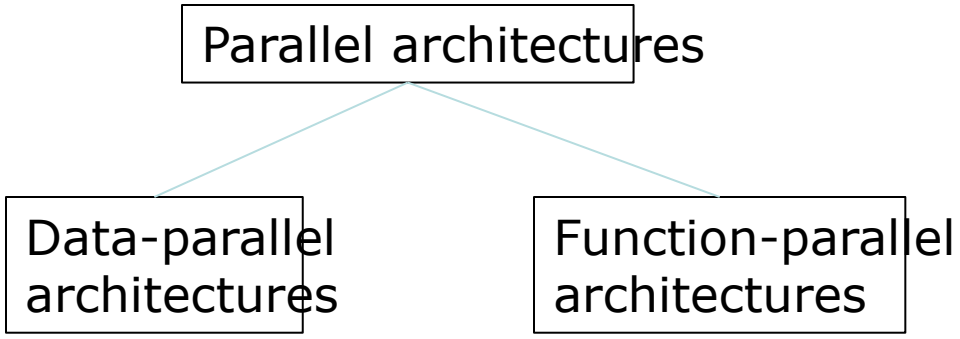
- What is parallel processing?
  - Parallel processing is a method to improve computer system performance by executing two or more instructions simultaneously.
- The goals of parallel processing.
  - One goal is to reduce the “wall-clock” time or the amount of real time that you need to wait for a problem to be solved.
  - Another goal is to solve bigger problems that might not fit in the limited memory of a single CPU.

Parallelism in Uniprocessor Systems

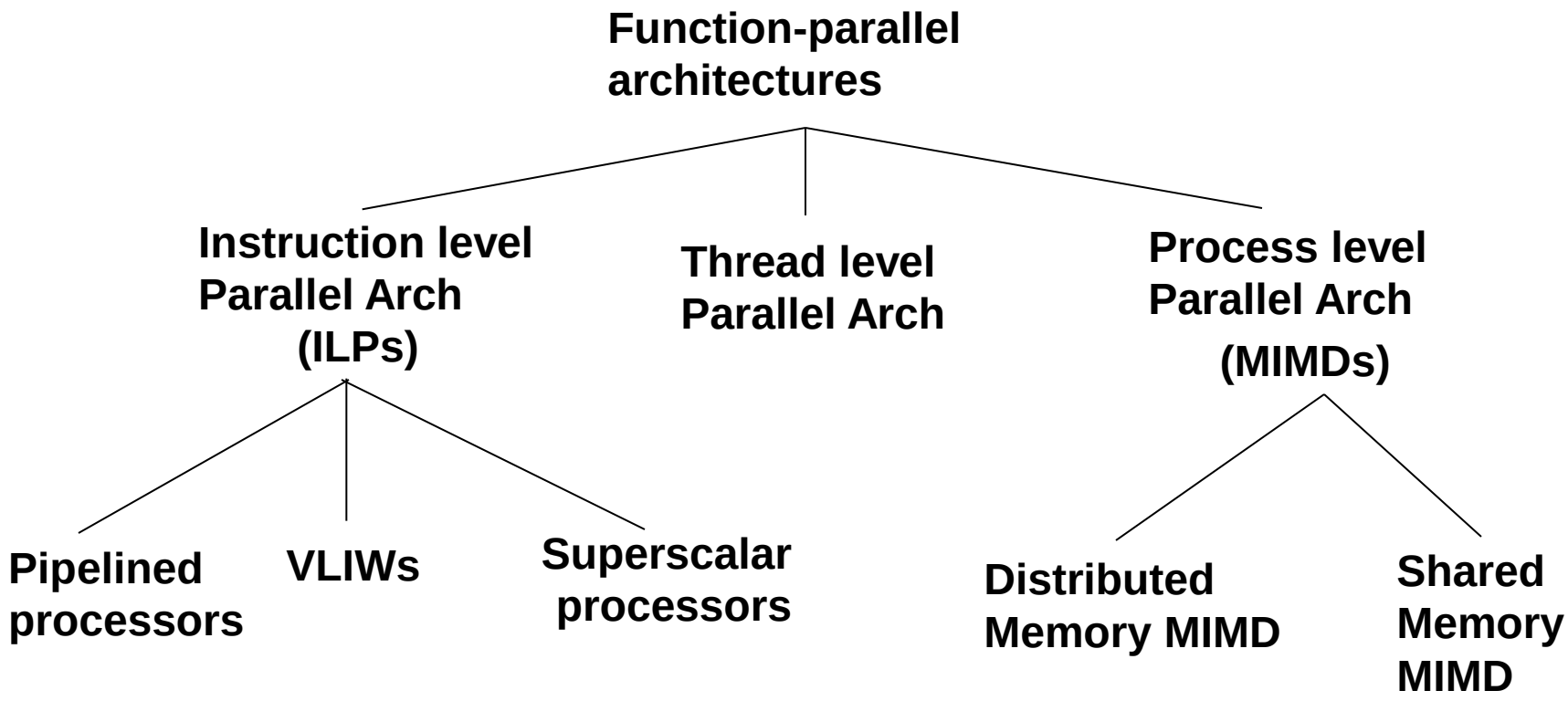
- It is possible to achieve parallelism with a uniprocessor system.
  - Some examples are the instruction pipeline, arithmetic pipeline, I/O processor.
- Note that a system that performs different operations on the same instruction is not considered parallel.
- Only if the system processes two different instructions simultaneously can it be considered parallel.

Modern classification (Sima, Fountain, Kacsuk)

- Classify based on how parallelism is achieved
  - by operating on multiple data: data parallelism
  - by performing many functions in parallel: function parallelism
    - Control parallelism, task parallelism depending on the level of the functional parallelism.



Control parallel architectures



ILP (Instruction level parallelism)

- Processors leverage parallel execution to make programs run faster, but it is invisible to programmer
- Instruction level parallelism (ILP)
  - Instructions appear to be executed in program order (*sequential semantics*).
  - BUT independent instructions can be executed in parallel by a processor without changing program correctness
  - Dynamic scheduling: processor logic dynamically finds independent instructions in an instruction sequence and executes them in parallel

- Instruction level parallelism
  - Different instructions within a stream executed in parallel
  - Pipelining, speculative execution, VLIW

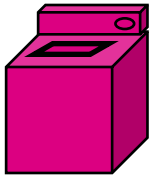
Pipelining

Executing Instruction in Partially Overlapped manner.

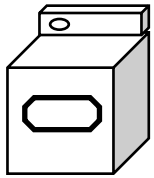
Pipelining Example

- Laundry Example: Three Stages

1. Wash dirty load of clothes

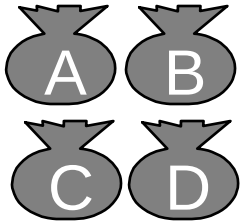


2. Dry wet clothes



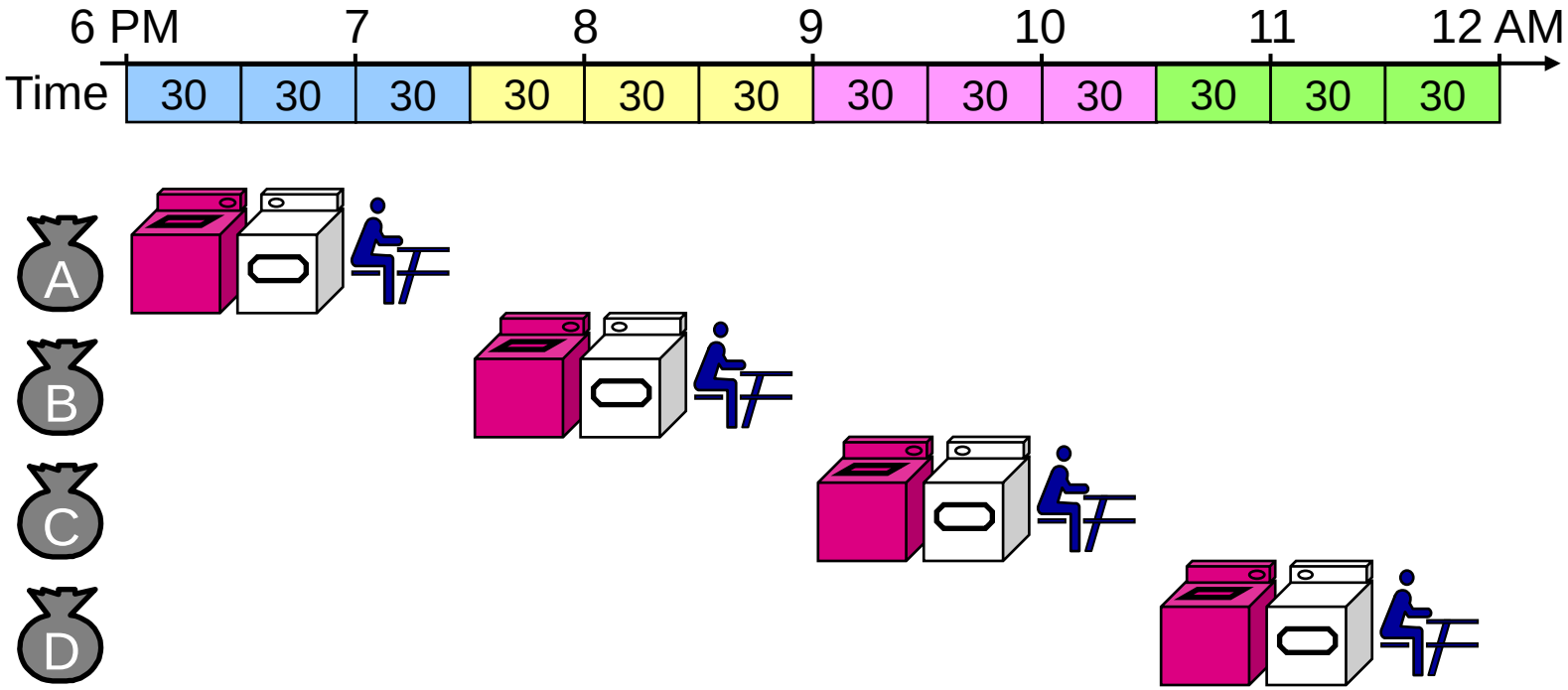
3. Fold and put clothes into drawers

- Each stage takes 30 minutes to complete



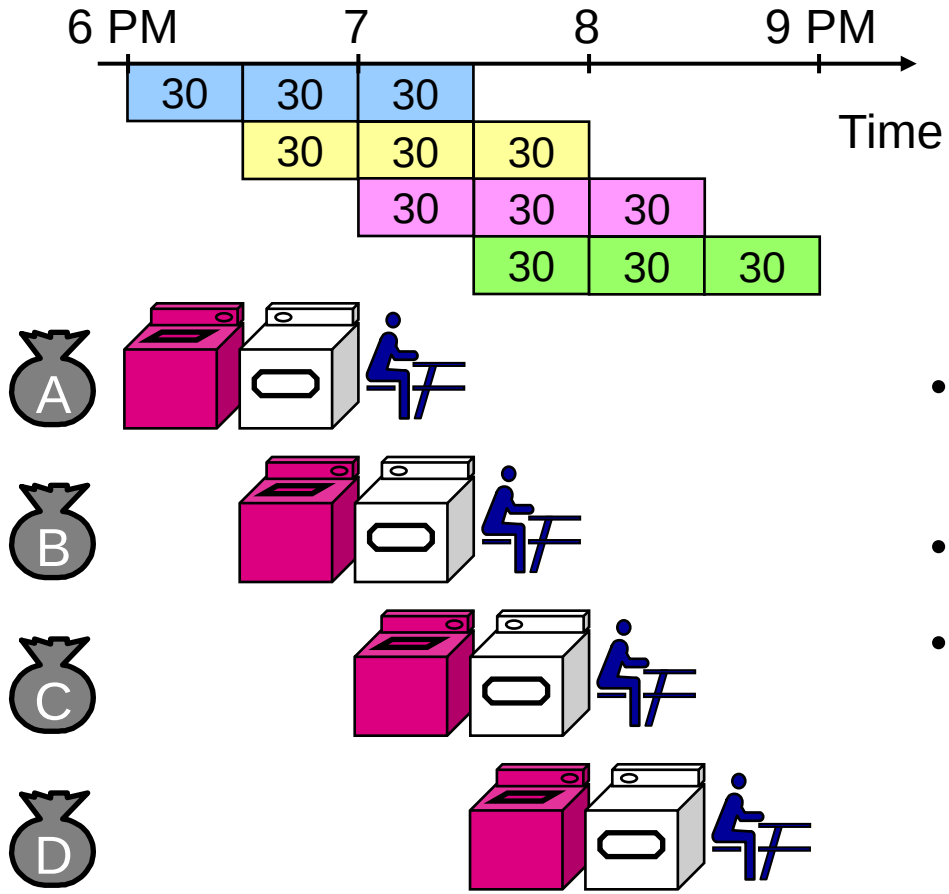
- Four loads of clothes to wash, dry, and fold

Sequential Laundry



- Sequential laundry takes 6 hours for 4 loads
- Intuitively, we can use pipelining to speed up laundry

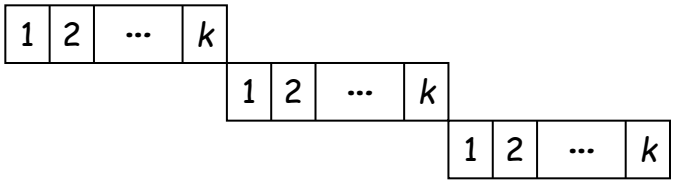
Pipelined Laundry: Start Load ASAP



- 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)

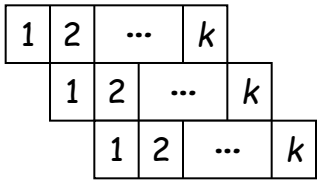
Serial Execution versus Pipelining

- 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

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

Pipelining

Latency:

It's the amount of time between when the instruction is issued and when it completes.

Throughput:

The number of instructions that complete in a span of time.

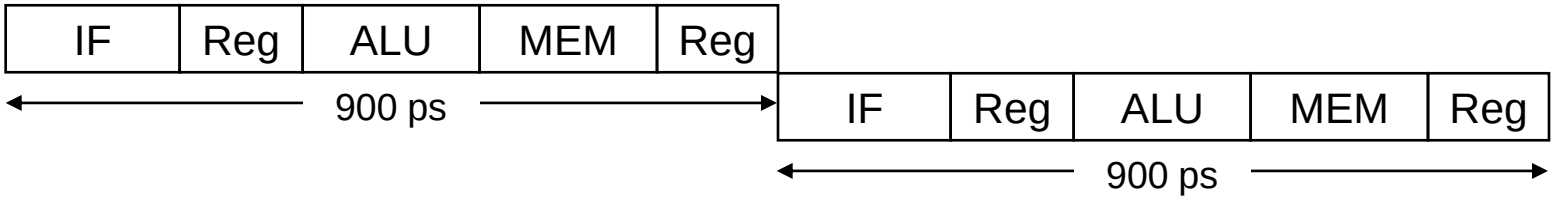
Cycle time:

time required to complete a single pipeline stage.

Single-Cycle vs Pipelined Performance

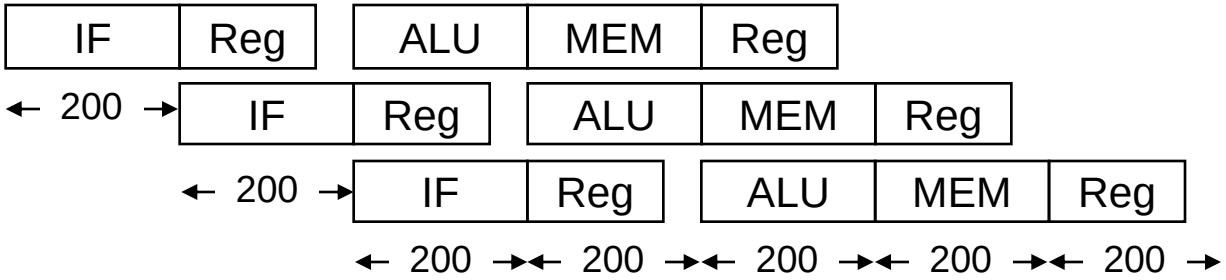
- Consider a 5-stage instruction execution in which …
  - Instruction fetch = ALU operation = Data memory access = 200 ps
  - Register read = register write = 150 ps
- What is the clock cycle of the single-cycle processor?
- What is the clock cycle of the pipelined processor?
- What is the speedup factor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

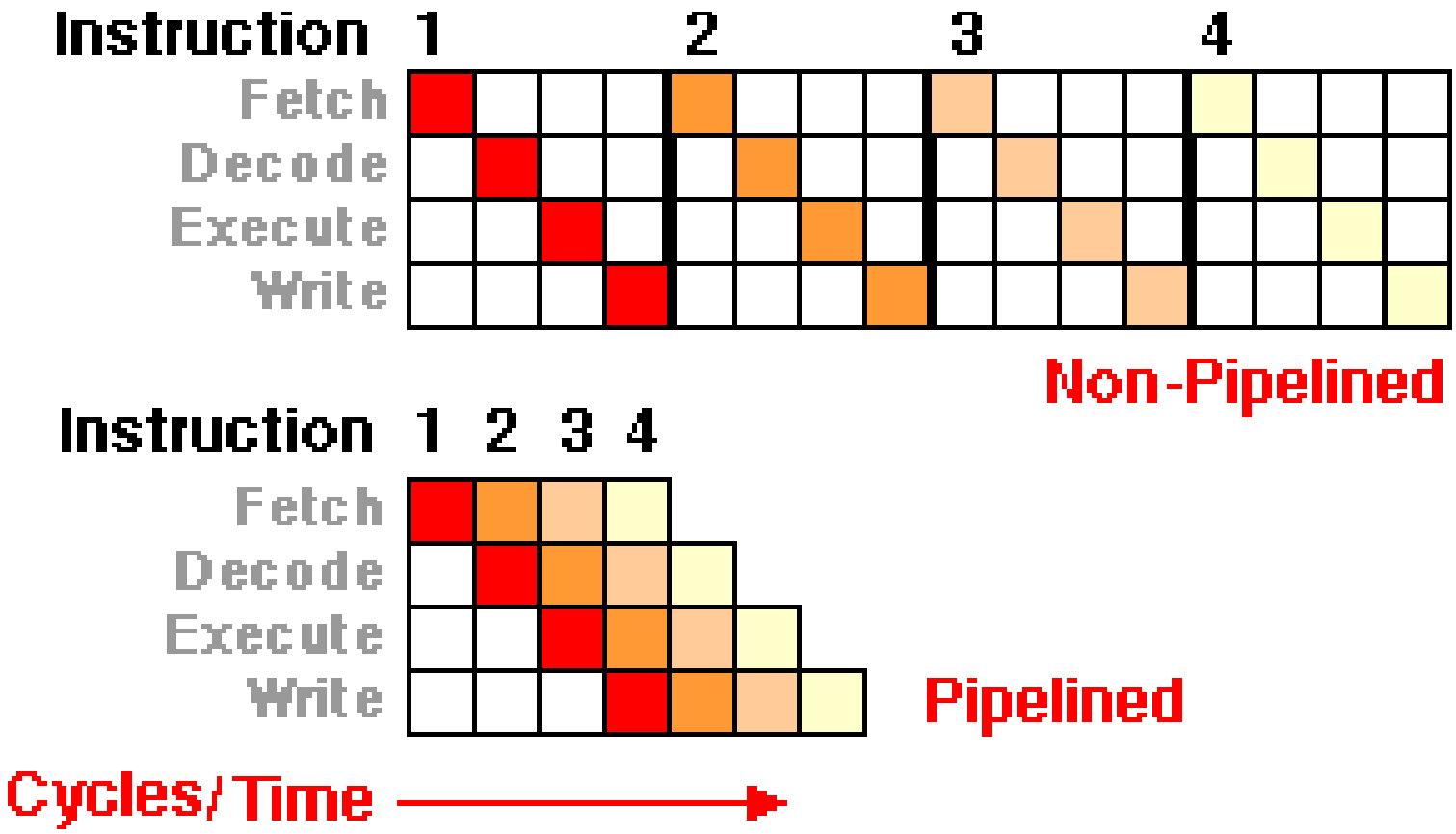


- CPI for pipelined execution = 1
  - One instruction completes each cycle (ignoring pipeline fill)
- Speedup of pipelined execution = 900 ps / 200 ps = 4.5
  - Instruction count and CPI are equal in both cases
- Speedup factor is less than 5 (number of pipeline stage)
  - Because the pipeline stages 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

Simple Pipelining



Simple Pipelining

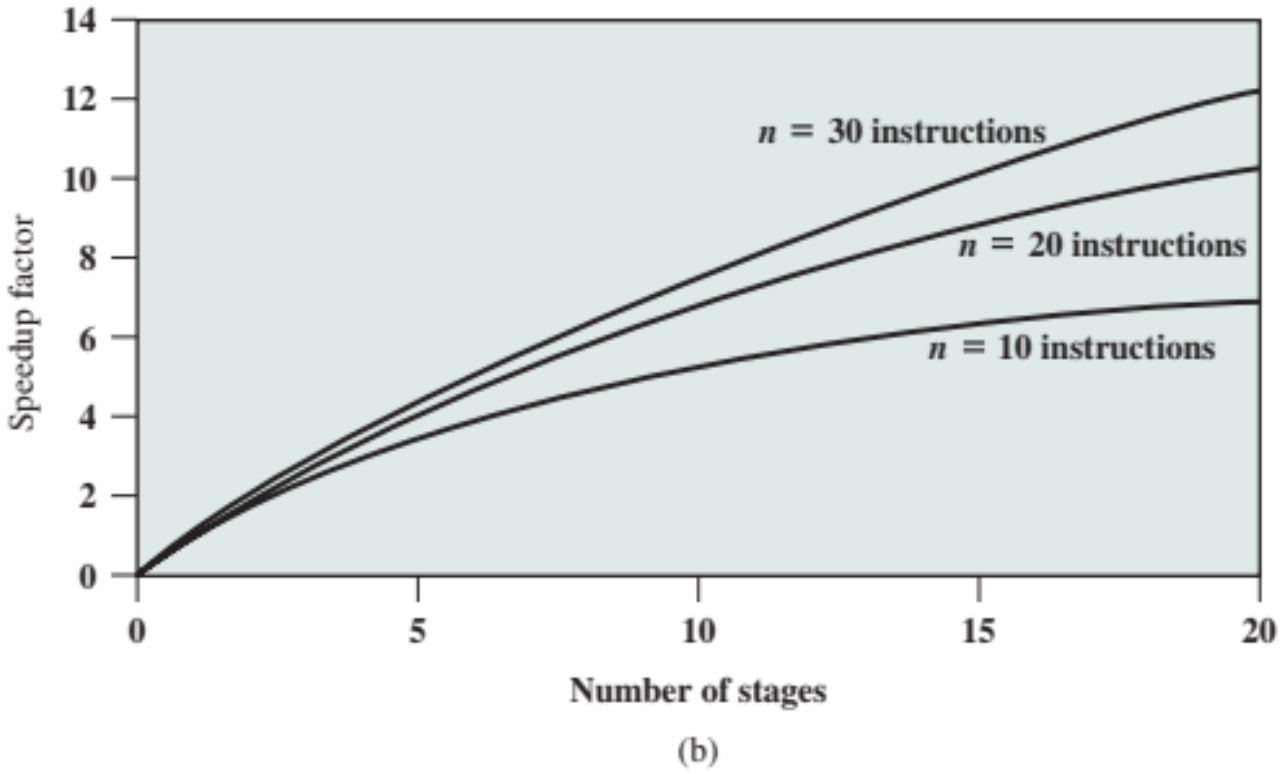


Figure 14.14 Speedup Factors with Instruction Pipelining

n - no. of instructions without branch.

Parts of different instructions execute in parallel in various units

- **Pipeline speedup ∝ pipeline stages**
- **Pipeline stalls reduce performance**

Pipeline Performance

Pipeline Performance

- The previous pipeline is said to have been stalled for two clock cycles.
- Stall is a bubble in pipeline
- Any condition that causes a pipeline to stall is called a hazard.
- Data hazard – An instruction stands to read or write the wrong data.
- Instruction (control) hazard – Instructions are fetched from the wrong path of the branch.
- Structural hazard – the situation when two instructions require the use of a given hardware resource at the same time.

Pipeline Performance

- Pipeline stall causes degradation in pipeline performance.
- We need to identify all hazards that may cause the pipeline to stall and to find ways to minimize their impact.

Pipelining

The 5 stages of the processor have the following latencies:

	Fetch	Decode	Execute	Memory	Writeback
a.	300ps	400ps	350ps	550ps	100ps
b.	200ps	150ps	100ps	190ps	140ps

Assume that when pipelining, each pipeline stage costs 20ps extra for the registers between pipeline stages.

Non-pipelined processor: what is the cycle time? What is the latency of an instruction? What is the throughput?

Because there is no pipelining, the cycle time must allow an instruction to go through all stages in one cycle. The latency is the same as cycle time since it takes the instruction one cycle to go from the beginning of fetch to the end of writeback. The throughput is defined as 1/CT inst/s.

a. CT = 300 + 400 + 350 + 550 + 100 = 1700ps
   Latency = 1700ps
   Throughput = 1/1700 inst/ps

Pipelining

Pipelined processor: What is the cycle time? What is the latency of an instruction? What is the throughput?

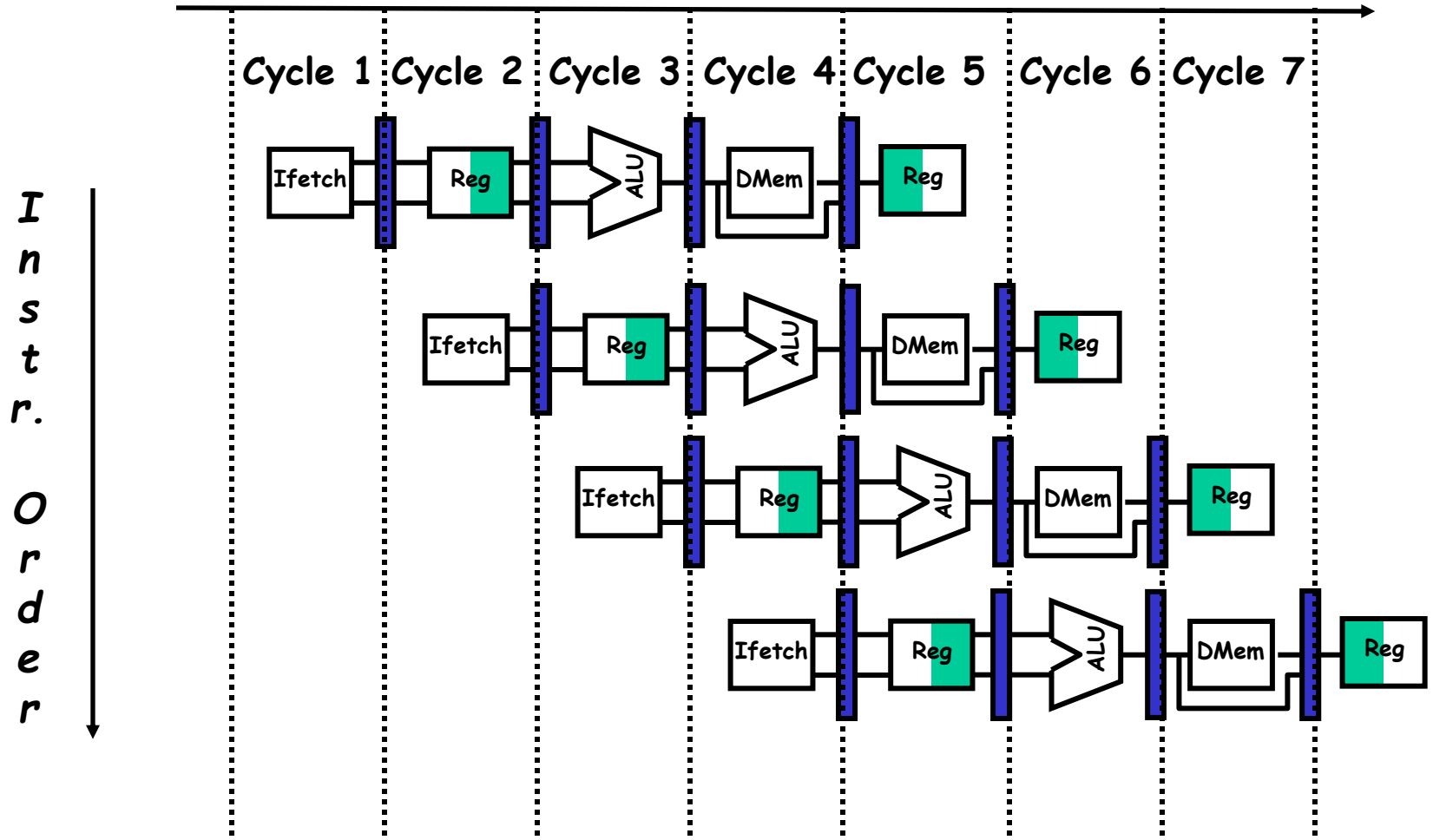
Pipelining reduces the cycle time to the length of the longest stage plus the register delay. Latency becomes CT*N where N is the number of stages as one instruction will need to go through each of the stages and each stage takes one cycle. The throughput formula remains the same.

a. CT = 550 + 20 = 570 ps
   Latency = 5 * 570 = 2850ps
   Throughput = 1/570 inst/ps

Use the Idea of Pipelining in a Computer

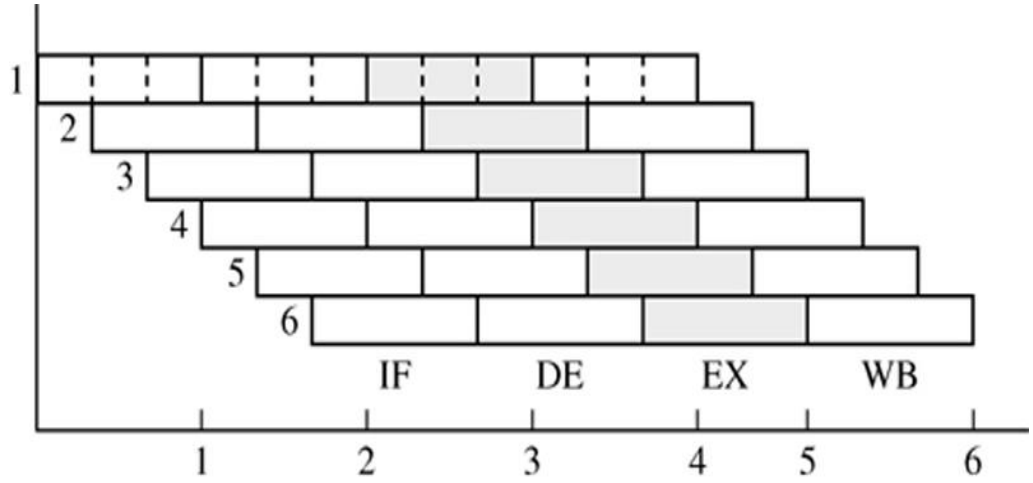
Fetch + Decode
+ Execution + Write

The Basic Pipeline For MIPS



Super-Pipelining Processor

- The machine can issue a new instruction every minor cycle.
- Parallelism = K x m.
- For this figure, //'ism = K x m
  = 4 x 3 = 12.

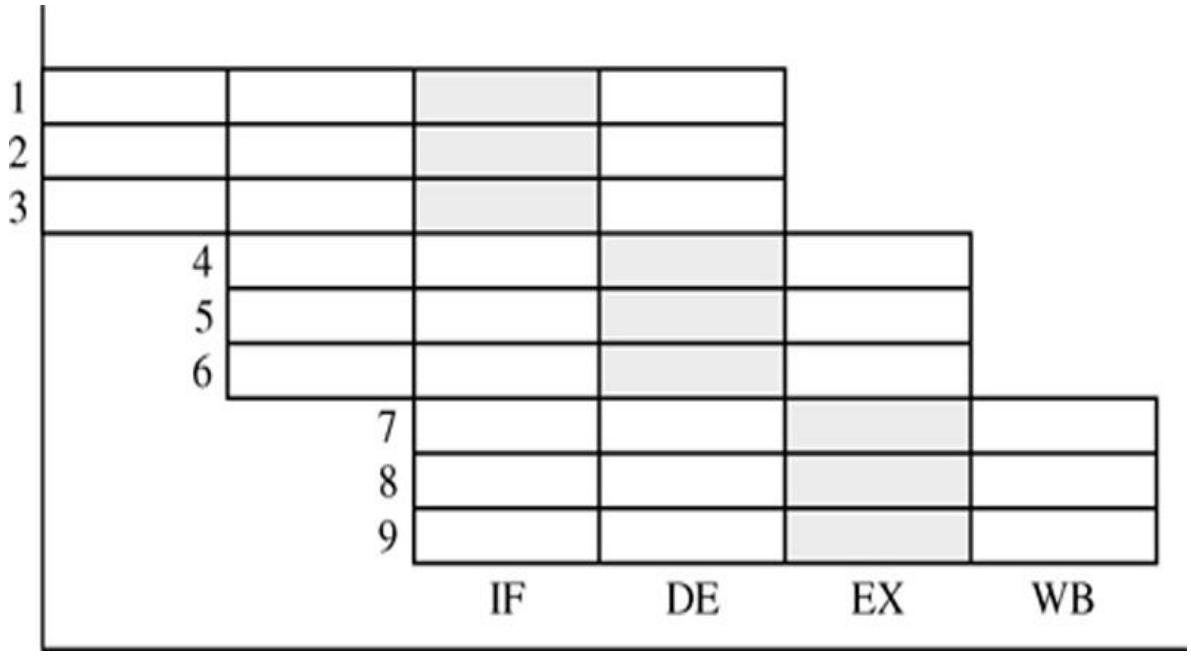


Super-pipelined machine of Degree m=3

Superscalar machine of Degree n=3

- The superscalar degree is determined by the *issue parallelism* <u>*n*</u>, the maximum number of instructions that can be issued in every machine cycle.
- Parallelism = K x n.
- For this figure, //'ism = K x n =

  4 x 3 =12



Superpipelined vs Superscalar

- Superpipelined processors have longer instruction latency than the SS processors which can degrade performance in the presence of true dependencies
- Superscalar processors are more susceptible to resource conflicts – but we can fix this with hardware !

Superpipelined Superscalar Machine

- **The superscalar degree is determined by the *issue parallelism n*, and the sub-stages m, & the number of stages k.**
- **Parallelism = K x m x n**
- **For this figure, //'ism = K x m x n**
  = 5 x 3 x 3 = 45