# Exercise walk-through

Processors, Parallel Processing and Virtual Machines ▪ 15.1

eXercise

# You must be able to

- compare **RISC** and **CISC** processors
- explain **pipelining** and the role of **registers**
- describe **Flynn's taxonomy** and a **virtual machine**

# Method — RISC vs CISC

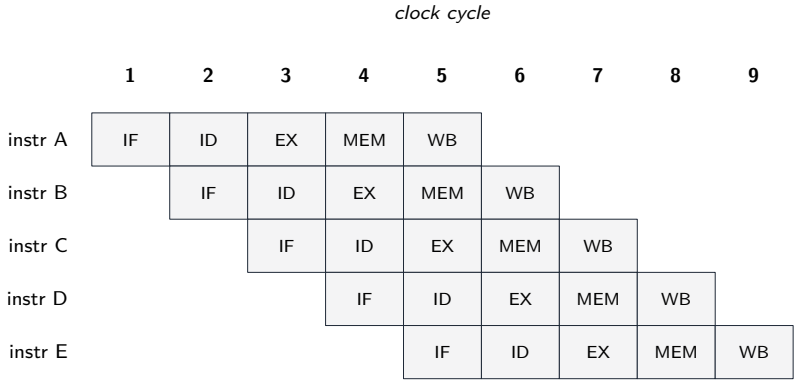
| **Feature** | **CISC** | **RISC** |
|---|---|---|
| Instruction set | many, complex | few, simple |
| Instruction length | variable | fixed |
| Memory access | many instructions | only **load/store** |
| Pipeline-friendly | harder | naturally |

RISC keeps data in **many registers** (fast) and only `load`/`store` touch memory.

# Method — Pipelining

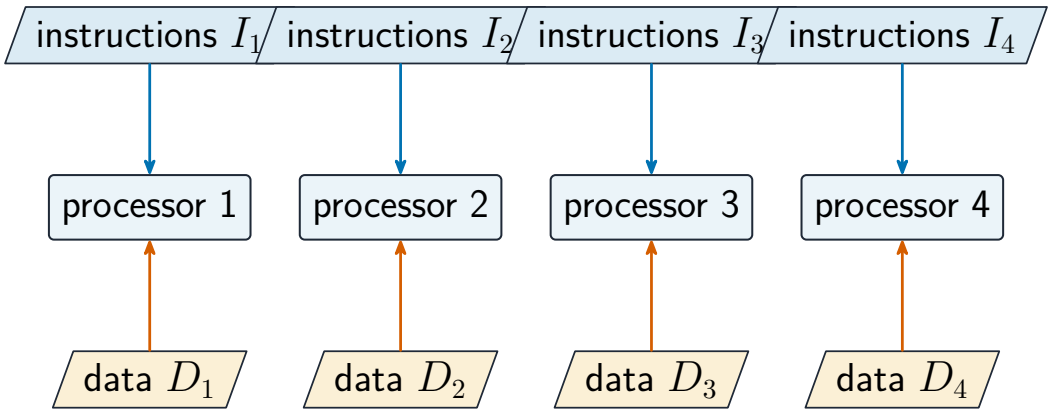
A **pipeline** overlaps the stages of consecutive instructions, like an assembly line, so once full, **one instruction completes per cycle**:
RISC's **fixed-length, simple** instructions make each stage take equal time. A pipeline can **stall** on a **hazard** — a data hazard (a needed result isn't ready) or a control hazard (a branch).

# Method — Pipelining




*pipeline full — one instruction finishes per cycle*

# Method — Pipelining




*MIMD parallelism runs many instruction streams*

# Method — Flynn's taxonomy and virtual machines

- **SISD** — one instruction, one data stream (single core).
- **SIMD** — one instruction on **many data items** (GPU).
- **MISD** — rare/theoretical.
- **MIMD** — many processors, different instructions and data (multi-core, clusters).

A **virtual machine** is **software** that behaves like a separate computer, running its own OS on top of a host — used for isolation, testing and running several OSes at once.

## Practice 2.1 ▪ 1 mark

State **one** difference between a **RISC** and a **CISC** processor.

# Solution 2.1

Method: RISC vs CISC

**Worked steps**

Any one: RISC has **few simple fixed-length** instructions, CISC has **many complex variable-length** ones; in RISC only load/store access memory B1.

## Practice 2.2 ▪ 1 mark

State what **pipelining** does.

# Solution 2.2

**Worked steps**

It **overlaps the stages** of consecutive instructions so more than one is processed at once (one finishes per cycle when full) B1.

## Practice 2.3 ▪ 2 marks

State what **SIMD** stands for and give **one** example of SIMD hardware.

# Solution 2.3

**Worked steps**

**Single Instruction, Multiple Data** B1; example: a **GPU** / graphics card / CPU vector unit B1.

## Practice 2.4 ▪ 1 mark

State what a **virtual machine** is.

# Solution 2.4

Method: Flynn's taxonomy and virtual machines

**Worked steps**

**Software** that behaves like a separate (virtual) computer, running its own operating system on a host machine B1.

## Exam-style 3.1 ▪ 2 marks

Describe **two** features of a RISC processor that make it well-suited to **pipelining**.

# Solution 3.1

Method: Pipelining

**Worked steps**

Any two: **fixed-length** instructions (each stage takes equal time); **simple** instructions (each ~1 cycle); register-to-register operations (only load/store touch memory), so stages are predictable B1 B1.

# Exam-style 3.2 ▪ 3 marks

Explain how **pipelining** improves performance, and name **one** type of hazard that can stall it.

# Solution 3.2

Method: Pipelining

**Worked steps**

While one instruction is being executed, the **next is being decoded and another fetched** B1, so instructions overlap and throughput rises to ~one per cycle B1; a **data** hazard (or control/branch hazard) can stall it B1.

## Exam-style 3.3 ▪ 2 marks

State the **four** categories of **Flynn's taxonomy**.

# Solution 3.3

Method: Flynn's taxonomy and virtual machines

**Worked steps**

**SISD, SIMD, MISD, MIMD** B1 B1 *for two correct, for all four.*

## Exam-style 3.4 ▪ 1 mark

State **one** characteristic of a **massively parallel** computer.

# Solution 3.4

**Worked steps**

Any one: **thousands of processors**; each with its **own (distributed) memory**; they communicate by **message passing**; it is MIMD B1.

## Exam-style 3.5 ▪ 4 marks

Identify **four** features of a **RISC** processor.

# Solution 3.5

**Worked steps**

Any **four**: a small set of **simple** instructions; **fixed-length** instructions; only **load/store** instructions access memory; **many general-purpose registers**; a **hard-wired** control unit; designed for **pipelining** (about one instruction per cycle) B1 *each, max 4.*