A-LEVEL COMPUTER SCIENCE • EXERCISE SHEET

15.1 Processors, Parallel Processing and Virtual Machines

Name: _____ Class: _____ Date: _____ **Total: 17 marks**

Objective

Build the skills to answer exam questions on **processors and parallelism** —RISC/CISC, pipelining, Flynn's taxonomy and virtual machines.

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**

1 Worked examples

Study these first. Each one shows the method for a question type used later —follow the steps and you can do the Practice and Exam-style questions yourself.

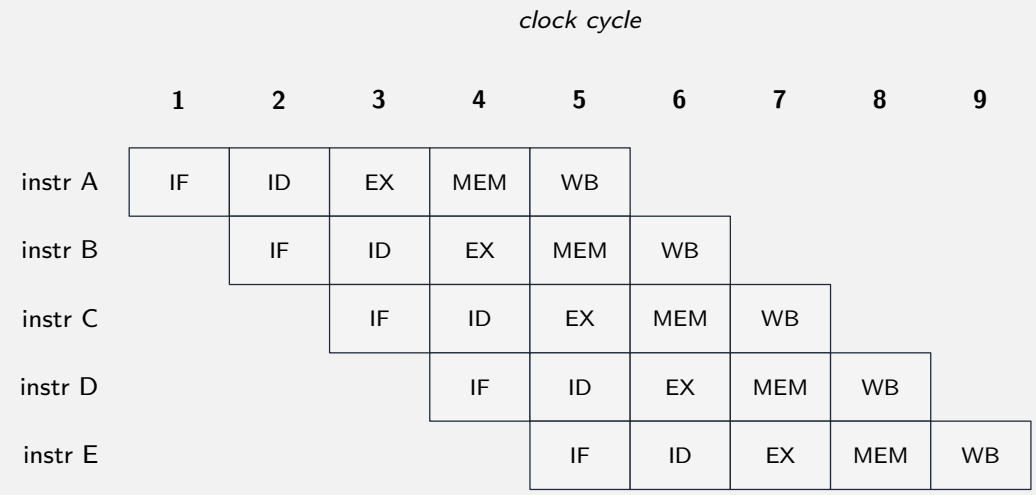
RISC vs CISC

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

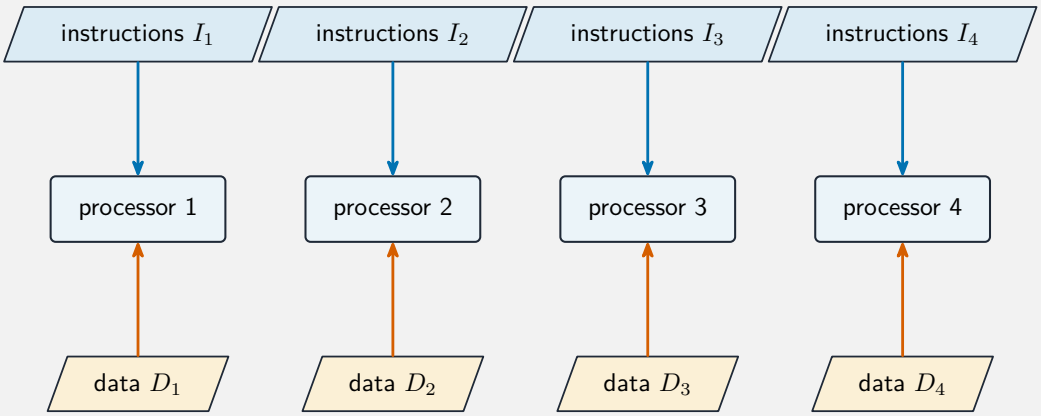
RISC keeps data in **many registers** (fast) and only `load/store` touch memory.

Pipelining

A **pipeline** overlaps the stages of consecutive instructions, like an assembly line, so once full, **one instruction completes per cycle**:



pipeline full — one instruction finishes per cycle



MIMD parallelism runs many instruction streams

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).

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.

2 Practice

Now apply the methods above.

2.1 State **one** difference between a **RISC** and a **CISC** processor. [1]

2.2 State what **pipelining** does. [1]

2.3 State what **SIMD** stands for and give **one** example of SIMD hardware. [2]

2.4 State what a **virtual machine** is. [1]

3 Exam-style questions

3.1 Describe **two** features of a RISC processor that make it well-suited to **pipelining**. [2]

3.2 Explain how **pipelining** improves performance, and name **one** type of hazard that can stall it. [3]

3.3 State the **four** categories of **Flynn's taxonomy**. [2]

3.4 State **one** characteristic of a **massively parallel** computer. [1]

3.5 Identify **four** features of a **RISC** processor. [4]

4 Go further

You are now ready for the real exam questions on this subtopic. Open the **15.1 Processors and parallelism** past-paper sheet in the Library, or try these in **Practice mode**:

- 9618/31 N24 —Q2 (features of a RISC processor)
- 9618/32 N24 —Q8 (features of a CISC processor)
- 9618/33 N24 —Q2 (RISC processor)

Solutions

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

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

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

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

3.1 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.

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

3.3 SISD, SIMD, MISD, MIMD.

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

3.5 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).