

## Section 2.10 — RISC Architecture in ARM (Mazidi)

### Chapter 2 · Section 2.10 — Exercises (Mazidi)

Problems are paraphrased to respect copyright. For background, see **Mazidi, Ch. 2 §2.10**.

**63) What do RISC and CISC stand for?**

**Answer:** RISC = Reduced Instruction Set Computer; CISC = Complex Instruction Set Computer.

**64) In \_\_\_\_\_ (RISC, CISC) architecture we can have 1-, 2-, 3-, or 4-byte instructions.**

**Answer:** CISC.

**Why:** CISC ISAs typically use **variable-length encodings** (e.g., 1–15 bytes in x86).

**65) In \_\_\_\_\_ (RISC, CISC) architecture instructions are fixed in size.**

**Answer:** RISC.

**Why:** RISC ISAs prefer **fixed-length instructions** (e.g., ARM/A32 uses 32-bit fixed; Thumb uses fixed 16-bit with some 32-bit encodings within that mode).

**66) In \_\_\_\_\_ (RISC, CISC) architecture instructions are mostly executed in one or two cycles.**

**Answer:** RISC.

**Why:** RISC designs emphasize **simple, pipelined, single-cycle** operations and load/store memory access.

**67) In \_\_\_\_\_ (RISC, CISC) architecture we can have an instruction to ADD a register to external memory.**

**Answer:** CISC.

**Why:** CISC allows **ALU operations directly on memory operands** (e.g., `ADD [mem], reg`), whereas RISC requires **load → operate → store**.

**68) True or False. Most instructions in CISC are executed in one or two cycles.**

**Answer:** False.

**Why:** CISC instructions often have **variable cycle counts** depending on addressing mode/memory access; they are not predominantly 1–2 cycles in the classic model.

### Notes for learners

- **ARM** is a **RISC** architecture: fixed-size instruction encodings per mode, **load/store** design, simple addressing in ALU ops.
- **CISC** favors **rich addressing modes** and **variable-length encodings**, enabling memory operands in ALU instructions.