1.1 PROCESSORS, INPUT/OUTPUT AND STORAGE · 1.1.1

CPU architecture & the fetch-decode-execute cycle

The components inside the processor, the registers and buses, the **fetch-decode-execute** cycle, and **Von Neumann vs Harvard** architecture. Spec 1.1.1(a)(b)(e).

01 Inside the CPU

ALU Arithmetic, logic and shift operations.

CU Decodes instructions, sends control signals, runs the cycle.

02 Registers

PC Address of the **next** instruction.

CIR The **current** instruction (opcode + operand).

MAR Address being read from / written to.

MDR Data / instruction in transit (a.k.a. MBR).

ACC Working value and ALU results.

03 Buses

Control CU signals: read, write, timing, interrupt.

Data The bits themselves. **Bidirectional**.

Address CPU to memory only. **One-directional**.

Width An **n**-line address bus reaches 2^n locations.

04 Worked example: ADD 20

Fetch PC(08)→MAR; mem→MDR, PC→09; MDR→CIR.

Decode opcode ADD, operand 20 (address) → MAR.

Execute mem[20]→MDR; ALU adds to ACC; store ACC.

05 Fetch · Decode · Execute — the register transfers

01
Fetch

PC → MAR. The instruction is read into the MDR and the **PC is incremented**. MDR → CIR.

02
Decode

The CU splits the CIR into **opcode** and **operand**. An address operand is copied to the MAR.

03
Execute

The opcode is carried out; the result is written to the **accumulator** or memory. Cycle repeats.

06 Von Neumann vs Harvard

Von Neumann

One shared memory and bus for instructions **and** data. Stored-program. Simple; suffers the **bottleneck**.

Harvard

Separate memories and buses for instructions and data. Faster (parallel fetch). Embedded / DSP.

07 Contemporary architecture

Hybrid One main memory (Von Neumann)...

Cache ...split into **instruction** and **data** caches (Harvard).

Why Parallel access near the core without two main memories.

Exam This is the "feature not part of standard Von Neumann".

FINAL PASS BEFORE THE EXAM

Rapid exam tips

Seven slips that quietly lose marks on Paper 1 questions about 1.1.1.

01

The **PC** holds the address of the **next** instruction; the **CIR** holds the **current** one. Never the same register.

02

MAR carries an address; **MDR** carries data or an instruction. Swapping them loses the mark instantly.

03

The **PC is incremented during fetch**, not at the end of the cycle.

04

Define Von Neumann as **shared memory and a single bus for both data and instructions** — not just "stores programs".

05

Address bus is **one-directional** (out of the CPU); the data bus is **bidirectional**.

06

In "which registers are used when X executes" questions, name each register **and** the value it holds at that step.

07

Harvard is faster because instructions and data sit on **separate buses**, so both can be fetched at the same time, avoiding the Von Neumann bottleneck.