Cambridge IGCSE · Computer Science 0478 · Notes

Computer architecture

Read online at **pastopic.com/cambridge/computer-science-0478/notes/computer-architecture**

Last checked 30 September 2026

What you need to know

By the end of this topic you should be able to:

1. **Say what the CPU is for.** The central processing unit processes the instructions and data that are put into a computer, so that a result can be output. It does this by running the fetch–decode–execute cycle, over and over.
2. **Say what a microprocessor is:** a processor built as an integrated circuit on a single chip.
3. **Give the purpose of every part of a CPU in a Von Neumann computer:**
   - the two units: the **arithmetic logic unit (ALU)** and the **control unit (CU)**;
   - the five registers: **program counter (PC)**, **memory address register (MAR)**, **memory data register (MDR)**, **current instruction register (CIR)** and **accumulator (ACC)**;
   - the three buses: **address bus**, **data bus** and **control bus**.
4. **Describe the fetch–decode–execute (FDE) cycle**: how an instruction (and any data) is fetched from RAM into the CPU, decoded and executed, which register each value is stored in, which bus carries it, and which unit does each job.
5. **Explain what a core, the cache and the clock are**, and how the number of cores, the size of the cache and the clock speed change how well a CPU performs.
6. **Say what an instruction set is and what it is for**: the list of all the commands (in machine code) that a CPU can process.
7. **Describe embedded systems**: what they are for, what they are like, how they differ from a general-purpose computer, and which devices usually contain one.

Understand it

What the CPU does

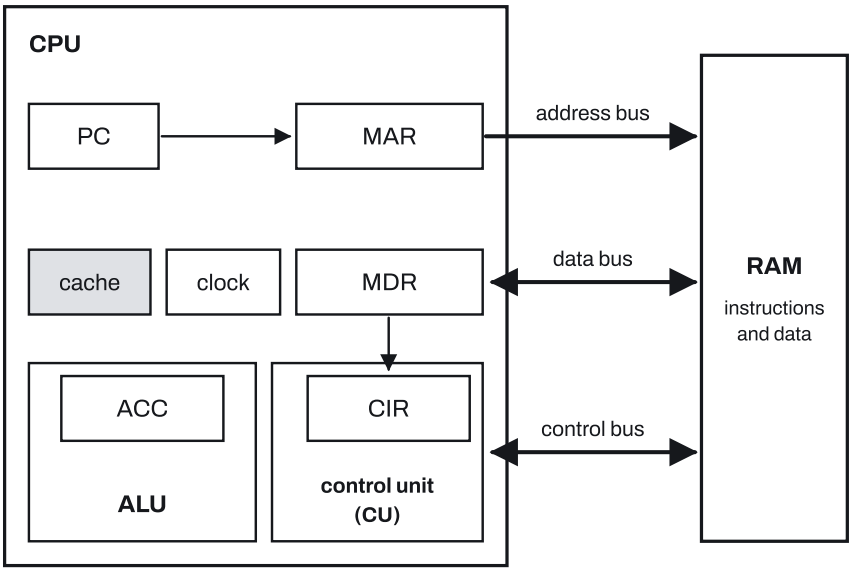
A program is a list of instructions. The **CPU** is the part of the computer that carries them out: it takes each instruction, works out what it means and does it, using the data it needs. Input goes in, the CPU processes it, and results come out. It repeats this for one instruction after another, billions of times a second. The repeating pattern is the **fetch–decode–execute cycle**.

A **microprocessor** is a processor made as one integrated circuit on a single chip. Small devices such as a microwave or a thermostat use a microprocessor to run their one job.

The Von Neumann idea: the stored program concept

In a **Von Neumann** computer, the **program's instructions and its data are stored together in the same memory (RAM)**. The CPU fetches the instructions from memory **one at a time, in order**, and executes each one before fetching the next. This is called the **stored program concept**. Because instructions and data share one memory, the same buses carry both.

Inside the CPU



The two units

- The **control unit (CU)** is in charge. It **decodes** each instruction, using the instruction set, and **sends control signals** (along the control bus) to the other components, so it manages how data moves through the CPU. The **CIR is inside the CU**.
- The **arithmetic logic unit (ALU) executes** instructions: it does the **calculations** (such as adding) and the **logic operations** (such as comparing two values: is A bigger than B?). The **ACC is inside the ALU**.

The five registers

A **register** is a tiny, very fast store inside the CPU. It holds one piece of data, one instruction or one address **temporarily**, while the FDE cycle needs it. Each register has one job:

Register	What it holds
Program counter (PC)	the **address** of the **next** instruction to be fetched
Memory address register (MAR)	the **address** in memory that is about to be read from or written to

Register	What it holds
Memory data register (MDR)	the **data or instruction** that has just come from memory, or is about to be sent to memory
Current instruction register (CIR)	the **instruction** that is being decoded and executed now
Accumulator (ACC)	the **result** of a calculation done by the ALU (an interim result)

Two pairs are easy to mix up. The PC and the MAR both hold **addresses**, never instructions: the PC holds where the *next* instruction is, the MAR holds the address being used *right now*. The MDR and the CIR both hold **instructions**: the MDR is the doorway from memory, the CIR is where the instruction waits while it is decoded.

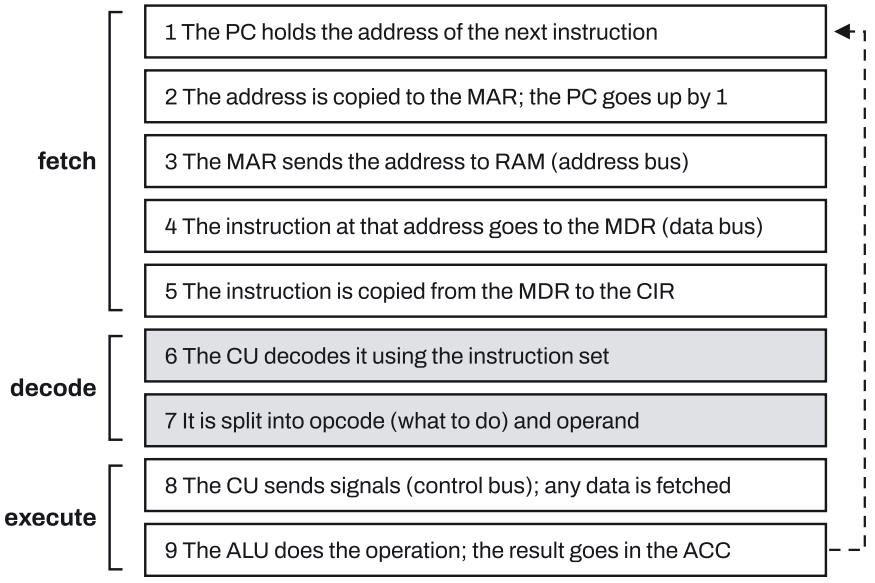
The three buses

A **bus** is a set of wires that carries signals between components.

Bus	Carries	Direction
Address bus	**addresses**, from the MAR to memory	one way: CPU to memory
Data bus	**data and instructions**, between memory and the MDR	both ways
Control bus	**control signals** from the CU, such as "read" or "write", and the clock's timing signals	both ways

The control bus does not *do* the controlling: it only **carries** the signals that the control unit sends.

The fetch–decode–execute cycle



Fetch. The PC holds the address of the next instruction. That address is copied into the MAR, and the PC is **incremented** (it goes up by 1), so it points at the instruction after. The MAR sends the address to RAM along the **address bus**. The instruction stored at that address is sent back along the **data bus** into the MDR. From the MDR it is copied into the CIR, which is in the control unit. The control unit sends signals on the **control bus** throughout (for example, "read from memory"). The four registers used in the fetch stage are the **PC, MAR, MDR and CIR**.

Decode. The control unit **decodes** the instruction in the CIR, using the **instruction set**, to work out what has to be done. The instruction is split into its **opcode** (the command, such as ADD) and its **operand** (the data, or the address of the data, it works on).

Execute. The control unit sends signals to the components that carry out the instruction. If the instruction needs data from memory, its address goes through the MAR and the data comes back through the MDR, just like an instruction. The **ALU** does any calculation or logic operation, and the result is stored in the **ACC**. Then the cycle starts again with the address now in the PC.

For example, suppose RAM address 10 holds `LOAD 50` and address 11 holds `ADD 51`, and addresses 50 and 51 hold 7 and 5. On the first cycle, the MAR gets 10, the PC becomes 11, and `LOAD 50` travels into the MDR and then the CIR. Decoding gives opcode LOAD and operand 50, so the value 7 is fetched and placed in the ACC. On the next cycle, the MAR gets 11 and the PC becomes 12; `ADD 51` makes the ALU add 5 to the ACC, which now holds 12.

Instruction set

An **instruction set** is the **list of all the commands a CPU can process**. The commands are **machine code** (binary). The control unit uses it to decode each instruction. Different kinds of CPU can have different instruction sets, which is why a program compiled for one kind of CPU may not run on another.

Core, cache and clock: what makes a CPU faster

Clock. The clock is a component that sends out a steady stream of pulses. Each pulse sets off one FDE cycle (at this level you can treat one pulse as one cycle), so the clock **controls how many FDE cycles are done each second**. The **clock speed** is the number of pulses (and so cycles) per second, measured in hertz: $1\ \text{GHz} = 1$ billion per second. A 2.4 GHz CPU can do about 2.4 billion FDE cycles each second. Note that the clock is the component; clock speed is a measurement of it.

- Higher clock speed → **more FDE cycles per second** → more instructions processed each second → better performance.

Core. A core is a **complete processing unit** inside the CPU, with its own ALU, control unit and registers, that can **fetch, decode and execute** instructions by itself. A dual-core CPU has 2 cores, a quad-core CPU has 4.

- More cores → **more instructions processed at the same time** (simultaneously), one on each core → better performance.

- A quad-core 3 GHz CPU could process up to $4 \times 3 = 12$ billion instructions per second. It won't always manage that: the program has to be able to split its work between cores, and the cores spend time passing data to each other.

Cache. Cache is a small amount of **very fast memory inside or next to the CPU**. It stores **frequently used data and instructions**, so the CPU can get them faster than from RAM.

- Bigger cache → more frequently used data and instructions can be kept there → **fewer slow trips to RAM** → better performance.

Change	What it means	Effect on performance
More cores	more processing units	more instructions processed **at the same time**
Higher clock speed	more pulses per second	more FDE cycles (instructions) **per second**
Larger cache	more fast memory near the CPU	more frequently used data/instructions accessed **faster than RAM**

Keep "at the same time" for cores and "per second" for clock speed. Faster clock speed does **not** mean cycles happen at the same time.

Embedded systems

An **embedded system** is a computer system **built into a larger device** to do **one dedicated job**. A washing machine's controller only runs washing cycles; a vending machine's only sells items. Typical characteristics:

- a **single, dedicated (limited) function**;
- **dedicated hardware**, only used for that function;
- it uses a **microprocessor**, not a full CPU;
- its program is usually **firmware** stored in ROM, and its function **can't easily be changed or reprogrammed**;
- it is **part of a larger device**, and often small, cheap to make and low on power.

Common examples: domestic appliances (washing machine, microwave, fridge or freezer, dishwasher, central heating or air-conditioning controller), cars (engine management, airbag, satellite navigation), security systems and burglar alarms, lighting systems (a security light), vending machines, calculators, ATMs, smart speakers and simple smartwatches.

A **general-purpose computer** such as a PC or a laptop is the opposite: it does **many different jobs**, you can **install new software**, plug in **peripherals** and upgrade its hardware, and it has a **CPU**.

A microprocessor controlling a device. In many embedded systems a sensor sends readings to the microprocessor all the time. The microprocessor:

1. receives the (digital) data from the sensor, **continuously**;
2. **compares** it with a **stored value** (such as the temperature the user set);

3. if they don't match, **sends a signal** to the device that makes the change (turn the heater on or off, open a door, show a message);
4. if they match, does nothing, and keeps checking.

Key facts and formulas

There is no formula list for this subject, so you must learn all of these.

Formula	Status
CPU: processes instructions and data by running the fetch–decode–execute cycle	Learn it
Microprocessor: a processor on a single integrated-circuit chip	Learn it
Stored program concept: instructions and data are stored together in memory and fetched and executed one at a time	Learn it
CU: decodes instructions (using the instruction set), sends control signals to manage the flow of data; contains the CIR	Learn it
ALU: carries out calculations and logic operations; contains the ACC	Learn it
PC: address of the next instruction · MAR: address about to be used · MDR: data/instruction to or from memory · CIR: instruction being decoded/executed · ACC: result of a calculation	Learn it
Address bus: addresses, one way · data bus: data and instructions, both ways · control bus: control signals, both ways	Learn it
Fetch: PC → MAR (PC + 1) → RAM (address bus) → MDR (data bus) → CIR	Learn it
Decode: CU decodes the instruction in the CIR using the instruction set (opcode + operand)	Learn it
Execute: CU sends signals; ALU does the operation; result in the ACC	Learn it
Clock speed in GHz = billions of FDE cycles per second; cycles per second (all cores) $\approx$ clock speed $\times$ number of cores	Learn it
More cores: more instructions at the same time · higher clock speed: more per second · larger cache: frequently used data accessed faster	Learn it
Instruction set: list of all the (machine code) commands a CPU can process	Learn it
Embedded system: dedicated function, dedicated hardware, microprocessor, firmware not easily changed, part of a larger device	Learn it

How to do it

In the Paper 1 questions we have tagged for this topic (35 questions, 2021–2025), the types came up this often. 22 of the 35 questions mix more than one type, so the counts add up to more than 35.

Question type	Questions
Name or identify components, registers, buses or stages	15
Match components to descriptions	8
Explain how cores, clock speed or cache affect performance	8
Describe the role of a register, unit or bus	8
Explain why a device is (or is not) an embedded system	8
Describe the fetch–decode–execute cycle	7
State what a core, clock speed or cache is	7
State the purpose of the CPU	3
Describe a microprocessor's role in a control system	3
Define or name an instruction set	3
Describe the stored program concept	2

1. Name or identify components, registers, buses or stages (15 questions)

1. **Read exactly what is asked**: a register, a unit, a bus, or a stage. The ALU is a unit, not a register, and a register is never a bus.
2. **Match the clue to the name** using the tables in "Understand it". Typical clues:
   - "the first register used in the fetch stage": **PC**;
   - "registers used in the fetch stage": any of **PC, MAR, MDR, CIR**;
   - "a register used in the execute stage": **ACC** (MAR and MDR also count, when data is fetched);
   - "the register that is part of the CU": **CIR**; "the register in the ALU": **ACC**;
   - "decodes an instruction" or "sends signals": **control unit**; "carries out calculations and logic operations": **ALU** (not the accumulator, which only stores the result);
   - "regulates the number of FDE cycles per second": the **clock** (not "clock speed");
   - "where instructions are fetched from": **RAM**.
3. **Circling from a list**: the real names are the only ones that count. Registers: accumulator, program counter, memory address register, memory data register, current instruction register. Buses: **address, data, control**. Invented names such as "instruction counter", "data counter", "binary register" or "fetch bus" are distractors. Components built into the CPU: ALU, CU, the five registers, cache; not RAM, ROM, the hard disk, the graphics card or the motherboard.
4. If a question asks for the buses used in **fetching**, the address comes from the MAR, but mark schemes have also accepted **control and data** when the address bus isn't on the list: read the options.

2. Match components to descriptions (8 questions)

Tables with some names and some descriptions missing; one mark per gap.

1. For a missing **description**, write the job in a few precise words. The descriptions examiners reward:
   - PC: stores the address of the next instruction to be fetched;
   - MAR: stores the address of the data or instruction about to be fetched;
   - MDR: stores data or instructions just fetched from, or about to be sent to, RAM;
   - CIR: stores the instruction being decoded and executed;
   - ACC: stores the interim result of a calculation;
   - cache: stores frequently used data and instructions;
   - clock: controls how many FDE cycles are performed per second;
   - core: a processing unit that can fetch, decode and execute instructions;
   - register: a component in the CPU that temporarily stores data, an instruction or an address;
   - CU: manages the flow of data through the CPU by sending control signals / decodes instructions.
2. For a missing **name**, look for the key word: "address of the **next**" → PC; "**about to be** fetched" → MAR; "**before or after** RAM" → MDR; "**being decoded**" → CIR; "**result** of calculation" → ACC; "**per second**" → clock; "fetch, decode **and** execute" → core; "list of **all the commands**" → instruction set; "**transmits** data between RAM and the CPU" → data bus.
3. **Tick grids** (for example, MAR / MDR / PC against six statements): some statements fit more than one, such as "it is a register in the CPU" (all three) or "it receives signals from the control unit" (all three). Tick every one that fits.
4. **Spot the wrong statements** and correct them: the usual errors are the MAR "storing instructions" (it stores an address), the MDR "receiving data from the PC" (the address goes from the PC to the MAR), and the ACC "storing the calculation" (it stores the result). Write the full correct statement.
5. Tables sometimes include a row from another topic, such as a MAC address. Answer it anyway.

3. Explain how cores, clock speed or cache affect performance (8 questions)

1. **Name the change**, then **give its consequence** in the same sentence. Each is usually 2 marks: one for the point, one for the expansion.
   - More cores → more instructions (FDE cycles) processed **at the same time**.
   - Higher clock speed → more FDE cycles, so more instructions, **per second**.
   - Larger cache → more frequently used data and instructions stored there, accessed **faster than from RAM**.
2. **"Explain two ways"** (4 marks) needs two different changes, each with its consequence. Two changes with no consequences earn only 2.
3. **"Explain the effect of changing from 2.4 GHz to 3.5 GHz"**: say it can perform more FDE cycles per second (3.5 billion instead of 2.4 billion), so performance **increases**.

Comparing processors. For each, multiply clock speed by the number of cores.

- **"Which can execute most instructions simultaneously?"**: the one with the **most cores**, whatever its clock speed. Say why: each core executes one instruction at a time, so 4 cores can execute 4 at once.
- **"Why can one execute more instructions per second than another?"**: work out billions per second for each, including the cores. A 2.5 GHz dual-core does $2 \times 2.5 = 5$ billion; a 5.2 GHz single core does 5.2 billion, which is more. Forgetting the second core is the common slip.
- A single core also avoids the delay (latency) of cores passing data to each other, which is another valid point.

4. Describe the role of a register, unit or bus (8 questions)

1. Say **what it stores, carries or does**, then **when or why** in the FDE cycle. Use the tables in "Understand it".
2. For the **CU** (often 3 marks), give three separate points: it sends **control signals** to the other components, ... using the **control bus**, ... to manage the flow of data through the CPU; it **decodes** instructions, ... using the **instruction set**.
3. For the **ALU** (3 marks): executes instructions; performs calculations (such as addition); performs logic operations (such as comparisons); stores interim results in the ACC.
4. For a **register** in general: temporarily stores data, an instruction or an address, for fast access during the FDE cycle.
5. For a **bus** in the FDE cycle: name it and say what it carries, between which components (the address bus carries the address from the MAR to RAM; the data bus carries the instruction from RAM to the MDR).
6. For the **PC**, say it stores the **address** of the next instruction. It doesn't store the instruction, and it doesn't count how many instructions have run.

5. Explain why a device is (or is not) an embedded system (8 questions)

1. **"Why is X an embedded system?"** (2 marks): two different characteristics, each tied to the device if you can:
   - it has a single, dedicated function (a calculator only does calculations);
   - it has dedicated hardware;
   - it uses a microprocessor;
   - its function can't easily be changed or reprogrammed (it runs firmware).
2. **"Why is X not an embedded system?"** (a PC, a laptop): write each point as a **contrast**: a general-purpose computer performs **many** functions, **whereas** an embedded system has one; its software **can easily be changed** or new software installed; you can **plug in peripherals**; it has a **CPU** rather than a microprocessor; it stands alone rather than being built into a larger device.
3. **"State what is meant by an embedded system"** (1 mark): a computer system built into a larger device to perform a dedicated function.
4. **Circle the embedded systems** in a list: domestic appliances, security and lighting systems, vending machines and car systems are embedded; PCs, laptops, smartphones and servers are general purpose.

The examiners' advice: don't rely only on "it does one task". Know the other characteristics too.

6. Describe the fetch–decode–execute cycle (7 questions)

Fill-the-gap paragraph (6–7 marks, choose from a list):

1. Say the cycle to yourself in order: **PC → MAR → (address bus) → RAM → (data bus) → MDR → CIR in the CU → decoded by the CU using the instruction set → executed by the ALU**.
2. Read the words either side of each gap. "The program counter contains the _____ of the next instruction" needs **address**, not "instruction".
3. Each term is used once. The list includes traps: "instruction counter", "memory instruction register", "character set", "decoding set", ROM, secondary storage.

Complete and annotate a diagram (usually a CU box and an ALU box, 4–5 marks). Write short labels with arrows, one fact each:

- the CIR is inside the CU, and stores the instruction;
- the instruction arrives from the MDR on the data bus;
- the CU decodes the instruction using the instruction set; it is split into opcode and operand;
- the CU sends a signal to the ALU to execute it (and to RAM, to fetch any data needed);
- the ACC is inside the ALU and stores the data or result;
- the ALU carries out the calculation or logic operation.

Draw and annotate the fetch stage (4 marks): boxes for PC, MAR, MDR, CIR/CU and RAM; arrows PC → MAR → RAM → MDR → CIR; label the address bus and the data bus on the right arrows, "PC is incremented", and the CU sending signals on the control bus.

"Describe what happens at the decode stage" (3 marks): the instruction goes from the MDR to the CIR in the CU (on the data bus); the control unit decodes it using the instruction set; it is split into opcode and operand.

7. State what a core, clock speed or cache is (7 questions)

- **Core**: an independent processing unit in the CPU that can fetch, decode and execute instructions (it performs the FDE cycle, with its own registers and units).
- **Clock speed**: the number of FDE cycles (clock pulses) performed each second. "3.5 GHz" means 3.5 billion cycles per second.
- **Cache**: stores **frequently used** data and instructions, so they can be accessed faster than from RAM. In a tick-box, choose "frequently used instructions", not "the next instruction" or "important instructions".

8. State the purpose of the CPU (3 questions)

To process (execute) instructions and data, by performing the fetch–decode–execute cycle. For 2 marks give both ideas. If asked to name its stages: fetch, decode, execute.

9. Describe a microprocessor's role in a control system (3 questions)

Use the context. Answer in steps (usually 3 marks):

1. The microprocessor **continuously receives** (digital) data from the sensor (a temperature sensor, an infrared sensor).
2. It **compares** the data with a **stored value**, using the number in the question (21 °C, 1 metre).
3. If the value is outside the target, it **sends a signal** to the output (heater, motor, screen) to act.
4. If it matches, it does nothing and keeps checking.

The sensor sends data all the time, not only when something happens. An actuator makes physical movement; it does not change a screen message.

10. Define or name an instruction set (3 questions)

"A list of all the (machine code) commands that can be processed by a CPU." Don't define it with the words "instruction" and "set" alone. If the question describes this list and asks for its name: **instruction set**.

11. Describe the stored program concept (2 questions)

Two points: instructions **and data** are stored **together in the same memory**; the instructions are **fetched and executed one at a time, in order** (sequentially).

Worked examples

The bold codes show where marks are earned, as in Cambridge mark schemes. The number is how many marks.

- **B1** mark for a correct result on its own, with no method needed.

Example 1

A computer has a CPU with a Von Neumann architecture. Complete the table by writing the missing component names and descriptions. **[5]**

Component	Description
memory address register (MAR)	?
?	stores the result of a calculation carried out by the ALU
clock	?

Component	Description
?	a processing unit in the CPU that can fetch, decode and execute instructions
?	carries signals from the control unit to other components

Answer

- MAR: stores the address of the data or instruction that is about to be fetched from (or written to) memory. **B1**
- **Accumulator (ACC)**. **B1**
- Clock: controls the number of fetch–decode–execute cycles performed per second. **B1**
- **Core**. **B1**
- **Control bus**. **B1**

"Clock speed" would not score for the third row: the clock is the component; clock speed is how fast it runs.

Example 2

A CPU performs the fetch–decode–execute cycle.

(a) Describe how an instruction is fetched from RAM into the CPU. **[5]**

(b) Give the name of the register that holds the instruction while it is decoded, and the component it is part of. **[2]**

(a) Any five of:

- the program counter (PC) holds the address of the next instruction; **B1**
- the address is copied from the PC to the memory address register (MAR); **B1**
- the PC is incremented; **B1**
- the address is sent from the MAR to RAM using the address bus; **B1**
- the instruction stored at that address is sent to the memory data register (MDR) using the data bus; **B1**
- the instruction is copied from the MDR to the current instruction register (CIR); **B1**
- the control unit sends signals on the control bus to manage each step. **B1**

(b) Current instruction register (CIR) **B1**, part of the control unit (CU) **B1**.

Example 3

Three computers have these processors.

Computer	Clock speed	Cores
X	3.0 GHz	4

Computer	Clock speed	Cores
Y	4.0 GHz	2
Z	4.6 GHz	1

(a) Identify which computer is most likely to execute the most instructions simultaneously. Justify your answer. **[3]**

(b) Explain why Computer Y can execute more instructions per second than Computer Z, even though Z has the faster clock. **[2]**

(c) Computer Z's cache is increased from 4 MB to 16 MB. Explain how this can improve its performance. **[2]**

(a) Computer X. **B1** It has 4 cores **B1**, and each core executes one instruction at a time, so X can execute 4 instructions at once, while Y can execute 2 and Z only 1. **B1**

(b) Y has two cores, each doing 4.0 billion cycles per second, so it can do up to $2 \times 4.0 = 8$ billion instructions per second **B1**; Z's one core does 4.6 billion per second, which is fewer. **B1**

(c) More frequently used data and instructions can be stored in the cache **B1**, so the CPU can fetch them faster than it could from RAM (fewer trips to RAM). **B1**

Example 4

A smart thermostat keeps a room at the temperature set by the user. It contains a temperature sensor and a microprocessor, and switches a heater on and off.

(a) Explain why the thermostat is an example of an embedded system. **[2]**

(b) The user sets the temperature to 21 °C. Explain how the microprocessor keeps the room at this temperature. **[4]**

(c) The thermostat's microprocessor has an instruction set. State what is meant by an instruction set. **[1]**

(a) Any two: it has a single, dedicated function (controlling the room's temperature) **B1**; it has dedicated hardware **B1**; it uses a microprocessor **B1**; its program is firmware that can't easily be changed **B1**.

(b)

- The sensor sends temperature data to the microprocessor continuously **B1** (converted from analogue to digital).
- The microprocessor compares each reading with the stored value, 21 °C. **B1**
- If the reading is below 21 °C, it sends a signal to switch the heater on; if it is above 21 °C, a signal to switch the heater off. **B1**
- If the reading equals 21 °C, no action is taken, and the process repeats. **B1**

So a reading of 19 °C switches the heater on, 23 °C switches it off, and 21 °C changes nothing.

(c) A list of all the machine code commands that the processor can process. **B1**

Common mistakes

- **The PC stores the instruction.** It stores the **address** of the next instruction. It also doesn't count how many instructions have run. Write "address" every time.
- **The MAR stores data or instructions.** The MAR only ever holds an **address**; the MDR holds what comes from, or goes to, that address.
- **"The accumulator stores calculations"** or **"the accumulator carries out calculations"**. The ALU carries out the calculation; the ACC stores the **result**.
- **Giving the ALU as a register** when asked for registers in the fetch stage. It is a unit. The fetch registers are PC, MAR, MDR and CIR.
- **Writing "clock speed" for the component** that controls the number of FDE cycles per second. The component is the **clock**; clock speed is its measurement.
- **Saying a higher clock speed lets cycles run "simultaneously".** Clock speed means more cycles **per second**; only more cores gives work done **at the same time**.
- **Describing the cache vaguely**, such as "stores data that has just been used or is about to be used". The rewarded idea is **frequently used** data and instructions, accessed faster than RAM.
- **Forgetting the cores** when working out instructions per second: a 2.5 GHz dual-core does 5 billion, not 2.5 billion.
- **Saying the control bus "controls" components.** It **carries** control signals; the control unit sends them.
- **Stating a point without its effect** in "explain" questions: "it has more cores" earns half the marks at most; add "so more instructions can be processed at the same time".
- **Embedded systems: only saying "it does one task".** Also give dedicated hardware, a microprocessor, or firmware that isn't easily reprogrammed.
- **Answering a sensor question in general terms.** Use the device and the numbers in the question (21 °C, 1 metre), and say the sensor sends data continuously.
- **Defining an instruction set as "a set of instructions".** Say it is the **list of all the commands** (machine code) the CPU can process.

How to get the marks

- **Use the exact names**: program counter, memory address register, memory data register, current instruction register, accumulator, control unit, arithmetic logic unit, address bus, data bus, control bus. Abbreviations (PC, MAR, ...) are accepted. Made-up names score nothing.
- **"Identify", "give", "state"** (1 mark each): the name or one precise fact. Don't add a second answer that contradicts it.
- **"Describe"** (the role, the decode stage, a process): one separate fact per mark, in order. For the FDE cycle, name the register **and** the bus at each step: "the address is sent from the MAR to RAM **using the address bus**".

- **"Explain"**: every point needs its consequence ("..., so more instructions are processed per second"). "Explain two ways" caps at half marks without the expansions.
- **"Justify"**: after choosing, give the reason from the data (4 cores, so 4 instructions at once), not just "it is better".
- **Numbers in the question** are there to be used: turn GHz and cores into instructions per second when you compare processors.
- **Gap-fills and tables**: one mark per gap, in the right place only. Each word from the list once. Read the words around the gap.
- **Diagram questions**: short labels on arrows, one fact each; show which register sits inside which unit (CIR in the CU, ACC in the ALU) and label the buses.
- **Context questions** (embedded systems, sensors): tie each point to the device in the question.
- **Tick one box**: ticking two scores nothing.

Quick check

1. Which register holds the address of the next instruction to be fetched, and what happens to it during the fetch stage?
2. Name the three buses and state what each carries.
3. A CPU has a clock speed of 3.2 GHz. (a) How many FDE cycles can each core perform per second? (b) If the CPU is dual-core, what is the most instructions it could process per second?
4. Circle the three embedded systems: dishwasher, laptop, burglar alarm, web server, car engine management system, desktop PC.
5. Two of these statements are wrong. Identify them and correct them. (A) The PC stores the address of the next instruction. (B) The MDR stores the address of the data about to be fetched. (C) The CIR stores the instruction being decoded. (D) The ACC stores the calculation.

Answers

1. The program counter (PC). Its address is copied to the MAR, then the PC is incremented (goes up by 1) so it points at the next instruction.
2. Address bus: addresses, from the MAR to memory. Data bus: data and instructions, between memory and the MDR. Control bus: control signals from the control unit.
3. (a) 3.2 billion cycles per second. (b) $2 \times 3.2 = 6.4$ billion instructions per second (one on each core at a time).
4. Dishwasher, burglar alarm, car engine management system. Each does one dedicated job inside a larger device; the laptop, web server and desktop PC are general-purpose computers.
5. B and D. (B) The **MAR** stores the address of the data about to be fetched (the MDR stores the data itself once fetched). (D) The ACC stores the **result** of a calculation (the ALU does the calculation).

Past questions to try

- **0478/11/O/N/24 Q5**: fetch registers, the decode stage, circling buses and the effect of a faster clock.
- **0478/13/M/J/24 Q8**: the CPU's role, registers, the ALU and how a new CPU can perform better.
- **0478/11/M/J/24 Q5**: drawing and annotating the fetch stage (with RAM and optical storage in the other parts).
- **0478/12/O/N/23 Q4**: characteristics of an embedded system, and circling examples.