4.1 Central Processing Unit (CPU) Architecture

Name: _____ Class: _____ Date: _____ **Total: 41 marks**

KEY WORDS register • von neumann • accumulator • control unit • memory data register

Objective

Build the skills to answer exam questions on **CPU architecture** – the Von Neumann model, the registers, the buses, and the fetch–execute cycle.

You must be able to:

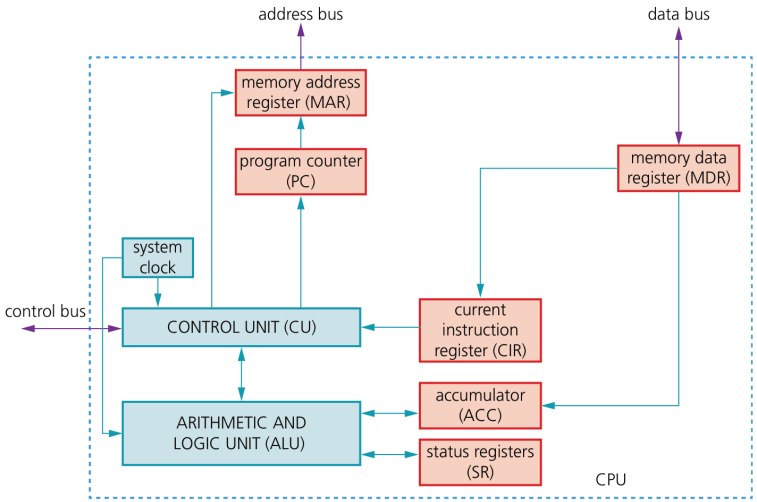
- explain the **Von Neumann** model and the **stored-program** concept
- state the role of the **registers**, the **ALU**, the **control unit** and the clock
- explain the **address**, **data** and **control buses**
- describe the stages of the **fetch–execute cycle** and the purpose of **interrupts**

1 Key knowledge and 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.

■ Von Neumann model and registers

In the **Von Neumann** model one memory holds **both program and data** (the **stored-program** idea), and the CPU runs the instructions one at a time, in order, unless a branch changes the flow.

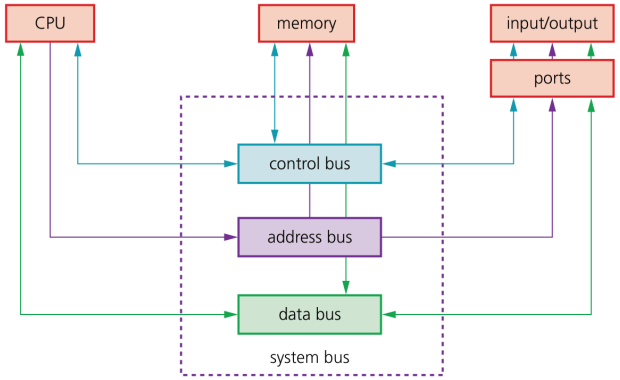


Registers are tiny, very fast stores inside the CPU. Each special-purpose register has a fixed job:

Register	Holds
PC (program counter)	address of the **next** instruction
MAR (memory address register)	the address being read/written
MDR (memory data register)	the data going to/from memory
CIR (current instruction register)	the instruction being decoded
ACC (accumulator)	the value the ALU is working on
IX (index register)	a value added to a base address (indexed addressing)

The **ALU** does arithmetic and logic; the **control unit** decodes instructions and sends control signals; the **clock** keeps everything in step.

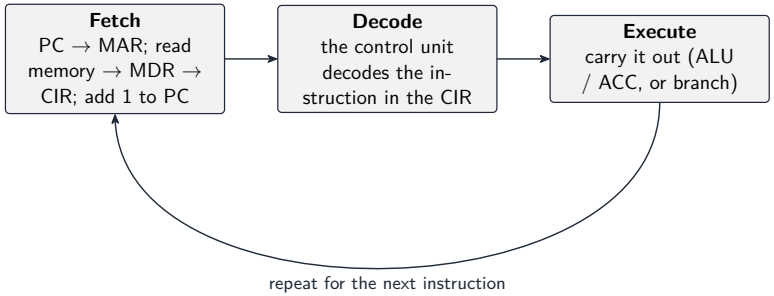
The three buses



The **address bus** is **one-way** (the CPU only ever *sends* an address to memory); the **data** and **control** buses are **two-way**. The **width** of a bus matters: a wider **data bus** carries more bits per transfer (faster), and a wider **address bus** can address **more** memory locations (each extra line doubles the number of addresses).

The fetch–execute cycle

Every instruction goes through the same three stages, then the cycle repeats:



An **interrupt** is a signal that makes the CPU pause the current program, run a service routine, then return – so urgent events are handled quickly.

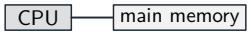
Key knowledge: full processor questions

- **SR** holds status flags; **IX** holds an offset added to an address. Main memory (IAS) holds instructions and data. General-purpose registers hold working values; special-purpose registers have defined roles.
- Fetch notation: `MAR ← [PC]`, `PC ← [PC] + 1`, `MDR ← [[MAR]]`, `CIR ← [MDR]`; then decode and execute. Brackets denote contents; double brackets denote memory contents at the address.
- Interrupt: finish the current instruction; check priority; save PC/register state; load the ISR address into PC; service the event; restore state and resume. Hardware examples include a timer or peripheral request; software examples include a service request or an exception.
- Higher clock frequency can increase instruction rate; architecture and cycles per instruction also matter. More cores help parallel tasks; a wider data bus transfers more bits; a wider address bus addresses more locations; cache reduces slower main-memory access.
- **USB**: bidirectional data and power to peripherals. **HDMI**: digital video and audio. **VGA**: analogue video, without audio.

2 Practice

Now apply the methods above.

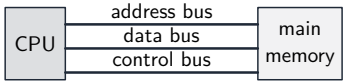
2.1 State what the **stored-program concept** means. [1]



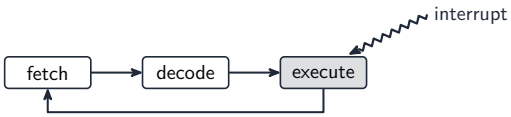
2.2 State the role of the **PC** and the **MAR**. [2]

2.3 State the difference between a **general-purpose** and a **special-purpose** register. [2]

2.4 State which bus is **one-way**, and explain why. [2]

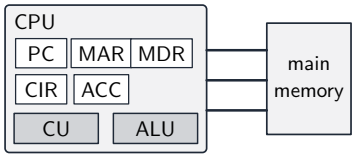


2.5 An interrupt arrives while an instruction is executing. Describe how the processor handles it and then continues the interrupted program. [5]



3 Exam-style questions

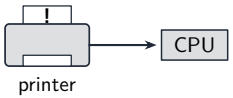
3.1 Describe the stages of the **fetch–execute cycle**. [4]



3.2 (a) State the role of the **ALU** and the **control unit**. [2]

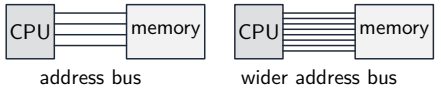
(b) State **two** factors that affect the performance of a CPU. [2]

3.3 Explain the purpose of an **interrupt**. [2]



3.4 During the fetch stage, describe the roles of the **MAR** and the **MDR**. [2]

3.5 A computer is redesigned with a **wider address bus**. Explain the effect this has on the computer. [2]

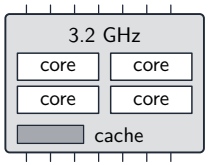


3.6 A processor has four cores, a clock speed of 3.2 GHz and a small amount of cache memory.

(a) **Complete** the table by describing the role of each register. [4]

register	role
Program Counter (PC)	
Memory Address Register (MAR)	
Memory Data Register (MDR)	
Current Instruction Register (CIR)	

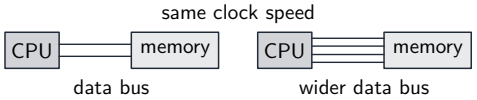
(b) **Explain** how increasing the **number of cores** can improve performance, and **give** one reason why four cores rarely make a program four times faster. [3]



(c) **Explain** how increasing the amount of **cache** improves performance. [2]

(d) **State** three differences between **Dynamic RAM** and **Static RAM**, and **identify** which is normally used for cache. [4]

(e) A second processor has the same clock speed but a **wider** data bus. **Explain** the effect on performance. [2]



Solutions

Each answer shows the steps, then the **result**.

2.1

- both the **program (instructions) and the data** are held together in the **same memory**

2.2

- PC holds the address of the **next** instruction
- MAR holds the address currently being read from or written to memory

2.3

- a special-purpose register has a **fixed role** in the cycle (e.g. PC, MAR)
- a general-purpose register holds **temporary values** chosen by the program

2.4

- the **address bus** is one-way
- the CPU only ever **sends** an address to memory – memory never sends an address back

2.5

Finish the current instruction; check whether the interrupt should be serviced, including priority; save PC/register state; load and run the ISR; restore the saved state and resume the interrupted program.

3.1

- **fetch:** the address in the PC is copied to the MAR; the instruction is read from memory into the MDR, then the CIR; the PC is increased by 1
- **decode:** the control unit decodes the instruction in the CIR
- **execute:** the instruction is carried out (ALU/ACC, or a branch)

3.2

(a) the ALU performs **arithmetic and logic** operations

- the control unit **decodes instructions and sends control signals**

(b) any two of: clock speed; number of cores; cache size; bus width

3.3

- an interrupt is a signal that makes the CPU **pause** the current program
- it runs a routine to handle an urgent event, then **returns** to where it left off

3.4

- the MAR holds the **address** of the instruction to fetch
- the MDR holds the **instruction/data** read back from that address

3.5

- a wider address bus has more lines, so it can carry **more distinct addresses**
- the CPU can then directly address **more memory** (each extra line doubles the addressable memory)

3.6

(a) **PC** – holds the address of the **next** instruction to be fetched

- **MAR** – holds the address currently being read from or written to
- **MDR** – holds the data or instruction just fetched from, or about to be written to, that address
- **CIR** – holds the instruction currently being decoded and executed

(b) each core can fetch and execute its own instructions, so several instructions run **at the same time**

- the program must be written to divide its work between cores, and parts of it are inherently sequential
- the cores also compete for the same memory and bus, so the gain is less than the core count suggests

(c) cache holds recently and frequently used instructions and data close to the processor

- it is far faster to access than main memory, so more cache means fewer slow main-memory fetches

(d) any three: DRAM stores each bit in a **capacitor** that must be **refreshed**, SRAM uses flip-flops and does not; SRAM is **faster**; SRAM is more expensive per bit and takes more space, so DRAM gives more capacity for the money

- **SRAM** is used for cache

(e) a wider data bus carries **more bits per transfer**

- each fetch moves more data and fewer transfers are needed for the same work, raising throughput without any change in clock speed