KNIGHTON GRANGE
EDUCATION

GCSE
COMPUTER SCIENCE

Cambridge OCR J277

1.1 – Systems architecture

1.1.1
Architecture of the CPU

Lesson Objectives

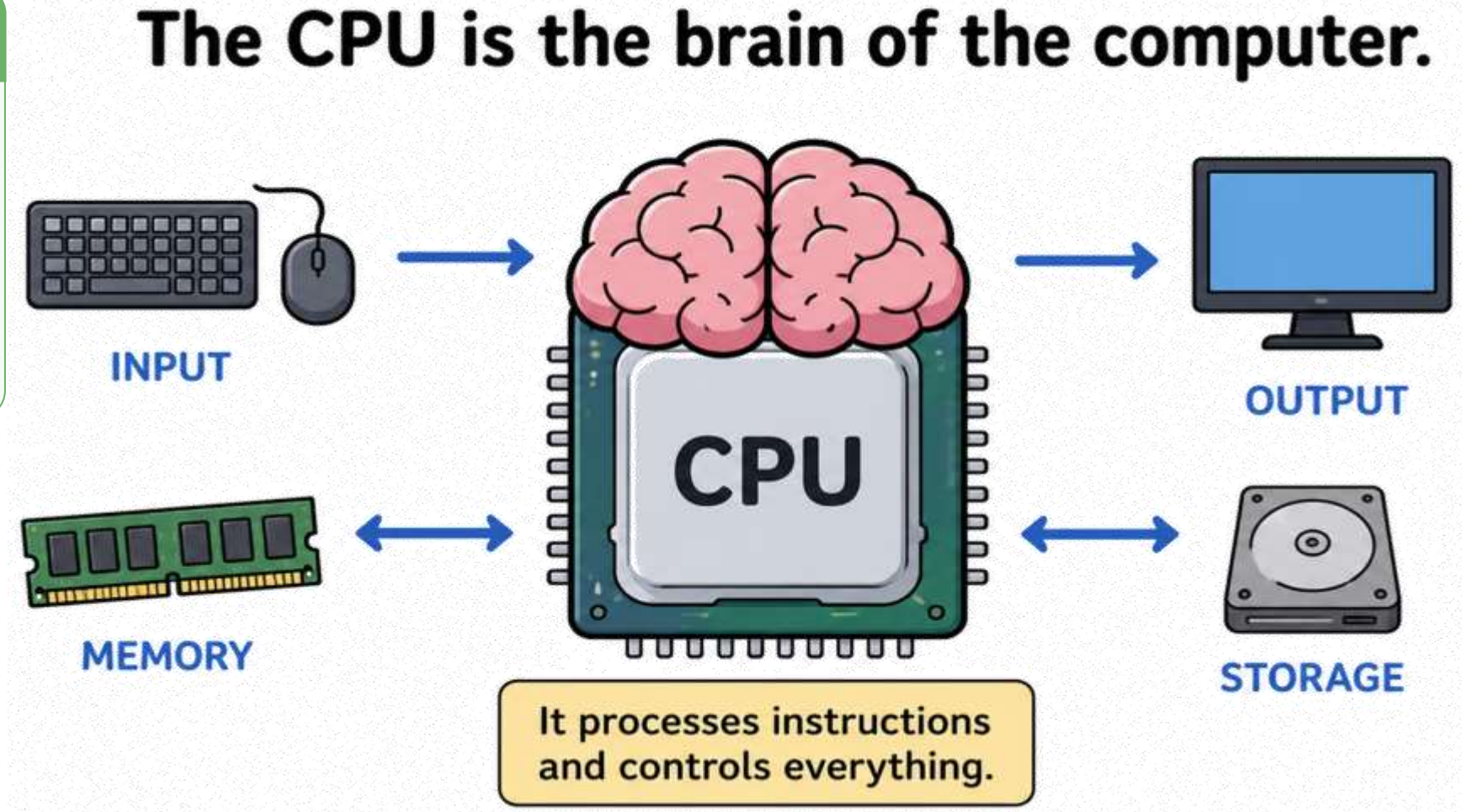
01	Describe the purpose of the CPU and explain its role in running programs.
02	Explain the stages of the fetch-execute cycle and how instructions are processed by the CPU.
03	Identify the main components of the CPU (ALU, Control Unit, Cache, and Registers) and describe the function of each.
04	Explain the principles of the Von Neumann architecture and why it is used in modern computers.
05	Describe the purpose of key CPU registers including the Memory Address Register (MAR), Memory Data Register (MDR), Program Counter (PC), and Accumulator.
06	Apply knowledge of CPU components and registers to explain how data and instructions move through the CPU during

Key Focus

The Purpose Of The CPU

Explanation

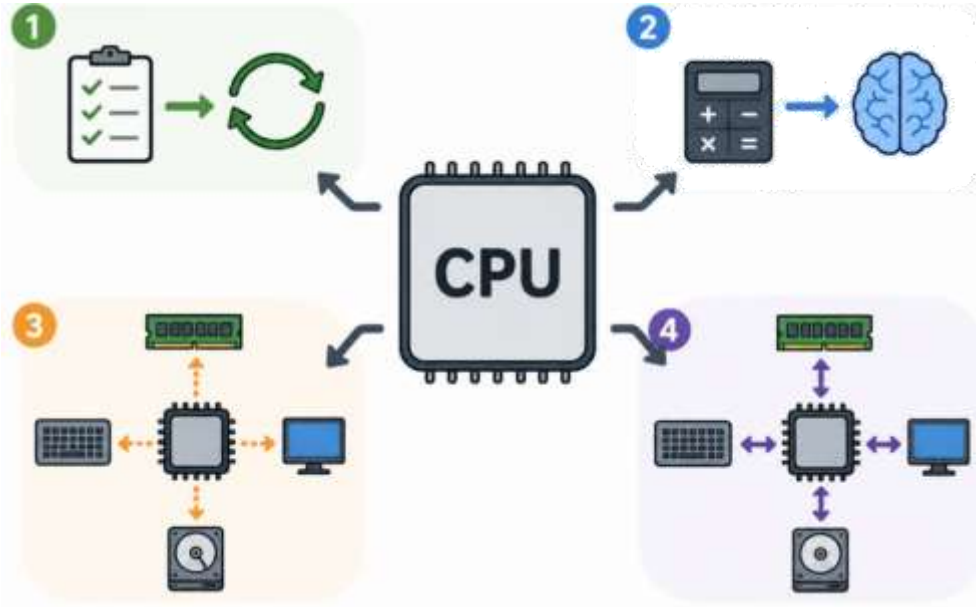
The Central Processing Unit (CPU) is often called the "**brain of the computer**" because it is responsible for processing instructions and controlling the operation of the entire computer system.



The Purpose of the CPU

Its main purposes are to:

1. **Execute program instructions** by carrying out the steps in the **fetch-execute cycle**.
2. **Process data** by performing calculations and logical operations using the **Arithmetic Logic Unit (ALU)**.
3. **Control and coordinate hardware** by sending signals to other components such as memory, storage, and input/output devices. This is done by the **Control Unit (CU)**.
4. **Manage the flow of data** between the CPU, memory, and other devices.



★ GCSE Definition

The CPU is the component that fetches, decodes, and executes program instructions while controlling the operation of the computer.

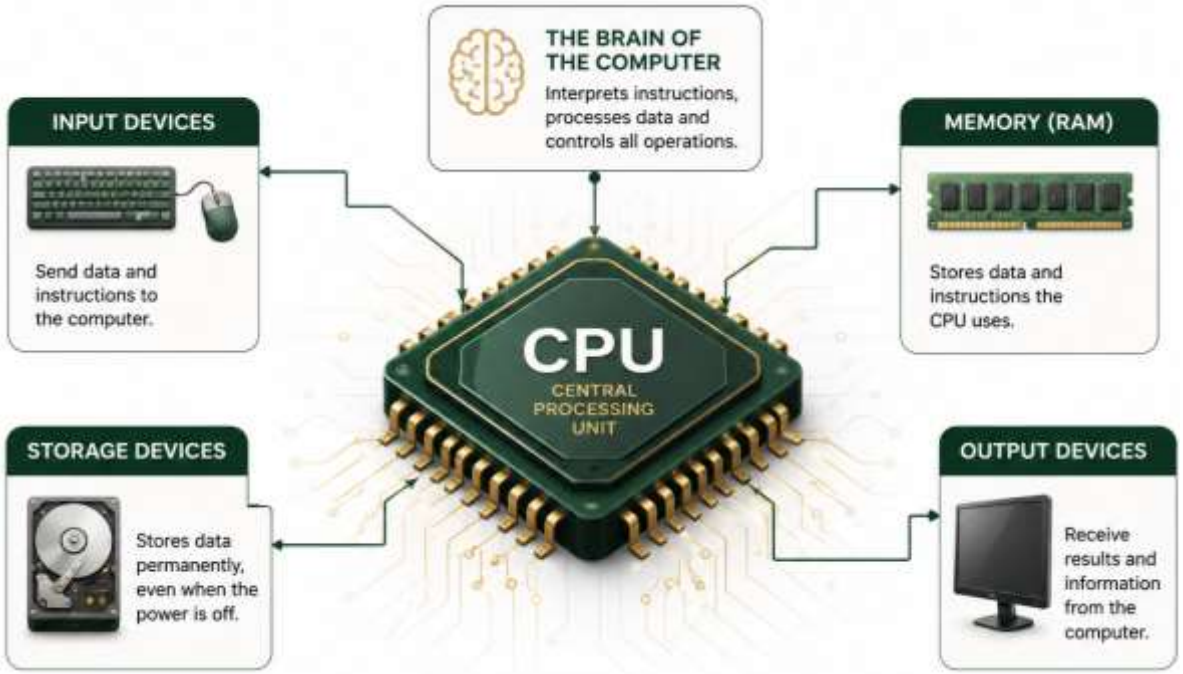
The Purpose of the CPU

Think of a computer as a school:

- The CPU is the headteacher.
- It decides what needs to happen next.
- It gives instructions to different parts of the school (memory, storage, input and output devices).
- It also solves problems and makes calculations when needed.

Example

- The CPU **fetches** the instructions for opening the browser from memory.
- It **decodes** the instructions to understand what they mean.
- It **executes** the instructions, loading the browser and displaying it on the screen.
- This process happens millions or even billions of times every second.



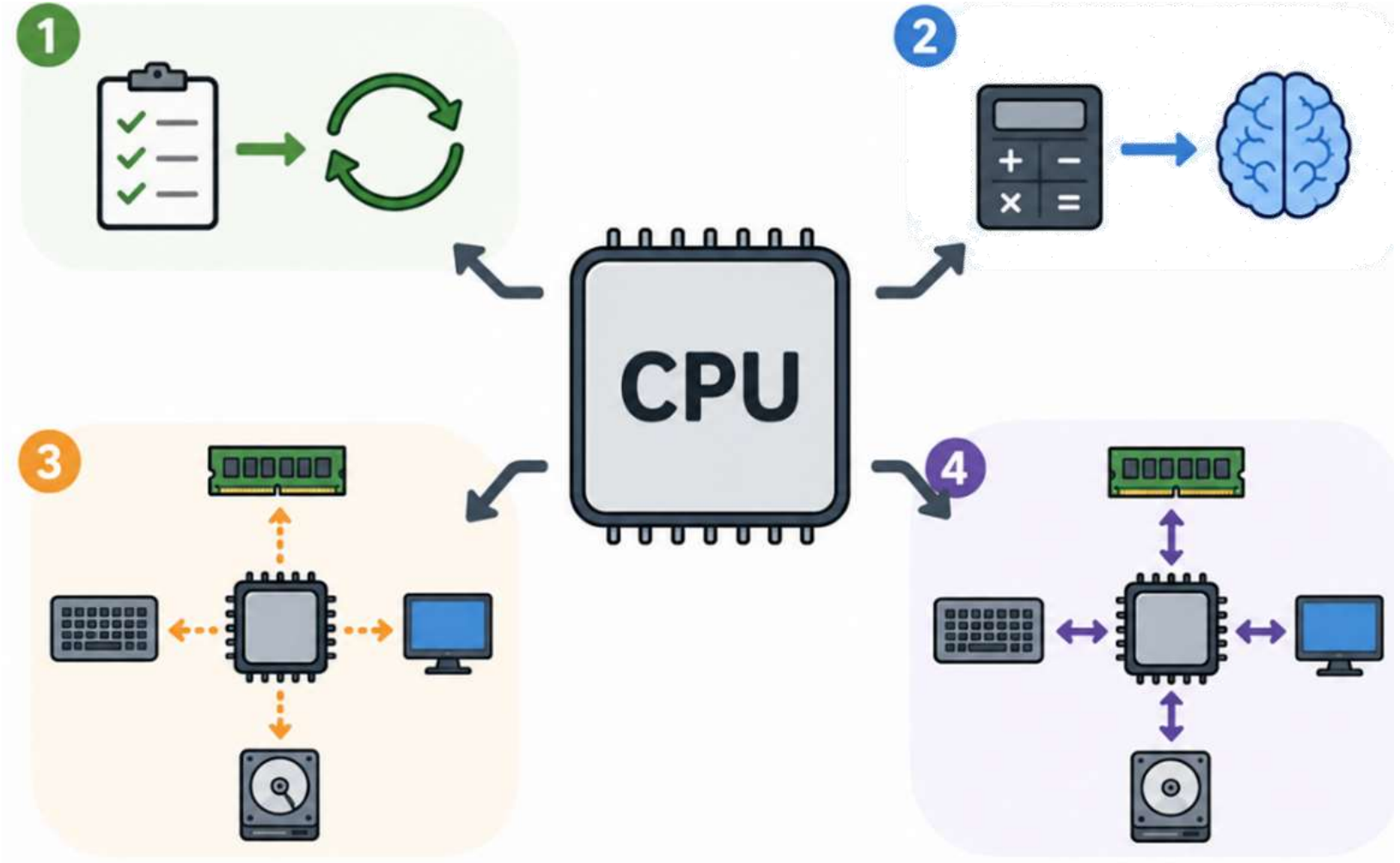
The Purpose of the CPU

Key GCSE Exam Point:

The CPU's job is not to permanently store data. Instead, it processes instructions and controls the computer's operations.

Common Misconception:

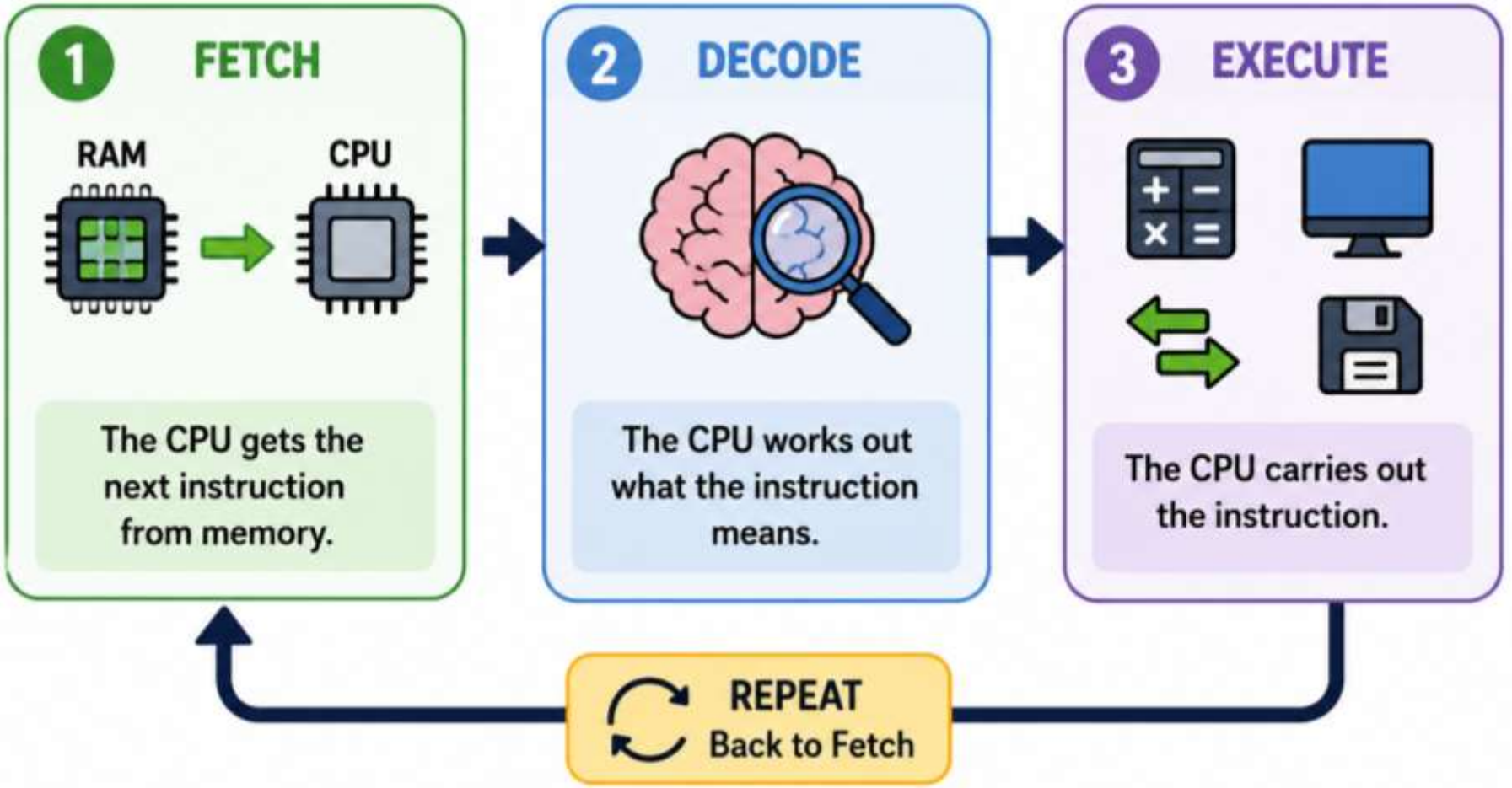
The CPU does not permanently store data.
It processes data and instructions.



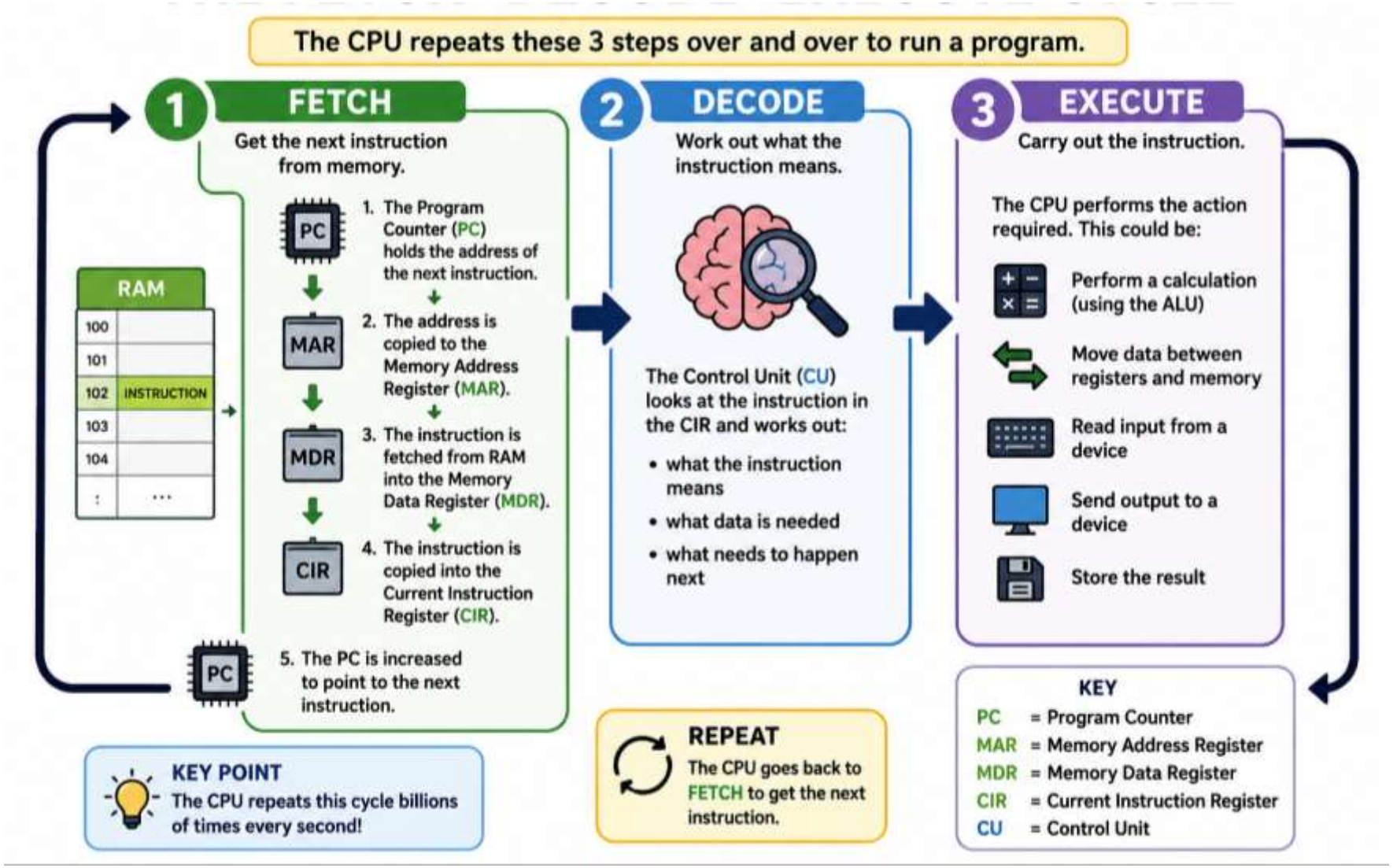
The Fetch-Execute Cycle

Explanation

The **fetch-execute cycle** is the process the **CPU** uses to run a program. It continuously **fetches**, **decodes**, and **executes** instructions stored in memory.



The Fetch-Execute Cycle



Common CPU Components and Their Functions

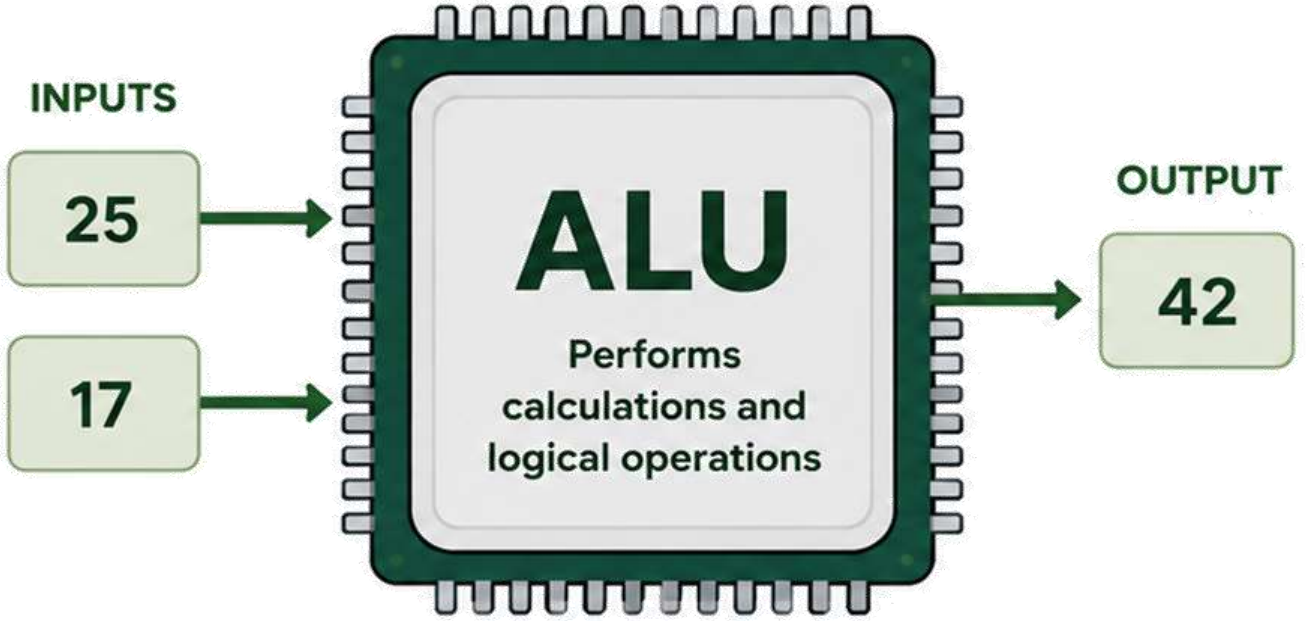
1. Arithmetic Logic Unit (ALU)

What is it?

The **Arithmetic Logic Unit (ALU)** is the part of the CPU that performs **calculations** and **logical decisions**.

It can:

- Perform maths calculations (+, −, ×, ÷)
- Compare numbers (>, <, =)
- Make logical decisions (AND, OR, NOT)



★ GCSE Definition

The ALU performs arithmetic calculations and logical operations.

Common CPU Components and Their Functions

1 ALU (Arithmetic Logic Unit)

The ALU is the part of the CPU that performs **calculations** and **logical operations.**

WHAT IT DOES

- + − × ÷ — **Performs arithmetic calculations** +, −, ×, ÷
- > < = — **Compares numbers** >, <, =
- AND OR NOT — **Performs logical operations** AND, OR, NOT

REAL-LIFE EXAMPLES

- When you calculate **25 + 17**, the ALU performs the calculation.
- When a game checks if your health is **greater than 0**, the ALU compares the values.

GCSE EXAM POINT

The ALU performs arithmetic calculations and logical operations.

Common CPU Components and Their Functions

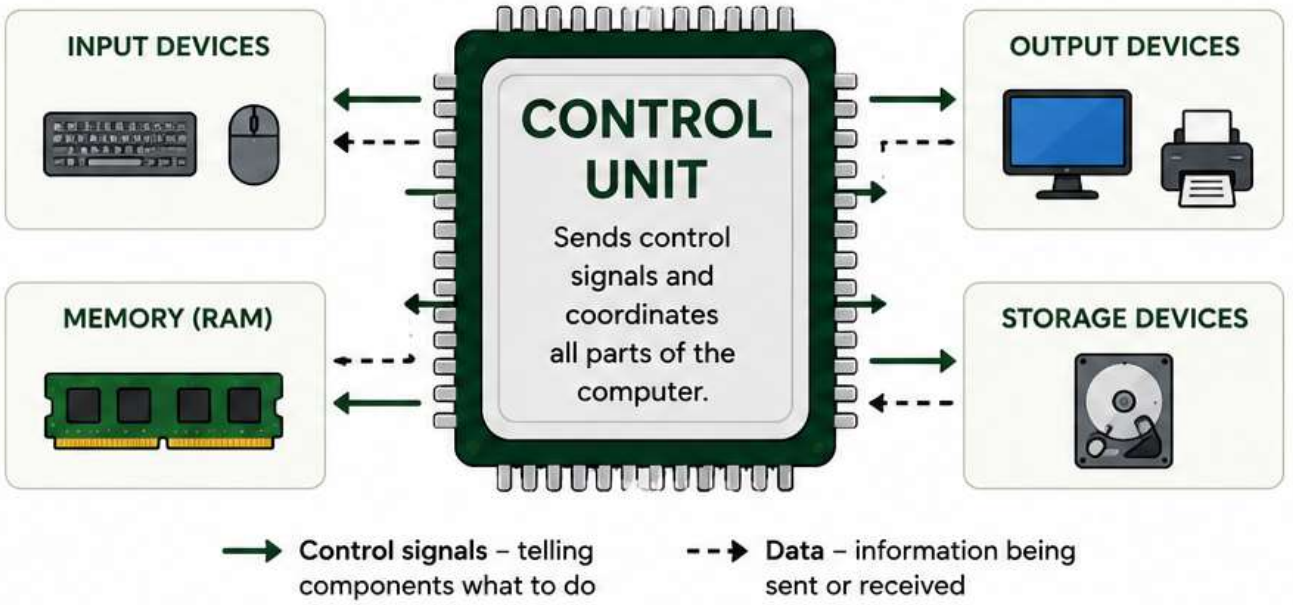
2. Control Unit (CU)

What is it?

- The **Control Unit (CU)** manages and controls the operation of the computer.
- It tells other components **what to do and when to do it**.

It can:

- Control the Fetch-Decode-Execute Cycle.
- Send control signals to memory.
- Communicate with input and output devices.
- Coordinate all hardware components.



★ GCSE Definition

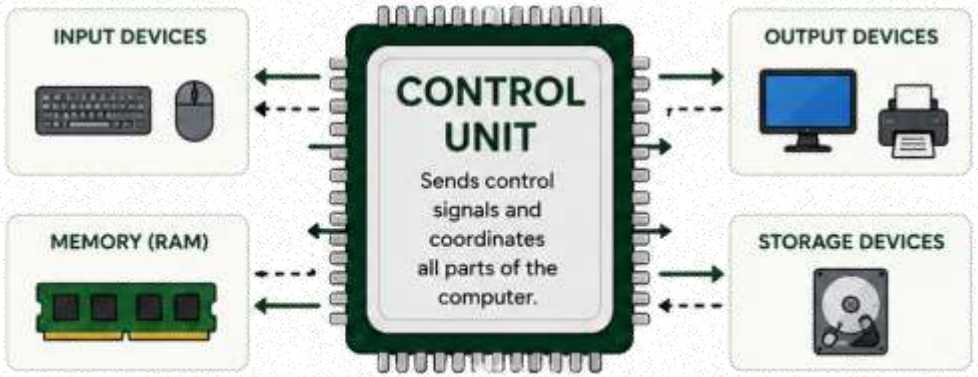
The Control Unit controls the operation of the CPU and coordinates the computer's hardware.

Common CPU Components and Their Functions

2 Control Unit (CU)

The Control Unit (CU) is the part of the CPU that **manages and controls** all activities in the computer.

Its job is to make sure the **right instruction** is carried out, by the **right component**, at the **right time**.



Control signals – telling components what to do

Data – information being sent or received

KEY FUNCTIONS

Controls the Fetch-Decode-Execute Cycle
It ensures each step happens in the correct order.

Sends control signals
Tells other components what to do and when to do it.

Manages data flow
Controls the movement of data between the CPU, memory and I/O devices.

Coordinates all components
Works with the ALU, registers, memory and I/O devices to carry out instructions.

HOW IT WORKS (EXAMPLE)

1. The CU **fetches** the next instruction from memory.
2. It **decodes** the instruction to understand it.
3. It sends control **signals** to the right component.
4. The instruction is **executed** and the cycle repeats.

REAL-LIFE ANALOGY

The Control Unit is like **traffic lights** at a busy junction. It decides who goes next, controls the flow and prevents chaos.

GCSE EXAM POINT

The Control Unit controls the operation of the CPU and coordinates the computer's hardware, ensuring instructions are carried out correctly.

Common CPU Components and Their Functions

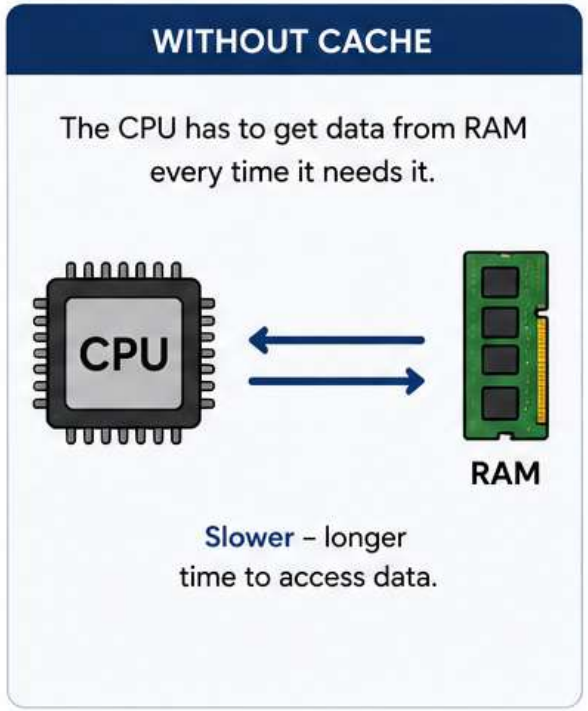
3. Cache Memory

What is it?

- **Cache** is a **very small, very fast memory** located inside or very close to the CPU.
- It stores **frequently used instructions and data** so the CPU can access them quickly.

Why is it needed?

- RAM is fast.
- Cache is **even faster**.
- Instead of going all the way to RAM every time, the CPU first checks the cache.
- If the data is there, it saves time.



★ GCSE Definition

Cache stores frequently used data and instructions so the CPU can access them more quickly.

Common CPU Components and Their Functions

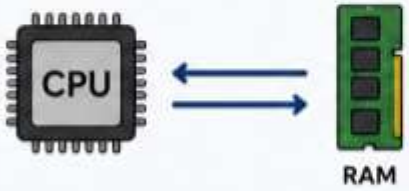
3 Cache Memory

Cache is a **small, high-speed memory** located inside or very close to the CPU.

It stores copies of **frequently used** instructions and data so the CPU can access them much **faster**.

WITHOUT CACHE

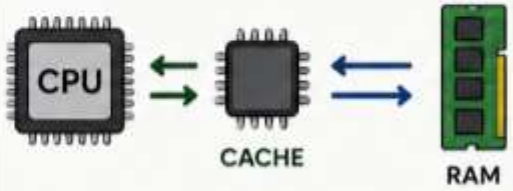
The CPU has to get data from RAM every time it needs it.



Slower – longer time to access data.

WITH CACHE

The CPU first checks the cache. If the data is there, it uses it instantly.



Faster – data is found in the cache.

If not in cache, it is fetched from RAM and stored in cache.

GCSE EXAM POINT

Cache stores frequently used data and instructions so the CPU can access them more quickly.

WHY IS CACHE IMPORTANT?

Increases speed
Reduces the time the CPU spends waiting for data.

Reduces bottlenecks
Less reliance on RAM means smoother performance.

Improves efficiency
Lower power usage because the CPU waits less often.

EXAMPLE

When you **watch a video**, frames you have just seen are stored in cache. If you rewind or replay, the CPU gets them from cache instantly instead of going back to RAM.

COMMON MISCONCEPTION

Cache is not the same as RAM.
Cache is much smaller but much faster.

Common CPU Components and Their Functions

4. Registers

What are they?

- Registers are **tiny storage locations inside the CPU**.
- They temporarily hold:
- instructions
- memory addresses
- data currently being processed

Why are they useful?

- The CPU constantly needs information while carrying out the Fetch-Decode-Execute Cycle.
- Registers keep this information **ready for immediate use**.

WHERE REGISTERS ARE USED

1. **During the Fetch stage**
The address of the next instruction is stored in a register.

2. **During the Decode stage**
The instruction and any data are held in registers.

3. **During the Execute stage**
Data to be processed is stored in registers for quick access.

4. **During the Write-back stage**
Results are stored in registers before being written back to memory.

★ GCSE Definition

Registers are very small storage locations inside the CPU that temporarily hold data and instructions currently being processed.

Common CPU Components and Their Functions

4 Registers

Registers are **very small, high-speed storage** locations **inside** the CPU.

They temporarily hold **data, instructions** and **memory addresses** that the CPU is **currently working** with.

WHERE REGISTERS ARE USED

1. **During the Fetch stage**
   The address of the next instruction is stored in a register.
2. **During the Decode stage**
   The instruction and any data are held in registers.
3. **During the Execute stage**
   Data to be processed is stored in registers for quick access.
4. **During the Write-back stage**
   Results are stored in registers before being written back to memory.

TYPES OF REGISTERS (EXAMPLES)

- **PC** — **Program Counter (PC)**
  Stores the address of the next instruction to be fetched.
- **IR** — **Instruction Register (IR)**
  Stores the current instruction being decoded.
- **MAR** — **Memory Address Register (MAR)**
  Stores the memory address being accessed.
- **MDR** — **Memory Data Register (MDR)**
  Stores data being read from or written to memory.

WHY REGISTERS ARE IMPORTANT

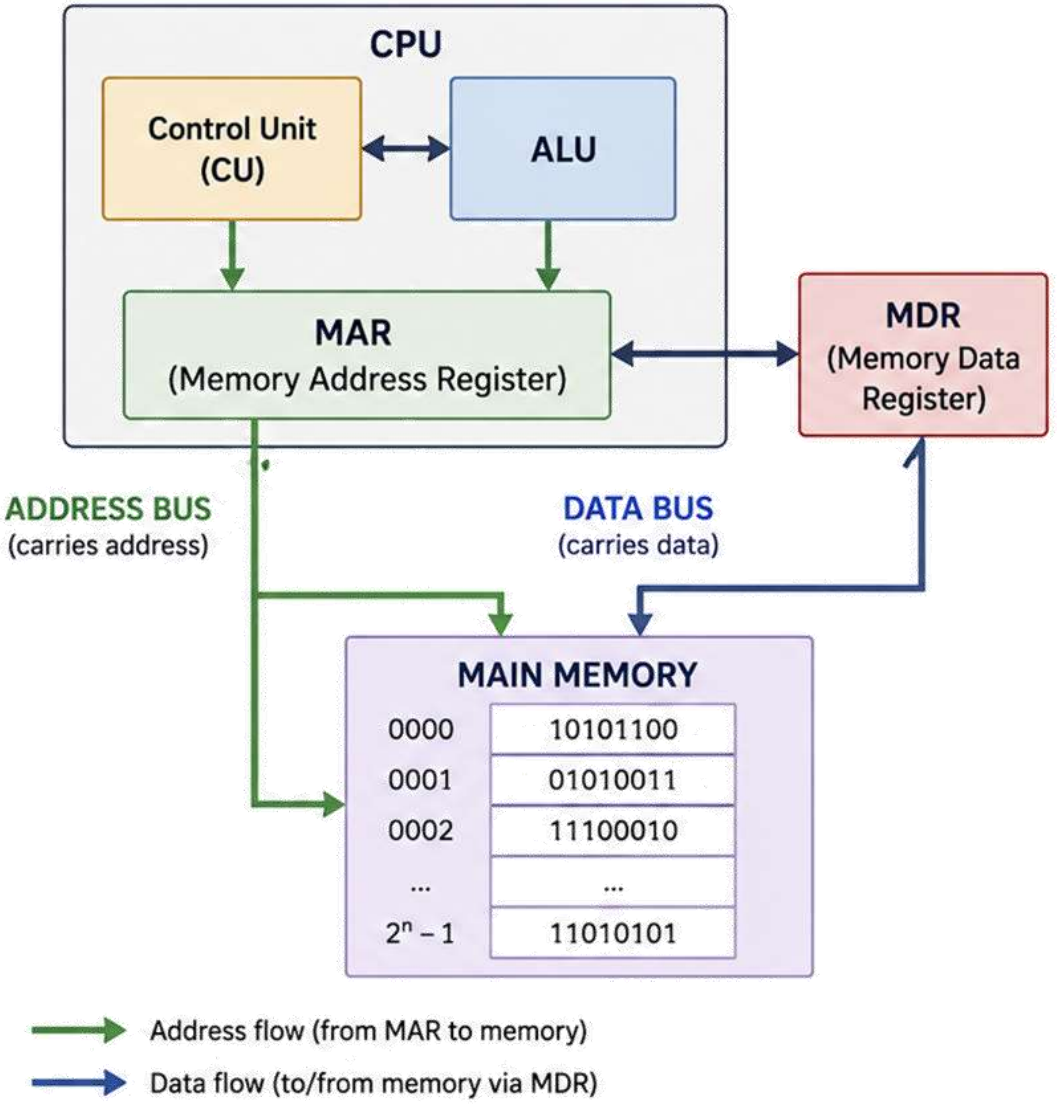
- **Extremely fast**
  Faster than cache and RAM, so the CPU can work quickly.
- **Reduce delays**
  Keep the data the CPU needs right at hand.
- **Improve performance**
  Essential for the CPU to process instructions efficiently.

Von Neumann Architecture Registers

Explanation

The CPU contains several **special-purpose registers**. Registers are tiny storage locations inside the CPU that hold data or instructions while they are being processed. They are much faster than RAM.

1. **MAR (Memory Address Register)**
2. **MDR (Memory Data Register)**
3. **Program Counter (PC)**
4. **Accumulator (ACC)**



Von Neumann Architecture Registers

Memory Address Register (MAR)

What is it?

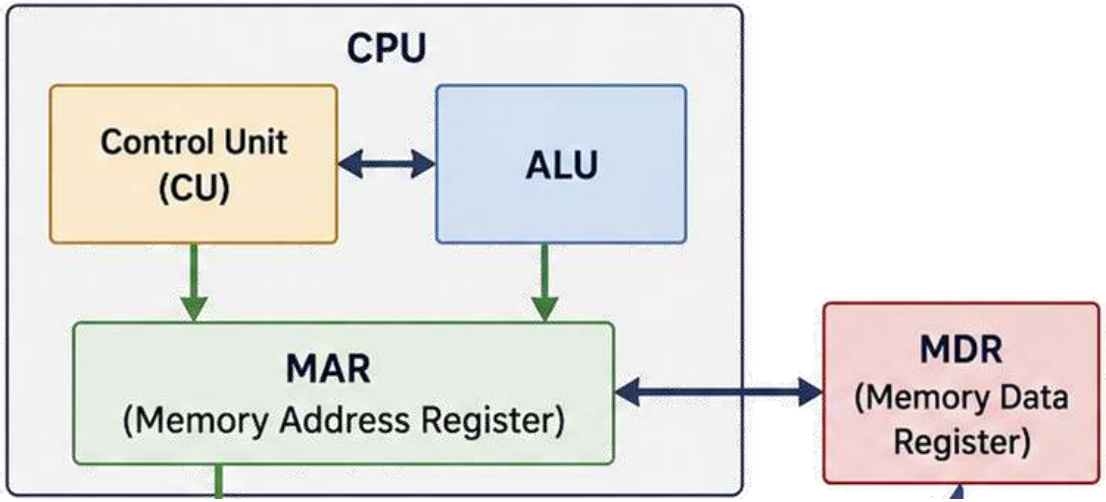
- The **Memory Address Register (MAR)** stores the **address of the memory location** that the CPU wants to access.
- It does **not** store the actual data.
- Instead, it stores **where** the data can be found.

During the Fetch-Decode-Execute Cycle

- The Program Counter sends the address of the next instruction to the MAR.
- The MAR then tells the memory which location to access.

★ GCSE Definition

The MAR stores the memory address of the data or instruction that the CPU wants to access.



Example

- Imagine you want to read a book from a library.
- The **MAR is like writing down the shelf number** where the book is located.
- You still need someone to go and collect the book.

Von Neumann Architecture Registers

MAR (Memory Address Register)

The MAR holds the address of the memory location that the CPU wants to access.

PURPOSE

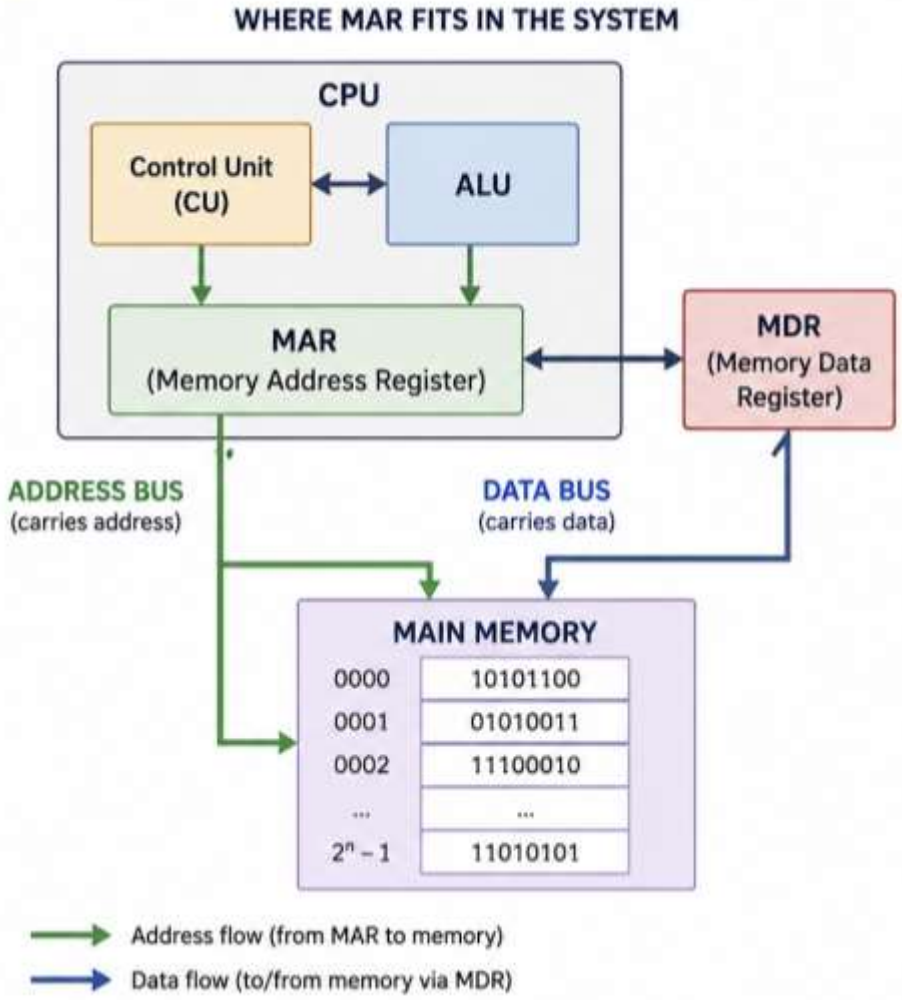
Stores the address of the memory location from which data will be read or to which data will be written.

KEY POINTS

- Holds an address, not the actual data.
- Sends the address to main memory.
- The number of bits in the MAR determines the maximum addressable memory.
- Part of the Von Neumann architecture.

EXAMPLE

If the MAR contains 0000000000101010_2, this means the CPU wants to access the memory location at address 42_{10}.



Von Neumann Architecture Registers

MDR (Memory Data Register)

What is it?

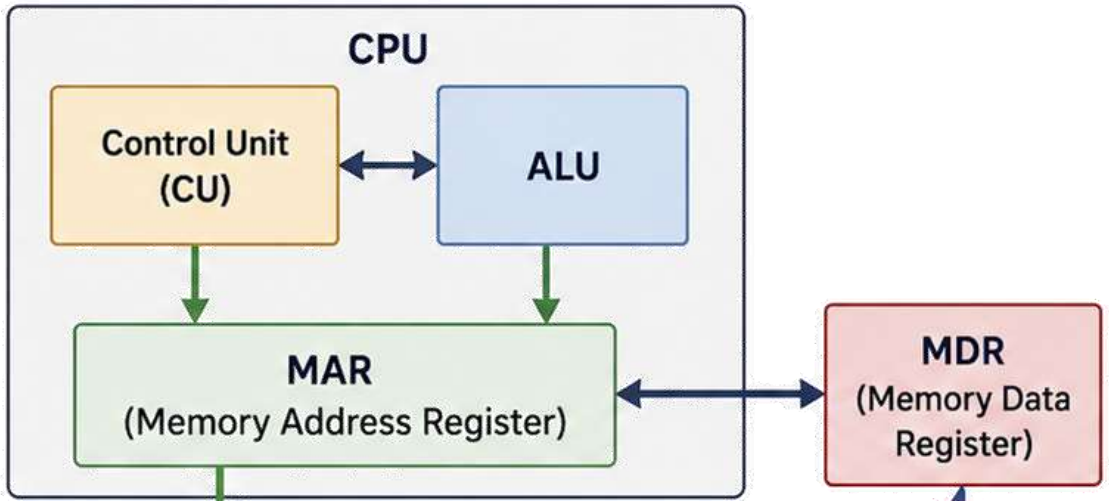
- The **Memory Data Register (MDR)** stores the **actual data or instruction** that is being transferred between the CPU and main memory.
- Unlike the MAR, the MDR stores the contents, not the address.

During the Fetch-Decode-Execute Cycle

- Once memory finds the correct address, it sends the instruction into the MDR.
- The MDR then passes it into the CPU.

★ GCSE Definition

The MDR stores the data or instruction being transferred to or from memory..



Example

Using the library example:

- The MAR knows the shelf number.
- The **MDR is the actual book** that has been collected from the shelf.

MAR (Memory Address Register)

The MAR holds the address of the memory location that the CPU wants to access.

PURPOSE

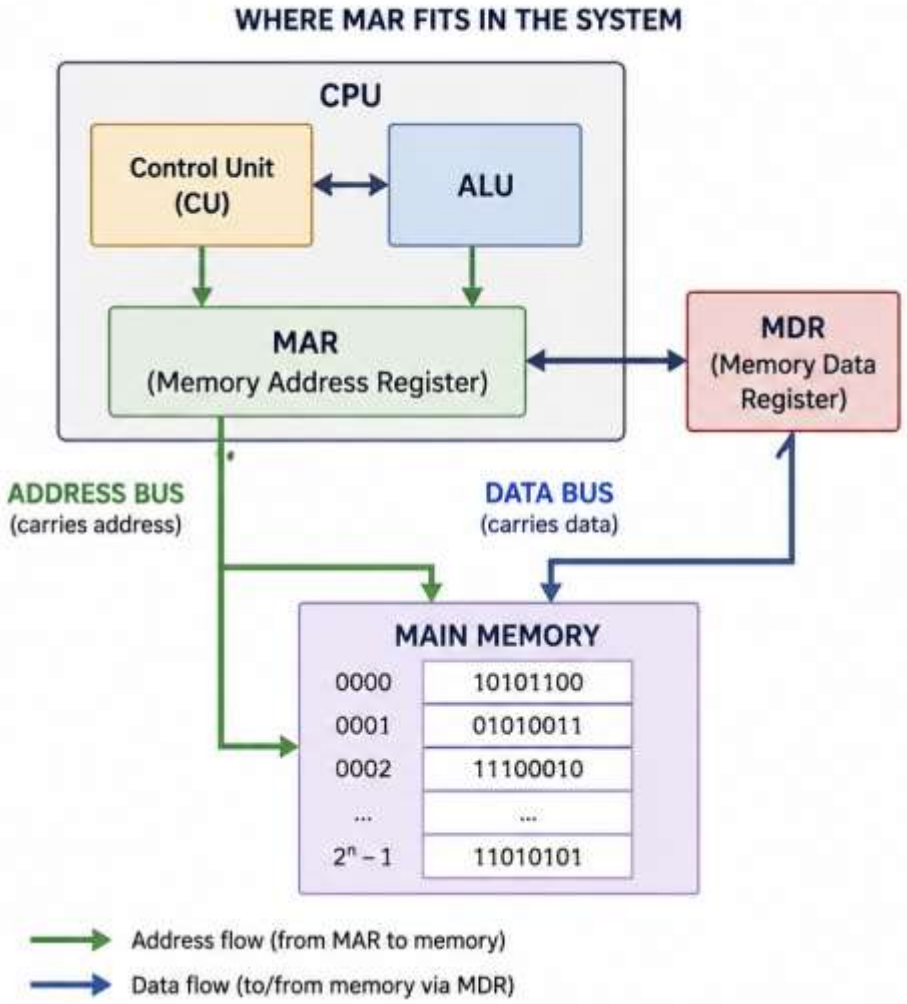
Stores the address of the memory location from which data will be read or to which data will be written.

KEY POINTS

- Holds an address, not the actual data.
- Sends the address to main memory.
- The number of bits in the MAR determines the maximum addressable memory.
- Part of the Von Neumann architecture.

EXAMPLE

If the MAR contains 0000000000101010_2, this means the CPU wants to access the memory location at address 42_{10}.



Von Neumann Architecture Registers

Program Counter (PC)

What is it?

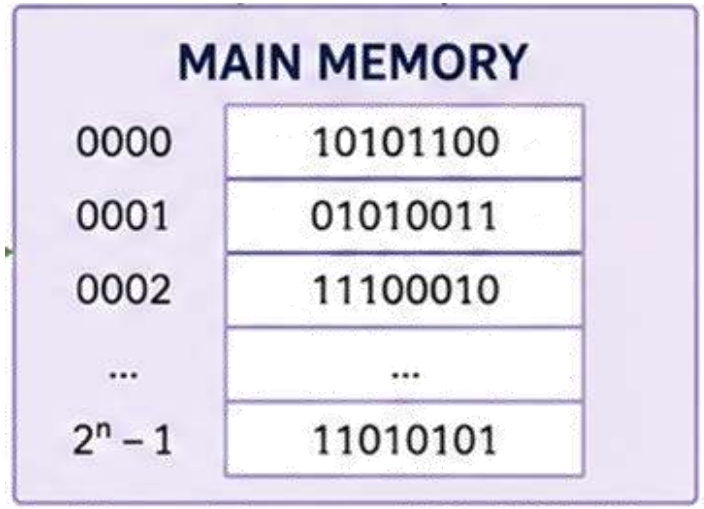
- The **Program Counter (PC)** stores the **address of the next instruction** that the CPU needs to fetch.
- After each instruction has been fetched, the Program Counter is updated so it points to the next instruction.

During the Fetch-Decode-Execute Cycle

- Holds the address of the next instruction.
- Copies the address into the MAR.
- Increases by one so it points to the next instruction.

★ GCSE Definition

The Program Counter stores the address of the next instruction to be fetched.



Example

- Imagine following a recipe.
- The Program Counter is like your finger pointing to the next step you need to read.
- Once you finish that step, your finger moves to the next one.

Von Neumann Architecture Registers

Program Counter (PC)

The Program Counter stores the address of the next instruction that the CPU will fetch from memory.

PURPOSE

- Holds the memory address of the next instruction to be fetched.
- Automatically increases after each instruction is fetched.

KEY POINTS

- Points to the next instruction in the program.
- After an instruction is fetched, the PC is updated (usually increased by 1).
- If a jump or branch occurs, the PC is set to a new address.
- Part of the Von Neumann architecture.

EXAMPLE

If the PC contains 000000000001010_2 (which is 10_{10}), the CPU will fetch the instruction stored at memory address 10.

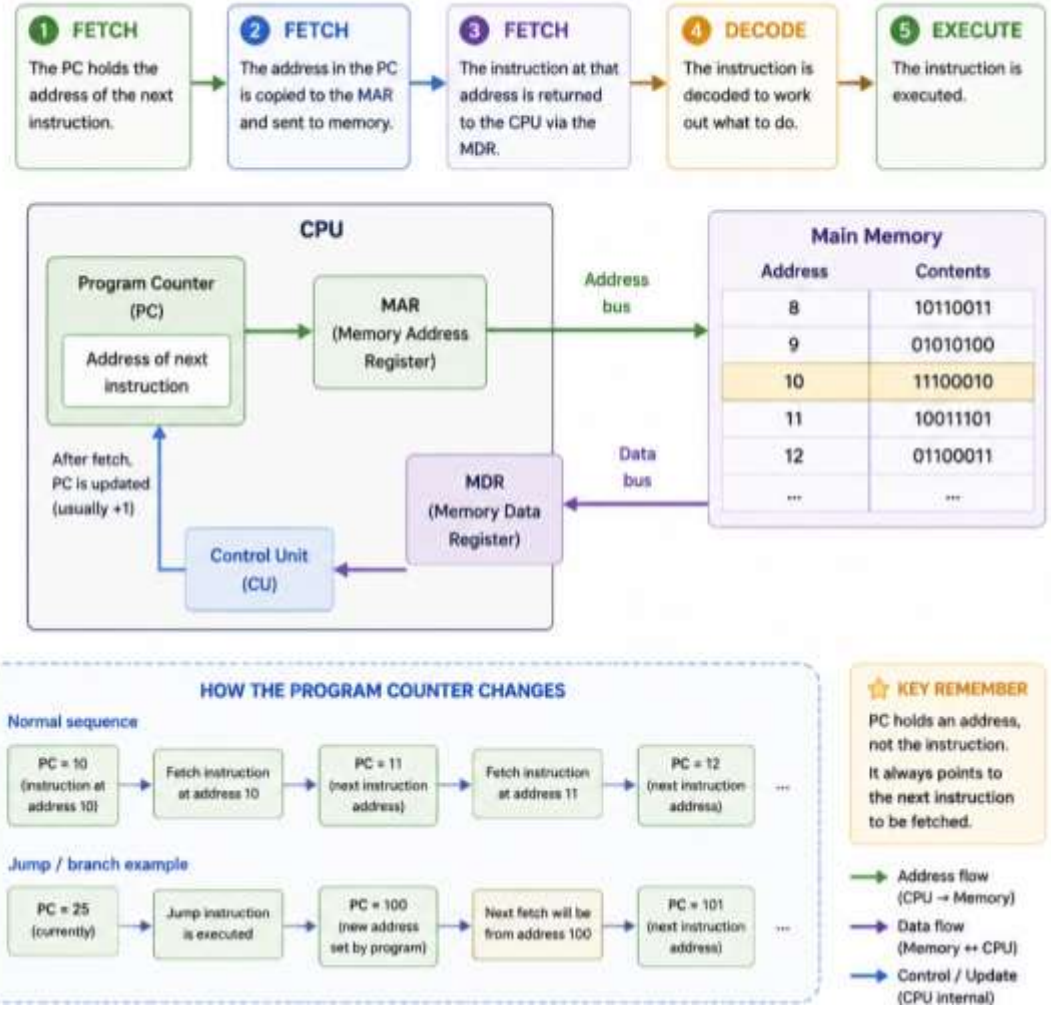
After fetching, the PC is updated to 000000000001011_2 (which is 11_{10}) to point to the next instruction.

EXAM TIP

Think: PC = Points to the next instruction!

It does not store the instruction itself, only the address.

WHERE THE PROGRAM COUNTER FITS IN THE FETCH–DECODE–EXECUTE CYCLE



Von Neumann Architecture Registers

Accumulator (ACC)

What is it?

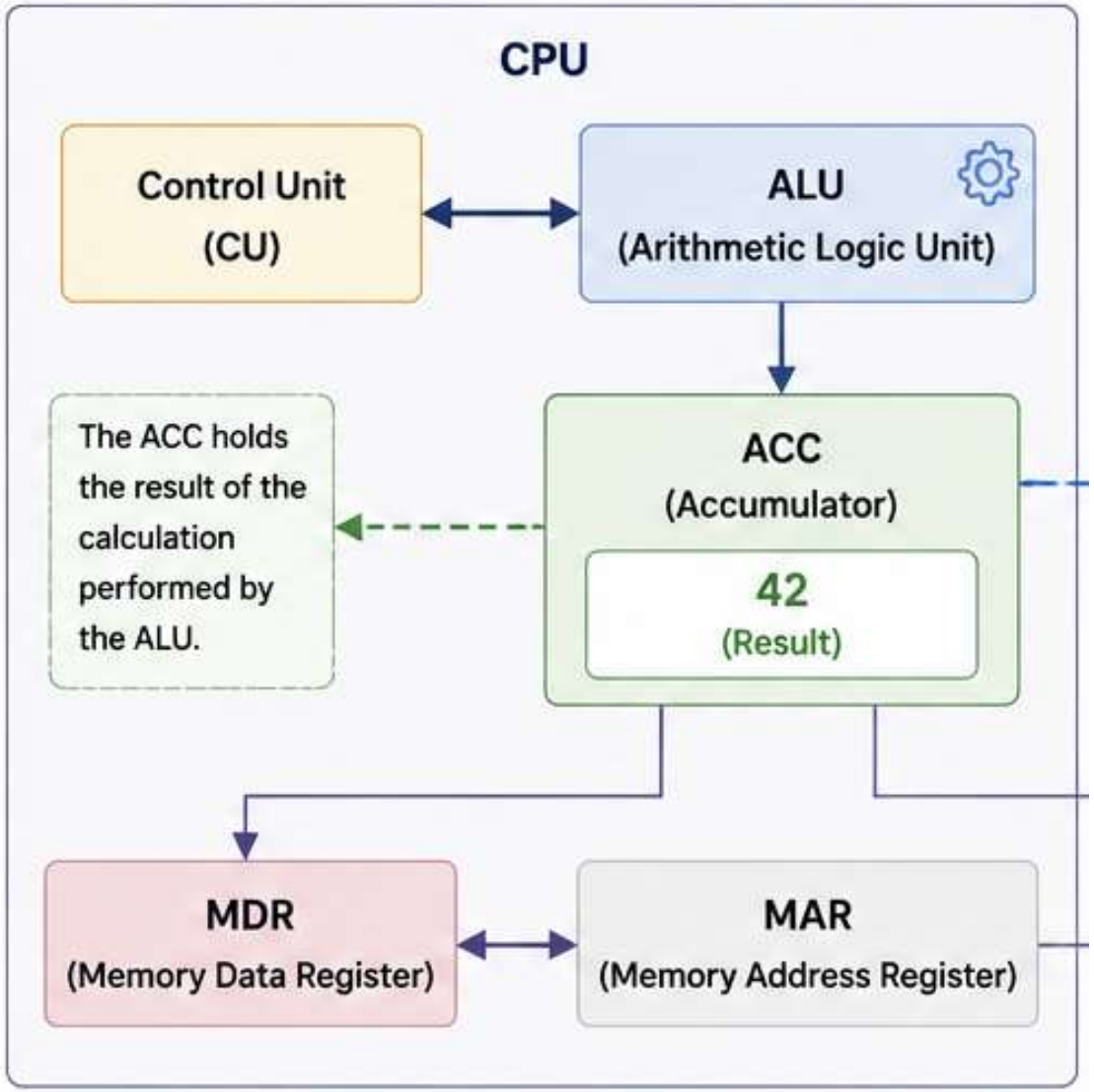
- The **Accumulator** stores the **results of calculations** carried out by the Arithmetic Logic Unit (ALU).
- It acts as temporary storage while calculations are being completed.

During the Fetch-Decode-Execute Cycle

- The ALU performs calculations.
- The result is placed into the Accumulator before it is stored elsewhere if required.

★ GCSE Definition

The Program Counter stores the address of the next instruction to be fetched.



Von Neumann Architecture Registers

ACC (Accumulator)

The Accumulator stores the results of calculations performed by the ALU. It acts as temporary storage while data is being processed.

PURPOSE

- Holds the results of arithmetic and logical operations.
- Stores data temporarily before it is written back to memory or used again.
- Works closely with the ALU.

KEY POINTS

- The ALU performs calculations.
- The result is placed in the Accumulator.
- The result can then be used in further calculations or saved to memory.
- Part of the Von Neumann architecture.

EXAMPLE

If the CPU needs to calculate 25 + 17:

1. The values 25 and 17 are sent to the ALU.
2. The ALU adds them together.
3. The result (42) is stored in the Accumulator.
4. The result can then be used again or written to memory.

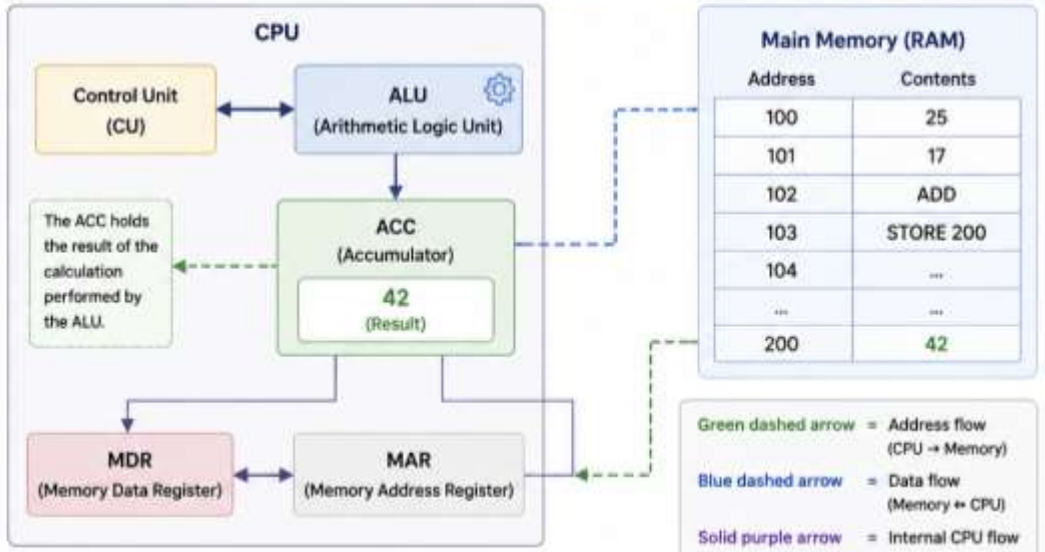
EXAM TIP

Think: ACC = Calculation results

It is like a temporary holding place for answers.

ACC IN THE FETCH-DECODE-EXECUTE CYCLE

1. **FETCH** — The next instruction is fetched from memory.
2. **DECODE** — The instruction is decoded to find out what to do.
3. **EXECUTE** — The ALU performs the calculation or operation.
4. **STORE RESULT** — The result of the operation is stored in the Accumulator.
5. **NEXT** — The result can be used again or saved to memory.



WORKED EXAMPLE

1. Values taken from memory: 25 + 17 (from addresses 100 and 101)
2. ALU adds the values: 25 + 17
3. Result stored in ACC: 42
4. Result can be used again or stored in memory: 42 → 200

REAL LIFE ANALOGY

The Accumulator is like the display on a calculator. The calculator (ALU) does the calculation and shows the answer on the display (Accumulator) until you use it again.

SUMMARY: The Accumulator stores the results of calculations from the ALU. It is temporary storage used during processing.

Lesson Summary

Today you have learned to:

- Describe the purpose of the CPU and explain its role in running programs.
- Explain the stages of the Fetch-Decode-Execute Cycle and how instructions are processed by the CPU.
- Identify the main components of the CPU (ALU, Control Unit, Cache and Registers) and describe the function of each.
- Explain the principles of the Von Neumann architecture and why it is used in modern computers.
- Describe the purpose of the Memory Address Register (MAR).
- Describe the purpose of the Memory Data Register (MDR).
- Explain the role of the Program Counter (PC).
- Describe the purpose of the Accumulator (ACC).
- Apply your knowledge of CPU components and registers to explain how data and instructions move through the CPU during program execution.

KNIGHTON GRANGE
EDUCATION

Thank you for using this resource.

Continue your learning at:
www.knightongrange.co.uk