# OCR Computer Science GCSE

## 1.1 – Systems architecture

Advanced Notes

# 1.1.1 Architecture of the CPU

## Purpose of the CPU

At the heart of every computer is a Central Processing Unit (CPU) which executes instructions in order to run programs. Processors contain an arithmetic logic unit, a control unit and numerous registers.

## Major CPU Components

| Component | Function |
|---|---|
| Arithmetic Logic Unit (ALU) | Performs mathematical calculations and logical operations as required. |
| Control Unit (CU) | The control unit marshals and controls the operation of the fetch-execute cycle, coordinating the operation of the CPU and sending commands to other components - for instance, requesting that the arithmetic logic unit perform a calculation. |
| Cache | Cache is a small, fast memory device located on the CPU that stores frequently used data and instructions. |

## Registers

Registers are fast-to-access storage locations, used to store small amounts of data needed temporarily by the CPU during processing.

| Register | Function |
|---|---|
| MAR (Memory Address Register) | Stores the address of the data to be fetched from or the address where the data is to be stored. |
| MDR (Memory Data Register) | Stores the data that is being fetched from or written to memory. It acts as a buffer between main memory and the CPU. |
| Program counter | Stores the address of the next instruction to be fetched from memory. It increments during each fetch-execute cycle to point to the next instruction. |
| Accumulator | Stores the results of calculations or operations carried out by the Arithmetic Logic Unit (ALU). Also temporarily holds data being processed. |

*Note: data is the actual value or instruction being stored, used, or processed whereas an address is a location in memory where data is stored.*

## The Fetch-Execute Cycle

The fetch-execute cycle is a continuous cycle performed by the processor. This cycle repeats millions of times per second; it has the following stages:

- The address of the next instruction to be fetched is transferred from the Program Counter to the Memory Address Register (MAR)
- The instruction is fetched from memory and copied into the Memory Data Register (MDR)
- The Program Counter is incremented
- The instruction is decoded by the Control Unit (CU)
- The decoded instruction is then executed by the CPU or ALU
- The process repeats for the next instruction so that the CPU performs continuously

## Von Neumann Architecture

Most modern CPUs follow the Von Neumann model, where instructions and data are stored in the same memory.

# 1.1.2 CPU performance

## CPU performance factors

| Factor | Description |
|---|---|
| Clock speed | The clock is a device that sends a regular electrical signal which switches between low and high voltage at a regular frequency. This signal is used to synchronise the computer system's components; it controls the number of instructions carried out each second.<br><br>With every tick of the clock, the CPU fetches and executes one instruction. The greater the clock speed, the faster the CPU can execute instructions, improving performance. |
| Cache size | A larger cache increases the amount of frequently used data that can be stored. This reduces the need for the CPU to access slower main memory (RAM), improving performance. |
| Number of cores | Cores are the individual processing units within a CPU. A CPU with more cores can process more instructions at once allowing it to handle multiple tasks simultaneously, making it faster and improving performance. However, not all programs are designed to use multiple cores, so performance may not always improve. |

## 1.1.3 Embedded systems

An embedded system is a computer system that is designed to perform specific, dedicated functions within a larger mechanical or electronic system. It is "embedded" into a device to control particular operations of that device.

Characteristics of embedded systems include:

- Designed for one specific task or set of related tasks
- Built into other devices and cannot easily be separated
- Have minimal or no user interface
- Optimised for efficiency and reliability
- Typically low power, small, and efficient

**Examples**

- Washing machines - embedded systems can control water temperature, cycle timing and motor speed
- Microwave ovens - embedded systems can manage cooking time, power levels and safety features
- Traffic lights - embedded systems can control timing sequences and respond to sensors