# Computer architecture

**Computer architecture** refers to the structure and organisation of a **computer system**. It specifies the **components** that make up a computer system and describes how these are **interconnected**, how they interact with each other, and how they are managed.

All stages **A brief history of computer architecture**

In the early days of computer systems, programming was performed by manually setting the position of a large number of switches and plugs, and then entering the input data. The output was produced by determining the position of some of the switches.

A prime example of a powerful early machine is Colossus, a set of computers developed from 1943 to 1945 to decipher the enemy coded messages during the second world war.

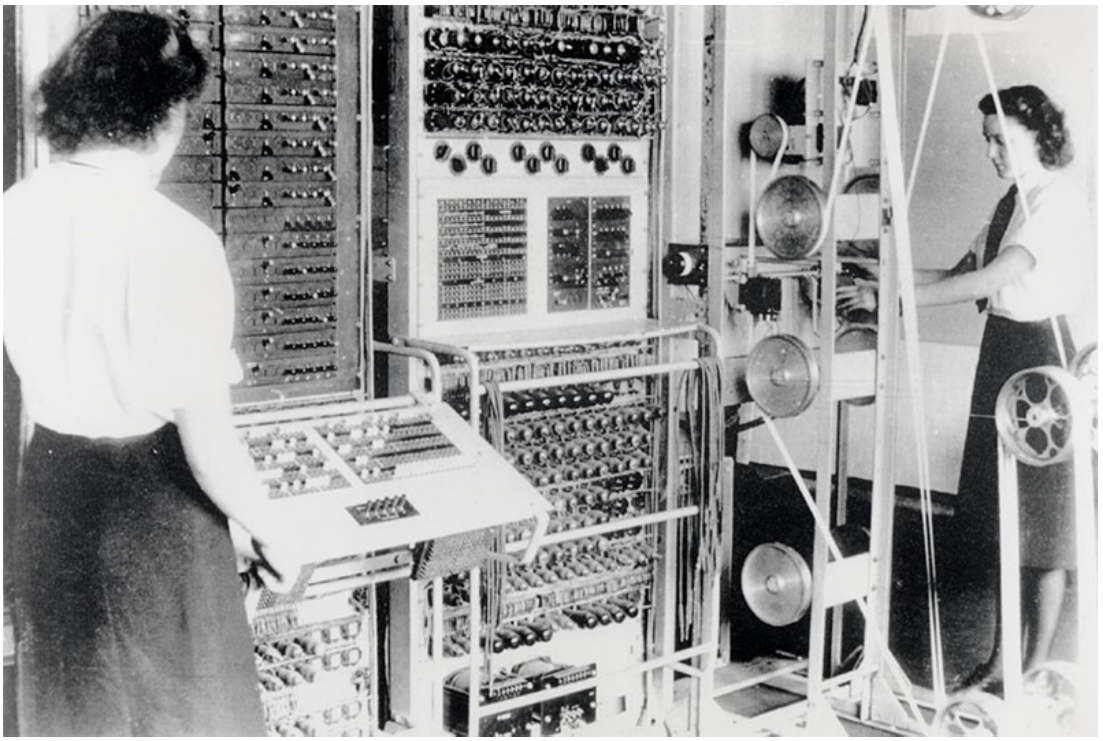

**Figure 1:** Dorothy Du Boisson *(left)* and Elsie Booker *(right)* operating the Colossus computer in 1943
The United Kingdom National Archives, document record FO850/234, Public Domain via Wikimedia Commons

One of the main issues of this approach was that reprogramming through setting switches was very inefficient and time consuming; every time a new calculation was programmed, the places of the switches changed and so the previous program was lost. Also, the computer system had to be reprogrammed for every new set of input data, even if it was to be used by the same calculation. Therefore, from the very early days, people sought techniques that would allow programs and data to be read from an easy-to-access source.

## The stored program concept

These issues were tackled by a new model for computer systems, which introduced a **memory unit** that could store **data** as well as **programs**. The concept of having a **stored program**, i.e. one that could be loaded into working (electrical) memory, was a major step forward. It allowed for instructions to be fetched from the **memory unit** and executed, one after the other, by a **processing unit** that was designed to perform arithmetic and logical operations. This meant that machines could be more **versatile**; different programs could be loaded onto the same machine architecture and the machine would perform different tasks. The stored program concept paved the way for the creation of **general-purpose computers** such as modern-day smartphones, tablets, laptops, and desktop computers.

The first electronic computer that applied the stored program concept was the **Manchester Baby**, a computer made by the University of Manchester in 1948, which included the first type of random access memory (RAM).

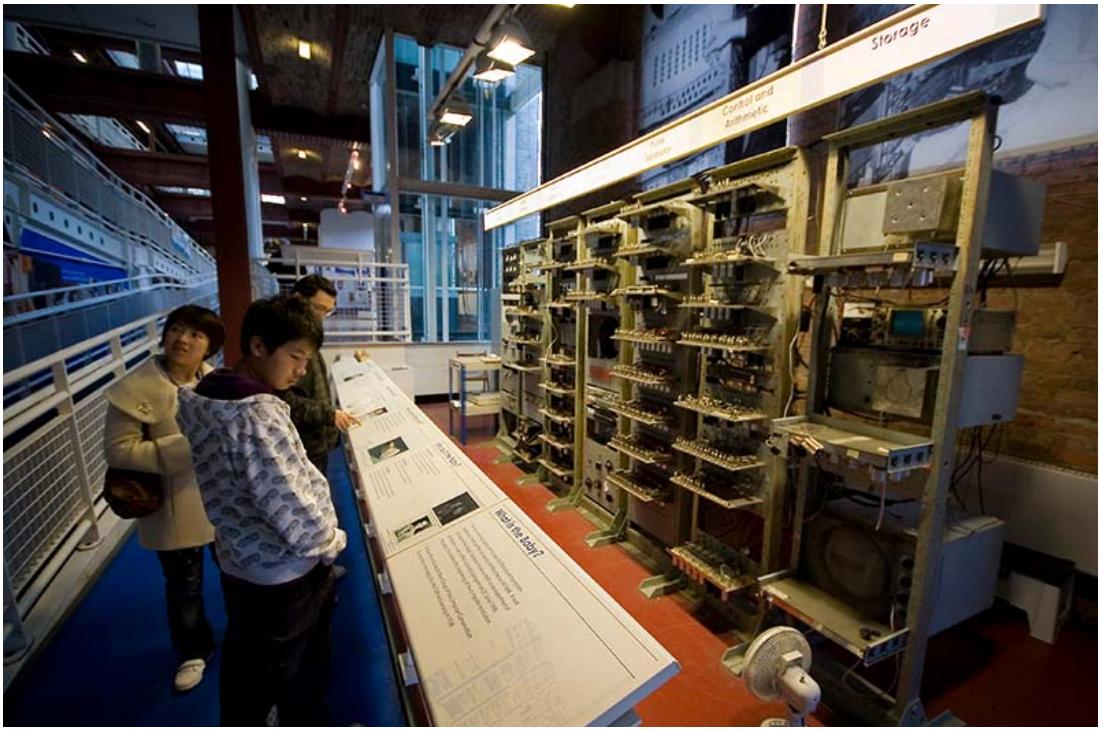


**Figure 2:** Replica of the Manchester Baby, on display at the Science and Industry Museum, Manchester
Parrot of Doom via Wikimedia Commons, CC BY-SA 3.0

Later on, the first electronic general-purpose digital computer was introduced. ENIAC was initially presented as a project from the University of Pennsylvania in 1946; it was improved to use a magnetic type of memory in 1953.

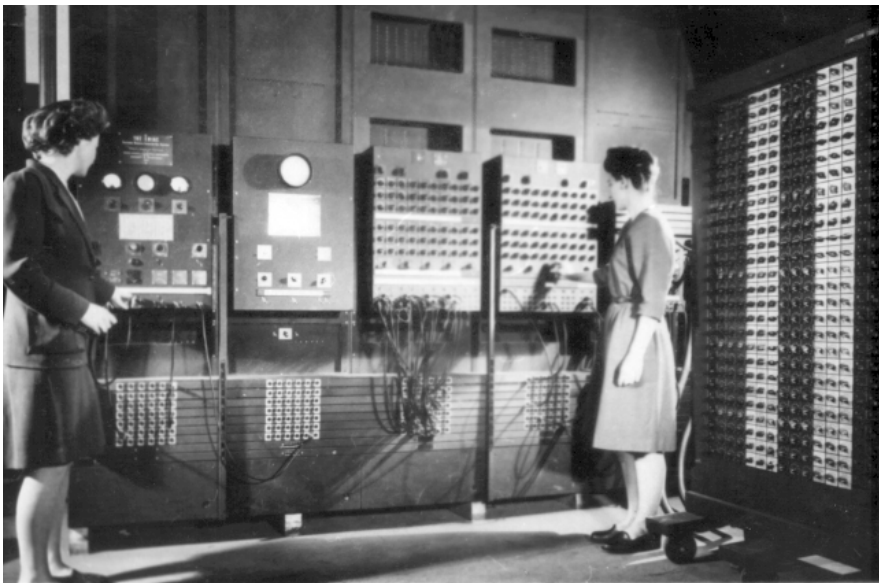

**Figure 3:** Programmers Betty Jean Jennings *(left)* and Fran Bilas *(right)* operating ENIAC's main control panel, c.1945
US Army photo from the archives of the ARL Technical Library, Public Domain, via Wikimedia Commons

All stages **Von Neumann architecture**

In the 1940s, **John von Neumann** and his team developed the concept of the stored program computer. The Von Neumann architecture used the idea of **storing program instructions and data** in **main memory** and moving them between memory and the processor. **Von Neumann architecture** is used in many modern-day computer systems.

**The von Neumann architecture** consists of

- a Processor
- a Memory unit that can communicate directly with the processor
- connections for input and output devices
- Secondary storage for saving/backing up data

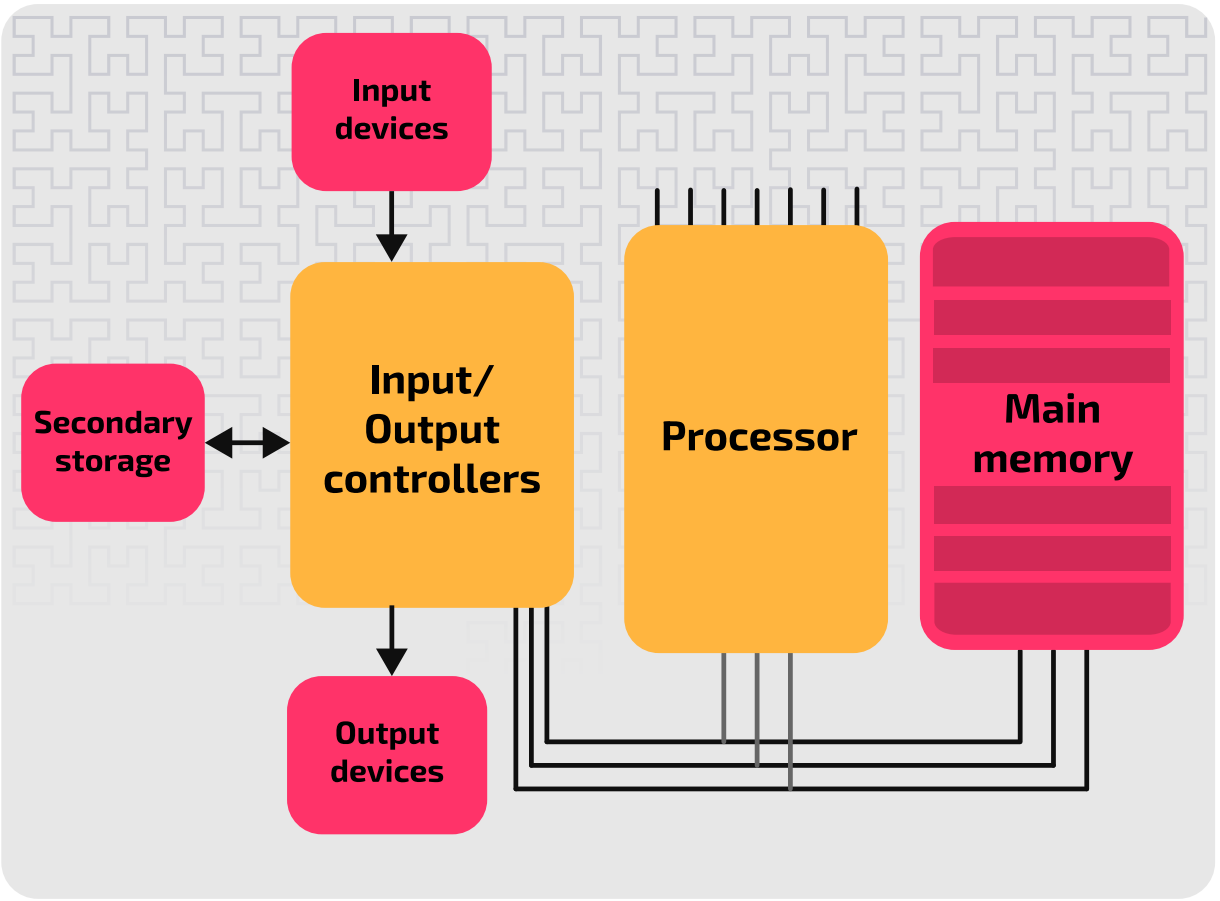


**Figure 4:** Von Neumann architecture model

The processor can access the instructions and data in the main memory as required to execute the program. It does this by using dedicated connections called **buses**:

- an **address bus** is used to identify the addressed location
- a **data bus** is used to transfer the contents to/from that location

This means that **the same address and data buses** are used in the process of transferring **instructions and data** between main memory and the processor. A third bus, the **control bus**, is used to synchronise and control operations.

A Level Harvard architecture

On the other hand, **Harvard architecture** keeps instructions and data **in separate memories**. The processor accesses these memories **using separate data and address buses**; the processor is connected to the 'instructions memory' using a dedicated set of address and data buses, and is connected to the 'data memory' using a **different** set of address and data buses. The Harvard architecture is used extensively in embedded systems, for example in digital signal processing (DSP) systems.

Many microcontroller devices use a Harvard-like architecture. Some types of smartphones use a modified Harvard architecture — although distinctions between modern processors are difficult to categorise in relation to such 'pure' theoretical models.

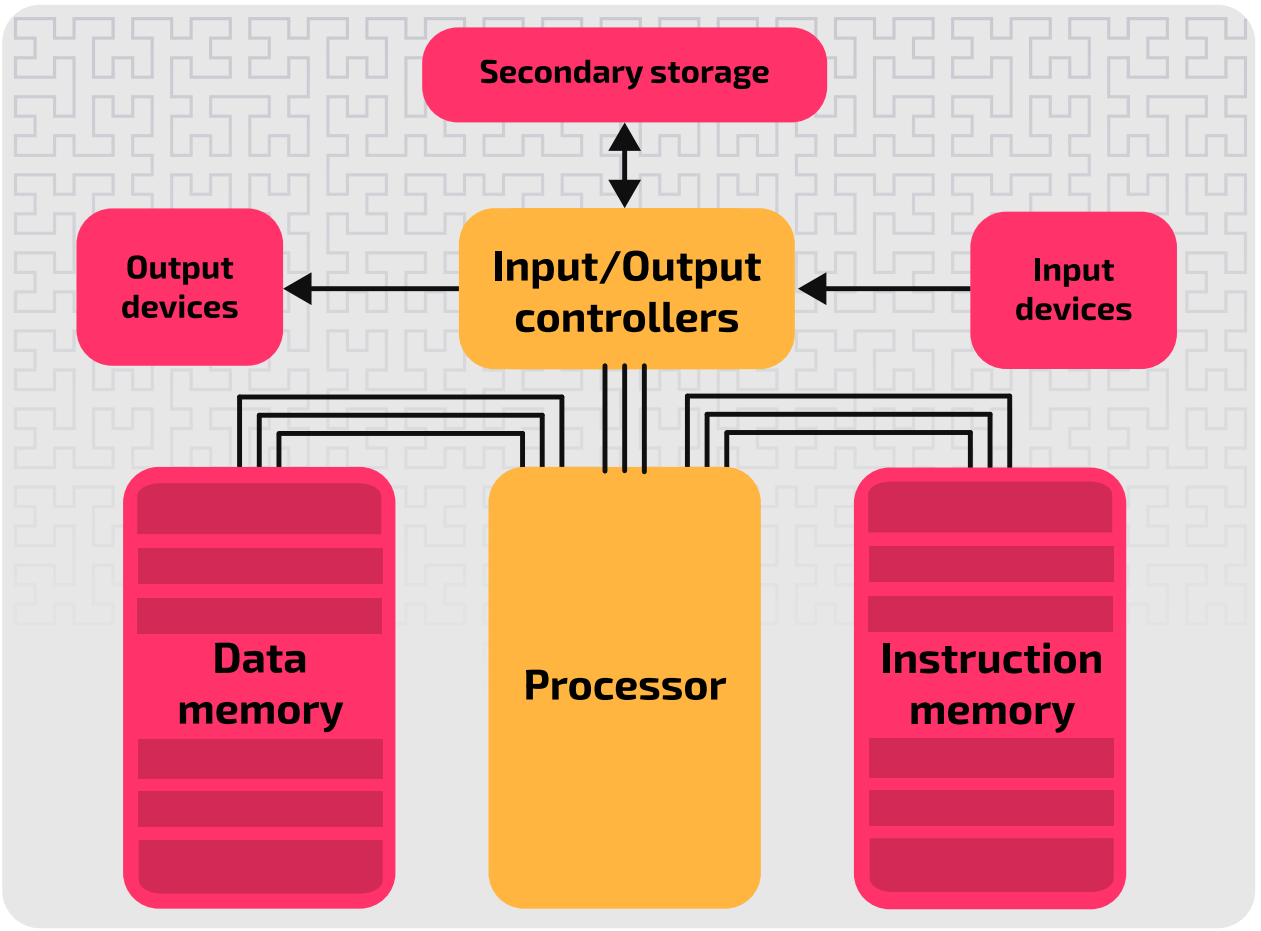


**Figure 5:** Harvard architecture model

A Level **Comparison between von Neumann and Harvard architecture**

Harvard architecture is characterised by the use of separate memory units and buses for instructions and data, which means that both memories can be accessed simultaneously. This minimises the issue of keeping the processor waiting while loading or saving data into memory, which in turn increases the processor performance. Von Neumann architecture uses the same address and data buses for both instructions and data, which means that both instructions and data share the same pathways.

Moreover, with Harvard architecture, each memory can be adapted to meet the needs of a particular system: the instruction and data memories can be different sizes, different word lengths, or implemented using a different type of technology. For example, for systems with a predetermined use (such as embedded systems), the instruction memory can be implemented as a read-only memory (ROM), which protects the programs from accidental or deliberate changes by hacking. On the other hand, von Neumann architecture allows for the instructions and data to be saved in the same memory, which can be exploited by hackers who could disguise instructions (malware) as data that the processor may execute unknowingly when attempting to read the data.

Therefore, Harvard architecture is commonly used in embedded systems, i.e. computer systems that are designed purposefully to perform a specific set of operations and are often used in situations where the speed of operation is very important. For example, embedded systems such as traffic-control cameras are required to process large amounts of data in real time.

Nonetheless, von Neumann architecture enables a more flexible use of the main memory, which allows the processor to run a variety of programs that are not known in advance. Therefore, von Neumann architecture is commonly used in general-purpose computers that are expected to accommodate the varying needs of the end users, for example, run numerous applications and switch between different tasks.