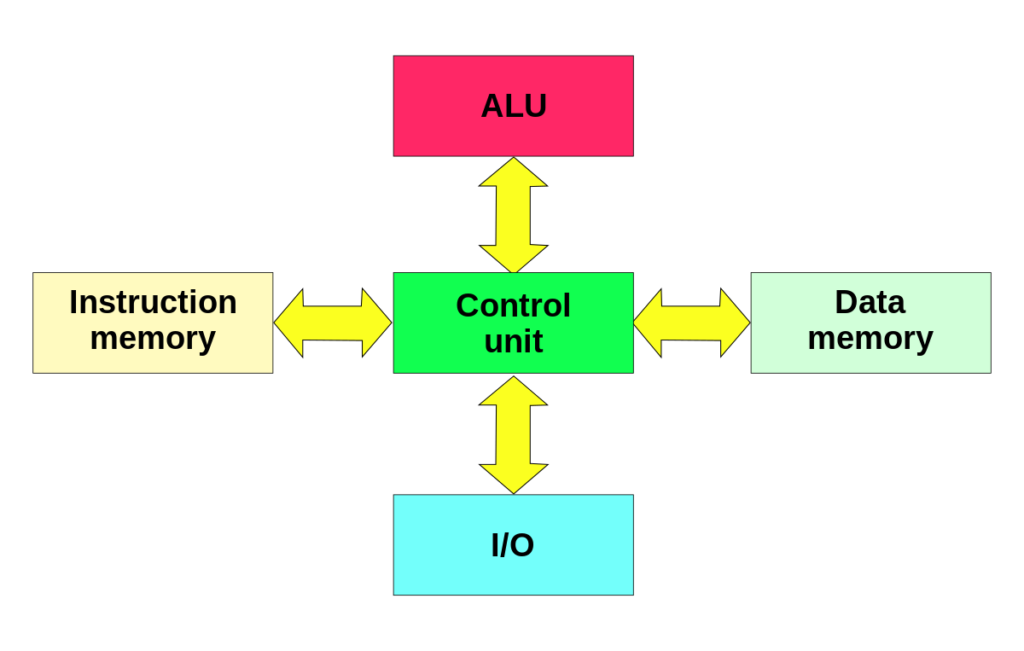
**Von-Neumann architecture**

In a Von-Neumann architecture, the same memory and bus are used to store both data and instructions that run the program. Since you cannot access program memory and data memory simultaneously, the Von Neumann architecture is susceptible to bottlenecks and system performance is affected.



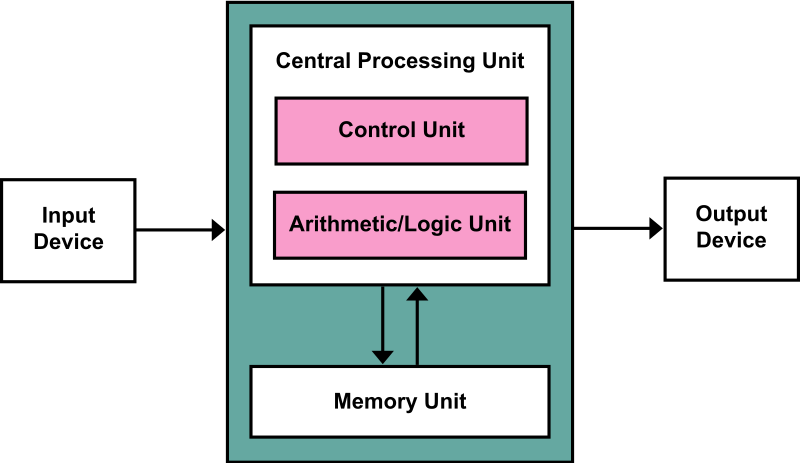
**Figure 2: The Harvard architecture has a separate bus for signals and storage. (Image: Wikimedia Commons)**

**Harvard Architecture**

The Harvard architecture stores machine instructions and data in separate memory units that are connected by different busses. In this case, there are at least two memory address spaces to work with, so there is a memory register for machine instructions and another memory register for data. Computers designed with the Harvard architecture are able to run a program and access data independently, and therefore simultaneously. Harvard architecture has a strict separation between data and code. Thus, Harvard architecture is more complicated but separate pipelines remove the bottleneck that Von Neumann creates.

**Modified Harvard Architecture**

The majority of modern computers have no physical separation between the memory spaces used by both data and programs/code/machine instructions, and therefore could be described technically as Von Neumann for this reason. However, the better way to represent the majority of modern computers is a “modified Harvard architecture.” Modern processors might share memory but have mechanisms like special instructions that keep data from being mistaken for code. Some call this “modified Harvard architecture.” However, modified Harvard architecture does have two separate *pathways (busses)* for signal (code) and storage (memory), while the memory itself is one shared, physical piece. The memory controller is where the modification is seated, since it handles the memory and how it is used.



**Figure 1: The Von Neumann architecture has been around since the 1940s. A clarifying trait is that a single bus used for both signal and storage. (Image: Wikimedia Commons)**

**The Von Neumann Bottleneck**

If a Von Neumann machine wants to perform an operation on some data in memory, it has to move the data across the bus into the CPU. When the computation is done, it needs to move outputs of the computation to memory across the same bus. The amount of data the bus can transfer at one time (speed and bandwidth) plays a large part in how fast the Von Neumann architecture can be. The throughput of a computer is related to how false the processors are as well as the rate of data transfer across the bus. The processor can be idle while waiting for a memory fetch, or it can perform something called speculative processing, based on what the processor might next need to do after the current computation is finished (once data is fetched and computations are performed).

The Von Neumann bottleneck occurs when data taken in or out of memory must wait while the current memory operation is completed. That is, if the processor just completed a computation and is ready to perform the next, it has to write the finished computation into memory (which occupies the bus) before it can fetch new data out of memory (which also uses the bus). The Von Neumann bottleneck has increased over time because processors have improved in speed while memory has not progressed as fast. Some techniques to reduce the impact of the bottleneck are to keep memory in cache to minimize data movement, hardware acceleration, and speculative execution. It is interesting to note that speculative execution is the conduit for one of the latest security flaws discovered by Google Project Zero, named Spectre.