

The Von Neumann Model.

John von Neumann, along with others, proposed the concept of the stored program that we use even today. The idea was to keep a program in the memory and read the instructions from it. He also proposed an architecture that clearly identified the components we have presented previously: ALU, control, input, output, and memory as illustrated in Figure 1. This architecture is known as the *von Neumann architecture*.

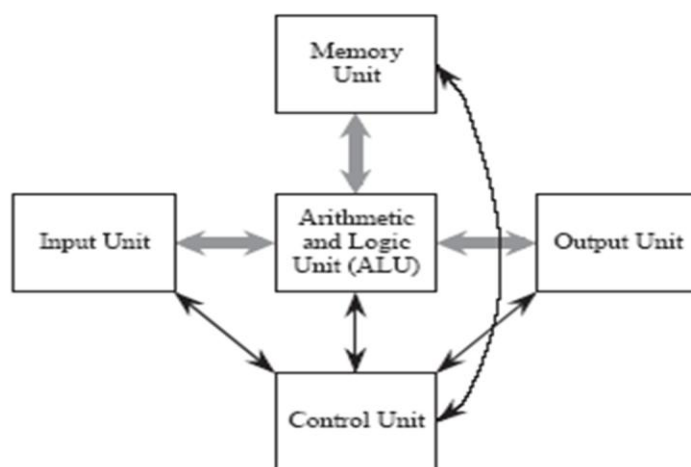


Figure 1: The von Neumann model of a digital computer. Thick arrows represent data paths. Thin arrows represent control paths.

This architecture uses what is known as the *stored program model*. In the von Neumann architecture, the **stored program** is the most important aspect of the von Neumann model. The key features of this architecture are as follows:

- There is no distinction between instructions and data. This requirement has several main implications:
 1. Instructions are represented as numbers, just like the data themselves. This uniform treatment of instructions and data simplifies the design of memory and software.
 2. Instructions and data are not stored in separate memories; a single memory is used for both. Thus, a single path from the memory can carry both data and instructions.
 3. The memory is addressed by location, regardless of the type of data at that location.

College of Computer Technology

- By default, instructions are executed in the sequential manner in which they are present in the stored program.

A program is stored in the computer's memory along with the data to be processed (A computer with a von Neumann architecture has a single memory space that contains both the instructions and the data, see figure 2). This can lead to a condition called the **von Neumann bottleneck**, it places a limitation on how fast the processor can run. Instructions and data must share the same path to the CPU from memory, so if the CPU is writing a data value out to memory, it cannot fetch the next instruction to be executed. It must wait until the data has been written before proceeding and vice versa.

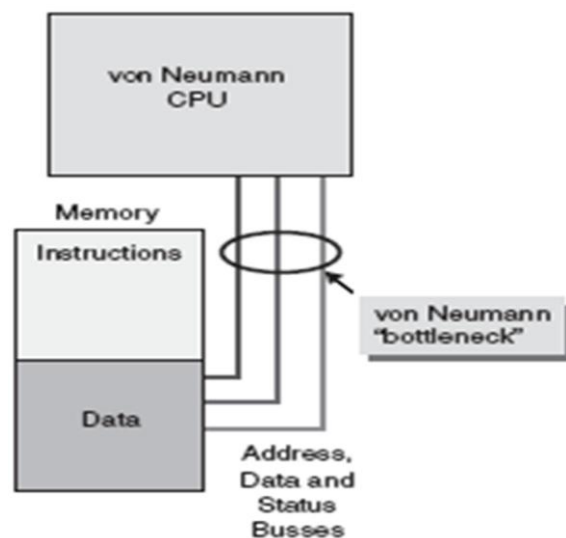


Figure 2: Memory architecture for the von Neumann

In contrast to the single memory concept used in the von Neumann architecture, the *Harvard architecture* uses separate memories for instructions and data. The term now refers to machines that have a single main memory but use separate caches for instructions and data.

Most processors now use two separate caches: one for instructions and the other for data. This design uses separate buses for instructions and data. processors typically use the Harvard architecture only at the CPU-cache interface.

In order to avoid the von Neumann bottleneck :-

- multi-level caches used to reduce miss penalty (assuming that the L1 cache is on-chip); and
- memory system are designed to support caches with burst mode accesses.

Programmed I/O (PIO).

I/O ports provide the basic access to I/O devices via the associated I/O controller. We still will have to devise ways to transfer data between the system and I/O devices using the I/O ports. A simple way of transferring data is to ask the processor to do the transfer. In this scheme of things, the processor is responsible for transferring data word by word. Typically, it executes a loop until the data transfer is complete. This technique is called *programmed I/O* (PIO).

The CPU sends commands to the I/O device through an I/O address. Suppose the CPU wants some data from a part of one of the disk drives. When a PC accesses data from a disk, it can't take it in 1 or 2 bytes, the smallest amount of data that the CPU can ask for is a 512 byte sector.

The operations that take place for programmed I/O are shown in the flowchart in Figure 3. The CPU first checks the status of the disk by reading a special register that can be accessed in the memory space, or by issuing a special I/O instruction. If the disk is not ready to be read or written, then the process loops back and checks the status continuously until the disk is ready. This is referred to as a **busy-wait**. When the disk is finally ready, then a transfer of data is made between the disk and the CPU.

After the transfer is completed, the CPU checks to see if there is another communication request for the disk. If there is, then the process repeats, otherwise the CPU continues with another task.

College of Computer Technology

One disadvantage of programmed I/O is that it wastes processor time. Another problem is that high priority devices are not checked until the CPU is finished with its current I/O task, which may have a low priority.

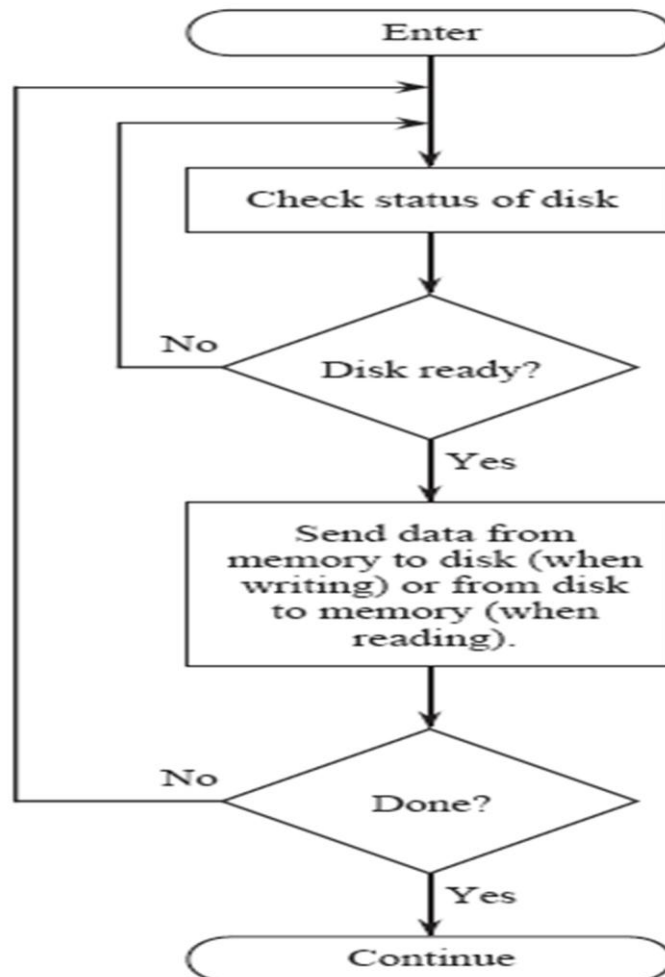


Figure 3 Programmed I/O flowchart for a disk transfer.

In addition to the Programmed I/O, two other mechanisms can be used to carry out I/O operations. These are interrupt-driven I/O and direct memory access (DMA).