CPU and Von Neumann Architecture

The CPU is the circuitry that controls the manipulation of data.

Parts of the CPU:

- ALU – Arithmetic / Logic Unit (circuits that perform the operations on the data)
- CU – Control Unit (The circuits coordinating the activities of the CPU)
- Registers (Quick, small stores of data within the CPU)

Major characteristics of this theorical model:

Main Memory	ALU & ACU	ALU & CU	Cache	Registers
Instructions to be executed are stored in the memory, alongside the data as binary values (The stored program concept)	Instructions are executed sequentially: One instruction at a time is fetched form the memory and passed to the CPU	ALU: Performs calculations and logical decisions CU: Sends signals to control how data moves around the CPU	Provides fast access to frequently used instructions and data	Program counter: Holds the address of the next instruction in memory Memory Address Register: Holds the address of where the data is to be fetched or stored Memory data register: Holds the data fetched from or to be written in memory Accumulator: Holds the results of calculations

(You can't access data and instructions at the same time)

Even if your CPU is absolutely rapid, if the connection between the memory and the CPU is very slow, you get the Von Neumann Bottleneck

Computers have a system clock which provides timing signals to synchronize circuits, steady rhythm.

CPU are designed to operate at a specific frequency – and the system clock is raised to this rate by the processor, giving the clock speed (Hz).

The CPU is much faster than the other components.

The CPU needs a certain amount of clock cycles per instruction.

The Fetch-Execute Cycle:

Fetch: The next instruction is retrieved by the CPU from main memory

Decode: The instruction is broken down to its individual components to determine what the instruction is, and what data is being used

Operator +	Operand
ADD	R1 R5

Execute: The CU activates the necessary circuitry / data transfers. The output of this stage is stored in a register, and data may be read / written from/to the main memory during this stage.

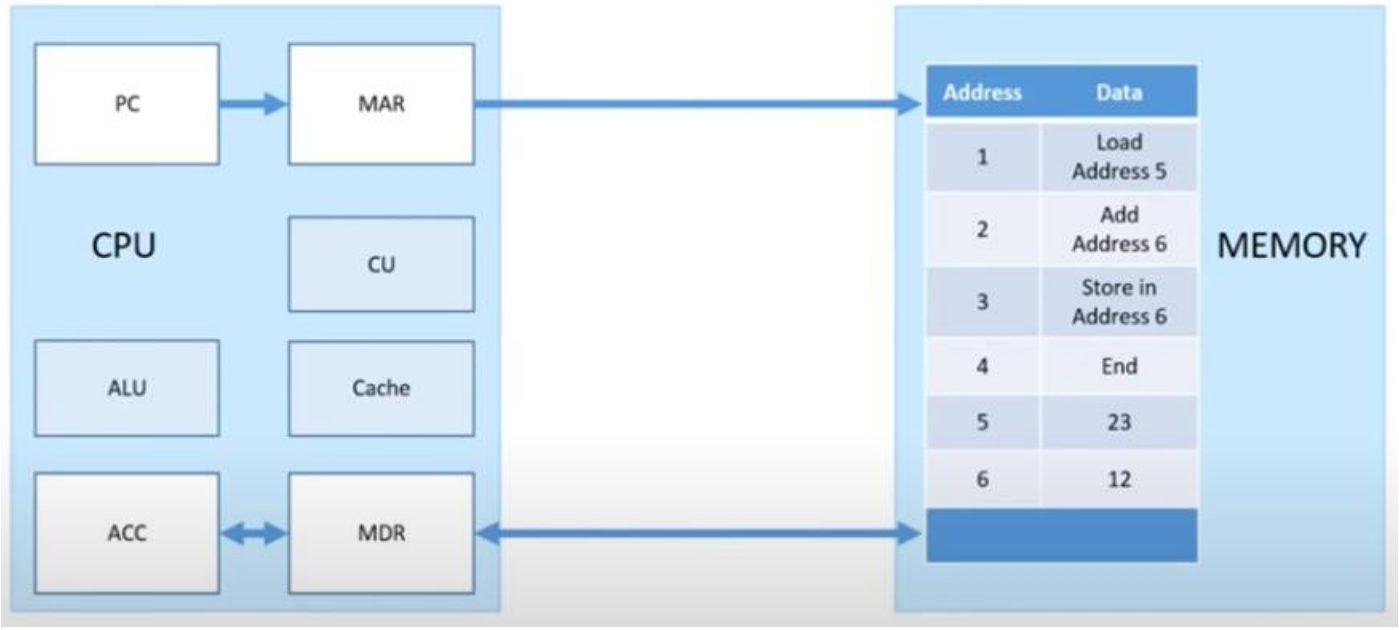
Short: The Von Neumann Architecture fetches instructions from memory, decodes, and executes them

Programs made up of sequences of instructions are stored in the main memory, outside of the CPU.

They are fetched one by one to the registers for processing.

Each instruction is stored in a location in the memory. Each location has an address.

Example:



1) The address in memory where the next instruction can be found is stored in the program counter: PC

➔ In this example the first instruction is at address 1 so the value in PC is 1

2) It is then passed to the MAR (Memory Address Register)

3) Then into the MDR (Memory Data Register) that will say 'load address 5

➔ 'Load Address 5' is passed from the Memory to the MDR

4) The PC (program counter) then points to the next instruction

-> PC is now set to 2.

5) A copy of what was loaded into the CPU is stored in the cache, so if it's needed afterward, we don't need to go back at the main memory in order to load it again.

-> Cache contains 'Load 5'

6) The control unit is going to decode this instruction

-> The CPU realizes it has to load and fetch the value at the address 5: 23
-> The MDR now contains 23
-> A copy of 23 is now in the cache as well

7) Our value can now be sent to the ACC (ACCumulator)

That is one instruction that has been fetched and executed.

8) The next instruction is passed from the PC to the MAC and to the memory

-> 2 is passed from PC to MAR, then to memory

9) The instruction at the address is sent to the MDR

-> 'Address 3: Add address 6 (content)' is sent from memory to the MDR, replacing the value 23 by "Add what is in address 6"

10) Copy of the instruction is stored into cache and the PC is incremented

11) The control unit passes the memory address register and fetches the value in address 6

12) The value at the address is sent to the MDR and to the cache

13) The ALU performs the calculation

14) The result is stored in the Accumulator

15) The program sends the address pointing to the next instruction to the MAR

16) Value is sent to MDR, copied in the cache, incrementation of the PC

-> The instruction in address 3 is sent to the MDR

17) The CU looks at the instruction and changes the memory address of the register to 6 and the value that's in the accumulator is sent back to the MDR, and then to the address specified in the instruction.

-> 35 is sent from the ACC, to the MDR, and then to the address 6

18) End instruction is sent from PC, to MAR, to Memory

19) End instruction is received by MDR. PC increments, CU decodes that instruction, and it's over.

More: https://www.youtube.com/watch?v=t8H6-anK0t4