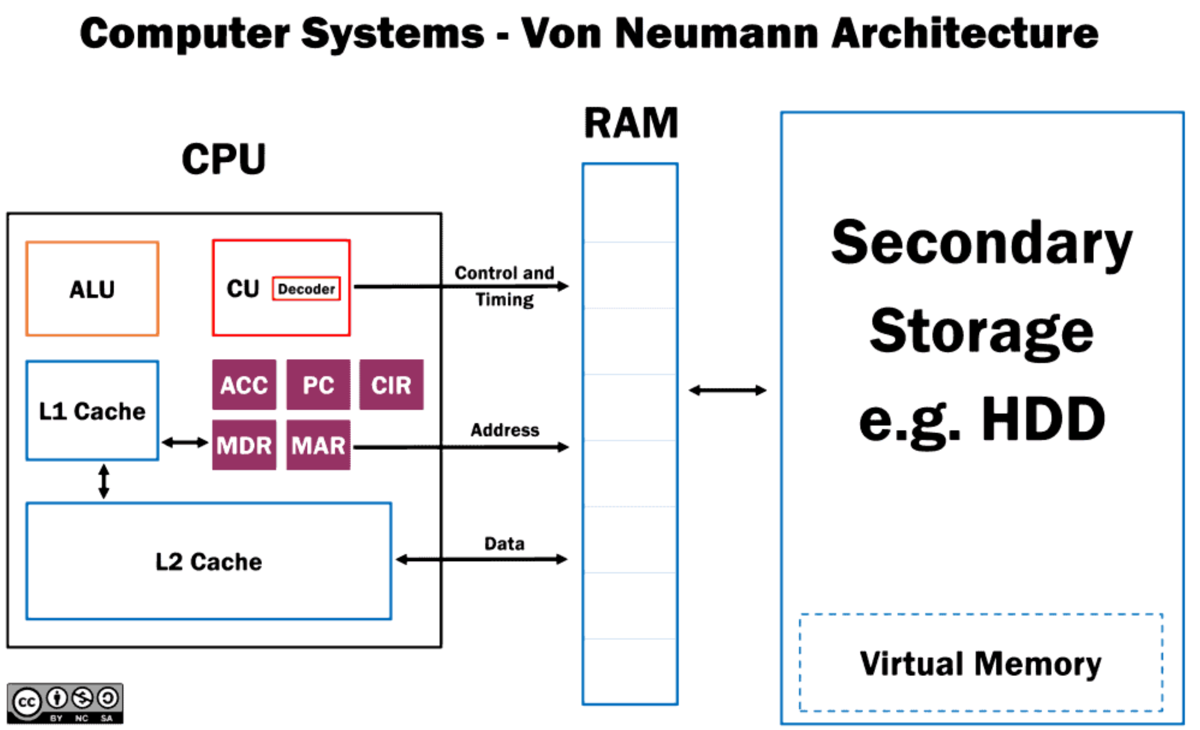
Central Processing Unit

The CPU executes or processes the instructions and data in a computer application to output a result.

The CPU contains the:
- Control unit (CU)
- Arithmetic Logic Unit (ALU)
- A number of registers
- A number of buses to connect the components

A microprocessor is an electrical device that works on an integrated system on a single chip and contains the circuitry necessary for a computer to perform logical operations. It only contains a CPU, memory is connected externally.

Main functions:
1) Receives input data
2) Processes data
3) Provides output data



Von Neumann Model:
Fetch → Decode → Execute

Random Access Memory

A volatile storage unit; volatile meaning dependent on a power source. It is called random because any storage location can be accessed at any time.

When running, it stores data, instructions, and the operating system.

The information in the RAM can be swapped and rewritten at any time. It is also known as the immediate access store as read/write operations can be completed considerably faster than a hard disk/ssd due to the way it functions.

Memory is partitioned into the address and its content, both stored as binary digits.

Opcode and Operand
Opcode: instruction
Operand: data to be processed

Address	Value
1101	0101
0110	0010
1011	1101

The stored program concept refers to instructions and data stored on the same RAM memory and both are fetched sequentially one at a time.

ROM

ROM (Read only memory) and contains the **BIOS** and **Bootstrap**. It stores data (instructions and memory) that is necessary for the computer to run and startup. This is non-volatile..

The **BIOS (basic input/output system)** is a component in the computer that is used whenever a computer is booted up;
checks initialisation settings for the computer;
checks if the hardware is running correctly;

Bootstrap loads the operating system into the RAM.

CPU Registers

These are high-speed storage units within the CPU that temporarily store data or instructions.

- **MAR** - Memory address register
  - Stores the address of the memory location that is currently being read or written to
- **MDR** - Memory data register
  - Stores data fetched from address in MAR
- **ACC** - Accumulator built into the **ALU**
  - Temporarily stores data from calculations from the ALU
- **PC** - Program counter

    - Stores the address of the memory location of the next instruction to be read
- **CIR** - Current instruction register
    - Stores the current instruction being decoded and executed

Units

ALU - Arithmetic logic unit

- Carries out logical and arithmetic operations to be carried out while a program is running

CU - Control unit

- Decodes instructions from memory using an instruction set
- Manages the operation of other components in the computer by sending control signals

Buses

Paths/wires for data to be transferred between the components of a computer through parallel data transmission.

CPU <-> RAM (bi-directional)

- Control bus
    - Carries signals to coordinate computer's activities (CU to RAM, RAM to CU)
    - A system clock sends timing signals through the control bus to coordinate the components in the computer
- Data bus
    - Exchanges data between the processor, memory, and input/output devices (between cache and RAM)
    - So that data can either be read (ram to cache) or written (cache to ram)
    - Can also be between the MDR and RAM

CPU -> RAM (uni-directional)

- Address bus
    - Carries signals correlated to memory addresses (**MAR** to RAM) between processors
    - Addresses cannot be carried back to the CPU

Fetch-Decode-Execute

Fetch

The address of the next instruction to be fetched stored in the **PC** is moved into the **MAR**.
The address in **MAR** is sent to **RAM** through the address bus.
Data from that address is moved into the **MDR** via the data bus.
The data in the **MDR** is copied over into the **CIR**.
The **PC** increments by one to prepare for the next instruction.

Decode

The **CU** decodes the instruction in the **CIR** into an opcode and an operand so the CPU can understand and execute the instruction.

Execute
Signals sent through the control bus coordinate the components. The **ALU** executes the instruction and stores the result in the **ACC** or written into memory. Additional data may be fetched from memory to complete the instruction.

Data in **ACC** is then transferred to the **MDR** and then written into **RAM**.

Fill in the blanks:
PC, CIR, MDR, MAR, ACC
The address of the next instruction to be fetched is stored in the ____. This address is then moved to the ____. Data in that address is moved into the ____ and placed in the ____. The ____ then increments by one to prepare for the next instruction.

PC, MAR, MDR, CIR, PC

Stage	**Address**	
`LOAD`	`10`	Fetches value in [10]
`ADD`	`11`	Adds value in [11]
`STORE`	`12`	Stores value in [12]

In the example above, lets say:

Instruction	Address
`LOAD 10`	`10`
`ADD 2`	`11`
`ADD 3`	`12`
`SUBTRACT 2`	`13`

PC: `[10]` - address of memory
MAR: `[10]` - copies memory address in PC
MDR: `LOAD 10` - fetches data in address at MAR
CIR: `LOAD 10` - copies data in MDR

Cores, cache, clock

The **system clock** regulates clock speed by sending timing signals through the control bus.

Clock cycle: One complete operation of the CPU. It is the basic unit of time of the CPU. A common clock speed is 3.5 to 4.0 GHz.

A higher clock speed typically means that the computer is faster as it processes more instructions at a faster rate, increasing performance.

Overclocking
Changing the clock speed to a value higher than the manufacturer's settings can be done by altering settings in the BIOS. This will allow the CPU to execute more instructions at a time however this will generate much more heat than it was designed to handle and the computer often crashes.

The cache is very high performance memory stored in the CPU. This has a very fast access rate as it is located within the CPU. It improves the CPU performance by storing frequently accessed instructions or data. When the CPU accesses memory, it accesses the cache first, and then RAM. A larger cache means that more data can be quickly accessed by the CPU, making execution faster, increasing performance.

A **core** has an ALU, CU, and all the registers. It can complete one clock cycle at a time. More cores mean that the CPU can execute more instructions at a time, increasing performance. However the cores need to communicate with each other through channels, which can reduce performance.

Instruction sets

Instructions are a set of **simple, low-level operations**.

An operation is made of an opcode and an operand. An opcode is the command that tells the CPU what to do, and an operand is the data to be acted on or a register in memory.

An instruction set is all the operations that a computer can understand and execute. Each operation has to be converted into binary, and tells the CPU what to perform. Program code must be translated into the instruction set understood by the computer to be run.

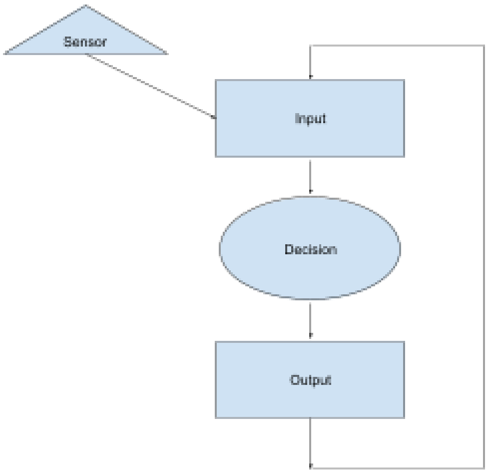
During the fetch-decode-execute cycle, an instruction is decoded into an opcode and operand so that the CPU knows what to perform when executing the instruction.

Instruction sets are specific to the processor. Some may be similar but not identical. An instruction set on one computer may not run on another with a different processor.

Embedded systems

A combination of software and electrical/electro-mechanical hardware to perform a specific task.

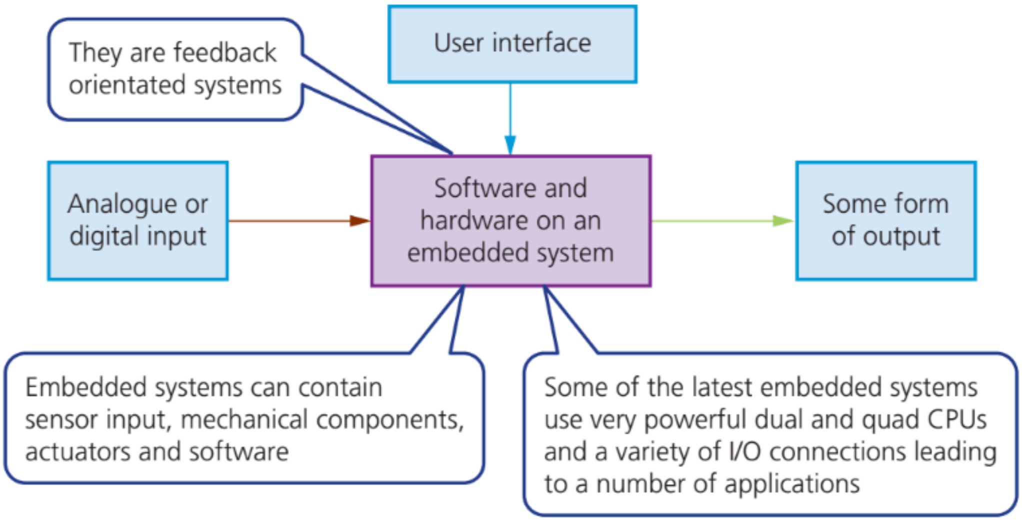
Feedback oriented:



Analogue sensor where ADC (analogue to digital converter) converts these signals to digital values

1) Transmits the digital signals to input
2) Decision is made in processor - input value compared to stored or decision value
3) Output - sends a signal to actuator

It then resamples after a given amount of time - receives and test the value from the input again



Ambient value: referring to the immediate surroundings

Embedded systems only do specific sets of functions, unlike a PC. They may be on microcontrollers, microprocessors, or microchips. It is always embedded into a single chip. The system will take an input either automatically through a sensor or manually by an operator, process it, then produce an output which will be the function of the system.

Perfect sensor response:
The *{sensor}* will take in/record/measure the *{what the sensor records}* of a/the *{area/person/target}* to/trying to *{goal}*, for example *{>3 examples}*, this analogue data is then sent to an ADC (analogue to digital converter) which turns it into machine code/digital format/binary code so the microprocessor can understand. The signal is sent to the microprocessor which *{what is done with values}* so *{goal}* can be achieved. *{IF CONDITIONS}* Depending on if *{condition}*, do *{goal}* otherwise do *{other goal}*. To do these goals, a signal is sent to a DAC, then to a *{device}* that *{does goal}*.