At the core of a computer sits the Central Processing Unit. This is what manages and executes all computation.

The CPU comprises three core components:

- Registers
- the Arithmetic Logic Unit (ALU)
- the Control Unit (CU)

  This method of putting together a computer is known as the **Von Neumann Architecture**. It was devised by John von Neumann in about 1945, well before any of the components that would be needed to produce it had actually been invented.

Registers

This is the part of the CPU that stores data. The memory cells that comprise it do not have capacitors (unlike RAM) so they cannot store very much data but they work faster, which is what is important.

There are five main types of register in the CPU: Pasted image 20220319175645.png

Arithmetic Logic Unit

This is the hub of the CPU, where the binary work gets done. It contains logic gates and executes processes on them. This is where the data stored by the registers is processed and altered.

It can execute arithmetic on binary numbers and logical operations.

This is the **core** that is referred to in hardware specs of computers, for instance *dual-core*, *quad core* etc.

Control Unit

The control unit takes the instructions in binary form from RAM memory (separate from the CPU, but connected) and then signal to the to ALU and memory registers what it is supposed to do to execute the instructions. Think of it as the overseer that gets the ALU and registers to work together to run program instructions.

In addition to the these three active components in the CPU, we also have:

- Buses

  Bundles of wires that transfer data between the CPU constituents. There is a bus to carry data, another for addresses and another for instructions.

- Input and output

  Devices that connect to the CPU, receive external data and output the results. For instance keyboards and monitors.

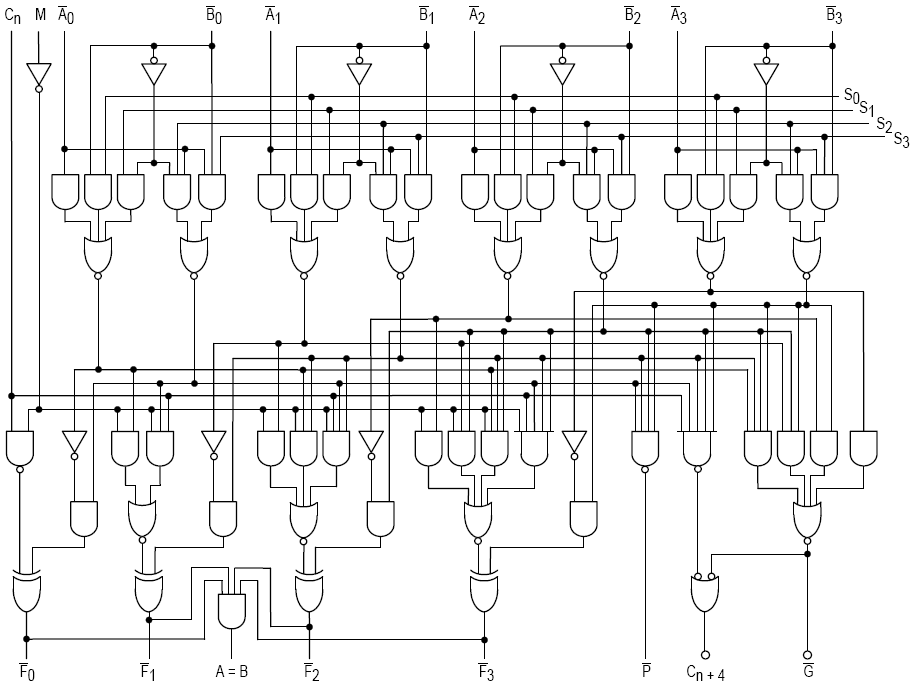


Figure 1: 74181aluschematic.png

Fetch, decode, execute

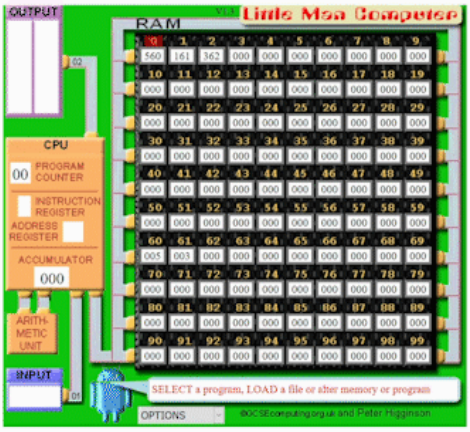
Fetch, decode, execute is the operating principle of the CPU. We will run through how this works with reference to the CPU components detailed above.

- **Fetch**
  - The Program Counter needs to keep track and sequence the different instructions that the CPU will work on. The first place it will look for an instruction is at the RAM address `0000` , equivalent to zero in the count - the starting point. This is address therefore copied to the Memory Address Register for future reference.
  - This memory-storing event constitutes an instruction so it is copied to the Instruction Register.
  - As the first instruction has been fetched, the system reaches the end of the first cycle. Thus the Program counter increments by 1 to log this.
  - At this point the next fetch cycle begins.
- **Decode**
  - Now that the instruction is fetched and stored in the RAM it needs to be decoded. It is therefore sent from the RAM to the Control Unit of the CPU.
  - There are two parts to the instruction:
    1. The operation code → the command that the computer will carry out.
    2. The operand → (that which will be operated on) an address in RAM where the data will be read and written to as part of the execution
  - The Control Unit converts the operation code and operand into instruction that are fed to the next execute cycle.
- **Execute**
  - Now the command will be executed. The operand is copied to the Memory Address Register and then passed to the Memory Data Register and the command is carried out by the ALU.

The Little Man Computer

The Little Man Computer is a simplified computer that works on Von Neuman architecture.It has all the CPU components we have detailed above. It is programmed in machine code (as we saw with the Fetch, Decode, Execute cycle above) but for simplicity it uses denary, not binary.

```
1xx - Add
2xx - Subtract
3xx - Store
5xx - Load
6xx - Branch
7xx - Branch if 0
8xx - Branch if +
901 - Input
902 - Output
000 - Halt
```



On the left is the instruction set. Each number constitutes and execution routine and the `xx` stand for the address in RAM that the execution will work on.

Each row of the RAM has a denary address, 1 through to 99. Each address can hold three digits.

- So the instruction `560` would mean *load the number at address 60.*
- The instruction `340` would mean *store a datum at address 40*

Working through a basic computation

We are going to add two numbers together as a basic example.

1. First we need to place the two numbers in RAM we are going to use `5` and `3`
   - At address `60` we will put the number `5` and at address `61` we will put the number `3`
   - We are going to start at address `0` in the top left of the RAM grid
2. The first instruction will be *load address 60* which in the assembly will be `560` . We put this in address `0`, our starting point.
3. This first instruction is now stored in the accumulator.
4. Now we want to *add this number (in the accumulator) to the number in address 61*
5. This second instruction is `161` . We write this in address `1`
6. Finally we want to store the output of the calculation in the RAM, let's say at address `62`
7. So we store the command `362` at address `2`