Lots of Bytes

Since we are typically dealing with large numbers of bytes, we use prefixes to capture orders of magnitude:

kilo	k	2^{10}
mega	m	2^{20}
giga	g	2^{30}
etc.		

Boolean Logic

How can a chip of silicon and wire do something that seems like it requires human thought?
To understand the answer, we need to understand *Boolean Logic*. This was developed by George Boole.

Boole was interested in language, and statements that could be codified as either "true" or "false". He went through the bible to see if it was consistent.

Boole's logic was used by Claude Shannon to develop switching theory in the 1930s.
Operators in phone lines couldn't cope with the volume of calls and needed to be replaced.
As Shannon researched switch theory, he came across Boolean algebra.

Logic Gates

A *logic gate* is an electronic circuit that "logically combines" inputs to produce outputs. By logically combining we mean

applying a logic operator to some operands to produce some output.
A logic gate corresponds to a logic operator.

Boole considered 3 logic operators: AND, OR, NOT.
There are logic gates for each of these and for combinations of them.

With these logic gates, Boole's and Shannon's work showed that we could construct any electronic circuit whose output depends only on its inputs.
E.g. we can add 2 + 5 = 7, since the output (7) only depends on the inputs (2 and 5).
These are called *combinatorial* or *combinational circuits*.

Modelling information using numbers is one thing (data), and processing that information is another.
By processing, we mean transforming/combining information to create new information.
Consider this expression:

$$2 + 3 = 5$$

This uses an operator (+) to map two entities (2 and 3) into a new entity (5). A relationship is created between the operands of + and its results.

We could write this functionally as follows:

$$+(2, 3) \rightarrow 5$$

In a sense we are creating new information by doing this. What is the meaning we attach to +? Addition (along with other arithmetic operations) allows us to equate many quantities, combined using specific rules, with a single result quantity.

There is a profundity to this that often goes unnoticed by the casual observer:
For a given context it can be said that the single value of the result has (in some sense) the same meaning as the combination of the operands.
The results and the combination of the operands are indistinguishable from each other.

In computing we say that they are *referentially transparent*.

When the values are computed in this way, information relating the combination of operands and the results is uncovered.
Every program is like this, with extensive and complex rules.

First, we consider only those functions whose oput is defined only by its inputs.

$$O = f(I)$$

Any function of this type can be defined in terms of the three logical operators: AND, OR, NOT.
This is pretty weird. Any program, e.g. driverless cars.

Truth Tables

AND

The effect of AND, OR, and NOT can be specified as a truth table, which lists all inputs and their outputs.

A	B	AND

F	F	F
F	T	F
T	F	F
T	T	T

A	B	AND
0	0	0
0	1	0
1	0	0
1	1	1

Algebraically, we write A AND B as A.B or just AB.

In an electronic diagram, the symbol looks like:

$_{B}^{A}$=D—AB

It's amazing that you can build addition from just logic operators.

OR

A	B	OR
0	0	0
1	0	1
0	1	1
1	1	1

Algebraically: A+B

Note: don't think of these symbols as multiply and plus, it's not good. Historically, that's why we use these symbols, but it's not good.

Circuit symbol:

$^{A}{}_{B}$=)>—A+B

NOT

A	NOT
0	1
1	0

Algebraically: A' or Ā

Circuit diagram:

A—|>o—Ā

Exclusive OR

A	B	Exor
0	0	0
0	1	1
1	0	1
1	1	0

Algebraically: A⊕B

Circuit symbol: $^{A}{}_{B}$=))>—A⊕B

This gate is not necessary, but it's useful. You can build it from AND, OR, and NOT, (using 5 gates), but it's often easier to use one gate than five, since we use it a lot.

Next, using logic gates, he will do addition, subtraction, multiplication, and division from first principles right up.