Chapter 1. Data Storage (Notes)

Brookshear, J., et al. *Computer Science: an Overview, Global Edition*, Pearson Education, Limited, 2019. *ProQuest Ebook Central*, https://ebookcentral.proquest.com/lib/qub/detail.action?docID=5676415.

A ***computational artifact*** is ***anything* created by a human using a computer**. People use tools and abstractions to create software, datasets, media, etc.

How are these artifacts encoded and stored in computers?

Bits and their Storage

All data, not just the numeric kind, is encoded as ***bits*** - short for **binary digits**. They can also represent characters, colours and sounds.

Boolean Operations

Conceptually, **1 = *true*** and **0 = *false***.

Operations that manipulate true and false values are called **Boolean Operations** (after mathematician George Boole).

The operations are:

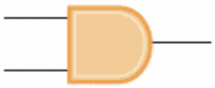
- **AND**: A conjunction between two expressions, both of which must be true
- **OR**: These statements are true when at least one of their components is true
- **XOR**: XOR is true when one expression is true and the other is false. When true it outputs the value 1 or *true*
- **NOT**: Only has 1 input, and its output is the opposite of its input. Thus if the input is true, then the output is false.

These also have pictorial representations with special symbols:

Gates and Flip-Flops

A device that **produces the output** of a Boolean operation is called a ***gate*** or **logic gate**.

AND



Inputs	Output
0 0	0
0 1	0
1 0	0
1 1	1

OR

Inputs	Output
0 0	0
0 1	1
1 0	1
1 1	1

XOR

Inputs	Output
0 0	0
0 1	1
1 0	1
1 1	0

NOT

Inputs	Output
0	1
1	0

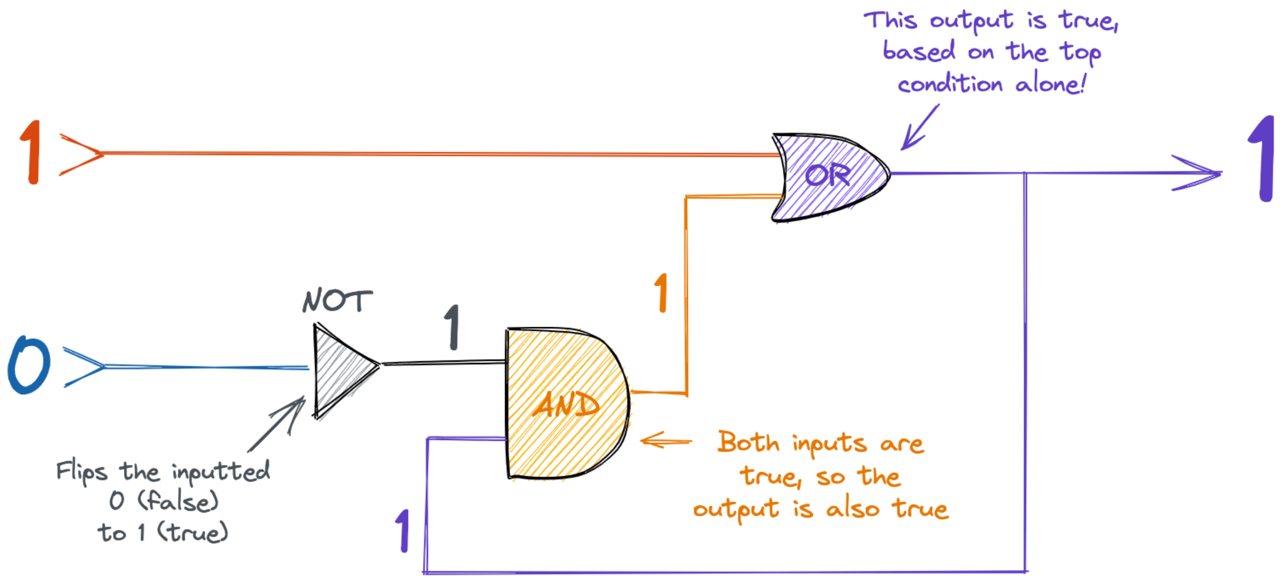
Figure 1.2 A pictorial representation of AND, OR, XOR, and NOT gates as well as their input and output values

In modern computers, gates are made of **transistors**. The digits 1 and 0 represent voltage levels - essentially **on** and **off** respectively.

Gates are the **building blocks for computers**, and Booleans are fundamental to all programming languages.

A ***flip-flop*** is a computer component that is a **fundamental unit of memory**. It is a **circuit** which produces an output value of 1 or 0. The output remains constant until a short pulse from another circuit causes it to switch to the other value.

A simple flip-flop circuit with example inputs and output:



This output **will remain 1 even if we subsequently change the top input to 0**, because the 1 from the AND gate is still in the circuit, meaning that the OR gate still outputs 1.

A computer engineer really only needs to know the **external properties** of a fiip flop, as opposed to the exact structure of a circuit.

Because it can be set to have an output of 1 or 0, a flip-flop is one means of **storing a bit**. Other circuits can adjust this value by sending pulses. Therefore **lots of flip-flops inside a computer can be used to store information encoded as 1s and 0s.**

Chips can house millions of flip-flops (see Very Large Scale Integration). Chips are then used as abstract tools in the construction of higher-level computer components.

Hexadecimal Notation

Computers process patterns of bits. These are bit strings that can be very long, are usually in multiples of four, and are called ***streams***.

Hexadecimal Notation, or **Base 16**, is a shorthand, using one symbol to represent a particular **combination of 4 bits** (so a stream of 12 bits can be written as 3 hexadecimal symbols). The base of 16 often follows the symbol in subscript (eg F_{16}, which is 15_{10}).

Bit pattern	Hexadecimal representation
0000	0x0
0001	0x1
0010	0x2
0011	0x3
0100	0x4
0101	0x5
0110	0x6
0111	0x7
1000	0x8
1001	0x9
1010	0xA
1011	0xB
1100	0xC
1101	0xD
1110	0xE
1111	0xF

Figure 1.6 The hexadecimal encoding system

Main Memory

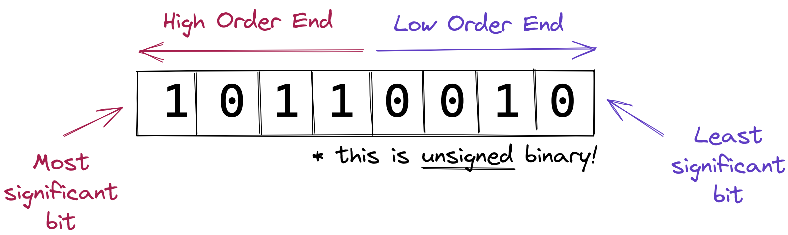
A computer contains lots of flip-flops, each one capable of storing a single bit. This bit reservoir is called the computer's ***main memory***.

Memory Organisation

Main memory is organised into ***cells***, usually of 8 bits (a **byte**). Thus a memory cell's capacity is one byte. A microwave has hundreds of cells in its main memory, while a desktop or smartphone has **billions**.

We visualise bits in a memory cell as being arranged in a row (although they're not really).

Organisation of a Byte Size Memory Cell



Each cell has a **unique** ***address*** in memory. We envision the cells as being placed in a single row and assigned addresses in **ascending numerical order** (cell 0, cell 1, cell 2, etc):

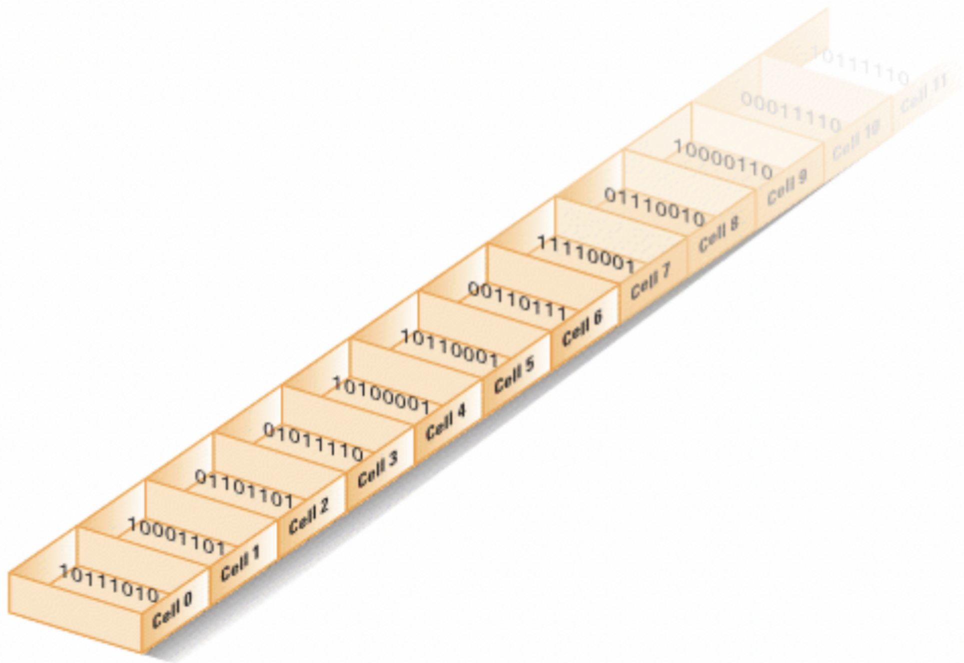


Figure 1.8 Memory cells arranged by address