Numbering Systems

Bits & Bytes

One **byte** contains 8-bits. It's an 8-digit binary number.

What is the largest number that can be stored in one byte?

$$(11111111)_2 = 255$$

Still, that's not very big! To accommodate that, we need to look at *many* bytes…

Kilobytes – they used to be big!

One **kilobyte (KB)** is equal to 1024 bytes*.

At one time, this was *a lot* of information! Our needs grew…

One **megabyte (MB)** equals 1024 KB ~ approx 1 million
One **gigabyte (GB)** equals 1024 MB ~ approx 1 billion
One **terabyte (TB)** equals 1024 GB ~ approx 1 trillion
One **petabyte (PB)** equals 1024 TB ~ approx 1 quadrillion

* Depending on the implementation, some say there are 1000 bytes in 1 KB. This makes the math a little easier!

New terminology

There are 3 different ways we use those units measure when we talk about computers:

1. Capacity - How much can something hold
2. Bandwidth - How much can be transferred at a time. Bandwidth is usually measured in bits (b) or bytes (B). Notice the difference in b/B.
3. Speed - How fast/often can it transfer information

All 3 types of measurement are a part of the data transfer rate of a computer component.

Examples:

1. Capacity - How much can something hold

   CPU cache - 8 MB (B = Bytes)

   USB Drive - 30 Mbps (lowercase ‘b’ = bit)

2. Bandwidth - How much can be transferred at a time

   Internet bandwidth - 100 Mbps (“bps” means bits per second.

3. Speed - How fast/often can it transfer information

All 3 types of measurement are a part of the data transfer rate of a computer component.

Analogy

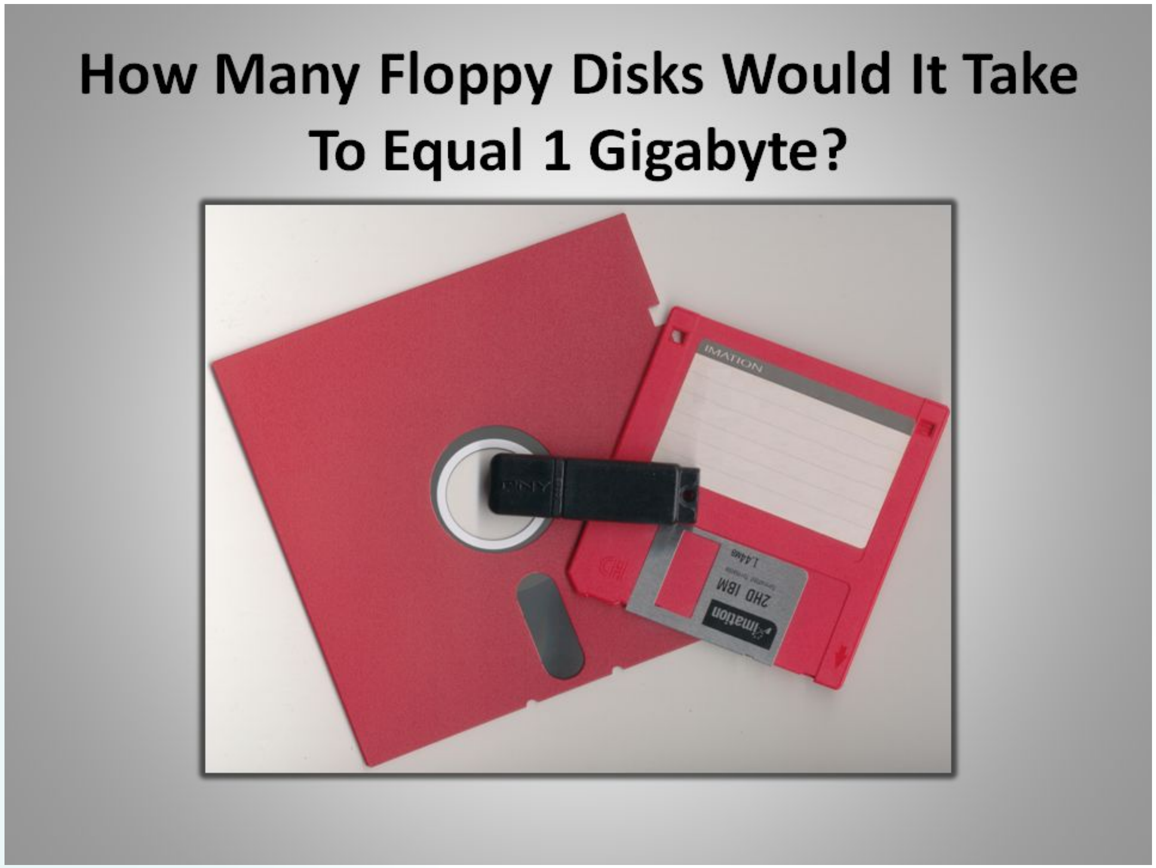
Imagine you are filling a room with boxes.

- Capacity is the number of boxes the room can hold
- Bandwidth is how many boxes you can fit through the door at one time
- Speed is how fast you can make the trip to grab more boxes.

So, increasing the size of the door OR how fast you can get more boxes will increase the data rate.

Increasing capacity only means it takes you longer to fill the room

Question:



Converting Between Units

What do we need to know?

- Each floppy disk holds 1.44 MB.
- One gigabyte holds 1024 MB.

To convert, simply divide!

1024 / 1.44 = 712 (rounded up)

Representing Large Numbers

Suppose we want to encode a decimal number that is *larger* than a byte (255). There's an issue when it comes to using binary to store larger numbers…

It takes too many bits!

Think about it…

It takes EIGHT digits in binary to represent the THREE digit decimal number 256.

Can you imagine trying to represent the number 123,456,789?

123,456,789

Here it is!

111010110111100110100010101

It’d be nice if we could represent these large binary numbers using a ‘smaller’ notation…

Hexadecimal

Hexadecimal is our solution!

- Digits use the values 0-9, A-F:
  - 0-9 hold their normal meaning
  - ‘A’ represents 10, ‘B’ represents 11, …, ‘F’ represents 15
- From right to left, each digit represents a power of 16.
- We often prefix a hexadecimal number with “0x” in order to indicate that it’s a hexadecimal number.

Examples of Hexadecimal

a) 0x0 represents the number 0

b) 0xA4 represents:

$4 \times \mathbf{16^0} = 4 \times \mathbf{1} = 4$

$A \times \mathbf{16^1} = 10 \times \mathbf{16} = 160$

$4 + 160 = 164$ in total

c) 0x100 represents:

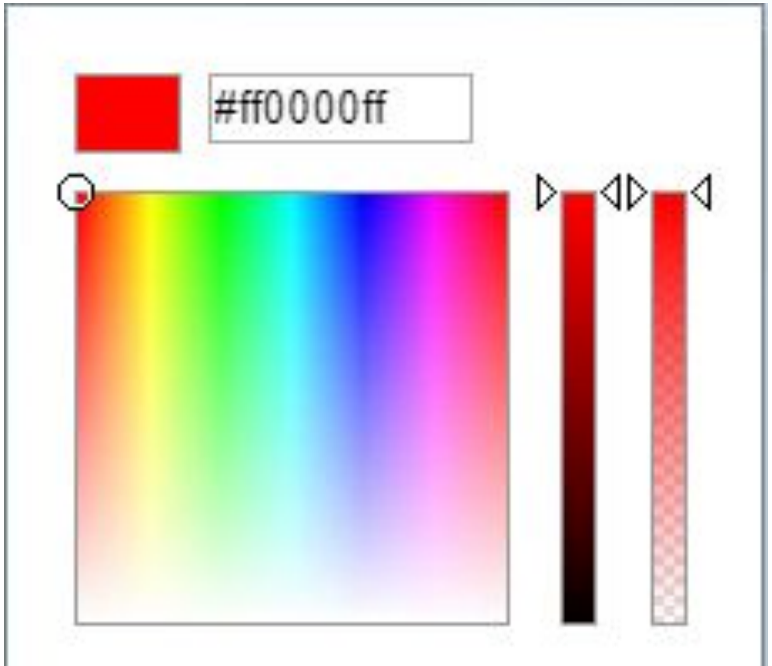
$1 \times \mathbf{16^2} = 256$ (Recall: This took *eight* bits in binary!)

Other uses of hexadecimal

Colours in computing are usually represented by a 6-digit hex number:

#123456

The first 2 digits are for red, the middle 2 are for green and the last 2 are for blue.



Each pair of digits can represent the numbers

0 (0) to 255 (FF)

Examples: #44AF17 #A11256 #3F49FF #BCA561

44_{16} (read 44 base 16) is 68_{10} (68 base 10) for Red

AF_{16} is 175_{10} for Green

17_{16} is 23_{10} for Blue

Hex is also used for...

- URL’s on the WWW (%20 is a space in a URL)
- Memory locations in RAM (more on that later)

Binary: 1011 1100 0011

Hexadecimal: B C 3