

## Units

Just like you have units like centimetres, metres and kilometres for measuring distance, computers need units for measuring digital information. You'll need to learn all of the unit names on this page and their sizes.

### Bits are the Smallest Measure of Data

- Computers use 1s and 0s to represent the flow of electricity.  
1 is used to show that electricity is flowing, and 0 shows that it is not flowing.
- All the data we want a computer to process must be converted into binary code (1s and 0s).
- Each 1 or 0 in a binary code is a bit (binary digit). For example, 1010 is 4 bits.
- The table below shows the size of other units of data:

A byte is big enough to store one character (like x, e, M or £).  
See p.27 for more info.

Most files (like songs, pictures and documents) are measured in kB or MB.

High definition videos and complex applications are often measured in gigabytes.

Secondary storage capacity is measured in gigabytes or terabytes.

Name	Size
Bit (b)	A single binary digit (1 or 0)
Nibble	4 bits
Byte (B)	8 bits
Kilobyte (kB)	1000 bytes
Megabyte (MB)	1000 kilobytes
Gigabyte (GB)	1000 megabytes
Terabyte (TB)	1000 gigabytes
Petabyte (PB)	1000 terabytes

You might see each unit defined to be  $1024$  (not  $1000$ ) times bigger than the previous unit. The main reason is that  $1024$  is a power of 2 which is helpful when dealing with binary data.

- Each bit can take one of two different values (either 1 or 0). This means that a nibble (4 bits) can take  $2^4 = 16$  different values, and a byte (8 bits) can take  $2^8 = 256$  different values.

### You can Convert between Different Units

Converting between units of data is usually pretty straightforward — just watch out when you have to switch between bits and bytes.

#### EXAMPLE:

Ashley has downloaded some images to her computer. Each image is 300 kilobytes.

- a) How many bits are in each image?

- First, convert to bytes by multiplying by 1000:  $300 \text{ kB} = 300 \times 1000 = 300\,000 \text{ Bytes}$
- There are 8 bits in a byte, so multiply by 8:  $300\,000 \text{ Bytes} = 300\,000 \times 8 = 2\,400\,000 \text{ bits}$

- b) She wants to copy 400 of these images onto her USB flash drive, which has 0.15 GB of free space left. Does she have enough space to store them all?

- Work out the total size of all the images:  $400 \times 300 = 120\,000 \text{ kB}$
- Now convert this to GB — first, divide by 1000 to get it in MB, then again to get it in GB:  
 $120\,000 \text{ kB} = 120\,000 \div 1000 = 120 \text{ MB}$   
 $120 \text{ MB} = 120 \div 1000 = 0.12 \text{ GB}$   
 So yes, she has enough space.

### This page has me in bits...

Keep working your way through that unit table until the size order is clear in your head — it might just show up on your exam. A bit is smaller than a nibble, and a nibble is less than a full byte. I know, hilarious.



# Binary Numbers

As computers only understand 1s and 0s, all data must be converted into binary to be processed. Binary can be used to represent all numbers in our standard number system.

## Counting in Binary is a bit like Counting in Denary

- 1) In our standard number system we have ten different digits (0, 1, 2, 3, 4, 5, 6, 7, 8, 9). This is called **denary**, **decimal** or **base-10**.
- 2) **Binary** only uses **two** different digits (0 and 1) — we call this **base-2**.
- 3) Counting in binary is similar to counting in denary, but the place values from **right to left** increase by **powers of 2** (e.g. 8, 4, 2, 1), instead of powers of 10 (e.g. 1000, 100, 10, 1).
- 4) The following table shows the **binary equivalents** of the **denary numbers 0-15**:

0 = 0	4 = 100	8 = 1000	12 = 1100
1 = 1	5 = 101	9 = 1001	13 = 1101
2 = 10	6 = 110	10 = 1010	14 = 1110
3 = 11	7 = 111	11 = 1011	15 = 1111

- 5) Most binary numbers are given as **8-bit** numbers, e.g. 01101011, which can represent the denary numbers **0 to 255**. The bit with the **largest** value (the **left-most bit**) is called the **most significant bit**, and the bit with the **smallest** value (the **right-most bit**) is called the **least significant bit**.

## Binary Numbers are easier to Convert using Tables

Drawing a table with binary **place values** in the first row makes binary to denary conversion easier.

### EXAMPLE:

Convert the 8-bit binary number 00110101 to a denary number.

- 1) Draw up a table with binary place values in the top row. Start with 1 at the right, then move left, **doubling** each time.

128	64	32	16	8	4	2	1
0	0	1	1	0	1	0	1

Each column is just a power of 2, i.e.  $2^3, 2^2, 2^1, 2^0$ .

- 2) Write the binary number 00110101 into your table.

- 3) **Add up** all the numbers with a 1 in their column:

$32 + 16 + 4 + 1 = 66$ . So 00110101 is 66 in denary.

This works with all binary numbers — just draw as many columns as you need, doubling each time.

## Convert Denary to Binary by Subtracting

When converting from **denary** to **binary**, it's easier to draw a **table** of binary place values, then **subtract them** from **largest** to **smallest**. Have a look at this example:

### EXAMPLE:

Convert the denary number 79 into an 8-bit binary number.

- 1) Draw an 8-bit table.
- 2) Move along the table, **only** subtracting the number in each column from your **running total** if it gives a **positive** answer.
- 3) Put a 1 in every column that gives a positive answer, and a 0 in the rest.

128	64	32	16	8	4	2	1
0	1	0	0	1	1	1	1

79 - 128 = -49  
79 - 64 = 15  
15 - 32 = -17  
15 - 16 = -1  
15 - 8 = 7  
7 - 4 = 3  
3 - 2 = 1  
1 - 1 = 0

So 79 converted to an 8-bit binary number is 01001111.

There are other methods to convert denary to binary, so just choose the one you are most comfortable with.

## Use powers of 2 to convert between binary and denary...

There's a really easy way to test yourself on this stuff. Write down a denary number between 0 and 255 and convert it to binary. Then write down an 8-bit binary number and convert it to denary.



# Binary Numbers

## Add Binary Numbers using Column Addition

As binary only uses 1s and 0s we can comfortably do  $0 + 0 = 0$ ,  $1 + 0 = 1$  and  $0 + 1 = 1$ .

Using binary we can't write  $1 + 1 = 2$ . Instead, we have to write  $1 + 1 = 10$ .

**EXAMPLES:** 1. Add the following 8-bit binary numbers together: 1000 1101 and 0100 1000

- 1) First, put the binary numbers into columns.
- 2) Starting from the right, add the numbers in columns.
- 3) When doing  $1 + 1 = 10$ , carry the 1 into the next column.

$$\begin{array}{r}
 1\ 0\ 0\ 0\ 1\ 1\ 0\ 1 \\
 +\ 0\ 1\ 0\ 0\ 1\ 0\ 0\ 0 \\
 \hline
 1\ 1\ 0\ 1\ 0\ 1\ 0\ 1 \\
 \hline
 \end{array}$$

So  $10001101 + 01001000 = 11010101$

2. Add the two 8-bit binary numbers below:

$$\begin{array}{r}
 0\ 0\ 1\ 1\ 0\ 0\ 1\ 1 \\
 +\ 0\ 1\ 1\ 1\ 1\ 0\ 0\ 1 \\
 \hline
 1\ 0\ 1\ 1\ 0\ 1\ 1\ 0\ 0 \\
 \hline
 1\ 1\ 1\ 1\ 1\ 1
 \end{array}$$

- 1) Start at the right-hand side and add each column.
- 2) Sometimes you'll get something like  $1 + 1 + 1 = 11$ , so you need to write 1, then carry 1 to the next column.

So  $00110011 + 0111001 = 10101100$

You can check your answer by converting the numbers and answer to denary, to make sure it still works.

## Overflow Errors occur when a Number has Too Many bits

- 1) Sometimes, during binary arithmetic you will get a result that requires more bits than the CPU is expecting — this is called overflow.
- 2) For example, in binary the 8-bit calculation  $1111\ 1111 + 0000\ 0001$  gives the 9-bit answer 1 0000 0000. Computers will see the 1 as an overflow error and just output 0000 0000, which is nonsense.
- 3) Computers usually deal with these extra bits by storing them elsewhere.
- 4) Overflow flags are used to show that an overflow error has occurred.

The most significant bit will be the first to overflow.

**EXAMPLE:** a) Add the 8-bit binary numbers below, giving your answer as an 8-bit binary number.

- 1) Add the binary numbers in the usual way.
- 2) The final calculation is  $1 + 1 = 10$ , so carry the 1.
- 3) You are left with a 9-bit answer — this is an overflow error.
- 4) Ignore the overflow to give your 8-bit answer. 01100101

$$\begin{array}{r}
 1\ 1\ 0\ 1\ 0\ 0\ 0\ 1 \\
 +\ 1\ 0\ 0\ 1\ 0\ 1\ 0\ 0 \\
 \hline
 1\ 0\ 1\ 1\ 0\ 1\ 0\ 1 \\
 \hline
 1\ 1
 \end{array}$$

b) Identify any problems that could be caused by giving your answer as an 8-bit number.

There is an overflow error which can lead to a loss of data and a loss of accuracy in your answer. It could also cause software to crash if it doesn't have a way of dealing with the extra bit.

## Binary, binary, quite contrary, how do you overflow...

Overflows occur when a calculation gives a result with more bits than are available to store it. This can be a real problem — programmers must make sure that they can't occur, or that they are dealt with.