REPRESENTING NUMBERS

Kira Clements, University of Bristol

DECIMAL NUMBERS

Decimal numbers are the standard numeral system used in everyday life, conveniently using the same number of digits as we have fingers!

Decimal numbers are known as **base 10**, referring to the number of different digits (0-9) and can be denoted by a subscript 10 suffix e.g. 123_{10}.

10^3	10^2	10^1	10^0	Decimal result
			7	$7\times10^0 = 7$
		8	2	$8\times10^1 + 2\times10^0 = 82$
	1	3	6	$1\times10^2 + 3\times10^1 + 6\times10^0 = 136$
4	9	2	0	$4\times10^3 + 9\times10^2 + 2\times10^1 = 4920$

Though it will likely be instinctive now, the **value** of a decimal number is calculated by the *sum of each of its digits multiplied by the corresponding power of 10*.

But the base doesn't have to be 10!

BINARY NUMBERS

Binary numbers are referred to as **base 2** as there are 2 possible digits (0, 1). Binary numbers are denoted by a subscript 2 suffix e.g. 101_2 or 0b prefix e.g. 0b101.

Each binary digit is called a **bit**, which are analogous to on/off switches i.e. they determine whether to include that power of 2 in the sum.

Most Significant Bit (MSB)

Least Significant Bit (LSB)

2^3	2^2	2^1	2^0	Decimal result
0	0	0	1	$2^0 = 1$
0	1	0	1	$2^2 + 2^0 = 4 + 1 = 5$
1	1	0	0	$2^3 + 2^2 = 8 + 4 = 12$
1	0	1	1	$2^3 + 2^1 + 2^0 = 8 + 2 + 1 = 11$

The **value** of a binary number is the *sum of the powers of 2 corresponding to the position of each 1 bit*.

HEXADECIMAL NUMBERS

Hexadecimal numbers are referred to as **base 16** as they make use of 16 digits:

0, 1, 2, 3, 4, 5, 6, 7, 8, 9, **a, b, c, d, e, f**

and are denoted by a subscript 16 suffix e.g. $7b1_{16}$ or 0x prefix e.g. 0x7b1.

16^3	16^2	16^1	16^0	Decimal result
			7	$7 \times 16^0 = 7$
		a	1	$10 \times 16^1 + 1 \times 16^0 = 160 + 1 = 161$
	3	1	0	$3 \times 16^2 + 1 \times 16^1 = 768 + 16 = 784$
5	0	0	f	$5 \times 16^3 + 15 \times 16^0 = 20480 + 15 = 20495$

The **value** of a hexadecimal number is the *sum of each of its digits multiplied by the corresponding power of 16*.

Hexadecimal is commonly used as shorthand notation for writing binary numbers…

HEXADECIMAL NUMBERS

Long binary numbers quickly become difficult to read and easy to make errors while writing out. It can be more convenient to instead write in hexadecimal, where each digit represents a 4-bit binary number:

Hexadecimal	0	1	2	3	4	5	6	7
Binary	0000	0001	0010	0011	0100	0101	0110	0111

Hexadecimal	8	9	a	b	c	d	e	f
Binary	1000	1001	1010	1011	1100	1101	1110	1111

Hexadecimal and binary are linked like this as a 4-bit binary number has a range of $2^4 = 16$ i.e. there are 16 possible 4-bit binary numbers (seen on the table above).

We therefore need 4 times fewer hexadecimal digits than binary digits to express the same value, making it much more efficient to read and write!

OCTAL NUMBERS

Octal numbers are referred to as **base 8** as they use 8 digits (0-7) and are denoted by a subscript 8 suffix e.g. 761_8 or 0o prefix e.g. 0o761. The **value** of an octal number is the *sum of each of its digits multiplied by the corresponding power of 8*.

8^3	8^2	8^1	8^0	Decimal result
			7	$7 \times 8^0 = 7$
		2	3	$2 \times 8^1 + 3 \times 8^0 = 16 + 3 = 19$
	6	0	1	$6 \times 8^2 + 1 \times 8^0 = 384 + 1 = 385$
4	0	1	0	$4 \times 8^3 + 1 \times 8^1 = 2048 + 8 = 2056$

Octal can also be considered a shorthand form of binary! Each octal digit represents a 3-bit binary number (a 3-bit binary number has a range of $2^3 = 8$).

Octal	0	1	2	3	4	5	6	7
Binary	000	001	010	011	100	101	110	111

BASE CONVERSION

As hexadecimal and octal can simply be considered shorthand for binary, it's a good idea to convert to binary before converting to another base.

Hexadecimal 5c9			
Hexadecimal	5	c	9
Binary	0101	1100	1001

Binary
010111001001

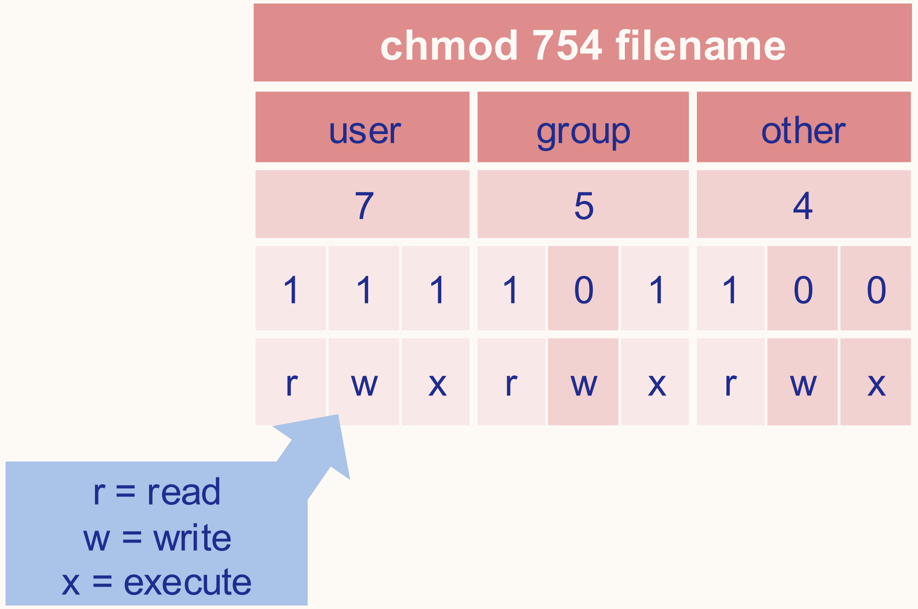
Octal 2711				
Octal	2	7	1	1
Binary	010	111	001	001

Decimal 1024 + 256 + 128 + 64 + 8 + 1 = **1481**												
Decimal	2048	1024	512	256	128	64	32	16	8	4	2	1
Binary	0	1	0	1	1	1	0	0	1	0	0	1

NOT JUST SHORTHAND

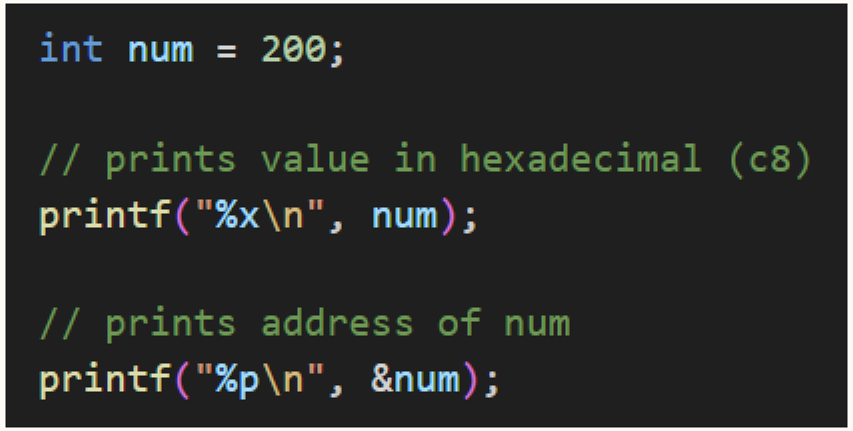
These representations have real utility within computing. For example, in Unix-like operating systems like Linux, file permissions are often displayed and set using octal notation.

Hexadecimal notation is widespread in computing to make long binary numbers human-readable, such as to represent memory addresses i.e. the location of where data is stored.



chmod 754 filename								
user			group			other		
7			5			4		
1	1	1	1	0	1	1	0	0
r	w	x	r	w	x	r	w	x

r = read
w = write
x = execute



```c
int num = 200;

// prints value in hexadecimal (c8)
printf("%x\n", num);

// prints address of num
printf("%p\n", &num);
```

BASE N VALUE

What about for a number system with **base N**, that uses D possible digits?
We can calculate the value of a number in base N using the following formula:

$$\sum_{i=0}^{D-1} x_i \cdot N^i$$

For a sequence of D digits, x_i, with positions, i, from 0 to D-1, where every digit $x_i < N$.

What's the value of 101?
A number can represent several values until we define the base/representation:

Representation	Notation	Decimal value
Decimal	101_{10}	101
Binary	101_2 or 0b101	5
Hexadecimal	101_{16} or 0x101	257
Octal	101_8 or 0o101	65

BASE N RANGE

What about the range of a number in **base N** i.e. the number of possible values?

N^D	For a sequence of D digits in base N.

Computers are finite machines that use a fixed amount of memory to store each data type.

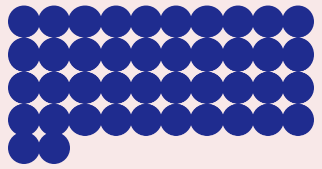
Range is an important consideration in programming, as this tells us the minimum and maximum values that we can represent with different data types.

For example, if an *unsigned int* is stored using 4 bytes (byte = 8 bits) then this can store $2^{4\times 8}$ values i.e. the binary equivalent of the decimal integers 0 to 4,294,967,295.

WHAT IS A NUMBER?

Binary numbers are the foundation of computer systems, but what can these values represent?

Quantity



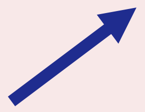
Colour

■

Character

*

Angle



Anything we want!