Number Systems

Introduction

When we count we usually use the denary number system, which is base 10. We count from 0 to 9, and then put a 1 in front and start again. Computers find it easier to work with binary, which is base 2, because it is only 0s and 1s, which can represent on or off, yes or no, or true or false,

Converting from Denary to Binary

To turn a denary number into a binary number, just put 1s in each column which is needed to make the number, using the column headings below. We usually work with 8 bits at a time, which is about 1 byte, so if the number is smaller; we need to put in leading zeros.

128	64	32	16	8	4	2	1

For example: 75_{10} in binary would be equal to 01001011_2 in binary, because 75 is the same as no lots of 128, one lot of 64, no lots of 32 or 16, one lot of 8, no lots of 4, and one lot of 2 and 1.

Converting from Denary to Octal

Octal is base 8, the principle for converting from denary to octal is similar as to that of denary to binary, although the column headings are obviously different, and you can have up to 8 lots of a number in each column.

512	64	8	1

For example 75_{10} in denary would be no lots of 512, one lot of 64, one lot of 8 and three lots of 1, so it would be 0113_8 in octal.

Converting from Denary to Hexadecimal

Some information is stored in computers as numbers in base 16, or hexadecimal sometimes just called hex. The principles are exactly the same for binary, octal, denary or any other base. But if you have to be able to count from 0 to 15, we would have to have 16 digits, as we only have ten (0 to 9) we have to use letters after that.

Hex Letter	Denary Representative
A	10
B	11
C	12
D	13
E	14
F	15

256	16	1

For example; 75_{10} in denary will be four lots of 16 and eleven lots of 1, so it is equal to $4B_{16}$ in hex.

Binary Coded Decimal

Some numbers look like numbers, but don't behave like numbers, for example you can not add together barcodes. Binary Coded Decimal (BCD) just represents four different digits in the number seperatley, using four binary digits for each denary digit.

8	4	2	1

For example; 7_{10} would be equal to 0111_2 because it is no lots of 8 and one lot of 4, 2 and 1. 5_{10} would be equal to 0101_2. So 75 in BCD would be 01110101 (just put the two together).

Negative Binary - sign magnitude

In sign magnitude method of storing negative binary numbers, the first bit, where the 128 was becomes a +/- bit, if there is a 1 here it is negative, whereas a 0 indicates that it is a positive binary number.

+/-	64	32	16	8	4	2	1

For example -75_{10} would be = to 11001011_2.

There are two problems with this method, firstly the biggest number that can be stored in one byte is half what it was, and secondly it is now storing two different types of data, a sign and a value, this makes it harder to do arithmetic operations.

Negative Binary - Twos complement

This gets round the problems with the sign magnitude method, in 2s complement the first bit of the bye becomes -128. 2s complement is very useful for addition and subtraction, because the negative part will do its self, and all we need to do is add it.

For example: -75_{10} would start with a 1, because it is a minus, we then work upwards adding more digits

-128	64	32	16	8	4	2	1

to get it up from -128 to -75. We need 53 to get up to -75 which is equal to 1 lot of 32, 16, 4 and 1. So -75 denary is equal to 10110101 in 2s complement binary.

Addition of Binary numbers

Addition in binary is much more simple than addition in denary, so computers can do it much faster, there are only four possible sums in binary:

0 + 0 = 0

0 + 1 = 1

1 + 0 = 1

1 + 1 = 0, carry 1

For example 75 + 14 in denary:

	-128	**64**	**32**	**16**	**8**	**4**	**2**	**1**	
75 =	0	1	0	0	1	0	1	1	
14 =	0	0	0	0	1	1	1	0	
	0	1	0	1	1	0	0	1	**= 89**
Carry				1	1	1			

Subtraction of Binary numbers

We use 2s complement when doing subtraction of binary numbers, (because 75 – 14 is the same as 75 + (-14)). When we do subtraction of binary nun umbers, we never do any subtraction; it's all addition

For example, 75 – 14

	-128	**64**	**32**	**16**	**8**	**4**	**2**	**1**	
75 =	0	1	0	0	1	0	1	1	
14 =	1	1	1	1	0	0	1	0	
	0	0	1	1	1	1	0	1	**= 61**
Carry	1					1			