Unit 6 Number systems

- People use the decimal system for counting.
- Computers use the binary or hexadecimal number system for counting.
- Negative numbers in binary are usually represented in two's complement form.

It is probably a consequence of the fact that we have ten distinct fingers that we use a base 10 counting system. Digital computers, however, operate on a base 2 numbering system because there are only two distinct states in a digital system, logic 0 and logic 1. It is a possibility that we could have also operated on a base 2 system if, a long time ago, we had used our two hands to count on. The difficulty is that counting efficiently beyond the limit of the base requires the concepts of zero and the concept of exponentiation of the base so it is perhaps not unreasonable to assume that the first men to count operated on a range which extended from 1 through to 10 and then followed 10 by "many".

Using base 10 and base 2 systems we then get the first few equivalences as shown in this table:

Decimal	Binary
0	0
1	1
2	10
3	11
4	100

so that the length of the number—the number of digits— increases as more and more powers of the base are introduced. In digital electronics and computing, a distinction is usually made at an early stage about the number of digits which will be used so, in an actual system, there will usually be leading zeros which pad out the number. Mechanical, rotary milometers in cars (odometers) usually leave the car showroom with a number like 000003 on the milometer and usually go to the scrap yard with a number of the order of 100000 on the milometer.

So, if we bear in mind that each digit will usually have an electronic circuit associated with it, we then get the more useful table of equivalences:

Decimal	Binary	Hexadecimal	Octal
0	00000000 00000000	0	0
1	00000000 00000001	1	1
2	00000000 00000010	2	2
3	00000000 00000011	3	3
4	00000000 00000100	4	4
5	00000000 00000101	5	5
6	00000000 00000110	6	6
7	00000000 00000111	7	7
8	00000000 00001000	8	10
9	00000000 00001001	9	11
10	00000000 00001010	A	12
11	00000000 00001011	B	13
12	00000000 00001100	C	14
13	00000000 00001101	D	15
14	00000000 00001110	E	16
15	00000000 00001111	F	17
16	00000000 00010000	10	20
17	00000000 00010001	11	21
18	00000000 00010010	12	22
218	00000000 11011010	DA	332

We have suppressed leading 0's in the decimal column in deference to our normal usage of numbers but retained them in the binary column. We have assumed that the binary column is associated with a computing system which contains a maximum of 16 bits but this is easily extended. These 0's and 1's are the natural language of digital electronics and computers but we find them singularly non memorable so two other systems are in general use for representing binary numbers. The most important and widely used is the **hexadecimal** system or base 16 system shown in the third column in which the digits 0 to 9 have their normal meaning but extra digits are used to represent the numbers 10, 11, 12, 13, 14 and 15 of the decimal system. Rather than invent new symbols for these digits, in addition to the 0 to 9 of the Arabic system, the existing characters A to F have been assigned the dual function of letters of the alphabet and hexadecimal digits which represent the hexadecimal equivalents of the decimal numbers from 10 to 15.

The difficulty is that the printed number 15 can now represent different numbers of apples, or whatever, depending on the numbering system used

so it is customary, in digital electronics and computing, to indicate the numbering system used as well as the number so that no possibility of ambiguity can arise. This is frequently done by putting a subscript D at the end of a decimal number, a subscript H at the end of a hexadecimal number and a subscript B at the end of a binary number so that we would now be able to write the equality $14_{\mathrm{H}} = 20_{\mathrm{D}} = 10100_{\mathrm{B}}$. Be warned, however, that there are other schemes used for denoting hexadecimal and binary numbers.

When there is a character A to F in the number there is no ambiguity but it is still good practice to be consistent in denoting the number system in use.

In the table, the fourth column shows the equivalent number in the octal or base 8 system. This was used in the early computer systems but has now been largely superseded and we will not consider it further.

We now have three systems in common usage and it is necessary to be able to convert number from one base to another.

Conversion from binary to hexadecimal is easy. Just group the binary digits in groups of four starting from the least significant digit and replace each group by its hexadecimal equivalent digit.

Conversion from hexadecimal to binary is similar. Replace each hexadecimal digit by its equivalent four binary digits. An example of these conversions would be $110\ 1100\ 1001_{\mathrm{B}} = 6\mathrm{C}9_{\mathrm{H}}$.

Conversions from hexadecimal to decimal are not as simple and require some arithmetic operations. The conversion is best done by remembering that a hexadecimal number such as $1A09_{\mathrm{H}}$ is equivalent to $1 \times 16^3 + 10 \times 16^2 + 0 \times 16^1 + 9 \times 16^0 = 6665$. So the rule is that the position of the digit indicates the power of the base which multiplies the digit.

Conversion from decimal to hexadecimal is the least straightforward. The conversion is performed by repeated division by 16. Take the example of 1357_{D} which is to be converted to hexadecimal. Successively divide 1357 by 16 noting the remainder at each stage.

$$\begin{array}{rl} 16\,)\,1357 & \\ \underline{} & \\ 16\,)\,84 & 13 = \mathrm{D} \\ \underline{} & \\ 16\,)\,5 & 4 = 4 \\ \underline{} & \\ 0 & 5 = 5 \end{array}$$

So the equivalence is $1357_{\mathrm{D}} = 54\mathrm{D}_{\mathrm{H}}$.

While these conversions are straightforward, they can be tedious if the numbers are large so it is worth while investing in a model of calculator which has the option of hexadecimal, binary and decimal mode operation.

This also has the advantage that it obviates the need to learn hexadecimal multiplication and division tables!

In the binary representation of numbers there are only the two symbols 0 and 1. In the decimal representation we have the symbols 0 to 9 and also the positive and negative signs, + and − and we can represent negative numbers by putting a − sign in front of the number. There is no extra sign available in binary which can be used to denote a negative number so we must denote negative numbers in some way by using the symbols 0 and 1.

The obvious approach is to designate one of the positions in the binary number as the sign position and then use 0 and 1 to represent the + and − signs. This then gives us a number which cannot be distinguished from an unsigned binary number. This proves to be an unsatisfactory approach if we require that a number obey the usual arithmetic rules and in particular the rule that a number and its negative should add to give a result of zero, that is $X + (-X) = 0$.

For example $+3A_H = 00111010_B$ and $-3A_H = 10111010_B = BA_H$. Addition then gives the undesired non zero result

+3A		00111010	
-3A	and	10111010	
+00		11110100	which is F4

An alternative approach is to use the two's complement notation for negative numbers. A 1's complement of a binary number is obtained by replacing all of the 0's by 1's and the 1's by 0's. The 2's complement is then obtained by adding 1 to the 1's complement.

Before obtaining the 1's complement it is necessary to specify the number of possible digits in the number and the leading 0's must be inserted. So in a 16 bit numbering system, the 2's complement of $26D_H$ is obtained as follows:

26D	=	0000 0010 0110 1101
1's complement	=	1111 1101 1001 0010
add 1		1
2's complement	=	1111 1101 1001 0011

Now let us apply the test $X + (-X) = 0$

+26D	=		0000 0010 0110 1101
-26D	=		1111 1101 1001 0011
0000	=	1	0000 0000 0000 0000

The carry is now discarded since wraparound has occurred, that is the result has exceeded the available size of the number and the stored number has cycled through zero.

If you try this on your calculator when it is in hexadecimal mode you will see that the binary or hexadecimal number will immediately be padded out with 1's or F's when you operate the change sign, $\pm$, button and when you add this negative number to the original number you will get the result of zero.

This wraparound effect is illustrated in the diagram in Figure 6.1 in which the available range or space for the number is spread around the circumference of a circle. Starting from the zero line, marked with a 0, the numbers on the circle increase as you move anti clockwise around the circle. In the present example of a 16 binary bit number you will see that the number 1000000000000 is at the position at 180° to the origin. The number increases until 1111111111111111 just below the origin. Addition of 1 to this number gives 0000000000000000 and there is no mechanism for recording that the count has moved through the origin so the origin and one full rotation are indistinguishable and wraparound is said to have occurred. Points on the top half of this circle represent the positive numbers. Points on the bottom half, which have a leading 1, represent negative numbers.

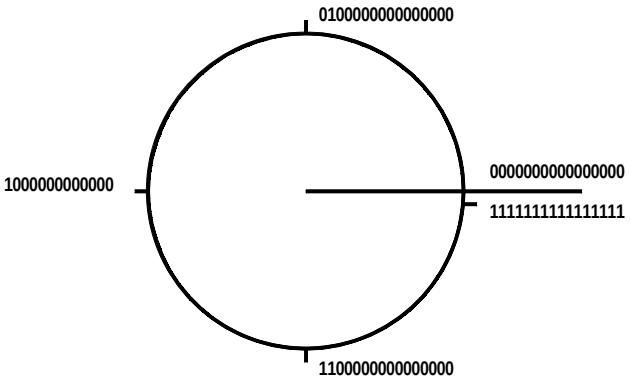

Figure 6.1:

6.1 Problems

6.1 Calculate the binary equivalents of 19_D, 1389_D, 320_H, $1AD_H$, $FF23_H$.

6.2 Calculate the decimal equivalents of 10101010_B 01010101_B, 1100110011000_B, FF_H, 100_H, $957CD_H$.

6.3 Calculate the hexadecimal equivalents of 256_D, 65536_D, 428343295_D, 1111111111111_B.

6.4 Form the two's complement of 101110110_B, 4CFD_H, 1392_D.

6.5 Calculate the binary equivalents of 5555_H and AAAA_H.

6.6 What is the largest decimal number which can be represented by 32 binary bits?

6.7 Many microcontrollers (single chip computers used in devices such as mobile phones, vending mahines, hand held instruments, etc.) do not have a program instruction for multiplication so the conversion of numbers from hexadecimal to decimal cannot use the algorithm described in this unit. The algorithm used is to successivly subtract the hexadecimal equivalent of the most significant decimal digit and count the number of subtractions necessary before a negative answer is obtained. Note that subtraction is implemented by adding the negative of a number. Convert the output of a 12 Bit Analog to Digital converter from the three hexadecimal digits to the four decimal digits by successive additions of $-1000_D = C18_H$, $-100_D = F9C_H$ and $-10_D = FF6_H$. As an example, convert $3A6_H$ to decimal using this repetitive method. (The answer is 934_D).