Number Systems

Eric McCreath

Decimal

Decimal is the most commonly used number system.

We are so familiar with decimal that we can both understand what numbers represent and perform operations on decimal numbers without much thought.

Decimal is base 10 and includes the digits:

$$0, 1, 2, 3, 4, 5, 6, 7, 8, 9$$

A decimal number can be partitioned into powers of 10.

$$167_{(10)} = 1 \times 10^2 + 6 \times 10^1 + 7 \times 10^0$$

$$3.14 = 3 \times 10^0 + 1 \times 10^{-1} + 4 \times 10^{-2}$$

Decimal

More generally:

$$...d_2d_1d_0.d_{-1}d_{-2}... = \sum_i d_i \times 10^i$$

Partitioning a decimal into its powers of 10 gives us a simple way of systematically and algorithmically performing operations (such as + - * /) on these numbers.

How would you calculate the following (without a calculator)?

$$158 + 345$$

Or calculate?

$$15 * 32$$

Octal

Octal is base 8. As 8 is a power of 2, converting between octal and binary is simple.

The digits of octal are:

$$0, 1, 2, 3, 4, 5, 6, 7$$

To write an octal number in C, just prefix the value with a "0". e.g. 010 is octal (in decimal the number 8)

The digits of an octal number can be understood in terms of powers of 8.

$$103_{(8)} = 1 \times 8^2 + 0 \times 8^1 + 3 * 8^0 = 67_{(10)}$$

Octal

More generally:

$$...d_2d_1d_0.d_{-1}d_{-2}... = \sum_i d_i \times 8^i$$

The above formula can be used to convert octal numbers to decimal.

Convert $42_{(8)}$ to decimal.

Octal

Converting from decimal to octal is a little more tricky. However, again we make use of the fact that each digit is a different power of 8.

The basic approach is to repeatedly divide the number by 8 and keep track of the remainder. This process is repeated until 0 is reached. Now the remainder values become the digits of the number converted to octal.

Say we wish to convert 132 (in decimal) to octal.

$$\begin{aligned} 132/8 &= 16 \text{ remainder } 4 \\ 16/8 &= 2 \text{ remainder } 0 \quad \Rightarrow 132_{(10)} = 204_{(8)} \\ 2/8 &= 0 \text{ remainder } 2 \end{aligned}$$

Try converting 42 (in decimal) to octal.

Hexadecimal

Hexadecimal is base 16 and like octal provides a simple way of converting numbers between hexadecimal and binary.

The digits of hexadecimal are:

$$0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F$$

In C hexadecimal numbers begin with "0x". e.g. 0x1B is the number 27 in decimal.

Hexadecimal is very commonly used in computers systems. Not only is the conversion to binary simple but it represents words nicely.

Hexadecimal

Conversion is similar to that of octal. Remember also that:

$$...d_2 d_1 d_0 . d_{-1} d_{-2} ... = \sum_i d_i \times 16^i$$

Converting hexadecimal to decimal example:

$$10E_{(16)} = 1 \times 16^2 + 0 \times 16^1 + 14 \times 16^0 = 270_{(10)}$$

Convert 0x4A (in hexadecimal) to decimal.

Converting decimal to hexadecimal example:

$$\begin{aligned} 174/16 &= 10 \text{ remainder } 14 \\ 10/16 &= 0 \text{ remainder } 10 \end{aligned} \Rightarrow 174_{(10)} = AE_{(16)}$$

Convert 42 (decimal) to hexadecimal.

Binary

Binary is base 2 and has digits:

$$0, 1$$

The meaning of digits:

$$...d_2 d_1 d_0 . d_{-1} d_{-2} ... = \sum_i d_i \times 2^i$$

Converting from binary to decimal and from decimal to binary follows the same pattern of both octal and hexadecimal.

Convert 1101 (in binary) to decimal.

Convert 42 (in decimal) to binary.

Binary

To convert from binary to hex one can simply group lots of 4 binary digits (align this with the least most significant digit and pad the left with zeros if needed) and then replace those binary digits with the hexadecimal number for that binary number.

$$1111010 \rightarrow (0111)(1010) \rightarrow 0x7A$$

Converting hex to binary is similar.

$$0xB1 \rightarrow (1011)(0001) \rightarrow 10110001$$

Converting between binary and octal is similar to the above. The only difference is that you use groups of 3 binary digits.

Converting between octal and hex is most simply done via binary (going via decimal is often a lot more work).

Exercise

Convert some numbers between the different representations by hand. Write a C program to check you have the correct results.

Write a C program that is given an integer on standard input (just use scanf) and outputs to the standard output the number in binary. Use the divide and remainder approach.