Bases and Number Representation

Reading: Chapter 2 (14 – 27) from the text book

Real Numbers and the Decimal Number System

- *Real Numbers* – are the familiar numbers of everyday life. Important types are:

  Natural numbers : 1, 2, 3, 4, 5, ….

  Integers : 0, 1, –1, 2, –2, 3, –3, ….

  Rational numbers : can be written as m/n, where m,n are integers and n is not 0

  e.g. 2/5, –13/721

  Irrational numbers : are the real numbers that aren't rational

Rational & Irrational Numbers

- Every rational no. can be written as either a terminating decimal (e.g. 1¾ = 1.75) or as a recurring decimal (e.g. 2/3 = 0.666666…., 2/7 = 0.285714285714285714….)
- The irrational nos are the real nos whose decimal expansions neither terminate nor recur. Examples include:

  √2 = 1.414213562373095048801688 72….

  π = 3.14159265358979323846264338

Place Value and Base

- A number such as 6245.37 is in decimal form, with each digit having a *place value*

  Example : The place value of the digit 6 in 6245.37 is 1000 = 10^3

- Expanded form : $6245.37 = 6\times10^3 + 2\times10^2 + 4\times10 + 5\times10^0 + 3\times10^{-1} + 7\times10^{-2}$

- In decimal form, place values are powers of 10 so the decimal system is said to have a *base* of 10

Note : base 10 requires *ten* digits (i.e. 0–9)

The Binary Number System

- Simplest number system is base 2, or *binary*
  uses the 2 digits (“bits”) 0 and 1
- Used exclusively in computers (ON/OFF switches, magnetised/unmagnetised memory elements)
- A typical binary number is 1011.101_2
- The subscript 2 denotes the base – the base should be included if it is not 10

Converting Binary to Decimal

- Example: Convert 1011.101_2 to decimal
- Solution: 11.625

  $= (1\times2^3) + (0\times2^2) + (1\times2^1) + (1\times2^0)$

  $+ (1\times2^{-1}) + (0\times2^{-2}) + (1\times2^{-3})$

  $= 8 + 2 + 1 + 0.5 + 0.125$

  $= 11.625$
- Exercise: Convert 110001.011_2 to decimal

Conversion from Decimal to Binary

- We'll begin by converting *integers*
- Example: Convert 183 to binary

  Solution: Note that the powers of 2 are

  1, 2, 4, 8, 16, 32, 64, 128, 256, 512, ….

  Now write 183 using just these powers.

  Thus 183 = 128 + 55

  = 128 + 32 + 23 = 128 + 32 + 16 + 7

  = 128 + 32 + 16 + 4 + 2 + 1 = 10110111_2

Decimal to Binary - A Better Way

- Previous method is not suitable for large nos.
- A better method is to repeatedly divide by 2, writing down the *quotient* and *remainder* at each step, until the quotient is zero
- Now write down the remainders in *reverse* order – this is the binary form of the integer
- Example : Convert 212 to binary
  Answer: 212 = 11010100_2
- Exercise : Convert 183 to binary

Example

2	212	
	106	0
	53	0
	26	1
	13	0
	6	1
	3	0
	1	1
	0	1

Thus 212 = 11010100_2

Decimal Fractions to Binary

- The halves bit in the representation of 0.4 is 0, because 0.4 is *less* than 0.5
- Expressed another way, the halves bit is 0 because 2×0.4 is *less* than 1
- Similarly, the halves bit in the binary form of 0.6 is 1, because 2×0.6 is *greater* than 1
- So: The halves bit in the binary representation of the decimal fraction *n* is the integer part of 2*n*

Method for Converting Fractions

- The method for calculating the halves bit can be extended to find the *complete* binary representation of a decimal fraction
- Method: Repeatedly multiply the fractional part by 2, writing down the *integer* and *fraction* at each step, until the fraction is 0
- Now write down the integers in order – this is the binary form of the fraction

Example of Converting a Fraction

Example: Convert 0.6875 to binary

Solution: Set out the calculations as shown:

	6875	2
1	3750	
0	7500	
1	5000	
1	0000	

Thus 0.6875 = 0.1011_2

Non-terminating Binary Fractions

- When using the method to change a decimal fraction to binary, the fractional part might never be 0 – so the method may never end
- This means that a binary fraction might not terminate (the same thing occurs with some decimal fractions – e.g. 2/3 = 0.666666…)
- Example: $0.4 = 0.01100110\ldots_2$ (truncated to 8 digits after the point)

Converting Decimal to Binary

- Exercise: Convert 0.65 to binary (truncate to 6 bits after the point)

  Answer: 0.65 = $0.101001\ldots_2$

- Exercise: Convert 35.65 to binary

  (Hint: Separately convert the integer and fractional parts of the number)

  Answer: 35.65 = $100011.101001\ldots_2$

The Octal and Hexadecimal Systems

- The methods for converting from binary (base 2) to decimal, and vice-versa, extend to bases other than 2

- The base 8 number system is known as *octal* we'll look at number conversions, then at why octal is useful in computing

- Octal is based on the digits 0–7; the place values are powers of 8

Converting Octal to Decimal

- Example: 253.64_8

  $= (2\times8^2) + (5\times8^1) + (3\times8^0) + (6\times8^{-1}) + (4\times8^{-2})$

  $= 128 + 40 + 3 + 0.75 + 0.0625$

  $= 171.8125$

- Exercise: Convert 172.4_8 to decimal

  Answer: $172.4_8 = 122.5$

Converting Decimal to Octal

- Example: Convert 103.6 to octal

  Solution: This is done in two parts:

  Convert 103 by repeated *division* by 8;

  Convert 0.6 by repeated *multiplication* by 8.

  Now combine the results, so

  $103.6 = 147.46314\ldots_8$

- Exercise: Convert 59.5625 to octal

  Answer: $59.5625 = 73.44_8$

Why is Octal Important?

- Computers use binary numbers exclusively
- However, binary numbers often have many digits – e.g. $9000_{10} = 10001100101000_2$
- Decimal uses fewer digits than binary, but conversions between the bases are awkward
- Octal has two advantages:
  - a fairly large base (so not too many digits)
  - easy to convert between octal and binary (as we'll see in the next slides)
- Thus: *octal is a good shorthand for binary*

Conversion from Binary to Octal

Example: Convert 11001011101.1101101_2 to octal

Method: Group the bits into sets of three on either side of the point, adding extra zeros on the right if required to complete a set of three bits. Then convert each set of 3 bits to a single octal digit.

Answer: 3135.664_8

A Problem with Octal

- Computers work in *bytes*, where 1 byte is equal to 8 bits
- Thus a single byte can represent the numbers from 00000000_2 to 11111111_2 (i.e. 0 to 255)
- However, if a byte is written in octal, there are only 2 bits on the left, so the first digit can't exceed 3 – i.e. it is never 4, 5, 6 or 7
- This will waste space – i.e. octal doesn't *efficiently* represent a byte

The Hexadecimal System

- The base 16 number system has the advantages of octal (i.e. a relatively large base, and easy conversions with binary) and it also efficiently represents a byte
- Base 16 system is called *hexadecimal* ('hex')
- Hex uses 16 digits – the familiar 0-9, and the upper-case letters A-F for 10-15, representation.
- It is now the preferred shorthand for binary

To Convert Between Hex & Decimal

- Example: Write $3AB.C_{16}$ in decimal form

  Solution: $3AB.C_{16}$

  $= (3\times16^{2}) + (10\times16^{1}) + (11\times16^{0}) + (12\times16^{-1})$

  $= 768 + 160 + 11 + 0.75 = 939.75$

- Example: Convert 730.203125 to hex

  Solution: Convert 730 using repeated division by 16, and 0.203125 by repeatedly multiplying by 16 – the answer is $2DA.34_{16}$

To Convert Between Binary & Hex

- The method is the same as for conversions between binary & octal, except that bits are grouped into sets of *four* (rather than three)
- Example: Convert 10110110011.0111101_2 to hexadecimal

  Answer: $5B3.7A_{16}$
- Example: Convert $3E7.B4_{16}$ to binary

  Answer: 1111100111.101101_2

An Application of Hex

- Hex can be used to specify colours in HTML,the language of the web
- Each colour is specified by a 6-digit hex no.
- The 1st 2 digits give the amount of Red (on a scale of 00-FF, or 0-255 in decimal), the next 2 are for Green, and the final 2 specify Blue
- Thus any one of 16,777,216 (= 256^3) colours can be obtained by a suitable mix of these 3 *primary colours*
- Black is specified as 000000, and White as FFFFFF

Example of Colour Specification

- The HTML code

```
<font color="#0000FF">DISCRETE</font>
<font color="#800000">MATHS</font>
<font color="#FF00FF">2008</font>
```

  produces DISCRETE MATHS 2008 in a browser
- Here 'DISCRETE' is in blue, 'MATHS' is in maroon (= medium red), and '2008' is in fuchsia (also called magenta) because red + blue = fuchsia

Arithmetic in Non-Decimal Bases

- The familiar methods used to add, subtract, multiply & divide numbers in the decimal system can be extended to other bases

- We'll concentrate on arithmetic in *binary* similar methods apply for other bases

Addition in binary

- The basic addition table is easy to write down

+	0	1
0	0	1
1	1	10

- In general, 2 binary nos are added in the usual column-by-column way, carrying a '1' to the next column on the left if necessary
- Example: $1101_2 + 101_2 = 10010_2$
- Exercise: Calculate $101101_2 + 10111_2$

  Answer: 1000100_2

Subtraction in Binary

- Example: $200514_{10} - 46732_{10}$
- The method for subtracting decimal nos, column-by-column from right to left, is also used for subtracting binary nos
- Example: $11011_2 - 1101_2 = 1110_2$
- Exercise: $10010_2 - 1011_2$

  Answer: 111_2

Multiplication in Binary

- The basic table is very easy to write down

×	0	1
0	0	0
1	0	1

- The usual method of 'long multiplication' for decimal nos applies also to binary nos
- Example: $10111_2 \times 1101_2 = 100101011_2$
- Exercise: Calculate $1011_2 \times 1010_2$

  Answer: 1101110_2

Division in Binary

- Again, the usual method for 'long division' applies to division of binary nos

- Example: $11101_2 \div 110_2 = 100.110\ldots._2$

- Note that at each step of the division, the divisor 110_2 'goes into' the number either once (if it is less than or equal to the number) or zero times (if it is greater than the number)