

## Data Structures And Number Systems

© Copyright Brian Brown, 1984-2000. All rights reserved.

This courseware uses HTML 3.0 extensions

### Introduction

A number system defines a set of values used to represent quantity. We talk about the number of people attending class, the number of modules taken per student, and also use numbers to represent grades achieved by students in tests.

Quantifying values and items in relation to each other is helpful for us to make sense of our environment. We do this at an early age, figuring out if we have more toys to play with, more presents, more lollies and so on.

The study of number systems is not just limited to computers. We apply numbers every day, and knowing how numbers work will give us an insight into how a computer manipulates and stores numbers.

Mankind through the ages has used signs or symbols to represent numbers. The early forms were straight lines or groups of lines, much like as depicted in the film *Robinson Crusoe*, where a group of six vertical lines with a diagonal line across represented one week.

Its difficult representing large or very small numbers using such a graphical approach. As early as 3400BC in Egypt and 3000BC in Mesopotamia, they developed a symbol to represent the unit 10. This was a major advance, because it reduced the number of symbols required. For instance, 12 could be represented as a 10 and two units (three symbols instead of 12 that was required previously).

The Romans devised a number system which could represent all the numbers from 1 to 1,000,000 using only seven symbols

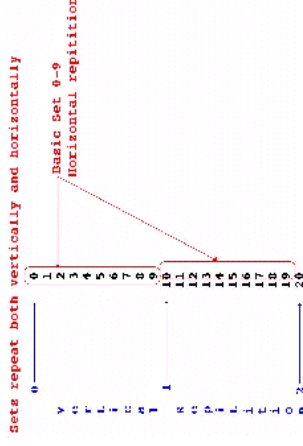
- I = 1
- V = 5
- X = 10
- L = 50
- C = 100
- D = 500
- M = 1000

A small bar placed above a symbol indicates the number is multiplied by 1000.

The number system in most common use today is the *Arabic* system. It was first developed by the Hindus and was used as early as the 3rd century BC. The introduction of the symbol 0, used to indicate the positional value of digits was very important. We thus became familiar with the concept of groups of units, tens of units, hundreds of units, thousands of units and so on.

In number systems, its often helpful to think of **recurring sets**, where a set of values is repeated over and over again.

Considering the decimal number system, it has a set of values which range from 0 to 9. **This basic set is repeated over and over, creating large numbers.**



Note how the set of values 0 to 9 is repeated, and for each repeat, the column to the left is incremented (from 0 to 1, then 2).

Each increase in value occurs, till the value of the largest number in the set is reached (9), at which stage the next value is the smallest in the set (0) and a new value is generated in the left column (the next value after 9 is 10).

09, 10 - 19, 20 - 29, 30 -39 etc

**We always write the digit with the largest value on the left of the number**

### Base Values

The base value of a number system is the number of different values the set has before repeating itself. For example, decimal has a base of ten values, 0 to 9.

- Binary = 2 (0, 1)
- Octal = 8 (0 - 7)
- Decimal = 10 (0 - 9)
- Duodecimal = 12 (used for some purposes by the Romans)
- Hexadecimal = 16 (0 - 9, A-F)
- Vigesimal = 20 (used by the Mayans)
- Sexagesimal = 60 (used by the Babylonians)

### Weighting Factor

The weighting factor is the multiplier value applied to each column position of the number. For instance, decimal has a weighting factor of TEN, in that each column to the left indicates a

multiplication value increase of 10 over the previous column on the right, ie; each column move to the left increases in a multiply factor of 10.

$$\begin{array}{r}
 200 = \\
 \text{-----} 0 * 10^0 = 0 * 1 = 0 \\
 \text{-----} 0 * 10^1 = 0 * 10 = 0 \\
 \text{-----} 2 * 10^2 = 2 * 100 = 200 \\
 \text{-----} \\
 200 \text{ (adding these up)}
 \end{array}$$

Lets consider another example of 312 decimal.

$$\begin{array}{r}
 312 = \\
 \text{-----} 2 * 10^0 = 2 * 1 = 2 \\
 \text{-----} 1 * 10^1 = 1 * 10 = 10 \\
 \text{-----} 3 * 10^2 = 3 * 100 = 300 \\
 \text{-----} \\
 312 \text{ (adding these up)}
 \end{array}$$

## Decimal Number System [Base-10]

This number system uses TEN different symbols to represent values. The set values used in decimal are

0 1 2 3 4 5 6 7 8 9

with 0 having the least value and nine having the greatest value. The digit or column on the left has the greatest value, whilst the digit on the right has the least value.

When doing a calculation, if the highest digit (9) is exceeded, a carry occurs which is transferred to the next column (to the left).

### Example of addition and exceeding the base set range

$$\begin{array}{r}
 8 + 4 \\
 \text{-----} \\
 8 \quad +1 \\
 10 \quad +2 \quad \text{Note 1:} \\
 11 \quad +3 \\
 12 \quad +4
 \end{array}$$

Note1: When 9 is exceeded, we go back to the beginning of the set (0), and carry a value of 1 over to the next column on the left.

### Another example of addition and exceeding the base set range

$$\begin{array}{r}
 198 + 4 \\
 \text{-----} \\
 198 \\
 199 \quad +1
 \end{array}$$

$$\begin{array}{r}
 200 \quad +2 \quad \text{Note 2:} \\
 201 \quad +3 \\
 202 \quad +4
 \end{array}$$

Note2: When 9 is exceeded, we go back to the beginning of the set (0), and carry a value of 1 over to the next column on the left. Thus the middle column (9) has 1 added to it, the next value in the set is 0, and we carry 1 (because the set was exceeded) to the next left column. Adding the carry value of 1 to 1 in the left most column gives.

## Positional Values [Units, Tens, Hundreds, Thousands etc Columns]

We probably got taught at school about positional values, in that columns represent powers of 10. This is expressed to us as columns of ones (0 - 9), tens (groups of 10), hundreds (groups of 100) and so on.

$$\begin{array}{r}
 237 = (2 \text{ groups of } 100) + (3 \text{ groups of } 10) + (7 \text{ groups of } 1) \\
 = (100 + 100) + (10 + 10 + 10) + (1 + 1 + 1 + 1 + 1 + 1 + 1) \\
 = (200) + (30) + (7) \\
 = 237
 \end{array}$$

Each column move to the left is 10 times the previous value.

## Binary Number System [Base-2]

The binary number system uses TWO values to represent numbers. The values are,

- 0
- 1

with 0 having the least value, and 1 having the greatest value. Columns are used in the same way in the decimal system, in that the left most column is used to represent the greatest value.

As we have seen in the decimal system, the values in the set (0 and 1) repeat, in both the vertical and horizontal directions.

$$\begin{array}{r}
 0 \\
 1 \\
 10 \quad \text{Note: goto value lowest in set, carry to left} \\
 11 \\
 100 \quad \text{Note: goto value lowest in set, carry to left} \\
 101 \\
 110 \quad \text{Note: goto value lowest in set, carry to left} \\
 111
 \end{array}$$

In a computer, a binary variable capable of storing a binary value (0 or 1) is called a BIT.

In the decimal system, columns represented multiplication values of 10. That was because there were 10 values (0 - 9) in the set. In this binary system, there are only two values (0 - 1) in the set, columns represent multiplication values of 2.

$1011 =$   
 $1 * 2^0 = 1$   
 $1 * 2^1 = 2$   
 $0 * 2^2 = 0$   
 $1 * 2^3 = 8$   
 $11$  (in decimal)

**Number Ranges in Binary Using A Specified Number Of Bits**

How many different values can be represented by a specific number of bits?

number of different values =  $2^n$

where  $n$  is the number of bits

eg.

$2^8$   
 $= 256$  different values

**Rules For Binary Addition**

Operation	Result
0 + 0	0
0 + 1	1
1 + 0	1
1 + 1	0 and Carry 1

$1011 + 101 =$   
 $1011$   
 $101$

1. Start at the right most column and apply the rules.
2.  $1 + 1$  is 0 and carry 1 onto the next column on the left

$1011$   
 $101$   
 $0$  and carry 1

which really looks like

$1011$   
 $111$   
 $0$

3. Now do the second column
4.  $1 + 1$  is 0, carry 1 onto the next column on the left

$1011$   
 $111$   
 $00$

$00$  and carry 1

which really looks like

$1011$   
 $111$   
 $1$   
 $00$

5. Now do the third column
6.  $1 + 1$  is 0, carry 1 onto the next column on the left

$1011$   
 $111$   
 $1$   
 $000$  and carry 1

which really looks like

$1011$   
 $111$   
 $1$   
 $000$

7. Now do the last column on the left
8.  $1 + 1$  is 0 and carry 1 to the left.

$1011$   
 $101$   
 $10000$

**Rules For Binary Subtraction**

Operation	Result
0 - 0	0
0 - 1	1 and borrow 1
1 - 0	1
1 - 1	0

**Rules For Binary Multiplication**

Operation	Result
0 * 0	0
0 * 1	0
1 * 0	0
1 * 1	1

**Sample Problems for Binary Addition and Subtraction**

## Converting Decimal to Binary

There are a number of ways to convert between decimal and binary. Lets start with converting the decimal value **254** to binary.

**Method 1:** Divide the number by 2, then divide what's left by 2, and so on until there is nothing left (0). Write down the remainder (which is either 0 or 1) at each division stage. Once there are no more divisions, list the remainder values in reverse order. This is the binary equivalent.

```

254 / 2 giving 127 with a remainder of 0
127 / 2 giving 63 with a remainder of 1
63 / 2 giving 31 with a remainder of 1
31 / 2 giving 15 with a remainder of 1
15 / 2 giving 7 with a remainder of 1
7 / 2 giving 3 with a remainder of 1
3 / 2 giving 1 with a remainder of 1
1 / 2 giving 0 with a remainder of 1
    
```

thus the binary equivalent is **11111110**

**Another example, 132 decimal**

```

132 / 2 giving 66 with a remainder of 0
66 / 2 giving 33 with a remainder of 0
33 / 2 giving 16 with a remainder of 1
16 / 2 giving 8 with a remainder of 0
8 / 2 giving 4 with a remainder of 0
4 / 2 giving 2 with a remainder of 0
2 / 2 giving 1 with a remainder of 0
1 / 2 giving 0 with a remainder of 1
    
```

thus the binary equivalent is **10000100**

**Method 2:** Each column represents a power of 2, so use this as a basis of calculating the number. It is sometimes referred to as the 8:4:2:1 approach.

Write down the binary number. Where a **1** appears in the column, add the column value as a power of 2 to a total.

<b>Weighting</b>	8	4	2	1	<b>Answer</b>		
<b>Binary Value</b>	1	0	1	1	<b>11</b>		
<b>Weighting</b>	8	4	2	1	<b>Answer</b>		
<b>Binary Value</b>	0	1	1	1	<b>7</b>		
<b>Weighting</b>	32	16	8	4	2	1	<b>Answer</b>
<b>Binary Value</b>	1	1	1	0	1	1	<b>59</b>
<b>Weighting</b>	32	16	8	4	2	1	<b>Answer</b>
<b>Binary Value</b>	1	0	1	0	1	0	<b>42</b>

### Sample Problems for Decimal to Binary Conversion

Binary numbers are

- cumbersome to write down
- long
- not very meaning to the average user
- are understood by computers

## Octal Number System [Base-8]

The octal number system uses EIGHT values to represent numbers. The values are,

0 1 2 3 4 5 6 7

with 0 having the least value and seven having the greatest value. Columns are used in the same way as in the decimal system, in that the left most column is used to represent the greatest value.

As we have seen in the decimal system, the values in the set (0 and 1) repeat, in both the vertical and horizontal directions.

0 - 7, 10 -17, 20 - 27, 30 - 37 .....

**Problem:** Convert 176 in octal to decimal.

Each column represents a power of 8,

```

176 =
----- 6 * 80 = 6
----- 7 * 81 = 56
----- 1 * 82 = 64
-----
126
    
```

Octal was used extensively in early mainframe computer systems.

## Hexadecimal Number System [Base-16]

The hexadecimal number system uses SIXTEEN values to represent numbers. The values are,

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

with 0 having the least value and F having the greatest value. Columns are used in the same way as in the decimal system, in that the left most column is used to represent the greatest value.

As we have seen in the decimal system, the values in the set (0 and 1) repeat, in both the vertical and horizontal directions.

0 - F, 10 -1F, 20 - 2F, 30 - 3F .....

Hexadecimal is often used to represent values [numbers and memory addresses] in computer

systems.

### Decimal - Binary - Hexadecimal

Decimal	Binary	Hexadecimal
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

### Converting hexadecimal to decimal

**Problem:** Convert 176 in hexadecimal to decimal.

Each column represents a power of 16,

$$\begin{array}{r}
 176 = \\
 \text{----} 6 * 16^0 = 6 \\
 \text{-----} 7 * 16^1 = 112 \\
 \text{-----} 1 * 16^2 = 256 \\
 \text{-----} \\
 \phantom{176 = } 374
 \end{array}$$

### Converting binary to hexadecimal

**Problem:** Convert 10110 to hexadecimal.

Each hexadecimal digit represents 4 binary bits. Split the binary number into groups of 4 bits, starting from the right.

$$\begin{array}{r}
 1 \phantom{0000} \\
 =1 \phantom{0000} \\
 =16 \text{ in hexadecimal}
 \end{array}$$

### Converting decimal to hexadecimal

**Problem:** Convert 232 decimal to hexadecimal.

Use the same method used earlier to divide decimal to binary, but divide by 16.

$$\begin{array}{l}
 232 / 16 = 14 \text{ with a remainder of } 8 \\
 14 / 16 = 0 \text{ with a remainder of } 14 \text{ (14 decimal = E)} \\
 = \mathbf{E8}_{16}
 \end{array}$$

**To avoid confusion, we often add a suffix to indicate the number base**

- 162<sub>h</sub> h means hexadecimal
- 162<sub>16</sub> 16 means base 16
- 162<sub>d</sub> d means decimal
- 162<sub>10</sub> 10 means base 10
- 162<sub>o</sub> o means octal
- 162<sub>8</sub> 8 means base 8
- 101<sub>b</sub> b means binary
- 101<sub>2</sub> 2 means base 2

### Sample Problems for Hexadecimal Conversion

## Representing Positive and negative Numbers in Binary

When a number of bits is used to store values, the **most significant bit** [the bit which has the greatest value, in the left most column] is used to store the sign [positive or negative] of the number. The remaining bits hold the actual value.



If the number is negative, the sign is **1**, and for positive numbers, the sign is **0**.

**Question:** *What is the range of numbers available when 8 bits are used.*

For 8 bits, one bit is for the sign, 7 for the number, so the range of  $2^7 = 127$  combinations

Because of problems doing addition and subtraction, negative numbers are usually stored in a different format to positive numbers.

More information on representing numbers

## Ones Complement

1's complement is a method of storing negative values. It simply inverts all 0's to 1's and all 1's to 0's.

Original Number	Binary value	1's Complement value
7	00000111	11110000
32	00100000	11011111
114	01110010	10001101

## Twos Complement

2's complement is another method of storing negative values. It is obtained by adding 1 to the 1's complement value.

Original Number	Binary value	1's Complement value	2's Complement value
7	00000111	11110000	11110001
32	00100000	11011111	11100000
114	01110010	10001101	10001110

Another way of generating a 2's complement number is to start at the least significant bit, and copy down all the 0's till the first 1 is reached. Copy down the first 1, then invert all the remaining bits.

The following table depicts both 1's and 2's complement using a range of 4 bits.

**Table of Complements**

Binary	1's Complement	2's Complement	Unsigned
0111	7	7	7
0110	6	6	6
0101	5	5	5
0100	4	4	4
0011	3	3	3
0010	2	2	2
0001	1	1	1
0000	0	0	0
1111	-0	-1	15
1110	-1	-2	14
1101	-2	-3	13
1100	-3	-4	12
1011	-4	-5	11
1010	-5	-6	10
1001	-6	-7	9
1000	-7	-8	8

**Note:** See how in the 1's complement case there are two representations for 0

## Gray Code

This is a *variable weighted code* and is cyclic. This means that it is arranged so that every transition from one value to the next value involves only **one bit change**.

The gray code is sometimes referred to as **reflected binary**, because the first eight values compare with those of the last 8 values, but in reverse order.

Decimal	Binary	Gray
0	0000	0000
1	0001	0001
2	0010	0011
3	0011	0010
4	0100	0110
5	0101	0111
6	0110	0101
7	0111	0100
8	1000	1100
9	1001	1101
10	1010	1111
11	1011	1110
12	1100	1010
13	1101	1011
14	1110	1001
15	1111	1000

The gray code is often used in mechanical applications such as shaft encoders.

### Modulo 2 Arithmetic

This is binary addition but the carry is ignored.

### Converting Gray to Binary

1. write down the number in gray code
2. the most significant bit of the binary number is the most significant bit of the gray code
3. add (using modulo 2) the next significant bit of the binary number to the next significant bit of the gray coded number to obtain the next binary bit
4. repeat step 3 till all bits of the gray coded number have been added modulo 2
5. the resultant number is the binary equivalent of the gray number

**Example, convert 1101101 in gray code to binary**

Gray	1101101		
1.	1101101	1	
2.	1101101	10	1 modulo 2 = 0
3.	1101101	100	0 modulo 2 = 0
4.	1101101	1000	0 modulo 2 = 0
3/4	1101101	10010	1 modulo 2 = 1
3/4	1101101	100100	0 modulo 2 = 0
3/4	1101101	1001000	0 modulo 2 = 0
3/4	1101101	10010001	0 modulo 2 = 1

Answer is 1001001

### Converting Binary to Gray

1. write down the number in binary code
2. the most significant bit of the gray number is the most significant bit of the binary code
3. add (using modulo 2) the next significant bit of the binary number to the next significant bit of the binary number to obtain the next gray coded bit
4. repeat step 3 till all bits of the binary coded number have been added modulo 2
5. the resultant number is the gray coded equivalent of the binary number

**Example, convert 1001001 in binary code to gray code**

	Binary	Gray	
1.	1001001		
2.	1001001	1	copy down the msb
3.	1001001	11	1 modulo 2 = 1
4.	1001001	110	0 modulo 2 = 0
3/4	1001001	1101	0 modulo 2 = 1
3/4	1001001	11011	1 modulo 2 = 1
3/4	1001001	110110	0 modulo 2 = 0
3/4	1001001	1101101	0 modulo 2 = 1

Answer is 1101101

## Excess 3 Gray Code

In many applications, it is desirable to have a code that is BCD as well as unit distance. A **unit distance** code derives its name from the fact that there is only one bit change between two consecutive numbers. The excess 3 gray code is such a code, the values for zero and nine differ only 1 bit, and so do all values for successive numbers.

Outputs from linear devices or angular encoders may be coded in excess 3 gray code to obtain multi-digit BCD numbers.

Decimal	Excess 3 Gray
0	0010
1	0110
2	0111
3	0101
4	0100
5	1100
6	1101
7	1111
8	1110
9	1010

