Number Systems

There are three important number systems which you should become familiar with.

These are **decimal**, **binary** and **hexadecimal**.

- The decimal system, which is the one you are most familiar with, utilizes ten symbols to represent each digit. These are 0, 1, 2, 3, 4, 5, 6, 7, 8 and 9.
- Similarly, the binary system is base 2.
- Hexadecimal is base 16 as shown below:

Number System	Radix	Symbols
Binary	2	0 1
Decimal	10	0 1 2 3 4 5 6 7 8 9
Hexadecimal	16	0 1 2 3 4 5 6 7 8 9 A B C D E F

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

Number Conversion

Now, let us assume that we are working predominantly with the hexadecimal system.

In this case, we split the binary pattern into groups of four bits as follows:

01011101_2 = 0101 1101 (Binary)

= 5 D (Hexadecimal)

The conversion to decimal :

$5D_{16} = (5 \times 16) + 13 = 93$

The result is: $01011101_2 = 5D_{16} = 93_{10}$

The symbols **$** and **h** are commonly used to signify hexadecimal notation, for example, $5D or 5Dh.

Data Representation

- The native language of digital computers is inherently **binary**.
- The range of numbers which can be represented is dependent on the number of bits used. A common unit of storage is a **byte** which is a group of **eight bits**.
- Eight bits can represent 2^8 unique states, i.e. 256 possible combinations.
- This fact can be used to our advantage to represent many different things on the computer. For example,

a byte can be used to represent 256 colors, 256 shades of grey, 256 shapes, 256 symbols, 256 names, or even 256 different numbers. More typically, we can use a byte to represent 256 sequential numbers such as the numbers from 1 to 256, or the numbers from 0 to 255.

Signed Numbers

As another example, how about the numbers from -128 to 127? Note that there are 256 unique numbers in this range.

One scheme to represent signed integers is to reserve one of the eight bits to indicate the sign of the number.

Following usual convention, the left-most bit (called the **most significant bit** or **MSB**) is dedicated as the sign bit. With this scheme, we can now have 128 positive and 128 negative numbers, typically the numbers from 0 to 127.

This scheme has the anomaly that there are two unique representations for positive and negative zero. A more serious problem is that the rules of binary arithmetic breakdown when we decrement by 1 from positive to negative numbers.

Two's complement representation:

The **two's complement binary** system overcomes both problems mentioned above. In this notation, the negative number is represented by forming the binary complement of the positive number and adding one.

1. Complement all bits (change all zeros to ones and vice versa)
2. Add one.

Note that in all the signed binary systems discussed, the sign is represented by the **MSB**.

Example:

+5: 0000 0101

To find -5: Complementing 00000101 yields 11111010

Adding one yields **1**1111011: -5

Sign = 1 means minus

Computer Arithmetic

Addition:

Unsigned Numbers: There is no sign bit

```
   01110101: 117            11111111: 255
 + 01100011:  99          + 00000001:   1
 ---------------          ---------------
   11011000: 216           100000000: 256
                          CARRY
```

Signed Numbers:

```
   11111111:  -1            11111111:  -1
 + 00000001:   1          + 11111111:  -1
 ---------------          ---------------
  100000000:   0           111111110:  -2
  ↑                         ↑
Sign(+)                   Sign(-)
```

By addition of signed numbers disregard any carry

Computer Arithmetic

Subtraction:

In practice, computers don't subtract. Instead, they negate the second operand and add.

```
  5: 00000101   ==>   00000101
- 1:-00000001       + 11111111
    ----------      ----------
                    100000100 : 4   CARRY=B̅O̅R̅R̅O̅W̅

  1: 00000001   ==>   00000001
- 5:-00000101       + 11111011
    ----------      ----------
                      11111100 :-4  NO CARRY means BORROW
```

Carry:

In the case of unsigned addition carry occurs when the correct result is too large to be represented.

Borrow:

In the case of unsigned subtraction borrow occurs when the value being subtracted is larger than the value it is subtracted from. If subtraction is performed by negating and adding , there is a borrow if the addition does not produce a carry.

Overflow:

In the case of signed addition and subtraction, overflow occurs when the correct result is either more positive than the largest possible positive value ore more negative than the smallest possible negative one. It is indicated by a result having incorrect sign.

There is overflow if:

poz + poz → neg	poz – neg → neg
neg + neg → poz	neg – poz → poz

©1999-2005 Dr. Feza BUZLUCA