Data Representation
(Numbering Systems)

Decimal System

Base 10 numbers are composed of 10 possible digits (0 – 9). Each digit of a number has a power of 10 associated with it based on its position within the number.

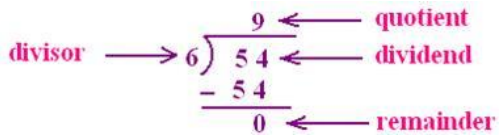
$157_{10} = 1*10^2 + 5*10^1 + 7*10^0$
$= 100 + 50 + 7$
$= 157_{10}$

Binary System

Base 2 numbers are composed of 2 possible digits (0 and 1). Each digit of a number has a power of 2 associated with it based on its position within the number.

$10011101_2 = 1*2^7 + 0*2^6 + 0*2^5 + 1*2^4 + 1*2^3 + 1*2^2 + 0*2^1 + 1*2^0$
$= 128 + 0 + 0 + 16 + 8 + 4 + 0 + 1$
$= 157_{10}$

Conversion from base 10 to base 2 requires that the base 10 number (the dividend) be divided by 2 (the divisor). The quotient always becomes the new/next dividend and the remainder always becomes a base 2 integer of 1 or 0. The first remainder is always the least significant bit (LSB) which resides at the 2^0 position and the very last remainder is always the most significant bit (MSB) which resides at the $2^{\text{Position Weight}}$ position. A conversion example is shown below.



$\frac{157}{2} = 78\_r\_1(\text{LSB})$ $\frac{78}{2} = 39\_r\_0$ $\frac{39}{2} = 19\_r\_1$ $\frac{19}{2} = 9\_r\_1$ $\frac{9}{2} = 4\_r\_1$ $\frac{4}{2} = 2\_r\_0$ $\frac{2}{2} = 1\_r\_0$ $\frac{1}{2} = 0\_r\_1(\text{MSB})$

Octal System

Base 8 numbers are composed of 8 possible digits (0 – 7). Each digit of a number has a power of 8 associated with it based on its position within the number.

$010\ 011\ 101_2 = 235_8$
$235_8 = 2*8^2 + 3*8^1 + 5*8^0$
$= 128 + 24 + 5$
$= 157_{10}$

$\frac{157}{8} = 19\_r\_5(\text{LSD})$ $\frac{19}{8} = 2\_r\_3$ $\frac{2}{8} = 0\_r\_2(\text{MSD})$

Hexadecimal System

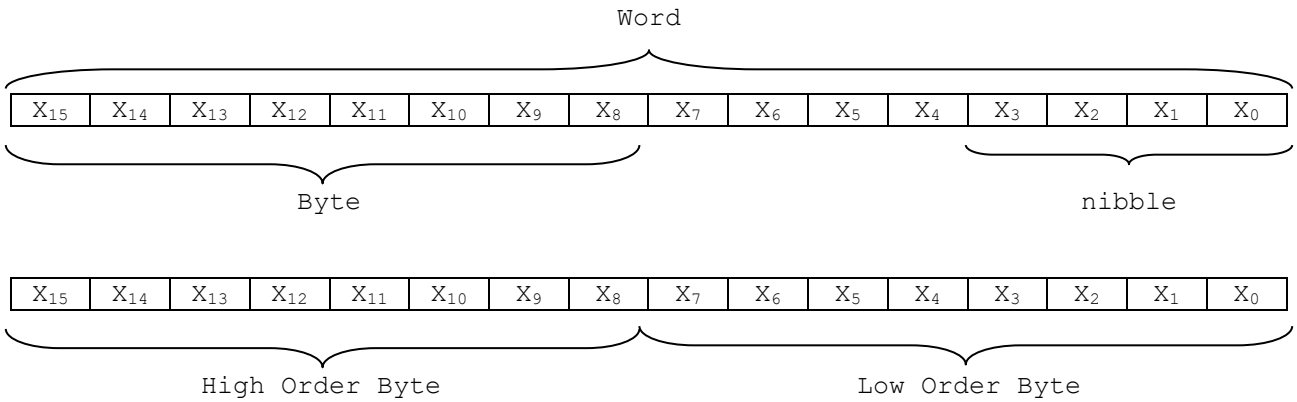
Base 16 numbers are composed of 16 possible digits (0 – F). Each digit of a number has a power of 16 associated with it based on its position within the number.

$1001\ 1101_2 = 9D_{16}$
$9D_{16} = 9*16^1 + 13*16^0$
$= 144 + 13$
$= 157_{10}$

$\frac{157}{16} = 9\_r\_13(\text{LSD})$ $\frac{9}{16} = 0\_r\_9(\text{MSD})$

Binary Formats
(Data Organization)

16-bit (binary digit) Representation



Single nibbles are mainly used for Binary Coded Decimal (BCD) and Hexadecimal (HEX) numbering systems where HEX is a shorthand version of Binary.

Binary Nibble (base 2)	BCD (base 10)	HEX (base 16)
0000	0	0
0001	1	1
0010	2	2
0011	3	3
0100	4	4
0101	5	5
0110	6	6
0111	7	7
1000	8	8
1001	9	9
1010	NOT USED	A
1011	NOT USED	B
1100	NOT USED	C
1101	NOT USED	D
1110	NOT USED	E
1111	NOT USED	F

When programming in any Language, all variables are numeric values. Probably the most important use for a Byte is the character code (ASCII).

Unsigned and Signed Numbers

Unsigned integers, which are non-negative, are represented in a straightforward binary manner. Number 200_{10} is represented by a single byte, unsigned integer as 11001000_2 (or $C8_{16}$).

Signed integers (which may be positive or negative) are represented in a more complicated way. There are three possible representations.

Signed Magnitude: Number $+56_{10}$ is represented by $0011\ 1000_2$. The most significant bit (MSB) bit of the byte is used as the sign bit. This bit is 0 if the number is positive and 1 if negative. The remainder of the number is the magnitude. The largest Byte value is $+127_{10}$, smallest is -127_{10}. Two drawbacks: +0 ($0000\ 0000_2$) and −0 ($1000\ 0000_2$) as well as complicated subtraction routines within the CPU.

(What does -56_{10} look like in binary using this convention? $1011\ 1000_2$)

One's Complement: This method is found by taking the inverse/complement of each bit (bitwise NOT). Again, number $+56_{10}$ is represented by $\underline{0}011\ 1000_2$. The one's complement of $+56_{10}$ is $\underline{1}100\ 0111_2$ which is the equivalent negating each bit. The same two problems arise as with signed magnitude: +0 ($\underline{0}000\ 0000_2$) and −0 ($\underline{1}111\ 1111_2$) as well as complicated CPU arithmetic. The largest Byte value is $+127_{10}$, smallest is -127_{10}.

Two's Complement: Modern computers use this method for arithmetic operations. This number is found by finding the one's complement of a number and adding a binary one to it.

$+56_{10} = \underline{0}011\ 1000_2$

The one's complement is $\underline{1}100\ 0111_2$

The two's complement is $\underline{1}100\ 0111_2$

$$\begin{array}{r} \underline{1}100\ 0111_2 \\ +0000\ 0001_2 \\ \hline \underline{1}100\ 1000_2 \end{array}$$

So -56_{10} is represented by $\underline{1}100\ 1000_2$ in two's complement form. Does taking the twos complement again produce $+56_{10}$?

When performing addition in a two's complement operation, the MSB might produce a carry. This carry is not used in the result, though we will look at it within the Flag Register when manipulating the instruction set. The range of the data byte representation is $+127_{10}$ to -128_{10}.

This topic will be covered in greater detail when discussing bitwise operators.