Binary Code Decimal (BCD): ad ogni cifra sono associati 4 bit, secondo la seguente tabella:

Cifra Decimale	Cifra BCD
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

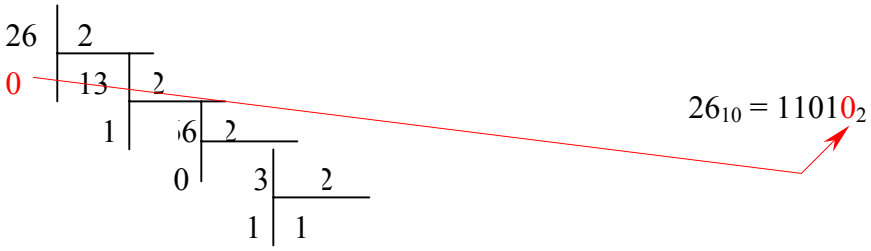
Quindi ogni numero viene scomposto in cifre decimali e ogni cifra viene tradotta, come ad esempio: 27 → 2 = 0010 e 7 = 0111 → 27 = 00100111

Sistema Binario: a quanto corrisponde il numero binario 1101_2 in decimale?

1	1	0	1
2^3	2^2	2^1	2^0
1×2^3	1×2^2	0×2^1	1×2^0
8	4	0	1
$1101_2 = 8 + 4 + 0 + 1 = 13$			

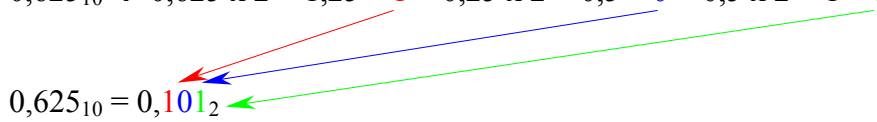
Per passare da base decimale a base binaria bisogna distinguere tra (a) Numeri interi e (b) Numeri frazionari.

(a) Numeri Interi, bisogna dividere il numero per 2 e il resto della divisione ci fornisce il codice binario a partire dalla cifra meno significativa, come ad esempio:



(b) Numeri Frazionari, si moltiplica la parte frazionaria per 2, se il numero ottenuto è maggiore o uguale a 1, si sottrae 1 e si considera come prima cifra dopo la virgola il numero 1, se invece il numero è minore di 1, si segna uno 0 e si torna a moltiplicare per 2 fino a quando non si ottiene 0, come ad esempio:

$0{,}625_{10}$ → 0,625 x 2 = 1,25 – 1 = 0,25 x 2 = 0,5 – 0 = 0,5 x 2 = 1 – 1 = 0



Per passare dalla rappresentazione binaria a quella ottale o esadecimale bisogna usare la seguente tabella:

Decimale	Binario	Ottale (sono 8)	Esadecimale (sono 16)
0_{10}	0000_2	0_8	0_{16}
1_{10}	0001_2	1_8	1_{16}
2_{10}	0010_2	2_8	2_{16}
3_{10}	0011_2	3_8	3_{16}
4_{10}	0100_2	4_8	4_{16}
5_{10}	0101_2	5_8	5_{16}
6_{10}	0110_2	6_8	6_{16}
7_{10}	0111_2	7_8	7_{16}
8_{10}	1000_2	10_8	8_{16}
9_{10}	1001_2	11_8	9_{16}
10_{10}	1010_2	12_8	A_{16}
11_{10}	1011_2	13_8	B_{16}
12_{10}	1100_2	14_8	C_{16}
13_{10}	1101_2	15_8	D_{16}
14_{10}	1110_2	16_8	E_{16}
15_{10}	1111_2	17_8	F_{16}

Quindi per esempio il numero decimale $\underline{133_{10}}$ corrisponderà a:
133 : 2 = 66 : 2 = 33 : 2 = 16 : 2 = 8 : 2 = 4 : 2 = 2 : 2 =1
$\underline{10000101_2}$ → $2^0 + 2^2 + 2^7 = 133_{10}$ → 10-000-$101_2 = \underline{205_8}$ → 1000-$0101_2 = \underline{85_{16}}$

Completamento a 1: per i numeri positivi si calcola il numero in codice binario, per quelli negativi si calcola il codice binario del valore assoluto del numero decimale assegnato, poi si aggiunge uno 0 a sinistra e si invertono gli 0 con gli 1 e viceversa, come ad esempio:
$5_{10} = 0101_2$ -5_{10} → $5_{10} = 0101_2$ → $-5_{10} = 1010_2$.

Completamento a 2: per i numeri positivi si calcola il numero in codice binario, per quelli negativi si calcola il codice binario del valore assoluto del numero decimale assegnato, poi si aggiungono tot 0 a sinistra per raggiungere n bit e si invertono gli 0 con gli 1 e viceversa ed infine si somma 1, come ad esempio: con n = 4bit
$5_{10} = 0101_2$ -5_{10} → $5_{10} = 0101_2$ → $1010_2 + 1$ → $-5_{10} = 1011_2$.

Rappresentazione in eccesso: si somma al numero decimale che vogliamo rappresentare il valore dell'eccesso e lo si trasforma in binario in complemento a 2, come ad esempio:
n = 4bit → eccesso 8
-3_{10} → $-3 + 8 = 5_{10} = 0101_2$
$+4_{10}$ → $4 + 8 = 12_{10} = 1100_2$

Regole per l'addizione naturale:

A	0	1	0	1
B	0	0	1	1
Somma	0	1	1	0
Riporto	0	0	0	1

Esempio: 12 + 15 + 156 = 183

00001100	12
00001111	15
10011100	156
10110111	183

Se provassi ora a sommare i valori 156 e 183 avrei un errore di overflow cioè la rappresentazione binaria del valore 339 richiede l'utilizzo di 9 bit, (**1**01010011).

Regole per la sottrazione naturale:

A	0	1	0	1
B	0	0	1	1
Sottrazione	0	1	1	0
Prestito	0	1	1	0

Esempio: 183 - 15 - 12 = 156

10110111	183
00001111	15
00001100	12
10011100	156

Se provassi ora a sottrarre a 156 il valore 183 avrei un errore di underflow cioè la rappresentazione binaria del valore -27 richiede l'utilizzo di 9 bit, (**1**11100101).

Standard IEEE 754: Questo formato di conversione cambia a seconda della precisione, semplice a 32 bit e doppia a 64 bit. Nella precisione semplice si ha:

1 bit	8 bit	23 bit
+ o -	esponente	mantissa

Nella precisione doppia si ha:

1 bit	11 bit	57 bit
+ o -	esponente	mantissa

Per la conversione dello standard IEEE 754 a precisione semplice, l'esponente si rappresenta in eccesso 127 e la mantissa viene rappresentata normalizzata, mentre per il segno lo 0 indica + e l'1 indica il -, come per esempio:
-29,1875 $29_{10} = 11101_2$ $0{,}1875_{10} = 0.0011_2$ $29{,}1875 = 11101.0011_2 = 0.1\,11010011 * 2^5$
5 +126 = 131 → 10000011 segno = 1 → -29,1875 = 11000001-11101001-10000000-00000000
gli 0 vengono aggiunti fino al riempire il numero di bit richiesti.
N.B.: essendo per forza, nella mantissa, il primo numero dopo la virgola un 1, esso viene omesso nella rappresentazione del numero nello standard IEEE 754