Sistemi di numerazione

Numeri naturali

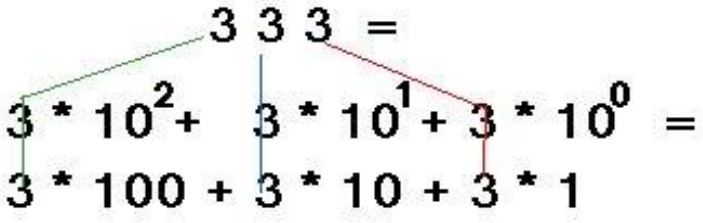
Noi rappresentiamo i numeri con una sequenza di cifre, ad esempio 10 : 0,1,2,3,4,5,6,7,8,9 , secondo una *notazione posizionale* .

Posizionale significa che il valore di ogni cifra dipende dalla sua posizione all'interno della sequenza.

E' possibile scegliere il numero di cifre differenti che si usano in una notazione posizionale e tale numero prende il nome di *base* ; in ogni numero la cifra all'estrema destra, detta LSB = LEAST SIGNIFICANT BIT *(cifra meno significativa)* ha il valore minore, quella all'estrema sinistra, MSB = MOST SIGNIFICANT BIT *(cifra più significativa)*, il valore maggiore.

Per una numerazione in base B, con B numero naturale > 1 , occorrono B cifre distinte.

Esempio in Base 10 :



FORMULA POLINIOMALE

In genere si usano anche le numerazioni :

- ***binaria* in base 2**
- ***ottale* in base 8**
- ***esadecimale* in base 16**

Nel *sistema binario* si usano le cifre 0 e 1.
La base 2 è quella più piccola teoricamente possibile per un sistema di numerazione.
Il valore posizionale è legato alle potenze di 2.

Nel *sistema ottale* si usano le cifre : 0,1,2,3,4,5,6,7.
Il valore posizionale è legato alle potenze di 8.

Nel *sistema esadecimale* si usano le cifre : 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F.
Il valore posizionale è legato alle potenze di 16.

Esempi di conversione : da una qualunque base B a base decimale

Esempio 1:
Codifica decimale : B=10

$$(5870)_{10} = 5 * 10^3 + 8 * 10^2 + 7 * 10^1 + 0 * 10^0$$

Esempio 2:
Codifica binaria : B=2

$$(101001011)_2 = 1 * 2^8 + 0 * 2^7 + 1 * 2^6 + 0 * 2^5 + 0 * 2^4 + 1 * 2^3 + 0 * 2^2 + 1 * 2^1 + 1 * 2^0 = (331)_{10}$$

Esempio 3:
Codifica ottale : B=8

$$(534)_8 = 5 * 8^2 + 3 * 8^1 + 4 = (348)_{10}$$

***Esempio 4*:**
Codifica esadecimale : B=16

$$(B7F)_{16} = 11 * 16^2 + 7 * 16^1 + 15 = (2943)_{10}$$

Regole di conversione tra sistemi di numerazione

La notazione posizionale è molto comoda perché fa capire il significato di un numero, qualunque sia la base con cui è stato scritto. Ma come si risolve il problema inverso, cioè dato un numero nella consueta base decimale, come si fa a scriverlo in una base assegnata, diversa da 10 ?

Si applica il metodo delle *divisioni successive* , che consiste nel prendere il numero da convertire e dividerlo per la base considerata ; prenderne il quoziente e dividerlo ancora per la base, e così via, fino a che non si ottiene come risultato un quoziente zero.

In questa sequenza di divisioni occorre segnare i resti che man mano si ottengono, perchè sono questi che, presi in senso inverso dall'ultimo al primo, costituiscono proprio la conversione cercata.

E' importante far notare che per i *numeri interi* si usa il metodo appena descritto delle *divisioni successive.*

Per i *numeri frazionari* si usa, per la parte a sinistra della virgola (parte intera), il metodo delle divisioni successive e per la parte a destra della virgola (parte frazionaria), il metodo delle *moltiplicazioni successive*.

Con tale metodo si prende come cifra binaria la parte intera e si moltiplica per 2 la parte decimale fino a quando la parte frazionaria è diventata nulla o quando si sia trovato un numero sufficiente di cifre binarie. Le varie parti intere vanno prese in ordine diretto.

Vediamo adesso qualche esempio di conversione dalla base 10 alla base 2, 8 e 16.

Esempio 1 : da base 10 a base 2

Numero intero

Conversione del numero 23 in base 2

	Quoziente	Resto
23 : 2 =	11	1
11 : 2 =	5	1
5 : 2 =	2	1
2 : 2 =	1	0
1 : 2 =	0	1

Arrivati a un quoziente zero il procedimento finisce; il risultato è dato dal numero 10111 in base 2.

Per controllare che l'esecuzione sia esatta, si applica la regola della notazione posizionale :

$$(10111)_2 = 1*2^4 + 0*2^3 + 1*2^2 + 1*2^1 + 1*2^0 = (23)_{10}$$

Questo procedimento va bene qualunque sia la base considerata.

Numero frazionario

Conversione del numero 0.65625 in base 2

	Parte intera
0.65625*2 =	1.31250
0.31250*2 =	0.62500
0.62500*2 =	1.250
0.250*2 =	0.500
0.5*2 =	1.00

Arrivati a tale risultato il procedimento finisce. Per cui la conversione è:

$$(0.65625)_{10} = (10101)_2$$

Allo stesso modo è possibile passare da una rappresentazione decimale ad una ottale o esadecimale, eseguendo le divisioni successive come si vede negli esempi successivi.

Esempio 2 : da base 10 a base 8

Conversione del numero $(678)_{10}$ in base 8

	Q	R
678 : 8 =	84	6
84 : 8 =	10	4
10 : 8 =	1	2
1 : 8 =	0	1

Quindi $(678)_{10}$ corrisponde a $(1246)_8$ La verifica che l'esecuzione sia stata corretta è lasciata come esercizio (basta applicare la formula polinomiale)

Esempio 3 : da base 10 a base 16

Conversione del numero 44 in base 16

44 :16 = 2 con resto 12 dove 12_{10} corrisponde a C_{16}
2 :16 = 0 con resto 2

Quindi 44_{10} corrisponde a $2C_{16}$.
Anche in questo caso la verifica è lasciata come esercizio.

RAPPRESENTAZIONE DEI NUMERI NEGATIVI IN BASE 2

Vi sono 2 possibili rappresentazioni :

1. **IN MODULO E SEGNO**
2. **IN COMPLEMENTO A DUE**

1. Nella prima , il modulo del numero viene scritto in BINARIO PURO e a questi bit si aggiunge , nella posizione MSB (la prima cifra a sx), il BIT DI SEGNO , che vale :

- **0 per i n° positivi**
- **1 per i n° negativi**

In questa rappresentazione, come si vede , vi sono 2 codifiche per lo zero , inoltre non è pratica per le operazioni di somma e sottrazione.

TAB N° RELATIVI

Bit di segno	Bit del modulo			Base10
0	0	0	0	+0
0	0	0	1	+1
0	0	1	0	+2
0	0	1	1	+3
0	1	0	0	+4
0	1	0	1	+5
0	1	1	0	+6
0	1	1	1	+7
1	0	0	0	-0
1	0	0	1	-1
1	0	1	0	-2
1	0	1	1	-3
1	1	0	0	-4
1	1	0	1	-5
1	1	1	0	-6
1	1	1	1	-7

2. Nella seconda rappresentazione, invece, c'è il Bit di segno, a cui seguono i bit che rappresentano il modulo del numero, secondo la regola del Complemento a 2 (CPL2) :

- **Si prende la stringa in Binario puro che rappresenta il modulo e se ne effettua il Complemento a 1, cioè si scambiano gli 0 con gli 1 e viceversa ;**
- **Al CPL1 si aggiunge 1**

Es. 1 : (- 57)$_{10}$ ⟶ (57)$_{10}$ = (1 1 1 0 0 1)$_2$ **CPL1** **+1** **1 1 1 0 0 1 ⟶ 0 0 0 1 1 0 ⟶ 0 0 0 1 1 1**

Per cui : (- 57)$_{10}$ = 1 0 0 0 1 1 1

- 1 **CPL1**

Es. 2 : (1 0 1 0 0 1 1)$_{CPL2}$ ⟶ 0 1 0 0 1 1 ⟶ 0 1 0 0 1 0 ⟶(1 0 1 1 0 1)$_2$

Per cui : (1 0 1 0 0 1 1)$_{CPL2}$⟶ (- 45)$_{10}$

- **N.B. Per ottenere il Binario puro dal CPL2 , si può anche effettuare il CPL1 e aggiungere 1, cioè il CPL2 del CPL2 è il Binario puro !!**

- **METODO RAPIDO : dato un numero in Binario puro, per ottenere il CPL2 si può operare così : si ricopia la stringa di bit dal LSB verso sx fino al primo 1 compreso e si fa il CPL1 dei bit rimanenti, sempre verso sx.**

Es : (101110100)$_{CPL2}$

Considero i bit relativi al modulo : 01110100 , ricopio i primi 3 bit da dx e faccio il CPL1 sui rimanenti, ottenendo : 10001100 . Questa stringa in Binario puro significa (140)$_{10}$, perciò (101110100)$_{CPL2}$ = (- 140)$_{10}$

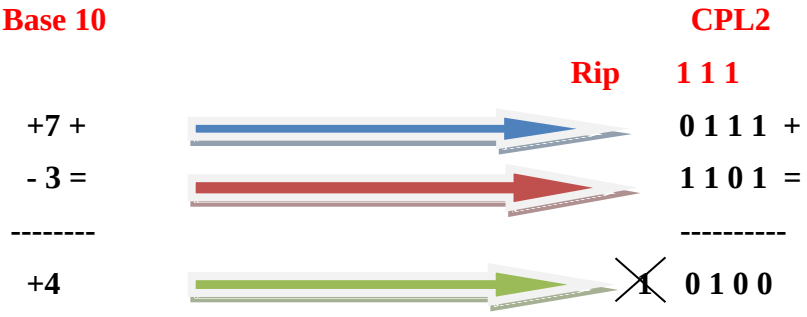
TABELLA a 4 BIT in CPL2 per i numeri relativi

0	0	0	0	0
0	0	0	1	+1
0	0	1	0	+2
0	0	1	1	+3
0	1	0	0	+4
0	1	0	1	+5
0	1	1	0	+6
0	1	1	1	+7
1	0	0	0	-8
1	0	0	1	-7
1	0	1	0	-6
1	0	1	1	-5
1	1	0	0	-4
1	1	0	1	-3
1	1	1	0	-2
1	1	1	1	-1

In generale , con N Bit in CPL2 posso rappresentare i n° da $-2^{(N-1)}$ a $+(2^{(N-1)}-1)$

APPLICAZIONE :

Per effettuare la SOTTRAZIONE tra 2 Numeri in Base 2 , si esegue la SOMMA tra il MINUENDO e il CPL2 del SOTTRAENDO, con eliminazione dell'eventuale ultimo riporto (a sx)



Operazioni nel sistema di numerazione binaria

Le operazioni che si eseguono sui numeri sono ovviamente le quattro operazioni fondamentali:

- **somma**
- **prodotto**
- **sottrazione**
- **divisione**

La somma

L'algoritmo della operazione di somma non cambia qualunque sia la base considerata.
Naturalmente, le regole da imparare nel caso di una base b sono relative alle sole b^2 posssibili combinazioni delle cifre da 0 a b-1.
Così le 100 regole della base 10 si riducono a 4 soltanto nella base 2:

0 + 0 = 0
0 + 1 = 1
1 + 0 = 1
1 + 1 = 0 con riporto di 1 (nella colonna a sinistra)

E' nell'ultima regola che si vede la semplificazione portata dalla base 2: a parità di significato (valore), un numero scritto in base 2 è molto più lungo dell'equivalente scritto in base 10, ma le regole per poi eseguire la somma sono di gran lunga più semplici.

Nel caso dell'elaboratore questo è essenziale; infatti, la velocità gli permette di non preoccuparsi eccessivamente della lunghezza dei numeri, mentre le regole relative alla somma delle coppie di cifre sono legate alla circuiteria elettronica che le deve eseguire, e un conto è complicare tale circuiteria per realizzare 100 regole e un altro è doverne realizzare solo 4.

Per la somma di due numeri positivi di lunghezza K possono essere necessari K+1 bit. Se sono disponibili solo K cifre si genera un errore di overflow (o trabocco).

Esempio:

A =$(11011)_2 = (27)_{10}$
B = $(00110)_2 = (6)_{10}$

$(11011+ 00110)_2 = (100001)_2$
$(27 + 6)_{10} = (33)_{10}$
Questa semplice verifica è lasciata come esercizio.

Il prodotto

Analogo all'operazione di somma è il prodotto. Anche in questo caso l'operazione si riduce a conoscere il prodotto di ciascuna coppia di cifre. Nel caso della base 2 tutto si riduce a ricordare l'esiguo numero di 2*2 regole, che sono:

0*0 = 0
0*1 = 0
1*0 = 0
1*1 = 1

In definitiva si hanno le regole seguenti:

- **il prodotto per zero dà sempre come risultato zero;**
- **il prodotto per 1 dà sempre come risultato il numero stesso.**

Esempio:

A = (1011)$_2$ = (11)$_{10}$ **B = (1101)$_2$ = (13)$_{10}$**

(1011*1101)$_2$ = (10001111)$_2$ **(13$_1$ * 11)$_{10}$ = (143)$_{10}$**

```
    1 0 1 1 *
    1 1 0 1 =
   ---------
    1 0 1 1   +
  0 0 0 0     +
 1 0 1 1      +
1 0 1 1       =
----------------------
1 0 0 0 1 1 1 1   >>>>>  ( 143 )10
```

La sottrazione

Le regole sono :

0 - 0 = 0
0 - 1 = 1 con Prestito di 1 (dalla colonna a sinistra)
1 - 0 = 1
1 - 1 = 0

ESERCIZI SUI SISTEMI NUMERICI

1. Conversioni da una base qualunque a base 10 :

$(256)_8 = 2*8^2 + 5*8^1 + 6*8^0 = 128+40+6 = (174)_{10}$

$(37C)_H = 3*16^2 + 7*16^1 + 12*16^0 = 768 +112 + 12 = (892)_{10}$

$(11001101)_2 = 1*2^7 + 1*2^6 + 1*2^3 + 1*2^2 + 1*2^0 = 128 + 64 + 8 + 4 + 1 = (205)_{10}$

$(534)_7 = 5*7^2 + 3*7^1 + 4*7^0 = 245 + 21 + 4 = (270)_{10}$

2. Da base 10 a una base qualunque : metodo delle divisioni successive per la base d' arrivo .

$(183)_{10} = (?)_2$ →

	Q	R	
183 : 2 =	91	1	
91 : 2 =	45	1	
45 : 2 =	22	1	
22 : 2 =	11	0	
11 : 2 =	5	1	
5 : 2 =	2	1	
2 : 2 =	1	0	
1 : 2 =	0	1	↑ $(10110111)_2$

Verifica : $2^7 + 2^5 + 2^4 + 2^2 + 2^1 + 2^0 = (183)_{10}$

$(183)_{10} = (?)_8$ →

	Q	R	
183 : 8 =	22	7	
22 : 8 =	2	6	
2 : 8 =	0	2	↑ $(267)_8$

Verifica : $2 \bullet 8^2 + 6 \bullet 8^1 + 7 \bullet 8^0 = (183)_{10}$

$(183)_{10} = (?)_H$ →

	Q	R	
183 : 16 =	11	7	
11 : 16 =	0	11	↑ $(B 7)_H$

Verifica : $11 \bullet 16^1 + 7 \bullet 16^0 = (183)_{10}$

3. Conversioni tra le basi 2 , 8 , H : si raggruppano i bit a 3 a 3 o a 4 a 4 .

010111101 → 010 111 101

↓ ↓ ↓

2 7 5 il numero è perciò $(275)_8$

010111101 → 0 1011 1101

↓ ↓

11=B 13=D il numero è $(BD)_H$

$(517)_8$ → 5 1 7
↓ ↓ ↓
101 001 111 → $(101001111)_2$

$(A3C)_H$ → A 3 C → $(101000111100)_2$
↓ ↓ ↓
1010 0011 1100

4. Numeri binari con virgola : $(110110 , 101011)_2$

PARTE INTERA : 110110 → $2^5 + 2^4 + 2^2 + 2^1 = (54)_{10}$

PARTE FRAZIONARIA : ,101011 → $2^{-1} + 2^{-3} + 2^{-5} + 2^{-6} =$

1/2 +1/8 +1/32 +1/64 = 0.5 + 0. 125 + 0.03125 + 0.015625 =$(0,671875)_{10}$

il n° completo è perciò $(54 , 671875)_{10}$

5. Conversione in binario di n° in base 10 , con virgola :

PARTE INTERA → divisioni successive per 2 → RESTI (in ordine inverso)

PARTE DECIMALE → moltiplicazioni successive per 2 → PARTI INTERE(in ordine diretto)

$(43 , 90625)_{10}$ → Parte intera = 43 → Parte decimale = 0,90625

Q	R
43 : 2 = 21	1
21 : 2 = 10	1
10 : 2 = 5	0
5 : 2 = 2	1
2 : 2 = 1	0
1 : 2 = 0	1 ↑

	P.Intera
0,90625 x 2 = 1 , 8125	↓
0,8125 x 2 = 1 , 625	
0,625 x 2 = 1 , 25	
0,25 x 2 = 0 , 5	
0,5 x 2 = 1 , 0	

$(43)_{10}$ = $(101011)_2$ $(0, 90625)_{10} = (11101)_2$

in definitiva , $(43, 90625)_{10} = (101011, 11101)_2$

6. Addizione in binario : Carry 1 1 1

1 0 0 1 1 1 0 1 + 157 +
0 1 0 1 1 0 0 1 = 89 =
1 1 1 1 0 1 1 0 $(246)_{10}$

7. Sottrazione in binario :

PRESTITO 1 1 1 1

1 0 1 0 0 1 - $(41)_{10}$
0 1 1 0 1 1 = $(27)_{10}$
---------- ----------
0 0 1 1 1 0 $(14)_{10}$