01 Sistemi numerici e Conversione

Noi contiamo in base 10, ma i computer lavorano in base 2. Visto che avremo a che fare con tanti numeri d'ora in poi, useremo la notazione dei numeri che specifica in che base sono scritti.

💡 Numero *a* in base *n* : a_n

Vediamo alcuni esempi di numeri scritti in base 2, 8, 10 e 16 *(nota che la base sedici deve avere 16 cifre: quindi si usano 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F)*:

❓ $111_2 = 7_8 = 7_{10} = 7_{16}$
$1101_2 = 15_8 = 13_{10} = D_{16}$

La conversione di numerida una base ad un altra avviene tramite divisioni ricorsive o con moltiplicazioni ricorsive.

Numero *(base 10)*	Base di arrivo	Resto
140	2	0
70	2	0
35	2	1
17	2	1
8	2	0
4	2	0
2	2	0
1	2	1
0	2	/

$140_{10} = 10001100_2$
$140 = 2^7 * 2^3 * 2^2$

Numero base(10)	Base di arrivo	Resto
1454	16	14 → E
90	16	10 → A
5	16	5
0	16	/

$1454_{10} = 5AE_{16}$

✅ La **conversione** di un numero **da base ottale a base binaria** può avvenire convertendo singolarmente le cifre che lo compongono **utilizzando 3bit** per poi comporre il risultato

Numero base(8)	Base di arrivo	Resto
16	2	———
1	2	1
0		/

Numero base(8)	Base di arrivo	Resto
6	2	0
3	2	1
1	2	1
0		/

$1_8 = 001_2, 6_8 = 110_2 \rightarrow 001110_2$

Operazioni su base binaria

Su base bianria sono definite le operazioni classiche e in più l'operazione di shift *che corrispondere a dividere o moltiplicare il numero per la sua base.*

- **Somma**

addendo 1	addendo 2	risultato
0	0	0
0	1	1
1	1	0, resto 1

- **Sottrazione**

addendo 1	addendo 2	risultato
0	0	0
0	1	1, con prestito
1	0	1
1	1	0

- **Shift**

0	1	1	1

  - Shift destro *(divisione)*

0	0	1	1

    La cella più a sx mancava e viene quindi sostituita con uno 0

  - Shift sinistro *(moltipliazione)*

1	1	1	0

    La cella più a dx mancava e viene quindi sostituita con uno 0

Rappresentazione di numeri negativi

Per la rappresentazione dei numeri negativi esistono 4 metodi differenti:

▼ **Modulo e segno**

Nella rappresentazione modulo e segno *ms* dati *n* bit quello più significativo verrà utilizzato per indicare il segno del numero, **1** per u numeri **negativi** e **0** per i numeri **positivi**.

Tramite MS si possono rappresentare i numeri tra $-2^{n-1} - 1$ e $+2^{n-1} - 1$

La rappresentazione dei numeri mediante MS presneta 2 prblimi principali:

- Doppia rappresentazione dello 0
- Spreco di 1 bit per il segno

> 💬 La somma causa overflow se utilizzo numeri con lo stesso segno.
> La sottrazione causa overflow se utilizzo numeri di segno opposto.
> Lo Shift avviene ignorando il bit di segno, qualora esso non sovrascriva altri dati, in quel caso avremo overflow.

▼ **Complemento ad 1**

Nella rappresentazione in complemento ad 1 *CA1*, dati gli *n* bit che compongono il numero positivo ne effettuo il complemento *(inversione dei bit)* per rappresentarlo come negativo, i numeri positivi in CA1 sono uguali ad i numeri positivi nella rappresentazione classica.

3= -3 =

0	0	1	1	1	1	0	0

Anche in questo caso persiste il problema della doppia rappresentazione dello 0.

▼ **Complemento a 2**

La rappresentazione in Complemeto a 2, *CA2* è uguale alla rappresentazione in CA1 se non fosse che una volta effettuata la conversione sommo 1 al risultato ottenuto.

3 = -3 =

0	0	1	1	1	1	0	1	

Anche in questo caso abbiamo una doppia rappresnetazione dello 0.

> ✅ Per scriver un numero in CA2 posso anche inizare a trascriverlo da dx verso sx fino a quando incontro il primo 1, dopo averlo trascritto inverto tutti i calori restanti.

> 💬 La somma avviene tenendo in considerazione anche il bit di sengo ed ignorando eventuale riporto.
> La sottrazione avviene facendo il CA2 del numero da sottrarre e svolgendo poi la somma.
> Lo Shift avviene ignorando il bit di segno, qualora esso non sovrascriva altri dati, in quel caso avremo overflow.

▼ **Eccesso 2^{n-1}**

I numeri in eccesso 2^{n-1} si ottengono complementando il MSB in CA2

Rappresentazione dei numeri con la virgola

Numeri in virgola fissa

Dato un numero non intero è possibile ottenerne una sua rappresetnazione in binario scomponendolo in 3 parti: **segno, parte intera, parte frazionaria.**

+6.625 = 110.101

Parte intera:

Numero	base	Resto
6	2	0
3	2	1
1	2	1
0		

Parte frazionaria:

Numero	base	Risultato	parte intera
0.625	2	1.25	1
0.25	2	0.5	0
0.50	2	1.0	1
0			

La trasformazione da numero frazionale decimale a numero binario frazionale avviene per moltiplicazioni ricorsive fino a 0, in oltre il numero viene letto dall'alto verso il basso.

Numeri in virgola mobile

Dato un numero non intero è possibile ottenerne una sua rappresetnazione in binario scomponendolo in 3 parti: **segno, esponente, mantissa.**

+662.52 = 6.6252*10^2 = 110.1010

Parte intera:

Numero	base	Resto
6	2	0
3	2	1
1	2	1
0		

Parte frazionaria:

Numero	base	Risultato	parte intera
0.6252	2	1.2504	1
0.2504	2	0.5008	0
0.5008	2	1.0016	1
0.0016	2	0.0032	0

La trasformazione da numero frazionale decimale a numero binario frazionale avviene per moltiplicazioni ricorsive fino a 0, in oltre il numero viene letto dall'alto verso il basso.