Sistemi di numerazione in base diversa da dieci: il sistema Binario ed altri (cenni).

A cura di Franco Fusi ed Ugo Santosuosso.

Il sistema di numerazione binario è un modo per rappresentare i numeri usando solo 2 cifre diverse lo zero e l'uno.

Questo sistema risulta particolarmente scomodo per l'essere umano, ma risulta di facile realizzazione a livello di circuiteria elettronica: infatti viene usato all'interno dei calcolatori digitali (dall'inglese “digit” = numero) per rappresentare i numeri con il minor impegno possibile di tecnologia e di componentistica (e quindi di costo).

In elettronica, una «cifra binaria» (binary digit = Bit) viene rappresentata con un singolo componente - normalmente un transistor - che lascia passare corrente nel caso rappresenti la cifra "1" e non la lascia passare in caso rappresenti la cifra "0".

In questo sistema di numerazione i numeri vengono rappresentati come segue:

0	0
1	1
10	2
11	3
100	4
101	5
110	6
111	7
.....	

Facciamo una digressione per chiarire meglio un concetto. Per adesso allontaniamoci dal sistema binario e torniamo al sistema cui siamo abituati fin'ora per rappresentare i numeri.

Come tutti i sistemi di numerazione noti, a parte quello usato nell'Impero Romano, Egizio, e nella civiltà Meso-americane prima della scoperta di Colombo, in cui il valore delle cifre era intrinseco - ovvero si

usavano simboli diversi per valori diversi es: X=10, L=50, D=500, M=1000 - il sistema di numerazione decimale e quello binario, sono sistemi di numerazione "posizionale" ovvero in cui la posizione all'interno del numero di una data cifra le assegna un "peso" o un "valore" diverso.

Ad esempio il numero decimale seguente che esprime il valore di «centoventitremila quattrocento cinquantuno», contiene nella sua notazione decimale due cifre "1":

$$\underline{1}23451\underline{1}$$

1 2

ma il valore della prima cifra uno **(1)** [*centomila*] è ben diverso dall'altra cifra uno **(2)** [*unità*].

Questo numero in rappresentazione binaria corrisponderebbe a «11110001000111011»; il valore rappresentato è sempre il medesimo.

Cosa vuol dire questo ?

Se consideriamo la cifra espressa in precedenza ovvero "123451" quale valore essa rappresenta ?

Valore = 1 * "centomila" + 2 * "diecimila" + 3 * "mille" + 4 * "cento" + 5 * "dieci" + 1 * "unità"

ovvero:

centomila + ventimila + tremila + quattrocento + cinquanta + uno

ed in linguaggio corrente:

centoventitremila quattrocentocinquantuno

Si può notare che possiamo cambiare la rappresentazione del linguaggio corrente come segue:

centomila	$= 10^5$	(10 alla quinta potenza)
diecimila	$= 10^4$	(10 alla quarta potenza)
mille	$= 10^3$	(10 alla terza potenza)
cento	$= 10^2$	(10 alla seconda potenza)
dieci	$= 10^1$	(10 alla prima potenza)
uno	$= 10^0$	(10 alla potenza zero)

Riscrivendo l'espressione del valore della cifra avremo quindi che:

$$\text{Valore} = 1 * 10^5 + 2 * 10^4 + 3 * 10^3 + 4 * 10^2 + 5 * 10^1 + 1 * 10^0 \ (*)$$

Come appare evidente questa è una somma "pesata" di potenze di base 10. Evidenziandolo meglio col colore si vede che:

$$\text{Valore} = 1 * 10^5 + 2 * 10^4 + 3 * 10^3 + 4 * 10^2 + 5 * 10^1 + 1 * 10^0$$

- **in rosso abbiamo la "base" del sistema numerale cioè 10 (siamo in presenza di una cifra epressa in un sistema in "base 10" o "decimale")**
- **in verde abbiamo "i pesi" di ogni potenza di 10 ovvero quante volte quel valore deve essere contato per avere il valore corretto della cifra**

In altro modo possiamo rappresentare il valore così:

posizione	6	5	4	3	2	1
pesi	**1**	**2**	**3**	**4**	**5**	**1**
potenze	10^5	10^4	10^3	10^2	10^1	10^0

La sequenza di cifre in verde è esattamente la rappresentazione decimale del valore espresso. La parte rossa è nell'uso quotidiano trascurato in quanto la posizione (posizione - 1 = esponente del potenza) indica il valore della potenza di 10 che quella cifra rappresenta.

In caso di una rappresentazione con una base generica avremo che la formula (*) diventa:

$$N = a_n * b^n + \ldots\ldots + a_2 * b^2 + a_1 * b^1 + a_0 * b^0 \ (**)$$

oppure, invertendo semplicemente l'ordine di scrittura:

$$N = a_0 * b^0 + a_1 * b^1 + a_2 * b^2 + \ldots\ldots + a_n * b^n$$

Per saperne di più sui vari sistemi puoi consultare anche questa pagina:
sistemi di numerazione.
[http://159.149.27.201/binfovet/bioin/biatica/numer.htm]

Torniamo ora al sistema binario, e vediamo di fare alcune considerazioni su di esso, prima di vedere la sua utilità ed i mezzi con cui utilizzarlo.

(.....)

Nota: Questo tipo di numerazione puo' essere usato come tipo di codifica; per esempio, una codifica molto semplice può essere la seguente:

0	corrisponde a **«A»**	
1	corrisponde a **«B»**	
10	corrisponde a **«C»**	
11	corrisponde a **«D»**	
100	corrisponde a **«E»**	
101	corrisponde a **«F»**	
	E così via	

Rappresentazione numerica all'interno di apparati digitali: aritmetica con un numero finito di cifre.

Abbiamo detto in precedenza che i numeri binari sono utilizzati all'interno delle macchine digitali per rappresentare i numeri per motivi di semplicita' realizzativa. All'interno di un calcolatore o di un "p.c." (**p**ersonal **c**omputer = calcolatore personale) la rappresentazione binaria di un numero non può avvenire allo stesso modo in cui noi la scriviamo: per questioni di limitazioni progettuali è possibile rappresentare solo un numero finito di cifre che compongono un numero. Per questo motivo si dice che all'interno di un p.c. - o di un calcolatore - si dice che si usa una aritmetica con un numero di cifre "**finito**", ovvero "**non infinito**".

Esempio:

il valore irrazionale

"radice di 2" = 1,414213562.....

oppure

"Pi-greco" = 3,141592653....

non puo' essere rappresentato che con un numero **limitato a priori** dei suoi infiniti decimali, a meno di non occupare per una sua rappresentazione - *che sarebbe comunque finita* - tutta la capacità circuitale del computer stesso. Questi numeri verranno rappresentati con un numero finito di decimali, di solito non superiori a 10.

In caso fosse richiesta una precisione superiore a questa è il software in funzione sul calcolatore a provvedere ad aumentare, con particolari algoritmi di cui non discuteremo in questa sede, il numero delle cifre decimali usate per fare i calcoli. Analogamente avviene per i numeri di tipo "intero" ovvero senza decimali.

Per questioni costruttive, all'interno di qualunque apparecchio digitale, è fisso e predefinito dal fabbricante, il numero massimo di cifre che qualunque valore rappresentato puo' occupare. Ovvero: il valore "tre" che in decimale si rappresenta come "3" ed in binario si rappresenta come "101" all'interno di un apparato digitale ha una rappresentazione con un "riempimento di zeri" fino a portarlo alla dimensione stabilita da costruttore. Se immaginiamo di lavorare con uno strumento che ha una rappresentazione a 8 bit il valore "tre" avrà la seguente rappresentazione:

00000101

Le cifre piu' significative della rappresentazione sono "riempite" di zeri.

Il tipo di rappresentazione numerica a lunghezza finita comporta dei problemi quando il numero che si vuol rappresentare supera il valore massimo pre-impostato dal produttore. In quel caso si ha un "overflow" se il numero è positivo, un "underflow" se il numero è negativo.

Vediamo un esempio di "overflow".

Il valore massimo che si puo' rappresentare con 8 Bit (binary digit = cifra binaria) è "DUECENTOCINQUANTACINQUE" che corrisponde a:

11111111

Se volessi rappresentare "DUECENTOCINQUANTASEI" avrei che esso sarebbe rappresentato come:

100000000

Solo la parte in rosso di tale rappresentazione numerica verrebbe conservata sullo strumento MENTRE LA PARTE IN BLU VERREBBE IRRIMEDIABILMENTE PERSA. Questo ci darebbe come risultato della misura il valore:

00000000

cioè "ZERO".

La cifra piu' significativa viene persa: si ha un errore rappresentazione; in questo caso di "overflow", ossia di sforamento verso l'alto. Per evitare questo, a livello di programmazione degli strumenti sono

state prese delle precauzioni che si rivelano efficaci nel 99% dei casi.

D'ora in poi faremo riferimento, a numeri la cui rappresentazione è di tipo "finito", evitando per quanto è possibile i casi limite negli esempi riportati di seguito.

Nota: *se all'interno di una macchina digitale venisse usata una rappresentazione numerica in base 10 i problemi di "overflow" resterebbero invariati.*

Vediamo ora piu' in dettaglio il sistema di numerazione binario.

Esso è un sistema posizionale come la maggior parte dei sistemi di numerazioni usati attualmente.

Puo' essere utilizzato per un sistema di numerazione associando ad ogni carattere binario un valore corrispondente alla potenza di 2. Alla posizione a destra (si parte dalla posizione destra in quanto gli arabi scrivono da destra a sinistra) si associa la potenza di 0, alla posizione seguente si associa la potenza di 1, poi la potenza di 2. Questo sistema è un sistema di numerazione per posizione.

La formula (**) diventa, nel caso in cui la base sia 2:

$$N= a_n * 2^n + \ .....+a_2 * 2^2 + a_1 * 2^1 + a_0 * 2^0$$

In questo caso avremo un sistema di numerazione binario. Come già detto, una cifra binaria viene chiamata Bit (contrazione delle parole inglesi **BI**nary digi**T** -> BI....T -> BIT).

In base a quanto detto il numero binario 1101 esprime il seguente valore

$1 * 2^0 = 1$
$0 * 2^1 = 0$
$1 * 2^2 = 4$
$1 * 2^3 = 8$

$$N=1 * 2^3 + 1 * 2^2 + 0 * 2^1 + 1 * 2^0 = 8 + 4 + 0 + 1 = 13$$

Conversione di un numero binario in numero decimale

Grazie a quello che abbiamo appena visto possiamo "costruire" un algoritmo - una sequenza di istruzioni logiche ed ordinate - che ci consenta di trasformare un numero binario in uno decimale

1. Porre a zero le variabili di conto N=0 e n=0
2. Posizionarsi alla destra del numero
3. Leggere il carattere seguente a sinistra
4. Se il carattere è vuoto, fermarsi (le cifre sono finite)
5. Sommare ad N il valore $\mathbf{2}^{n}$ moltiplicato per la cifra alla posizione n ed incrementare n di 1
6. Tornare al passo 3 e ripetere le istruzioni successive.

Esempio:

Prendiamo il numero binario "1 0 1 1 1 0" e proviamo ad applicare l'algoritmo appena visto.

Passo 1: N=0 ed n=0.
Passo 2: Ci si posiziona alla destra del numero.

cifra	**1**	**0**	**1**	**1**	**1**	**0**
posizione dell'algoritmo						^
posizione della cifra	5	4	3	2	1	0

Passo 3: il numero alla sinistra della posizione e' "0"

Passo 4: non e' "vuoto"

Passo 5: $\mathbf{N} = N + 0 * 2^{0} = N + 0 = 0 + 0 = 0$
n=n+1=1

Passo 6: torno al passo 3

cifra	**1**	**0**	**1**	**1**	**1**	**0**
posizione dell'algoritmo					^	
posizione della cifra	5	4	3	2	1	0

Passo 3: il numero alla sinistra della posizione e' "1"

Passo 4: non e' "vuoto"

Passo 5: $\mathbf{N} = \mathbf{N} + \mathbf{1} * 2^{1} = N + 2 = 0 + 2 = 2$
n=n+1=2

Passo 6: torno al passo 3

cifra	**1**	**0**	**1**	**1**	**1**	**0**
posizione dell'algoritmo				^		
posizione della cifra	5	4	3	2	1	0

Passo 3: il numero alla sinistra della posizione e' "1"

Passo 4: non e' "vuoto"

Passo 5: $\mathbf{N} = \mathbf{N} + \mathbf{1} * 2^{2} = N + 2 = 2 + 4 = 6$
n=n+1=3

Passo 6: torno al passo 3

cifra	**1**	**0**	**1**	**1**	**1**	**0**
posizione dell'algoritmo			^			
posizione della cifra	5	4	3	2	1	0

Passo 3: il numero alla sinistra della posizione e' "1"

Passo 4: non e' "vuoto"

Passo 5: **N = N + 1** * 2^3 = N + 8 = 6 + 8 = 14
n=n+1=4

Passo 6: torno al passo 3

cifra	**1 0 1 1 1 0**
posizione dell'algoritmo	^
posizione della cifra	5 4 3 2 1 0

Passo 3: il numero alla sinistra della posizione e' "0"

Passo 4: non e' "vuoto"

Passo 5: **N = N + 0** * 2^4 = N + 0 = 14 + 0 = 14
n=n+1=5

Passo 6: torno al passo 3

cifra	**1 0 1 1 1 0**
posizione dell'algoritmo	^
posizione della cifra	5 4 3 2 1 0

Passo 3: il numero alla sinistra della posizione e' "1"

Passo 4: non e' "vuoto"

Passo 5: **N = N + 1** * 2^5 = N + 32 = 14 + 32 = 46
n=n+1=6

Passo 6: torno al passo 3

cifra	**1 0 1 1 1 0**
posizione dell'algoritmo	^
posizione della cifra	5 4 3 2 1 0

Passo 3: il numero alla sinistra della posizione e' " "

Passo 4: e' "vuoto" - FINE

Riassumendo:

$0 * 2^0 = 0$
$1 * 2^1 = 2$
$1 * 2^2 = 4$
$1 * 2^3 = 8$
$0 * 2^4 = 0$
$1 * 2^5 = 32$

sommando tutti questi numeri si ottiene **N= 46**

Conversione di decimale in binario

Si può utilizzare un sistema assai rapido

1. Se il numero da convertire e' "1" non c'e' altro da convertire, altrimenti
   a. si divide il numero per 2 e si segna il resto
   b. si sostituisce il valore con il quoziente della divisione
   c. si torna al passo 1
2. si leggono i resti in ordine opposto a quello con cui sono stati trovati.

Esempio:

sia N= 59

passo 1: il valore non e' 1
passo 1.a:

	Valore	Divisore	Quoziente	Resto
1	59	2	29	1

passo 1.b:

	Valore	Divisore	Quoziente	Resto
1	59	2	29	1
2	29			

passo 1.c: torno al passo 1
passo 1: il valore non e' 1
passo 1.a:

	Valore	Divisore	Quoziente	Resto
1	59	2	29	1
2	29	2	14	1

omettendo tutti gli altri passaggi intermedi del processo alla fine si giunge ad ottenere

	Valore	Divisore	Quoziente	Resto
1	59	2	29	1
2	29	2	14	1
3	14	2	7	0
4	7	2	3	1
5	3	2	1	1
6	1	2-	0-	1

(i colori indicano come all'interno della procedura il risultato dell'operazione precedente venga utilizzato in quella successiva)

Da cui, "passo n.2" ,leggendo l'ultima colonna dal basso verso l'alto abbiamo:

N= 1 1 1 0 1 1

Ricapitolando:

59 : 2 resto 1
29 : 2 resto 1
14 : 2 resto 0
7 : 2 resto 1
3 : 2 resto 1
1 : 2 resto 1

Si leggono i resti dal basso verso l'alto **N= 111011**

NOTA MOLTO BENE:

Nel sistema numerico binario
un numero pari termina sempre con un bit 0,
un numero dispari termina sempre con un bit 1.

Se vuoi altri esempi di conversione da
Esempi dalla base dieci a una base qualunque.
consulta questa pagina.
[http://159.149.27.201/binfovet/bioin/biatica/binesem.htm]

Il sistema octale.

Si basa sulla numerazione in base 8. Viene usato principalmente per questioni di comodità di rappresentazione per l'essere umano che lavora con le macchine digitali. Infatti con tale sistema si raggruppano terne di bit ottenendo così un compromesso fra "immediatezza della comprensione" (per l'uomo) del valore ed "orientamento alla macchina" del sistema di numerazione.

Si considerano 3 bit successivi

000 -> 0
001 -> 1
010 -> 2
011 -> 3
100 -> 4
101 -> 5
110 -> 6
111 -> 7

Per cui il numero binario

1 0 1 1 1 0 si scrive 56

101 = **5**
110 = **6**

Il sistema esadecimale.

Si tratta di un sistema di numerazione che si basa su un "alfabeto" formato da 16 simboli diversi. Poichè il sistema numerale arabo (quello con cui correntemente rappresentiamo i numeri) ha soltanto 10 simboli diversi si e' pensato, per poter rappresentare cifre basate sul 16, di aggiungere le prime sei lettere dell'alfabeto. Si ottiene quindi la seguente base si simboli.

0,1,2,3,4,5,6,7,8,9,**A,B,C,D,E,F**

I simboli aggiuntivi hanno il valore binario rappresentato nella tabella seguente:

In decimale	In esadecimale	In binario
10	A	1010
11	B	1011
12	C	1100
13	D	1101
14	E	1110
15	F	1111

E' abbastanza evidente che essi riempiono il "gap", il "vuoto" rappresentativo di tutti i simboli che vanno dal 10 (decimale) al 15 (decimale) o meglio dal 1010 (binario) al 1111 (binario). In questo modo anche tali valori sono rappresentati con una singola cifra.

In questo sistema numerale quando si supera il "15" (in base 10) o il valore "F" (in base esadecimale) si ha il valore "10" (in base esadecimale) ovvero "16" in base decimale. Quando si

vogliono rappresentare numeri piu' grandi la regola da seguire è identica a quella del sistema decimale, a patto di ricordarsi di aggiungere i simboli in più, come si può vedere nella tabella seguente:

In decimale	In esadecimale
.....	.....
14	E
15	**F**
16	**10**
17	11
18	12
.....	.....
25	**19**
26	**1A**
27	1B
28	1C
29	1D
30	1E
31	**1F**
32	**20**
33	21
34	22
.....	.....

NOTA: Può accadere di trovare su molti testi che parlano di macchine digitali la rappresentazione del simbolo di un simbolo numerico senza specificare la base di rappresentazione. Se in certi casi è evidente la sua appartenenza al sistema esadecimale come nel caso delle cifre che contengono valori "alfabetici" [1B, 9C, BA, 3FF, E125DFA] questo non accade con cifre che contencono solo caratteri numerali "arabi" tipo: 19, 210, 1989, ecc... Questi possono essere indifferentemente numeri in base dieci e numeri in base sedici. La differenza di valore fra le due rappresentazioni è grande. Per evitare confusione talvolta si trova il valore "21" in base esadecimale scritto come "21h" oppure "0x21"

Anche questo sistema numerale è importante nell'uso di apparati digitali in quanto è, come il sistema ottale, un compromesso fra "immediatezza della comprensione" (per l'uomo) del valore ed "orientamento alla macchina" del sistema di numerazione.

Come gia' detto sopra il passaggio da un sistema binario ad un sistema esadecimale si effettua considerando quattro bit successivi e rimpiazzandoli con i corrispondenti valori esadecimali. Nella tabella di seguito riportiamo le corrispondenze fra questi 3 sistemi di numerazione:

Tavola di corrispondenza

Base decimale	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Base esadecimale	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**A**	**B**	**C**	**D**	**E**	**F**
Base Binaria	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111

in questo modo un singolo Byte (otto bit) è rappresentato da un numero esadecimale con 2 cifre la prima per i quattro bit più significativi e la seconda per quelli meno significativi:

0x0B, 0xAB, 0x5E, 0xFF ....

dove nell'esempio visto sopra [0xAB] si ha il seguente raggrppamento

A	B
1010	1011
MSB	LSB

la sigla MSB indica i bit più significativi [Most Significant Bits] mentre la sigla LSB indica i bit meno significativi [Less Significant Bits].

Esempi di conversione: "Decimale - Esadecimale" e viceversa.

Vediamo alcune applicazioni di conversione fra sistemi diversi di numerazione. Riportiamo di seguito la formula generica già scritta sopra e vedremo come usarla nel caso del sistema esadecimale.

$$N = a_n * b^n + ...+a_2 * b^2 + a_1 * b^1 + a_0 * b^0$$

In questo caso la base è 16 quindi:

$$N = a_n * 16^n + ...+a_2 * 16^2 + a_1 * 16^1 + a_0 * 16^0$$

Se vogliamo convertire il numero 27 in base 10 alla sua rappresentazione in base esadecimaleavremo che, applicando la formula (in questo caso n=1)

$$27 = 16 + 11 = 1*16^1 + 11*16^0 = 1*16^1 + B*16^0$$

Cioè:

27 in base 10 = **1B** in base 16.

Ora vediamo il caso contrario: una conversione da base 16 a base 10. Vogliamo sapere che rappresentazione ha il numero FB3 in base 16 nella base 10. In questo caso si ha che n=2 ed applicando la formula:

$$0xFB3 = F*16^2 + B*16^1 + 3*16^0 =$$

$$= 15*16^2 + 12*16^1 + 3*16^0 =$$

$$= 15 * 256 + 12 * 16 + 3 =$$

$$= 3840 + 176 + 3 =$$

$$= 4019$$

Per convertire un ottetto in esadecimale si suddivide in due gruppi di 4 bits, che corrispondono a

2	A	D	5
0010	1010	1101	0101

Ad esempio 10111001 si divide in due gruppi e si traduce in B9 essendo (anche semplicemente consultando la "Tavola di corrispondenza") .

1011= B
1001= 9

Come si può vedere la conversione binaria/esadecimale e' molto meno laboriosa di quella binaria/decimale.

Le operazioni con il sistema binario.

Con ogni sistema di numerazione sono definite delle operazioni note come "somma fra 2 numeri", "moltiplicazione fra 2 numeri" e così via e per ognuna di esse viene definito un "modus operandi" specifico.

Con le operazioni fra numeri nel sistema decimale impariamo a lavorare fin da piccoli e non usiamo più alcuno sforzo quando le utilizziamo - ricordiamo qui giusto a titolo di esempio le famose/famigerate "tabelline pitagoriche" per le moltiplicazioni che abbiamo dovuto imparare a memoria.

Per il sistema binario, come per gli altri sistemi di numerazione, le operazioni - e le relative tabelle - subiscono degli adattamenti al sistema stesso, in modo da avere operazioni coerenti con il metodo rappresentativo scelto. Vediamo in dettaglio le operazioni con il sistema binario.

La somma

E' una operazione che si presenta come una serie finita di combinazioni possibili (ovvero ha 4 casi) e segue le regole generali dell'addizione nel sistema decimale, ma con i limiti imposti dalla numerazione binaria (solo 2 cifre). Questi casi sono rappresentati nella tabella seguente:

+	0	1
0	0	1
1	1	10

Nota:

per il sistema decimale la tabella delle addizioni sarebbe cosi':

+	0	1	2	3	...	9
0	0	1	**2**	**3**	**...**	**9**
1	1	2	**3**	**4**	**...**	**10**
2	**2**	**3**	**4**	**5**	**...**	**11**
....						
9	**9**	**10**	**11**	**12**	**...**	**18**

La parte color malva rappresenta il sottoinsieme che governa il sistema binario, ovvero la tabella pecedente, dove al simbolo 10 (in base due) e' stato sostitutito il simbolo 2 (in base dieci), il cui valore è il medesimo.

Nota: in seguito ometteremo, per ovvi motivi, di riportare una tabella pitagorica per l'operazione "moltiplicazione in base 10".

Esempio:

Si voglia ottenere la somma in binario dei numeri 11 e 13. Il modo di operare è il seguente:

sommo la prima colonna (applico le regole della tabella alle cifre in giallo degli addendi)

addendo (11)		1	0	1	1
addendo (13)		1	1	0	1
riporto				***1***	***0***
somma					**0**

sommo la seconda colonna

addendo (11)		1	0	1	1
addendo (13)		1	1	0	1
riporto			***1***	*1*	*0*
somma				**0**	**0**

sommo la terza colonna

addendo (11)		1	0	1	1
addendo (13)		1	1	0	1
riporto		***1***	*1*	*1*	*0*
somma			**0**	**0**	**0**

sommo la quarta colonna

addendo (11)		1	0	1	1
addendo (13)		1	1	0	1
riporto	***1***	*1*	*1*	*1*	*0*
somma	**1**	**1**	**0**	**0**	**0**

ricapitolando:

	1011	**11**
	1101	**13**
Somma	**11000**	**24**

La moltiplicazione

Anche questa è una operazione che si presenta come una serie finita di combinazioni possibili (ovvero ha 4 casi). Come per la somma, esiste una tabellina delle moltiplicazioni che risulta essere ovviamente ridotta rispetto a quella del sistema decimale; in ogni caso sono applicabili le regole imparate per il

sistema decimale in quanto a livello di rappresentazione simbolica il sistema binario è un sottoinsieme del sistema decimale *(ovvero i simboli del sistema binario sono contenuti all'interno del sistema decimale)*.

*	0	1
0	0	0
1	0	1

Esempio:

si vogliano moltiplicare due numeri binari 1011 x 101:

		1	**0**	**1**	**1**	x
			1	**0**	**1**	=
		1	0	1	1	1
	0	0	0	0	-	0
1	0	1	1	-	-	1
1	**1**	**0**	**1**	**1**	**1**	**Risultato**

Nota bene:
visto che nella moltiplicazione binaria, come in quella decimale, se si moltiplica un numero per 1 il numero resta inalterato e se invece lo si moltiplica per zero esso diventa zero, per operare rapidamente una moltiplicazione basta semplicemente trascrivere il moltiplicando se la cifra del moltiplicatore è uno altrimenti si scrivono tanti zeri quante sono le cifre del moltiplicando. Ovviamente vanno rispettare le regole di scorrimento a sinistra delle cifre nel procedere dell'operazione.

I numeri negativi

Fino ad ora si sono trattati solo dei numeri positivi.

La rappresentazione di numeri negativi implica un sistema di codifica, che implica l'utilizzo di un bit, detto "bit di segno" solitamente è il bit piu' significativo di un byte - ovvero il primo a sinistra. Se tale bit è zero i bit successivi sono considerati rappresentazione di un numero positivo altrimenti se tale bit è uno i bit successivi sono considerati rappresentazione di un numero.

In questa sede ci limiteremo a trattare numeri interi rappresentati al massimo da 8 bit [1 Byte] per questioni di semplicità di comprensione; le problematiche non cambiano nel caso reale dove i numeri vengono rappresentati da più Byte consecutivi - in genere 2,4 oppure 8.

Come detto sopra, prendiamo un bit e posizioniamolo alla sinistra del numero, se il numero è positivo attribuiamo il bit 0, se negativo attribuiamo il bit 1.

Es: dato il numero:

00000101 corrispondente a "cinque"

se consideriamo il primo bit come bit di segno allora tale rappresentazione vale "più cinque".

In questo caso il numero dei bit -1 rappresenta il valore assoluto. Mentre 10000101 corrisponde a -5.

Questo sistema presenta una serie di inconvenienti:

- **per effettuare una operazione sugli interi positivi e negativi si deve verificare il bit del segno;**
- **si deve definire una operazione di sottrazione dei valori assoluti;**
- **esiste una configurazione rappresentata come "10000000" che viene letta come zero negativo.**

Il complemento a 1

Per risolvere le difficoltà possiamo considerare un numero negativo come complemento. Potremo avere:

complemento a 1 di 0 = 1
complemento a 0 di 1 = 0

Per esempio consideriamo il numero 6 in binario

0110

il suo complemento sarà:

1001

Attenzione! In questa rappresentazione persiste l'inconveniente di avere un doppio 0: "+0" o "zero positivo" e "-0" o "zero negativo" .

Il complemento a 2

Per convenzione operiamo un complemento ad 1 ed aggiungiamo 1 al risultato ottenuto.

Se consideriamo il numero binario 00001011 il suo complemento ad 1 risulterà corrispondere a: 11110100, il complemento a 2 risulterà 11110101 ed é ottenuto aggiungendo +1.

L'addizione di un numero con il suo opposto sarà:

00001011

11110101

100000000

In questo tipo di rappresentazione il numero binario 10000000 corrisponde a -128.

Il complemento a 2 possiede un solo 0.

La virgola

Un primo approccio è quello di considerare una parte intera ed una parte decimale. Se consideriamo una rappresentazione in 8 bit possiamo definire una parte costituita da 4 bit per la parte intera e 4 bit per la parte decimale.

In questo caso possiamo rappresentare dei numeri positivi che vanno da 0000.0000 a 1111.1111 ossia da 0.0 a 15.9375. Un campo di variazione che appare ridotto, se inoltre si considerano i numeri positivi e negativi le possibilità si riducono ad un campo tra -8.9375 e + 7.9375.

La virgola mobile

Se consideriamo i numeri frazionari con l'ausilio di una rappresentazione esponenziale (logaritmo). Un numero può essere rappresentato in diversi modi. Ad esempio 1984.128 può essere rappresentato come:

$19.84128 * (10^{2})$
$0.1984128 * (10^{4})$
$1984128 * (10^{-3})$

Si osserva che la virgola varia al variare della potenza del 10 utilizzata. Definiremo una rappresentazione normalizzata quella che consente di avere la virgola in posizione tale che la prima cifra significativa sia a destra della virgola.

Ad esempio:

$$\mathbf{0.1011101 \times 2^{5}}$$

è una rappresentazione normalizzata.

Si definisce la parte delle cifre significative in una rappresentazione normalizzata, mentre l'**esponente** é una potenza del 2 associata a questa rappresentazione normalizzata.

Relativamente all'esempio in questione:

0.1011101 e' la
00000101 (ovvero 5) e' l'esponente.

Quindi, la rappresentazione di tale valore in una memoria di un apparato digitale in cui è necessari 1 byte (otto bit) per cifra, sarà la seguente:

mantissa	esponente
0101110100000101	

mentre in un apparato in cui sono necessari 2 byte per cifra, sarà la seguente:

bit riservati per la mantissa	bit riservati per l'esponente
0000000001011101	0000000000000101
in blu il Most Significant Byte	in rosso il Less Significant Byte

e cosi' via.

In generale, possiamo avere una rappresentazione di un numero con virgola mobile con una quantità arbitraria di bit riservati per l'esponente e per la mantissa.

In questa rappresentazione si può considerare il Most Significant Bit sia della mantissa che dell'esponente come bit di segno, così da poter rappresentare cifre di tipo "reale" e "con segno". La loro rappresentazione sara' la seguente (il bit di segno viene riportato in verde):

in un apparato a 1 byte:

bit riservati per la mantissa	bit riservati per l'esponente
0101110100000101	

in un apparato a 2 byte:

bit riservati per la mantissa	bit riservati per l'esponente
0000000001011101	0000000000000101
in blu il Most Significant Byte	in rosso il Less Significant Byte

Rappresentazione dei caratteri

Se volessimo rappresentare le lettere dell'alfabeto, i numeri e dei caratteri speciali, ci accorgeremmo di avere la necessità di rappresentare almeno 36 caratteri, inoltre esiste la necessità di definire le maiuscole e le minuscole. Attualmente si adotta un codice **ASCII** (**A**merican **S**tandard **C**ode for **I**nformation **I**nterchange) a 8 bit, anche se alcuni computer utilizzano un ASCII a 7 bit. Per molti computer I.B.M. e compatibili si ha una rappresentazione con un codice **EBCDIC** (**E**xtended **B**inary **C**oded **D**ecimal **I**nterchange **C**ode) che utilizza 8 bit.