ARITMETICA BINARIA

Le operazioni che possono essere fatte sui numeri binari, sono le stesse che vengono effettuate sui numeri decimali. Due numeri binari possono essere quindi sommati, sottratti, moltiplicati e divisi.

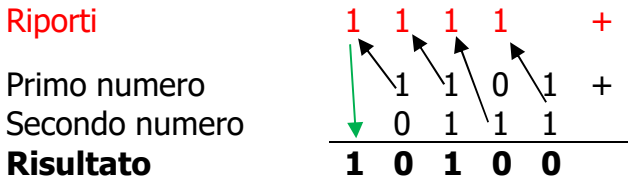
SOMMA.

La somma viene eseguita secondo le regole per la somma di due bit, di seguito riportate:

0 + 0 = 0
1 + 0 = 1
0 + 1 = 1
1 + 1 = 10 in cui 1 è il riporto alla cifra successiva.

Esempio:

supponiamo di voler sommare i due numeri binari: 1101 e 0111, avremo:



Effettivamente 1101_2 (13_{10}) + 0111_2 (7_{10}) = 10100_2 (20_{10})

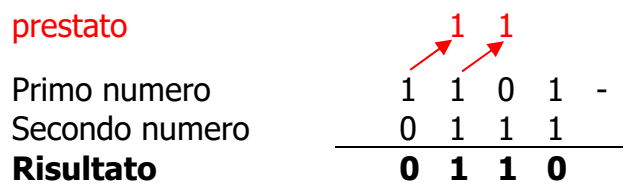
SOTTRAZIONE.

La sottrazione viene eseguita secondo le regole per la sottrazione di due bit, di seguito riportate:

0 - 0 = 0
1 - 0 = 1
1 - 1 = 0
0 - 1 = 1 dopo essersi prestato 1 dalla cifra a sinistra che diventa 0: se la cifra a sinistra è 0 ed alla sua sinistra c'è un 1, lo 0 diventa 1.

Esempio:

supponiamo di voler sottrarre i due numeri binari: 1101 e 0111, avremo:



Effettivamente 1101_2 (13_{10}) - 0111_2 (7_{10}) = 0110_2 (6_{10})

supponiamo di voler sottrarre i due numeri binari: 10010 e 1101, avremo:

prestato		1	1		1	
Primo numero	1	0	0	1	0	-
Secondo numero		1	1	0	1	
Risultato	**0**	**0**	**1**	**0**	**1**	

Effettivamente 10010_2 (18_{10}) - 1101_2 (13_{10}) = 101_2 (5_{10})

In realtà la sottrazione viene eseguita con la regola del complemento a 2 come vedremo in seguito.

MOLTIPLICAZIONE.

La moltiplicazione viene eseguita secondo le regole per la moltiplicazione di due bit, di seguito riportate:

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

Come per la moltiplicazione con i numeri decimali, si moltiplica ogni cifra del secondo fattore per tutte le cifre del primo. Dopo aver moltiplicato la prima cifra del secondo fattore, si passa a moltiplicare la seconda, spostando di una posizione a sinistra il risultato del prodotto della seconda cifra del secondo fattore per tutte le cifre del primo fattore. Si continua così finchè non si sono moltiplicate tutte le cifre del secondo fattore. I risultati delle moltiplicazioni così scalate, si sommano per ottenere il risultato della moltiplicazione.

Esempio:

supponiamo di voler moltiplicare i due numeri binari: 110 e 10, avremo:

Primo fattore		1	1	0	*
Secondo fattore			1	0	
		0	0	0	+ → 110 * 0 = 000
	1	1	0		→ 110 * 1 = 110
Risultato	**1**	**1**	**0**	**0**	

Effettivamente 110_2 (6_{10}) * 10_2 (2_{10}) = 1100_2 (12_{10})

In realtà la moltiplicazione, ad esempio può essere sostituita con un insieme di somme: 3 * 2 = 6 può essere calcolata o come la somma di tre volte il numero 2 (2+2+2 = 6) o la somma di due volte il numero 3 (3 + 3 = 6).

DIVISIONE.

Anche la divisione segue le stesse regole della divisione tra i numeri decimali. Ricordiamo che la divisione si compone di quattro parti: dividendo (numero che vogliamo dividere), divisore (numero per cui vogliamo dividere il dividendo), quoziente (risultato intero della divisione), resto (parte restante della divisione).

Per spiegare come avviene la divisione tra numeri binari facciamolo tramite esempi.
Supponiamo di voler dividere il numero 10010_2 (18_{10}) per 10_2 (2_{10}):
Il divisore è 10, quindi il minimo numero di cifre del dividendo da considerare due che segniamo come mostrato in figura:
ora 10 del dividendo sta 1 volta al divisore

1	0	0	1	0	1	0
					1	

Otteniamo:

1	0	0	1	0	1	0
1	0				1	
=	=	0				

Abbiamo sottratto alle prime due cifre (10) del dividendo 10 ottenuto dal prodotto di ciò che compare nel quoziente (1) per il divisore (10). Quindi abbiamo abbassato lo 0 a destra di 10 nel dividendo. Ovviamente 0 diviso 10 ci sta 0 volte.

1	0	0	1	0	1	0
1	0				1	0
=	=	0				
		0				
		=	1			

Ora abbassiamo 1 a destra di 100 nel dividendo. Ovviamente 1 diviso 10 ci sta 0 volte. E sottraiamo…. Quindi abbassiamo l'ultimo 0 del dividendo.

1	0	0	1	0	1	0	
1	0				1	0	0
=	=	0					
		0					
		=	1				
			0				
			1	0			

A questo punto 10 sta al divisore 1 volta:

```
1  0  0  1  0 | 1  0
1  0           | 1  0  0  1
=  =  0
      0
      =  1
         0
         1  0
         1  0
         =  =
```

Effettivamente 10010_2 (18_{10}) diviso 10_2 (2_{10}) = 1001_2 (9_{10}) con resto 0.

Altro esempio. Vogliamo dividere 10001_2 (17_{10}) per 11_2 (3_{10})

```
1  0  0  0  1 | 1  1
   1  1       | 1  0  1
=  =  1  0
      0  0
      1  0  1
         1  1
      =  1  0
```

Effettivamente 10001_2 (17_{10}) diviso 11_2 (3_{10}) = 101_2 (5_{10}) con il resto di 10_2 (2_{10}).

I NUMERI RELATIVI.

I numeri relativi sono numeri interi che possono assumere valori negativi o nulli (0) oltre che positivi. I numeri relativi sono segnati dal simbolo – (meno) per indicare i valori negativi, + quelli positivi.
Ma come si rappresentano i numeri relativi binari all'interno dell'elaboratore?

Modulo e segno.

All'interno dell'elaboratore per rappresentare un numero relativo in binario, si usa la notazione in *modulo e segno*. Questa notazione pone al bit più significativo (il bit più a sinistra) dei bytes assegnati per rappresentare il numero intero, il "segno". Questo bit viene infatti chiamato comunemente "bit di segno" ed assume il valore 1 se il numero da rappresentare è negativo, 0 altrimenti.
Gli altri bit vengono utilizzati per rappresentare il numero intero senza segno (in modulo).

Supponiamo, ad esempio, di voler rappresentare in binario, il numero relativo -5 in un byte.

Bit di segno	modulo						
1	0	0	0	0	1	0	1
-	5						

Se volessimo rappresentare il numero +7, analogamente:

Bit di segno	modulo						
0	0	0	0	0	1	1	1
+				7			

Il problema di questo tipo di rappresentazione è che, durante operazioni di calcolo tra numeri relativi, può verificarsi un ***overflow (trabocco).***

Un overflow consiste in un riporto (o un prestito) in un'operazione aritmetica, che "sporca" il bit di segno, ottenendo quindi un risultato sbagliato.
Facciamo un esempio considerando, per semplicità di calcolo, i numeri su cui effettuare un'operazione aritmetica (supponiamo la somma) contenuti in semi-byte, e supponiamo di voler sommare +5 con +5 il cui risultato è banalmente +10:

	Bit di segno				
riporto	1		1		
	0	1	0	1	+
	0	1	0	1	=
risultato	1	0	1	0	
	-		2		

A causa dell'overflow otteniamo come risultato –2, invece di +10.

Complemento a 2.

Un altro modo per rappresentare i numeri negativi, più in generale i numeri relativi, è il complemento a 2.

Questo metodo trasforma un numero negativo in un altro che lo "rappresenta". Quindi se questo numero negativo viene sommato ad un numero positivo questa somma produce, in realtà, la "differenza" tra i due numeri dati. In poche parole, se devo fare la seguente operazione 9 – 7 = 2, possiamo pensare di svolgere la seguente somma 9 + (-7) = 2. Il problema è riuscire a rappresentare -7 con un altro numero. Il numero che rappresenta il numero negativo si chiama *complemento a 2*. L'operazione di complementazione a 2 in binario, si ottiene facendo i seguenti passaggi:

$7_{10} = 0111_2$

Il primo passo consiste nel trasformare gli 1 in 0 e gli 0 in 1 del numero binario che vogliamo complementare:

$\Longrightarrow \qquad 0111_2 \qquad 1000_2$

quindi si somma 1 al numero ottenuto:

$1000_2 + 1 = 1001_2.$

Il numero ottenuto (1001_2) si chiama "complemento a 2" del numero dato (nel nostro caso 7) e rappresenta il suo negativo.
Possiamo quindi effettuare la nostra sottrazione trasformata in realtà tra la somma di 9 e il complemento di 7. Questa somma però deve avere un piccolo artificio, nel caso si produca

un riporto oltre i bit necessari per rappresentare il più grande di questi, questo riporto va eliminato.

In definitiva:

		riporti	1			1			
9	+			1	0	0	1	+	
-7	(il suo complemento)			1	0	0	1	=	
			~~1~~	0	0	1	0		
						2			

L'ultimo riporto, come si può notare, è stato eliminato.