4. Operazioni aritmetiche con i numeri binari

Contare con i numeri binari
Prima di vedere quali operazioni possiamo effettuare con i numeri binari, iniziamo ad imparare a contare in binario:
Iniziamo a contare da 0, supponendo di avere a disposizione solo le cifre '0' e '1'. Dopo aver contato '0' e '1' viene il "due" (scritto in parole perché in binario non esiste una cifra che lo descrive): rappresentiamo questo numero con '10', cioè una duina e zero unità.
Proseguendo il conteggio si ha il numero '11' cioè una duina ed una unità, e così via:

```
  0
  1
 10
 11
100
101
110
111
…
```

Per incrementare di un'unità il procedimento è quindi molto semplice: si comincia incrementando la cifra più a destra, se questa è zero diventa 1, se è uno diventa zero col riporto di uno, riporto che sommiamo alla cifra immediatamente a sinistra.

La somma aritmetica:
Cominciamo con il considerare senza segno i numeri binari della tabella precedente; come detto, se formattati a 8 bit (cioè della dimensione di un Byte) il loro valore va da 00000000 (0) a 11111111 (255).
La somma aritmetica binaria è poi molto simile a quella decimale; le cifre dei due addendi sono sommate colonna dopo colonna, tenendo conto dell'eventuale riporto, da sommare con gli eventuali operandi della colonna adiacente a sinistra.
Vediamo la regola applicata nei 4 casi possibili:

Binario	Decimale
0 + 0 = 0 con riporto di 0	0 + 0 = 0 (00)
0 + 1 = 1 con riporto di 0	0 + 1 = 1 (01)
1 + 0 = 1 con riporto di 0	1 + 0 = 1 (01)
1 + 1 = 0 con riporto di 1	1 + 1 = 2 (10)

Vediamo alcuni esempi:

Binario	Decimale
Riporto ... 100 + 10 = ---------- 110	 4 + 2 = ---------- 6

Binario	Decimale
Riporto 111. 1101 + 11 = ---------- 10000	13 + 3 = ---------- 16

Binario	Decimale
Riporto 1.. 110 + 11 = ---------- 1001	6 + 3 = ---------- 9

Bisogna comunque notare che i processori eseguono questo compito in pochi nanosecondi, con numeri a 8, 16, 32 bit.

Se la somma viene operata con dati formattati (per esempio a 8 bit) può succedere che il risultato non sia attendibile; l'errore si manifesta quando si verifica la necessità di generare un riporto:

Binario	Decimale
Riporto 1....... 11001011 + 10000100 = ---------- 01001111	203 + 132 = ---------- 79(?)

Questo succede perché la somma di 203 e 132 darebbe come risultato 335 in decimale, che in binario ha bisogno di 9 bit, provocando un `overflow`, cioè un utilizzo di bit non consentiti. Infatti nel nostro caso abbiamo solo 8 bit, così il bit più significativo viene troncato; il risultato è `01001111`, cioè 79 in decimale. Si nota come 79 corrisponda esattamente al risultato corretto 335 meno 256 (il peso del bit mancante).

Per ovviare a questo inconveniente i costruttori di processori hanno previsto un particolare bit (`flag Carry`) che assume comunque il valore del riporto della somma di numeri binari della stessa dimensione
Questo bit (insieme ad altri dello stesso tipo) è conservato in un particolare Registro del processore 80x86, facilmente consultabile, per consentire al programmatore assembly di prendere (eventualmente) i provvedimenti del caso.
Infine vediamo la tabella di verità di una somma aritmetica tra due numeri di 1 bit.

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

La sottrazione aritmetica
L'aritmetica sui numeri binari senza segno prevede naturalmente anche sottrazione, anche in questo caso del tutto simile a quella decimale; la cifra del minuendo è sottratta dalla cifra del sottraendo colonna dopo colonna, tenendo conto dell'eventuale prestito da sottrarre con gli eventuali operandi della colonna adiacente a sinistra.
La regola da applicare a ciascuna colonna è, nei 4 casi possibili:

Binario	Decimale
0 – 0 = 0 con prestito di 0	(00) 0 – 0 = 0
0 – 1 = 1 con prestito di 1	(10) 2 – 1 = 1
1 – 0 = 1 con prestito di 0	(01) 1 – 0 = 1
1 – 1 = 0 con prestito di 0	(01) 1 – 1 = 0

L'interpretazione decimale della tabella mostra che la seconda sottrazione non può essere fatta se lo 0 del sottraendo non viene inteso come cifra meno significativa del numero 2 (10 in binario).

Binario	Decimale
Prestito 1.. 101 - 11 = ---------- 010	 5 - 3 = ---------- 2

Binario	Decimale
Prestito .1.. 1101 - 11 = ---------- 1010	 13 - 3 = ---------- 10

Continuando con una panoramica sulle istruzioni presenti nel processore 80x86, esso prevede 2 istruzioni specifiche, la SUB (Sottrazione aritmetica tra numeri interi) e la SBB (Sottrazione aritmetica tra numeri interi, con prestito), eseguita in pochi nanosecondi con operandi di tutte le dimensioni (a 8, 16, 32 bit). Anche in questo caso va segnalata la possibilità di avere problemi di overflow anche nell'esecuzione della sottrazione; in questo caso l'errore si manifesta quando si tenta di sottrarre un minuendo più grande del sottraendo.
La flag Carry assume in questo caso comunque il valore del prestito (borrow) della differenza di numeri binari della stessa dimensione ed è facilmente consultabile (insieme ad altri bit dello stesso tipo) in un particolare Registro del processore 80x86, a disposizione del programmatore assembly.
Infine vediamo la tabella di verità di una sottrazione aritmetica tra due numeri di 1 bit.

A	B	Differenza	Prestito
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

La moltiplicazione aritmetica
La regola da applicare a ciascuna colonna è, nei 4 casi possibili, la stessa applicata nelle operazioni con numeri decimali:

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

Vediamo quindi un esempio di come moltiplicare 1010 (10 in decimale) per 11 (3 in decimale)

Binario	Decimale
1010 * 11 = ---------- 1010 1010. ---------- 11110	10 * 3 = ---------- 30

Per moltiplicare basta quindi utilizzare il metodo che tutti conosciamo per la moltiplicazione di numeri a base 10.
Si moltiplica la cifra più a destra del secondo fattore per tutte le cifre del primo fattore, poi la cifra immediatamente più a sinistra del secondo fattore per tutte le cifre del primo (traslando ogni volta verso sinistra di una posizione) e così via per tutte le cifre del secondo fattore.
I risultati vengono infine sommati.

Anche in questo caso il processore 80x86 prevede una specifica istruzione, MUL, eseguita in pochi nanosecondi con operandi di tutte le dimensioni (a 8, 16, 32 bit).

Analogamente a quanto succede nella moltiplicazione a base 10, per cui se ad un numero decimale aggiungo uno zero a destra, si ottiene il numero di partenza moltiplicato per 10, per i numeri binari, dato che sono a base 2, aggiungendo uno 0 a destra si moltiplica il numero per 2.

Binario	Decimale
101 * 10 = 1010 ---------- 101 * 100 = 10100 ---------- 101 * 1000 = 101000 ---------- 101 * 10000 = 1010000	5 * 2 = 10 ---------- 5 * 4 = 20 ---------- 5 * 8 = 40 ---------- 5 * 16 = 80

La divisione aritmetica
Ovviamente anche l'operazione della divisione è presente come istruzione nel processore 80x86 (DIV, Divisione tra numeri Interi senza segno). Ricordando quanto detto prima per la moltiplicazione di un numero per 2, 4, 8… è interessante notare che "togliendo" n zeri alla destra di un numero binario lo si divide per 2^n.

Binario	Decimale
1010 : 10 = 101 ---------- 10100 : 100 = 101 ---------- 1010000 : 100 = 10100 ---------- 1010000 : 1000 = 1010	10 : 2 = 5 ---------- 20 : 4 = 5 ---------- 80 : 4 = 20 ---------- 80 : 8 = 10

Nel caso più generale, anche in questo caso le regole per effettuare la divisione sono molto simili alle regole per la divisione tra numeri in base 10.
Prendiamo come esempio la seguente divisione: `10011011 : 1011`.
La domanda a cui si risponde è: quante volte sta `1011` in `10011011`?
Si comincia cercando la prima parte del dividendo che sia maggiore (o uguale) al divisore, nel nostro caso le prime 5 cifre del dividendo, partendo da sinistra, sono costituiscono un numero maggiore del divisore: `10011 > 1011`.
Quante volte sta `1011` in `10011`? 1 volta, quindi nel risultato si comincia scrivendo `1`.
Il resto è pari alla differenza tra `10011` e `1011`, cioè: `1000`.
A questo punto si procede prendendo un'altra cifra del dividendo (la prima ancora non utilizzata, cioè la sesta partendo da sinistra). Questa cifra viene posta a destra del resto. (che diventa `10000`)
Si continua calcolando quante volte sta `1011` (il divisore) in `10000` (il numero ottenuto poco prima). Il risultato è 1 (dato che `10000` è maggiore di `1011`), con resto `10000 – 1011 = 101`.
Ripetendo quanto fatto finora, si aggiunge al resto la settima cifra partendo da sinistra del dividendo, ottenendo questo numero: `1011`.
Quante volte sta il divisore `1011` in `1011` (numero calcolato qui sopra)? Ovviamente 1 volta con resto `0`.
Terminiamo con l'ultimo passaggio, portando l'ultima cifra del dividendo a destra dell'ultimo resto ottenendo `01`. Quante volte sta il divisore `1011` in `01`? 0 volte con resto `01`.
Quindi il risultato della divisione è: `1110` con resto `1`.