7. Numeri binari negativi

Rappresentazione in modulo e segno

I numeri binari che abbiamo visto finora sono detti "unsigned integer", cioè interi senza segno, dato che rappresentano numeri non negativi.
Se risulta necessario utilizzare anche numeri negativi, come accade di solito, si deve suddividere l'insieme dei numeri rappresentabili con le cifre a disposizione in due insiemi. Supponendo di lavorare con numeri da 16 bit, gli interi senza segno rappresentabili sono come abbiamo visto prima $2^{16} = 65536$, mentre dovendo dividere l'insieme in due parti avremo che i numeri interi positivi rappresentabili sono $2^{15} = 32768$; lo stesso vale per gli interi negativi: $2^{15} = 32768$.
Per identificare i due insiemi si utilizza il bit più significativo: se è uguale a 1 allora il numero è negativo, altrimenti zero.
Vediamo nella tabella qualche esempio:

Binario	Decimale
1011	- 3
1010	- 2
1001	- 1
1000	0
0000	0
0001	+ 1
0010	+ 2
0011	+ 3

Come vediamo servono n bit per codificare numeri da 0 a 2^{n-1} e da 0 a -2^{n-1}.
Il primo difetto che salta all'occhio è che ci sono due modi per rappresentare lo '0': 1000 e 0001, come se si facesse distinzione tra '+0' e '-0'.
Quello visto fin qui è solo uno dei metodi per rappresentare i numeri binari, il prossimo che vediamo è detto "rappresentazione in complemento a 2".

Rappresentazione in complemento a 2

Il complemento a due è il metodo più diffuso per la rappresentazione dei numeri negativi in informatica.
La sua enorme diffusione è data dal fatto che i circuiti di addizione e sottrazione non devono esaminare il segno di un numero rappresentato con questo sistema per determinare quale delle due operazioni sia necessaria, permettendo tecnologie più semplici e maggiore precisione; si utilizza infatti un solo circuito, il sommatore, sia per l'addizione che per la sottrazione.

Un numero binario di *n* cifre può rappresentare con questo metodo i numeri compresi fra -2^{n-1} e $+2^{n-1}-1$, così un binario di 8 cifre può rappresentare i numeri compresi tra -128 e +127.
Da notare che esiste una sola rappresentazione dello zero: quando tutti i bit sono zero, eliminando così la ridondanza dello zero che si verifica con la rappresentazione in modulo e segno.

Questo metodo consente di avere un'unica rappresentazione dello zero, e di operare efficientemente addizione e sottrazione sempre avendo il primo bit a indicare il segno.
Per ottenere il corrispettivo negativo partendo da un numero positivo e il viceversa è sufficiente invertire tutti i bit (si applica l'operazione NOT), ed infine si somma 1.

Vediamo 3 esempi:

- `0000 0101` rappresenta il numero `+ 5`.
  Il primo passo consiste nell'invertire tutti i bit, si ottiene quindi: `1111 1010`.
  L'ultimo passo consiste nel sommare 1 al numero precedente. Si ottiene `1111 1011` che rappresenta `-5` in complemento a `2`.

- `1111 0100` rappresenta il numero `-12`
  Il primo passo consiste nell'invertire tutti i bit, si ottiene quindi: `0000 1011`.
  L'ultimo passo consiste nel sommare 1 al numero precedente. Si ottiene `0000 1100` che rappresenta `+ 12`

- `0000 0000` rappresenta il numero `0`
  Vediamo che procedendo come sopra otteniamo che la stessa rappresentazione per lo `0`.
  Invertendo tutti i bit otteniamo `1111 1111`. Sommando 1 al numero ottenuto otteniamo `1 0000 0000`, un numero di 9 bit. Dato che si lavora con 8 bit l'`overflow` viene ignorato, quindi il risultato è: `0000 0000`