Rappresentazione in modulo e segno

Nella rappresentazione in modulo e segno, viene usato il bit più a sinistra, cioè il più significativo, per indicare esplicitamente il segno con la seguente convenzione: 0 = segno positivo; 1 = segno negativo.
Gli altri bit disponibili sono utilizzati per rappresentare il valore assoluto come numero binario «puro». L'intervallo dei valori rappresentabili con n bit è: $-2^{(n-1)}-1 <= x <= +2^{(n-1)}-1$. La quantità di valori è comunque 2^N e non 2^N-1 , come specificato sopra, perché ci sono due rappresentazioni del valore zero.

Rappresentazione in complemento a due

Si definisce il complemento in una base B di un valore x espresso con n cifre: $C_B(x) = B^n - x$
Per esempio il complemento a dieci di 47 è 53 cioè $C_{10}(47)=10^2 - 47=53$ dove n=2 sono le cifre di 47 e possiamo vedere che si può ottenere, oltre che con l'applicazione della formula, anche facendo il complemento a nove di ogni singola cifra e sommando 1 al risultato (cioè il complemento a 9 di 4 che è 5 e il complemento di 7 che è 2, mi dà 52, a cui aggiungo 1 ed ottengo 53). Questa osservazione è molto utile per fare il complemento a due di un valore binario. Supponiamo ad esempio di volerlo calcolare per il valore 47 che in binario si scrive : 00101111 (lavorando con registri ad 8 bit); possiamo definire e applicare la funzione $C_2(x)= 2^8 - x$ nel caso di cioè 256 - 47= 209; in binario $C_2(00101111)= 100000000 - 00101111=$ 11010001 ma in modo più agevole possiamo fare il complemento a uno di ogni cifra (che consiste banalmente nello scambiare i valori zero con uno e viceversa) e poi sommare 1: 00101111 complementato a 1 (cioè invertendo le cifre) diventa 11010000 e poi sommando 1 → 11010001

In alternativa al suddetto metodo, ancora più semplicemente, si può ottenere il complemento a due di un numero binario nel seguente modo: scorrere il numero da destra lasciando i bit inalterati fino al primo uno (compreso) e invertire il valore dei successivi bit (cioè se 47 si scrive 00101111, lascio inalterato il primo uno a destra 0 ◄0 ◄1 ◄0 ◄1 ◄1 ◄1◄ [1] e inverto, muovendomi verso sinistra, il resto 110100).

Con il metodo del complemento a due il bit più significativo rappresenta il segno, rispettando la convenzione più diffusa: 0=segno positivo; 1=segno negativo. La regola generale per questa rappresentazione è la seguente: se il valore è positivo lo si rappresenta normalmente in binario avendo però cura di scrivere anche gli zeri non significativi a sinistra (in modo che sia sempre presente il bit del segno); se il valore è negativo si converte in binario e poi si effettua il complemento a due.
L'intervallo dei valori rappresentabili in complemento a due con n bit è: $-2^{(n-1)} <= x <= +2^{(n-1)}-1$
Il massimo intero positivo rappresentabile è in modulo inferiore di uno al minimo negativo rappresentabile perché il valore zero fa intrinsecamente parte dei valori positivi. La rappresentazione in complemento a due

è la più utilizzata anche per un motivo legato ai calcoli aritmetici: la sottrazione tra due valori espressi in questa forma può infatti essere effettuata comodamente facendo una somma tra il primo valore ed il complemento a due del secondo (scartando il bit di riporto oltre il bit del segno).
Il fatto che le sottrazioni si trasformino in somme è molto importante ai fini della semplificazione dei circuiti interni del sistema di elaborazione in quanto non sono necessari dispositivi adibiti a svolgere queste operazioni. Se poi si tiene conto che le moltiplicazioni si possono ottenere «via software» come successioni di somme e le divisioni analogamente come successioni di sottrazioni, emerge che l'unico circuito aritmetico necessario alla macchina è, almeno in linea teorica, il circuito sommatore.
Nello svolgere le operazioni di somma e sottrazione occorre però porre attenzione agli errori che si possono verificare e che sono dovuti alla limitatezza dell'intervallo dei valori rappresentabili. Si possono infatti verificare situazioni di *overflow* o traboccamento dovute al fatto che il risultato di una operazione esce da tale intervallo.
Ci sono vari sistemi per accorgersi di questo tipo di errori:

- la somma tra due valori positivi pare fornire un risultato negativo o comunque il segno del risultato è opposto a quello che sarebbe lecito attendersi;
- esaminando i valori coinvolti nell'operazione ci si accorge che il risultato eccede i limiti dei valori rappresentabili prima ancora di svolgere l'operazione;
- si riscontra una situazione particolare relativamente al riporto sul bit del segno e sul bit ulteriore a sinistra di quello del segno. Chiariamo meglio quest'ultimo punto:
    1. se si ha:
        - a)un riporto sul bit del segno e sul bit a sinistra di esso,
        - b)non si ha riporto su nessuno dei due bit,

       allora il risultato è corretto (scartando l'eventuale bit eccedente a sinistra dal risultato);

    2. se si ha un riporto sul bit del segno e non sul bit a sinistra di esso o viceversa, allora il risultato non è corretto e si ha una situazione di overflow.

       Esempio del caso1:
       -50+90=+40 in binario diventa 11001110+01011010 cioè 00101000 con rip. 1 eliminato senza provocare una segnalazione di errore (situaz. a)
       55-99= - 44 in binario diventa 00110111+10011101=11010100 (situaz. b)
       Esempio del caso2:
       55+99 cioè 00110111+01100011 diventa 10011010 con overflow (analogam -55-99)
       Articolo sviluppato tenendo anche conto di quanto scritto quì :

http://scuola.linux.it/docs/altre_scuole/planck/rappr-dati/rappr-dati4.html