BIT e Computer

Cosa è un bit? In informatica è l'acronimo di Binary digit

un bit è una cifra binaria, (in inglese "**b**inary dig**it**") ovvero uno dei due simboli del sistema numerico binario classicamente chiamati *zero* (0) e *uno* (1).
(si può usare la parola bit anche per l'unità di misura dell'informazione (dall'inglese "**b**inary un**it**"), definita come la quantità minima di informazione che serve a discernere tra due possibili alternative equiprobabili)

Rappresentazioni di numeri binari

I numeri binari, in campo informatico, non sono utilizzati esclusivamente per memorizzare numeri interi positivi ma, mediante alcune convenzioni, è possibile scrivere numeri binari con segno e parte decimale senza introdurre nuovi caratteri (come la virgola e il segno meno, non memorizzabili su di un bit).

Rappresentazione in modulo e segno

Questo è il modo più semplice per rappresentare e distinguere numeri positivi e negativi: al numero binario vero e proprio viene anteposto un bit che, per convenzione, assume il valore 0 se il numero è positivo ed assume il valore 1 se il numero è negativo.
Il grande difetto di questa rappresentazione è quello di avere due modi per scrivere il numero 0:
00000000 e 10000000 significano infatti +0 e -0.

Rappresentazione in complemento a 2

Questo metodo di rappresentazione ha notevoli vantaggi, soprattutto per effettuare somme e differenze: in pratica ai numeri viene anteposto un bit di valore zero; se poi il numero è negativo è necessario convertirlo in complemento a 2.

Per fare il complemento a 2 è sufficiente leggere il numero da destra verso sinistra e invertire tutte le cifre a partire dal primo bit pari a 1 (escluso). Per fare un esempio:

$$(-12)_{10} = (-01100)_2 = (10100)_{CA2}$$

Come è possibile notare seguendo questo metodo il primo bit diventa automaticamente il bit del segno (come per il metodo precedente). Viene però risolto il problema dell'ambiguità dello 0 (in complemento a 2 il numero 00000 e 10000 hanno significati diversi: 00000 vale il numero 0 mentre 10000 in complemento a 2 diventa 10000 e quindi vale 16) e vengono enormemente facilitate le operazioni di somma e differenza, che si riducono alla sola operazione di somma. Vediamo qualche esempio di complemento a 2 e di somma con numeri binari.

Per capire meglio vediamo un altro metodo per trovare il complemento a due di un numero binario:
si invertono, o negano, i singoli bit. Si aggiunge infine 1 al valore del numero trovato con questa operazione.
Esempio
Rappresentazione del numero -5 con 8 bit in complemento a 2.
Si scrive innanzitutto la rappresentazione binaria del numero 5:
$(5)_{10} = (0000\ 0101)_2$
Si invertono i bit in modo che 0 diventi 1, e 1 diventi 0:
1111 1010
A questo punto abbiamo ottenuto il "complemento a uno" del numero 5. Per ottenere il complemento a due aggiungiamo 1:
1111 1010+
1=
1111 1011 complemento a 2 di 5 che rappresenta il numero negativo -5

Esempio: Sommare i numeri $(5)_{10}$ =$(0000\ 0101)_2$ e $(-5)_{10} = (1111\ 1011)_2$.
0000 0101 +
1111 1011 =
1 0000 0000 .
Si ignora l'overflow poiché abbiamo scelto una rappresentazione a 8 bit e quindi il risultato è 0 = 0000 0000.

Esempio: Sommare i numeri 5 = 0000 0101 e 1 = 0000 0001.
0000 0101 +
0000 0001 =

0000 0110 .
Il risultato è 6=0000 0110.

Esempio: Sommare i numeri 5 e -1.
Innanzitutto si scrive il complemento a 2 del numero binario che rappresenta il numero -1:
0000 0001è il numero 1 rappresentato con 8 bit
si invertono i bit: 1111 1110;
si somma 1: 1111 1111 = -1complemento a 2
Si fa la somma:
0000 0101 +
1111 1111 =
---
1 0000 0100
si ignora l'overflow e il risultato è $(4)_{10}=(0000\ 0100)_2$

Osservazione
Il complemento a due di un numero negativo ne restituisce il modulo (valore assoluto).
Invertendo i bit della rappresentazione del numero -5 (sopra) otteniamo: 0000 0100;
Aggiungendo 1 otteniamo: 0000 0101, che è appunto la rappresentazione del numero +5 in forma binaria.
Proviamo col numero -1:
1111 1111 = -1 (complemento a 2); si invertono i bit: 0000 0000; si somma 1: 0000 0001=$(1)_{10}$.

Osservazione
Si noti che il complemento a due dello zero è zero stesso: invertendone la rappresentazione si ottiene un byte di 8 bit pari a 1, e aggiungendo 1 si ritorna a tutti 0 (l'overflow viene ignorato).
0000 0000 [inversione]→ 1111 1111
[somma 1]→
1111 1111 +
0000 0001 =
1 0000 0000 [si ignora l'overflow]→
0000 0000

un altro esempio di sottrazione che diventa somma:

$(5)_{10} - (10)_{10} = (5)_{10} + (-10)_{10} = (0000\ 0101)_2 - (00001010)_2 = (0000\ 0101)_{CA2} + (1111\ 0110)_{CA2} = (1111\ 1011)_{CA2} = -(0000\ 0101)_2 = (-5)_{10}$

Il complemento a 2 consente di operare le operazioni di addizione e sottrazione in modo più efficiente rispetto alle altre rappresentazioni, motivo per cui è quella comunemente utilizzata nei calcolatori elettronici per i numeri interi.

Rappresentazione a virgola fissa

Dato che in un bit non è rappresentabile la virgola il metodo più semplice per rappresentare numeri frazionari è quello di scegliere arbitrariamente la posizione della virgola (ad es. se si sceglie di usare 4 bit per la parte intera e 4 per la parte frazionaria: 10100101 significa 1010,0101).

Rappresentazione in virgola mobile P754

Esistono innumerevoli modi per rappresentare numeri in virgola mobile ma il sistema più utilizzato è lo standard IEEE P754; questo metodo comporta l'utilizzo della notazione scientifica, in cui ogni numero è identificato dal segno, da una mantissa (1,xxxxx) e dall'esponente (n^{yyyyy}). La procedura standard per la conversione da numero decimale a numero binario P754 è la seguente:

1. Prima di tutto il numero, in valore assoluto, va convertito in binario.
2. Il numero va poi diviso (o moltiplicato) per 2 fino a ottenere una forma del tipo 1,xxxxxx.
3. Di questo numero viene eliminato l'1 iniziale (per risparmiare memoria)
4. Il numero di volte per cui il numero è stato diviso (o moltiplicato) per 2 rappresenta l'esponente: questo valore (decimale) va espresso in eccesso 127, ovvero è necessario sommare 127 e convertire il numero risultante in

binario. Nel caso di rappresentazione a precisione doppia il valore dell'esponente viene espresso in eccesso 1023.

A questo punto abbiamo raccolto tutti i dati necessari per memorizzare il numero: in base al numero di bit che abbiamo a disposizione possiamo utilizzare tre formati: il formato a precisione singola (32 bit), il formato a precisione doppia (64 bit) e il formato a precisione quadrupla (128 bit).

1. Nel primo caso possiamo scrivere il valore utilizzando 1 bit per il segno, 8 bit per l'esponente e 23 bit per la mantissa.
2. Nel secondo caso servirà 1 bit per il segno, 11 bit per l'esponente e 52 per la mantissa.
3. Nel terzo caso servirà 1 bit per il segno, 15 bit per l'esponente e 112 per la mantissa.

Per esempio, convertiamo il valore$(-14{,}3125)_{10}$ in binario P754 single:

1. Convertiamo prima di tutto il numero: $(14)_{10} = (1110)_2$ per la parte intera e $(0{,}3125)_{10} = (0{,}0101)_2$ per la parte decimale. Quindi il numero definitivo è $(1110{,}0101)_2$ (segno escluso).
2. Dividiamo poi il numero per 2 per ottenere la seguente notazione:$(1110{,}0101)_2 = (1{,}1100101)_2 * 2^3$
3. La mantissa diventa, quindi: 1100101 e l'esponente espresso in bae 10 è 3.
4. Per esprimere l'esponente in eccesso 127, infine: $127 + 3 = (130)_{10} = (10000010)_2$

Il numero, alla fine, sarà espresso nel formato:

1 10000010 11001010000000000000000

Segno esponente mantissa