Informazione dell'Informatica

(Ovvero: Informatica per Comuni Mortali)

11 dicembre 2025

1 Premessa

Diamo per scontato che in Informatica, le informazioni sono rappresentati in bit. Nel caso non fosse comprensibile di tale concetto, è consigliato riccorrere a LLM per l'aiuto.

2 Sistemi di Numerazione

I Numeri disponibili in sè sono da 0 a 9, una cifra non ha un valore fisso; Questo dipende direttamente da quale positizione si trova.

La formula generale

Un numero N in base B qualsiasi è la somma delle sue cifre(c) moltiplicate per la base eleveta alla posizione(i)

$$N = \sum_i C_i \cdot B^i$$

Per noi Comuni Mortali

La formula sopra è per chi vuole fare il fenomeno ad essere rigorosi. Noi guardiamo l'esempio pratico:
In base 10: $3668 = 3 \cdot 10^3 + 6 \cdot 10^2 + 6 \cdot 10^1 + 8 \cdot 10^0$

2.1 Da Decimale a Qualsiasi Base: La Divisione

Non ci complichiamo la vita. Usiamo la divisione ripetuta.

- Prendiamo il numero N e dividiamo per la base B
- Il **RESTO** diventa la cifra (leggere al contrario, dal basso verso l'alto!).
- Ripetiamo l'operazione con il quoziente della divisione

Esempio: Conversione 121_{10} in $binario_2$

Numero	Diviso 2	Resto (Bit)
121	60	**1** (↑ LSB)
60	30	**0**
30	15	**0**
15	7	**1**
7	3	**1**
3	1	**1**
1	0	**1** (↑ MSB)

Risultato: $121_{10} = 1111001_2$

- *LSB(Least Significant Bit) = Bit meno significativo*
- *MSB (Most Significant Bit) = Bit più significativo*

3 Numeri Negativi

La conversione normale ci permette di avere il valore assoluto(modulo), in informatica significa binario puro(o unsigned, senza segno).
Ma se volessimo avere con il segno negativo? Ecco che entrano in gioco i seguenti metodi.

3.1 Modulo e Segno (M&S)

La differenza tra "Binario Puro" e "Modulo e Segno" è come li spendiamo i bit a nostra disposizione.
Immaginiamo di avere a disposizione 8 bit

Binario Puro	**Modulo e Segno**
8 bit	**7 bit**
0 a 255$(2^8 - 1)$	**-127 a +127**
No numeri negativi	1 bit per il segno

Usiamo il primo bit per il segno $(0 = +, 1 = -)$.
Esempio:

$$+5 = 0101$$
$$-5 = 1101$$

3.1.1 I problemi del M&S

1. **Il Bug:** Abbiamo lo 0 positivo (0000) e lo 0 negativo (1000).
2. Difficile fare operazioni
   Es. $5 + (-5)$

$$\begin{array}{r} 0101 \\ +1101 \\ \hline \color{red}{1}\,0010 \end{array}$$

   Non c'è spazio oltre i 4 bit, 1 viene tagliato fuori.

$$0010 = 2 \neq 0$$

3.2 Complemento a 1 (C1)

Possiamo utilizzare **Complemento a 1** per risolvere i problemi del **M&S**
La regola è **invertire** tutti i bit di un numero per ottenere il suo negativo (fare semplicemente il *Not Logico*)

Metodo "Pigro"

Semplicemente: **INVERTIRE TUTTI I BIT**. Dove c'è 0 metti 1, dove c'è 1 metti 0.

Operazione con C1 e End-around carry (Riporto circolare)
Esempio: $5 + (-2)$

- $+5 = 0101$
- $-2 = 1101$ (in C1 è l'inverso bit a bit di 0010)

$$\begin{array}{r} 0101 \\ +1101 \\ \hline 1\,0010 \end{array} \Rightarrow \begin{array}{r} 0010 \\ +\quad 1 \\ \hline 0011 \end{array} = +3 \text{ (Torna!)}$$

Conclusione: Se avessimo troncato il bit, il risultato sarebbe stato 0010(+2), errato. Con il riporto circolare otteniamo 0011(+3), corretto.
Nota: Questo metodo è scomodo per l'hardware (richiede un secondo ciclo di somma).

Problema: *End-around carry* (Riporto circolare). Troppo sbattimento hardware.

3.3 Complemento a 2 (C2) - Lo Standard

Ecco la regola d'oro:

$$C_2(N) = C_1(N) + 1$$

Procedura:

1. Fai il "Metodo Pigro" (Inverti tutto).
2. **Aggiungi 1**. [cite: 229-232]

4 IEEE 754: Virgola Mobile

La formula che fa paura, ma che serve:

$$VALORE = (-1)^S \cdot (1, M) \cdot 2^{(E-Bias)}$$

Dove:

- **S (Segno):** 1 bit.

- **E (Esponente):** 8 bit. Bias = 127.
- **M (Mantissa):** 23 bit.

ATTENZIONE: IL MALEDETTO 0.4

Attenzione al numero **0,4**. È UNA BESTIA. In binario è **periodico** (0011 all'infinito). Non potendo scrivere infiniti bit, dobbiamo troncare → Perdita di precisione!