# SISTEMI DI NUMERAZIONE

## ARITMETICA

Cambi di base, esadecimale
e le quattro operazioni in binario

**Prof. A. Perna**

**IllogicoMatematico**

# Indice

# 1 Cosa serve prima di partire

**Cosa devi già sapere**

- **Le potenze**, e in particolare quelle di 2. Servono a memoria come le tabelline:

| $n$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|
| $2^n$ | 1 | 2 | 4 | 8 | 16 | 32 | 64 | 128 | 256 | 512 |

  Ricorda che $b^0 = 1$ per ogni base $b$: è la colonna delle unità.

- **La divisione con resto**: $23 : 5$ dà quoziente 4 e resto 3, cioè $23 = 5 \cdot 4 + 3$, con il resto sempre *minore* del divisore. È il motore di tutte le conversioni.
- **Addizione in colonna con il riporto**: $7 + 5 = 12$, si scrive 2 e si riporta 1. In binario si fa uguale, cambia solo la soglia.
- **Sottrazione con il prestito**: in $52 - 8$ si prende un'unità dalle decine, che vale dieci.
- **Moltiplicazione e divisione in colonna**, quelle lunghe, con i prodotti parziali incolonnati e le cifre che si abbassano.

Se questi cinque punti sono solidi, tutto il capitolo è una riapplicazione: cambia il numero di simboli disponibili, non il metodo.

# 2 Che cos'è una base

Contiamo per dieci perché abbiamo dieci dita. Non c'è nessun altro motivo: il dieci non ha niente di speciale in matematica, ed è solo un'abitudine molto radicata.

Quello che conta davvero è l'idea sotto: la **posizione** di una cifra ne cambia il valore.

**Formula 2.1: Notazione posizionale:** In base $b$ un numero scritto con le cifre $c_n \dots c_2 c_1 c_0$ vale

$$c_n \cdot b^n + \cdots + c_2 \cdot b^2 + c_1 \cdot b^1 + c_0 \cdot b^0$$

Le cifre ammesse vanno da 0 a $b - 1$: sono $b$ **simboli**, e il simbolo «$b$» non esiste. La base si scrive a pedice: $(1011)_2$, $(3407)_{10}$.

**Esempio Svolto 2.1: Il valore posizionale che già conosci:**
**1.** In base 10, $(3407)_{10}$ vuol dire

$$3 \cdot 10^3 + 4 \cdot 10^2 + 0 \cdot 10^1 + 7 \cdot 10^0 = 3000 + 400 + 0 + 7$$

**2.** Nessuno lo scrive mai così, ma è quello che si fa leggendo «tremilaquattrocentosette».

Il 3 vale tremila *perché sta nella quarta posizione*, non per come è fatto.
**3.** Cambiare base vuol dire cambiare quanto vale ciascuna posizione: in base 2 le colonne non valgono $1, 10, 100, 1000$ ma $\mathbf{1}, \mathbf{2}, \mathbf{4}, \mathbf{8}$.

**Formula 2.2: Le basi che si incontrano:**

| Base | Cifre | Dove si usa |
|---|---|---|
| 2 (binario) | $0, 1$ | dentro ogni computer: acceso/spento |
| 8 (ottale) | 0–7 | permessi dei file su Linux |
| 10 (decimale) | 0–9 | la vita di tutti i giorni |
| 16 (esadecimale) | 0–9, $A$–$F$ | colori, memoria, codici |

**Le cifre che non esistono**

$(129)_8$ non è un numero scritto male in piccolo: è **scritto male e basta**. In base 8 le cifre sono $0, 1, 2, 3, 4, 5, 6, 7$, e il simbolo 8 non c'è — come in base 10 non c'è un simbolo unico per «dieci».
Regola: in base $b$ la cifra più grande è $b - 1$.

**$(10)_2$ non è dieci**

$(10)_2$ si legge «uno zero in base due» e vale 2. Leggerlo «dieci» è l'errore che rovina tutto il capitolo: la scrittura è la stessa, il valore no. Quando c'è ambiguità si scrive sempre il pedice.

## 3 Da una base qualunque a base 10

È la conversione più facile: si applica la definizione.

**Metodo 3.1: Da base $b$ a base 10:**

1. Scrivi sotto ogni cifra la potenza della base che le compete, partendo da $b^0$ **a destra**.
2. Moltiplica ogni cifra per la sua potenza.
3. Somma tutto: il risultato è in base 10.

**Esempio Svolto 3.1: Da binario a decimale:**
**1.** Converto $(1011)_2$. Le potenze, da destra: $2^0 = 1, 2^1 = 2, 2^2 = 4, 2^3 = 8$.
**2.** Moltiplico e sommo:

$$1 \cdot 8 + 0 \cdot 4 + 1 \cdot 2 + 1 \cdot 1 = 8 + 0 + 2 + 1 = \mathbf{11}$$

**3. Scorciatoia**: in binario le cifre sono solo 0 e 1, quindi basta **sommare i valori delle colonne dove c'è un 1**. Qui: $8 + 2 + 1 = 11$.
**4.** Un caso più lungo: $(11010110)_2$. Colonne con l'uno: $128, 64, 16, 4, 2$.

$$128 + 64 + 16 + 4 + 2 = \mathbf{214}$$

**Esempio Svolto 3.2: Da base 4 e da base 16:**
**1.** $(2301)_4$: potenze di 4, cioè $64, 16, 4, 1$.

$$2 \cdot 64 + 3 \cdot 16 + 0 \cdot 4 + 1 \cdot 1 = 128 + 48 + 0 + 1 = \mathbf{177}$$

**2.** $(3A7)_{16}$: le potenze sono $256, 16, 1$, e $A$ vale 10.

$$3 \cdot 256 + 10 \cdot 16 + 7 \cdot 1 = 768 + 160 + 7 = \mathbf{935}$$

**3. Controllo grossolano**, da fare sempre: la cifra più a sinistra è 3 e la sua colonna vale 256, quindi il numero sta fra 768 e 1024. 935 ci sta: plausibile.

## 4 Da base 10 a base 2

Nell'altro verso serve un metodo, ed è sempre lo stesso: **dividere per la base e tenere i resti**.

**Metodo 4.1: Divisioni successive:**

1. Dividi il numero per 2 e scrivi il **resto** (0 o 1).
2. Dividi per 2 il quoziente ottenuto, e scrivi il resto.
3. Vai avanti finché il quoziente diventa 0.
4. Leggi i resti **dal basso verso l'alto**: quello è il numero in binario.

**Esempio Svolto 4.1: Da decimale a binario:**

**1.** Converto 37.

$$\begin{aligned} 37 : 2 &= 18 \quad \text{resto} \quad 1 \\ 18 : 2 &= 9 \quad \text{resto} \quad 0 \\ 9 : 2 &= 4 \quad \text{resto} \quad 1 \\ 4 : 2 &= 2 \quad \text{resto} \quad 0 \\ 2 : 2 &= 1 \quad \text{resto} \quad 0 \\ 1 : 2 &= 0 \quad \text{resto} \quad 1 \end{aligned}$$

**2.** I resti dal basso: $1, 0, 0, 1, 0, 1$.

$$37 = (\mathbf{100101})_{\mathbf{2}}$$

**3. Controllo** riconvertendo: $32 + 4 + 1 = 37$. Torna.

**I resti si leggono dal basso**

Scrivere i resti nell'ordine in cui escono dà $(101001)_2$, che vale 41: un altro numero. L'ultimo resto è la cifra **più significativa**, perché viene dalla divisione più «grande».
Il modo per non sbagliare mai: dopo aver scritto il risultato, riconverti. Costa dieci secondi e chiude la questione.

**Perché funziona**

Dividere per 2 e prendere il resto vuol dire chiedersi «questo numero è pari o dispari?», cioè leggere l'ultima cifra binaria. Poi si butta via quella cifra (il quoziente è il numero senza l'ultima cifra) e si ricomincia. È lo stesso motivo per cui in base 10, dividendo per 10, il resto è l'ultima cifra: $347 : 10 = 34$ con resto 7.

**Esempio Svolto 4.2: Un secondo caso, più grande:**
Converto 200. I resti, nell'ordine in cui escono, sono $0, 0, 0, 1, 0, 0, 1, 1$ (otto divisioni). Letti dal basso:

$$200 = (\mathbf{11001000})_{\mathbf{2}}$$

**Controllo**: $128 + 64 + 8 = 200$. Torna.
**Scorciatoia per il binario**: si può anche procedere per sottrazioni, con la tavola delle potenze. La più grande potenza di 2 che sta in 200 è 128; resta 72. Poi 64: resta 8. Poi 8: resta 0. Le colonne usate sono 128, 64, 8, e il numero è 11001000. Stesso risultato, spesso più veloce.

## 5 Da base 10 a una base qualunque

Il metodo non cambia: si divide per la base, qualunque essa sia.

**Esempio Svolto 5.1: Lo stesso numero in tre basi:**
**1. In base 8.** Divido 200 per 8:

$$\begin{aligned} 200 : 8 &= 25 \quad \text{resto} \quad 0 \\ 25 : 8 &= \ 3 \quad \text{resto} \quad 1 \\ 3 : 8 &= \ 0 \quad \text{resto} \quad 3 \end{aligned}$$

Dal basso: $200 = (\mathbf{310})_{\mathbf{8}}$. Controllo: $3 \cdot 64 + 1 \cdot 8 + 0 = 200$.
**2. In base 16.** $200 : 16 = 12$ con resto 8; $12 : 16 = 0$ con resto 12. Il resto 12 si scrive $C$.

$$200 = (\mathbf{C8})_{\mathbf{16}}$$

Controllo: $12 \cdot 16 + 8 = 192 + 8 = 200$.
**3. In base 5.** $200 : 5 = 40$ resto 0; $40 : 5 = 8$ resto 0; $8 : 5 = 1$ resto 3; $1 : 5 = 0$ resto 1.

$$200 = (\mathbf{1300})_{\mathbf{5}}$$

Controllo: $125 + 3 \cdot 25 = 125 + 75 = 200$.

**Il resto va scritto come cifra della base d'arrivo**

In base 16 un resto di 12 non si scrive «12»: si scrive $C$. Se lo lasci in cifre decimali, $(128)_{16}$ e $(C8)_{16}$ diventano indistinguibili — e il primo vale 296, il secondo 200.

**Metodo 5.1: Da una base all'altra, in generale:** Per passare da base $b$ a base $d$ quando nessuna delle due è 10:

1. converti da $b$ a 10 (somma dei valori posizionali);
2. converti da 10 a $d$ (divisioni successive).

La base 10 fa da ponte perché è quella in cui sappiamo fare i conti a mente. Con basi che sono potenze l'una dell'altra c'è una strada più corta: è il prossimo paragrafo.

## 6 Cambi rapidi fra basi potenza di 2

Fra base 2 e le basi 4, 8, 16 non serve passare dalla base 10: basta **raggruppare le cifre**. È il trucco che rende comodo l'esadecimale.

**Formula 6.1: Quante cifre per gruppo:** Se la base d'arrivo è $2^k$, ogni sua cifra corrisponde a $k$ **cifre binarie**:

| Base | $k$ | gruppi |
|---|---|---|
| $4 = 2^2$ | 2 | 2 bit per cifra |
| $8 = 2^3$ | 3 | 3 bit per cifra |
| $16 = 2^4$ | 4 | 4 bit per cifra |

I gruppi si contano **da destra**; l'ultimo, a sinistra, si completa con zeri.

**Esempio Svolto 6.1: Da binario a ottale:**

**1.** Converto $(110101110)_2$ in base 8: gruppi da 3, da destra.

$$\underbrace{110}_{6}\ \underbrace{101}_{5}\ \underbrace{110}_{6}$$

**2.** Ogni gruppo si legge come un numeretto binario di tre cifre: $110 = 6$, $101 = 5$, $110 = 6$.

$$(110101110)_2 = (\mathbf{656})_8$$

**3. Controllo** passando dalla base 10: il binario vale 430, e $6 \cdot 64 + 5 \cdot 8 + 6 = 384 + 40 + 6 = 430$. Torna.

**Esempio Svolto 6.2: Da binario a esadecimale, con lo zero davanti:**

**1.** Converto lo stesso $(110101110)_2$ in base 16: gruppi da 4, da destra.

$$\underbrace{0001}_{1}\ \underbrace{1010}_{A}\ \underbrace{1110}_{E}$$

**2.** Il gruppo di sinistra aveva una cifra sola: si completa con zeri **a sinistra**, che non cambiano il valore.

$$(110101110)_2 = (\mathbf{1AE})_{\mathbf{16}}$$

**3. Controllo**: $1 \cdot 256 + 10 \cdot 16 + 14 = 256 + 160 + 14 = 430$. Come sopra: torna.

**Si raggruppa da destra, si completa a sinistra**

Raggruppando da sinistra si ottengono cifre giuste ma *nel posto sbagliato*, e il numero cambia. E gli zeri di completamento vanno **davanti**: 0001 vale 1, mentre 1000 vale 8.
Il motivo è che il valore di una cifra dipende dalla distanza dall'unità, che sta a destra.

**Esempio Svolto 6.3: Nel verso opposto:**
Da $(2F)_{16}$ a binario: ogni cifra diventa 4 bit.

$$2 \to 0010 \qquad F \to 1111$$

$$(2F)_{16} = (00101111)_2 = (\mathbf{101111})_\mathbf{2}$$

Gli zeri iniziali si possono togliere, come in base 10 non si scrive 0047.
**Controllo**: $(2F)_{16} = 2 \cdot 16 + 15 = 47$, e $32 + 8 + 4 + 2 + 1 = 47$. Torna.

**Perché fra 8 e 16 non funziona direttamente**

16 non è una potenza di 8, quindi non c'è un numero intero di cifre ottali per ogni cifra esadecimale. Si passa **dal binario**: da ottale a binario (3 bit per cifra), poi si riraggruppa a 4 per l'esadecimale. Il binario fa da ponte fra tutte le basi potenza di due.

## 7 Il sistema esadecimale

Sedici cifre: le dieci solite più sei lettere. Serve perché **un byte sta in due cifre esadecimali**, e leggere $FF$ è più comodo che leggere 11111111.

**Formula 7.1: Le cifre esadecimali:**

| **hex** | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **dec** | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |

Si scrivono **maiuscole**: $A = 10$, e si va avanti in ordine fino a $F = 15$. Ogni cifra vale esattamente quattro bit.

**Esempio Svolto 7.1: Avanti e indietro con l'esadecimale:**
**1. Verso il decimale.** $(2F)_{16} = 2 \cdot 16 + 15 = \mathbf{47}$.
**2. Verso l'esadecimale.** $47 : 16 = 2$ con resto 15, e $15 = F$; poi $2 : 16 = 0$ con resto 2. Dal basso: $(\mathbf{2F})_{\mathbf{16}}$.
**3. Un byte intero.** $(FF)_{16} = 15 \cdot 16 + 15 = 255$: è il numero più grande che sta in otto bit, e infatti $(11111111)_2 = 255$.

**Dove lo incontri davvero**

Il colore `#1E90FF` sono tre byte: $1E$ di rosso (30), 90 di verde (144), $FF$ di blu (255). Nei linguaggi di programmazione l'esadecimale si scrive col prefisso `0x` (`0xFF`) e il binario con `0b` (`0b1011`): sono convenzioni di scrittura: il pedice resta il modo di indicare la base in matematica.

**Le lettere sono cifre, non incognite**

In $(3A7)_{16}$ la $A$ non è una lettera da sostituire: **è** la cifra dieci. E $B$ vale 11, non 8 né 2. Chi confonde $B$ con 8 sbaglia di tre unità in ogni colonna in cui compare.

## 8 Addizione binaria

Da qui in avanti il capitolo è tutto sulle operazioni. Si lavora in colonna come in base 10, con una tavola molto più corta da ricordare.

**Formula 8.1: La tavola dell'addizione:**

$$0+0=0$$
$$0+1=1$$
$$1+0=1$$
$$1+1=10 \quad \text{(scrivo 0, riporto 1)}$$
$$1+1+1=11 \quad \text{(scrivo 1, riporto 1)}$$

L'ultima riga serve quando nella colonna arriva anche un riporto.

**Metodo 8.1: Sommare in binario:**

1. Incolonna i due numeri a destra, come in base 10.
2. Somma colonna per colonna, partendo da destra.
3. Quando la somma è 2 scrivi 0 e riporta 1; quando è 3 scrivi 1 e riporta 1.
4. Il riporto finale, se c'è, diventa una cifra in più a sinistra.
5. **Controlla** convertendo tutto in base 10.

**Esempio Svolto 8.1: Un'addizione con riporto a catena:**
**1.** Calcolo $(1011)_2 + (110)_2$.

$$\begin{array}{r} \text{riporti } 1\ 1\ 1 \\ 1\,0\,1\,1 \\ +\ \ 0\,1\,1\,0 \\ \hline 1\,0\,0\,0\,1 \end{array}$$

**2. Colonna per colonna, da destra.** Unità: $1+0=1$. Due: $1+1=10$, scrivo 0 riporto 1. Quattro: $0+1+1=10$, scrivo 0 riporto 1. Otto: $1+0+1=10$, scrivo 0 riporto 1. Il riporto finale diventa la cifra davanti.
**3.** Risultato: $(\mathbf{10001})_2$.
**4. Controllo in base 10**: $11+6=17$, e $(10001)_2 = 16+1=17$. Torna.

**$1+1$ fa 10, non 2**

Il simbolo 2 in binario non esiste. Scriverlo in una colonna è come scrivere «dieci» in una colonna in base 10: quello che si fa è **scrivere zero e riportare uno**.
E attenzione alla catena: se il riporto incontra un'altra coppia di uni, si propaga. In $(1111)_2 + (1)_2$ attraversa tutto il numero e il risultato è $(10000)_2$ — esattamente come $999 + 1$ in base 10.

## 9 Sottrazione binaria

Stesso schema, con il prestito. L'unica riga da tenere a mente è $10 - 1 = 1$: due meno uno.

**Formula 9.1: La tavola della sottrazione:**

$$\begin{aligned} 0 - 0 &= 0 \\ 1 - 0 &= 1 \\ 1 - 1 &= 0 \\ 0 - 1 &= 1 \quad \text{con prestito dalla colonna a sinistra} \end{aligned}$$

Il prestito vale **due** (non dieci): la colonna a sinistra vale il doppio di quella corrente.

**Esempio Svolto 9.1: Una sottrazione con prestito:**
**1.** Calcolo $(1010)_2 - (11)_2$.

$$\begin{array}{r} 1\,0\,1\,0 \\ -\,0\,0\,1\,1 \\ \hline 0\,1\,1\,1 \end{array}$$

**2. Unità**: $0 - 1$ non si può, quindi chiedo in prestito alla colonna dei due. Quella ha 1: lo presto, e nell'unità diventa 10 (cioè due). Ora $10 - 1 = 1$: scrivo 1.
**3. Colonna dei due**: aveva 1 ma l'ha prestato, quindi vale 0. Devo togliere 1: non si può, e chiedo alla colonna dei quattro. Quella ha 0, e chiede a sua volta agli otto. Il prestito attraversa due colonne — come in $1000 - 1$ in base 10.
**4.** Risultato: $(\mathbf{111})_2$.
**5. Controllo**: $10 - 3 = 7$, e $(111)_2 = 4 + 2 + 1 = 7$. Torna.

**Il prestito vale due**

Chi arriva dalla base 10 scrive «0 diventa 10» e poi calcola $10 - 1 = 9$. In binario 10 è **due**, quindi $10 - 1 = 1$.
Se il risultato di una sottrazione binaria contiene una cifra diversa da 0 o 1, è sicuramente sbagliato.

**Se il secondo numero è più grande**

Con i naturali la sottrazione $(11)_2 - (1010)_2$ non si può fare, come $3 - 10$ non si può nei naturali. Per i numeri negativi si usa il complemento a due (par. 12), che è anche il modo in cui il computer fa *tutte* le sottrazioni.

## 10 Moltiplicazione binaria

È la più facile delle quattro, perché la tavola pitagorica del binario ha quattro righe e nessuna da imparare a memoria.

**Formula 10.1: La tavola della moltiplicazione:**

$$0 \cdot 0 = 0 \qquad 0 \cdot 1 = 0 \qquad 1 \cdot 0 = 0 \qquad 1 \cdot 1 = 1$$

Ogni prodotto parziale è quindi **o tutto zeri, o una copia del primo numero**, incolonnata sotto la cifra che si sta usando.

**Metodo 10.1: Moltiplicare in binario:**

1. Per ogni cifra del secondo numero, da destra, scrivi un prodotto parziale: **zero** se la cifra è 0, una **copia del primo numero** se la cifra è 1.
2. Sposta ogni prodotto parziale di una posizione a sinistra rispetto al precedente, come in base 10.
3. Somma i prodotti parziali con le regole dell'addizione binaria.

**Esempio Svolto 10.1: Una moltiplicazione:**
**1.** Calcolo $(1101)_2 \times (101)_2$.

$$\begin{array}{rr} & 1101 \\ \times & 101 \\ \hline & 1101 \\ & 0000 \\ 1101 & \\ \hline 1000001 & \end{array}$$

**2.** La cifra delle unità del moltiplicatore è 1: copio 1101. La successiva è 0: scrivo zeri, spostati di uno. La terza è 1: copio 1101, spostato di due.
**3.** Sommo i tre prodotti parziali: $(\mathbf{1000001})_2$.
**4. Controllo**: $13 \times 5 = 65$, e $(1000001)_2 = 64 + 1 = 65$. Torna.

**Moltiplicare per una potenza di due**

Moltiplicare per $(10)_2 = 2$ vuol dire **aggiungere uno zero a destra**, esattamente come in base 10 moltiplicare per 10. Per 4 due zeri, per 8 tre.

$$(1101)_2 \times (100)_2 = (110100)_2 \qquad 13 \times 4 = 52$$

È il motivo per cui nei computer moltiplicare per potenze di due è l'operazione più veloce che esista: si chiama *scorrimento*.

## 11 Divisione binaria

È la divisione lunga di sempre, ed è più semplice del solito: a ogni passo la cifra del quoziente può essere solo 0 o 1, quindi non c'è niente da «indovinare». La domanda è una sola: *il divisore ci sta o no?*

**Metodo 11.1: Dividere in binario:**

1. Prendi da sinistra le prime cifre del dividendo, quante bastano perché il numero formato sia $\geq$ divisore.
2. Scrivi 1 nel quoziente e **sottrai** il divisore.
3. Abbassa la cifra successiva.
4. Se quello che hai è minore del divisore, scrivi 0 nel quoziente e abbassa un'altra cifra; se no scrivi 1 e sottrai.
5. Finite le cifre, quello che resta è il **resto**.

**Esempio Svolto 11.1: Una divisione con resto:**
**1.** Calcolo $(110110)_2 : (101)_2$.
**2.** Prime tre cifre: $110 \geq 101$, quindi il quoziente comincia con 1 e sottraggo: $110 - 101 = 1$.
**3.** Abbasso una cifra: ho 11, minore di 101. Scrivo 0 nel quoziente e abbasso ancora: 111. Ora $111 \geq 101$: scrivo 1 e sottraggo, $111 - 101 = 10$.
**4.** Abbasso l'ultima cifra: 100, minore di 101. Scrivo 0 nel quoziente, e non ci sono altre cifre da abbassare.
**5.** Quoziente $\mathbf{(1010)_2}$, resto $\mathbf{(100)_2}$.
**6. Controllo**: $54 : 5 = 10$ con resto 4; e $(1010)_2 = 10$, $(100)_2 = 4$. La prova della divisione: $5 \cdot 10 + 4 = 54$. Torna.

**Il resto deve essere minore del divisore**

Se alla fine il resto è $\geq$ divisore, da qualche parte hai scritto 0 nel quoziente dove ci stava un 1. È lo stesso controllo della divisione in base 10, ed è gratis: guarda il resto e confrontalo col divisore.

## 12 Complemento a due

Nel computer non esiste il segno meno: c'è solo una fila di bit. I numeri negativi si rappresentano così, e il vantaggio è grosso: **la sottrazione diventa un'addizione**, e serve un circuito solo.

**Formula 12.1: Complemento a due su $n$ bit:** Per rappresentare $-x$ su $n$ bit:

$$\text{complemento a due di } x = 2^n - x$$

In pratica, senza fare la sottrazione: si **invertono tutti i bit** di $x$ (0 diventa 1 e viceversa) e si **aggiunge 1**.
Il **primo bit a sinistra è il segno**: 0 positivo, 1 negativo. Su 8 bit si rappresentano i numeri da $-128$ a $+127$.

**Esempio Svolto 12.1: Scrivere un numero negativo:**
**1.** Rappresento $-37$ su 8 bit.
**2. Parto dal positivo**: $37 = (00100101)_2$, scritto con tutti e otto i bit.
**3. Inverto tutti i bit**: 11011010.
**4. Aggiungo 1**: $11011010 + 1 = \mathbf{(11011011)_2}$.
**5. Controllo con la formula**: $2^8 - 37 = 256 - 37 = 219$, e $(11011011)_2 = 128 + 64 + 16 + 8 + 2 + 1 = 219$. Torna.
**6.** Il primo bit è 1: è un numero negativo, come deve essere.

**Esempio Svolto 12.2: Una sottrazione fatta come somma:**
**1.** Calcolo $50 - 37$ su 8 bit, con il complemento a due.
**2.** $50 = (00110010)_2$ e $-37 = (11011011)_2$ (esempio precedente).
**3. Sommo** in binario:

$$00110010 + 11011011 = 1\ 00001101$$

**4. Il nono bit si butta via**: si lavora su 8 bit, e quel riporto esce dalla fila. Resta $(00001101)_2 = \mathbf{13}$.
**5. Controllo**: $50 - 37 = 13$. Torna.

**Il primo bit non è una cifra come le altre**

$(10000000)_2$ su 8 bit *in complemento a due* non vale 128: vale $-128$. Prima di leggere un numero binario bisogna sapere se è **senza segno** (tutti i bit contano) o **in complemento a due** (il primo bit è il segno). Sono due convenzioni diverse sullo stesso oggetto, e il numero di bit va sempre dichiarato.

**Perché proprio «invertire e aggiungere uno»**

Invertire tutti i bit di $x$ su $n$ bit dà $(2^n - 1) - x$, perché $x$ e il suo inverso sommati danno tutti uni, cioè $2^n - 1$. Aggiungendo 1 si ottiene $2^n - x$, che è la definizione. Non è una ricetta magica: è la stessa formula, scritta in modo che un circuito possa eseguirla.

## 13 La verifica, e come si scrive il risultato

Questo capitolo ha un vantaggio raro: **ogni risultato si controlla da soli, sempre**, e il controllo è meccanico.

**Metodo 13.1: Il controllo, qualunque sia l'esercizio:**

1. Converti in **base 10** tutto quello che compare: i dati e il tuo risultato.
2. Rifai l'operazione in base 10, dove sai già farla.
3. I due numeri devono coincidere.
4. Nelle divisioni aggiungi la prova: divisore $\times$ quoziente $+$ resto $=$ dividendo.

**Esempio Svolto 13.1: Il controllo che smaschera:**
Qualcuno scrive $(1011)_2 + (11)_2 = (1120)_2$. Due cose non vanno: nel risultato compare un 2, che in binario non esiste, e il conto in base 10 dà $11 + 3 = 14$, mentre $(1120)_2$ non è nemmeno un numero binario. Il risultato giusto è $(1110)_2$, che vale 14.
La cifra impossibile è il primo campanello: **in un numero binario ci sono solo 0 e 1**. Vale anche per le altre basi: in base 8 nessuna cifra può essere 8 o 9.

Come si scrive il risultato, in breve: **sempre con il pedice della base**, anche quando sembra ovvio; senza zeri inutili davanti (si scrive $(101)_2$, non $(00101)_2$) tranne quando si lavora su un numero fisso di bit, e allora si scrivono tutti; le cifre esadecimali maiuscole. Nelle divisioni si scrivono **quoziente e resto**, tutti e due, con la loro base.

## Indice analitico