**📌 1. La rappresentazione dell’informazione**

I computer **non capiscono lettere o numeri** come noi. Tutto, **ma proprio tutto**, viene convertito in **bit**, cioè cifre binarie: **0** e **1**.

Un bit è l'unità minima di informazione.  
Un gruppo di 8 bit forma un **byte**.

Con i bit si rappresentano:

* numeri,
* lettere (tramite codici come **ASCII** o **Unicode**),
* colori,
* immagini,
* suoni,
* istruzioni per il processore.

**📌 2. Sistema binario e sistema esadecimale**

🔢 Sistema binario (base 2)

È il sistema usato dal computer. Usa solo due cifre: 0 e 1.

Esempio:

* Decimale 5 = **101** in binario  
  (perché 1×2² + 0×2¹ + 1×2⁰ = 4 + 0 + 1 = 5)

🔡 Sistema esadecimale (base 16)

Usa 16 simboli: da 0 a 9 e da A ad F (dove A=10, B=11, ..., F=15).  
È usato per **scrivere più compatto** il binario.

Esempio:

* Binario: 11110000 = Esadecimale: F0

**📌 3. Rappresentazione degli interi senza segno**

Un numero **senza segno** (unsigned) è sempre **positivo**.

Esempio con 1 byte (8 bit):

* Minimo: 00000000 = 0
* Massimo: 11111111 = 255 (cioè 2⁸ − 1)

Si usa per:

* quantità che non possono essere negative (età, dimensioni, contatori…)

**📌 4. Rappresentazione degli interi con segno**

Qui si usano **alcuni bit per rappresentare il segno**.

Il metodo più usato è il **complemento a 2**:

* Se il primo bit è 0 ➝ numero positivo
* Se il primo bit è 1 ➝ numero negativo

Esempio con 8 bit:

* 00000001 = +1
* 11111111 = −1
* Intervallo: da **−128 a +127**

**📌 5. Rappresentazione dei numeri reali**

I numeri con la virgola si rappresentano con il formato **floating point** (virgola mobile), secondo lo standard **IEEE 754**.

Es. 32 bit (float):

* 1 bit per il **segno**
* 8 bit per l’**esponente**
* 23 bit per la **mantissa** (parte decimale)

Questo consente di rappresentare numeri **molto piccoli o molto grandi**, ma **non tutti con precisione assoluta** (si perde qualcosa dopo un certo numero di cifre).

### Facciamo qualche esempio:

**Esempio: 5,375 (decimale)**

Dividi in:

parte intera: 5 → 101

parte frazionaria: 0,375 → ?

Moltiplicazioni:

0,375 × 2 = 0,75 → 0

0,75 × 2 = 1,5 → 1

0,5 × 2 = 1,0 → 1 Non abbiamo più parte frazionaria ci fermiamo.

Parte frazionaria: 0.011

Risultato totale: 101.011

**Esempio: 0,1 (decimale)**

Attenzione: non è rappresentabile esattamente in binario, come 1/3 in decimale!

0,1 × 2 = 0,2 → 0

0,2 × 2 = 0,4 → 0

0,4 × 2 = 0,8 → 0

0,8 × 2 = 1,6 → 1

0,6 × 2 = 1,2 → 1

0,2 × 2 = 0,4 → 0

… e così via

📌 Risultato binario: 0.000110011001100... (periodico)

**📌 6. La rappresentazione delle immagini**

Un'immagine digitale è composta da **pixel**, ognuno con un **colore**.

Ogni colore è rappresentato da 3 componenti:

* **R**ed
* **G**reen
* **B**lue

Ogni componente è solitamente un **byte (8 bit)** → valori da 0 a 255.

Esempio:

* Bianco = (255, 255, 255)
* Nero = (0, 0, 0)
* Rosso = (255, 0, 0)

Un'immagine Full HD (1920x1080) ha oltre 2 milioni di pixel. Se ogni pixel usa 3 byte, l'immagine “pesa” circa **6 MB non compressa**.

**Conversione da decimale a binario**

Esempio: convertiamo decimale 13 in binario

1. Dividi il numero per 2 e **scrivi il resto**.
2. Prendi il **quoziente** e ripeti finché non ottieni 0.
3. Il numero binario si ottiene **leggendo i resti dal basso verso l’alto**.

✏️ Passaggi

| **Divisione** | **Quoziente** | **Resto** |
| --- | --- | --- |
| 13 ÷ 2 | 6 | 1 |
| 6 ÷ 2 | 3 | 0 |
| 3 ÷ 2 | 1 | 1 |
| 1 ÷ 2 | 0 | 1 |

✅ Risultato: 13 in binario = **1101**

(Leggi i resti **al contrario**: dal basso verso l’alto)

**🔢 Proviamo un altro esempio: 25**

| **Divisione** | **Quoziente** | **Resto** |
| --- | --- | --- |
| 25 ÷ 2 | 12 | 1 |
| 12 ÷ 2 | 6 | 0 |
| 6 ÷ 2 | 3 | 0 |
| 3 ÷ 2 | 1 | 1 |
| 1 ÷ 2 | 0 | 1 |

**Risultato:** 25 = **11001** in binario

Invece per convertire da decimale ad esadecimale si può passare facilmente da binario prendendo 4 bit per volta ed usando la tabella:

### Tabella binario / decimale / esadecimale

Puoi usare anche la **conversione tramite binario**, se lo conosci bene:

* 1 cifra esadecimale = 4 bit binari
* Esempio: 11111111 in binario = F + F = **FF**

**🧠 ESERCIZI – Rappresentazione dell’informazione**

📌 1. Sistemi di numerazione

**Esercizio 1.1**  
Converti i seguenti numeri **decimali** in **binario**:

a) 5

b) 12

c) 255

**Esercizio 1.2**  
Converti i seguenti numeri **binari** in **decimale**:

a) 1101

b) 100000

c) 11111111

**Esercizio 1.3**  
Converti i seguenti numeri binari in **esadecimale**:

a) 10101010

b) 11001100

c) 11110000

📌 2. Interi con e senza segno

**Esercizio 2.1**  
Qual è il valore massimo rappresentabile con:

a) 4 bit senza segno

b) 4 bit con segno (complemento a 2)

**Esercizio 2.2**  
Scrivi in binario (8 bit) i seguenti numeri interi:

a) 25 (senza segno)

b) −25 (con segno, complemento a 2)

📌 3. Numeri reali (floating point semplificato)

**Esercizio 3.1**  
In un formato semplificato a 8 bit (1 bit segno, 3 bit esponente, 4 bit mantissa), rappresenta:

a) +1.5

b) −0.75

📌 4. Immagini digitali

**Esercizio 4.1**  
Un’immagine è larga 100 pixel e alta 50 pixel. Ogni pixel usa 3 byte (RGB).

a) Quanto spazio occupa (in byte)?

b) Quanto spazio in kilobyte (KB)?

**Esercizio 4.2**  
Dati i seguenti valori RGB, indica il colore:

a) (255, 0, 0)

b) (0, 255, 0)

c) (255, 255, 255)

**✅ SOLUZIONI**

**1.1**

a) 5 → 101

b) 12 → 1100

c) 255 → 11111111

**1.2**

a) 1101 → 13

b) 100000 → 32

c) 11111111 → 255

**1.3**

a) 10101010 → AA

b) 11001100 → CC

c) 11110000 → F0

**2.1**

a) 4 bit unsigned: 2⁴ − 1 = **15**

b) 4 bit signed: da −8 a +7

**2.2**

a) 25 → 00011001

b) −25 → complemento a 2 di 25:

* 25 = 00011001
* Inverti = 11100110
* +1 = 11100111  
  → Risultato: **11100111**

**3.1**  
(Salto per ora, ma posso farteli se vuoi approfondire lo standard IEEE 754)

**4.1**

a) 100×50×3 = **15.000 byte**

b) 15.000 ÷ 1024 ≈ **14,65 KB**

**4.2**

a) Rosso

b) Verde

c) Bianco