I.I.S. *"Francesco Alberghetti"* Imola

Liceo Scientifico Scienze Applicate

Codifica dell'informazione

Il sistema binario, esadecimale e cenni di logica booleana

Materiale didattico di Informatica

Prof. Federico Mazzini

Anno scolastico 2025 – 2026

Indice

1 Informazione, dato e rappresentazione

Nel contesto dell'informatica tutto ruota intorno a tre concetti fondamentali:

- **dato**
- **informazione**
- **rappresentazione**

Questi termini vengono spesso usati in modo intuitivo. Risulta quindi utile fissare una distinzione chiara.

1.1 Dato

Per **dato** si intende un elemento grezzo, non ancora interpretato. Un numero scritto su un foglio, una sequenza di simboli, una misura in forma numerica costituiscono dati.

Un esempio:

- La scritta `42` rappresenta un dato. Da sola, non indica se si parla di una temperatura, di un'età, di un punteggio o di un codice.

1.2 Informazione

L'**informazione** nasce quando un dato viene interpretato in un contesto.

Esempi:

- `42` accompagnato dalla parola "anni" viene interpretato come un'età.
- `42 °C` indica una temperatura.
- `42 km` rappresenta una distanza.

Il significato dipende quindi dal **contesto** e dalle **regole di interpretazione** condivise.

1.3 Rappresentazione

La **rappresentazione** riguarda il modo in cui un'informazione viene codificata. La stessa informazione può avere rappresentazioni differenti.

Esempi:

- Il numero quarantadue può essere rappresentato come `42` in decimale, `101010` in binario, `2A` in esadecimale.
- Una nota musicale può essere rappresentata come simbolo sul pentagramma, come frequenza in Hz, come sequenza di campioni digitali.

Nel calcolatore ogni rappresentazione deve essere ricondotta a una sequenza di bit. Questo costituisce il punto di partenza per tutto il resto.

1.4 Perché serve una rappresentazione digitale

I dispositivi elettronici lavorano con segnali elettrici che assumono stati distinti (ad esempio, presenza o assenza di tensione). Risulta naturale sfruttare due soli stati logici e rappresentarli con i simboli 0 e 1.

Tutta l'informatica digitale si basa su questa scelta: due stati ben separati sono più affidabili da distinguere fisicamente rispetto a molti stati intermedi. Da qui nasce la rappresentazione binaria.

2 Bit, byte e quantità di informazione

2.1 Il bit

Il **bit** (*binary digit*) costituisce l'unità minima di informazione trattata da un calcolatore. Può assumere solo due valori possibili:

$$0 \text{ oppure } 1$$

2.2 Il byte e i suoi multipli

Per ragioni pratiche i bit vengono raggruppati in blocchi di 8. Otto bit formano un **byte**. Il byte rappresenta l'unità di misura di base per la memoria e per la capacità di memorizzazione.

Multipli più comuni:

- 1 KB (kilobyte) = 1024 byte
- 1 MB (megabyte) = 1024 KB
- 1 GB (gigabyte) = 1024 MB
- 1 TB (terabyte) = 1024 GB

Perchè proprio 1024?

L'uso di 1024 anziché 1000 è dovuto alla natura binaria dei calcolatori: $1024 = 2^{10}$.

2.3 Numero di stati rappresentabili

Con n bit è possibile rappresentare 2^n configurazioni diverse. Ogni bit può assumere 2 valori, quindi le combinazioni totali si ottengono moltiplicando:

$$2 \cdot 2 \cdot 2 \cdots 2 = 2^n$$

In maniera molto intuitiva:

- con 1 bit $\rightarrow$ 2 stati (0,1)
- con 2 bit $\rightarrow$ 4 stati (00,01,10,11)
- con 3 bit $\rightarrow$ 8 stati

- con 8 bit $\rightarrow$ 256 stati
- con 16 bit $\rightarrow$ 65536 stati

Questa formula è alla base di tutte le scelte di codifica: numero di colori rappresentabili, caratteri possibili, livelli di intensità sonora, ecc.

2.4 Quantità di informazione e scelta del numero di bit

Maggiore è il numero di stati da distinguere, maggiore deve essere il numero di bit utilizzati. Ad esempio:

- per rappresentare i numeri da 0 a 9 non basta usare 3 bit (8 stati), sono necessari almeno 4 bit (16 stati), anche se alcuni rimangono inutilizzati.
- per rappresentare tutti i caratteri di base di una tastiera vengono usati 7 o 8 bit.
- per rappresentare un colore in formato RGB a 24 bit vengono sfruttati 8 bit per ciascuno dei tre canali (rosso, verde, blu).

Perché proprio 1 byte = 8 bit?

Nei primi calcolatori (dal 1950) non esisteva una dimensione "giusta" per il gruppo di bit: furono sperimentate soluzioni con 6, 7, 9 bit e oltre. Il blocco da 8 bit si è però imposto nel tempo come standard perché rappresenta un buon compromesso tra semplicità hardware e capacità informativa.

Con 8 bit si hanno infatti $2^8 = 256$ combinazioni possibili. Questo numero di stati è sufficiente a rappresentare:

- le lettere maiuscole e minuscole dell'alfabeto inglese;
- le cifre da 0 a 9;
- la punteggiatura e vari simboli speciali;
- alcuni codici di controllo (ad esempio invio, lo spazio, ecc.).

Inoltre, 8 è una potenza di 2 ($2^3 = 8$).

3 I sistemi di numerazione posizionale

Per comprendere il sistema binario risulta utile definire il funzionamento del sistema decimale.

3.1 Sistema decimale

Il sistema decimale si basa sulla **base 10**. Ciò significa che:

- sono disponibili 10 **simboli**: da 0 a 9;
- ogni **posizione** rappresenta una **potenza** di 10.

Esempio:

$$472 = 400 + 70 + 2 = 4 \cdot 10^2 + 7 \cdot 10^1 + 2 \cdot 10^0$$

Il valore di una cifra dipende quindi sia dal simbolo, sia dalla posizione.

3.2 Definizione di sistema posizionale

Un sistema di numerazione posizionale è caratterizzato da:

- una base b (numero di simboli disponibili);
- un insieme di cifre da 0 a $b - 1$;
- il valore di ciascuna cifra è dato dalla cifra stessa moltiplicata per una potenza della base.

Un numero generico in base b può essere scritto come:

$$(a_n a_{n-1} \dots a_1 a_0)_b$$

e interpretato come:

$$a_n b^n + a_{n-1} b^{n-1} + \cdots + a_1 b^1 + a_0 b^0$$

3.3 Sistema binario

Il sistema binario utilizza base 2. Le cifre possibili sono solo due:

$$0 \text{ e } 1$$

Ogni posizione rappresenta una potenza di 2:

$$\dots, 2^4, 2^3, 2^2, 2^1, 2^0$$

Esempio di interpretazione:

$$(1011)_2 = 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 = 8 + 0 + 2 + 1 = 11_{10}$$

3.4 Perché il binario è usato nei calcolatori

La scelta della base 2 non è casuale. Due stati logici distinti si prestano a essere rappresentati con facilità mediante livelli di tensione differenti (ad esempio, alto/basso, presenza di corrente elettrica oppure no). La semplicità a livello fisico si traduce in affidabilità e velocità.

4 Conversioni tra decimale e binario

4.1 Conversione da binario a decimale

La conversione da binario a decimale segue direttamente la definizione di sistema posizionale.

Procedura:

1. si scrive il numero binario separando le cifre;
2. si associa a ogni cifra la potenza di 2 corrispondente (partendo da destra con 2^0);
3. si moltiplica ogni cifra (0 o 1) per la propria potenza di 2;
4. si sommano i risultati.

Esempio:

$$\begin{aligned}(11010)_2 &= 1 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0 \\ &= 16 + 8 + 0 + 2 + 0 = 26_{10}\end{aligned}$$

4.2 Conversione da decimale a binario: metodo delle divisioni successive

Il metodo più comune per la conversione da decimale a binario si basa su divisioni successive per 2.

Procedura generale:

1. si divide il numero decimale per 2;
2. si registra il resto (0 o 1);
3. si sostituisce il numero con il quoziente della divisione;
4. si ripete fino a ottenere quoziente 0;
5. si leggono i resti dal basso verso l'alto.

Esempio: conversione di 19 in binario.

$$\begin{aligned}19 : 2 = 9 &\quad \text{e resto } 1 \\ 9 : 2 = 4 &\quad \text{e resto } 1 \\ 4 : 2 = 2 &\quad \text{e resto } 0 \\ 2 : 2 = 1 &\quad \text{e resto } 0 \\ 1 : 2 = 0 &\quad \text{e resto } 1\end{aligned}$$

Lettura dei resti dal basso verso l'alto:

$$19_{10} = (10011)_2$$

4.3 Osservazioni sulle conversioni

Le procedure di conversione non rappresentano solo un esercizio di calcolo: consentono di comprendere come il calcolatore interpreti i numeri e mostrano il legame stretto tra rappresentazione binaria e struttura interna della macchina.

5 Il sistema esadecimale

5.1 Definizione

Il sistema esadecimale utilizza base 16. Ciò significa che vengono impiegati 16 simboli distinti:

- le cifre decimali da 0 a 9;
- le lettere maiuscole da A a F.

Le lettere rappresentano i valori decimali da 10 a 15, come mostrato in tabella:

Simbolo	Valore decimale
A	10
B	11
C	12
D	13
E	14
F	15

Un numero esadecimale viene talvolta scritto con un prefisso (ad esempio `0x2A`) per distinguerlo da un numero decimale.

5.2 Collegamento con il binario

La base 16 risulta particolarmente comoda in ambito informatico per il legame diretto con la base 2:

$$2^4 = 16$$

Questa uguaglianza implica che **ogni gruppo di 4 bit** può essere rappresentato con una **singola cifra esadecimale** e viceversa.

Tabella di corrispondenza tra tutti i blocchi possibili di 4 bit e le cifre esadecimali:

Binario	Esadecimale
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	A
1011	B
1100	C
1101	D
1110	E
1111	F

Questa corrispondenza rende l'esadecimale una sorta di "abbreviazione leggibile" del binario.

5.3 Conversione binario – esadecimale

Grazie alla corrispondenza per gruppi di 4 bit, la conversione risulta diretta e meccanica.

Da binario a esadecimale

Procedura generale:

1. si parte dal numero binario;
2. si raggruppano i bit in blocchi da 4 a partire da destra (se necessario si aggiungono zeri a sinistra nell'ultimo blocco);
3. per ciascun blocco di 4 bit si individua la cifra esadecimale corrispondente nella tabella;
4. si scrivono le cifre esadecimali nello stesso ordine.

Esempio 1 Conversione di $(10110110)_2$ in esadecimale.

$$10110110_2$$

Raggruppamento in blocchi da 4 bit partendo da destra:

$$1011 \quad 0110$$

Da tabella:

$$1011_2 = B_{16}, \quad 0110_2 = 6_{16}$$

Quindi:

$$(10110110)_2 = (B6)_{16}$$

Esempio 2 Conversione di $(1100101)_2$ in esadecimale.

Il numero possiede 7 bit, non un multiplo di 4. Si completa il blocco più a sinistra aggiungendo uno zero:

$$(1100101)_2 = (0110\ 0101)_2$$

Blocchi da 4 bit:

$$0110 \quad 0101$$

Da tabella:

$$0110_2 = 6_{16}, \quad 0101_2 = 5_{16}$$

Quindi:

$$(1100101)_2 = (65)_{16}$$

Da esadecimale a binario

Procedura generale:

1. si considera ciascuna cifra esadecimale;
2. per ogni cifra si sostituisce il corrispondente blocco di 4 bit;
3. si concatenano i blocchi ottenuti.

Esempio 3 Conversione di $(3F)_{16}$ in binario.

- $3_{16} \to 0011_2$
- $F_{16} \to 1111_2$

Quindi:

$$(3F)_{16} = (0011\ 1111)_2 = (00111111)_2$$

Esempio 4 Conversione di $(A2)_{16}$ in binario.

- $A_{16} \to 1010_2$
- $2_{16} \to 0010_2$

Quindi:

$$(A2)_{16} = (1010\ 0010)_2 = (10100010)_2$$

5.4 Perché usare l'esadecimale

La scrittura binaria di numeri lunghi risulta poco leggibile e difficile da controllare a colpo d'occhio. L'esadecimale offre:

- una **forma più compatta**: riduce la lunghezza delle sequenze rispetto al binario;
- una **relazione diretta con il binario**: la conversione avviene per gruppi di 4 bit, senza calcoli complessi;
- una **maggiore leggibilità** per esseri umani in contesti tecnici (programmazione di basso livello, analisi della memoria, debug).

In molti strumenti di sviluppo, valori che la macchina tratta in binario vengono presentati in esadecimale perché risulta un compromesso efficace tra "vicinanza all'hardware" e comprensione umana.

5.5 Perché si usa l'esadecimale

Le sequenze di bit risultano poco leggibili quando diventano lunghe. L'esadecimale permette di riscrivere gruppi di bit in una forma più compatta e comprensibile, mantenendo comunque una corrispondenza esatta con il binario.

In pratica, i bit vengono raggruppati a gruppi di 4 e ogni gruppo viene sostituito con una cifra esadecimale. In questo modo una lunga sequenza di 0 e 1 si trasforma in pochi simboli più facili da interpretare.

Esempio Si consideri il numero binario:

$$(101010)_2$$

Poiché il numero contiene 6 bit, si aggiungono zeri a sinistra fino a ottenere un multiplo di 4:

$$(101010)_2 = (0010\ 1010)_2$$

Si convertono i blocchi separatamente:

$$0010_2 = 2_{16}, \quad 1010_2 = A_{16}$$

Si ottiene quindi:

$$(101010)_2 = (2A)_{16}$$

Questo esempio mostra come una sequenza binaria venga trasformata in una forma più compatta e leggibile. L'esadecimale consente quindi di "tradurre" il linguaggio dei bit in una notazione più adatta alla lettura umana, pur rimanendo rigorosamente legata alla rappresentazione interna del calcolatore.

6 Logica booleana

6.1 Decisioni sì/no

Nella vita quotidiana le decisioni spesso dipendono da condizioni che possono essere considerate vere o false.

Esempi:

- "se piove, allora si usa l'ombrello";
- "se l'età è maggiore o uguale a 18, allora è consentito votare";
- "se la batteria è scarica, allora si ricarica il dispositivo".

Ogni affermazione del tipo "condizione → azione" può essere formalizzata con la logica booleana.

6.2 Variabile booleana

Una **variabile booleana** può assumere solo due valori:

- vero (spesso indicato con 1);
- falso (spesso indicato con 0).

Questo modello è perfettamente compatibile con la rappresentazione mediante bit.

6.3 Operatori logici fondamentali

Gli operatori logici di base sono:

NOT (negazione): inverte il valore di verità.

AND (congiunzione): è vero solo se entrambe le condizioni sono vere.

OR (disgiunzione inclusiva): è vero se almeno una delle condizioni è vera.

6.4 Tabelle di verità

Le tabelle di verità descrivono il comportamento di un operatore logico rispetto a tutti i possibili input.

Operatore NOT (negazione)

L'operatore NOT inverte il valore logico della variabile: il vero diventa falso e il falso diventa vero.

A	NOT A
0	1
1	0

Operatore AND (congiunzione)

L'operatore AND restituisce vero solo quando entrambe le variabili assumono valore vero.

A	B	$A \wedge B$
0	0	0
0	1	0
1	0	0
1	1	1

Operatore OR (disgiunzione)

L'operatore OR restituisce vero quando almeno una delle due variabili è vera.

A	B	$A \vee B$
0	0	0
0	1	1
1	0	1
1	1	1

6.5 Proprietà degli operatori logici

Gli operatori AND e OR possiedono diverse proprietà utili nella semplificazione delle espressioni booleane.

Proprietà commutativa Per AND e OR vale:

$$A \wedge B = B \wedge AA \vee B = B \vee A$$

Proprietà associativa L'ordine di raggruppamento non modifica il risultato:

$$(A \wedge B) \wedge C = A \wedge (B \wedge C)(A \vee B) \vee C = A \vee (B \vee C)$$

Proprietà distributiva AND e OR si distribuiscono in modo analogo a moltiplicazione e somma:

$$A \wedge (B \vee C) = (A \wedge B) \vee (A \wedge C)A \vee (B \wedge C) = (A \vee B) \wedge (A \vee C)$$

6.6 Leggi di De Morgan

Le leggi di De Morgan permettono di trasformare la negazione di una congiunzione in una disgiunzione di negazioni e viceversa:

$$\neg(A \wedge B) = (\neg A) \vee (\neg B)\neg(A \vee B) = (\neg A) \wedge (\neg B)$$

Queste trasformazioni risultano fondamentali nella progettazione di circuiti logici e nella semplificazione di condizioni nei programmi.

6.7 Espressioni booleane

Un'espressione booleana combina variabili booleane e operatori logici.

Esempio:

$$(A \wedge B) \vee (\neg C)$$

Un'espressione di questo tipo può rappresentare una regola decisionale, una condizione di controllo o il comportamento di un circuito.

7 Codifica dei caratteri

7.1 Dal simbolo al numero

Per memorizzare e trasmettere testo, ogni carattere deve essere associato a un codice numerico. Questo codice viene poi rappresentato in binario e memorizzato come sequenza di bit.

7.2 Codifica ASCII

La codifica **ASCII** (American Standard Code for Information Interchange) è una delle prime standardizzazioni per la rappresentazione dei caratteri.

Caratteristiche principali:

- utilizzo di 7 bit per carattere (128 possibili simboli);
- presenza di lettere maiuscole e minuscole dell'alfabeto inglese;
- cifre da 0 a 9;
- segni di punteggiatura;
- alcuni caratteri di controllo (invio, tabulazione, ecc.).

Esempi:

- il carattere `'A'` corrisponde al codice decimale 65;
- il carattere `'0'` corrisponde a 48;
- lo spazio corrisponde a 32.

7.3 Limiti di ASCII

ASCII è stato progettato per l'inglese. Non include lettere accentate, caratteri di alfabeti non latini, simboli di lingue asiatiche o emoji.

Con la diffusione globale dei calcolatori è diventato necessario un sistema molto più ricco e flessibile.

7.4 Unicode

Unicode nasce con l'obiettivo di assegnare un codice univoco a ogni carattere di ogni lingua, simbolo matematico, segno grafico, emoji.

- Ogni simbolo è identificato da un "punto di codice" (code point), spesso scritto nella forma `U+XXXX`.
- Esempio: la lettera `"A"` ha code point `U+0041`.
- La lettera "è" ha un code point diverso, così come le varie emoji.

Unicode definisce l'insieme dei simboli, ma non il modo specifico in cui vengono rappresentati in memoria.

7.5 UTF-8

UTF-8 è una delle possibili **codifiche** di Unicode. Le sue caratteristiche fondamentali sono:

- codifica a lunghezza variabile: da 1 a 4 byte per carattere;
- compatibilità con ASCII: i primi 128 caratteri (quelli ASCII) vengono rappresentati con un singolo byte identico ad ASCII;
- caratteri non ASCII (accenti, simboli, emoji) utilizzano 2, 3 o 4 byte.

Esempio qualitativo:

- `'A'` $\rightarrow$ 1 byte;
- `'è'` $\rightarrow$ tipicamente 2 byte in UTF-8;
- un'emoji richiede invece più byte.

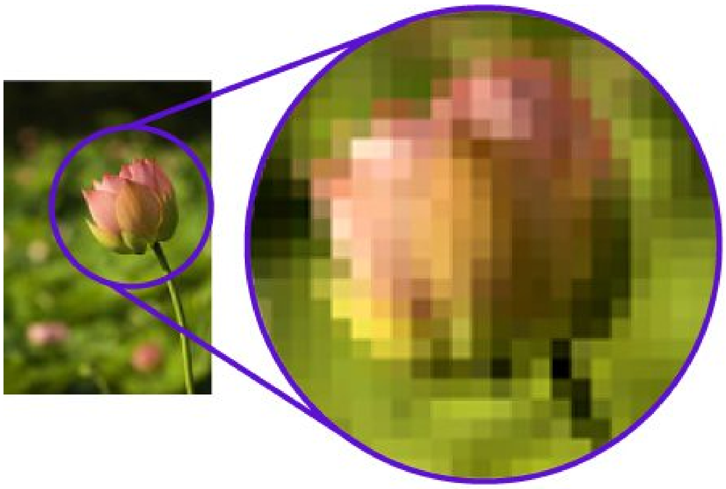
Questa codifica risulta oggi estremamente diffusa, poiché permette di combinare compattezza per il testo semplice e flessibilità per simboli complessi.

8 Codifica delle immagini

8.1 Immagini raster

Le immagini digitali di uso comune (fotografie, screenshot, immagini mostrate su schermo) sono generalmente rappresentate in formato **raster**.

Un'immagine raster è composta da una griglia regolare di **pixel**, ovvero i più piccoli elementi visivi che la costituiscono. Ogni pixel contiene un'informazione relativa al colore.



Ingrandimento di un'immagine raster con evidenza della griglia di pixel.

Quando un'immagine raster viene ingrandita oltre una certa soglia, la struttura a pixel diventa visibile, producendo l'effetto "sgranato".

8.2 Risoluzione

La **risoluzione** indica il numero di pixel presenti in orizzontale e verticale. Determina il livello di dettaglio dell'immagine.

Esempi comuni:

- 800×600 pixel → bassa definizione
- 1920×1080 pixel → Full HD
- 3840×2160 pixel → 4K

8.3 Profondità di colore

La **profondità di colore** indica quanti bit vengono utilizzati per rappresentare il colore di ciascun pixel. Valori tipici:

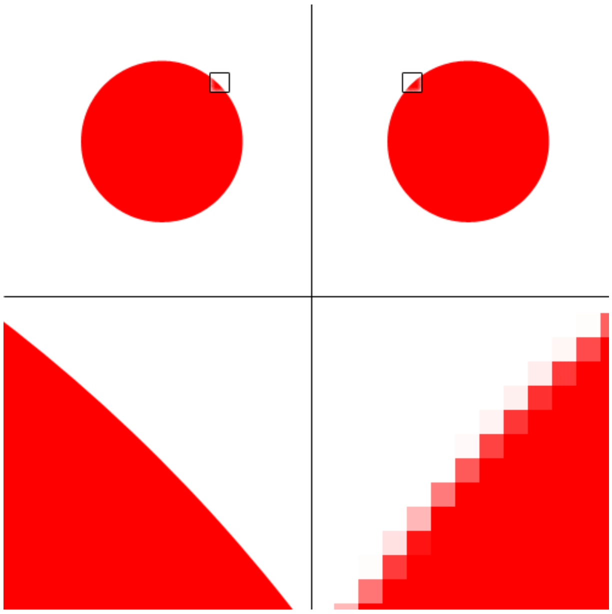
- 1 bit → solo bianco e nero
- 8 bit → 256 livelli di grigio
- 24 bit → RGB, oltre 16 milioni di colori

Il numero di colori rappresentabili è dato dal valore: 2^n

8.4 Immagini vettoriali

Le **immagini vettoriali** non sono descritte tramite pixel, ma attraverso formule matematiche che definiscono linee, curve e forme geometriche. Ogni elemento viene rappresentato come oggetto scalabile, il cui aspetto non dipende dalla risoluzione dello schermo.

Grazie a questa caratteristica, l'ingrandimento non comporta perdita di qualità: i contorni rimangono sempre nitidi e privi di sgranature. Questo rende le immagini vettoriali particolarmente adatte per loghi, icone, diagrammi e grafici tecnici.



Confronto tra immagine raster e vettoriale a forte ingrandimento.

Un comportamento analogo si osserva in molti file PDF: durante lo zoom, testo e linee restano perfettamente definiti perché descritti mediante elementi vettoriali. Se il PDF contiene invece immagini raster, l'ingrandimento evidenzia la struttura a pixel e produce sgranature visibili.