



Università degli Studi di Cagliari  
Corso di Laurea in Ingegneria Biomedica

# **ELEMENTI DI INFORMATICA**

[https://www.unica.it/unica/page/it/gianluca\\_marcialis](https://www.unica.it/unica/page/it/gianluca_marcialis)

A.A. 2021/2022

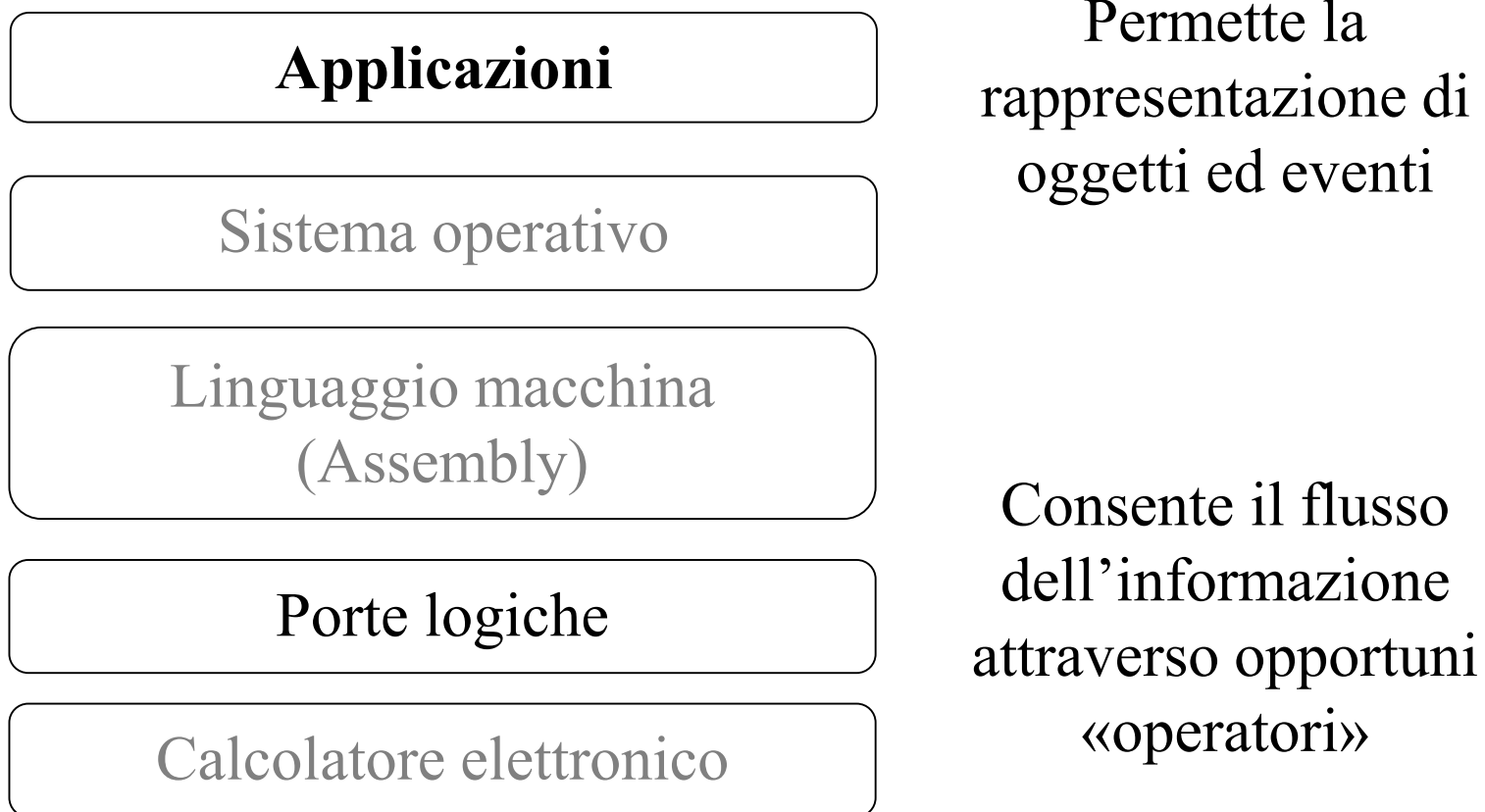
Docente: **Gian Luca Marcialis**

**ALGEBRA DI BOOLE**  
**CODIFICA BINARIA DELL'INFORMAZIONE**

# ALGEBRA BOOLEANA

- Definizione
- Operatori AND, OR, NOT
- Proprietà degli operatori
- Teoremi dell'algebra booleana
- Espressioni booleane
- Altri operatori

# Livelli di «astrazione» di un sistema informatico



# Algebra di Boole

- Definita da George Boole nel 1854 per la manipolazione di espressioni logiche attraverso modelli matematici
- Insieme di operazioni definite su un insieme (supporto)  $B$  composto da due elementi:  $\{0, 1\}$
- Le operazioni sono:
  - AND:  $B \times B \rightarrow B$
  - OR:  $B \times B \rightarrow B$
  - NOT:  $B \rightarrow B$
- Esse sono caratterizzate da determinate **proprietà**

# Tabella della verità degli operatori fondamentali

## ➤ Equivalenti alla

- definizione di complementazione, intersezione ed unione nella teoria degli insiemi
- definizione di negazione, somma e prodotto *logico*

x	<b>NOT(x)</b> Inversione	x	y	<b>AND(x, y)</b>	<b>OR(x, y)</b>
0	1	0	0	0	0
1	0	0	1	0	1
La rappresentazione dei valori delle funzioni booleane per ogni possibile valore degli ingressi viene chiamata <b>“tabella della verità”</b>		1	0	0	1
		1	1	1	1

# Rappresentazione degli operatori

- $AND(x, y) = xy = x \cdot y = x \wedge y$
- $OR(x, y) = x + y = x \vee y$
- $NOT(x) = \bar{x} = x'$
- Utilizzando queste rappresentazioni ed aggiungendo le parentesi, possono scriversi espressioni complesse in modo relativamente semplice. Per esempio:

$$(x + y) \cdot z + \bar{y}$$
$$xy + zw$$

# Proprietà degli operatori booleani

- Proprietà dell'involutione di NOT:
  - $\text{NOT}(\text{NOT}(x)) = x$
- Proprietà dell'idempotenza:
  - $x + x = x; \quad x * x = x$
- Proprietà dell'elemento neutro per OR e AND:
  - $x + 0 = x; \quad x * 1 = x$
- Proprietà dell'elemento nullo per OR e AND:
  - $x + 1 = 1; \quad x * 0 = 0$
- Proprietà dell'elemento complementare per OR e AND:
  - $x + x' = 1; \quad x * x' = 0$
- Proprietà commutativa di AND e OR
  - $x + y = y + x$
  - $x * y = y * x$
- Proprietà associativa di AND e OR
  - $x + (y + z) = (x + y) + z = x + y + z$
  - $x * (y * z) = (x * y) * z = x * y * z$
- Proprietà distributiva di AND e OR
  - $x * (y + z) = x * y + x * z$
  - $x + (y * z) = (x + y) * (x + z)$
- Ogni proprietà si dimostra con la **tabella di verità**, eguagliando riga per riga primo e secondo membro per ogni possibile configurazione delle variabili booleane

# Teoremi dell'algebra booleana

## ➤ Assorbimento

- $x * (x + y) = x$
- $x + (x * y) = x$

## ➤ De Morgan

- $(x + y)' = x' * y'$
- $(x * y)' = x' + y'$

- I teoremi si possono dimostrare scrivendo le **tabelle di verità** delle espressioni booleane a sinistra ed a destra, e verificandone l'esatta coincidenza, oppure applicando le **proprietà** di ciascun operatore



# Espressioni booleane

- Espressioni algebriche che presentano variabili ed operatori booleani, e restituiscono un valore booleano (0 o 1)
- Come tutte le espressioni algebriche, esse possono presentarsi in una forma ridondante e complessa, che può essere *semplificata* applicando le proprietà degli operatori booleani ed i teoremi dell'algebra booleana
- La semplificazione porta a rappresentare le espressioni in modo più compatto, comprensibile, efficiente
- Ipotizzando che ogni operatore booleano presenti un *costo*, la semplificazione permette di ridurre il costo complessivo dell'intera espressione

# Esempio di semplificazione

$\overline{A}\overline{B} + B + \overline{A}C =$	Espressione iniziale (costo 6)
$= B + A\overline{B} + \overline{A}C =$	Commutatività della somma
$= (B + A) \cdot (B + \overline{B}) + \overline{A}C =$	Distributività della somma
$= (B + A) \cdot 1 + \overline{A}C =$	Propr. El. complementare
$= B + A + \overline{A}C =$	Proprietà dell'elemento neutro
$= B + (A + \overline{A}) \cdot (A + C) =$	Distributività della somma
$= B + A + C.$	Espressione semplificata (costo 2)

# Semplificazione di espressioni: esercizi

➤  $AB + A'B = ?$

➤  $AB + B = ?$

➤  $ABC + A'B = ?$

➤  $A' + AB = ?$

➤  $A'B' + AB = ?$

➤  $A'C + A'BC' = ?$

➤  $B' + C' + BC = ?$

➤  $B' + C + BC = ?$

➤  $B'C + C' + AB = ?$

➤  $A' + B + AB + A'B' = ?$

➤  $A' + B' + A'B + A'B' = ?$

➤  $A'C + BC' + A'B'C = ?$

Verificate il corretto esito della semplificazione con le tabelle di verità

# Perché l'algebra booleana?

- I valori '0' o '1' di una variabile booleana possono essere associati ai valori di "falsità" o "verità" di una espressione in linguaggio naturale
- Scrivendo algoritmi la conoscenza delle funzioni dell'algebra booleana può essere molto utile
- Es.
  - Siano date le variabili booleane *StudiaBiomedica* , *FrequentaPrimoAnno* , *EsamePassato*
  - Scrivere l'espressione della variabile booleana *SegueElementiInformatica* attraverso gli operatori AND, OR, NOT:

$$\begin{aligned} \textit{SegueElementiInformatica} &= \\ &= \textit{StudiaBiomedica} \cdot (\textit{FrequentaPrimoAnno} + \overline{\textit{EsamePassato}}) \end{aligned}$$

# La «diversità» in informatica

- Dati due valori booleani  $A$  e  $B$ , progettare una funzione  $F(A,B)$  tale che:
  - Se  $A$  è diverso da  $B$ , allora  $F = \text{'true'}$  («vero», 1)
  - Altrimenti,  $F = \text{'false'}$  («falso», 0)
- Come lo esprimiamo con gli operatori a nostra disposizione?
  - AND, OR, NOT

# La diversità: operatori XOR e XNOR

x	y	XOR(x, y) x è diverso da y?	XNOR(x, y) x è uguale a y?
0	0	0 = no	1 = sì
0	1	1 = sì	0 = no
1	0	1 = sì	0 = no
1	1	0 = no	1 = sì

# Altri operatori

x	y	NAND(x, y)	NOR(x, y)
0	0	1	1
0	1	1	0
1	0	1	0
1	1	0	0

# Espressione degli operatori booleani

$$\text{➤ } XOR(A, B) = A \oplus B = \bar{A} \cdot B + A \cdot \bar{B}$$

$$\text{➤ } XNOR(A, B) = \overline{A \oplus B} = A \cdot B + \bar{A} \cdot \bar{B}$$

$$\text{➤ } NAND(A, B) = \overline{A \cdot B}$$

$$\text{➤ } NOR(A, B) = \overline{A + B}$$

Verificare che le espressioni in rosso coincidano con la definizione degli operatori date nelle slide precedenti tramite le tabelle di verità



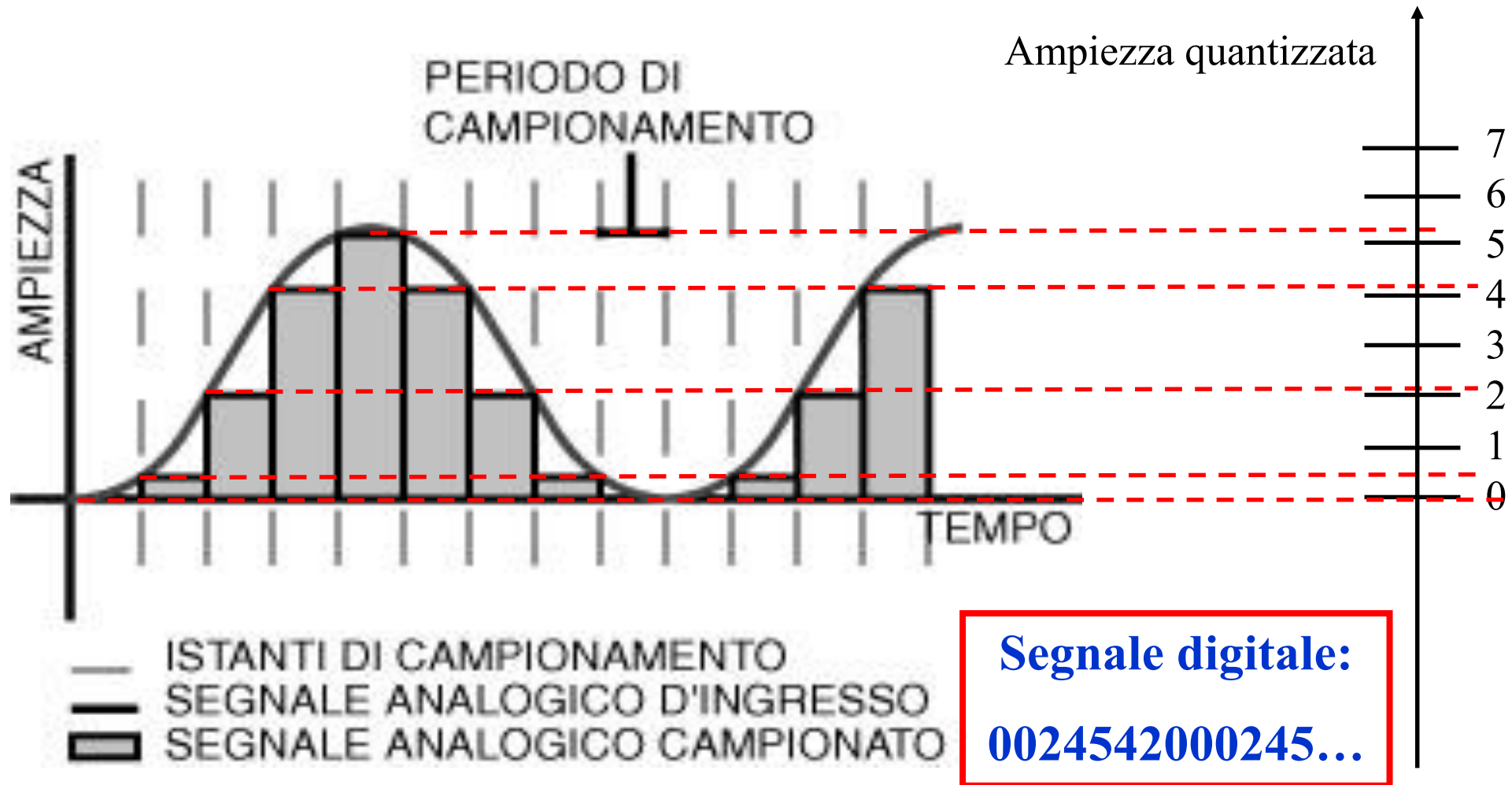
# RAPPRESENTAZIONE BINARIA DELLE INFORMAZIONI

- Analogico vs. Digitale
- Rappresentazione posizionale
  - Conversione binaria-decimale
  - Aritmetica elementare
- Rappresentazione di interi senza e con segno
- Rappresentazione di numeri in virgola mobile
- Codifica dei caratteri e delle immagini

# Analogico Vs. Numerico (*digitale*)

- Sistemi **analogici**: la grandezza da misurare viene rappresentata con un'altra grandezza “più pratica” da utilizzare (**continua**, e “proporzionale” ad essa)
- Sistemi **numerici (*digitali*)**: la grandezza da misurare viene rappresentata da un numero
  - Esempi: disco in vinile Vs. CD, telefono mobile TACS Vs. GSM
- E' più semplice correggere gli errori nella trasmissione di numeri (segnale “discreto”) piuttosto che nella trasmissione di una grandezza continua
  - Ad esempio le cifre possibili potrebbero essere solo due (sistema binario)

# Esempio di codifica analogico-digitale



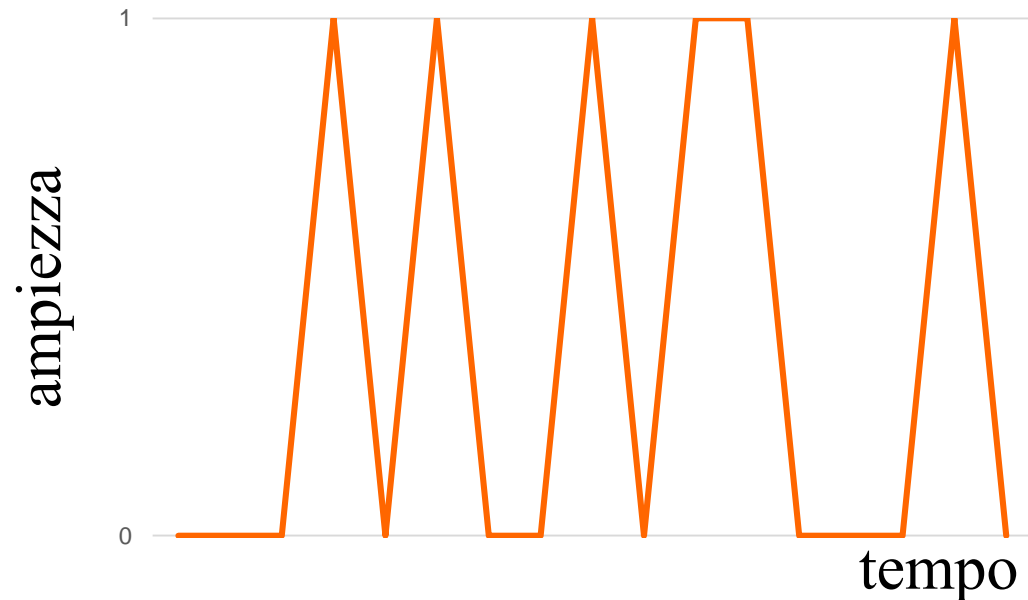
Ciascuna delle cifre componenti il segnale viene infine convertita in una sequenza di **bit**

# Binarizzazione del segnale digitale

**Segnale digitale:**  
**0024542000245...**

**Segnale binarizzato:**  
**000000010100101100010000000000...**

Cifra	Bit
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111



# Codifica binaria

- Tutti i dati devono essere *codificati in forma binaria* per poter essere comprensibili a un calcolatore
- Il *bit* è l'unità di informazione. Corrisponde allo *stato* di un dispositivo fisico a *due stati*
  - Ad es. tensione elettrica, polarizzazione magnetica, ecc.
- I *due stati* vengono interpretati come 0 o 1
  - Falso/Vero, Spento/Acceso
- Scelta di due soli stati: motivazioni tecnologiche
  - Minore probabilità di guasti ed errori

# Codifica binaria (cont.)

- I bit vengono organizzati in:
  - **byte** (sequenze di 8 bit)
  - **parole** (**word**, sequenze di byte che compongono una *cella* di memoria centrale; tipicamente 16, 32, 64 bit)
- I numeri interi e frazionari, i caratteri, le immagini, i suoni, ecc. possono essere tradotti in byte e parole
- Il calcolatore è in grado di operare sia la *codifica* in binario che la *decodifica* in decimale
  - E' totalmente trasparente per l'utente e, a volte, anche per il programmatore

# Numeri naturali

- $\{0, 1, 2, 3, \dots\}$
- Sistema usato comunemente: *arabico*
- Numeri rappresentati come sequenze ordinate di cifre, in **base dieci** (dieci cifre: 0, 1, ..., 9)
- Sistema **posizionale**: il significato di ciascuna cifra (unità, decine, centinaia, ecc.) dipende dalla posizione che occupa nella sequenza
- Altri sistemi:
  - **additivi**: bastoncini (ciascuno rappresenta una unità)
  - sistema di numerazione romano (es.: 3000  $\equiv$  MMM, non posizionale!)

# Rappresentazione posizionale

- Sistema in base  $p$ : le cifre sono  $(0, \dots, p-1)$
- Un qualsiasi numero  $N_p$  è rappresentato dalla sequenza:

$$a_n a_{n-1} \dots a_0$$

$$N_p = a_n \cdot p^n + \dots + a_0 \cdot p^0 = \sum_{i=0}^n a_i \cdot p^i$$

$a_n$  è la **cifra più significativa**,  $a_0$  è la **cifra meno significativa**

- Es.:  $587_{10} = (5 \cdot 10^2 + 8 \cdot 10^1 + 7 \cdot 10^0)$
- Per passare ad un'altra base  $q$  è sufficiente esprimere i coefficienti  $a_i$  e le potenze  $p^i$  in base  $q$



# I sistemi binario, ottale, esadecimale

- Nei calcolatori le basi usate sono 2, 8, e 16 (sistemi *binario*, *ottale*, *esadecimale*)
- Base  $p = 2$ . Cifre dell'alfabeto: 0 e 1  
Es.:  $101001011_2 = (1 \cdot 2^8 + 0 \cdot 2^7 + 1 \cdot 2^6 + 0 \cdot 2^5 + 0 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0)_{10} = (256 + 64 + 8 + 2 + 1)_{10} = 331_{10}$
- Base  $p = 8$ . Cifre dell'alfabeto: 0, 1, ..., 7  
Es.:  $534_8 = (5 \cdot 8^2 + 3 \cdot 8^1 + 4 \cdot 8^0)_{10} = 348_{10}$
- Base  $p = 16$ . Cifre: 0, 1, ..., 9, A, B, C, D, E, F  
Es.:  $B7F_{16} = (11 \cdot 16^2 + 7 \cdot 16^1 + 15 \cdot 16^0)_{10} = 2943_{10}$

# Conversione decimale-binario

- Innanzi tutto, richiamiamo la definizione:

$$N_p = a_n \cdot p^n + \dots + a_0 \cdot p^0 = \sum_{i=0}^n a_i \cdot p^i$$

- Conversione decimale-binario significa, partendo da un numero  $N_p$  espresso in base  $p$ , ottenere un numero  $N_{p'}$  espresso in base  $p'$ .
- In altri termini, significa ottenere i coefficienti  $a_i$  legati alla base  $p'$  a partire da  $N_p$ .
- Per questo scopo, sfruttiamo appunto la definizione, ricordando che, dividendo  $N_p$  per la base  $p$  si ottiene un quoziente  $Q$  ed un resto  $R$
- $R$  sarà compreso tra 0 e  $p-1$ .

# Conversione decimale-binario: il concetto

- In generale, quindi  $N_p = Q * p + R$
- Sia  $N_p = 7$  e  $p = 10$ .
- Vogliamo rappresentare il numero con un'altra base  $p' = 2$ . Il cambio di base imporrà che  $N_p = Q' * p' + R' = N_{p'}$
- $7_{10} = 3 * 2 + 1 =$   
 $= (1 * 2 + 1) * 2 + 1 =$   
 $= ((0 * 2 + 1) * 2 + 1) * 2 + 1 =$   
 $= 0 * 2^3 + 1 * 2^2 + 1 * 2^1 + 1 * 2^0$   
 $= 0111_2$
- In altre parole 7, espresso in base 10, può essere espresso con la sequenza **0111** in base **2**
  - Lo '0' nella posizione più significativa, in questo caso, può anche essere omesso, in quanto non ha effetto nella conversione opposta
- A questo punto possiamo avvicinarci all'algoritmo vero e proprio con un altro esempio, slegandoci dalla definizione

# Esempio di conversione da decimale a binario: algoritmo delle divisioni successive

$331 : 2 = 165$	$(331 \bmod 2 = 1)$ <i>bit meno significativo</i>
$165 : 2 = 82$	$(165 \bmod 2 = 1)$
$82 : 2 = 41$	$(82 \bmod 2 = 0)$
$41 : 2 = 20$	$(41 \bmod 2 = 1)$
$20 : 2 = 10$	$(20 \bmod 2 = 0)$
$10 : 2 = 5$	$(10 \bmod 2 = 0)$
$5 : 2 = 2$	$(5 \bmod 2 = 1)$
$2 : 2 = 1$	$(2 \bmod 2 = 0)$
$1 : 2 = 0$	$(1 \bmod 2 = 1)$ <i>bit più significativo</i>

$$331_{10} = 101001011_2$$

# L'algoritmo delle divisioni successive in “pseudo-codice”

- Ingresso (Input): una sequenza di cifre decimali  $X$  (il numero da convertire)
- Uscita (Output): una sequenza di  $n$  cifre binarie (il numero convertito)  $\{b_{n-1} \dots b_0\}$
- $i = 0$ ;      (operatore “=”: assegnazione)
- **Ripeti**
  - $Q = \text{quoziente di } X/2$ ;
  - $R = \text{resto di } X/2$ ;
  - $b_i = R$ ;
  - $X = Q$ ;
  - $i = i + 1$ ;
- **Finché**  $Q \neq 0$ 
  - Nota “ $\neq$ ” significa “diverso da”

# Esercizio

➤ Scriviamo in forma algoritmica (“pseudo-codice”) il metodo per convertire un numero binario  $b_{N-1} \dots b_0$  ad N bit, dove  $b_i$  è l' $i$ -esimo bit, in un numero decimale X.

➤ Soluzione

- $i=0$ ;
- $X = 0$ ;
- Ripeti
  - $X = X + b_i * 2^i$
  - $i = i + 1$
- Finché  $i < N$

# Nota bene

- Nella realtà non disponiamo di un numero infinito di bit per rappresentare i valori numerici
  - Costo e spazio
  - I bit sono implementati ciascuno da un dispositivo elettronico!
- Quindi saremo sempre vincolati ad un numero di bit prefissato
- Quanti valori posso rappresentare con  $n$  bit?

# La rappresentazione dei numeri interi con segno

## ➤ Segno e valore

- Si separano “valore assoluto” e “segno” del numero: il primo si converte come visto in precedenza (divisioni successive), al secondo si dedica un bit specifico detto appunto “di segno”, di solito nella posizione più significativa

## ➤ In eccesso

- Si traslano di un valore  $K$ , detto “eccesso” o “polarizzazione”, i valori da rappresentare in modo che risultino tutti come fossero “positivi”

## ➤ Complemento a due



# Esempio

➤ Avendo a disposizione **tre** bit

	Rappresentazione decimale			
Rappresentazione binaria	Valore senza segno	Segno e valore	Eccesso 3	Complemento a due
000	0	0	-3	0
001	1	1	-2	1
010	2	2	-1	2
011	3	3	0	3
100	4	0	1	-4
101	5	-1	2	-3
110	6	-2	3	-2
111	7	-3	4	-1

# Rappresentazione in complemento a due

- Il complemento a due, indicato con  $C_2$ , di un numero  $X$  ad  $n$  bit è definito come segue:

$$X + C_2 = 2^n$$

- Poiché  $2^n$  si rappresenta con  $n+1$  bit, dei quali solo quello più significativo è pari ad 1, è ragionevole *rappresentare*  $-X$  con  $C_2$ .
- Per calcolare  $C_2$  (ovvero  $-X$ ) dato  $X$ , si dimostra che:
  - Vanno invertiti tutti i bit di  $x$  (si applica l'operatore NOT a ciascuno di essi)
  - Si somma 1 al risultato

# Complemento a due: un esempio

- Sia  $X = 1001$ , di cui si vuole calcolare il complemento a due
- Prima di tutto, si calcola  $X' = 0110$
- Poi si somma 1:  $X' + 1 = 0110 + 1 = 0111$
- Per sapere se il calcolo è corretto, la verifica si può svolgere come segue:
  - Si fa il complemento a due del valore ottenuto, e si controlla se esso coincide col valore iniziale  $X$ , oppure
  - Si converte il valore ottenuto in un numero decimale con segno, e si controlla se esso coincide con il valore  $-X$  espresso in decimale

# Conversione binario-decimale di un numero in complemento a due

- Se un valore  $X$  in complemento a 2 è dato ad  $n$  bit, la traduzione in decimale segue l'algoritmo:

$$X_{(10)} = -2^{n-1} \cdot b_{n-1} + \sum_{i=0}^{n-2} 2^i \cdot b_i$$

- Esempio:  $X = (1001)_2$ ;  $n=4$
- $X_{(10)} = -2^{n-1} \cdot b_{n-1} + \sum_{i=0}^{n-2} 2^i \cdot b_i = -2^3 + 1 = -8 + 1 = -7$
- Il numero dato  $X$  corrisponde in decimale al valore -7.
- Esercizio: scrivere il procedimento in forma algoritmica.

# Somma di numeri binari

x	y	Somma (XOR)	Riporto (AND)
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

- La somma di due numeri binari si esegue attraverso **l'algoritmo con propaga-zione del riporto** già noto per la somma di numeri decimali
- Se, nella somma di numeri di n bit, il bit di riporto generato dalla somma dei bit più significativi con il precedente riporto è 1, il numero ottenuto non può essere rappresentato con n bit: **overflow**

# Numeri frazionari in virgola «fissa»

- Estensione della rappresentazione posizionale con esponente della base espresso come valore negativo:

$$X = \sum_{i=0}^{N-1} b_i \cdot 2^i + \sum_{i=-1}^{-M} b_i \cdot 2^i$$

- Esempio:

$$\begin{aligned} 12.\mathbf{25}_{10} &= 1100.\mathbf{01}_2 \\ 0.01_2 &= 0 \cdot 2^{-1} + 1 \cdot 2^{-2} = 0.25_{10} \end{aligned}$$

# Algoritmo delle moltiplicazioni successive per la conversione di numeri frazionari in binario

- Serve per convertire un valore frazionario decimale ( $<1$ ) nel corrispondente frazionario binario
- Ingresso: valore frazionario decimale  $X$  (esempio: 0.25)
- Uscita: valore frazionario binario  $f_{-1} \dots f_{-n}$  corrispondente ( $f_{-1}$  è il bit più significativo)
- $i = -1$ ;
- **Ripeti**
  - $M = X * 2$ ;
  - $f_i$  = Parte intera di  $M$ ;
  - $X$  = Parte frazionaria di  $M$ ;
  - $i = i - 1$ ;
- **Finché**  $X \neq 0$ ;

# Esecuzione dell'algoritmo

- Convertire il valore decimale 0.75 in binario frazionario utilizzando l'algoritmo delle moltiplicazioni successive
  - $X=0.75$
  - $i=-1$
  - $M=2*X=2*0.75=1.5$
  - $f_i=f_{-1}$ =Parte Intera di  $M=1$
  - $X$ =Parte Frazionaria di  $M=0.5$
  - $i=i-1$  N.B. ora  $i=-2$
  - $X \neq 0$ ? ➔ sì, ripeto le operazioni precedenti
  - $M=2*0.5=1.0$
  - $f_i=f_{-2}=1$
  - $X=0$
  - $i=i-1$
  - $X \neq 0$ ? ➔ no, algoritmo termina

➤ Risultato:  $(0.75)_{10} = (0.11)_2$



# Rappresentazione in virgola mobile

- Risulta utile usare la notazione scientifica:

$$\pm M \times B^E$$

- Un numero reale può essere memorizzato con una parola con tre campi:
  - Segno ( $\pm$ )
  - Mantissa (M)
  - Esponente (E).
- Questa notazione è conosciuta come “floating point” (“virgola mobile”).

# Esercizio

- Convertire il valore decimale 12.25 nel formato in virgola mobile  $1.b \cdot 2^E$ , con
- $E$  esponente a tre bit espresso in complemento a 2,
  - $b$  mantissa frazionaria a cinque bit,
  - l' $1$  in parte intera è quello nella posizione più significativa del numero binario.

# Algoritmo «dall'alto»

1. Convertire la parte intera in binario
2. Convertire la parte frazionaria in binario
3. Portare l'1 più significativo come unico elemento in parte intera, in modo da generare il numero nel formato desiderato  $1.b \cdot 2^E$
4. Convertire l'esponente in complemento a due con tre bit

# Soluzione

- Fase 1: per convertire la parte intera si applica l'algoritmo delle divisioni successive  $\rightarrow 12_{10} = 1100_2$
- Fase 2: per convertire la parte frazionaria si applica l'algoritmo delle moltiplicazioni successive  $\rightarrow 0.25_{10} = 0.01_2$
- Il numero ottenuto è 1100.01
- Fase 3: moltiplichiamo e dividiamo per  $2^E$ , con E tale che si possa scrivere  $1100.01 = 1.10001 * 2^3$
- Fase 4: convertiamo l'esponente 3 in complemento a due con tre bit ancora con le divisioni successive
- Poiché è positivo non sono necessari altri passaggi e si ottiene infine:

$$12.25_{10} = 1.10001 * 2^{011}$$

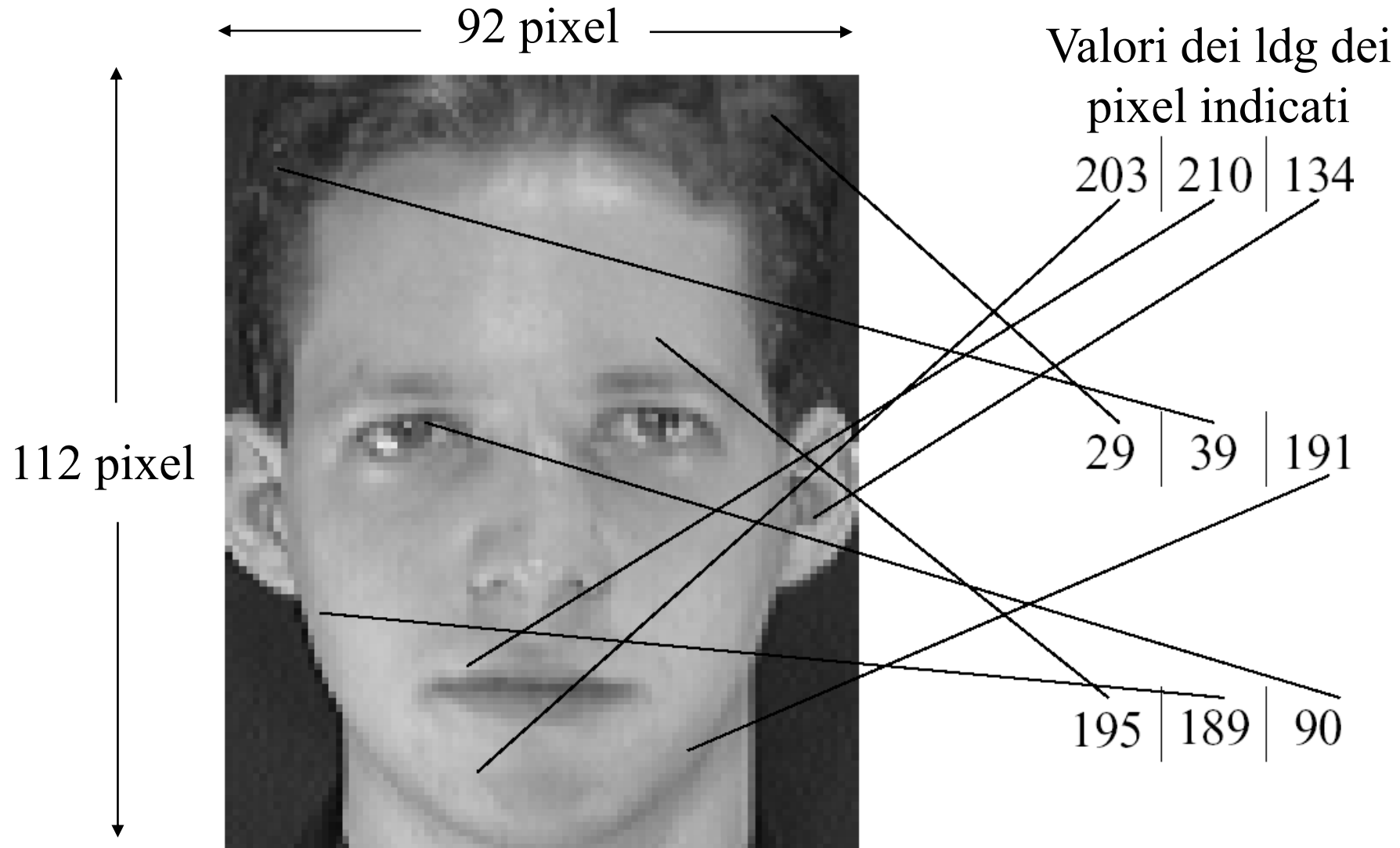
# Codifica dei caratteri

- I caratteri vengono codificati tramite sequenze di bit
  - Codice più usato: **ASCII** (**A**merican **S**tandard **C**ode for **I**nformation **I**nterchange) a 7 bit (128 caratteri), di solito esteso a 8 bit (256 caratteri)
  - Tre categorie di caratteri
    - Caratteri di comando**: codici di trasmissione o di controllo
    - Caratteri alfanumerici**: da 'A' a 'Z', da 'a' a 'z' e da '0' a '9'
    - Simboli**: punteggiatura e operatori aritmetici
- Le lettere accentate, i caratteri greci ecc. fanno parte del codice esteso

# Codifica delle immagini

- L'immagine è suddivisa in punti (pixel) e ciascun punto è codificato con un numero che corrisponde
  - A un particolare colore
  - A un particolare tono di grigio nelle immagini b/n
- In genere si utilizza un numero di colori o di sfumature di grigio che sia potenza di 2 per rappresentare un'immagine come sequenza di byte
- Deve essere memorizzata anche la dimensione dell'immagine e la risoluzione (dpi, "*dot per inch*")

# Codifica delle immagini a livelli di grigio



## Struttura della rappresentazione (PGM)

[illegible]



# Codifica del colore: il formato BitMaP (BMP)



Ad ogni pixel sono associati tre valori tra 256 possibili:

- **Uno per il livelli di rosso (Red)**
- **Uno per il livello di verde (Green)**
- **Uno per il livello di blu (Blue)**

Il colore finale è dato dalla media dei valori di queste tre fonti ottenendo tutte le gradazioni possibili ( $256^3$ )

Altro formato: TIFF

# Formati «compressi»

## ➤ JPEG (Joint Photographic Expert Group)

- A partire da una immagine BMP, si rappresentano solo le componenti in frequenza percepibili dall'occhio umano, omettendo le altre. L'immagine risultante richiede *meno* byte di quella originale.



## ➤ Altro formato compresso: GIF

# Sensori analogico-digitale



Scanner per documenti



Videocamera 3D

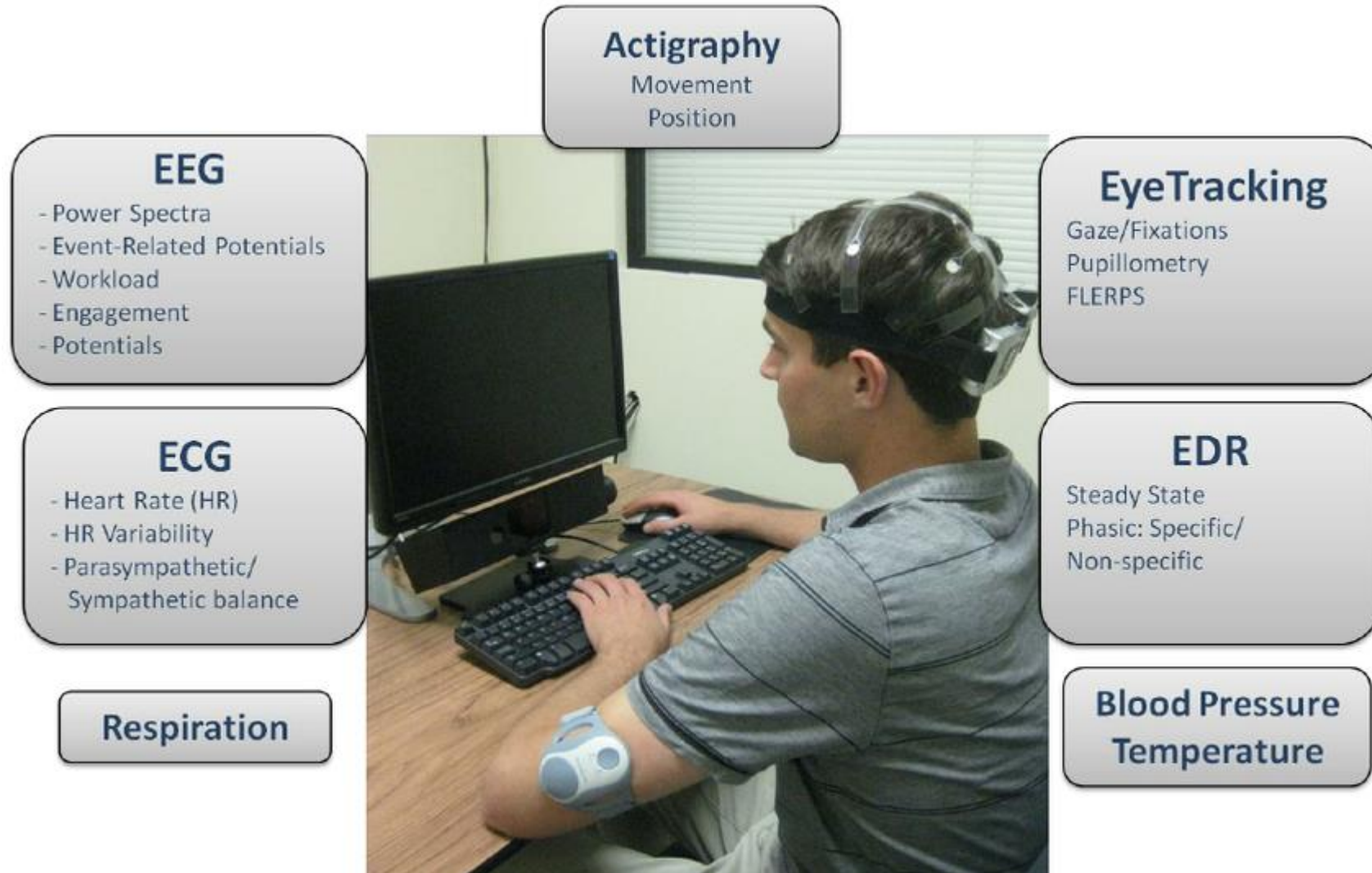


Registratore

Scanner biometrici (impronte digitali, impronte palmari, iride)



# Altri sensori...





# Per saperne di più...

- Codifica binaria dell'informazione
  - Mandrioli, et al., Capitolo 2
- I caratteri ASCII
  - Mandrioli, et al., Appendice A
- Numeri binari e algebra booleana
  - Schneider, Gersting, "Informatica", Apogeo, Capitolo 4
- Rappresentazione di immagini
  - Mezzaluna, Piccolo, "Capire l'informatica", DeAgostini, Cap. 2