

# Lezione 2:

## Codifica binaria dell'informazione

Codifica binaria  
Elaborazione di dati binari

# Materiale didattico

- ◆ Lucidi delle lezioni, disponibili al sito:

<http://wwwinfo.deis.unical.it/~irina>

- ◆ Oppure sul sito di iCampus:

<http://icampus.deis.unical.it/>

# Il concetto di Informazione

- ◆ Informatica è scienza della rappresentazione e dell'elaborazione dell'informazione, studia le caratteristiche dell'informazione ed i modi di usarla, immagazzinarla, elaborarla e trasportarla in modo automatico
- ◆ Informazione non esiste senza un supporto fisico ma si distingue da esso

**Canzone**

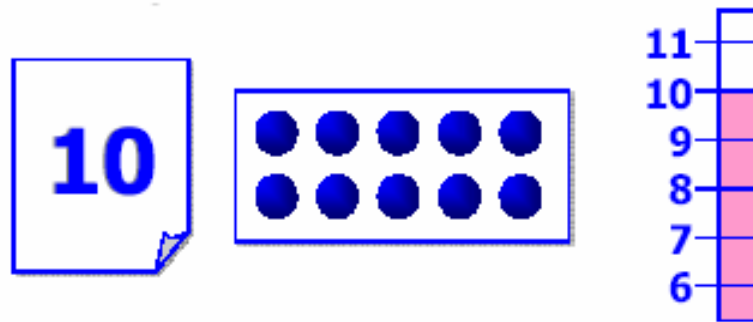
**Area che trasporta onda sonora**

**Compact disk**

**Spartito**

# Il concetto di Informazione

- ◆ La stessa informazione può essere scritta su supporti differenti

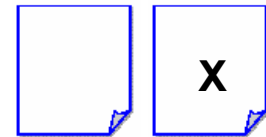


- ◆ Stesso supporto può portare informazioni differenti



# Il concetto di Informazione

- ◆ Condizione necessaria perché un supporto sia in grado di portare informazione è che esso può assumere **configurazioni differenti**, a ognuna delle quali venga associata una differente **entità di informazione**
- ◆ Un supporto che possa presentarsi sempre e comunque in un unico modo non può portare alcuna informazione
- ◆ Il caso più semplice è quando le configurazioni possibili sono due
- ◆ Per interpretare le differenti configurazioni del supporto è necessario un **codice**, cioè una regola, la cui validità è concordata per convenzione, che ad ogni possibile configurazione del supporto associa un'entità di informazione.



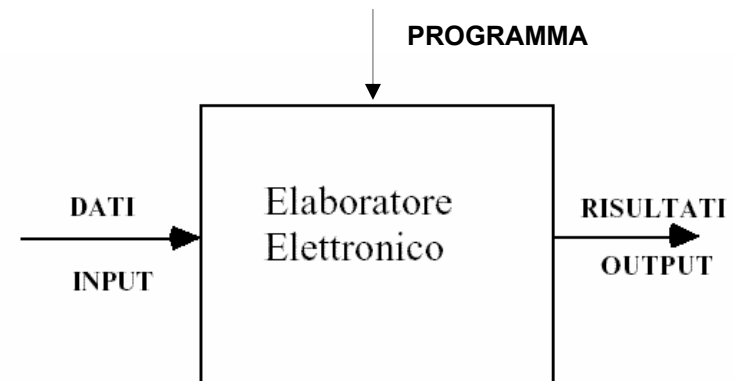
**Superato**



**Non Superato**

# Codifica dell'informazione

- ◆ L'informazione gestita dai sistemi di elaborazione deve essere **codificata** per poter essere memorizzata, elaborata, scambiata,...
- ◆ Necessario codificare dati e istruzioni
  - Algoritmo = insieme di istruzioni che operano su dati
  - Per eseguire un programma è necessario codificare e memorizzare sia i dati sia il programma (le istruzioni)
- ◆ L'esecutore automatico deve essere in grado di:
  - Memorizzare istruzioni e dati
  - Manipolare istruzioni e dati



# Sistemi di codifica

- ◆ Detto anche “codifica” o “codice”
- ◆ Usa un insieme di simboli di base (**alfabeto**)
- ◆ I simboli dell’alfabeto possono essere combinati ottenendo differenti **configurazioni**
  - Sono distinguibili l’una dall’altra
  - Sono dette anche “codici”, “stati”
- ◆ Associa ogni configurazione ad una particolare entità di informazione
  - la configurazione diventa un modo per rappresentarla

# Sistemi di codifica: numeri (es.)

## ◆ Alfabeto

- cifre: “0”, “1”, “2”, ..., “9”
- separatori: decimale (“,”), migliaia (“.”)
- segni: positivo (“+”), negativo (“-”)

## ◆ Regole di composizione (sintassi)

- definiscono le combinazioni ammissibili (ben formate)
  - 12.318,43      OK
  - 12,318,43      **ERRORE!**



# Sistemi di codifica: numeri (es.)

## ◆ Codice (semantica)

- Associano ad ogni configurazione un'entità di informazione

- $2.318,43 = 2 \times 10^3 + 3 \times 10^2 + 1 \times 10^1 + 8 \times 10^0 + 4 \times 10^{-1} + 3 \times 10^{-2}$   
 $\quad \quad \quad \begin{smallmatrix} 3 & 2 & 1 & 0 & -1 & -2 \end{smallmatrix}$

## ◆ Sistemi diversi possono usare lo stesso alfabeto

- $123,456 = 1 \times 10^2 + 2 \times 10^1 + 3 \times 10^0 + 4 \times 10^{-1} + 5 \times 10^{-2} + 6 \times 10^{-3}$  [IT]
- $123,456 = 1 \times 10^5 + 2 \times 10^4 + 3 \times 10^3 + 4 \times 10^2 + 5 \times 10^1 + 6 \times 10^0$  [UK]

# Sistemi di codifica

- ◆ Si utilizzano standard internazionali per risolvere problemi di compatibilità tra calcolatori di tipo e marca diversi
- ◆ Vedremo brevemente:
  - Codifica di numeri
  - Codifica di caratteri
  - Codifica di dati multimediali

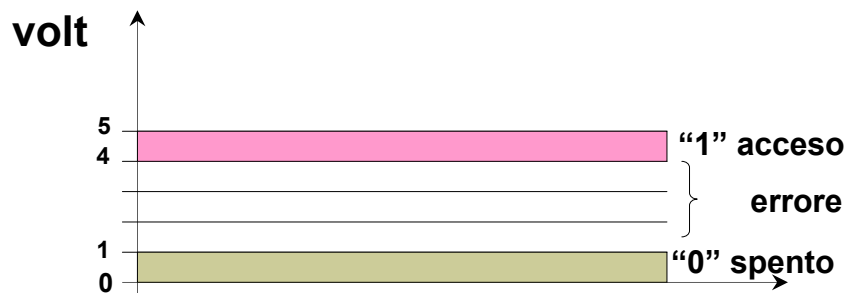
# Codifica binaria

## ◆ Gli elaboratori utilizzano la codifica binaria dell'informazione

- codifica binaria è caratterizzata dall'alfabeto, che contiene solo 2 simboli

## ◆ Perché solo due simboli?

- Differenti tensioni elettriche, polarità magnetiche, ...
- Riduce errori (ad es. causati da rumore nei segnali)



# Codifica binaria

## ◆ BIT (BInary digiT)

- unità elementare di informazione rappresentabile con dispositivi elettronici
- con 1 bit si possono rappresentare 2 stati
  - 0/1, on/off, sì/no

## ◆ Combinando più bit si può codificare un numero maggiore di stati

- con 2 bit possono rappresentare 4 stati
- con **K** bit si possono rappresentare  **$2^K$**  stati

## ◆ Quanti bit servono per codificare N stati?

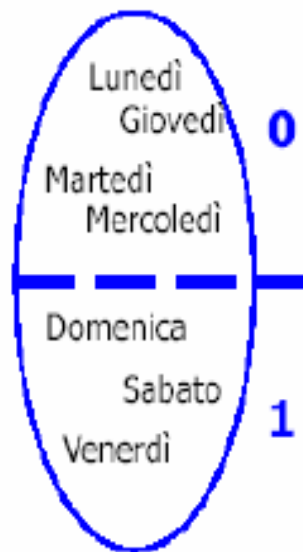
- $N \leq 2^K \rightarrow K \geq \log_2 N \rightarrow \mathbf{K \geq \lceil \log_2 N \rceil}$

0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

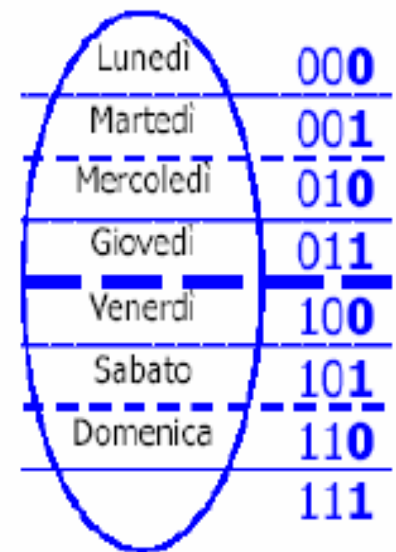
# Es.: i giorni della settimana in binario



**1 bit**  
**2 "gruppi"**



**2 bit**  
**4 "gruppi"**



**3 bit**  
**8 "gruppi"**

# Codifica binaria: unità di misura

◆ **BYTE** = 8 bit

■ può rappresentare  $2^8 = 256$  stati

◆ **Multipli di Byte:**

■ **KiloByte** (KB) =  $2^{10}$  byte = 1024 byte  $\cong 10^3$  byte

■ **MegaByte** (MB) =  $2^{20}$  byte  $\cong 10^6$  byte

■ **GigaByte** (GB) =  $2^{30}$  byte  $\cong 10^9$  byte

■ **TeraByte** (TB) =  $2^{40}$  byte  $\cong 10^{12}$  byte

# Sistemi di codifica

## ◆ Vedremo brevemente:

- **Codifica di numeri**
  - Naturali
  - Interi
  - Razionali
- Codifica di caratteri
- Codifica di dati multimediali

# Sistema di numerazione posizionale

- ◆ Sistema di numerazione posizionale con **base  $\beta$** 
  - $\beta$  simboli (**cifre**) corrispondono ai numeri da 0 a  $\beta-1$
  - i numeri naturali maggiori o uguali a  $\beta$  possono essere rappresentati da una sequenza di cifre
- ◆ Se un numero naturale  $N$  è rappresentato in base  $\beta$  dalla sequenza di  $n$  cifre

$$\mathbf{a_{n-1} \ a_{n-2} \ \dots \ a_1 \ a_0}$$

allora  $N$  può essere espresso come segue:

$$N = \sum_{i=0}^{n-1} a_i \beta^i = a_{n-1} \beta^{n-1} + a_{n-2} \beta^{n-2} + \dots + a_2 \beta^2 + a_1 \beta + a_0$$



# Codifica dei numeri naturali

## ◆ Esempio:

- tredici può essere espresso mediante potenze di 2 come:


$$13_{10} = 1 \times 8 + 1 \times 4 + 0 \times 2 + 1 \times 1$$

- cioè può essere rappresentato dalla sequenza di bit  
(*stringa binaria*)

1 1 0 1

# Conversione decimale-binario


$18 : 2 = 9$	resto 0
$9 : 2 = 4$	resto 1
$4 : 2 = 2$	resto 0
$2 : 2 = 1$	resto 0
$1 : 2 = 0$	resto 1



**10010**



$137 : 2 = 68$	resto 1
$68 : 2 = 34$	resto 0
$34 : 2 = 17$	resto 0
$17 : 2 = 8$	resto 1
$8 : 2 = 4$	resto 0
$4 : 2 = 2$	resto 0
$2 : 2 = 1$	resto 0
$1 : 2 = 0$	resto 1



**10001001**



# Codifica dei numeri naturali

## ◆ Quindi

- Numero = sequenza di bit (codifica in base 2)
- Con **K** bit si rappresentano i numeri da 0 a  **$2^K - 1$**

## ◆ Esempi:

- **2** = sequenza **1 0**
- **3** = sequenza **1 1**
- **4** = sequenza **1 0 0**
- .....

# Codifica dei numeri interi

- ◆ Modulo e segno
- ◆ Complemento a 1
- ◆ Complemento a 2
  - comunemente usata nei sistemi reali

# Modulo e segno

- ◆ 1 bit per rappresentare esplicitamente il segno
  - 0 → +
  - 1 → -
- ◆ Gli altri bit rappresentano il valore assoluto del numero come binario puro
- ◆ Esempi (su 8 bit):
  - -2 → 1 0000010
  - +5 → 0 0000101

# Modulo e segno (2)

## ◆ Note:

- Segno completamente disgiunto dal valore del numero
- Posizione del bit del segno, entro la stringa, irrilevante

## ◆ Difetti:

- Il valore zero ha due distinte rappresentazioni:
  - 1000000000 → -0
  - 0000000000 → +0
- Non permette di utilizzare le usuali regole di calcolo per eseguire le operazioni.  $(X + (-X)) = 0$

+5	0 0000101
- 5	1 0000101
<hr/>	
0	1 0001010

# Complemento a uno

## ◆ Approccio

- La rappresentazione dei numeri negativi si ottiene dalla rappresentazione del numero positivo invertendo i bit

## ◆ Esempi (su 8 bit compreso il bit del segno) :

- $+5 \rightarrow 0\ 0000101$
- $-5 \rightarrow 1\ 1111010$

## ◆ Difetti

- Simili a quelli della rappresentazione in modulo e segno  
( $+0 \rightarrow 00000000$      $-0 \rightarrow 11111111$ )

## Complemento a due (1)

- ◆ Algoritmo per calcolare la rappresentazione in complemento a 2 di un numero **negativo**:

1. si rappresenta il valore assoluto in binario
2. si invertono tutte le cifre (1→0 e viceversa)
3. si somma 1

- ◆ Esempio

■ 35	→	00100011	<b>35+(-35)=0</b> <div style="border: 1px solid red; padding: 2px; display: inline-block;"><div style="text-align: right;">00100011 11011101 ----- 10000000</div></div>
■ -35	→	11011100	
		+           1	
		----- 11011101	

- ◆ Anche in questo caso il primo bit indica il segno

0 = positivo, 1 = negativo

- ◆ Rappresentazione univoca dello 0
- ◆ Permette le usuali regole di calcolo per eseguire le operazioni, per esempio effettuare le somme algebriche



# Complemento a due (2)

## ◆ Approccio

- Un numero negativo è rappresentato con la codifica del suo complemento a 2 (positivo)
- in una codifica a K bit:  $C_K(x) = 2^K + x$

## ◆ Esempio ( $K = 8$ , $2^K = 256$ )

- $n = -35 \rightarrow 00100011$
- $C_K(-35) = 256 - 35 = 221 \rightarrow 11011101$

## ◆ Proprietà:

- $C_K(-n) = 2^K - n = (2^K - 1) - n + 1$
- $(2^K - 1) - n = \text{complem. a uno di } n$  (basta invertire le cifre)

## Osservazioni

### ◆ Rappresentazione dello 0

- modulo e segno: rappresentazione ambigua
  - $+0 = 00000000$        $-0 = 10000000$
- complemento a uno: rappresentazione ambigua
  - $+0 = 00000000$        $-0 = 11111111$
- complemento a due: rappresentazione univoca
  - il complemento a due di  $0\dots0$  è ancora  $0\dots0$

### ◆ Intervallo di rappresentazione con K bit

- modulo e segno:       $[ -(2^{K-1}-1), \quad + 2^{K-1}-1 ]$
- complemento a uno:       $[ -(2^{K-1}-1), \quad + 2^{K-1}-1 ]$
- complemento a due:       $[ -2^{K-1}, \quad + 2^{K-1}-1 ]$

# Codifica dei numeri interi

Codice	Nat	MS	C2
0000	0	0	0
0001	1	1	1
0010	2	2	2
0011	3	3	3
0100	4	4	4
0101	5	5	5
0110	6	6	6
0111	7	7	7

Codice	Nat	MS	C2
1000	8	-0	-8
1001	9	-1	-7
1010	10	-2	-6
1011	11	-3	-5
1100	12	-4	-4
1101	13	-5	-3
1110	14	-6	-2
1111	15	-7	-1

# Codifica dei numeri interi

Codice	Nat	MS	C2
0000	0	0	0
0001	1	1	1
0010	2	2	2
0011	3	3	3
0100	4	4	4
0101	5	5	5
0110	6	6	6
0111	7	7	7
1000	8	-0	-8
1001	9	-1	-7
1010	10	-2	-6
1011	11	-3	-5
1100	12	-4	-4
1101	13	-5	-3
1110	14	-6	-2
1111	15	-7	-1

# Codifica dei numeri razionali

## ◆ Fixed point (virgola fissa)

- Un numero razionale è rappresentato come una coppia di numeri interi: la parte intera e la parte decimale
- $12,52 \rightarrow \textcolor{red}{<12; 52>}$

## ◆ Floating point (virgola mobile)

- Un numero razionale è rappresentato come un intero moltiplicato per una opportuna potenza di 10, cioè con una coppia <mantissa, esponente>
- $12,52 = 1252/100 = 1252 * 10^{-2} \rightarrow \textcolor{red}{<1252; -2>}$

# Operazioni algebriche: Errori

## ◆ Problema

- Gli elaboratori elettronici utilizzano un numero fissato di bit per rappresentare un dato tipo di numeri
- un'operazione può produrre un valore non rappresentabile: il numero di bit disponibili è minore di quelli necessari

## ◆ Overflow

- Il valore assoluto del risultato è maggiore della massima quantità rappresentabile
- l'approssimazione con la massima quantità rappresentabile potrebbe implicare un notevole errore

## ◆ Underflow

- il risultato è minore (in valore assoluto) della minima quantità rappresentabile
- nella rappresentazione in virgola mobile, corrisponde ad un overflow dell'esponente
- il risultato è approssimato con **0** (e si segnala la condizione)

# Codifica di caratteri

- ◆ Si associa un codice ad ogni simbolo dell'alfabeto
- ◆ Codifica **ASCII** (American Standard Code for Information Interchange)
  - Caratteri speciali, punteggiatura, a-z, A-Z, 0-9
  - Utilizza 7 bit (128 caratteri)
  - I codici ASCII estesi usano 8 bit (256 caratteri)
- ◆ Codifica **EBCDIC** (Extended Binary-Coded Decimal Interchange Code)
  - Utilizza 8 bit (256 caratteri)
- ◆ Codifica **UNICODE** (per rappresentare l'insieme dei caratteri utilizzati in Europa)
  - Utilizza 16 bit (65536 caratteri)
  - I primi 128 caratteri sono gli stessi di ASCII
  - Gli altri corrispondono ad altri alfabeti (greco, cirillico,...)
  - Non copre i simboli (oltre 200.000) di tutte le lingue!

# Codice ASCII (7 bit)

	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
010	sp	!	"	#	\$	%	&	'	(	)	*	+	,	-	.	/
011	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
100	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
101	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
110	`	a	b	c	d	e	f	g	h	I	j	k	l	m	n	o
111	p	q	r	s	t	u	v	w	x	Y	z	{		}	~	canc



# Codifica di dati multimediali

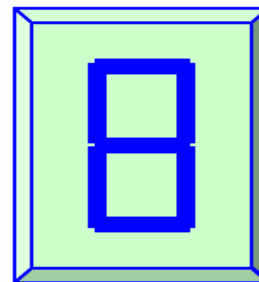
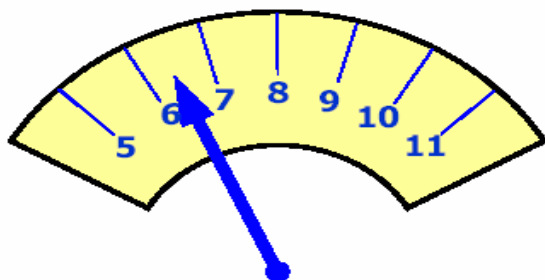
## ◆ Applicazioni multimediali

- elaborano anche tipi di informazione differenti da testi e numeri

## ◆ Esempi di dati multimediali:

- diagrammi
- immagini e filmati
- suoni e sequenze sonore

# Codifica analogica e codifica digitale



## ◆ Codifica analogica

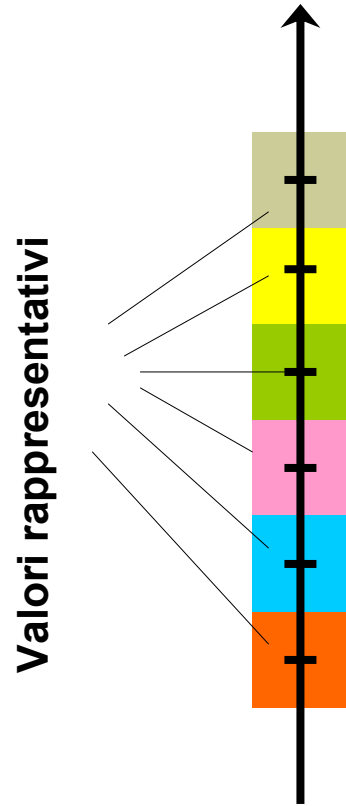
- le configurazioni possono variare in maniera continua su un intervallo prefissato
- esiste una relazione di **analogia** tra l'insieme delle configurazioni e l'insieme delle informazioni

## ◆ Codifica digitale

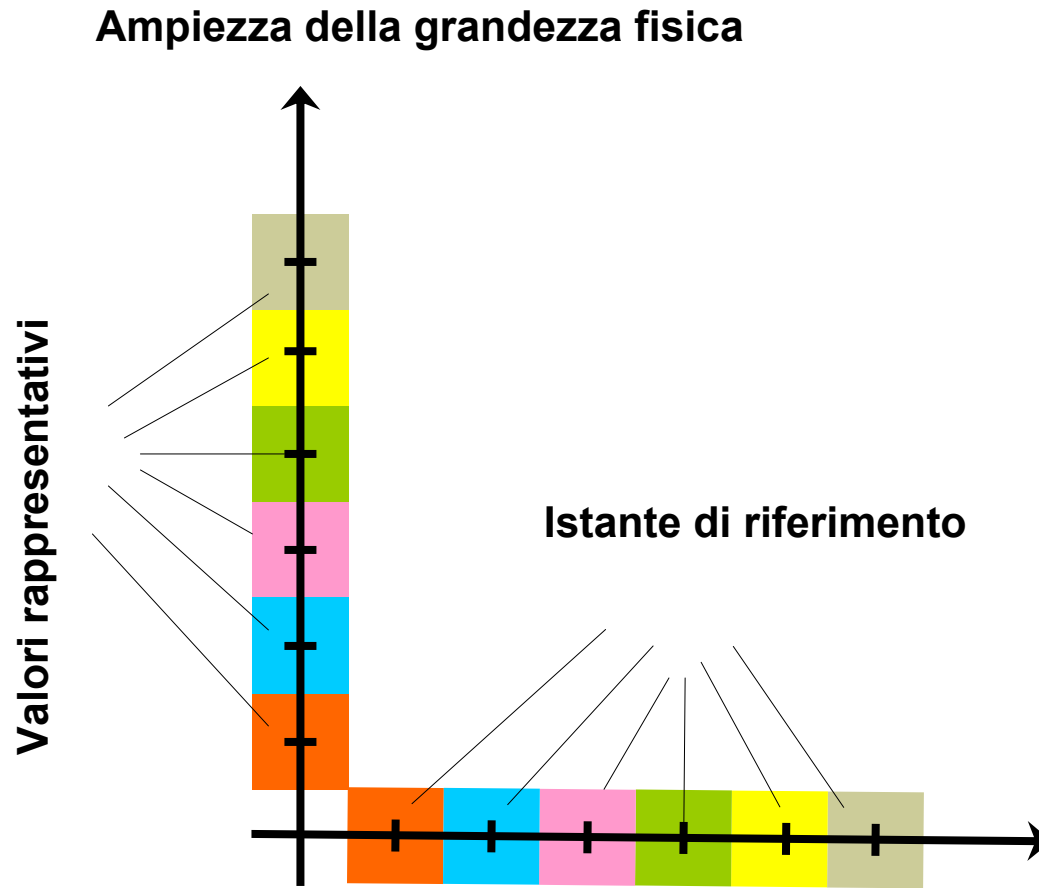
- le entità di informazione vengono codificate mediante configurazioni convenzionali
- è un'approssimazione della realtà (continua): l'errore dipende dalla precisione della codifica (numero di bit)

# Digitalizzazione di grandezze fisiche

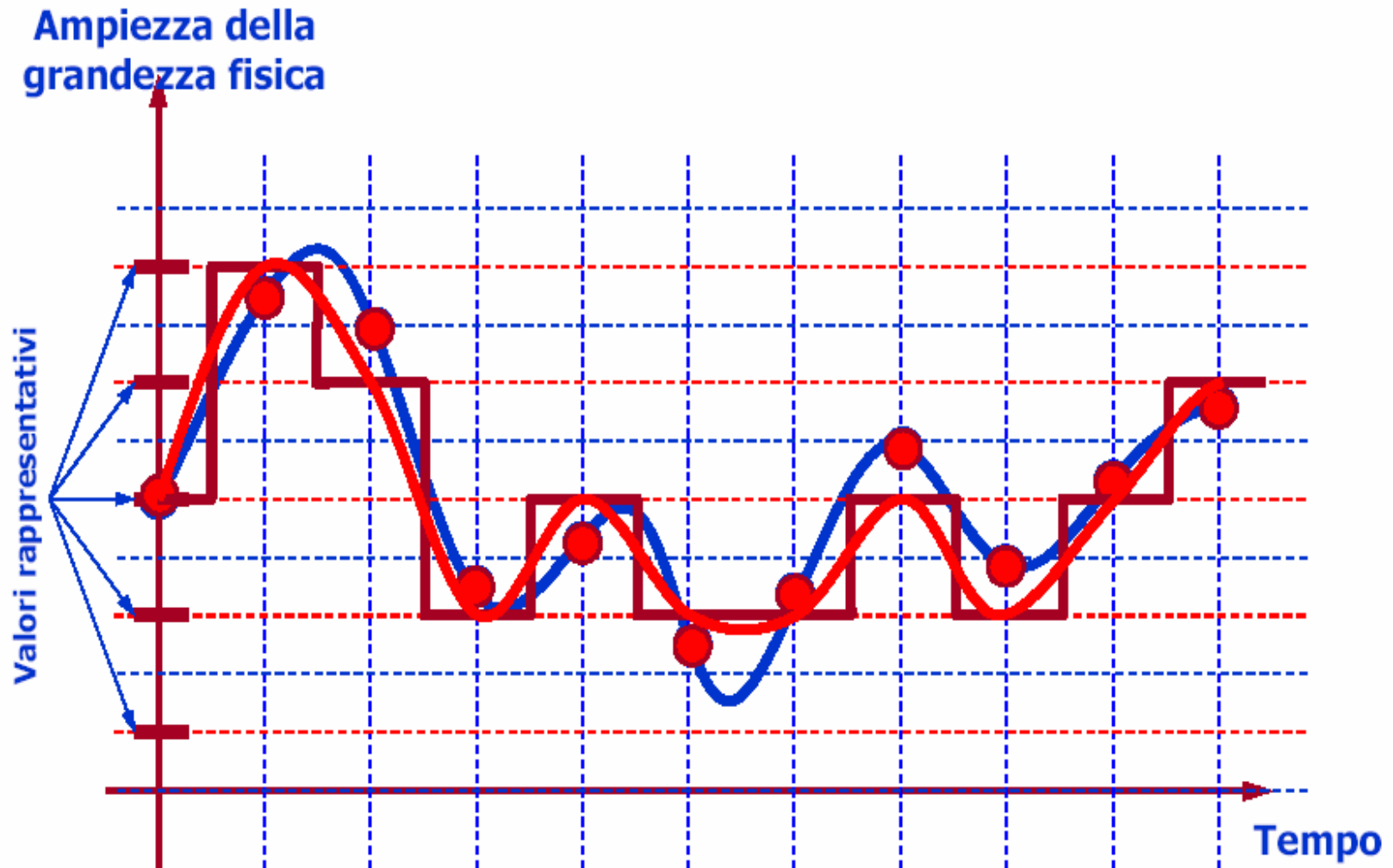
Ampiezza della grandezza fisica



# Digitalizzazione di grandezze fisiche



# Digitalizzazione di grandezze fisiche

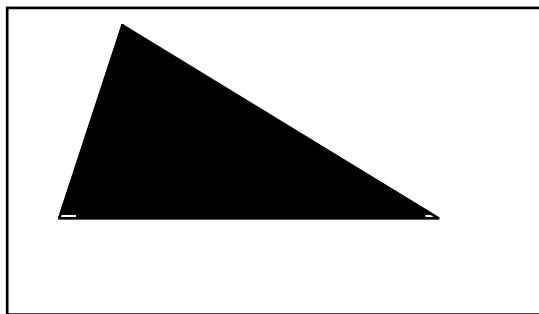


# Codifica di immagini

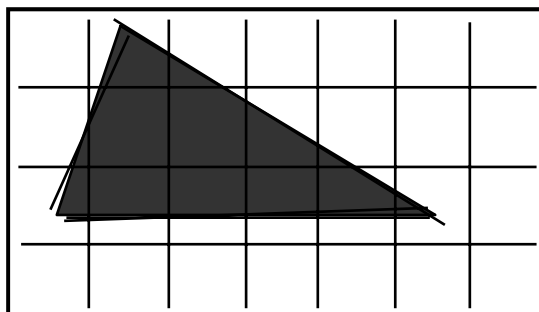
- ◆ Esistono numerose tecniche per la memorizzazione e l'elaborazione di immagini
- ◆ Immagini digitalizzate = sequenze di bit!
  - L'immagine viene *discretizzata*, cioè rappresentata come insieme di pixel
    - ogni pixel ha associato un numero che descrive un particolare colore (o tonalità di grigio)
  - Inoltre si mantengono la dimensione, la risoluzione (punti per pollice), ed il numero di colori utilizzati

# Codifica di immagini

- ◆ Consideriamo un'immagine in bianco e nero, senza ombreggiature o livelli di chiaroscuro



- ◆ Si suddivide l'immagine con una griglia formata da righe orizzontali e verticali a distanza costante



# Codifica di immagini

## ◆ pixel (picture element)

- ogni quadratino derivante dalla suddivisione dell'immagine

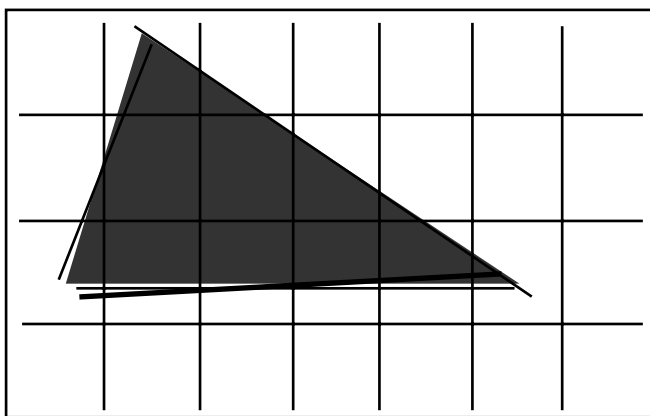
## ◆ Codifica di un pixel:

- il simbolo “0” viene utilizzato per la codifica di un pixel corrispondente ad un quadratino bianco (in cui il bianco è predominante)
- il simbolo “1” viene utilizzato per la codifica di un pixel corrispondente ad un quadratino nero (in cui il nero è predominante)



# Codifica di immagini

- ◆ Poiché una sequenza di bit è lineare, si deve definire una convenzione per ordinare i pixel della griglia
  - Hp: assumiamo che i pixel siano ordinati dal basso verso l'alto e da sinistra verso destra



0 <sub>22</sub>	1 <sub>23</sub>	0 <sub>24</sub>	0 <sub>25</sub>	0 <sub>26</sub>	0 <sub>27</sub>	0 <sub>28</sub>
0 <sub>15</sub>	1 <sub>16</sub>	1 <sub>17</sub>	0 <sub>18</sub>	0 <sub>19</sub>	0 <sub>20</sub>	0 <sub>21</sub>
0 <sub>8</sub>	1 <sub>9</sub>	1 <sub>10</sub>	1 <sub>11</sub>	1 <sub>12</sub>	0 <sub>13</sub>	0 <sub>14</sub>
0 <sub>1</sub>	0 <sub>2</sub>	0 <sub>3</sub>	0 <sub>4</sub>	0 <sub>5</sub>	0 <sub>6</sub>	0 <sub>7</sub>

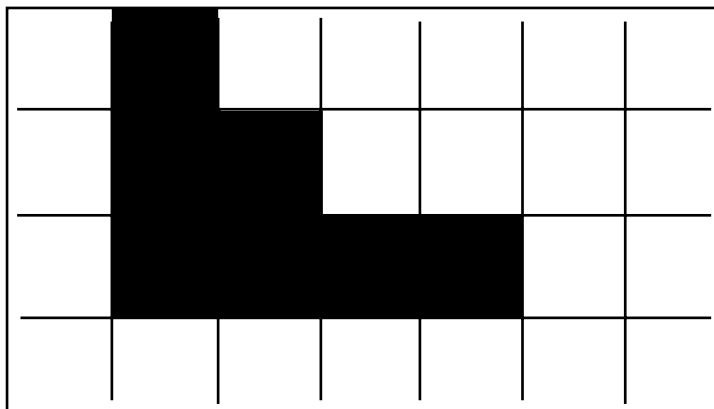
- ◆ La rappresentazione della figura è data dalla stringa:

**0000000 0111100 0110000 0100000**

# Codifica di immagini

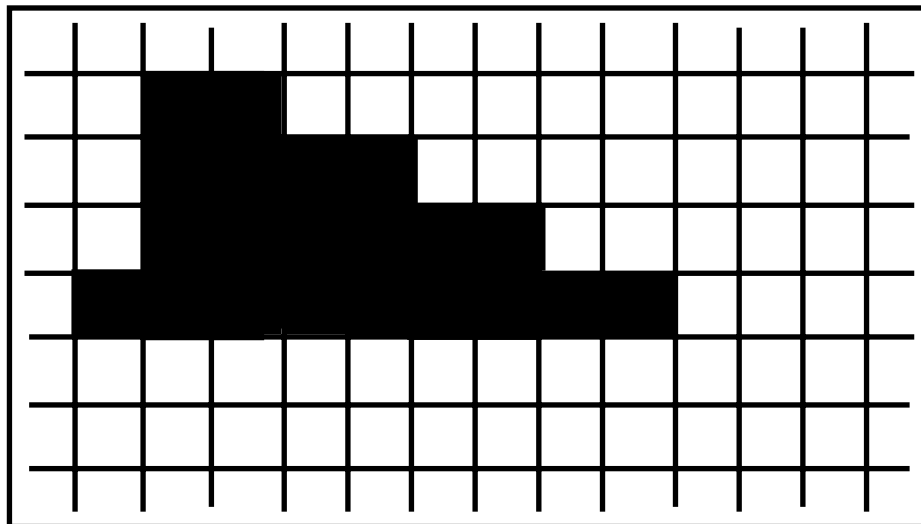
## ◆ Approssimazione:

- nella codifica si ottiene un'approssimazione della figura originaria
  - non sempre il contorno della figura coincide con le linee della griglia
- Riconvertendo in immagine la stringa **0000000011110001100000100000** si ottiene:



# Codifica di immagini

- ◆ La rappresentazione sarà più fedele all'aumentare del numero di pixel
  - ossia al diminuire delle dimensioni dei quadratini della griglia in cui è suddivisa l'immagine



# Immagini con toni di grigio

- ◆ Le consuete immagini “in bianco e nero” hanno delle sfumature (**livelli di intensità di grigio**)
- ◆ Per codificare immagini con sfumature:
  - si fissa un insieme di livelli (*toni*) di grigio, cui si assegna convenzionalmente una rappresentazione binaria
  - per ogni pixel si stabilisce il livello medio di grigio e si memorizza la codifica corrispondente a tale livello
- ◆ Per memorizzare un pixel non basta un solo bit
  - con **4** bit si possono rappresentare  **$2^4=16$**  livelli di grigio
  - con **8** bit ne possiamo distinguere  **$2^8=256$** ,
  - con **K** bit ne possiamo distinguere  **$2^K$**

# Immagini a colori

## ◆ Analogamente possono essere codificate le immagini a colori:

- bisogna definire un insieme di sfumature di colore differenti, codificate mediante una opportuna sequenza di bit

## ◆ codifica *bitmap*

- Indica la rappresentazione di un'immagine mediante la codifica dei pixel

# Immagini a colori

- ◆ Il numero di byte richiesti dipende da
  - **risoluzione**
  - **numero di colori** che ogni pixel può assumere
- ◆ *Es:* per distinguere **256** colori sono necessari 8 bit per la codifica di ciascun pixel
  - la codifica di un'immagine formata da  $640 \times 480$  pixel richiederà 2457600 bit (307200 byte)
- ◆ I monitor tipici utilizzano
  - risoluzione:  $640 \times 480$ ,  $1024 \times 768$ ,  $1280 \times 1024$
  - numero di colori per pixel: da 256 fino a 16 milioni
- ◆ Tecniche di **compressione**
  - riducono notevolmente lo spazio occupato dalle immagini

# Codifica di filmati

- ◆ Immagini in movimento sono memorizzate come sequenze di fotogrammi
- ◆ In genere si tratta di sequenze compresse di immagini
  - ad esempio si possono registrare solo le variazioni tra un fotogramma e l'altro
- ◆ Esistono vari formati (comprendente il sonoro):
  - *mpeg*, *avi* (microsoft), *quicktime* (apple), *mov*

# Codifica di sequenze sonore

- ◆ L'onda sonora (analogica) viene misurata ad intervalli regolari (**campionamento**)
  - Minore è l'intervallo di campionamento e maggiore è la qualità del suono
- ◆ CD musicali:
  - 44000 campionamenti al secondo, 16 bit per campione.
- ◆ Alcuni formati:
  - *.mov, .wav, .mpeg, .avi*
  - **.midi** usato per l'elaborazione della musica al PC



# Compressione dei dati

## ◆ Vantaggio:

- risparmio di risorse per memorizzazione e trasmissione

## ◆ Esempio: codifica a lunghezza variabile

- Alfabeto: {A, C, G, T}
- Una sequenza ATTACCG... di 1 milione caratteri da rappresentare
- Codifica a lung. fissa: memoria richiesta = **2 mil. di bit**  
A=00, C=01, G=10, T=11 → ATTACCG... = 00111100010110...
- Diverse frequenze dei simboli:
  - **f(A)=50%, f(C)=25%, f(G)=12.5%, f(T)=12.5%**
- Si scelgono codici dei simboli con lunghezze (in bit) inversamente proporzionali alle frequenze:
  - **A=0, C=10, G=110, T=111**
  - $(1 \times 50\% + 2 \times 25\% + 2 \times 3 \times 12.5\%) \times 10^6 = \mathbf{1.75 \text{ milioni di bit}}$
- NB: la nuova sequenza binaria deve essere decodificabile!

# Compressione dei dati

## ◆ Lossless

- Senza perdita di informazione
- Utilizzata per programmi, documenti,...

## ◆ Lossy

- Con perdita di informazione
- Rapporto di compressione variabile dall'utente
- Immagini:
  - *GIF, JPEG* (elimina lievi cambiamenti di colore)
- Animazioni:
  - *MPEG* (memorizza solo differenze tra fotogrammi)
- Audio:
  - *MP3* (elimina suoni a basso volume sovrapposti con suoni ad alto volume)

# Compressione JPEG (esempio)

## Codifica Bitmap

- 800 x 600
  - 16,8 mln colori (24 bit)
- dimensione = 1.440.000 byte  
 $\approx 1406$  KB

Fattore qualità 90% ( 253 KB )



Fattore qualità 10% ( 12 KB )



Fattore qualità 1% ( 9 KB )



# Elaborazione di dati binari

# Elaborazione dei dati

## ◆ Dati

- rappresentati mediante **sequenze di bit**

## ◆ Operazioni

- come vengono effettuate su dati codificati in binario?

## ◆ Hardware

- fornisce operazioni primitive utilizzabili per risolvere problemi attraverso programmi
- da cosa è costituito?

# Operazioni algebriche: esempio

Somma di due numeri naturali:

$$5 = 1 \times 4 + 0 \times 2 + 1 \times 1 \quad +$$

$$2 = 0 \times 4 + 1 \times 2 + 0 \times 1 \quad =$$

---

$$7 = 1 \times 4 + 1 \times 2 + 1 \times 1$$

È esprimibile mediante operazioni sui bit!

# Operazioni algebriche: somma e sottrazione su interi

*Somme fra "cifre":*

$$0 + 0 = 0$$

$$1 + 0 = 1$$

$$0 + 1 = 1$$

$$1 + 1 = 10$$

**← riporto**

1	1				
<hr/>					
1	0	1	1	0	+
1	0	1	0	1	=
<hr/>					
1	0	1	0	1	1

**prestito →**

			0
1	1	1	0
<hr/>			
1	1	1	0
0	1	0	1
<hr/>			
1	0	0	1

**Vantaggio della codifica *complemento a 2*:**

La sottrazione equivale alla somma del minuendo col complemento del sottraendo

# Operazioni logiche

## ◆ Valori di verità (boolean):

■ **FALSO** = 0

**VERO** = 1

## ◆ Operatori logici:

■ X **OR** Y = VERO

sse X=VERO oppure Y=VERO

■ X **AND** Y = VERO

sse X=VERO e Y=VERO

■ **NOT** X = VERO

sse X=FALSO

<u>A</u>	<u>not A</u>	<u>A B</u>	<u>A and B</u>	<u>A B</u>	<u>A or B</u>
0	1	0 0	0	0 0	0
1	0	0 1	0	0 1	1
		1 0	0	1 0	1
		1 1	1	1 1	1

■ permettono di esprimere **operazioni bit a bit**



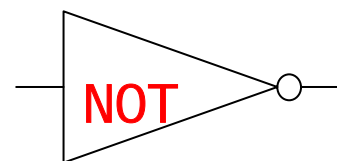
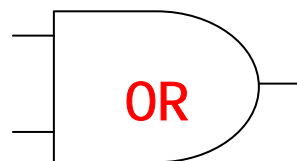
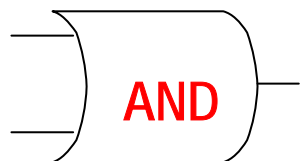
# Circuiti logici

## ◆ Circuito logico

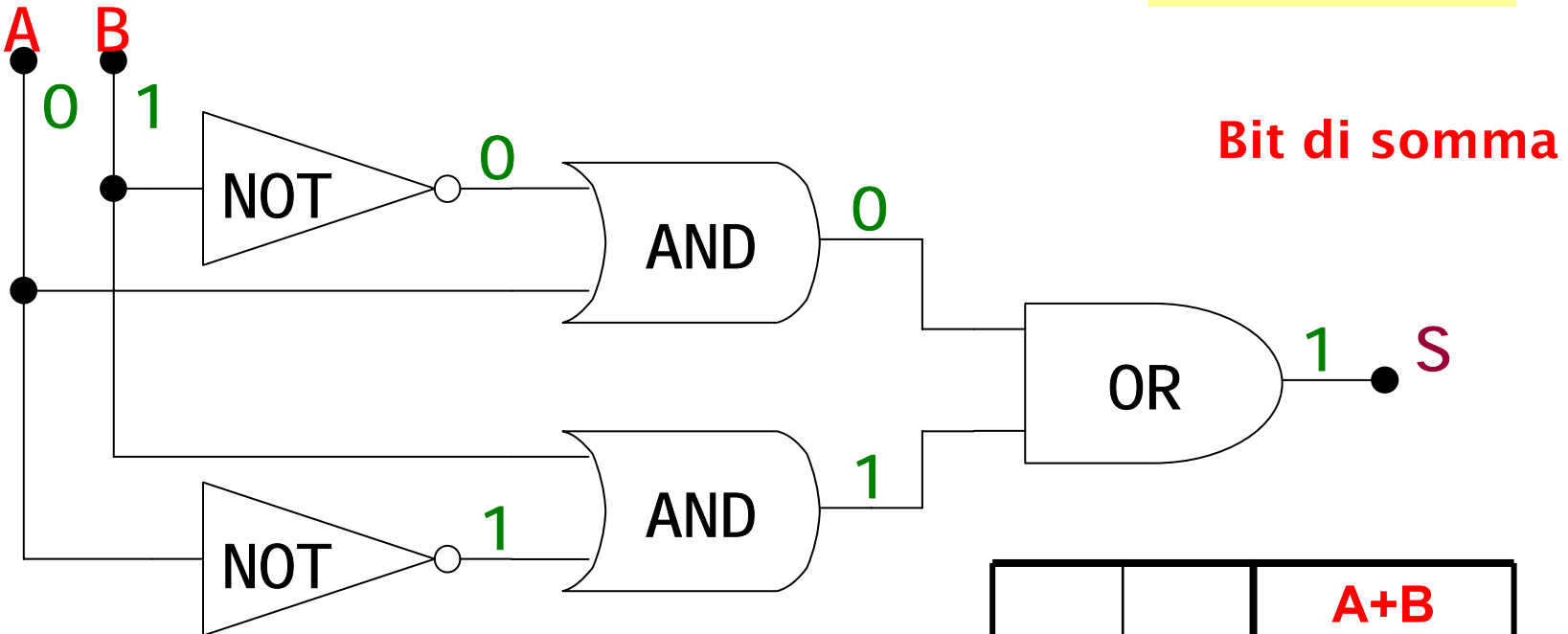
- trasforma ingressi (input) in uscite (output), e definisce una funzione  $f: \text{input} \rightarrow \text{output}$
- input, output = sequenze di bit
- composizione di porte logiche

## ◆ Porte logiche

- circuiti elementari corrispondenti agli operatori logici booleani



# Circuiti logici: somma di due bit

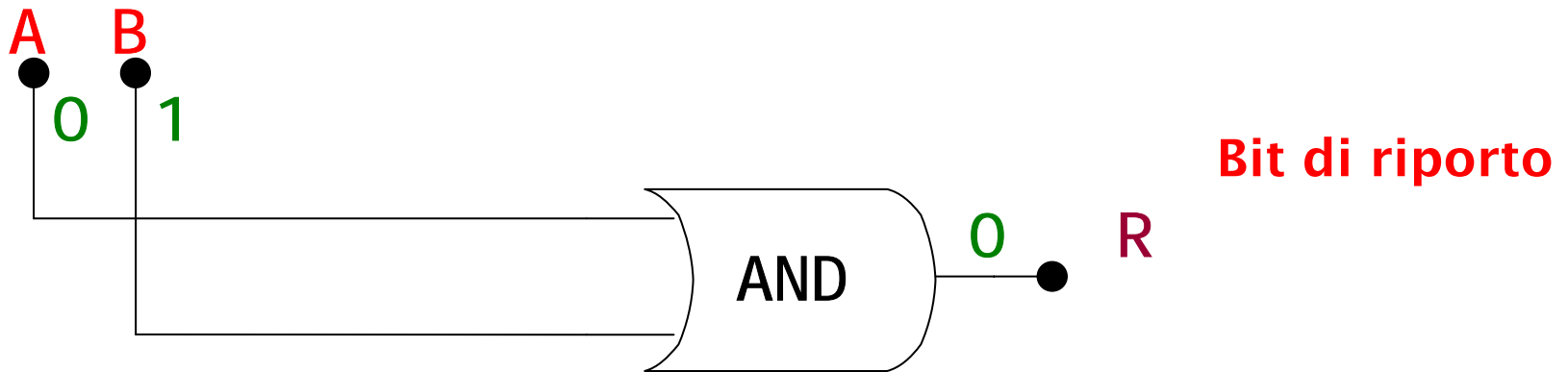


$$S = ((\text{NOT } A) \text{ AND } B) \text{ OR } (A \text{ AND } (\text{NOT } B))$$

$$R = A \text{ AND } B$$

A	B	A+B	
		R	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

# Circuiti logici: somma di due bit



A	B	A+B	
		R	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

$$S = ((\text{NOT } A) \text{ AND } B) \text{ OR } (A \text{ AND } (\text{NOT } B))$$

$$R = A \text{ AND } B$$

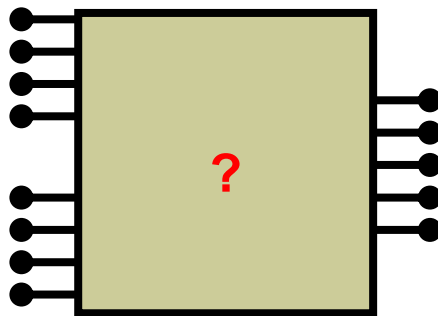
# Circuiti logici

## ◆ Codificano funzioni su dati binari

- $f$ : input  $\rightarrow$  output
- input, output = sequenze di bit

## ◆ Due categorie

- Combinatori
- Sequenziali: il loro output dipende anche dallo stato attuale del circuito (oltre che dall'input)



# Elaborazione dei dati

## ◆ Dati:

sequenze di bit

## ◆ Operazioni:

circuiti logici

## ◆ Hardware:

realizzazione fisica dei circuiti logici