

Potenziamento in Informatica

Riccardo Ortale

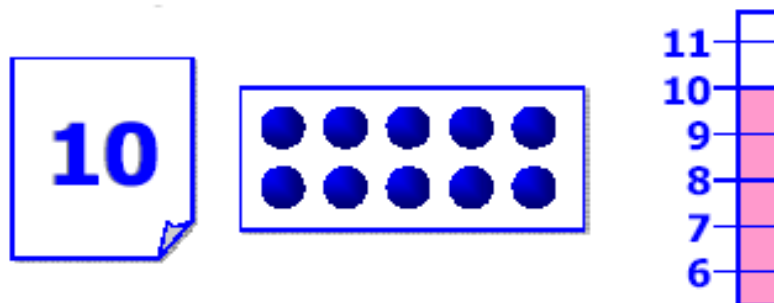
Codifica dell'Informazione

CODIFICA DI DATI E ISTRUZIONI

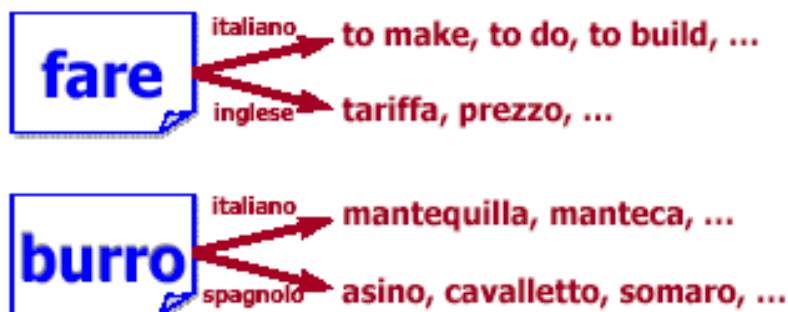
- ◆ Algoritmi
 - Istruzioni che operano su dati
- ◆ Per scrivere un programma è necessario rappresentare dati e istruzioni in un formato tale che l'esecutore automatico sia in grado di
 - Memorizzare istruzioni e dati
 - Manipolare istruzioni e dati
- ◆ Le informazioni gestite dai sistemi di elaborazione devono essere codificate
 - per poter essere memorizzate, elaborate, scambiate,...

CODIFICA DELL'INFORMAZIONE

- ◆ La stessa informazione si può rappresentare in modi differenti



- ◆ Stessa rappresentazione per informazioni differenti



Sistema di codifica (o *codifica*, o *codice*)

- Usa un insieme di simboli di base (**alfabeto**)
- I simboli dell'alfabeto possono essere combinati ottenendo differenti **configurazioni** (o *codici*, o *stati*), distinguibili l'una dall'altra
- Associa ogni configurazione ad una particolare entità di informazione (la configurazione diventa un modo per rappresentarla)

SISTEMI DI CODIFICA: NUMERI INTERI (DECIMALI)

- ◆ Alfabeto
 - Cifre “0”, “1”, “2”, ..., “9”
 - separatore decimale (“,”)
 - separatore delle migliaia (“.”)
 - Segni positivo (“+”) e negativo (“-”)
- ◆ Regole di composizione (**sintassi**)
 - Definiscono le combinazioni ben formate
 - 12.318,43
 - 12,318.43
- ◆ Codice (**semantica**)
 - Associano ad ogni configurazione un’entità di informazione
 - $12.318,43 = 1 \times 10^4 + 2 \times 10^3 + 3 \times 10^2 + 1 \times 10 + 1 \times 10^0 + 4 \times 10^{-1} + 3 \times 10^{-2}$
- ◆ Lo stesso alfabeto può essere usato per codici diversi
 - $123,456 = 1 \times 10^2 + 2 \times 10 + 3 \times 10^0 + 4 \times 10^{-1} + 5 \times 10^{-2} + 6 \times 10^{-3}$ [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]

CODIFICA BINARIA

- ◆ Codifica binaria: usa un alfabeto di **2** simboli
- ◆ Utilizzata nei sistemi informatici
 - Si utilizza una grandezza fisica (luminosità, tensione elettrica, la corrente elettrica), per rappresentare informazione
 - Si divide in due intervalli l'insieme dei valori che la grandezza può assumere: ogni intervallo corrisponde ad un simbolo
- ◆ Solo 2 simboli al fine di ridurre la probabilità di errore
 - Tanto più simboli si devono distinguere e tanto meno la rivelazione sarà affidabile (gli intervalli della grandezza fisica saranno meno ampi)

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, si/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

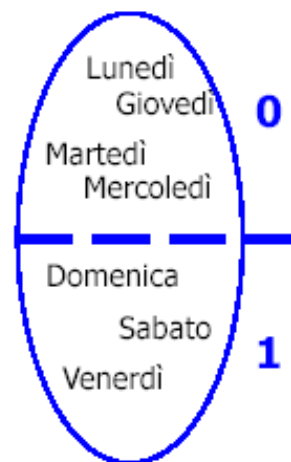
◆ Quanti bit sono necessari per codificare N oggetti?

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

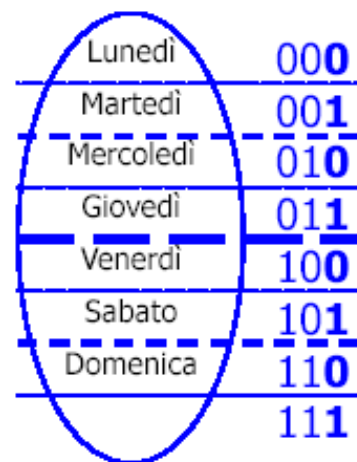
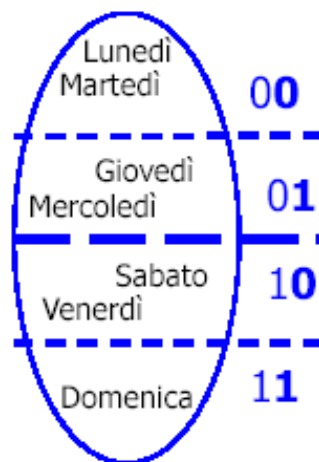
I GIORNI DELLA SETTIMANA IN BINARIO



1 bit
2 "gruppi"



2 bit
4 "gruppi"



3 bit
8 "gruppi"

CODIFICA BINARIA: UNITA' DERIVATE

- ◆ **Byte** = 8 bit
 - può rappresentare $2^8 = 256$ stati
- ◆ **KiloByte** (KB) = 2^{10} byte = 1.024 byte $\cong 10^3$ byte
- ◆ **MegaByte** (MB) = 2^{20} byte = 1.048.576 byte $\cong 10^6$ byte
- ◆ **GigaByte** (GB) = 2^{30} byte = 1.073.741.824 byte $\cong 10^9$ byte
- ◆ **TeraByte** (TB) = 2^{40} byte = 1.099.511.627.776 byte $\cong 10^{12}$ byte

CODIFICA DEI NUMERI NATURALI

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

$$a_{n-1} a_{n-2} a_{n-3} \dots a_1 a_0$$

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:* 13 può essere espresso in funzione delle potenze di 2 come:

$$13 = 1 \times 8 + 1 \times 4 + 0 \times 2 + 1 \times 1$$


cioè può essere rappresentato dalla sequenza di bit

1 1 0 1

CONVERSIONE DECIMALE-BINARIO

- ◆ Si calcolano i resti della divisione per 2


$18 : 2 = 9$	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 INTERI

◆ Modulo e segno

- Il bit più a sinistra rappresenta il segno del numero (0 = '+', 1 = '-')
- Esempio: $+7 = 0111$, $-7 = 1111$
- Valori da $-2^{k-1}+1$ a $2^{k-1}-1$
- Con $k=4$ bit: da $-2^3+1=-7$ a $2^3-1=+7$
- Attenzione ci sono due zeri!
 $+0=0000$ e $-0=1000$

CODIFICA DEI NUMERI INTERI

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

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

Complemento a 2

- si rappresentano i valori da -2^{k-1} a $2^{k-1}-1$
 - con 4 bit i valori vanno da -8 a $+7$
 - con 8 bit i valori vanno da -128 a $+127$
- Attenzione: c'è una sola rappresentazione dello 0
 - con 4 bit è $+0_{\text{dieci}} = 0000_{\text{c2}}$ mentre $1000_{\text{c2}} = -8_{\text{dieci}}$

CODIFICA DEI NUMERI INTERI

◆ Complemento a 2

Metodi per calcolare la rappresentazione di $-X$ a partire da quella di X

- Effettuare il complemento di ogni bit di X e aggiungere poi 1
 - rappresentazione di $+6_{\text{dieci}} = 0110_{\text{C2}}$ (NB ci vogliono 4 bit!!)
 - complemento di tutti i bit $\Rightarrow 1001_{\text{C2}}$ (corrisponderebbe a -7_{dieci})
 - aggiungere 1 $\Rightarrow 1010_{\text{C2}}$ (che corrisponde a -6_{dieci})

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

◆ Complemento a 2

- i numeri positivi iniziano con **0**, quelli negativi con **1**
- data la rappresentazione di un numero su **k** bit, la rappresentazione dello stesso numero su **k+1** bit si ottiene aggiungendo (a sinistra) un bit uguale al primo (**estensione del "segno"**)
 - Rappresentazione di -6 su 4 bit = 1010
 - Rappresentazione di -6 su 5 bit = 11010
 - Rappresentazione di -6 su 8 bit = 11111010
- la sottrazione si effettua come somma algebrica
 - $4 - 6 = +4 + (-6) = 0100 + 1010 = 1110 = -2$
 - $9 - 6 = +9 + (-6) = 01001 + 11010 = [1]00011 = +3$

CODIFICA DI CARATTERI

- ◆ Associando un simbolo dell'alfabeto ad ogni numero possiamo codificare tutte le lettere
- ◆ 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**
 - Utilizza 16 bit (65536 caratteri)
 - I primi 128 caratteri di UNICODE sono gli stessi di ASCII
 - I successivi corrispondono ad altri alfabeti (greco, cirillico, ebraico, ...)
 - Non riesce a coprire i simboli (oltre 200.000) di tutte le lingue!

ASCII SU 7 BIT

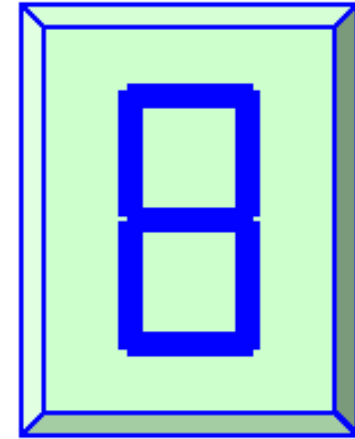
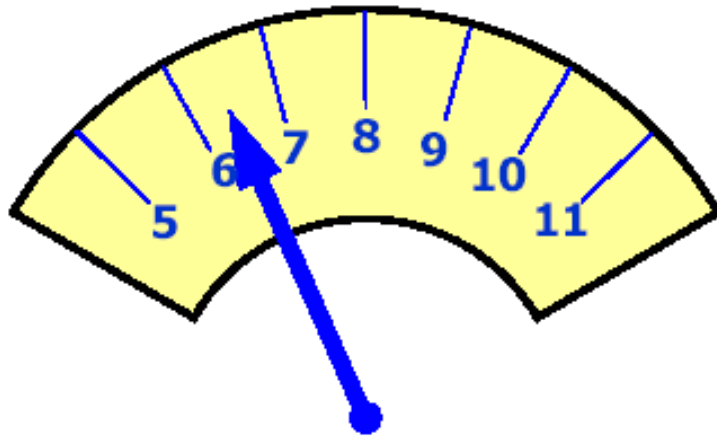
	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
010	sp	!	"	#	\$	%	&	'	()	*	+	,	-	.	/
011	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
100	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
101	P	Q	R	S	T	U	V	W	X	Y	Z	[\]	^	_
110	`	a	b	c	d	e	f	g	h	I	j	k	l	m	n	o
111	p	q	r	s	t	u	v	w	x	Y	z	{		}	~	canc

“Ciao” = 1000011 1101001 1100001 1101111

“24” = 0110001 0110011

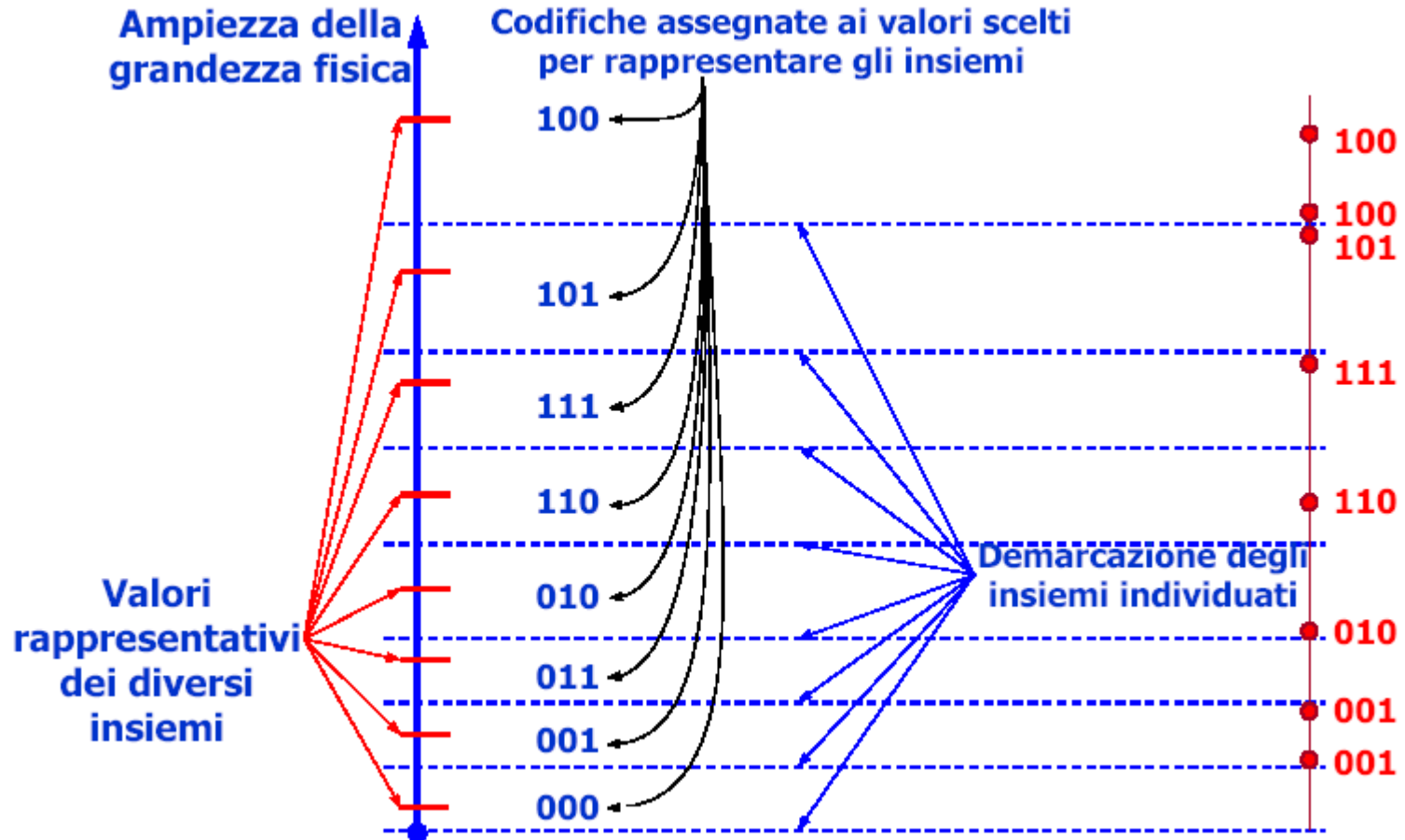
“3 kg” = 0110011 0100000 1101011 1100111

CODIFICA ANALOGICA E CODIFICA DIGITALE



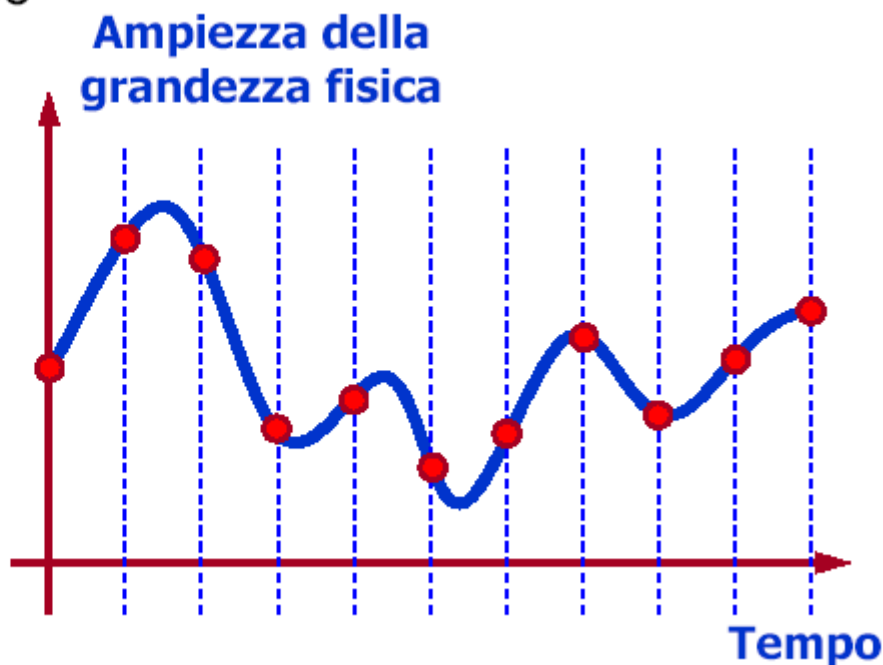
- ◆ **Codifica analogica:** le configurazioni possono variare in maniera continua su un insieme prefissato; esiste una relazione di **analogia** tra l'insieme delle configurazioni e l'insieme delle informazioni
- ◆ **Codifica digitale:** le entità di informazione vengono codificate mediante configurazioni convenzionali

QUANTIZZAZIONE

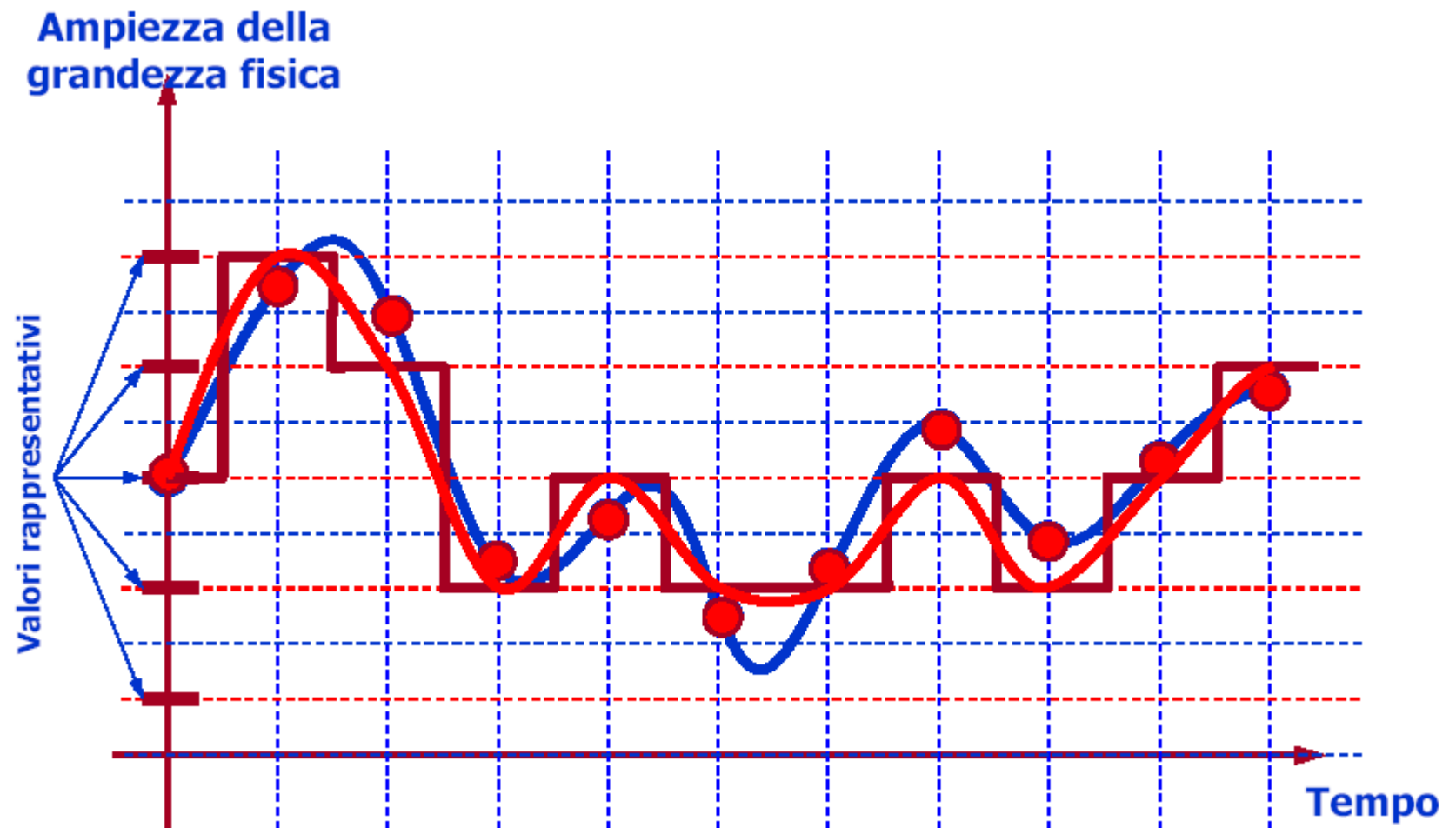


CAMPIONAMENTO

- La grandezza varia nel tempo e non può essere rappresentata da un solo valore.
- I valori di riferimento debbono essere rilevati in diversi istanti di tempo (frequenza di campionamento).
- La quantizzazione deve poi essere ripetuta per ogni valore campionato.



DIGITALIZZAZIONE



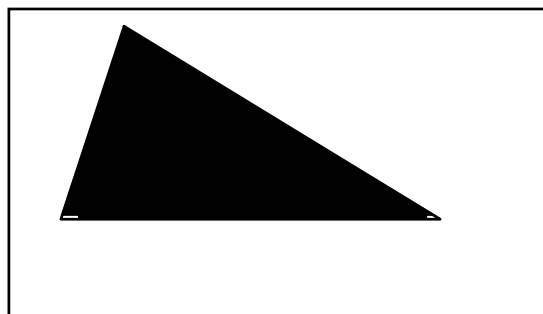
VANTAGGI DELLA CODIFICA DIGITALE

- **Rumore:** effetto dell'ambiente sul supporto.
- Quanto un supporto è "immune" al rumore?
 - Codifica analogica: ogni configurazione è lecita dal punto di vista informativo e quindi risulta impossibile distinguere il rumore dal segnale.
 - Codifica digitale: un valore binario è associato a un insieme di configurazioni valide quindi si può
 - riconoscere il rumore che porta in configurazioni non lecite
 - trascurare il rumore che non fa uscire il segnale dall'insieme associato alla stessa configurazione

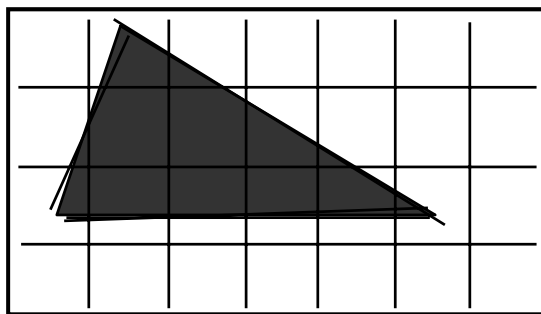


CODIFICA DI IMMAGINI

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



- ◆ Suddividiamo l'immagine mediante una griglia formata da righe orizzontali e verticali a distanza costante

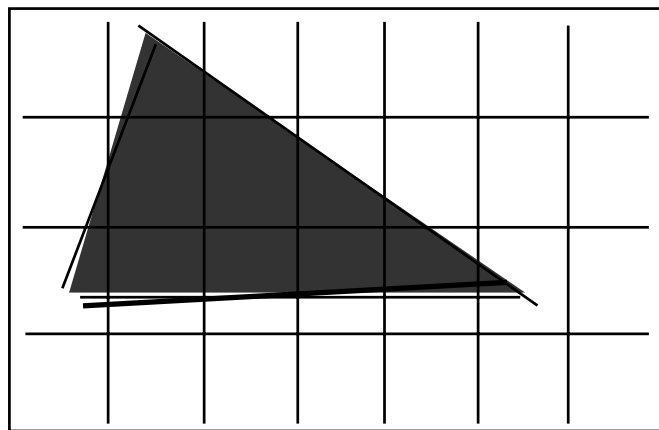


CODIFICA DI IMMAGINI

- ◆ Ogni quadratino derivante da tale suddivisione prende il nome di **pixel** (picture element) e può essere codificato in binario secondo la seguente convenzione:
 - 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
- ◆ **ipotesi**: assumiamo che i pixel siano ordinati dal basso verso l'alto e da sinistra verso destra



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

La rappresentazione della figura è data dalla stringa binaria

000000 0111100 0110000 0100000

CODIFICA DI IMMAGINI A COLORI

- ◆ Il numero di byte richiesti dipende dalla **risoluzione** e dal **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 640×480 pixel richiederà 2457600 bit (307200 byte)
- ◆ I monitor tipici utilizzano
 - risoluzione: 640×480 , 1024×768 , 1280×1024
 - numero di colori per pixel: da 256 fino a 16 milioni

COMPRESSIONE DEI DATI

◆ Lossless

- Senza perdita di informazione
- Programmi, documenti

◆ Lossy

- Con perdita di informazione
- Rapporto di compressione variabile dall'utente
- Immagini: GIF, JPEG (elimina lievi cambiamenti di colore)

COMPRESSIONE DEI DATI

◆ Esempio:

- {A, C, G, T}
- A=00, C=01, G=10, T=11
- ATTACCGAAACTTCTCTCGGGTG... → 1 milione caratteri = 2 milioni di bit
- 00111100010110...
- $\text{fr}(A)=50\%$, $\text{fr}(C)=25\%$, $\text{fr}(G)=12.5\%$, $\text{fr}(T)=12.5\%$
- A=0, C=10, G=110, T=111
- 011111101010110...
- $1 \times 50\% + 2 \times 25\% + 2 \times 3 \times 12.5\% \times 10^6 = 1.75$ milioni di bit → risparmio di 250000 bit!
- La nuova successione binaria deve essere decodificabile

JPEG: Fattore qualità 90/100 (253KB)

800x600

16, 8ml n
col ori
24 bi t

Bi tmap:
1440000
byte

JPEG:
258971
byte



JPEG: Fattore qualità 50/100 (30KB)



JPEG: Fattore qualità 25/100 (20KB)



JPEG: Fattore qualità 10/100 (12KB)



JPEG: Fattore qualità 1/100 (9KB)



CODIFICA DELLE ISTRUZIONI

Istruzioni aritmetico-logiche	
Codice	Istruzione
01100000	ADD
01100100	SUB
01111110	AND
...	...

Codice Operativo			
Codice Operativo	Indirizzo 1		
Codice Operativo	Indirizzo 1	Indirizzo 2	
Codice Operativo	Indirizzo 1	Indirizzo 2	Indirizzo 3

◆ Linguaggio macchina

- A ogni istruzione è assegnato un codice univoco, detto **codice operativo**
- E' necessario specificare dove leggere gli **operandi** (dati) dell'istruzione e dove scrivere il risultato
- Il numero di dati che ogni istruzione manipola è variabile in funzione dell'istruzione stessa



FINE

