

***Modulo 2:***  
**RAPPRESENTAZIONE DEI DATI**  
**I sistemi di numerazione**

*Ing. Maria Grazia Celentano*

# SISTEMI DI NUMERAZIONE

- Il nostro sistema di numerazione è il sistema decimale.  
Tutto ha origine dal fatto che abbiamo 10 dita, quindi, all'inizio, abbiamo imparato a contare fino a 10.  
Se fossimo nati ragni avremmo contato fino ad otto ed useremmo un sistema di numerazione ottale, se fossimo nati gatti avremmo contato fino a 4 e useremmo un sistema quattrale, millepiedi fino a mille, ecc.

## Come conta un calcolatore?

- Un computer è un'apparecchiatura elettronica quindi capisce solo due stati: passa corrente, non passa corrente, o meglio, acceso/spento. È come se avesse solo due dita.
- Per questo motivo la codifica dei numeri utilizzata in informatica è la codifica binaria.
- Quindi non 10 cifre, da 0 a 9, come noi umani. Solo due: 0 e 1.



# SISTEMI DI NUMERAZIONE

Come si fa a scrivere un numero in codice binario?

Come si può convertire un numero da decimale a binario e viceversa?

- Ragioniamo su come sono scritti i numeri che utilizziamo.
- Ad es., consideriamo i due numeri 324 e 432. Sono due numeri diversi anche se sono formati dalle stesse cifre. Sono diversi perché la posizione delle cifre è diversa. Infatti il valore dei numeri è diversa a seconda della posizione delle sue cifre.
- Si chiama **notazione posizionale**.
- Alle scuole elementari, abbiamo imparato che nel numero 324, 3 è la cifra delle centinaia, 2 la cifra delle decine, 1 quella delle unità. Ogni cifra ha un peso diverso a seconda della posizione che occupa.

Riassumendo:

- Abbiamo una serie di dieci simboli: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9.
- Il loro significato dipende dalla posizione che assumono nella “parola” che codifica un numero.
- Ad esempio:

$$1846 = 1 \times 1000 + 8 \times 100 + 4 \times 10 + 6 \times 1$$

In particolare, scritto con le potenze del 10:

$$1846 = 1 \times 10^3 + 8 \times 10^2 + 4 \times 10^1 + 6 \times 10^0$$

# BIT

**Un'informazione può essere rappresentata  
con una successione di due simboli 0 e 1  
detti BIT**

**Il BIT** (dall'inglese **B**inary **digi**T) è  
l'**Unità elementare di informazione**

Esempio:  $10011_2 = 19$

# BIT

- In termini pratici il *bit* viene realizzato utilizzando le proprietà dell'energia elettrica (assenza di carica o presenza di carica).
- Da un punto di vista prettamente fisico il *bit* è un sistema a 2 stati:
- può infatti essere indotto in uno dei due stati distinti rappresentanti 2 valori logici
  - no o si
  - falso o vero
  - semplicemente 0 o 1

# Rappresentazione binaria dell'informazione

- Con un unico *bit* possono essere rappresentate 2 differenti informazioni, ad esempio del tipo: si/no, on/off, 0/1
- Mettendo insieme più *bit* è possibile rappresentare un numero, anche molto elevato, di informazioni.
  - con 2 *bit* possono essere rappresentate 4 differenti informazioni: 00, 01, 10, 11
  - con 3 *bit* è possibile rappresentare 8 differenti informazioni: 000, 001, 010, 011, 100, 101, 110, 111
  - con 4 *bit* è possibile rappresentare 16 differenti informazioni: 0000, 0001, 0010, ....., 1110, 1111
  - e così via...

# Rappresentazione binaria dell'informazione

- In generale con  **$n$  bit** è possibile rappresentare  **$2^n$  differenti informazioni**
- Negli esempi precedenti:
  - con 2 bit ->  $2^2=4$  informazioni
  - con 3 bit ->  $2^3=8$  informazioni
  - con 4 bit ->  $2^4=16$  informazioni.

I PC operano su sequenze di ben 32 bit o 64bit.

Questo vuol dire che sono in grado di processare blocchi di informazione ognuno dei quali può codificare ben

$2^{32} = 4'294'967'295$  informazioni differenti.

# Rappresentazione binaria dell'informazione

- Viceversa, per rappresentare  $m$  differenti informazioni occorrono  $n$  bit, tali che  $2^n \geq m$ .

Ad esempio:

- per rappresentare 57 informazioni diverse sono necessari almeno 6 bit. In base alla formula precedente  $2^6 = 64 > 57$
- Infatti, le possibili combinazioni di 6 bit sono 64:  
000000, 000001, 000010, ..., 111110, 111111

# BYTE

**Insieme di 8 cifre binarie  
viene chiamato BYTE**

- **(dall'inglese BinarY ocTEt)**

**con un byte si possono rappresentare 256 valori  
da 0 a 255**

# BYTE

- Pertanto, con un *byte* è possibile rappresentare  $2^8 = 256$  differenti informazioni.
- Il ***byte*** è utilizzato come **unità di misura** per indicare le dimensioni della memoria, la velocità di trasmissione, la potenza di un elaboratore.
- **Usando sequenze di *byte* (e quindi di *bit*) si possono rappresentare caratteri, numeri, immagini, suoni.**

## Multipli del byte

- **Kilobyte (kB) = 1.024 byte**
- **Megabyte (MB) = 1.048.576 byte**
- **Gigabyte (GB) = 1.073.741.824 byte**
- **Tera byte (TB) = 1.024 Giga byte**

## Multipli del byte

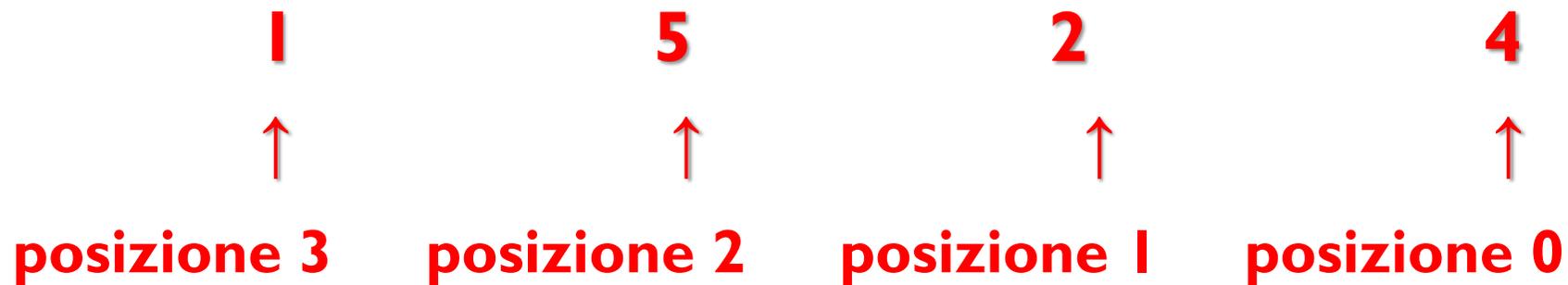
- **Kilobyte (kB)** = 1.024 byte
- **Megabyte (MB)** = 1.024 \* 1 kB =  
= 1.024 \* (1.024 byte) = 1.048.576 byte
- **Gigabyte (GB)** = 1.024 \* 1 MB =  
= 1.024 \* (1.024 \* 1 kB) =  
= 1.024 \* 1.024 \* (1.024 byte) =  
= 1.073.741.824 byte

# SISTEMI DI NUMERAZIONE

- Il **sistema decimale** è quello utilizzato comunemente per la rappresentazione dei numeri. Esso è basato su 10 differenti cifre, dalla cifra 0, alla cifra 9, ed è di tipo **posizionale**.
- Il termine posizionale deriva dal fatto che, a seconda della posizione che una cifra occupa nella rappresentazione di un numero, essa è caratterizzata da un peso.

# SISTEMI DI NUMERAZIONE

Ad esempio, si consideri il numero 1524;  
la posizione delle cifre obbedisce al seguente schema:



- La cifra 4, nella posizione 0, è quella meno significativa poiché rappresenta le **unità**;
- La cifra 2, nella posizione 1, rappresenta le **decine**;
- La cifra 5, nella posizione 2, rappresenta le **centinaia**;
- La cifra 1, nella posizione 3, rappresenta le **migliaia**.
- Le cifre più significative sono quelle nelle posizioni più alte (a sinistra), mentre quelle meno significative sono quelle nelle posizioni più basse (a destra).

# SISTEMI DI NUMERAZIONE

Il precedente numero, 1524, può essere espresso nel seguente modo:

$$1 \cdot 10^3 + 5 \cdot 10^2 + 2 \cdot 10^1 + 4 \cdot 10^0 = 1000 + 500 + 20 + 4 = 1524$$

Si noti che il numero più grande che è possibile rappresentare con  $n$  cifre in notazione decimale è:

**$10^n - 1$**

Posizione	Peso	Potenza di 10
0	Unità	$10^0=1$
1	Decine	$10^1=10$
2	Centinaia	$10^2=100$
3	Migliaia	$10^3=1000$
4	Decine di migliaia	$10^4=10000$
...	...	...

# SISTEMI DI NUMERAZIONE

- Anche il **sistema binario**, basato sulle cifre 0 e 1, è di tipo posizionale (cioè a ogni cifra è associato un peso in base alla sua posizione).
- Le posizioni sono equivalenti a quelle della rappresentazione decimale. Se si considera il numero binario 10100101, si ha:

1	0	1	0	0	1	0	1
↑	↑	↑	↑	↑	↑	↑	↑
pos. 7	pos. 6	pos. 5	pos. 4	pos. 3	pos. 2	pos. 1	pos. 0

# SISTEMI DI NUMERAZIONE

- Il peso relativo alla posizione è definito di seguito:

Posizione (peso)	Potenza di 2
0	$2^0=1$
1	$2^1=2$
2	$2^2=4$
3	$2^3=8$
4	$2^4=16$
...	...
8	$2^8=256$
...	...

## Sistemi in base B

- La base definisce il numero di cifre diverse nel sistema di numerazione
- La cifra di minor valore è sempre lo 0
- Le altre sono, nell'ordine  $1, 2, \dots, B-1$

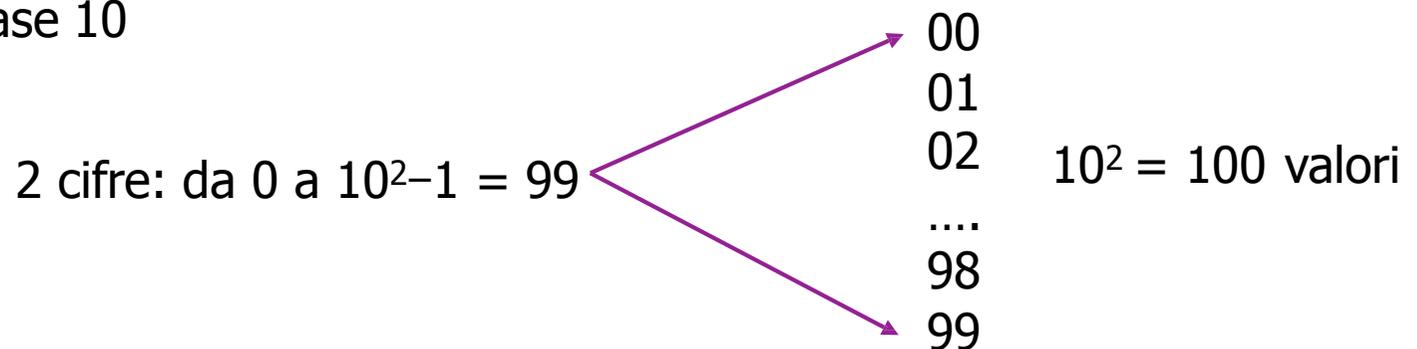
Un numero **intero** N si rappresenta con la scrittura  $(c_n c_{n-1} \dots c_2 c_1 c_0)_B$

$$N = c_n B^n + c_{n-1} B^{n-1} + \dots + c_2 B^2 + c_1 B^1 + c_0 B^0$$

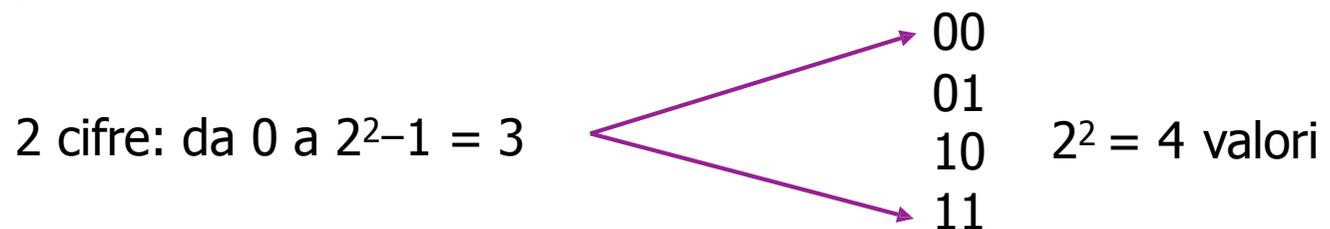
# Numeri interi senza segno

Con  $n$  cifre in base  $B$  si rappresentano tutti i numeri interi positivi da 0 a  $B^n - 1$  ( $B^n$  numeri distinti)

Esempio: base 10



Esempio: base 2



# Il sistema binario (B=2)

La base 2 è la più piccola per un sistema di numerazione

Cifre: 0 1 - bit (binary digit)

Esempi:

$$(101101)_2 = \boxed{1 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0} = 32 + 0 + 8 + 4 + 0 + 1 = (45)_{10}$$

Forma  
polinomiale

# Dal bit al byte

- Un **byte** è un insieme di 8 bit (un numero binario a 8 cifre)

$b_7b_6b_5b_4b_3b_2b_1b_0$

- Con un byte si rappresentano i numeri interi fra 0 e  $2^8-1 = 255$

00000000

00000001

00000010

00000011

.....

11111110

11111111

$2^8 = 256$  valori distinti

- È l'elemento base con cui si rappresentano i dati nei calcolatori
- Si utilizzano sempre dimensioni multiple (di potenze del 2) del byte: 2 byte (16 bit), 4 byte (32 bit), 8 byte (64 bit)...

# Dal byte al kilobyte

- Potenze del 2  $2^4 = 2*2*2*2 = 16$   
 $2^8 = 256$   
 $2^{16} = 65536$

$$\begin{aligned} 2^{10} &= 1024 && \text{(K=Kilo)} \\ 2^{20} &= 1048576 && \text{(M=Mega)} \\ 2^{30} &= 1073741824 && \text{(G=Giga)} \end{aligned}$$

- Cosa sono KB (Kilobyte), MB (Megabyte), GB (Gigabyte)?

$$\begin{aligned} 1 \text{ KB} &= 2^{10} \text{ byte} = 1024 \text{ byte} \\ 1 \text{ MB} &= 2^{20} \text{ byte} = 1048576 \text{ byte} \\ 1 \text{ GB} &= 2^{30} \text{ byte} = 1073741824 \text{ byte} \\ 1 \text{ TB} &= 2^{40} \text{ byte} = 1099511627776 \text{ byte (Terabyte)} \end{aligned}$$

## Conversione binario → decimale

- Basta moltiplicare ogni bit per il suo peso e sommare il tutto.
- Ad esempio, il numero decimale corrispondente al numero binario 10100101 può essere espresso nel seguente modo:

$$\begin{aligned} 1 \cdot 2^7 + 0 \cdot 2^6 + 1 \cdot 2^5 + 0 \cdot 2^4 + 0 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 &= \\ &= 128 + 0 + 32 + 0 + 0 + 4 + 0 + 1 = 165 \end{aligned}$$

## Conversione binario → decimale

- Convertire in decimale i seguenti numeri binari:

1) 10

2) 10001

3) 1001010101

- **Soluzione**

1)  $10_2 = 1 \cdot 2^1 + 0 \cdot 2^0 = 2_{10}$

2)  $10001_2 = 1 \cdot 2^4 + 0 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = 17_{10}$

3)  $1001010101_2 =$

$$= 1 \cdot 2^9 + 0 \cdot 2^8 + 0 \cdot 2^7 + 1 \cdot 2^6 + 0 \cdot 2^5 + 1 \cdot 2^4 + 0 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 =$$
$$= 597_{10}$$

## Conversione decimale → binario

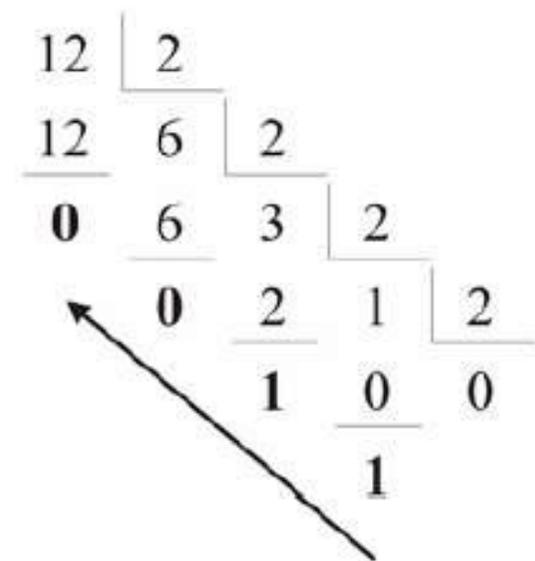
- Si utilizza l'algoritmo della divisione.
- Si divide il numero decimale per 2 ripetutamente finché il risultato non è 0 e si prendono i resti delle divisioni in ordine inverso.

**Esempio:** Convertire il numero decimale 12 in binario.

**Soluzione:**  $12_{10} = 1100_2$ .

**Controprova:**

$$\begin{aligned} 1100_2 &= 1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 0 \cdot 2^0 = \\ &= 8 + 4 = 12 \end{aligned}$$



# Conversione decimale → binario Esercizio

Convertire in binario i seguenti numeri decimali:

1) 128

2) 67217

3) 100169

# La Codifica dei Caratteri

- A differenza di una persona, un sistema elettronico distingue solo due diversi stati fisici: acceso o spento, tensione alta o tensione bassa, passaggio di corrente o assenza di corrente, etc.
- Il problema è: **come poter far comprendere tutti i caratteri da noi conosciuti ad un sistema che comprende solo due stati, che possiamo esemplificare come stato 0 e stato 1?**
- Si deve ricorrere ad un **processo di codifica**.

# La Codifica dei Caratteri

- Un esempio di codifica è il codice Morse: un codice, utilizzato per il telegrafo, che permette di codificare delle lettere dell'alfabeto con dei segnali sonori lunghi o corti.
- Nel codice Morse ogni lettera è formata da tre segnali. L'esempio più famoso è la richiesta di SOS:

S = ---                      O = \_ \_ \_                      S = ---

- Allo stesso modo per la codifica delle lettere nel calcolatore si segue un procedimento simile: ad ogni carattere è associata una sequenza di segnali, di 0 e 1.

# La Codifica dei Caratteri

- Se avessi 3 cifre potremmo codificare sono i seguenti caratteri:

000 → A  
001 → B  
010 → C  
011 → D  
100 → E  
101 → F  
110 → G  
111 → H

- Se avessimo 4 cifre tutte le combinazioni possibili di 4 segnali, utilizzando due simboli (0 e 1) corrisponde a  $2^4 = 16$ .
- È stato stabilito a livello internazionale che per ogni carattere si usano 8 cifre, quindi  $2^8 = 256$  combinazioni.

00000001 → A  
00000010 → B

# La Codifica dei Caratteri

Una sequenza di 8 numeri 0 e 1 in informatica è chiamata **byte**. La singola cifra è chiamata **bit**. Quindi il bit può valere o 0 o 1.

Riassumendo:

- **il bit (binary digit) costituisce l'unità elementare di memorizzazione;**
- **un gruppo di 8 bit viene detto byte e consente di codificare 256 ( $2^8$ ) simboli o dati elementari diversi.**
- La codifica più diffusa è la **codifica ASCII** (*American Standard Code for Information Interchange*), che usa 7 bit per codificare i caratteri (inclusi in un byte con il primo bit a 0).
- Successivamente è stata introdotta la **codifica ASCII estesa**: codifica anche simboli speciali (es. è, à, ü), con il primo bit a 1; non è realmente standard.
- Un'altra codifica meno comune è l'**UNICODE**: 16 bit, 65536 caratteri, permette di rappresentare caratteri per tutti gli alfabeti.
- Ecco le tabelle di codifica ASCII standard ed estesa.

Byte	Cod	Char	Byte	Cod	Char	Byte	Cod	Char	Byte	Cod	Char
00000000	0	Null	00100000	32	Spc	01000000	64	@	01100000	96	`
00000001	1	Start of heading	00100001	33	!	01000001	65	A	01100001	97	a
00000010	2	Start of text	00100010	34	"	01000010	66	B	01100010	98	b
00000011	3	End of text	00100011	35	#	01000011	67	C	01100011	99	c
00000100	4	End of transmit	00100100	36	\$	01000100	68	D	01100100	100	d
00000101	5	Enquiry	00100101	37	%	01000101	69	E	01100101	101	e
00000110	6	Acknowledge	00100110	38	&	01000110	70	F	01100110	102	f
00000111	7	Audible bell	00100111	39	'	01000111	71	G	01100111	103	g
00001000	8	Backspace	00101000	40	(	01001000	72	H	01101000	104	h
00001001	9	Horizontal tab	00101001	41	)	01001001	73	I	01101001	105	i
00001010	10	Line feed	00101010	42	*	01001010	74	J	01101010	106	j
00001011	11	Vertical tab	00101011	43	+	01001011	75	K	01101011	107	k
00001100	12	Form Feed	00101100	44	,	01001100	76	L	01101100	108	l
00001101	13	Carriage return	00101101	45	-	01001101	77	M	01101101	109	m
00001110	14	Shift out	00101110	46	.	01001110	78	N	01101110	110	n
00001111	15	Shift in	00101111	47	/	01001111	79	O	01101111	111	o
00010000	16	Data link escape	00110000	48	0	01010000	80	P	01110000	112	p
00010001	17	Device control 1	00110001	49	1	01010001	81	Q	01110001	113	q
00010010	18	Device control 2	00110010	50	2	01010010	82	R	01110010	114	r
00010011	19	Device control 3	00110011	51	3	01010011	83	S	01110011	115	s
00010100	20	Device control 4	00110100	52	4	01010100	84	T	01110100	116	t
00010101	21	Neg. acknowledge	00110101	53	5	01010101	85	U	01110101	117	u
00010110	22	Synchronous idle	00110110	54	6	01010110	86	V	01110110	118	v
00010111	23	End trans. block	00110111	55	7	01010111	87	W	01110111	119	w
00011000	24	Cancel	00111000	56	8	01011000	88	X	01111000	120	x
00011001	25	End of medium	00111001	57	9	01011001	89	Y	01111001	121	y
00011010	26	Substitution	00111010	58	:	01011010	90	Z	01111010	122	z
00011011	27	Escape	00111011	59	;	01011011	91	[	01111011	123	{
00011100	28	File separator	00111100	60	<	01011100	92	\	01111100	124	
00011101	29	Group separator	00111101	61	=	01011101	93	]	01111101	125	}
00011110	30	Record Separator	00111110	62	>	01011110	94	^	01111110	126	~
00011111	31	Unit separator	00111111	63	?	01011111	95	_	01111111	127	Del

TABELLA ASCII STANDARD

# Codifica delle immagini

- L'utilizzo delle immagini nei computer è stato reso possibile dall'aumentata potenza di calcolo e di memoria dei computer che finalmente sono riusciti a gestire la grossa mole di dati contenuta in una semplice immagine.
- La codifica delle immagini è più complessa rispetto a quella dei numeri e dei caratteri.
- Una immagine è, per sua natura, un insieme continuo di informazioni: non è divisibile in cifre, come un numero, o in lettere come una parola. Una immagine è un tutto unico.
- La soluzione più comune prevede la scomposizione dell'immagine in una griglia di tanti elementi (punti o pixel, picture element) che sono l'unità minima di memorizzazione.

# Codifica delle immagini

- Ogni pixel assume come valore il colore medio dell'area che rappresenta. La griglia è ordinata dal basso verso l'alto e da sinistra verso destra, e corrisponde ad una matrice costituita dai valori dei pixel.
- Chiaramente l'insieme dei valori dei pixel è una approssimazione dell'immagine. La precisione della codifica dipende dal numero di pixel nella griglia (risoluzione).



# Codifica delle immagini

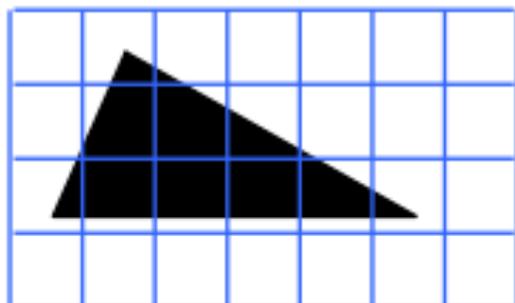


immagine e griglia

0 = bianco

1 = nero

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

rappresentazione in pixel

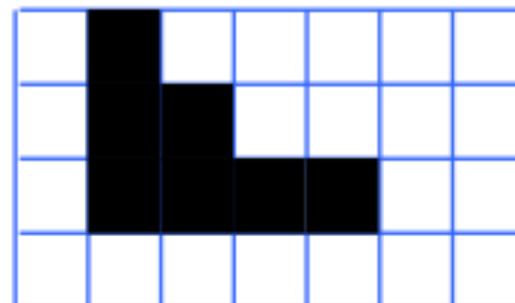


immagine digitale



# Codifica delle immagini

- Le immagini si suddividono in **raster** e **vettoriali**.
- Le **immagini raster** sono idealmente suddivise in punti (pxel) disposti su una griglia a righe orizzontali e verticali. In origine, in un'immagine raster di tipo **bitmap** (mappa di bit) era possibile rappresentare solo due colori: bianco (bit 0) e nero (bit 1).
- L'accezione attualmente in uso prevede diverse profondità di colore per i pixel dell'immagine. Con 8 bit è possibile rappresentare 256 colori (es. le tonalità di grigio), con 16 più di 32000 colori, e così

# Codifica delle immagini

- Ad esempio, quando si parla di un'immagine alla **risoluzione di  $640 \times 480$  pixel**, significa che i pixel totali sono appunto  $640 \cdot 480 = 307200$  e che essi sono disposti su una griglia di 640 colonne (larghezza) e 480 righe (altezza).
- Un'immagine di  $640 \times 480$  pixel con 256 colori occupa  $640 \times 480 \times 8 = 2457600$  bit = 307200 byte = 307.2 Kbyte.

# Codifica dei filmati

- Sono sequenze di immagini compresse (ad esempio si possono registrare solo le variazioni tra un fotogramma e l'altro).
- Esistono vari formati (compresi i suoni):
  - avi (microsoft)
  - mpeg (il più usato)
  - quicktime mov (apple)

# Codifica dei suoni

- L'onda sonora viene misurata (campionata) a intervalli regolari. Minore è l'intervallo di campionamento, maggiore è la qualità del suono.
- Per i CD musical si ha: 44000 campionamenti al secondo, 16 bit per campione.

# **RAPPRESENTAZIONE DEI DATI**

## **SISTEMI DI MISURA**

**FINE**

**Dipartimento di Storia, Società e Studi sull'Uomo  
Università del Salento**