DMIF, Università di Udine

Tecnologie Digitali per il Cibo e la Ristorazione

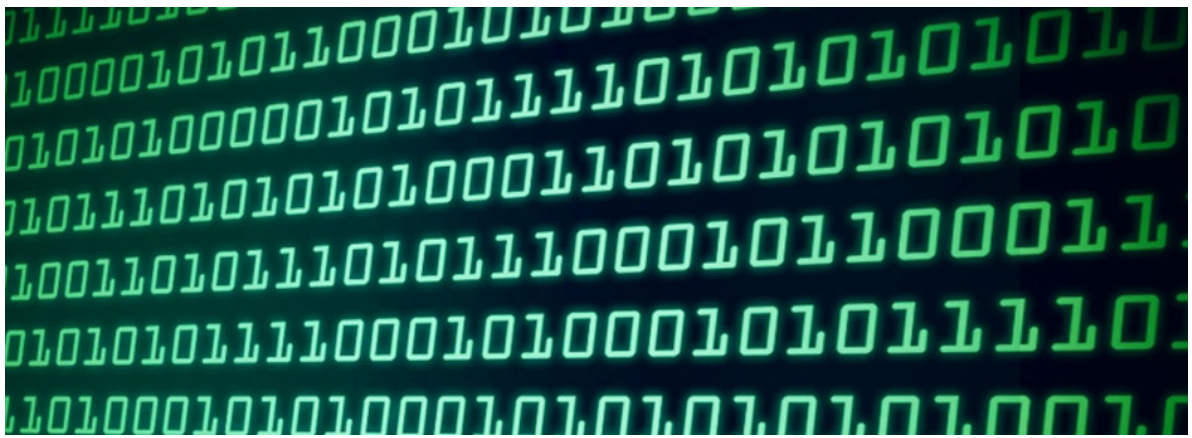
Rappresentazione dell'informazione

Andrea Brunello
andrea.brunello@uniud.it

A.A. 2021–2022

Bit e informazione

- Nei calcolatori l'informazione è memorizzata sotto forma di pattern di 0 e 1, per ragioni di efficienza
- I singoli 0 e 1 vengono detti **bit** (binary digit)
- I bit posso essere utilizzati per rappresentare valori numerici, caratteri, immagini, suoni, filmati, …



Bit, Byte, Megabyte, …

- Per ragioni storiche, 8 bit = 1 byte

Multiples of Bits		
Unit (Symbol)	**Value (SI)**	**Value (Binary)**
Kilobit (Kb) (Kbit)	10^3	2^{10}
Megabit (Mb) (Mbit)	10^6	2^{20}
Gigabit (Gb) (Gbit)	10^9	2^{30}
Terabit (Tb) (Tbit)	10^{12}	2^{40}
Petabit (Pb) (Pbit)	10^{15}	2^{50}
Exabit (Eb) (Ebit)	10^{18}	2^{60}
Zettabit (Zb) (Zbit)	10^{21}	2^{70}
Yottabit (Yb) (Ybit)	10^{24}	2^{80}

Multiples of Bytes		
Unit (Symbol)	**Value (SI)**	**Value (Binary)**
Kilobyte (kB)	10^3	2^{10}
Megabyte (MB)	10^6	2^{20}
Gigabyte (GB)	10^9	2^{30}
Terabyte (TB)	10^{12}	2^{40}
Petabyte (PB)	10^{15}	2^{50}
Exabyte (EB)	10^{18}	2^{60}
Zettabyte (ZB)	10^{21}	2^{70}
Yottabyte (YB)	10^{24}	2^{80}

Operazioni booleane

- Possiamo associare il valore *falso* al bit 0 e *vero* al bit 1
- Le operazioni booleane manipolano i valori dei bit
  - *NOT*: nega (scambia) il valore di verità ricevuto in input
  - *AND*: restituisce *vero* se entrambi i valori di verità ricevuti in input sono veri
  - *OR*: restituisce *vero* se almeno uno dei due valori di verità ricevuti in input è vero
  - *XOR*: restituisce *vero* se esattamente uno dei due valori di verità ricevuti in input è vero (derivabile)

Operazioni booleane
Esempi

- $NOT(1) = 0$
- $AND(0, 1) = 0$
- $AND(1, 1) = 1$
- $OR(0, 0) = 0$
- $OR(1, 0) = 1$
- $XOR(1, 1) = 0$
- $XOR(0, 1) = 1$

Altri operatori logici che possono essere derivati: NAND, NOR, XNOR

Algebra booleana

- L'**algebra booleana** è il ramo dell'algebra in cui le variabili possono assumere solamente i valori *vero* (1) e *falso* (0), e le operazioni fondamentali sono gli operatori booleani
- Riveste un ruolo di fondamentale importanza nell'informatica
- L'algebra consente di combinare operazioni booleane
- Esempi:
  - $AND(NOT(1), 1) = 0$
  - $AND(1, OR(1, 0)) = 1$
  - $XOR(NOT(1), NOT(AND(1, 0))) = 1$

- Notazione per le tre operazioni fondamentali:
  - $AND(A, B) = AB$
  - $OR(A, B) = A + B$
  - $NOT(A) = \overline{A}$
- Alcune proprietà:

Name	AND form	OR form
Identity law	1A = A	0 + A = A
Null law	0A = 0	1 + A = 1
Idempotent law	AA = A	A + A = A
Inverse law	$A\overline{A} = 0$	$A + \overline{A} = 1$
Commutative law	AB = BA	A + B = B + A
Associative law	(AB)C = A(BC)	(A + B) + C = A + (B + C)
Distributive law	A + BC = (A + B)(A + C)	A(B + C) = AB + AC
Absorption law	A(A + B) = A	A + AB = A
De Morgan's law	$\overline{AB} = \overline{A} + \overline{B}$	$\overline{A + B} = \overline{A}\overline{B}$

Verifichiamo che $A(A+B)$ sia equivalente ad A

A	B	$A+B$	$A(A+B)$
0	0	0	0
1	0	1	1
0	1	1	0
1	1	1	1

Verifichiamo che $\overline{AB}$ sia equivalente a $\overline{A}+\overline{B}$

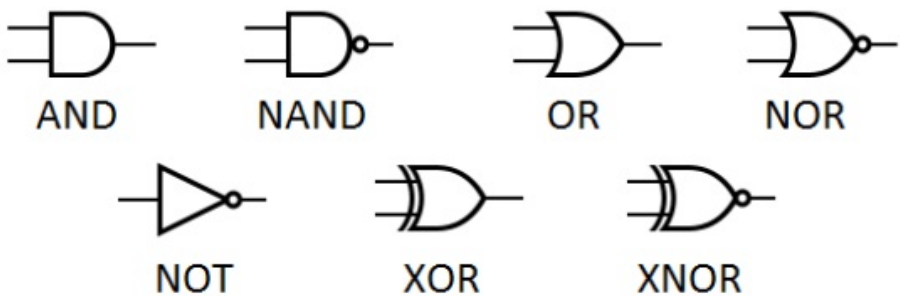
A	B	AB	$\overline{AB}$
0	0	0	1
1	0	0	1
0	1	0	1
1	1	1	0

A	B	$\overline{A}$	$\overline{B}$	$\overline{A}+\overline{B}$
0	0	1	1	1
1	0	0	1	1
0	1	1	0	1
1	1	0	0	0

$$
\begin{aligned}
A\overline{B}C + AB\overline{C} + ABC + ABC &= \\
A\overline{B}C + AB\overline{C} + ABC &= \\
A\overline{B}C + AB(\overline{C} + C) &= \\
A\overline{B}C + AB1 &= \\
A\overline{B}C + AB &= \\
A(\overline{B}C + B) &= \\
A(C + B) \quad . &
\end{aligned}
$$

Porte logiche

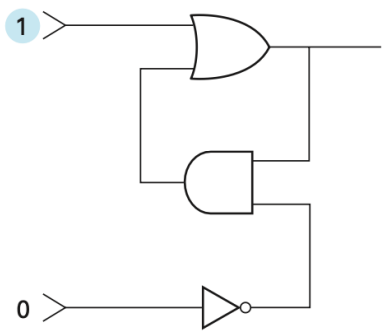
- All'interno del calcolatore, gli operatori booleani sono implementati mediante porte logiche
- Una porta logica è un piccolo circuito in cui i valori di 0 e 1 sono rappresentati da diversi livelli di voltaggio
- Le porte logiche sono i mattoni fondamentali del calcolatore



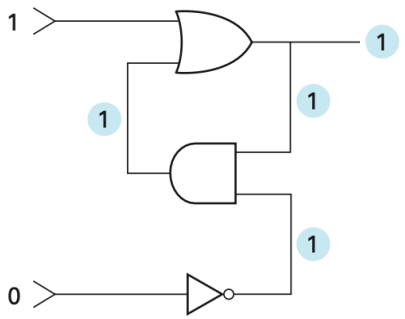
Flip-flop

- Oltre che per effettuare operazioni booleane, i circuiti costituiti da porte logiche possono anche essere utilizzati per **memorizzare informazioni**
- È il caso dei **flip-flop**
- Ciascun flip-flop è in grado di memorizzare il valore di un bit (memoria SRAM)

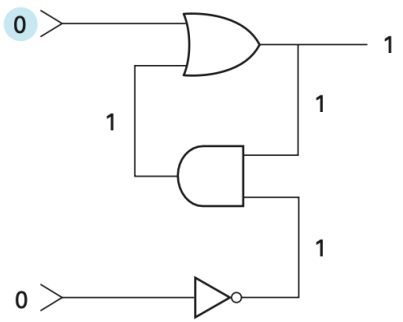
a. First, a 1 is placed on the upper input.



b. This causes the output of the OR gate to be 1 and, in turn, the output of the AND gate to be 1.



c. Finally, the 1 from the AND gate keeps the OR gate from changing after the upper input returns to 0.



Codifica dell'informazione

Vediamo ora come è possibile codificare in pattern di 0 e 1:

- Testi
- Numeri
- Immagini
- Suoni

Codifica di testi

- **Idea:** associare un diverso pattern di bit a ciascun carattere
  - Il testo può così essere rappresentato dalla concatenazione dei pattern di bit associati ai caratteri
- **Problema:** quali pattern associare ai diversi caratteri?
  - Negli anni ’40 e ’50, sistemi diversi utilizzavano codifiche diverse
  - Difficoltà riguardanti interoperabilità e scambio dei dati
- **Soluzione:** standardizzazione

Codifica di testi
ASCII

- American Standard Code for Information Interchange
- Pubblicato dall'ANSI (American National Standards Institute) nel 1963
- Deriva dai codici telegrafici, ideato per le telescriventi
- 7 bit $\rightsquigarrow$ 128 caratteri rappresentabili

Decimal	Binary	Octal	Hex	ASCII	Decimal	Binary	Octal	Hex	ASCII	Decimal	Binary	Octal	Hex	ASCII	Decimal	Binary	Octal	Hex	ASCII
0	00000000	000	00	NUL	32	00100000	040	20	SP	64	01000000	100	40	@	96	01100000	140	60	`
1	00000001	001	01	SOH	33	00100001	041	21	!	65	01000001	101	41	A	97	01100001	141	61	a
2	00000010	002	02	STX	34	00100010	042	22	"	66	01000010	102	42	B	98	01100010	142	62	b
3	00000011	003	03	ETX	35	00100011	043	23	#	67	01000011	103	43	C	99	01100011	143	63	c
4	00000100	004	04	EOT	36	00100100	044	24	$	68	01000100	104	44	D	100	01100100	144	64	d
5	00000101	005	05	ENQ	37	00100101	045	25	%	69	01000101	105	45	E	101	01100101	145	65	e
6	00000110	006	06	ACK	38	00100110	046	26	&	70	01000110	106	46	F	102	01100110	146	66	f
7	00000111	007	07	BEL	39	00100111	047	27	'	71	01000111	107	47	G	103	01100111	147	67	g
8	00001000	010	08	BS	40	00101000	050	28	(	72	01001000	110	48	H	104	01101000	150	68	h
9	00001001	011	09	HT	41	00101001	051	29	)	73	01001001	111	49	I	105	01101001	151	69	i
10	00001010	012	0A	LF	42	00101010	052	2A	*	74	01001010	112	4A	J	106	01101010	152	6A	j
11	00001011	013	0B	VT	43	00101011	053	2B	+	75	01001011	113	4B	K	107	01101011	153	6B	k
12	00001100	014	0C	FF	44	00101100	054	2C	,	76	01001100	114	4C	L	108	01101100	154	6C	l
13	00001101	015	0D	CR	45	00101101	055	2D	-	77	01001101	115	4D	M	109	01101101	155	6D	m
14	00001110	016	0E	SO	46	00101110	056	2E	.	78	01001110	116	4E	N	110	01101110	156	6E	n
15	00001111	017	0F	SI	47	00101111	057	2F	/	79	01001111	117	4F	O	111	01101111	157	6F	o
16	00010000	020	10	DLE	48	00110000	060	30	0	80	01010000	120	50	P	112	01110000	160	70	p
17	00010001	021	11	DC1	49	00110001	061	31	1	81	01010001	121	51	Q	113	01110001	161	71	q
18	00010010	022	12	DC2	50	00110010	062	32	2	82	01010010	122	52	R	114	01110010	162	72	r
19	00010011	023	13	DC3	51	00110011	063	33	3	83	01010011	123	53	S	115	01110011	163	73	s
20	00010100	024	14	DC4	52	00110100	064	34	4	84	01010100	124	54	T	116	01110100	164	74	t
21	00010101	025	15	NAK	53	00110101	065	35	5	85	01010101	125	55	U	117	01110101	165	75	u
22	00010110	026	16	SYN	54	00110110	066	36	6	86	01010110	126	56	V	118	01110110	166	76	v
23	00010111	027	17	ETB	55	00110111	067	37	7	87	01010111	127	57	W	119	01110111	167	77	w
24	00011000	030	18	CAN	56	00111000	070	38	8	88	01011000	130	58	X	120	01111000	170	78	x
25	00011001	031	19	EM	57	00111001	071	39	9	89	01011001	131	59	Y	121	01111001	171	79	y
26	00011010	032	1A	SUB	58	00111010	072	3A	:	90	01011010	132	5A	Z	122	01111010	172	7A	z
27	00011011	033	1B	ESC	59	00111011	073	3B	;	91	01011011	133	5B	[	123	01111011	173	7B	{
28	00011100	034	1C	FS	60	00111100	074	3C	<	92	01011100	134	5C	\	124	01111100	174	7C	|
29	00011101	035	1D	GS	61	00111101	075	3D	=	93	01011101	135	5D	]	125	01111101	175	7D	}
30	00011110	036	1E	RS	62	00111110	076	3E	>	94	01011110	136	5E	^	126	01111110	176	7E	~
31	00011111	037	1F	US	63	00111111	077	3F	?	95	01011111	137	5F	_	127	01111111	177	7F	DEL

Codifica di testi
ASCII esteso

- L'ISO (International Organization for Standardizaton) ha sviluppato negli anni diverse estensioni ad ASCII, basate su codici a 8 bit
- Primi 128 caratteri identici all'ASCII originale
- Secondi 128 caratteri dipendenti dalle lingue che si vogliono supportare:
  - ISO Latin 1: per le lingue dell'Europa occidentale
  - ISO 8859-2: per le lingue dell'Europa orientale
  - ISO 8859-5: per l'alfabeto Cirillico
  - …

- L'ASCII ha delle fondamentali limitazioni:
  - 128 (o 256) caratteri sono insufficienti per certe lingue dell'Europa orientale o dell'Asia
  - Non è possibile creare documenti che contengono caratteri provenienti da estensioni ASCII diverse
- Sviluppo dello standard **Unicode**:
  - Unicode Transformation Formats (UTFs): UTF-8, UTF-16 (65.536 caratteri), UTF-32 (>4 miliardi caratteri)
  - Retrocompatibile con ASCII

Codifica di numeri

- Considerare i numeri come stringhe e codificarli utilizzando ad esempio codici UTF-8 sarebbe inefficiente (24 bit per codificare '375')

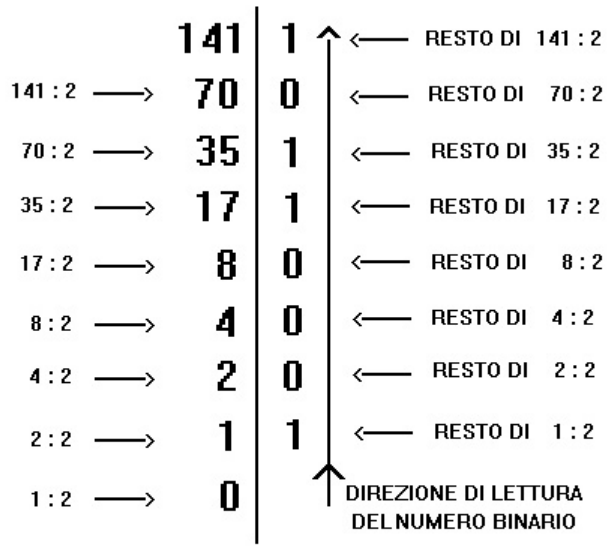
- Se l'informazione è prettamente numerica, possono essere utilizzati approcci basati sul **sistema numerico binario**
  - Complemento a 2 per i numeri interi
  - Notazione IEEE 754 per i numeri decimali

> *There are only 10 types of people in the world: those who understand binary and those who don't — Unknown.*

- Si ricordi il funzionamento dell'abaco: le palline possono corrispondere a unità, decine, centinaia, ...
- Dato un numero in base 10, ogni sua cifra rappresenta una quantità $\in \{0, \ldots, 9\}$ da moltiplicare per la relativa potenza di 10
  - $375_{10} = 3 * 10^2 + 7 * 10^1 + 5 * 10^0$
- Possiamo ragionare allo stesso modo in base 2, dove le quantità $\in \{0, 1\}$ e vengono considerate potenze di 2
  - $101110111_2 = 2^8 + 2^6 + 2^5 + 2^4 + 2^2 + 2^1 + 2^0$ $(= 375_{10})$
- Abbiamo utilizzato 9 bit per rappresentare 375

Codifica di numeri
Conversione fra base 2 e base 10 (numeri interi)

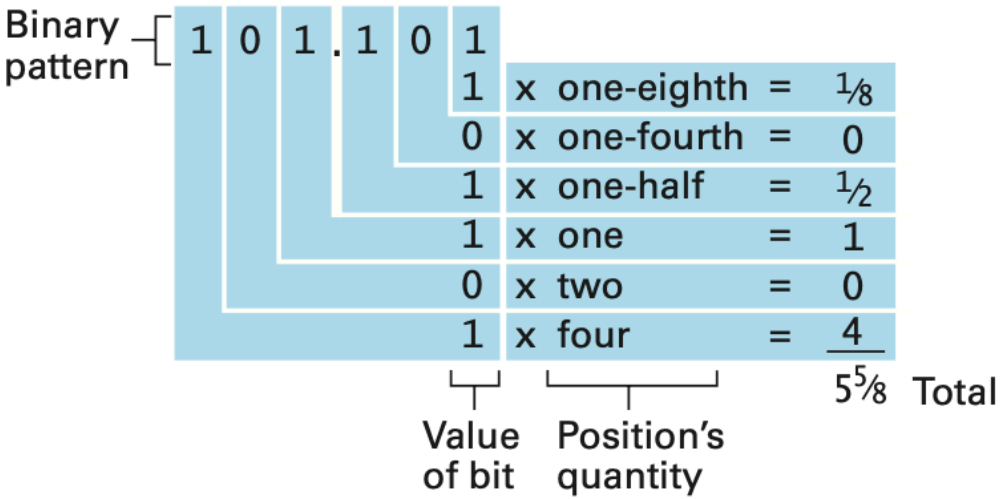
- Da base 2 a base 10:
  - $10001101_2 = 2^7 + 2^3 + 2^2 + 2^0 = 141_{10}$
- Da base 10 a base 2:
  - Dividere il numero per due, dividere il quoziente ottenuto ancora per due, e così via fino ad ottenere quoziente zero
  - Il numero binario corrispondente si ottiene prendendo i resti a partire dall'ultimo



141 = 10001101

Codifica di numeri
Conversione fra base 2 e base 10 (numeri frazionari)

- Da base 2 a base 10:



- Da base 10 a base 2:
  - Moltiplicare la parte decimale per 2 e osservare la cifra che appare nella parte intera, moltiplicare la nuova parte decimale per due, e così via fino ad ottenere parte decimale zero (o fino al numero di bit a disposizione)
  - Il numero binario si ottiene prendendo le cifre che appaiono nelle parti intere, a partire dalla prima

- Conversione di 18.59_{10} in base 2
- Trattiamo separatamente la parte intera e la parte decimale

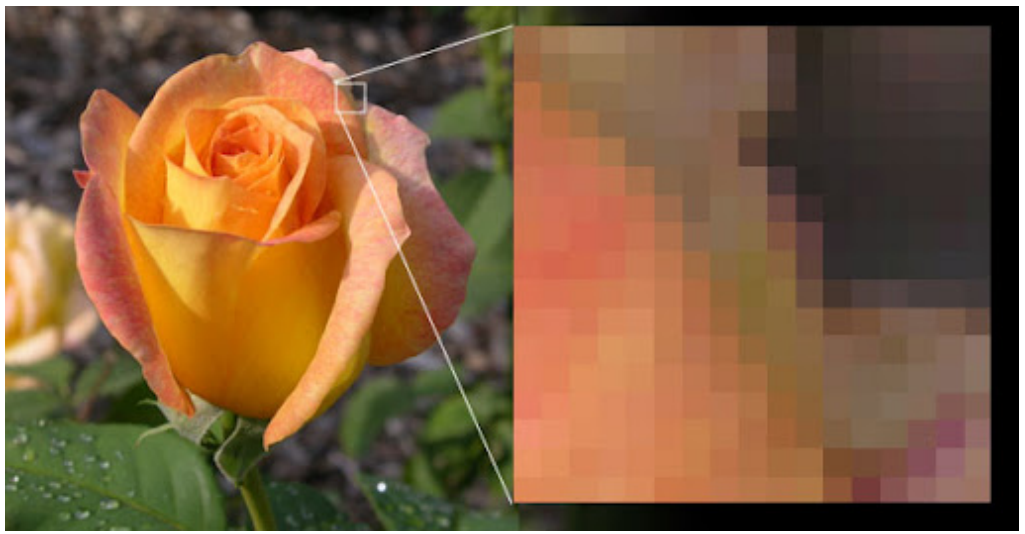
18	0
9	1
4	0
2	0
1	1
0	

0.59*2 = 1.18
0.18*2 = 0.36
0.36*2 = 0.72
0.72*2 = 1.44
0.44*2 = 0.88
0.88*2 = 1.76

- Otteniamo 10010.100101_2 (valore approssimato)

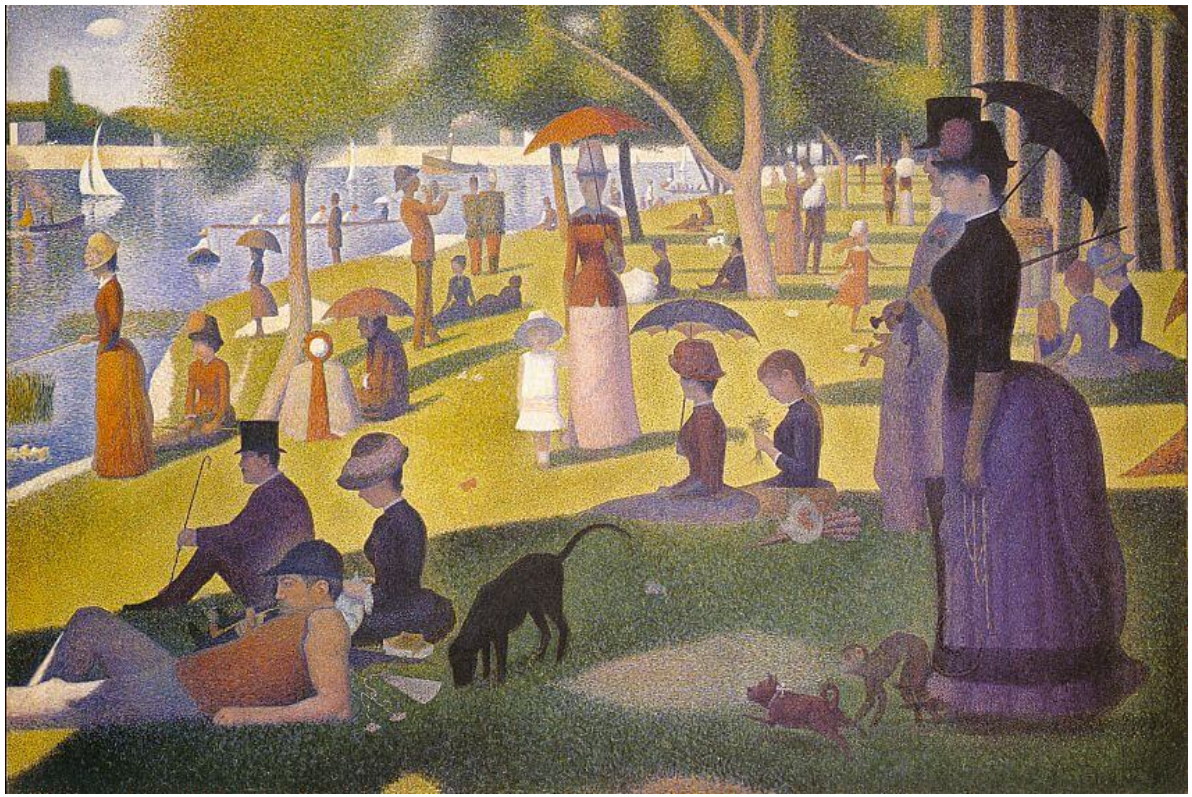
Codifica di immagini

- Il modo più semplice di codificare un'immagine è rappresentarla come un insieme di punti, detti **pixel** (picture element)
- La matrice di punti che rappresenta un'immagine è detta **bitmap**

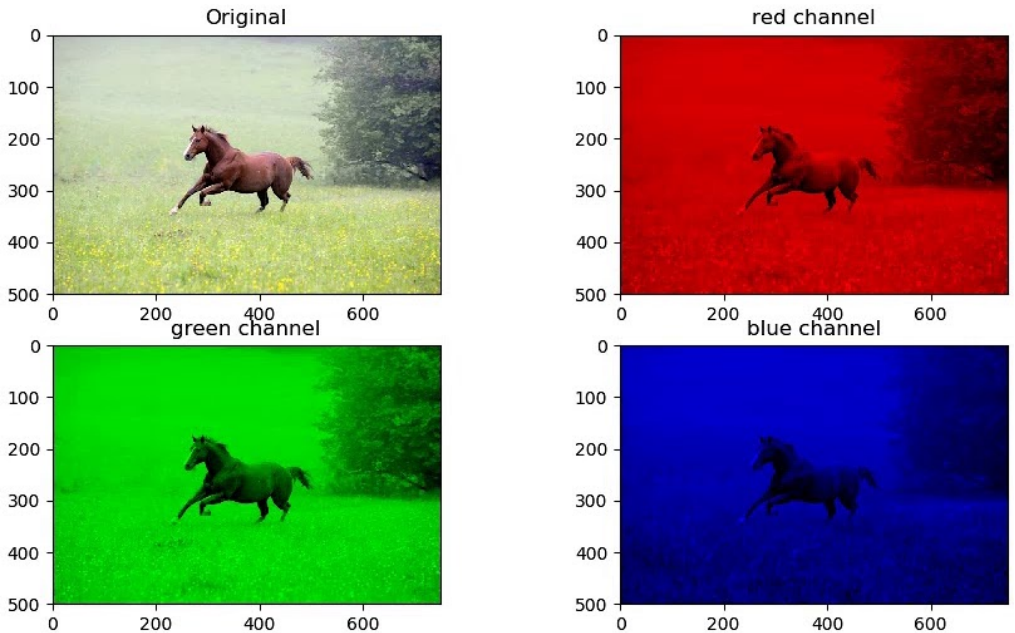


Codifica di immagini

Un po' come avviene nei quadri dei pittori puntinisti, salvo che nei bitmap consideriamo una matrice/griglia regolare



- I metodi per codificare i singoli pixel cambiano a seconda del tipo di immagine:
  - Immagini b/n: singolo valore binario
  - Immagini in scala di grigi: 8 bit (256 livelli)
  - Immagini a colori: RGB (3 x 8 bit, 16.8 milioni di colori)



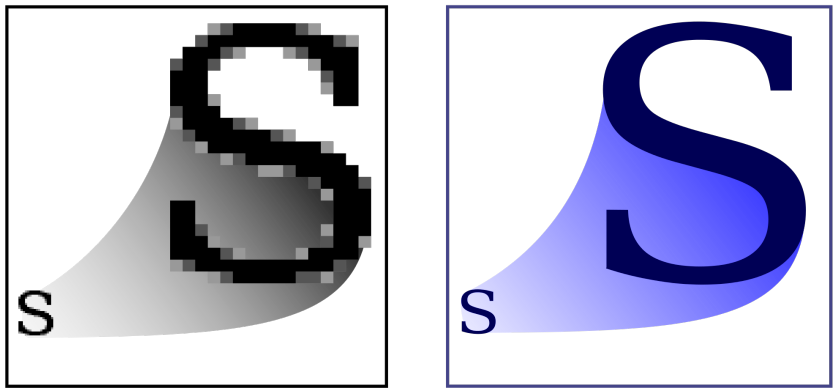
- Per rappresentare il colore può essere utilizzata la notazione HTML #RRGGBB, basata su cifre esadecimali
- Ad esempio #54BC9E corrisponde alla sequenza di 24 bit 010101001011110010011110
- Per trasformare una stringa binaria in esadecimale è sufficiente suddividerla in blocchi di 4 numeri (eventualmente con 0 padding a sinistra), e convertire ciascun blocco seguendo la tabella

Bit pattern	Hexadecimal representation
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

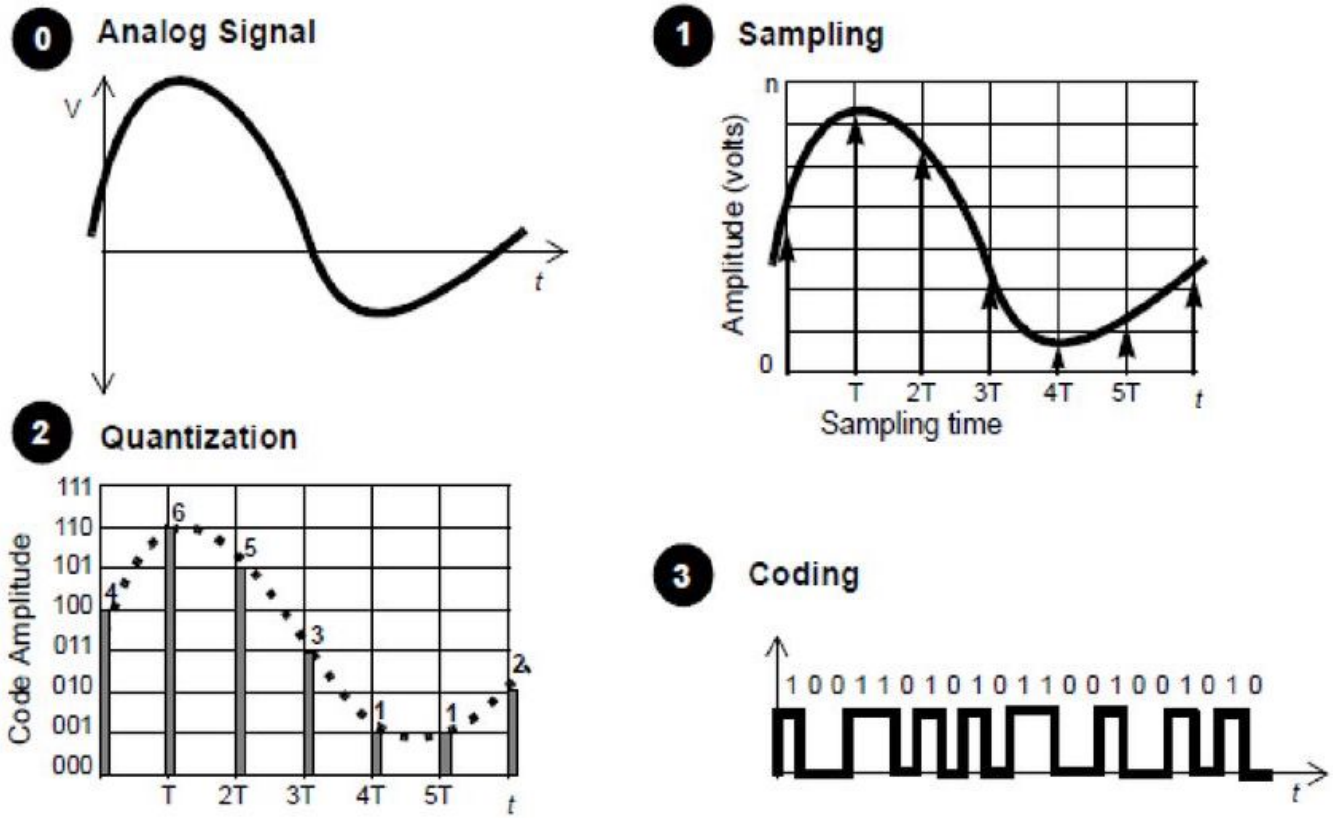
- Formato bitmap, non compresso: .bmp
- Formati compressi:
  - JPEG
    - buono per le foto
    - compressione lossy
    - rapporto di dimensioni di circa 10:1
    - nessun supporto per le trasparenze
    - formato compresso standard nella fotografia digitale e nella condivisione di immagini
  - PNG
    - buono per le immagini generate digitalmente, o con contorni netti / alto contrasto
    - compressione lossless
    - file più grandi rispetto a JPEG
    - supporto per le trasparenze

PNG vs JPEG

- Uno svantaggio delle immagini codificate in formato bitmap è che esse non possono essere facilmente riscalate
- Alternativamente, è possibile rappresentare le immagini come collezioni di oggetti geometrici, es. linee o curve
- Tali oggetti possono poi essere visualizzati a grandezze diverse senza perdere in qualità
- Questo è il caso, ad esempio, di immagini salvate in formato SVG, o dei font di tipo TrueType



Codifica di suoni



- Sampling rate: 8.000 Hz (telefono), 44.100 Hz (CD)
- Bit per quantizzazione: 16 bit, 32 bit