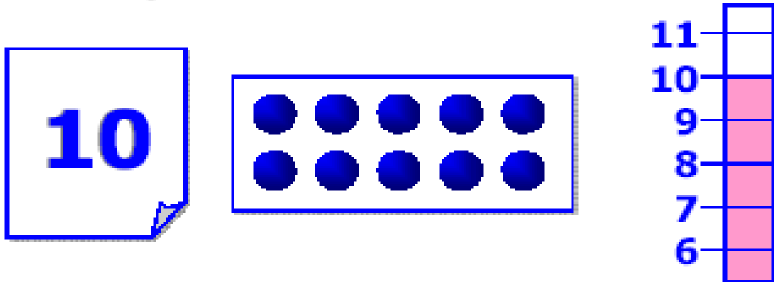
Introduzione alla codifica digitale

Codifica

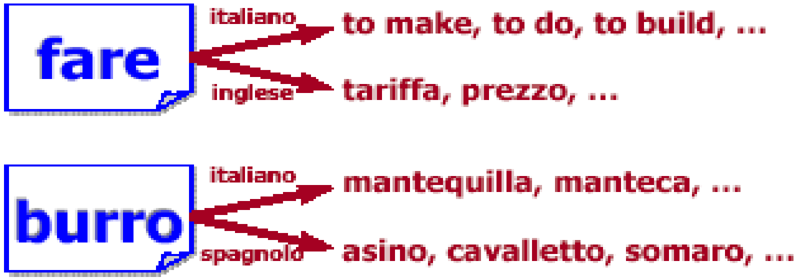
- L’informazione gestita dai sistemi di elaborazione deve essere codificata
  - per poter essere memorizzata, elaborata, scambiata, …
- E’ necessario codificare sia dati che istruzioni
  - Per scrivere un programma è necessario codificare dati e istruzioni
- L’esecutore automatico deve essere in grado di:
  - Memorizzare istruzioni e dati
  - Manipolare istruzioni e dati

Informazione e codifica

- La stessa informazione si può codificare in modi differenti



- Stessa codifica per informazioni differenti



Sistema di codifica

- Detto anche “codice”
- Usa un insieme di simboli di base (alfabeto)
- I simboli dell’alfabeto possono essere combinati ottenendo differenti configurazioni (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 (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!*
- Codice (semantica)
  - Associano ad ogni configurazione un'entità di informazione
    - 2.318,43 = $2\times10^{3}+3\times10^{2}+1\times10^{1}+8\times10^{0}+4\times10^{-1}+3\times10^{-2}$
- Sistemi diversi possono usare lo stesso alfabeto
  - 123,456 = $1\times10^{2}+2\times10^{1}+3\times10^{0}+4\times10^{-1}+5\times10^{-2}+6\times10^{-3}$ [**IT**]
  - 123,456 = $1\times10^{5}+2\times10^{4}+3\times10^{3}+4\times10^{2}+5\times10^{1}+6\times10^{0}$ [**UK**]

Codifica binaria

- Codifica binaria: usa un alfabeto di **2** simboli
- Usata nei sistemi informatici (livello fisico)
  - si usa una grandezza fisica (luminosità, tensione elettrica, corrente elettrica) per rappresentare l'informazione
- Solo 2 simboli per ridurre la probabilità di errore
  - tanti più simboli si devono distinguere e tanto meno la rilevazione sarà affidabile in presenza di "rumore"

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

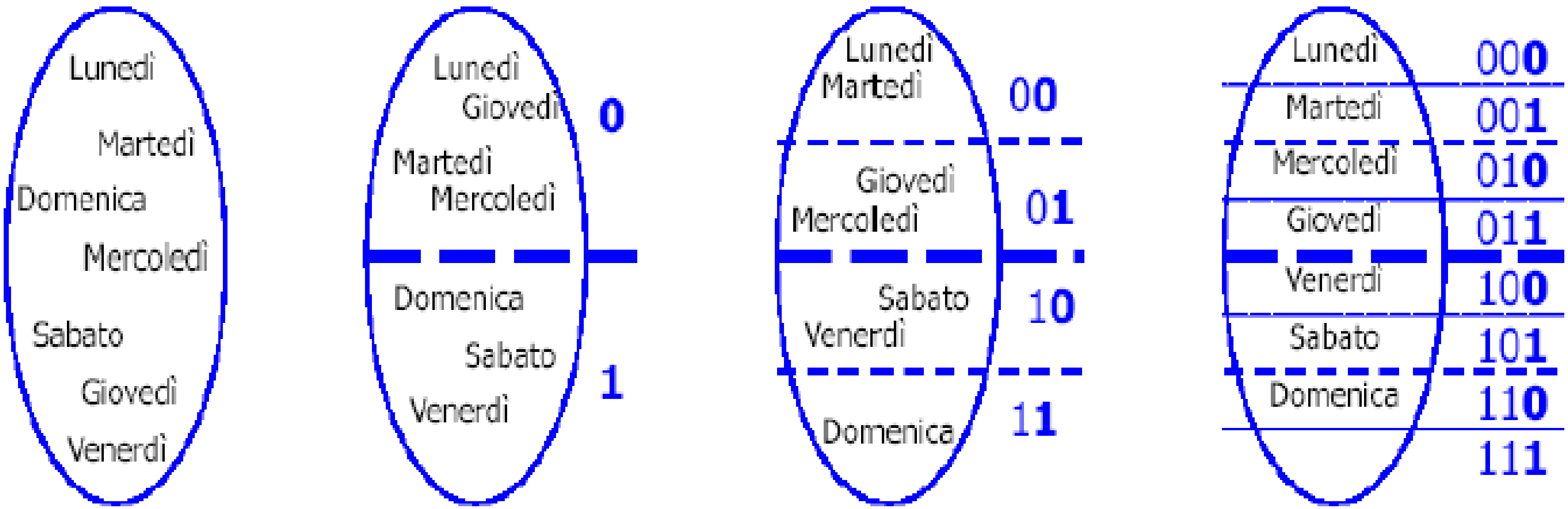
Codifica binaria: unità di misura

- **BYTE** = 8 bit
- **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

Codifica binaria

- 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

Es.: i giorni della settimana in binario



1 bit
2 "gruppi"

2 bit
4 "gruppi"

3 bit
8 "gruppi"

Codifica binaria

- Con **K** bit si possono rappresentare **2^K** stati
- Quanti bit servono per codificare N stati?

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

Codifica dei numeri naturali

- Sistema di numerazione posizionale con **base β**
  - *β* simboli (**cifre**) corrispondono ai numeri da *0* a *β-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

$$a_{n-1}\,a_{n-2}\,\ldots a_1\,a_0$$

allora *N* può essere espresso come segue:

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

Codifica dei numeri naturali

- *Esempio*:
  - 13 può essere espresso mediante potenze di 2 come:

$$13 \quad = \quad \mathbf{1} \times 8 + \mathbf{1} \times 4 + \mathbf{0} \times 2 + \mathbf{1} \times 1$$

  - 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
  - ....

Esercizi

- DA BASE 2 A BASE 10
  - $10011_2 = 1 \times 2^4 + 0 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 =$
    $= 19_{10}$
- DA BASE 8 A BASE 10
  - $7201_8 = 7 \times 8^3 + 2 \times 8^2 + 0 \times 8^1 + 1 \times 8^0 = 3713_{10}$
- DA BASE 16 A BASE 10
  - $A02F7_{16} = A \times 16^4 + 0 \times 16^3 + 2 \times 16^2 + F \times 16^1 + 7 \times 16^0 =$
    $= 656119_{10}$
- ALTRI ESEMPI
  - $72F1_{16} = ????_{10}$
  - $1011_2 = ????_{10}$
  - $1407_8 = ????_{10}$

Esercizi (2)

- Conversioni da base 10 a base 8

313	1
39	7
4	4

$313_{10} = 471_8$

- Conversioni da base 10 a base 16

313	9
19	3
1	1

$313_{10} = 139_{16}$

Esercizi (3)

- Conversioni da base 2 a base 8

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

$000\ 100\ 111\ 001_2 = 0\ 4\ 7\ 1_8$

- Conversioni da base 2 a base 16

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

$0001\ 0011\ 1001_2 = 1\ 3\ 9_{16}$

- Proprietà:
  - Ciascuna cifra ottale (esadecimale) corrisponde ad un gruppo di 3 (4) cifre binarie

Parte frazionaria

- Per convertire ***la sola parte frazionaria***, si moltiplica il numero per 2, sottraendo 1 dal prodotto se è maggiore di 1 e continuando a moltiplicare per 2 il risultato così ottenuto fino a quando non si ottiene un risultato uguale a 0 oppure un risultato già ottenuto in precedenza

- Il numero binario si ottiene scrivendo ***la serie delle parti intere dei prodotti*** ottenuti, ***iniziando dal primo***
- Se si ottiene un risultato già ottenuto in precedenza, il numero sarà ***periodico***, anche se ***non lo era*** in base decimale

$$0{,}35 \cdot 2 = 0{,}7$$
$$0{,}7 \cdot 2 = 1{,}4$$
$$0{,}4 \cdot 2 = \boxed{0}{,}8$$
$$0{,}8 \cdot 2 = \boxed{1}{,}6$$
$$0{,}6 \cdot 2 = \boxed{1}{,}2$$
$$0{,}2 \cdot 2 = \boxed{0}{,}4$$

$$(0{,}35)_{10} = (0{,}01\overline{0110})_2$$

Codifica dei numeri interi

- Vari tipi di codifiche:
  - 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:

+5	0 0000101
- 5	1 0000101
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 → 00000101
  - -5 → 11111010
- Difetti
  - Stessi difetti della rappresentazione in modulo e segno (+0→ 00000000 -0→11111111)

Complemento a due

- Approccio
  - Un numero è rappresentato con la codifica del suo complemento a 2 (positivo)
  - in una codifica a K bit: **$C_K(x) = 2^K + x$**
- Esempio ($K = 4$, $2^K = 16$)
  - $x = +5$
  - $C_K(+5) = 16 + 5 = 21 \rightarrow$ ~~1~~0101
  - $x = -5$
  - $C_K(-5) = 16 - 5 = 11 \rightarrow$ 1011
- Osservazione
  - Anche in questo caso il primo bit indica il segno
    - 0 = positivo
    - 1 = negativo

Complemento a due (2)

- Algoritmo per calcolare la rappresentazione in complemento a 2 di un numero negativo:
  - si rappresenta il valore assoluto in binario
  - si invertono tutte le cifre (1->0 e viceversa)
  - si somma 1
- Esempio
  - 5 $\rightarrow$ 0101
  - -5 $\rightarrow$ 1011= 1010+1

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…0 è ancora 0…0
- Intervallo di rappresentazione con K bit
  - modulo e segno: $[\ -(2^{K-1}-1),\ +2^{K-1}-1\ ]$
  - complemento a uno: $[\ -(2^{K-1}-1),\ +2^{K-1}-1\ ]$
  - complemento a due: $[\ -2^{K-1},\ +2^{K-1}-1\ ]$

Operazioni algebriche: somma e sottrazione su interi

Somme fra "cifre":

0 + 0 = 0 **1 + 0 = 1** **0 + 1 = 1** **1 + 1 = 10**

← riporto **prestito →**

1		**1**			
	1	**0**	**1**	**1**	**0 +**
	1	**0**	**1**	**0**	**1 =**
1	**0**	**1**	**0**	**1**	**1**

		0	
1	**1**	**1**	**0 -**
0	**1**	**0**	**1 =**
1	**0**	**0**	**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 $\rightarrow$ **<12; 52>**
- Floating point (virgola mobile)
  - Un numero razionale e' rappresentato come un intero moltiplicato per una opportuna potenza di10, cioè con una coppia <mantissa, esponente>
  - $12{,}52 = 1252/100 = 1252 * 10^{-2}$ $\rightarrow$ **<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
  - Caratteri speciali, punteggiatura, a-z, A-Z, 0-9
  - Utilizza 7 bit (128 caratteri)
  - I codici ASCII estesi usano 8 bit (256 caratteri)
- Codifica UNICODE
  - 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

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 lunghezza fissa: memoria richiesta = 2 milioni 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 x 50% + 2 x 25% + 2 x 3 x 12.5%) x 10^6 = 1.75 milioni di bit
- Attenzione:
  - la nuova sequenza binaria deve essere decodificabile!

Codifica di dati multimediali

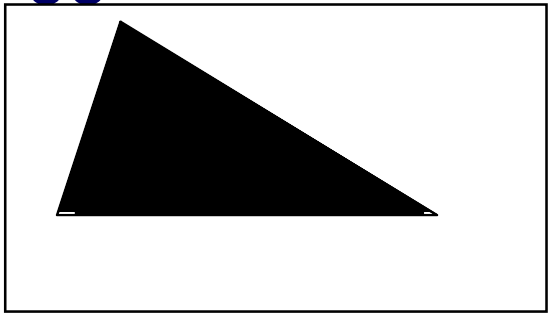
- Applicazioni multimediali
  - elaborano anche tipi di informazione differenti da testi e numeri
- Esempi di dati multimediali:
  - immagini
  - filmati
  - sequenze sonore

Codifica di immagini

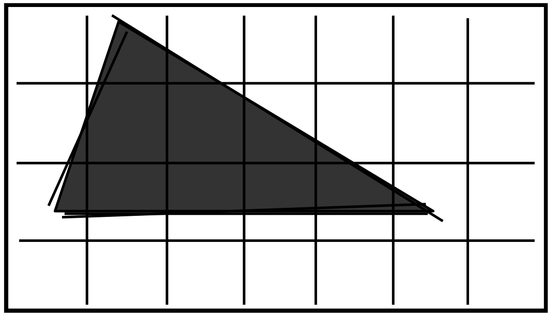
- Immagini digitalizzate = sequenze di bit!
  - L'immagine viene *discretizzata*, cioè rappresentata con sequenze di pixel
  - Ogni pixel ha associato un numero che descrive un particolare colore (o tonalità di grigio)

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

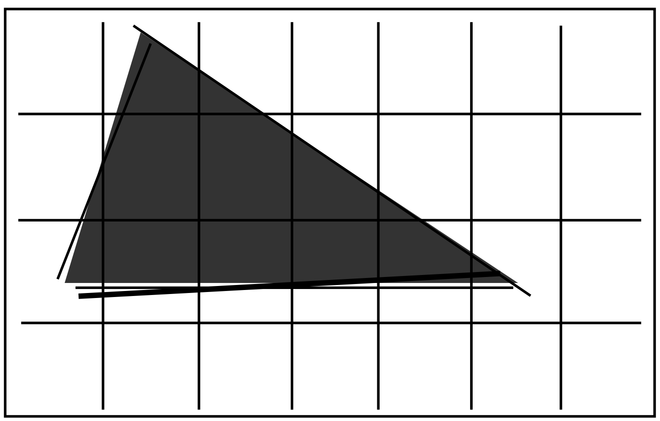


Codifica di immagini

- **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 bianco (o in cui il bianco è predominante)
  - il simbolo “**1**” viene utilizzato per la codifica di un pixel nero (o 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
  - Assumiamo che i pixel siano ordinati dal basso verso l'alto e da sinistra verso destra



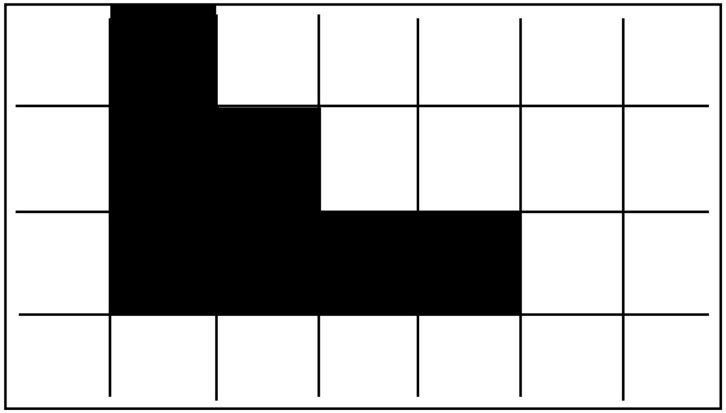
0_{22}	1_{23}	0_{24}	0_{25}	0_{26}	0_{27}	0_{28}
0_{15}	1_{16}	1_{17}	0_{18}	0_{19}	0_{20}	0_{21}
0_{8}	1_{9}	1_{10}	1_{11}	1_{12}	0_{13}	0_{14}
0_{1}	0_{2}	0_{3}	0_{4}	0_{5}	0_{6}	0_{7}

- La rappresentazione della figura è data dalla stringa:

0000000 0111100 0110000 0100000

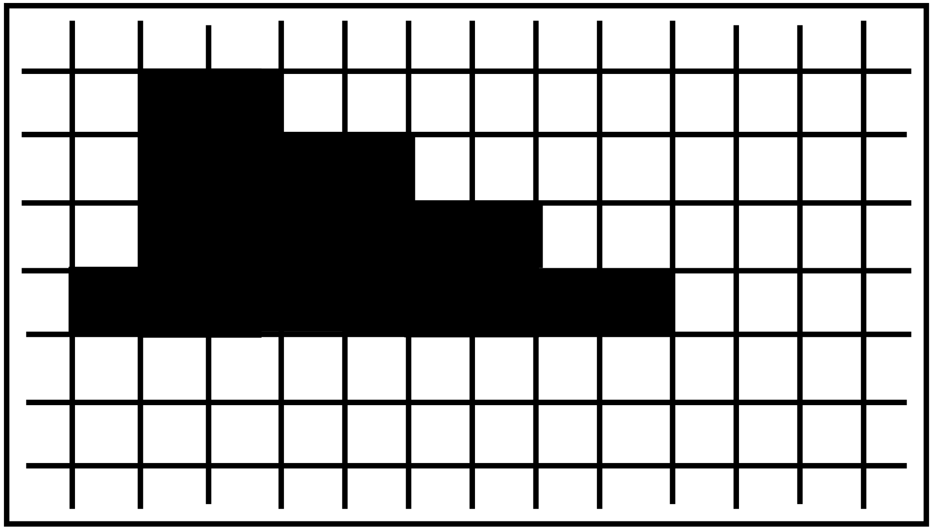
Codifica di immagini

- 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 $\mathbf{2^4=16}$ livelli di grigio
  - con **8** bit ne possiamo distinguere $\mathbf{2^8=256}$
  - con **K** bit ne possiamo distinguere $\mathbf{2^K}$

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 640×480 pixel richiederà 2457600 bit (307200 byte)
- Tipicamente
  - risoluzione: 1024×768, 1600×900, 1280×720 («HD-ready»), 1920×1080 («Full HD»), 3840x2160 («4K»)
  - numero di colori per pixel: da 256 fino a 16 milioni
- Tecniche di **compressione lossy**
  - riducono notevolmente lo spazio occupato dalle immagini

Compressione JPEG (esempio)

Codifica Bitmap

- **800 x 600**
- **16,8 mln colori (24 bit)**

dimensione = 1.440.000 byte

≈ 1406 KB

Fattore qualità 90% (253 KB)

Fattore qualità 10% (12 KB)

Fattore qualità 1% (9 KB)

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

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