Rappresentazione dell’informazione

Tipi di informazione

codifica

informazione

rappresentazione binaria

decodifica

Mondo esterno

Computer: memorizzazione, elaborazione

Rappresentazione delle informazioni

Idea di fondo

- usare presenza/assenza di carica elettrica
- usare passaggio/non passaggio di corrente/luce

Usiamo cioè una rappresentazione binaria (a due valori) dell'informazione

L'unità minimale di rappresentazione è il BIT (BInary digiT – cifra digitale): 0 o 1

Informazioni complesse

Con 1 bit rappresentiamo solo 2 diverse informazioni:

si / no - on / off - 0 / 1

Mettendo insieme più bit possiamo rappresentare più informazioni:

00 / 01 / 10 / 11

Informazioni complesse si memorizzano come sequenze di bit

Informazioni complesse

- Per codificare i nomi delle 4 stagioni bastano 2 bit

- Ad esempio:
  - 0 0 per rappresentare Inverno
  - 0 1 per rappresentare Primavera
  - 1 0 per rappresentare Estate
  - 1 1 per rappresentare Autunno

- Quanti bit per codificare i nomi dei giorni della settimana?

Informazioni complesse

In generale, con N bit, ognuno dei quali può assumere 2 valori, possiamo rappresentare 2^N informazioni diverse (tutte le possibili combinazioni di 0 e 1 su N posizioni)

viceversa

Per rappresentare M informazioni dobbiamo usare N bit, in modo che $2^N >= M$

Esempio

Per rappresentare 57 informazioni diverse dobbiamo usare gruppi di almeno 6 bit. Infatti:

$$2^6 = 64 > 57$$

Cioè un gruppo di 6 bit può assumere 64 configurazioni diverse:

000000 / 000001 / 000010 .../ 111110 / 111111

Informazioni

- Numeri
  - Interi positivi
  - Positivi e negativi
  - Reali
- Testi

Informazioni tradizionali

- Immagini fisse
  - Vettoriali
  - Bitmap
- Audio
- Video

Informazioni multimediali

Sistemi numerici

Sistemi numerici

- Per determinare un sistema numerico serve:
  - un insieme limitato di simboli (le ***cifre***), che rappresentano quantità prestabilite (1, 2, V, X, M)
  - le **regole** per costruire i numeri:
    - sistemi numerici posizionali
    - sistemi numerici non posizionali

Sistemi numerici

- Sistemi numerici non posizionali:
  - valore delle cifre è indipendente dalla posizione

- Sistemi numerici posizionali:
  - il valore delle cifre dipende dalla loro posizione all'interno del numero (ogni posizione ha un peso)

Sistemi numerici posizionali

- Esempio:

$$N = d_3\ d_2\ d_1\ d_0\ ;\ V(N) = d_3*p_3 + d_2*p_2 + d_1*p_1 + d_0*p_0$$

  - N → rappresentazione del numero
  - V(N) → valore del numero

- Sistemi a **base fissa**:
  - $p_i = r^i$ dove:
    - r è la base del sistema
    - d_i rappresentano le cifre

Sistema decimale

- È un sistema numerico posizionale a base fissa

- Il sistema decimale utilizza:
  - r = 10
  - d = 0,1,2,3,4,5,6,7,8,9

Sistema decimale

$$8427$$

$$=$$

$$8*10^3 + 4*10^2 + 2*10^1 + 7*10^0$$

Sistema binario

- Anche il sistema binario è un sistema numerico posizionale a base fissa

- Il sistema binario utilizza:
  - r = 2
  - d = 0,1

- Ogni cifra è detta **bit** (da **BI**nary digi**T**)

Sistema binario

$$1011 = 1*2^3 + 0*2^2 + 1*2^1 + 1*2^0 = 11_{10}$$

Somma binaria

- La tabella di definizione è:
  - 0 + 0 = 0
  - 0 + 1 = 1
  - 1 + 0 = 1
  - 1 + 1 = 0 con riporto di 1
  - 1 + 1 + 1 = 1 con riporto di 1
- Esempi

Moltiplicazione e divisione

- Si utilizzano le stesse procedure:
  - per la moltiplicazione: somma e scorrimento
  - per la divisione: differenza e scorrimento

- Shift a sinistra di n -> moltiplico per 2^n
- Shift a destra di n -> divisione intera per 2^n

Conversioni di base

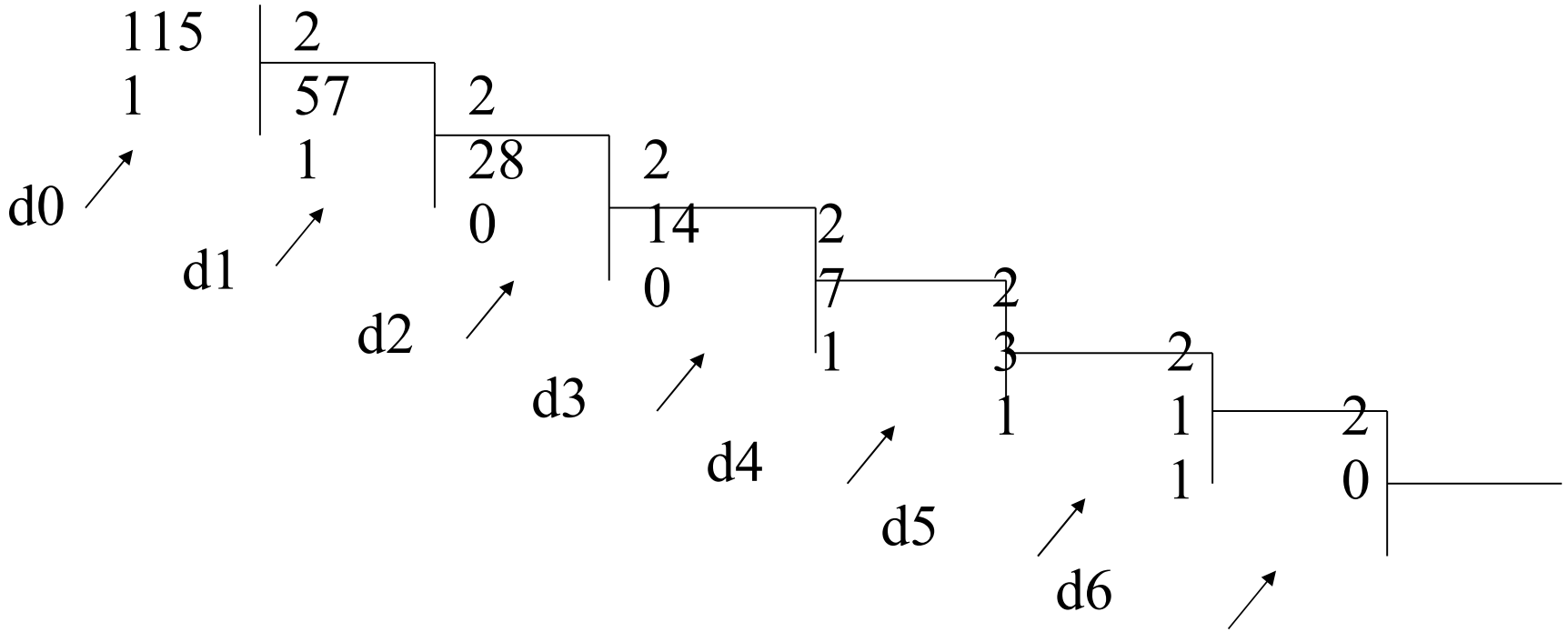
Dal sistema binario a quello decimale

- Utilizzando la definizione:
  - $1010_2 = (1*8 + 0*4 + 1*2 + 0*1)_{10} =$
    $= (8+2)_{10} = 10_{10}$
- Oppure si può utilizzare il seguente formato:
  - $N = ((d_{n-1}*r + d_{n-2})*r + d_{n-3}) \ldots)*r + d_0$

Conversioni di base

Dal sistema decimale a quello binario

- Esempio: $115_{10} = 1110011_{2}$:



Altri sistemi utilizzati

- Sistema ottale:
  - r = 8
  - d = 0,1,2,3,4,5,6,7
- Sistema esadecimale:
  - r = 16
  - d = 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F
- I suddetti sistemi assumono una importanza particolare nel mondo dei calcolatori !

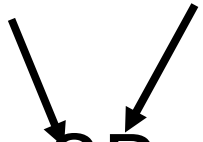
Conversione da binario ad esadecimale

- Esiste una corrispondenza diretta tra cifre esadecimali e il corrispondendte binario

10001011_2

$16 = 2^4$!!!

1000 1011



$8B_h$

Rappresentazione dei numeri nei calcolatori

Numero di cifre necessario

- Le macchine hanno vincoli spaziali:
  - è necessario conoscere il massimo valore rappresentabile:
  - con *n* bit si può rappresentare al massimo il numero 2^n -1
  - è facile determinare che per poter rappresentare fino ad X, sono necessari un numero n di bit pari a:

$$n = INT(\ \log_2 (X+1)\ )$$

Rappresentazione dei numeri nei calcolatori

- Esiste un limite al numero di bit impiegati per rappresentare un numero
- Tale limite dipende da:
  - intervallo di variabilità
  - occupazione di memoria
- Dato che la rappresentazione è formata da un numero finito di bit, se si supera tale limite si ha errore (*overflow*)

Numeri negativi

- Esistono diverse possibilità di rappresentazione:
  - **modulo e segno**
  - **complemento a 2**

Modulo e segno

- Convenzione per il bit più significativo:
  - 0 : segno positivo
  - 1 : segno negativo
- esistono due rappresentazioni per lo ‘0’

+ 5 ---> 00101
- 10 ---> 11010

+ 0 ---> 00000
- 0 ---> 10000

Complemento a 2 (complemento alla base)

- Dato X in base 2 di n cifre:

  $2^n - X$ (complemento a 2 del numero X)

- Se X=01011, e numero di cifre 5

  2^5-X = 100000 – 01011 = 10101

- ***Regoletta pratica***: il complemento a 2 si trova analizzando i bit del numero a partire da destra: si riportano invariati tutti gli zeri fino al primo bit a 1, si riporta invariato questo stesso bit a 1, si complementano (0→1, 1→0) tutti gli altri bit

Complemento a 2 (complemento alla base)

- complemento a 2:
  - per definizione il complemento a 2 di X è 2^n - X
  - unica rappresentazione dello ‘0’

- I numeri positivi hanno il bit più significativo (segno) posto a zero.
- I numeri negativi sono rappresentati dal complemento a 2 del corrispondente numero positivo, segno compreso. Pertanto, i numeri negativi hanno il bit più significativo sempre a 1.

  - Esempio: +3 <=> 00011
    - 3 <=> 11101

Uso dei numeri negativi

- Usando modulo e segno:
  - la somma algebrica di numeri positivi e negativi può generare problemi
  - servono sistemi hardware specifici per la gestione corretta del formato
  - E' necessario riconoscere il segno dal primo bit

Usando il complemento a due:

  - La sottrazione si esegue con una somma!

Numeri negativi: intervallo valori rappresentabili

Rappresentazione modulo e segno

$$-2^{n-1} + 1 \leq N \leq 2^{n-1} - 1$$

Rappresentazione in complemento a due

$$-2^{n-1} \leq N \leq 2^{n-1} - 1$$

Memorizzazione su calcolatore e codici

Memorizzazione su calcolatore

- L'unità atomica è il **bit** (BInary DigiT)
- L'insieme di 8 bit è detto **byte**
- **Word:** (tipicamente 16, 32 o 64bit): insieme di bit la cui dimensione è una importante caratteristica del calcolatore considerato. Infatti essa influenza:
  - La larghezza degli indirizzi
  - La dimensione dei registri del processore
  - Larghezza dei bus (word o multipli di essa)
  - **Double-word**: il doppio di una word

Intervalli di variabilità

- **bit**: Numero di configurazioni: 2
  intervallo di variabilità: *[0-1]*

- **byte**: Numero di configurazioni: 256
  intervallo di variabilità:
  dipende dal tipo di memorizzazione

Tipi di memorizzazione

- Modulo: 256 configurazioni,
  [0, 255]
- Modulo e segno: 256 configurazioni,
  [-127, +127]
- Complemento a 2: 256 configurazioni,
  [-128, +127]

Il Byte

- Una sequenza di 8 bit viene chiamata *Byte*
  - 0 0 0 0 0 0 0 0
  - 0 0 0 0 0 0 0 1
  - .....

byte = 8 bit = 2^8 = 256 informazioni diverse

Usato come unità di misura per indicare

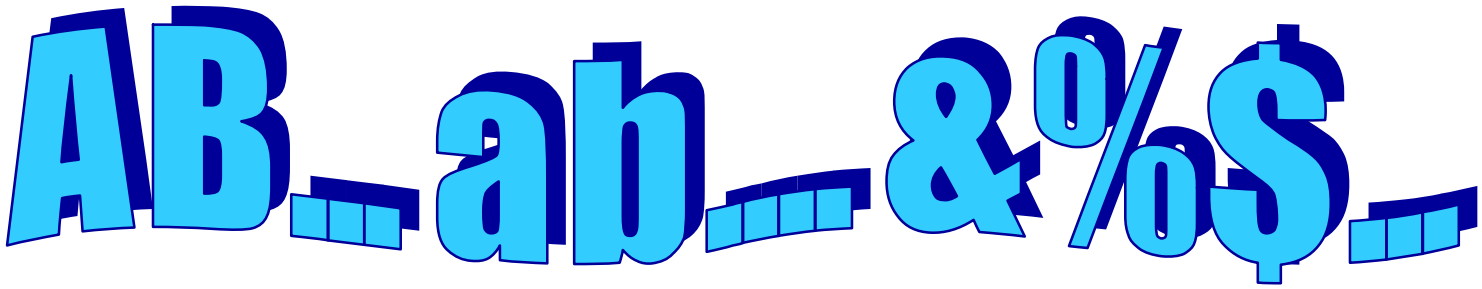
- le dimensioni della memoria
- la velocità di trasmissione
- la potenza di un elaboratore

Usando sequenze di byte (e quindi di bit) si possono rappresentare caratteri, numeri immagini, suoni.

Altre unità di misura

- KiloByte (KB), MegaByte (MB), GigaByte (GB)
- Per ragioni storiche in informatica Kilo, Mega, e Giga indicano però le *potenze di 2* che più si avvicinano alle corrispondenti potenze di 10
- Più precisamente
  - 1 KB = 1024 x 1 byte = $2^{10} \sim 10^3$ byte
  - 1 MB = 1024 x 1 KB = $2^{20} \sim 10^6$ byte
  - 1 GB = 1024 x 1 MB = $2^{30} \sim 10^9$ byte
- I multipli del byte vengono utilizzati come unità di misura per la capacità della memoria di un elaboratore

La Codifica dei Caratteri



Codici per i simboli dell'alfabeto

- **Per rappresentare i simboli dell'alfabeto anglosassone (0 1 2 … A B … A b …) bastano 7 bit**
  - **Nota: *B* e *b* sono simboli diversi**
  - **26 maiuscole + 26 minuscole + 10 cifre + 30 segni di interpunzione+… -> circa 120 oggetti**

- **Per l'alfabeto esteso con simboli quali &, %, $, … bastano 8 bit come nella codifica accettata universalmente chiamata ASCII esteso**
- **Per manipolare un numero maggiore di simboli si utilizza la codifica UNICODE a 16 bit**

Codifica ASCII

- La codifica **ASCII** (**A**merican **S**tandard **C**ode for **I**nterchange **C**ode) utilizza codici su 7 bit
  ($2^7 = 128$ caratteri diversi)
- Ad esempio
  - 1 0 0 0 0 0 1 rappresenta A
  - 1 0 0 0 0 1 0 rappresenta B
  - 10 0 0 0 1 1 rappresenta C

- Le parole si codificano utilizzando sequenze di byte
  - 1000010 1000001 1000010 1000001
    B A B A

Altri codici di codifica

- **ASCII ESTESO**
  - Usa anche il primo bit di ogni byte
  - 256 caratteri diversi
  - non è standard (cambia con la lingua usata)
- **UNICODE**
  - standard proposto a 16 bit (65.536 caratteri)
- **EBCDIC**
  - altro codice a 8 bit della IBM (quasi in disuso)

ASCII esteso

00000000	Null	00100000	Spc	01000000	@	01100000	`
00000001	Start of heading	00100001	!	01000001	A	01100001	a
00000010	Start of text	00100010	"	01000010	B	01100010	b
00000011	End of text	00100011	#	01000011	C	01100011	c
00000100	End of transmit	00100100	$	01000100	D	01100100	d
00000101	Enquiry	00100101	%	01000101	E	01100101	e
00000110	Acknowledge	00100110	&	01000110	F	01100110	f
00000111	Audible bell	00100111	'	01000111	G	01100111	g
00001000	Backspace	00101000	(	01001000	H	01101000	h
00001001	Horizontal tab	00101001	)	01001001	I	01101001	i
00001010	Line feed	00101010	*	01001010	J	01101010	j
00001011	Vertical tab	00101011	+	01001011	K	01101011	k
00001100	Form Feed	00101100	,	01001100	L	01101100	l
00001101	Carriage return	00101101	-	01001101	M	01101101	m
00001110	Shift out	00101110	.	01001110	N	01101110	n
00001111	Shift in	00101111	/	01001111	O	01101111	o
00010000	Data link escape	00110000	0	01010000	P	01110000	p
00010001	Device control 1	00110001	1	01010001	Q	01110001	q
00010010	Device control 2	00110010	2	01010010	R	01110010	r
00010011	Device control 3	00110011	3	01010011	S	01110011	s
00010100	Device control 4	00110100	4	01010100	T	01110100	t
00010101	Neg. acknowledge	00110101	5	01010101	U	01110101	u
00010110	Synchronous idle	00110110	6	01010110	V	01110110	v
00010111	End trans. block	00110111	7	01010111	W	01110111	w
00011000	Cancel	00111000	8	01011000	X	01111000	x
00011001	End of medium	00111001	9	01011001	Y	01111001	y
00011010	Substitution	00111010	:	01011010	Z	01111010	z
00011011	Escape	00111011	;	01011011	[	01111011	{
00011100	File separator	00111100	<	01011100	\	01111100	|
00011101	Group separator	00111101	=	01011101	]	01111101	}
00011110	Record Separator	00111110	>	01011110	^	01111110	~
00011111	Unit separator	00111111	?	01011111	_	01111111	Del

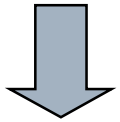
Codice ASCII

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Source: www.LookupTables.com

Numeri in ASCII

Le cifre 0..9 rappresentate in Ascii sono simboli o caratteri NON quantità numeriche



Non possiamo usarle per indicare quantità e per le operazioni aritmetiche. (Anche nella vita di tutti giorni usiamo i numeri come simboli e non come quantità: i n. telefonici)

Codice ASCII - note

- I caratteri alfabetici sono consecutivi e in ordine (alfabetico)
- La distanza tra una lettera minuscola e la corrispondente maiuscola è costante
- Le cifre decimali sono consecutive e in ordine
(da 0 a 9)

Codifica di immagini

Codifica di immagini

- Un’immagine è un insieme continuo di informazioni
  - A differenza delle cifre e dei caratteri alfanumerici, per le immagini non esiste un'unità minima di riferimento
- Problema: rendere digitale una informazione prettamente analogica

Codifica di immagini

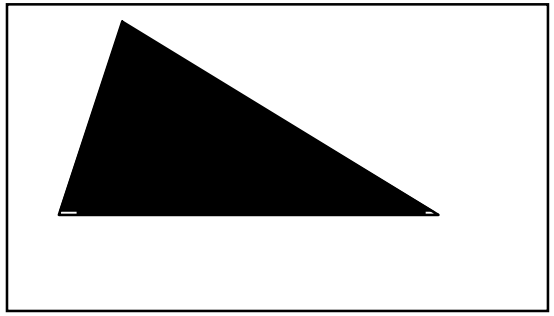
- **Esistono numerose tecniche per la memorizzazione digitale e l'elaborazione di un'immagine**
  - **una prevede la scomposizione dell'immagine in una *griglia* di tanti elementi (punti) che sono l'unità *minima* di memorizzazione;**
  - **La seconda strada prevede la presenza di strutture elementari di natura più complessa, quali *linee*, *circonferenze*, *archi*, etc.**

Codifica delle immagini B/N

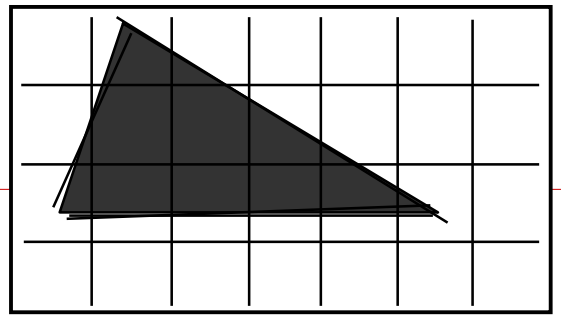
- Dividere l'immagine in una griglia a righe orizzontali e verticali
- Ogni quadratino della griglia è un **pixel** (picture element)
- Codificare ogni pixel con:
  - 0 se il pixel è bianco
  - 1 se il pixel è nero
- Convenire un ordinamento per i bit usati nella codifica

Codifica delle immagini B/N

- 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



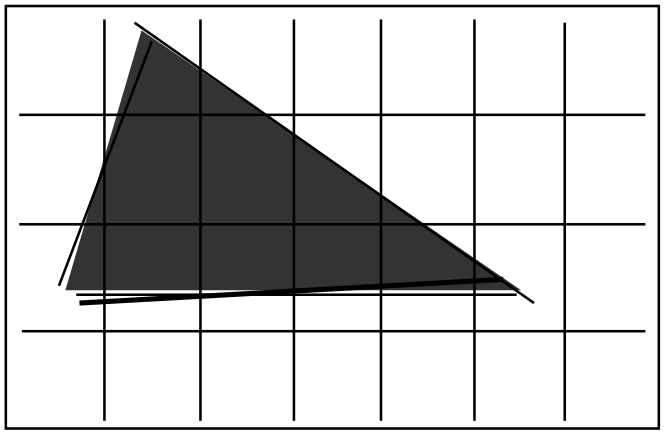
Codifica delle immagini B/N

- 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 delle immagini B/N

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

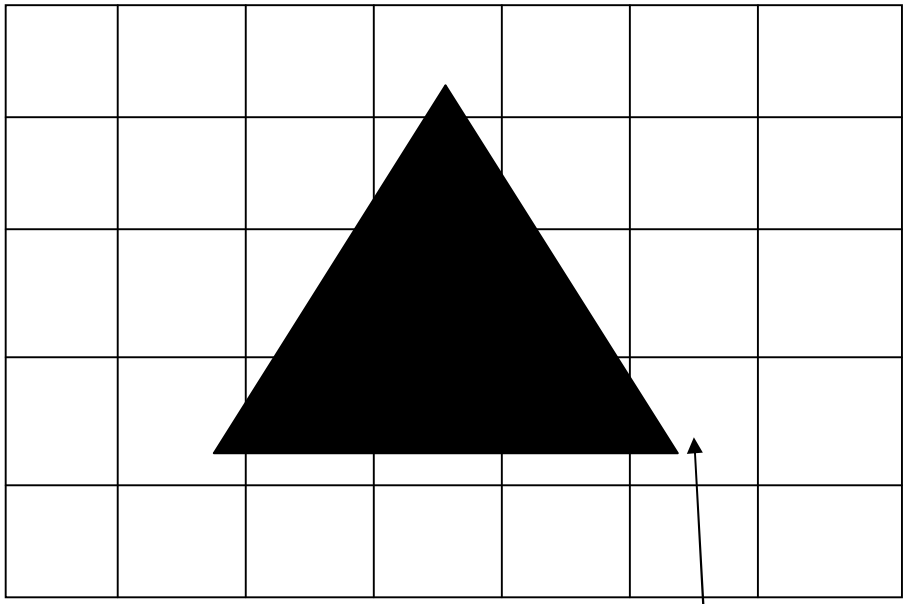


0_{22}	1_{23}	0_{24}	0_{25}	0_{26}	0_{27}	0_{28}
0_{15}	1_{16}	1_{17}	0_{18}	0_{19}	0_{20}	0_{21}
0_{8}	1_{9}	1_{10}	1_{11}	1_{12}	0_{13}	0_{14}
0_{1}	0_{2}	0_{3}	0_{4}	0_{5}	0_{6}	0_{7}

La rappresentazione della figura è data dalla stringa binaria

0000000 0111100 0110000 0100000

Codifica di un'immagine B/N



Pixel = 1

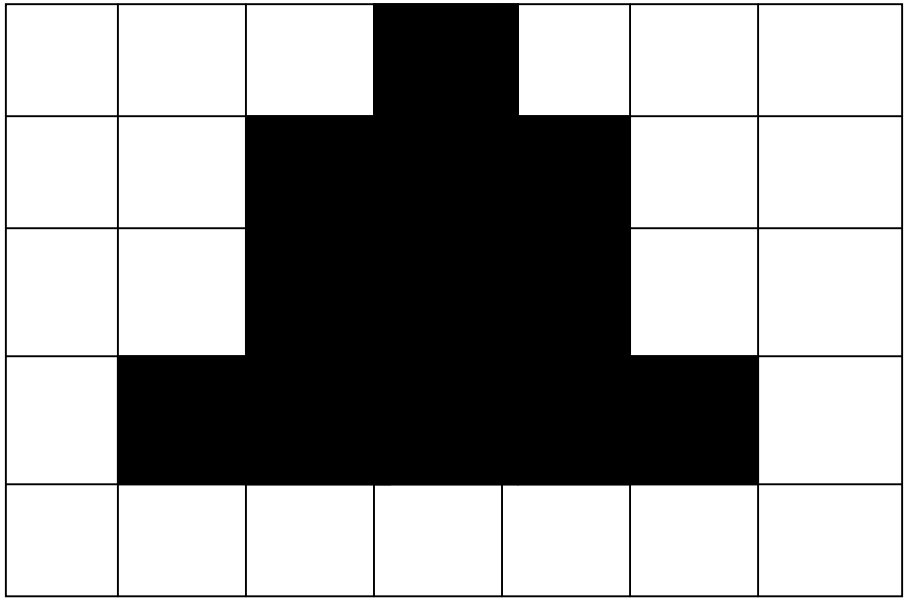
0 0 0 1 0 0 0
0 0 1 1 1 0 0
0 0 1 1 1 0 0
0 1 1 1 1 1 0
0 0 0 0 0 0 0

codifica

Decodifica

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

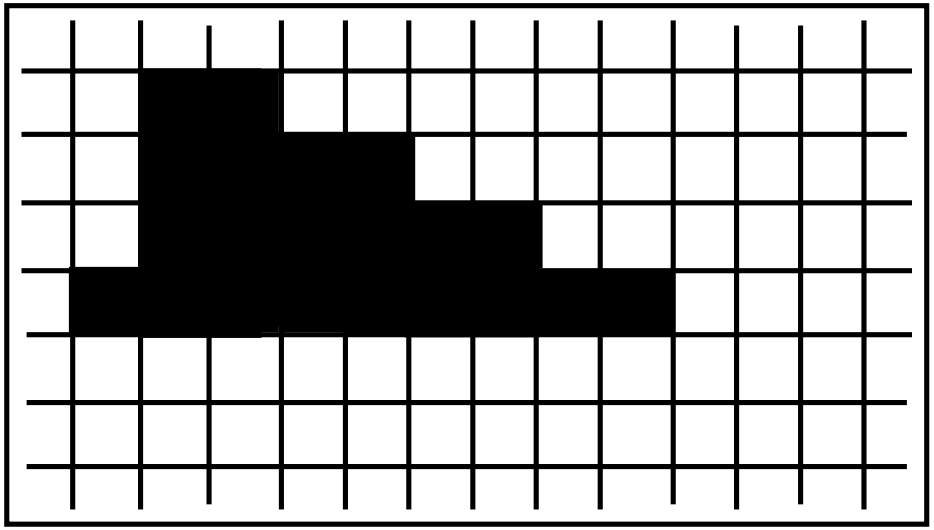
Codifica



Immagine

Codifica delle immagini B/N

- **Non sempre il contorno della figura coincide con le linee della griglia**
  - **nella codifica si ottiene un'approssimazione della figura originaria**
- **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**



Codifica delle immagini B/N

Quindi: le immagini sono rappresentate con un certo livello di approssimazione, o meglio, di risoluzione, ossia il numero di pixel usati per riprodurre l'immagine.

Risoluzioni tipiche

- 640 x 480 pixel; 800 x 600 pixel
- 1024 x 768 pixel; 1280 x 1024 pixel

Immagini in toni di grigio

- Le immagini in bianco e nero hanno delle sfumature, o **livelli di intensità di grigio**

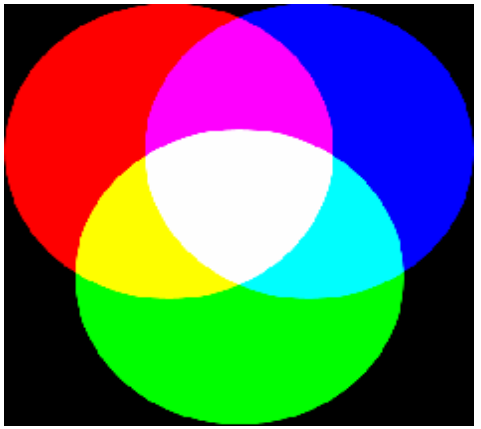
- Per codificare immagini con sfumature:
  - si fissa un insieme di livelli (*toni*) di grigio, cui si assegna convenzionalmente una rappresentazione binaria
  - per ogni pixel si stabilisce il livello medio di grigio e si memorizza la codifica corrispondente a tale livello

- Per memorizzare un pixel non è più sufficiente 1 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 e rappresentarle mediante una opportuna sequenza di bit

- Nella codifica RGB si utilizzano tre colori
  - rosso (Red), verde (Green) e blu (Blue)



- Ad ogni colore si associa un certo numero di sfumature codificate su N bit
  (2^N possibili sfumature)

- Esempio
  - con 2 bit per colore si ottengono 4 sfumature per colore
  - con 8 bit per colore si ottengono 256 sfumature per colore e 256^3(16 milioni) possibili colori

Immagini a colori

- La qualità dell'immagine dipende
  - dal numero di punti in cui viene suddivisa (*risoluzione*)
  - dai toni di colore permessi dalla codifica;

Bitmap

- La rappresentazione di un'immagine mediante la codifica a pixel viene chiamata **bitmap**
- Il numero di byte richiesti per memorizzare un bitmap dipende dalla risoluzione e dal numero di colori
- Esempio
  - se la risoluzione è 640x480 con 256 colori occorrono 2.457.600 bit = 300 KB

Bitmap

- I formati bitmap più conosciuti sono
  - BITMAP (.bmp),
  - GIF (.gif),
  - JPEG (.jpg)
  - TIFF (.tiff)
- In tali formati si utilizzano metodi di *compressione* per ridurre lo spazio di memorizzazione
  - Aree dello stesso colore si rappresentano in modo "abbreviato".
- E' in genere possibile passare da un formato ad un altro

Codifica vettoriale delle immagini

- Si utilizza quando le immagini da memorizzare hanno caratteristiche geometriche ben definite
- Il disegno da memorizzare può essere facilmente *scomposto in elementi base* come una linea o un arco di circonferenza
- La memorizzazione dell'intera immagine avviene tramite la codifica di ogni singola parte

Codifica vettoriale delle immagini

- Richiede poco spazio
- Per definire un segmento basteranno le coordinate dei due estremi (Linea dal punto <10;12> a <20; 30>)
- Il formato più diffuso è il PostScript (ps, eps)
  - usato anche per la stampa dei testi
- Altri formati: wmf, cdr (CorelDraw)

Codifica dei filmati

- **Immagini in movimento sono memorizzate come sequenze di fotogrammi**
  - **Si sfrutta la limitatezza della capacità percettiva dell'occhio umano**
  - **la sequenza continua di immagini viene *discretizzata* ottenendo una serie di immagini (frame) che variano velocemente, ma a intervalli stabiliti**

- **In genere si tratta di sequenze compresse di immagini**
  - **ad esempio si possono registrare solo le variazioni tra un fotogramma e l'altro**

Codifica dei filmati

- **Esistono vari formati (comprendente il sonoro):**
  - ***mpeg*** **(il piu' usato)**
  - ***avi*** **(microsoft)**
  - ***quicktime*** **(apple)**
  - ***mov***
- **E' possibile ritoccare i singoli fotogrammi**

Codifica dei suoni

- Si effettuano dei campionamenti su dati analogici
  - L'onda sonora viene misurata (campionata) ad intervalli regolari
- Si rappresentano i valori campionati con valori digitali
- La frequenza del campionamento determina la fedeltà della riproduzione del suono
  - Minore è l'intervallo di campionamento e maggiore è la qualità del suono

CD musicali: 44000 campionamenti al secondo, 16 bit per campione

Codifica dei suoni

Alcuni formati:

.mov

.wav

.mpeg

.avi

.midi - usato per l'elaborazione della musica al computer