Informatica

Terzo anno

Prof. A. Longheu

RAPPRESENTAZIONE DELL'INFORMAZIONE

- Per poter rappresentare le informazioni è necessario **codificare** le informazioni.
- I **simboli** per la codifica possono essere rappresentati da:
  una tensione elettrica {Vhigh (alta), Vlow (bassa)};
  un differente stato di polarizzazione magnetica (positiva, negativa);
  luce e buio.
- I valori ad essi associati sono 0 e 1 e vengono detti bit (binary digit); tutti i dati (numeri, parole, in generale oggetti) sono rappresentati da sequenze di **bit**.
- Per cui ogni informazione viene rappresentata nel calcolatore in forma binaria.
- Il motivo per cui il calcolatore utilizza i valori 0 e 1 deriva dal fatto che i transistor, con cui ancora oggi sono realizzati tutti i dispositivi elettronici, sono più facili da gestire se "solo" accesi o spenti piuttosto che cono differenti livelli di tensione o corrente

Informazioni complesse

Con 1 bit rappresentiamo solo 2 diverse informazioni:

si/no - on/off - 0/1

Mettendo insieme piu' bit possiamo rappresentare piu' informazioni:

00 - 01 - 10 - 11

Informazioni complesse

*In generale, con **N** bit, ognuno dei quali puo' assumere **2** valori, possiamo rappresentare 2^N informazioni diverse*

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:

$2^6 = 64 > 57$

Cioe' un gruppo di 6 bit puo' assumere 64 configurazioni diverse:

000000 / 000001 / 000010 …/ 111110 / 111111

Il byte:

In informatica ha assunto particolare importanza il concetto di:

byte = 8 bit = 2^8 = 256 inf. diverse

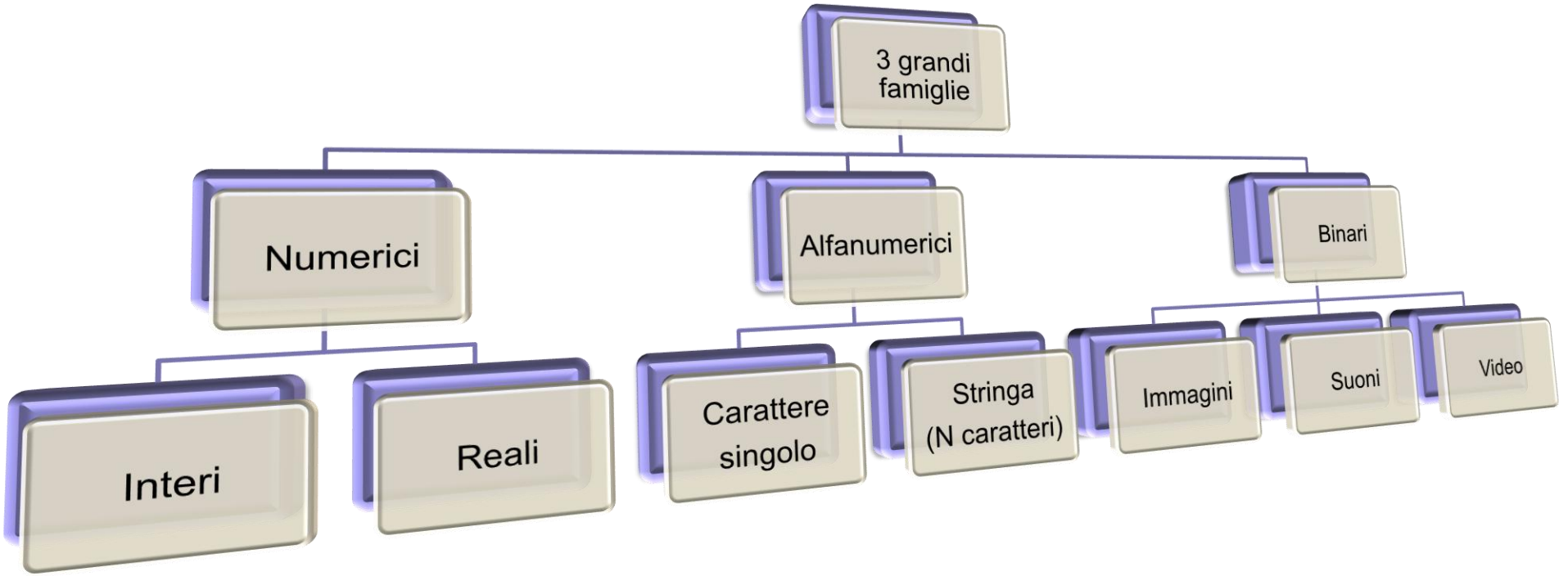
Il byte e' usato come unita' di misura per indicare le dimensioni della memoria, la velocita' di trasmissione, la potenza di un elaboratore

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

bit, Byte, KiloByte, MegaByte, …

bit = solo due stati, “0” oppure “1”.
Byte = 8 bit, quindi 28 = 256 stati
KiloByte [**KB**] = 2^{10} Byte = 1024 Byte ~ 10^3 Byte
MegaByte [**MB**] = 2^{20} Byte = 1 048 576 Byte ~ 10^6 Byte
GigaByte [**GB**] = 2^{30} Byte ~ 10^9 Byte
TeraByte [**TB**] = 2^{40} Byte ~ 10^{12} Byte
PetaByte [**PB**] = 2^{50} Byte ~ 10^{15} Byte
ExaByte [**EB**] = 2^{60} Byte ~ 10^{18} Byte

Tipi di dato


3 grandi famiglie
Numerici
Alfanumerici
Binari
Interi
Reali
Carattere singolo
Stringa (N caratteri)
Immagini
Suoni
Video

Numeri Interi

- I numeri interi relativi possono essere rappresentati in due modi:

  modulo e segno

  complemento a due.

Modulo e segno

- Per rappresentare un numero intero con segno utilizzando N bit, si usa il primo bit per rappresentare il segno (0-positivo, 1-negativo), il resto dei bit per il modulo (il numero senza segno).

  Esempio: -37 su 16 bit

  1 000000000010101

- Date N cifre potro' rappresentare i numeri nell'intervallo

$$(2^{N-1}-1) \qquad +(2^{N-1}-1)$$

Il complemento a 2

- Metodi per calcolare la rappresentazione in complemento a due di –X a partire da quella di X
  - Effettuare il complemento di ogni bit di X e aggiungere poi 1
    - rappresentazione di $+6_{dieci} = 0110_{C2}$
    - complemento di tutti i bit $\Rightarrow 1001_{C2}$
    - aggiungere 1 $\Rightarrow 1010_{C2}$ (che corrisponde a -6_{dieci})
  - Partendo da destra e andando verso sinistra, lasciare invariati tutti i bit fino al primo 1 compreso, complementare tutti gli altri bit.
    - rappresentazione di $+6_{dieci} = 0110_{C2}$ (NB ci vogliono 4 bit!!)
    - gli ultimi due bit (_._.**1**.**0**) rimangono invariati
    - gli altri due bit vengono complementati (**1**.**0**.**1**.**0**$_{C2}$)

Complemento a due

- Dato un numero binario di N bit, il complemento a 2 di tale numero si ottiene tramite il seguente algoritmo:
  - si procede dal bit meno significativo verso quello più significativo
  - se si incontrano tutti bit 0, essi vengono lasciati inalterati
  - se si incontra il primo bit 1, anche esso viene lasciato inalterato
  - tutti i bit successivi al primo bit 1, vengono invertiti (0 diviene 1, e viceversa)

Esempio

- **Esempio1**: si determini il complemento a 2 del numero 10100.

  Tutti i bit 0 a partire dal bit meno significativo sono lasciati inalterati e così anche il primo bit 1.

  Tutti gli altri bit vengono invertiti, ottenendo: 01100.

- **Esempio**: si determini il complemento a 2 del numero 01101001.

  In questo caso non esistono bit 0 a partire dal bit meno significativo. Sono il primo bit 1 viene lasciato inalterato. Gli altri vengono invertiti, ottenendo: 10010111.

Rappresentazione in complemento a due di un numero relativo

- La rappresentazione in complemento a 2 di un numero intero relativo, si effettua nella seguente maniera:

  i numeri interi positivi sono rappresentati in modulo e segno (con il bit di segno = 0)

  i numeri interi negativi sono rappresentati realizzando il complemento a 2 del numero intero in valore assoluto

Esempi

- **Esempio:** si voglia convertire il numero 105 con 8 bit

  Essendo il numero positivo, la sua rappresentazione in complemento a 2 coincide con quella in modulo e segno, pari a 01101001

- **Esempio:** si voglia convertire il numero -105 con 8 bit

  Essendo il numero negativo, la sua rappresentazione in complemento a due si ottiene determinando il complemento a 2 del numero binario corrispondente al valore assoluto (105), ossia di 01101001. Il complemento a 2 è 10010111, che è la rappresentazione di -105.

Perchè la Rappresentazione in Complemento a 2 è più Conveniente ?

- La rappresentazione in complemento a 2 viene utilizzata maggiormente rispetto quella modulo e segno per differenti motivi.

  Il più rilevante è relativo ai vantaggi ottenibili nell'esecuzione di operazioni elementari come la somma e la sottrazione. Queste due operazioni sono quelle che vengono più frequentemente realizzate in un computer, e, dunque, un risparmio nel tempo necessario alla loro esecuzione comporta un indiscusso aumento delle prestazioni di un computer.

Infatti…..

Addizioni e Sottrazioni tra numeri binari in Rappresentazione Modulo e Segno.

- Dati due numeri binari in rappresentazione modulo e segno, l'operazione di addizione può avvenire solo se i due segni sono gli stessi, ossia il bit più significativo è per entrambi i numeri uguale a 0 o a 1. In tal caso la somma tra i due numeri con stesso segno viene realizzata con le regole dell'addizione applicate a tutti i bit tranne il bit di segno. Il numero binario risultante sarà ottenuto aggiungendo il bit di segno ai bit ottenuti dalla somma.
- Dati due numeri binari in rappresentazione modulo e segno, l'operazione di sottrazione può avvenire solo se i due segni sono diversi. In tal caso la sottrazione tra i due numeri viene realizzata con le regole della sottrazione applicate a tutti i bit tranne il bit di segno, sottraendo il numero più piccolo in valore assoluto al numero più grande in valore assoluto. Il numero binario risultante sarà ottenuto aggiungendo ai bit ottenuti dalla sottrazione il bit di segno del numero in valore assoluto più grande.

Addizioni e Sottrazioni tra numeri binari in Rappresentazione in Complemento a Due.

- Dati due numeri binari in complemento a due, sia l'operazione di addizione che quella di sottrazione avviene semplicemente applicando le regole dell'addizione a tutti i bit compreso il bit di segno.

Esempi

- **Esempio:** Siano dati i numeri a 4 bit 0010 (+2) e 1010 (-6). Si voglia determinare la sottrazione.

  A tal fine basta applicare semplicemente le regole dell'addizione ai bit:

  0010+
  1010=

  _____
  1100 (-4)

  Il numero binario risultante è già il risultato con il segno giusto.

- **Esempio:** Siano dati i numeri in complemento a 2 a 6 bit 001100 (+12) e 100000 (-32). Si voglia determinare la sottrazione.

  A tal fine basta applicare semplicemente le regole dell'addizione ai bit:

  001100+
  100000=

  _____
  101100 (-20)

  Il numero binario risultante è già il risultato con il segno giusto.

Numeri Reali

- I numeri reali possono essere rappresentati con due modalità:

  virgola fissa

  virgola mobile.

Virgola fissa

- La rappresentazione in virgola fissa consiste nel rappresentare un numero reale con segno tramite N bit, supponendo fissa la posizione della virgola.
- In un numero rappresentato in virgola fissa a N bits, viene utilizzato
  - un bit per il segno
  - I bits per rappresentare la parte intera
  - D bits per rappresentare la parte decimale
- (ovviamente sarà N = I + D +1).

RAPPRESENTAZIONE DI NUMERI FRAZIONARI IN VIRGOLA FISSA

Un numero frazionario e' rappresentato come una coppia di numeri interi: la parte intera e la parte decimale.

12,52 <12; 52>

<1100; 110100>

Rappresentazione in Virgola Mobile.

- In un numero rappresentato in virgola mobile vengono stabiliti un certo numero di bit assegnati per codificare il segno (s), la mantissa (m) ed un certo numero di bit per codificare l'esponente (e).
- A differenza degli interi, non esistono convenzioni adottate universalmente, una proposta di standard è la IEEE-754

Esempio

- A titolo di esempio si consideri la seguente rappresentazione. Il bit più significativo è il bit del segno (s), mentre l'esponente è stato rappresentato con 7 bits (e) e la mantissa con i restanti 24 bits (m).

  s (31 bit)e(30...24 bit)m(23...0 bit)

- Il numero occupa in tutto 32 bit.
- La mantissa è rappresentata in modulo (bits 0-23) e segno (bit 31).
- L'esponente è rappresentato in 7 bits in eccesso 64 cioè, il vero esponente si ottiene dalla sua rappresentazione sottraendo 64. Ad esempio:

  0000000 corrisponde ad un esponente -64 (perchè 0-64=-64)

  1111111 corrisponde ad un esponente +63 (perchè 127-64=63)

Numero reale

- Si procede nel seguente modo:

  si normalizza il numero in modo che la parte intera sia 0. La normalizzazione si ottiene moltiplicando o dividendo il numero per la minima potenza di 10, in modo che la parte intera ottenuta dalla divisione sia 0. La potenza di 10 così trovata rappresenta l'esponente della codifica binaria (e). In base al fatto che si è operata una moltiplicazione o una divisione, l'esponente può essere positivo o negativo.

  si converte il numero ottenuto (quello con la parte intera pari a 0) utilizzando la rappresentazione a Virgola Fissa in cui solo 1 bit è utilizzato per codificare la parte intera mentre tutti gli altri sono utilizzati per codificare la parte razionale.

  Si codifica in binario l'esponente trovato al passo 1, ricordando che l'esponente è in eccesso 64.

  Si mettono assieme segno, esponente e mantissa trovati ai passi precedenti.

Esempio

- Si converta il numero N=18.75 nella rappresentazione binaria in virgola mobile. Si procede nel seguente modo:

  si normalizza il numero in modo che la parte intera si 0. Nel nostro caso occorre spostare la virgola di 2 posizioni verso sinistra, ossia dividere per 100 (102). Quindi si ottiene:

  N=0.1875 e=-2

- si converte il numero N ottenuto al passo 1 utilizzando la rappresentazione a Virgola Fissa in cui solo 1 bit è utilizzato per codificare la parte intera mentre tutti gli altri sono utilizzati per codificare la parte razionale.
- Nel nostro caso: il numero binario in virgola fissa è 0011.
- Si codifica in binario l'esponente. Ricordando che l'esponente è in eccesso 64 deve essere: e - 64, cioè e=-66. La sua rappresentazione in binario su 7 bits è 1000010
- In definitiva si ha: 18.75 = 01000010000011000000.....000

Problemi con la rappresentazione dei numeri

- In matematica, i numeri possono essere infinitamente grandi o infinitamente piccoli
- In informatica NO, perché un numero può occupare N bit, che anche per N grande è sempre una dimensione finita, e di contro se un numero è molto piccolo, la macchina potrebbe approssimarlo a zero
- **Overflow** è l'errore che si verifica se il risultato di un calcolo eccede la capacità massima della macchina; il contrario è un errore (per noi umani) chiamato **underflow**
- Occorre quindi pensare da informatico e non da matematico, qualche esempio:
  - 0,00000000000000000000000000000001 potrebbe diventare zero, quindi occhio se è usato come divisore
  - (X/Y)*Y potrebbe non restituire X se X e Y sono con la virgola e troppo grandi o troppo piccoli

Codici Alfanumerici

- I simboli che vengono usati per rappresentare testi sono, come noto, i caratteri alfanumerici, cioè l'insieme costituito
  - lettere dell'alfabeto
  - dieci cifre decimali.
  - simboli come lo spazio
  - i segni di interpunzione
  - i simboli per indicare il passaggio alla riga o alla pagina successiva, ecc.
- Questo insieme di caratteri alfanumerici può essere facilmente rappresentato attribuendo in maniera univoca a ciascuno dei suoi elementi un numero intero (codice).

Codifica ASCII

- La codifica ASCII (che si pronuncia ASKI), prende il nome da **American Standard Code for Information Interchange**.
- Utilizza 7 bit per un totale di 128 simboli rappresentabili.
- Da notare che i caratteri dell'alfabeto e le cifre numeriche successive hanno codice anch'esso successivo (ad esempio a ha codice 97, b codice 98, c codice 99, il numero 0 ha codice 48, il numero 1 codice 49, etc.)
- Tra le più utilizzate codifiche ASCII (entro i primi 128 simboli) vi sono:
  - ~ (tilde) codice 126
  - { codice 123
  - } codice 125
- I caratteri di controllo (non riproducibili) hanno i codici più bassi.
- Il blank (Spazio) è il primo dei caratteri riproducibili.
- Le maiuscole/minuscole sono ordinate (codice Progressivo).

LA CODIFICA DEI CARATTERI

E' necessario convenire un codice per rappresentare i caratteri.
*Il codice **ASCII** (**A**merican **S**tandard **C**ode for **I**nterchange **C**ode) usa i primi 7 bit di ogni byte:*

$$2^7 = 128 \text{ caratteri diversi}$$

Sufficienti per l'alfabeto anglosassone

Codifiche derivate dalla codifica ASCII

- Esistono numerose estensioni della codifica ASCII. Tali estensioni derivano dalla necessità di codificare simboli legati a particolari lingue, e dal fatto che operando su 8 bit, la codifica ASCII consente l'utilizzo dell'ottavo bit, lasciando gli altri 7 inalterati.
- Tutte le estensioni della codifica ASCII non modificano tale codifica ma aggiungono semplicemente altri 128 simboli.
- Tra le estensioni più diffuse vi è la ISO Latin 1.

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

Codifica Unicode

- La codifica Unicode supera i limiti della codifica ASCII e relativi derivati, in quanto estende il numero di simboli codificabili.
- Utilizza 16 bit nella prima versione (UTF-16) e 32 bit in quella più recente (UTF-32)
- Attualmente sono più di 95.000 simboli codificati con lo standard Unicode.
- Essi sono divisi in:
  - Script Moderni: Latino, Greco, Giapponese, Cinese, Koreano, etc.
  - Script Antichi: Sumero, Egiziano, etc.
  - Segni Speciali
- La codifica Unicode è in ogni caso compatibile con la codifica ASCII

Caratteri ammessi

- I linguaggi moderni usano caratteri codificati con un codice a 16 bit chiamato Unicode
- il primo codice ASCII è a 7 bit (ISO 646) e contiene tutti i caratteri "standard" della lingua inglese; con il bit 8, si hanno altri 128 caratteri, scelti in base alla lingua da supportare, dando luogo agli standard ISO 8859, ad esempio ISO8859-1 è per l'Europa occidentale e contiene caratteri come è,ò,à,ù,ì,ç...;
- utilizzando 16 bit si possono rappresentare insiemi di caratteri fonetici e ideogrammi (indispensabile per cinese e giapponese); si ottiene il codice UNICODE a 16 bit;
- i primi 128 caratteri sono identici all'ISO 646 e i primi 256 sono gli stessi dell'ISO 8859-1; utilizzando infine 32 bit, si ottiene lo standard ISO 10646, che raccoglie i simboli utilizzati da tutte le lingue

NUMERI IN ASCII

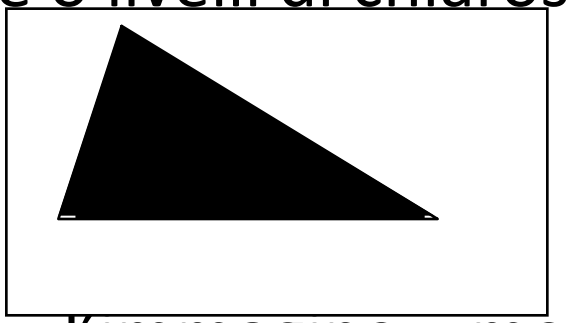
*Le cifre 0..9 rappresentate in Ascii sono simboli o caratteri **NON** quantita' numeriche*
Non possiamo usarle per indicare quantita' e per le operazioni aritmetiche. (Anche nella vita di tutti giorni usiamo i numeri come simboli e non come quantita': i n. telefonici, i cap…)

CODIFICA DI IMMAGINI

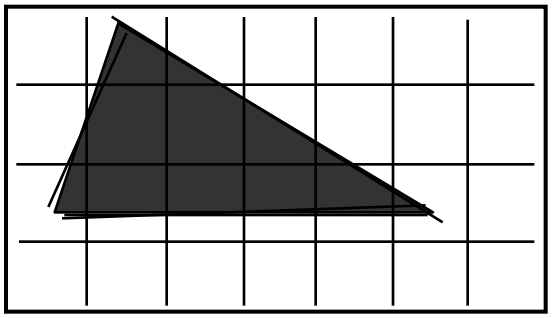
- Esistono numerose tecniche per la memorizzazione digitale e l'elaborazione di un'immagine
- Immagini = sequenze di bit!
- L’immagine viene digitalizzata cioè rappresentata con sequenze di pixel
- Ogni pixel ha associato un numero che descrive un particolare colore (o tonalità di grigio)
- Inoltre si mantengono la dimensione, la risoluzione (numero di punti per pollice), e il numero di colori utilizzati

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



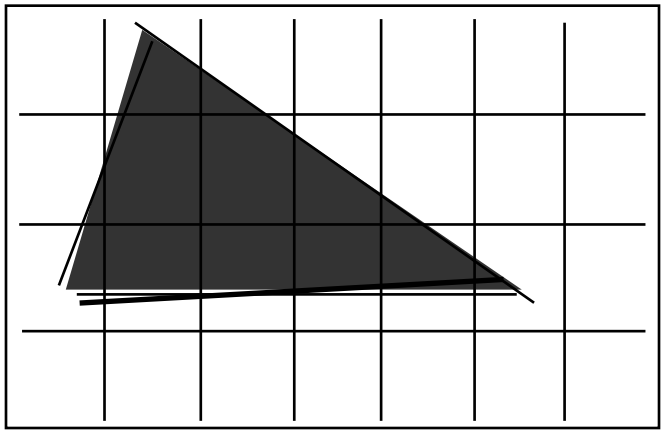
CODIFICA DI IMMAGINI

- 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

Hp: assumiamo che i pixel siano ordinati dal basso verso l'alto e da sinistra verso destra



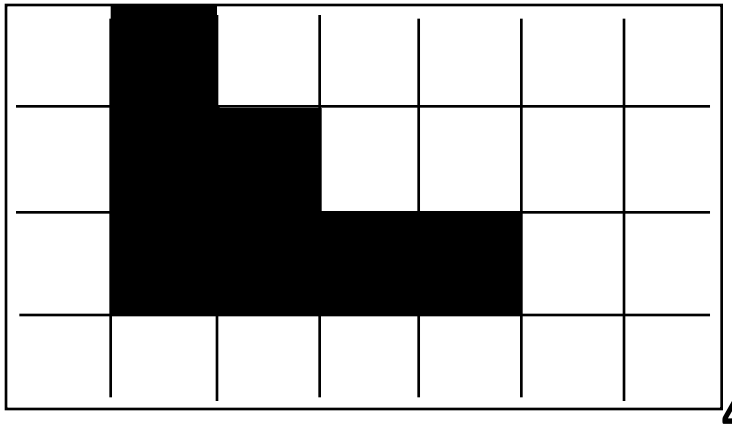
0 22	1 23	0 24	0 25	0 26	0 27	0 28
0 15	1 16	1 17	0 18	0 19	0 20	0 21
0 8	1 9	1 10	1 11	1 12	0 13	0 14
0 1	0 2	0 3	0 4	0 5	0 6	0 7

La rappresentazione della figura è data dalla stringa binaria

0000000 0111100 0110000 0100000

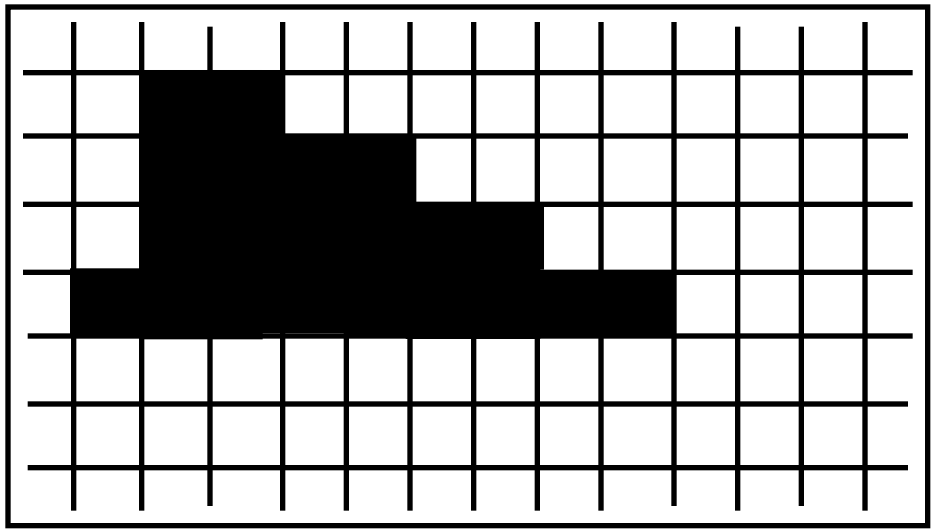
CODIFICA DI IMMAGINI

- Non sempre il contorno della figura coincide con le linee della griglia: nella codifica si ottiene un'approssimazione della figura originaria
- Se riconvertiamo la stringa
  00000000111100011000001000000
  in immagine otteniamo



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



CODIFICA DELLE IMMAGINI

***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:*

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

*N.B. Ha importanza anche il **dot pitch**, il grado di definizione del pixel: 0,25 - 0,28)*

CODIFICA DI IMMAGINI CON 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

CODIFICA DI 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

- La rappresentazione di un'immagine mediante la codifica dei pixel, viene chiamata **codifica bitmap**

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
- Tecniche di **compressione** consentono di ridurre notevolmente lo spazio occupato dalle immagini

CODIFICA DELLE IMMAGINI

- *E' possibile risparmiare spazio nella rappr. delle immagini: Aree dello stesso colore si rappresentano in modo "abbreviato".*
- *Esistono vari formati di codifica:*
  (bmp, gif, jpg, pict, tiff)
- *E' in genere possibile passare da un formato ad un altro.*

CODIFICA VETTORIALE DELLE IMMAGINI

- *Istruzioni su come disegnare l'immagine*

p.e.: Linea dal punto <10;12> a <20; 30>

- *Idonea per immagini di tipo geometrico (p.e., progetti ingegneristici).*
- *Richiede poco spazio.*
- *Il formato piu` diffuso e` il PostScript (ps, eps) - anche per la stampa dei testi*
- *Altri formati: wmf, cdr (CorelDraw)*

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

CODIFICA DI SUONI

- L'onda sonora viene misurata (campionata) ad intervalli regolari
- Minore e l'intervallo di campionamento e maggiore e la qualità del suono
- CD musicali: 44000 campionamenti al secondo, 16 bit per campione.
- Alcuni formati:
  - *.mov*, *.wav*, *.mpeg*, *.avi*
  - ***.midi*** usato per l'elaborazione della musica al PC

CODIFICA DEI SUONI

L'onda sonora viene misurata (campionata) ad intervalli regolari

Minore e l'intervallo di campionamento e maggiore e la qualita' del suono

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

CODIFICA DEI SUONI

Alcuni formati:

- ---.mov
- ---.wav
- ---.mpeg, ---.avi,
- formato **midi** usato per l'elaborazione della musica al computer