

Rappresentazione e Codifica dell'Informazione

Capitolo 1

Il concetto di Informazione

- *Informazione*
 - deriva da *informare*, ossia dare forma
 - ... fa riferimento ad un *concetto astratto* che può coincidere con qualunque notizia o racconto.
- L'informazione è qualcosa che viene comunicato in una qualsiasi forma scritta o orale

Informazione e regole di interpretazione

- Nella Teoria dell'Informazione l'informazione viene associata al concetto di *messaggio*,
 - anche se esso ha il solo compito di rappresentarla e trasportarla.
- affinché un messaggio possa essere interpretato, mittente e destinatario debbano aver concordato un *insieme di regole con le quali scrivere*, e in seguito leggere, il messaggio stesso.
- perché esista informazione, il messaggio deve contribuire *ad eliminare incertezza*:
 - se una sorgente di messaggi inviasse ripetutamente un solo simbolo, la quantità di informazione ricevuta a destinazione sarebbe nulla.
- i messaggi possono aumentare le conoscenze se l'informazione ricevuta si aggiunge a quelle preesistenti
 - ... dopo aver ricevuto un messaggio, *si sa di più rispetto* alle situazioni precedenti

Rappresentazione dell'Informazione

- Perché persone o macchine possano utilizzare un'informazione hanno bisogno che essa sia appropriatamente “rappresentata”.
 - se non fosse esistita la scrittura non avremmo un resoconto oggettivo degli avvenimenti dell'uomo dalla sua nascita fino ad oggi.
- scrivere, leggere ed elaborare informazioni implica che chi lo fa abbia preliminarmente concordato un *codice*, ossia una serie di regole e convenzioni da seguire.

Rappresentazione Analogica e Discreta

- Rappresentazione Analogica.
 - .. le proprietà del fenomeno rappresentato sono *omomorfe* alla forma della rappresentazione.
 - ... la rappresentazione *varia in analogia con la grandezza reale*:
 - una grandezza è rappresentata in modo continuo e la gran parte delle grandezze fisiche della realtà sono di tipo continuo.
- Rappresentazione Discreta
 - ... utilizza un insieme finito di rappresentazioni distinte che vengono messe in relazione con alcuni elementi dell'universo rappresentato
 - è un'approssimazione di quella analogica

Codifica

- Un'informazione per essere correttamente elaborata deve essere codificata in una rappresentazione comprensibile all'elaboratore stesso.
 - La *codifica* è l'insieme di convenzioni e di regole da adottare per trasformare un'informazione in una sua rappresentazione e viceversa.
 - la stessa informazione può essere codificata in modi diversi (rappresentazioni diverse) a seconda del contesto: le rappresentazioni araba o romana dei numeri ne sono un esempio (i simboli "1" e "I" costituiscono due codifiche diverse della stessa informazione numerica).

Codici

- Un *codice* è un sistema di simboli che permette la rappresentazione dell'informazione ed è definito dai seguenti elementi:
 - i simboli che sono gli elementi atomici della rappresentazione;
 - l'*alfabeto* che rappresenta l'insieme dei simboli possibili: con cardinalità (n) del codice si indica il numero di elementi dell'alfabeto;
 - le parole codice o stringhe che rappresentano *sequenze possibili (ammissibili) di simboli*: per lunghezza (l) delle stringhe si intende poi il numero di simboli dell'alfabeto da cui ciascuna parola codice risulta composta;
 - il linguaggio che definisce le regole per costruire parole codici che abbiano significato per l'utilizzatore del codice.

Parole Codice

- Siano allora $V = \{v_1, v_2, \dots, v_m\}$ l'insieme degli m valori diversi di una data informazione e $A = \{s_1, s_2, \dots, s_n\}$ un alfabeto composto da n simboli distinti. Si considerino diverse lunghezze delle parole codice:
 - con $l = 1$ si hanno tante parole codice diverse (n^1) quanti sono i simboli dell'alfabeto;
 - con $l = 2$ si hanno tante parole codice diverse quante sono le combinazioni con ripetizione degli n simboli nelle due posizioni, ossia n^2 .
 - con $l = 3$ si hanno n^3 parole codice diverse.
- In generale il numero di parole codice differenti è uguale a n^l .

Esempio

- Fissato l'alfabeto $A = \{-,.\}$ del codice Morse con $n=2$, si hanno

n^l	$l=1$	$l=2$	$l=3$	$l=4$
	-	--	---	----
2	.	-.	--.	---.
		.-	-.-	----
4		..	-..	---.
			.-.	---.
			..-	---.
8			...	---.
				-.--
				-.-
				.-.
				.-.
				..--
				..-
16				...-
			

Corrispondenza Biunivoca

- Se la codifica deve mettere in corrispondenza biunivoca i valori dell'informazione con le parole codice
 - ... ad ogni v_i deve corrispondere una ed una sola sequenza $s_{1i} s_{2j} \dots s_{ni} \dots$
- allora la lunghezza l deve essere scelta in modo che: **$n' \geq m$**

Ridondanza

- Nel caso di $n' > m$, non tutte le configurazioni possibili (parole codice) vengono utilizzate per la rappresentazione

Informazione	Suoi Valori	Sue rappresentazioni
Giorni settimana	lunedì, martedì, mercoledì, giovedì, venerdì, sabato, domenica	--- , --. , -. , -.. , .-- , .-. , ..-
Colori semaforo	rosso , giallo , verde	-- , -. , .-
Risposta	si, no	- , .

Codifica a Lunghezza fissa e variabile

- Codifica a lunghezza fissa:
 - tutte le parole codice hanno sempre la stessa lunghezza fissata da particolari esigenze applicative.
- Codifica a lunghezza variabile
 - Non tutte le parole codice hanno la stessa lunghezza
 - La scrittura è un caso di codifica a lunghezza variabile come è possibile verificare in un qualsiasi vocabolario.
 - I calcolatori adottano codifiche a lunghezza fissata e definita

Rappresentazione Digitale Binaria

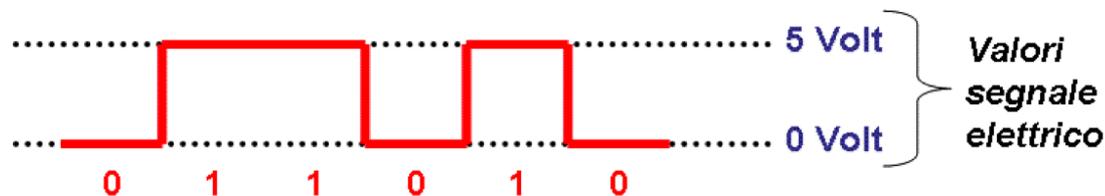
- Ai fini informatici assume particolare interesse la *rappresentazione binaria digitale*
 - basata su un alfabeto costituito da due soli simboli distinti, che assumono convenzionalmente la forma di “0” e “1”.
 - Tali due simboli rappresentano le unità minime di rappresentazione e memorizzazione digitale e vengono denominate *bit* da “binary digit”.
 - Il bit è l’unità di misura dell’elaborazione

Rappresentazione Digitale Binaria

- NOTA: Solitamente si indica con digitale la rappresentazione basata sui bit, anche se essa teoricamente sottintende una rappresentazione con qualsiasi tipo di cifre.
 - ... la diffusione dell'informatica ha comportato un'estensione del significato del termine e *digitale* assume il significato di informazione codificata in contrapposizione con *analogico* che invece descrive la realtà nelle sue infinite forme e varietà.

Perché Binaria

- La rappresentazione digitale, semplifica la memorizzazione delle informazioni e rende i sistemi digitali meno soggetti ai disturbi elettrici rispetto ai sistemi analogici.
 - Non a caso la rappresentazione delle informazioni all'interno dell'elaboratore si basa sull'alfabeto binario $\{0,1\}$ in quanto i supporti di memorizzazione delle informazioni, i registri di memoria, vengono realizzati con componenti elementari semplici detti flip-flop, che operano in due soli stati possibili.



Analogico vs Digitale

- Alcune informazioni nascono già in formato digitale grazie a strumenti che operano direttamente con tale rappresentazione: tra essi il calcolatore elettronico con le sue tante applicazioni, ma anche le moderne macchine fotografiche digitali , i telefoni cellulari, i registratori di suoni e video.

Analogico vs Digitale

- Per elaborare con un calcolatore delle grandezze reali di tipo continuo, bisogna utilizzare la loro rappresentazione digitale con una approssimazione che dipende dal processo di trasformazione in grandezze a valori discreti e dalla precisione della rappresentazione digitale dei numeri

Il codice binario

- il codice binario utilizza un alfabeto $A = \{0,1\}$ con $n=2$.
 - Le informazioni numeriche vengono quindi rappresentate mediante stringhe di bit di lunghezza l che producono 2^l configurazioni (parole codice) diverse.
 - Viceversa se si devono rappresentare K informazioni diverse occorrono $\log_2 K$ bit per associare ad esse codici diversi.

Byte e Words

- Per ragioni legate alla costruzione dei moderni calcolatori, è d'uso fare riferimento a stringhe con l uguale ad 8 che vengono dette *byte*.
- Sequenze di bit più lunghe di un byte sono invece denominate *word*, la loro lunghezza dipende dalle caratteristiche del sistema, ma è sempre un multiplo del byte: 16, 32, 64 o 128 bit

Sigla	Nome	Numero byte	Numero bit
B	Byte	1	8
KB	KiloByte	$2^{10}=1024$	8.192
MB	MegaByte	$2^{20}=1.048.576$	8.388.608
GB	GigaByte	$2^{30}=1.073.741.824$	8.589.934.592
TB	TeraByte	$2^{40}=1.099.511.627.776$	8.796.093.022.208

Come funziona nei moderni calcolatori

- Con otto bit si rappresentano solo 2^8 (256) valori diversi.
- ... nel caso in cui un solo byte non fosse sufficiente per rappresentare i K valori dell'informazione, allora si individua il numero b di byte tale che:
 - $2^{(b*8)} \geq K$
- In altri termini, all'interno dei moderni calcolatori, la codifica è a lunghezza fissa e adotta parole codice con una lunghezza che ha valori multipli di 8

Numero di byte b	Numero di bit ($b*8$)	$2^{(b*8)}$	Configurazioni
1	8	2^8	256
2	16	2^{16}	65.536
3	24	2^{24}	16.777.216
4	32	2^{32}	4.294.967.296

Precisione Finita

- L'adozione di stringhe a lunghezza finita e definita implica che i numeri gestiti siano a precisione finita,
 - ossia siano quelli rappresentati con un numero finito di cifre, o più semplicemente definiti all'interno di un prefissato intervallo di estremi $[min,max]$ determinati

Overflow e Underflow

- Nei sistemi di calcolo con numeri a precisione finita, le operazioni possono causare errori quando il risultato prodotto non appartiene all'insieme dei valori rappresentabili.
 - Si dice condizione di *underflow* quella che si verifica quando il risultato dell'operazione è minore del più piccolo valore rappresentabile (min).
 - Si chiama *overflow* la condizione opposta, ossia quella che si verifica quando il risultato dell'operazione è maggiore del più grande valore rappresentabile (max).
 - Infine il risultato dell'operazione non appartiene all'insieme quando non è compreso nell'insieme dei valori rappresentabili, pur non essendo né troppo grande né troppo piccolo.

esempio

- calcolatrice decimale dotata di sole tre cifre, con intervallo di definizione formato da numeri **interi** compresi nell'intervallo $[-999,+999]$

Operazione	Condizione
$200 + 100$	risultato rappresentabile
$730 + 510$	Overflow
$-500 - 720$	Underflow
$2 : 3$	risultato non rappresentabile

Algebra con precisione finita

- Anche l'algebra dei numeri a precisione finita è diversa da quella convenzionale poiché alcune delle proprietà:
 - proprietà associativa: $a + (b - c) = (a + b) - c$
 - proprietà distributiva: $a \times (b - c) = a \times b - a \times c$
- non sempre vengono rispettate in base all'ordine con cui le operazioni vengono eseguite

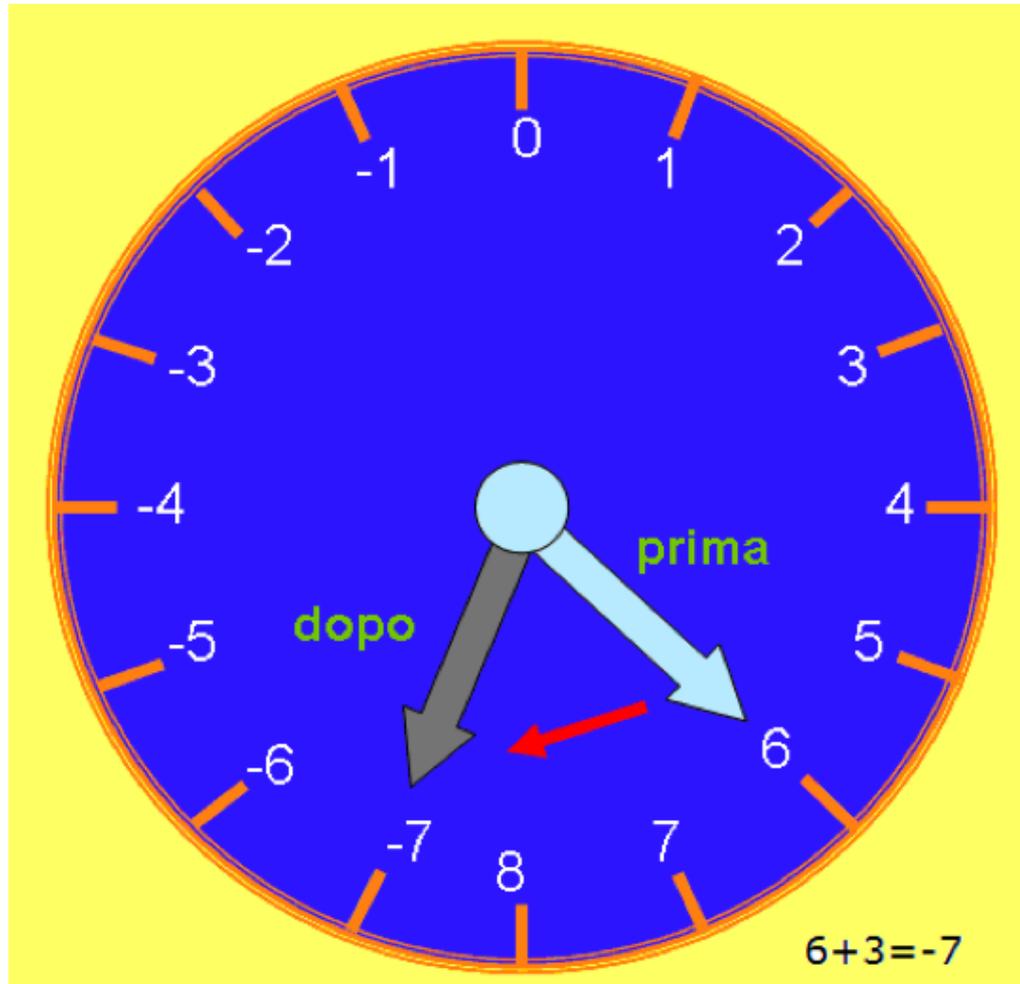
a	b	c	$a + (b - c)$	condizione	$(a + b) - c$	Condizione
100	900	600	$100 + (900 - 600)$	ok	$(100 + 900) - 600$	Overflow
a	b	c	$a \times (b - c)$	condizione	$a \times b - a \times c$	Condizione
200	90	88	$200 \times (90 - 88)$	ok	$200 \times 90 - 200 \times 88$	Overflow

Sistemi Periodici

- Per la periodicità i valori esterni all'intervallo di definizione vengono ricondotti ad esso prendendo il resto della divisione dei valori per il periodo

<i>intervallo</i>	<i>periodo</i>	<i>Valore</i>	<i>divisione</i>	<i>resto</i>
[0,360]	360	1200	1200 : 360	120
[0,60]	60	61	61 : 60	1
[0,60]	60	55	60 : 55	55

Calcolatrice per Sistemi di Numerazione finiti



Sistema Binario

- Il sistema binario ha una importanza capitale in informatica in quanto consente di rappresentare numeri mediante la combinazione di due soli simboli, ovvero di codificare i numeri direttamente in bit, secondo la notazione interna dei circuiti numerici.
- Inoltre all'interno dei calcolatori viene adottata un'algebra dei numeri a precisione finita con un intervallo di definizione che dipende dal numero di byte associato alla rappresentazione.

1	0	1	0	0	1	0	1	165
0	0	0	0	0	0	0	1	1
1	1	1	1	1	1	1	1	255
0	0	0	0	0	0	0	0	0
<i>Peso 7</i>	<i>Peso 6</i>	<i>Peso 5</i>	<i>Peso 4</i>	<i>Peso 3</i>	<i>Peso 2</i>	<i>Peso 1</i>	<i>Peso 0</i>	

LSB e MSB

- In un byte, il bit più a destra è quello meno significativo
 - (posizione o peso 0, detto anche *LSB* da *Least Significant Bit*)
- ... mentre quello più a sinistra è quello più significativo
 - (posizione o peso 7, detto anche *MSB* da *Most Significant Bit*)
- Poiché un byte può rappresentare 2^8 valori diversi, si possono, ad esempio con 8 bit gestire i seguenti intervalli di numeri interi:
 - [0, 255] (in binario [00000000,11111111])
 - [-127, 128] (in binario [11111111,01111111])
- ... entrambi costituiti da 256 numeri.

Sistemi di Numerazione

- Un sistema di numerazione può essere visto come un insieme di simboli (cifre) e regole che assegnano ad ogni sequenza di cifre uno ed un solo valore numerico.
- I sistemi di numerazioni vengono di solito classificati in sistemi *posizionali* e *non posizionali*.
 - Nei sistemi posizionali ogni cifra della sequenza ha un'importanza variabile a seconda della relativa posizione
 - nel sistema decimale la prima cifra a destra indica l'unità, la seconda le centinaia, etc...
 - nei sistemi non posizionali ogni cifra esprime una quantità non dipendente dalla posizione
 - ... nel sistema romano il simbolo "L" esprime la quantità 50 indipendentemente dalla posizione.

Numerazione posizionale pesata

$$N = c_i \times b^i + c_{i-1} \times b^{i-1} + c_{i-2} \times b^{i-2} + \dots + c_2 \times b^2 + c_1 \times b^1 + c_0 \times b^0 + c_{-1} \times b^{-1} + c_{-2} \times b^{-2} + \dots$$

Nel caso dei numeri interi scompaiono le potenze negative della base e la formula diventa:

$$N = c_i \times b^i + c_{i-1} \times b^{i-1} + c_{i-2} \times b^{i-2} + \dots + c_2 \times b^2 + c_1 \times b^1 + c_0 \times b^0$$

- Un sistema di numerazione posizionale è quindi definito dalla base (o radice) utilizzata per la rappresentazione.
- In un sistema posizionale in base **b** servono **b** simboli per rappresentare i diversi valori delle cifre compresi tra 0 e (**b**-1).

Base	Denominazione	Valori delle cifre
10	Decimale	0 1 2 3 4 5 6 7 8 9
2	Binaria	0 1
8	Ottale	0 1 2 3 4 5 6 7
16	Esadecimale	0 1 2 3 4 5 6 7 8 9 A B C D E F

Proprietà delle Rappresentazioni

- Nel passaggio da una base all'altra alcune proprietà dei numeri si perdono.
 - ... ad esempio un risultato di una divisione può essere periodico nella base dieci ma non è detto che lo sia in un'altra base, così come la proprietà di un numero di essere divisibile per cinque ha senso solo se la base è maggiore di cinque.
- conversione nella base 10 da qualsiasi base **b**, calcolando la sommatoria dei prodotti delle cifre per i pesi.
 - Ad esempio:
 - $(101111)_2 = 1 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 = 32 + 8 + 4 + 2 + 1 =$
 - $(142)_5 = 1 \times 5^2 + 4 \times 5^1 + 2 \times 5^0 = 25 + 20 + 2 =$
 - $(47)_{10} = 4 \times 10^1 + 7 \times 10^0$
- NOTA: L'impiego nella base 2 di un minor numero simboli rispetto al sistema decimale (2 contro 10) implica che lo stesso numero abbia una parola-codice più lunga in notazione binaria che non in quella decimale

Ottale ed Esadecimale

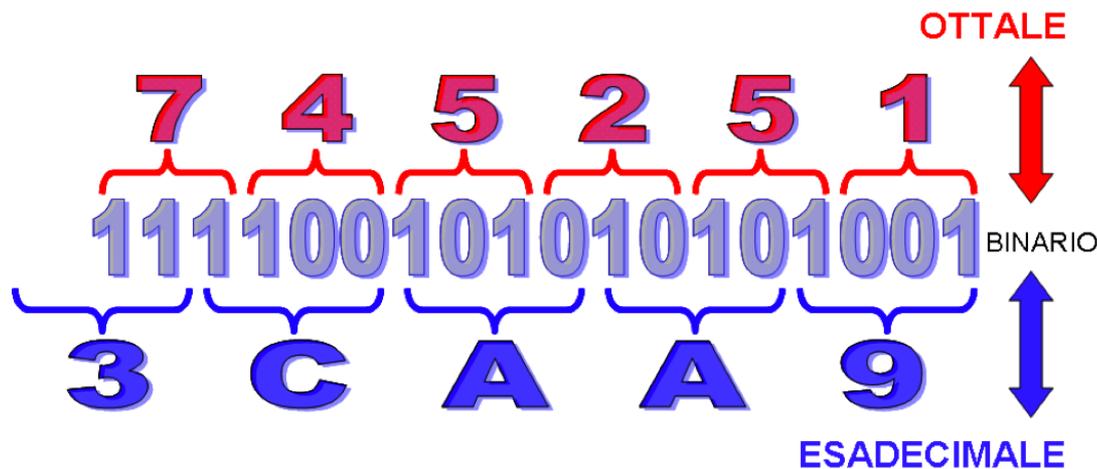
- Per rappresentare le dieci cifre ci vogliono $\log_2 10$ bit ($\cong 3,3$ bit), solitamente la stringa di cifre in bit è approssimativamente tre volte più lunga di quella decimale
..
 - $(1001101)_2 = 1 \times 2^6 + 0 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = 64 + 0 + 0 + 8 + 4 + 0 + 1 = (77)_{10}$
- Così, per evitare di dover trattare con stringhe di bit troppo lunghe, sono molto usati il sistema *ottale* ed *esadecimale*.

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

Esadecimale	Binario
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
A	1010
B	1011
C	1100
D	1101
E	1110
F	1111

Relazione tra numeri in basi potenze di due

- La trasformazione di un valore da binario in ottale è molto semplice dato che una cifra del sistema ottale è rappresentabile esattamente con tre bit del sistema binario il cui valore è uguale proprio alla cifra rappresentata.
 - La conversione avviene raggruppando le cifre binarie in gruppi di tre a partire dalla posizione di peso minore.
 - La conversione opposta è ugualmente semplice: ogni cifra ottale viene esplosa esattamente nelle tre cifre binarie che la rappresentano.
 - La rappresentazione esadecimale è ancora più compatta: il processo di conversione è equivalente a quello binario-ottale ma le cifre binarie devono essere raggruppate in gruppi di quattro.



Conversione decimale - binario

- Dato un valore d decimale, nel caso di $b=2$:

$$d_{pi} = c_i \times 2^i + c_{i-1} \times 2^{i-1} + \dots + c_2 \times 2^2 + c_1 \times 2^1 + c_0 \times 2^0$$

$$d_{pf} = c_{-1} \times b^{-1} + c_{-2} \times b^{-2} + \dots$$

$$\text{con: } d = d_{pi} + d_{pf}$$

- se si divide la parte intera per 2:

$$d_{pi} / 2 = c_i \times 2^{i-1} + c_{i-1} \times 2^{i-2} + \dots + c_2 \times 2^1 + c_1 \times 2^0 + c_0 \times 2^{-1}, \text{ con } c_0 \text{ resto della divisione.}$$

- Se ora si divide la parte intera ottenuta precedentemente (d_{pi1}) ancora per la base 2:

$$d_{pi1} / 2 = c_i \times 2^{i-2} + c_{i-1} \times 2^{i-3} + \dots + c_2 \times 2^0 + c_1 \times 2^{-1}, \text{ con } c_1 \text{ resto della divisione}$$

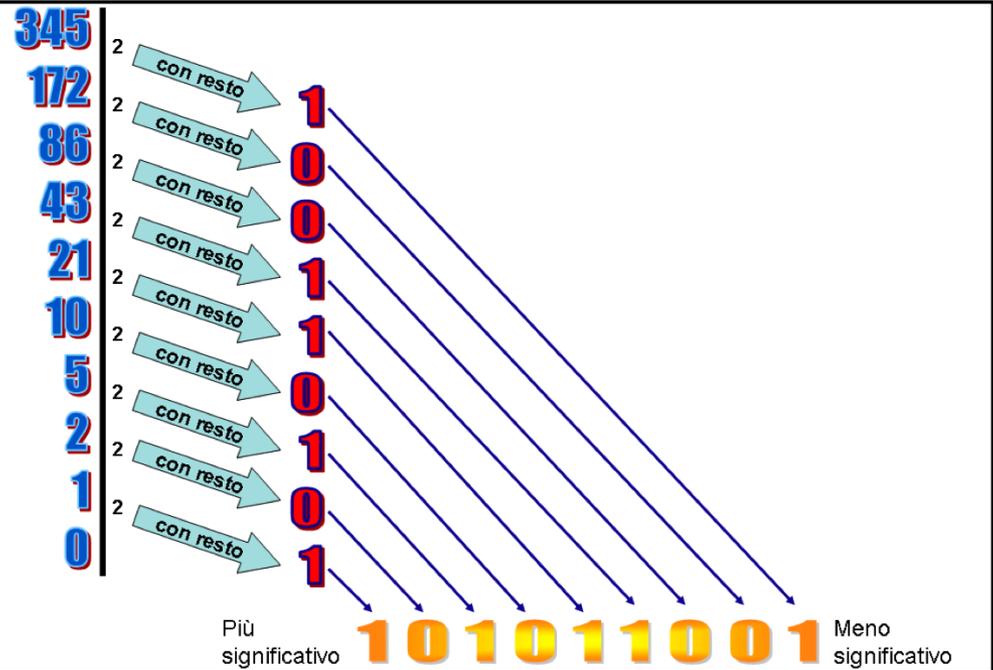
- il procedimento deve essere ripetuto fino a quando si ottiene un quoziente uguale a 0.

Procedimento

ALGORITMO

- 1) dividere la parte intera del numero d per la base b
- 2) scrivere il resto della divisione
- 3) se il quoziente è maggiore di zero, usare tale risultato al posto del numero d di partenza e continuare dal punto 1)
- 4) se il quoziente è zero, scrivere tutte le cifre ottenute come resto in sequenza inversa

Si noti che l'algoritmo consente di convertire un numero intero in base dieci in una qualunque base b . Nel caso di $b = 2$ si ottiene la conversione in binario del numero assegnato.



Conversione per numeri frazionari

- Per la conversione della parte frazionaria si procede al contrario moltiplicando per 2:

$$d_{pf} \times 2 = c_{-1} \times b^0 + c_{-2} \times b^{-1} + \dots$$

$$d_{pf1} = c_{-2} \times b^{-1} + \dots$$

per spostare c_{-1} a sinistra della virgola che diventa parte intera.

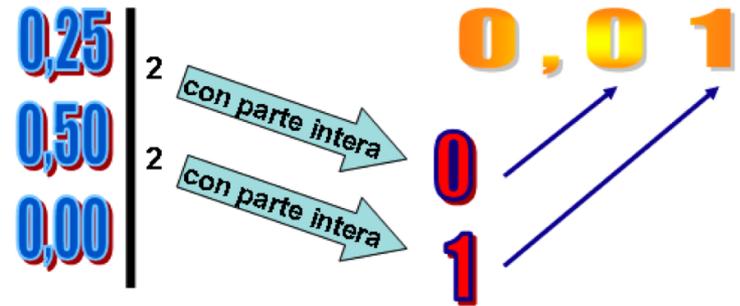
- si continua a moltiplicare per 2 solo la parte frazionaria fino a quando non si verifica una delle seguenti condizioni:
 - la parte frazionaria $d_{pfi-esima}$ non si annulla;
 - la parte frazionaria $d_{pfi-esima}$ si ripete con periodicità;
- ci si accontenta di una rappresentazione approssimata con un numero di bit inferiore a quello che andrebbero calcolati per raggiungere una delle condizioni precedenti.
 - solo la prima condizione garantisce una conversione senza approssimazione

... procedimento

ALGORITMO

- 1) moltiplicare la parte frazionaria del numero d per la base b
- 2) scrivere la parte intera del prodotto
- 3) se la nuova parte frazionaria del prodotto è diversa da zero o non si ripete periodicamente, oppure si non sono state determinate le cifre binarie prefissate, usare tale risultato al posto del numero d di partenza e continuare dal punto 1)
- 4) se la nuova parte frazionaria verifica una delle tre condizioni di terminazione, scrivere tutte le cifre ottenute come parte intera nell'ordine in cui sono state calcolate

Si noti che l'algoritmo consente di convertire un numero frazionario in base dieci in una qualunque base b . Nel caso di $b = 2$ si ottiene la conversione in binario del numero assegnato.



Operazioni sui binari

- ... si definiscono la tavola dell'addizione e la tabellina del prodotto per le cifre binarie.

a	b	a+b	a*b
0	0	0	0
0	1	1	0
1	0	1	0
1	1	10	1

Si noti che $1+1$ fa 0 con riporto 1

Esempi

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

Somma

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

Sottrazione

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

Prodotto

Rappresentazione numeri negativi in segno e modulo

- Segno e Modulo
 - Poiché il segno assume due soli valori (“+” oppure “-“), allora lo si può codificare con un singolo bit utilizzando il bit più significativo per indicarlo
 - ad esempio, “0” per indicare un valore positivo ed “1” per indicarne uno negativo.
 - Con l bit, $l - 1$ di essi vengono attribuiti alla rappresentazione del valore assoluto del numero, e il bit più a sinistra (MSB) alla rappresentazione del segno.

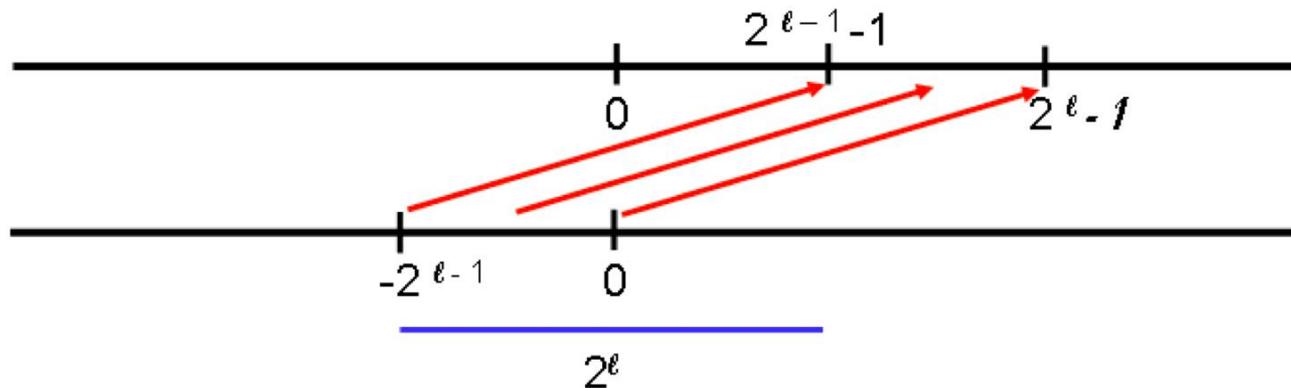


Rappresentazione in segno e modulo

- ... consente di codificare tutti i numeri relativi appartenenti all'intervallo:
 $[-2^{l-1} + 1, 2^{l-1} - 1]$
- con 2^{l-1} valori positivi e altrettanti negativi: per un totale di 2^l valori diversi.
 - ... ad esempio, Per $l=8$ sono rappresentabili tutti i numeri relativi appartenenti all'intervallo $[-127, 127]$.
- Problemi del segno e modulo:
 - ... sono presenti due configurazioni dello zero, lo "0" positivo (00000000) e lo "0" negativo (10000000), → le operazioni di somma e sottrazione devono essere corrette nell'attraversamento dello zero.
 - Richiede un algoritmo complesso per effettuare somma e sottrazione in presenza delle diverse combinazioni dei segni degli operandi.

Complemento a due

- ... le configurazioni che hanno il bit più significativo uguale a zero, cioè quelle comprese nell'intervallo $[0, 2^{\ell-1}-1]$, rappresentano se stesse (numeri positivi)
- le configurazioni col bit più significativo uguale a uno, cioè quelle rientranti nell'intervallo $[2^{\ell-1}, 2^{\ell}-1]$, rappresentano i numeri negativi che si ottengono traslando a sinistra l'intervallo di 2^{ℓ} , cioè l'intervallo $[-2^{\ell-1}, -1]$.



Come si calcola

- Il complemento a due x'' di un valore negativo x si calcola sottraendo il valore assoluto di x a 2^l :

$$x'' = 2^l - |x|$$

- Se si definisce il complemento alla base b di una cifra c come:

$$c' = b - 1 - c$$

allora il complemento a 2 si ottiene complementando alla base tutte le cifre del valore assoluto del numero x e sommando poi 1 al valore ottenuto.

Esempio: $N = 00110101$

Complemento alla base delle cifre =

11001010+

00000001=

11001011

Osservazioni

- Nella rappresentazione per complemento a 2, i valori rappresentati sono compresi nell'intervallo: $[-2^l - 1, 2^l - 1 - 1]$ sono sempre 2^l :
 - $[0, 2^l - 1]$ per i valori positivi
 - $[-2^l - 1, -1]$ per i valori negativi.
- l'intervallo non è simmetrico:
 - $2^l - 1$ valore assoluto del minimo
 - $2^l - 1$ valore del massimo
- esiste una sola rappresentazione dello zero.

Esempi

- Con 8 bit, ad esempio, si rappresentano i numeri naturali nell'intervallo $[0, 2^8-1]$, cioè $[0, 255]$, oppure i numeri relativi nell'intervallo $[-2^7, 2^7-1]$, cioè $[-128, 127]$. Con 16 bit (2 byte) si rappresentano i numeri naturali nell'intervallo $[0, 2^{16}-1]$, cioè $[0, 65535]$, oppure i numeri relativi nell'intervallo $[-2^{15}, 2^{15}-1]$, cioè $[-32768, 32767]$.

Numeri Negativi

- ... se si ha una sequenza di n bit che rappresenta un numero intero con segno, con i numeri negativi rappresentati in complemento a 2, allora, per ottenere il numero rappresentato, si procede nel seguente modo.
- Si esamina il bit di segno.
 - Se esso è zero, il numero rappresentato è non negativo e lo si calcola con la normale conversione binario-decimale.
 - Se invece il bit di segno è uno, allora si tratta di un numero negativo, per cui, per ottenerne il valore assoluto, si complementano tutti i bit e si somma 1 al risultato.

2^8-	1	0	0	0	0	0	0	0	
9 =		0	0	0	0	1	0	0	1
-9		1	1	1	1	0	1	1	1

Si complementano
le cifre

Si somma 1

9	0	0	0	0	1	0	0	1
	1	1	1	1	0	1	1	0
								1
-9	1	1	1	1	0	1	1	1

Interpretazione del complemento a due

- Dati / bit, con la prima posizione che parte da zero, si ha che il peso della cifra più significativa c_{l-1} è -2^{l-1} :

$$c_{l-1} \times (-2^{l-1}) + c_{l-2} \times (2^{l-2}) + \dots + c_1 \times 2^1 + c_0 \times 2^0$$

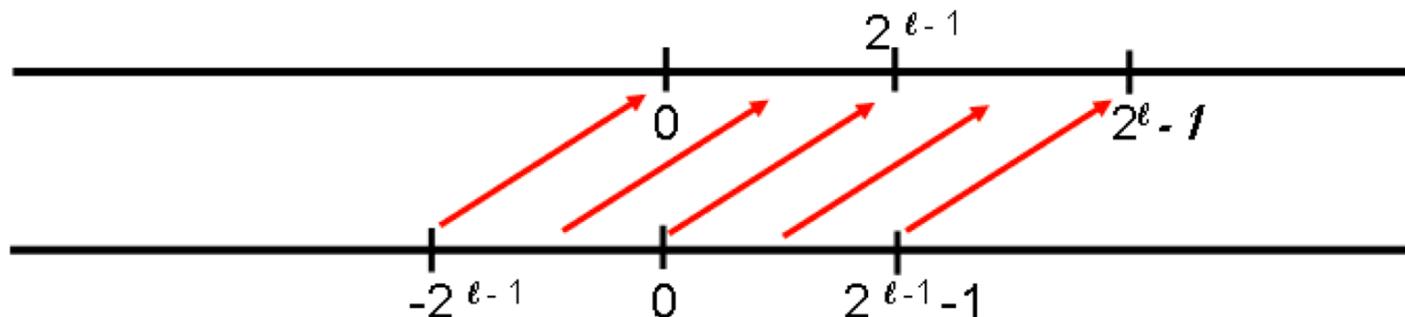
1	1	1	1	0	1	1	1	*
-2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	
-128	64	32	16	8	4	2	1	=
-128	64+32+16+4+2+1=119						-9	

Complemento diminuito

- Il *complemento a uno* (x') del numero x si differenzia dal *complemento a 2* (x'') dello stesso numero per una unità:
$$x' = x'' - 1$$
- Il *complemento a 1* di un numero si ottiene semplicemente complementando tutte le cifre del numero.
- è stato usato in alcuni calcolatori, ma è stato abbandonato perché
 - ... nonostante è semplice la determinazione dei numeri negativi
 - ... Ha una doppia rappresentazione dello zero che complica le operazioni di somma e sottrazione.

Rappresentazione per eccessi

- I numeri negativi si determinano come somma di se stessi con 2^{l-1} dove l è il numero di bit utilizzati.
- il sistema è identico al complemento a due con il bit di segno invertito.
 - In pratica i numeri compresi in $[-2^{l-1}, 2^{l-1}-1]$ sono mappati tra $[0, 2^l-1]$
 - In tale rappresentazione, il numero binario che rappresenta 2^{l-1} sarà associato allo zero, mentre i valori minori di 2^{l-1} ai numeri negativi e quelli maggiori a quelli positivi.
 - Nel caso di $n = 8$ i numeri appartenenti a $[-128, 127]$ sono mappati nell'intervallo $[0, 255]$ (con i numeri da 0 a 127 considerati negativi, il valore 128 corrisponde allo 0 e quelli maggiori di 128 sono positivi).



Overview

Valore decimale di N	N in binario (8 bit)	-N in segno e modulo	-N complemento a 1	-N complemento a 2	-N eccesso $2^7 = 128$
0	00000000	10000000	11111111	Non esiste	10000000
1	00000001	10000001	11111110	11111111	01111111
10	00001010	10001010	11110101	11110110	01110110
50	00110010	10110010	11001101	11001110	01001110
100	01100100	11100100	10011011	10011100	00011100
127	01111111	11111111	10000000	10000001	00000001
128	10000000	Non esiste	Non esiste	10000000	00000000

Quali si usano?

- Le rappresentazioni in complemento a due ed eccesso sono le più efficienti per svolgere operazioni in aritmetica binaria poiché permettono di trattare la sottrazione tra numeri come una somma tra numeri di segno opposto:

$$(X - Y) = (X + (-Y))$$

- È così possibile costruire dei circuiti che fanno solo addizioni.
 - Si noti che tale proprietà ha validità solo nel caso di rappresentazioni finite dei numeri

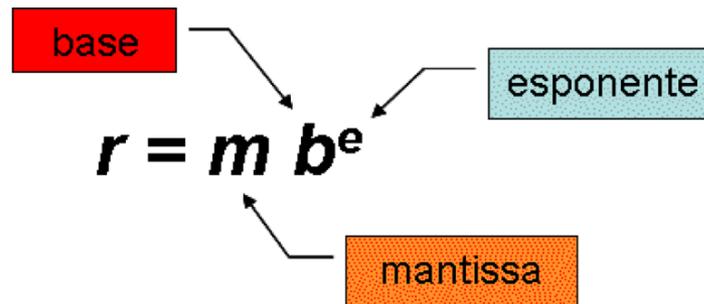
13 -	0	0	0	0	1	1	0	1	+
10 =	1	1	1	1	0	1	1	0	=
3	1	0	0	0	0	0	1	1	
	0	0	0	0	0	0	1	1	

Si noti che il 9 bit di overflow si perde in quanto la rappresentazione si compone di soli 8 bit

13 -	0	0	0	0	1	1	0	1	+
20 =	1	1	1	0	1	1	0	0	=
-7	1	1	1	1	1	0	0	1	

Numeri Reali

- I numeri reali vengono rappresentati in binario attraverso la seguente notazione scientifica: $r = m b^e$, con m numero frazionario detto *mantissa*, la *base* b numero naturale prefissato ed e numero intero chiamato *esponente* o *caratteristica*.
 - L'esponente determina l'ampiezza dell'intervallo di valori preso in considerazione, mentre il numero di cifre della mantissa determina la precisione del numero (ossia con quante cifre significative sarà rappresentato)

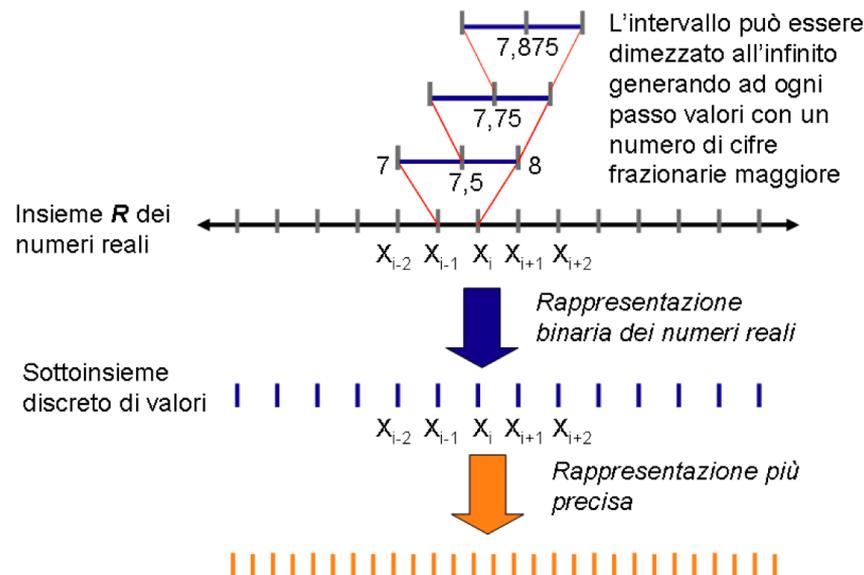


Notazione Scientifica

- la sua indipendenza dalla posizione della virgola;
- la possibilità di trascurare tutti gli zeri che precedono la prima cifra significativa con la normalizzazione della mantissa;
- la possibilità di rappresentare con poche cifre numeri molto grandi oppure estremamente piccoli;
- la dipendenza del valore rappresentato dalla mantissa e dall'esponente se si adottano specifiche convenzioni per la base e la mantissa;

Rappresentazione finita e discreta dei numeri reali

- In un intervallo reale, comunque piccolo, esistono infiniti valori (i numeri reali formano un continuo).
- I valori rappresentabili in binario appartengono invece ad un sottoinsieme che contiene un numero finito di valori reali ognuno dei quali rappresenta un intervallo del continuo.
- In altri termini, diviso l'insieme dei numeri reali in intervalli di fissata dimensione, si ha che ogni x appartenente all'intervallo $[X_i, X_{i+1}[$ viene sostituito con X_i .



Effetti delle Approssimazioni

- ... CALCOLO NUMERICO, che si pone come obiettivo la ricerca di algoritmi appropriati per la soluzione di problemi matematici che fanno largo uso dei numeri reali.
 - Difatti un qualsiasi calcolo numerico sarebbe privo di senso, qualora non si avesse un'idea del tipo e dell'entità degli errori che si possono commettere.
- I numeri reali rappresentabili in binario godono della seguente proprietà:

$$\frac{|x - X|}{|X_{i+1} - X_i|} < \varepsilon$$

- dove ε rappresenta l'errore che si commette sostituendo x con X , e :
 - $X = X_i$ se si approssima per difetto;
 - $X = X_{i+1}$ se si approssima per eccesso.
- Il valore ε dipende dalla rappresentazione finita (numero finito di cifre) utilizzata per i numeri reali.

Esempio errori di arrotondamento

NUMERO	ARROTONDAMENTO	ERRORE
0,00347	0,0035	$3 \cdot 10^{-5} = 0.3 \cdot 10^{-4}$
0,000348	0,0003	$48 \cdot 10^{-6} = 0.48 \cdot 10^{-4}$
0,00987	0,0099	$3 \cdot 10^{-5} = 0.3 \cdot 10^{-4}$
0,000987	0,0010	$13 \cdot 10^{-6} = 0.13 \cdot 10^{-4}$

- ...Per un aritmetica a quattro cifre decimali
- con un errore massimo sull'ultima cifra di 0.5 ($0.5 \cdot 10^{-4}$).
- In generale se **-m** è il peso della cifra meno significativa, l'errore massimo che si commette è:

$$\varepsilon = \frac{1}{2} \times 10^{-m}$$

Overflow e Underflow

- I numeri reali rappresentabili sono definiti in un insieme limitato con estremi predefiniti *[-minreal, maxreal]*.
 - Overflow: condizione che si verifica quando i valori o sono più piccoli di minreal o più grandi di maxreal;
 - Underflow: condizione che si verifica quando un valore, per effetto delle approssimazioni, viene confuso con lo zero.

Rappresentazione Normalizzata

- la rappresentazione in virgola mobile, fissata la base, consente di esprimere lo stesso valore con infinite coppie (mantissa, esponente)
 - ad esempio: 48×10^3 è uguale a 4800×10^0 , ma anche a $4,8 \times 10^2$
- È allora possibile scegliere, tra le infinite coppie *quella che preserva il maggior numero di cifre significative con la normalizzazione della mantissa*.
 - Per esempio, per i numeri minori di uno quando la cifra più a sinistra è uno zero, si traslano (shift) verso sinistra le cifre diverse da zero (significative) decrementando l'esponente di tante cifre quante sono le posizioni scalate: in questo modo si ottiene un'altra coppia distinta, ma avente il medesimo valore del precedente (ad esempio $0,0025 \times 10^0$ è equivalente $2,5000 \times 10^{-3}$).
- In generale la forma normalizzata della mantissa obbliga che *la sua prima cifra sia diversa da zero e che la sua parte intera sia in generale un numero minore dalla base*.

Esempio di Normalizzazione

- rappresentazione con $b = 10$,
- cinque cifre per la mantissa considerata minore di 10,
- due cifre per l'esponente,
- rappresentazione normalizzata con la prima cifra diversa da zero;
- si hanno le seguenti rappresentazioni normalizzate:

Numero	Valore
0,384	$3,8400 \times 10^{-1}$
1345	$1,3450 \times 10^3$
64350	$6,4350 \times 10^4$
333	$3,3300 \times 10^2$
0,0048	$4,8000 \times 10^{-3}$
0,0000001	$1,0000 \times 10^{-8}$

... condizione di overflow quando

$$x > 9,9999 \times 10^{99}$$

e di underflow quando

$$x < 1,0000 \times 10^{-99}$$

Osservazione

- ... gli intervalli $[X_i, X_{i+1}]$ non hanno tutti la stessa ampiezza a causa della finitezza del numero di cifre della mantissa:
 - man mano che ci si avvicina alla condizione di overflow gli intervalli si fanno sempre più ampi
 - ... mentre intorno alla condizione di underflow non solo si addensano ma diventano sempre più piccoli.
- Con l'esempio precedente è facile osservare il fenomeno confrontando gli intervalli $[1.0000 \times 10^{-99}, 1.0001 \times 10^{-99}]$ e $[9.9998 \times 10^{99}, 9.9999 \times 10^{99}]$.

Operazioni in Virgola Mobile

- ... le operazioni non solo si complicano ma possono generare errori di approssimazione.
 - Ad esempio la somma e la sottrazione richiedono l'allineamento degli esponenti: $100 \times 10^0 + 100 \times 10^{-2} = 100 \times 10^0 + 1 \times 10^0 = 101 \times 10^0$
 - mentre per il prodotto e la divisione servono operazioni separate sulle mantisse e sugli esponenti: $100 \times 10^0 * 100 \times 10^{-2} = (100 * 100) \times 10^{(0-2)} = 10000 \times 10^{-2}$
- L'allineamento degli esponenti produce come effetto indesiderato quello di far scomparire alcune cifre rappresentative del numero.
 - Ad esempio la somma dei numeri seguenti: $1,9099 \times 10^1 + 5,9009 \times 10^4$ nell'esempio precedente, diventa: $0,0001 \times 10^4 + 5,9009 \times 10^4$,
 - ... con il troncamento delle cifre 9099 del numero con esponente più piccolo.
- **NB1:** se si sottraggono numeri di valore quasi uguale, le cifre più significative si eliminano fra loro e la differenza risultante perde un certo numero di cifre significative o anche tutte (*cancellazione*).
- **NB2:** la divisione per valori molto piccoli può facilmente superare il valore di overflow.

Perché rappresentare in virgola mobile?

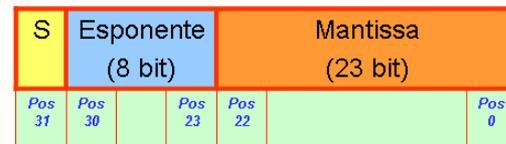
- .. se si conviene che le mantisse siano trasformate in valori minori di 10 con operazioni interne, un *numero reale può essere rappresentato nella memoria di un calcolatore con un numero intero indicativo della parte decimale della mantissa e con un altro numero intero per l'esponente.*
- Per esempio il numero $0,1230 \times 10^{-9}$ viene rappresentato con la coppia di numeri interi (1230,-9) e gestito con operazioni interne che ne costruiscono l'effettivo valore.
- Analogamente in base due

$$1. \boxed{\text{xxxxxxxx}} \times 2^{\boxed{\text{yyyy}}} \quad \leftarrow \text{esponente}$$

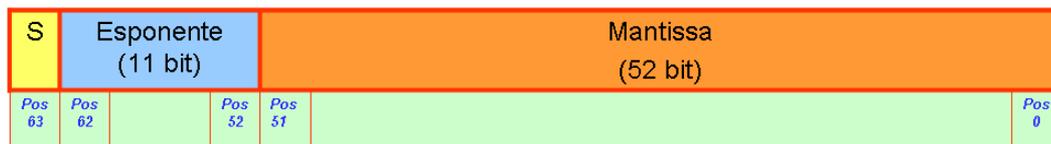
↑
rappresentazione in base 2 della
“parte significativa” della mantissa

Standard Virgola mobile

- Standard 754 IEEE: definisce principalmente tre formati numerici a virgola mobile:
 - singola precisione o precisione semplice (32 bit),
 - doppia precisione (64 bit),
 - precisione estesa (80 bit).



Singola precisione (32 bit)



Doppia precisione (64 bit)

Rappresentazione VM precisione singola e doppia

- ... un bit per il segno del numero complessivo, (zero per positivo ed uno per negativo);
- otto bit nel caso della singola precisione (11 per la doppia precisione) per l'esponente rappresentato per eccesso così da non doverne indicare il segno;
- 23 bit nel caso della singola precisione (52 per la doppia) per la mantissa.
- La mantissa è normalizzata per cui comincia sempre con un 1 seguito da una virgola binaria, e poi a seguire il resto delle cifre.
 - Lo standard prevede l'assenza sia del primo bit che del bit della virgola perché sono sempre presenti:

Argomento	Precisione singola 32 Bit	Precisione doppia 64 bit
Bit del segno	1	1
Bit per l'esponente	8	11
Bit per la mantissa	23	52
Cifre decimali mantissa	Circa 7 (23/3.3)	Circa 15 (52/3.3)
Esponente (rappresentazione)	base 2 ad eccesso 127	base 2 ad eccesso 1023
Esponente (valori)	[-126, 127]	[-1022, 1023]

Operatori Booleani

- Sulle stringhe di bit sono anche definiti operatori che lavorano bit a bit (*bitwise operator*). Essi sono detti *booleani* e sono:
 - **AND**: dati due bit restituisce il valore 1 se e solo se i bit erano entrambi posti a 1, in tutti gli altri casi il risultato è 0; l'AND è detto anche *prodotto logico*.
 - **OR**: dati due bit restituisce il valore 0 se e solo se i bit erano entrambi posti a 0, in tutti gli altri casi il risultato è 1; l'OR è anche detto *somma logica*.
 - **NOT**: dato un bit restituisce il valore 0 se esso era posto a 1, restituisce invece 1 se il bit era posto a 0; il NOT viene anche detto operatore di *negazione* o di *complementazione*.

Tavola di Verità

a	b	<i>NOT</i> a	a <i>AND</i> b	a <i>OR</i> b
0	0	1	0	0
0	1	1	0	1
1	0	0	0	1
1	1	0	1	1

Esempio

0	0	0	0	1	1	0	1	AND
1	1	1	0	1	1	0	0	=
0	0	0	0	1	1	0	0	

0	0	0	0	1	1	0	1	OR
1	1	1	0	1	1	0	0	=
1	1	1	0	1	1	0	1	

0	0	0	0	1	1	0	1	NOT
1	1	1	1	0	0	1	0	

Proposizioni Logiche

- Una *proposizione semplice* è qualsiasi enunciato che può assumere valore vero o falso
 - Esempio “vi sono altre cifre a sinistra delle ultime considerate”
 - Queste condizioni possono assumere un valore vero o falso, cioè un valore logico
 - In modo più sintetico la condizione usata come esempio che si può anche esprimere con l’espressione:
 - (numero di cifre a sinistra) > 0

Operatori di Confronto

- Permettono il confronto di valori qualsiasi.
- Gli operatori di relazione più noti sono quelli che permettono di confrontare quantità numeriche:
 - uguale (simbolo '=')
 - diverso (simbolo '≠')
 - maggiore (simbolo '>')
 - minore (simbolo '<')
 - maggiore o uguale (simbolo '≥')
 - minore o uguale (simbolo '≤')

Predicati Semplici con Operatori di relazione

- Gli operatori di relazione permettono di esprimere predicati semplici
 - $1 < 2$ (valore: vero)
 - $7 < 0$ (valore: falso)
 - $x \geq 0$ (valore: dipende cosa rappresenta 'x')
 - oggi = mercoledì (valore: dipende cosa rappresenta 'oggi')
- Non sono operatori logici, ma consentono di costruire espressioni che possono essere usate come argomenti di operatori logici

Proposizioni Composte: esempio

- Ad esempio l'algoritmo seguito dall'impiegato di un ufficio postale nell'accettare conti corrente da parte degli utenti in coda potrebbe contenere un passo del tipo:
 - accetta un altro conto corrente se l'utente ha già consegnato meno di 5 conti corrente o non ci sono altri utenti in coda
 - Oppure:
 - accetta un altro conto corrente se non accade che l'utente ha già consegnato almeno 5 conti corrente e ci sono altri utenti in coda

La logica delle Proposizioni

- La verità di un predicato che contiene operatori logici può essere però facilmente calcolata a partire dalla verità dei predicati componenti e dalle proprietà degli operatori che li uniscono
- Conoscendo tali proprietà è agevole:
 - costruire predicati complessi, confrontarli per verificarne l'equivalenza
 - trasformarli per individuare la formulazione più conveniente.

Operatori logici

- $\{ F , V \}$: valori (F=falso, V=vero)
- OR : operatore binario chiuso
detto anche *disgiunzione* logica
- AND : operatore binario chiuso
detto anche *congiunzione* logica
- NOT : operatore unario chiuso
detto anche *negazione* logica

Algebra di Boole

- La logica delle proposizioni è un'algebra di Boole
- Legge di dualità:
Da qualsiasi identità booleana se ne può trarre un'altra per dualità, sostituendo cioè ad ogni operatori e ai valori F e T il rispettivo duale

Proprietà

commutativa:

$$a \text{ OR } b = b \text{ OR } a$$

$$a \text{ AND } b = b \text{ AND } a$$

associativa:

$$(a \text{ OR } b) \text{ OR } c = a \text{ OR } (b \text{ OR } c)$$

$$(a \text{ AND } b) \text{ AND } c = a \text{ AND } (b \text{ AND } c)$$

idempotenza:

$$a \text{ OR } a = a$$

$$a \text{ AND } a = a$$

Proprietà

distributiva:

$$a \text{ AND } (b \text{ OR } c) = (a \text{ AND } b) \text{ OR } (a \text{ AND } c)$$

$$a \text{ OR } (b \text{ AND } c) = (a \text{ OR } b) \text{ AND } (a \text{ OR } c)$$

del minimo e del massimo:

$$a \text{ AND } F = F$$

$$a \text{ OR } V = V$$

del complemento:

$$a \text{ AND } (\text{NOT } a) = F$$

$$a \text{ OR } (\text{NOT } a) = V$$

Teorema di De Morgan

$$a \text{ OR } b = \text{NOT } ((\text{NOT } a) \text{ AND } (\text{NOT } b))$$

$$a \text{ AND } b = \text{NOT } ((\text{NOT } a) \text{ OR } (\text{NOT } b))$$

Dimostrazione usando le tabelle di verità

Risoluzione dell'esempio

- *Poniamo:*
 - *A*: l'utente ha già consegnato meno di 5 conti corrente
 - *B*: ci sono altri utenti in coda
 - *A'*: l'utente ha già consegnato almeno 5 conti corrente

- La prima condizione corrisponde alla formula:

$$A \text{ OR } (\text{NOT } B)$$

- La seconda condizione corrisponde alla formula:

$$\text{NOT } (A' \text{ AND } B)$$

osservando che $A' = \text{NOT } A$ diventa:

$$\text{NOT } (\text{NOT } A \text{ AND } B)$$

Risoluzione dell'esempio

- Costruiamo le tabelle di verità delle due espressioni
 - valore di verità dell'espressione composta in corrispondenza di ogni possibile combinazione di valori di verità delle condizioni atomiche componenti (A e B)
- Le due condizioni sono equivalenti se le due tabelle di verità sono uguali

Confronto tabelle di verità

prima condizione

A	B	NOT B	A OR (NOT B)
F	F	V	V
F	V	F	F
V	F	V	V
V	V	F	V

seconda condizione

A	B	NOT A	(NOT A) AND B	NOT ((NOT A) AND B)
F	F	V	F	V
F	V	V	V	F
V	F	F	F	V
V	V	F	F	V

Rappresentazione Testi

Il testo è uno degli oggetti digitali più diffuso nel mondo informatico. Molte sono le applicazioni che generano e manipolano i cosiddetti documenti elettronici. Un testo digitale è una stringa di simboli ad ognuno dei quali viene associato un codice binario secondo un prefissato standard.

L a c a s a
01001100 01100001 00100000 01100011 01100001 01110011 01100001

Rappresentazione dei Caratteri

- Per rappresentare i caratteri esistono vari codici
- gli standard accettati sono:
 - EBCDIC 8 bit
 - ASCII 7 bit (ASCII esteso 8 bit = 256 simboli)
- Lo standard e' importante affinchè sistemi diversi comunichino tra loro

ANSI ASCII

- Alla fine degli anni sessanta l'ente americano di standardizzazione ANSI (American National Standards Institute) decise di fissare un alfabeto che consentisse a tutti i calcolatori, anche di produttori diversi, di poter comunicare tra loro o con i dispositivi ad essi collegati.
- I simboli dell'alfabeto vennero elencati in una tabella per codificare, attraverso la posizione assunta da loro in essa, vari tipi di caratteri: alfabetici, numerici, di punteggiatura, simbolici, e anche alcuni codici da usare come controllo della comunicazione tra una macchina e l'altra (per esempio, per segnalare l'inizio o la fine di una trasmissione).
- Il trasferimento di un insieme di informazioni da un calcolatore all'altro su una rete poteva così essere effettuato attraverso un linguaggio comune, costituito da tale forma di codifica.
- La tabella fu chiamata *ASCII*, ossia *American Standard Code for Information Interchange*.

Codice ASCII

00 nul	01 soh	02 stx	03 etx	04 eot	05 enq	06 ack	07 bel
08 bs	09 ht	0A nl	0B vt	0C np	0D cr	0E so	0F si
10 dle	11 dc1	12 dc2	13 dc3	14 dc4	15 nak	16 syn	17 etb
18 can	19 em	1A sub	1B esc	1C fs	1D gs	1E rs	1F us
20 sp	21 !	22 "	23 #	24 \$	25 %	26 &	27 '
28 (29)	2A *	2B +	2C ,	2D -	2E .	2F /
30 0	31 1	32 2	33 3	34 4	35 5	36 6	37 7
38 8	39 9	3A :	3B ;	3C <	3D =	3E >	3F ?
40 @	41 A	42 B	43 C	44 D	45 E	46 F	47 G
48 H	49 I	4A J	4B K	4C L	4D M	4E N	4F O
50 P	51 Q	52 R	53 S	54 T	55 U	56 V	57 W
58 X	59 Y	5A Z	5B [5C \	5D]	5E ^	5F _
60 `	61 a	62 b	63 c	64 d	65 e	66 f	67 g
68 h	69 i	6A j	6B k	6C l	6D m	6E n	6F o
70 p	71 q	72 r	73 s	74 t	75 u	76 v	77 w
78 x	79 y	7A z	7B {	7C	7D }	7E ~	7F del

UNICODE

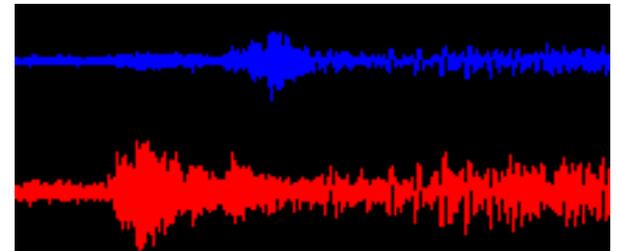
- Uno standard che si propone di affrontare il problema del multilinguismo è *Unicode (Universal Encoding)*.
 - assegna un numero univoco ad ogni simbolo in maniera indipendente dal programma, dalla piattaforma e dalla lingua (e relativo alfabeto): il suo scopo è quello di creare una codifica delle scritture a livello universale.
 - si basa sulla codifica ASCII, ma va oltre la limitazione dell'alfabeto latino potendo codificare caratteri scritti in tutte le lingue del mondo. Originariamente si basava su una codifica a 16 bit che dava la possibilità di codificare più di 65 mila caratteri.
- Per rappresentare qualunque carattere, compresi quelli cinesi e tutte le loro varianti, è stato proposto lo standard *Unicode (ISO-10646)* che utilizza l'*UTF (Unicode Transformation Format)*.
- I formati UTF possono essere a 8, 16 e 32 bit.
 - L'UTF-8 si basa su parole di 8 bit (1 byte) per la codifica dei caratteri; ed usa da 1 a 4 byte per carattere: i primi 128 valori, che iniziano col bit 0, utilizzano 1 byte per carattere e corrispondono all'ASCII, i successivi 1920 (dal 128 al 2047) utilizzano 2 bytes per codificare greco, cirillico, copto, armeno, ebraico, arabo. Per gli altri caratteri si usano 3 o 4 bytes. UTF-16 utilizza parole di 16 bit per codificare i caratteri, viene utilizzato da Java, ed è la rappresentazione interna usata da Windows e Mac OS-X.

I dati multimediali

Immagini



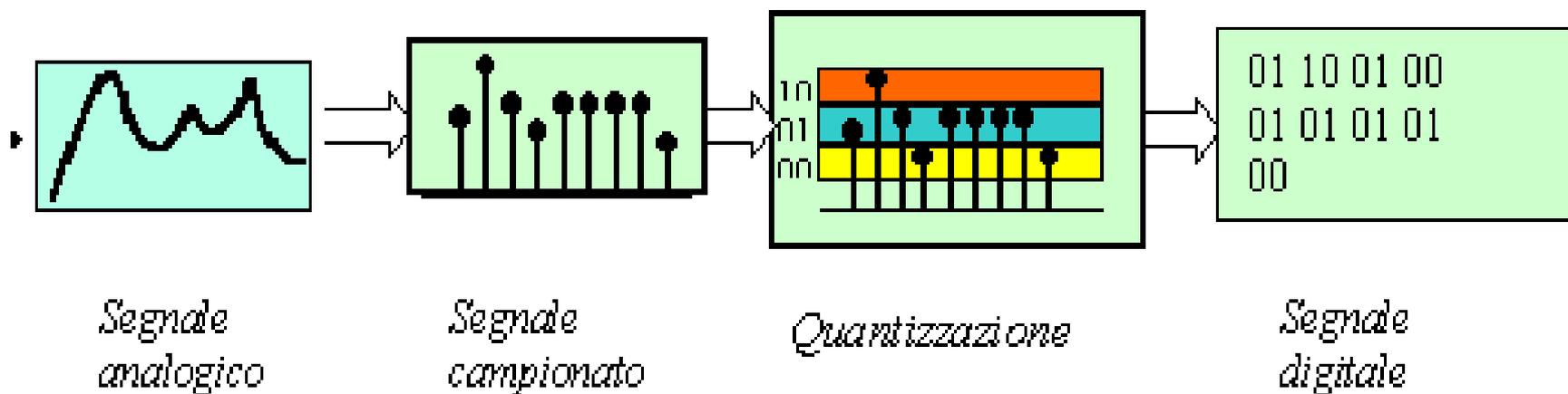
Suoni



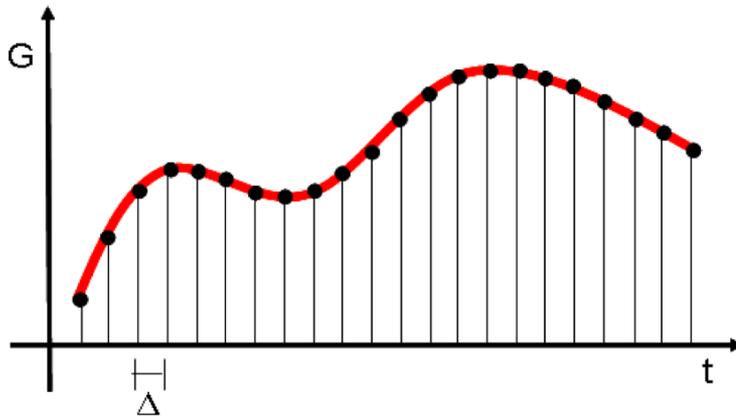
Da analogico a digitale

- Un media analogico può essere rappresentato matematicamente sempre come una funzione *continua* del tempo, mentre una rappresentazione digitale è una rappresentazione discreta di questa.
- La trasformazione da analogico a digitale si realizza per mezzo una operazione detta *campionamento ed una di quantizzazione*:
 - a intervalli regolari di tempo, si va a osservare quali valori assume la funzione analogica e se ne conservano le osservazioni o *campioni*
 - l'operazione di *quantizzazione* approssima *i campioni* ad un certo numero prefissato di livelli

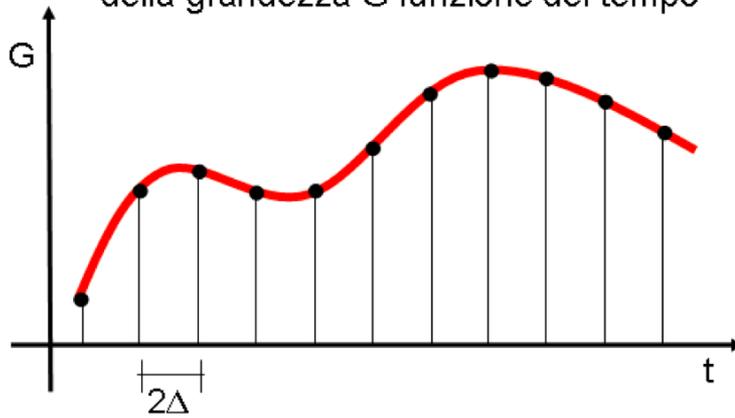
Convertitori Analogici-Digitali



Campionamento

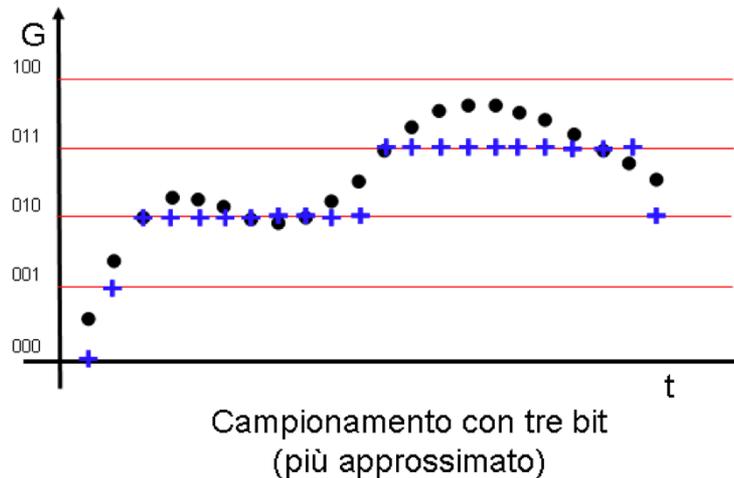
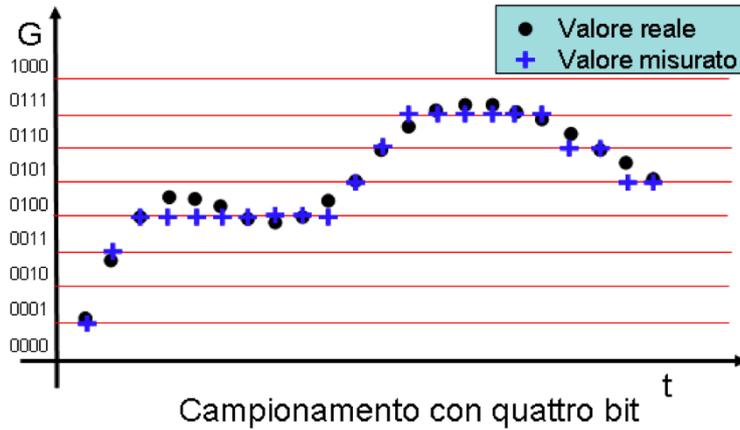


Campionamento con periodo Δ
della grandezza G funzione del tempo



Campionamento meno preciso
con periodo 2Δ della stessa grandezza

Quantizzazione



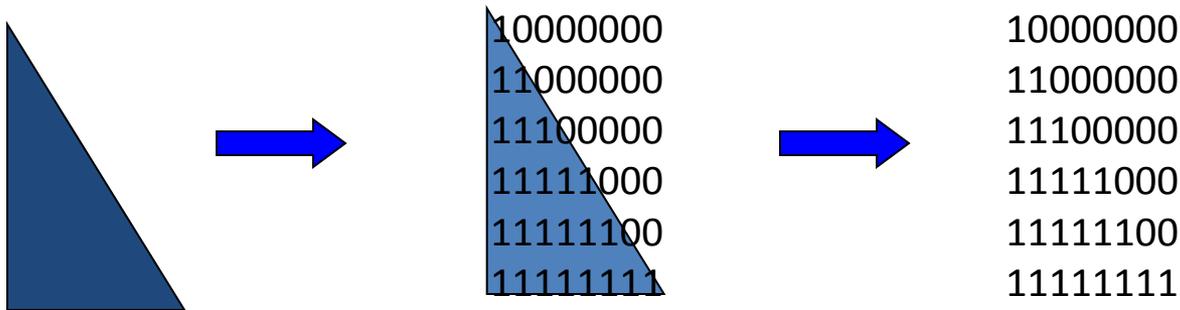
$G(t_1)$	0001	000
$G(t_2)$	0011	001
$G(t_3)$	0100	010
$G(t_4)$	0100	010
$G(t_5)$	0100	010
$G(t_6)$	0100	010
$G(t_7)$	0100	010
$G(t_8)$	0100	010
$G(t_9)$	0100	010
$G(t_{10})$	0100	010
$G(t_{11})$	0101	010
$G(t_{12})$	0110	011
$G(t_{13})$	0111	011
$G(t_{14})$	0111	011
$G(t_{15})$	0111	011
$G(t_{16})$	0111	011
$G(t_{17})$	0111	011
$G(t_{18})$	0111	011
$G(t_{19})$	0110	011
$G(t_{20})$	0110	011
$G(t_{21})$	0101	011
$G(t_{22})$	0101	010

Codifica delle Immagini

- Nel mondo reale, una immagine è un insieme continuo di informazioni
 - luce, colore
- Il calcolatore tratta informazioni discrete
- E' allora necessario scomporre l'informazione in un insieme finito di elementi che verranno codificati con sequenze di bit

Le Immagini BITMAP

- La scomposizione più ovvia consiste nel suddividere l'immagine in un reticolo di punti detti **pixel** (**picture element**)



Bitmap BN e GreyLevel

- Ogni punto del reticolo viene codificato con uno o più bit.
 - per immagini a due soli colori, bianco e nero,
 - 1 bit/pixel
 - per immagini a livelli di grigio, (256 livelli),
 - 8 bit/pixel

Dispositivi di Acquisizione

- Gli “occhi” elettronici (ad esempio, dispositivi ad accoppiamento di carica) acquisiscono una “areola” del mondo, registrandone le caratteristiche di colore, luminosità, etc,
- Più piccola è l’areola, più vicine sono tra loro, più numerose sono, e migliore è la “qualità” dell’immagine acquisita
- Il concetto di risoluzione è legato a *quanto sono fitti i punti che visualizzano l’immagine*.
 - Maggiore è la risoluzione dell’immagine, maggiore è la possibilità di distinguere dettagli in essa presenti.
 - Tutti i pixel contenuti in una immagine digitale hanno dimensioni identiche. La loro dimensione è determinata dalla risoluzione alla quale l’immagine viene digitalizzata:
 - ad esempio la risoluzione di 600 dpi indica che ciascun pixel misura 1/600 di pollice.

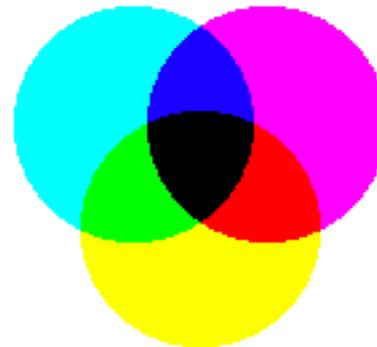
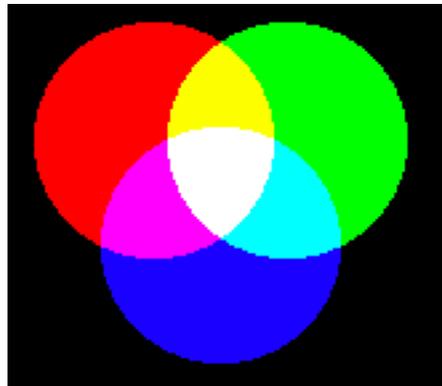


Le immagini a colori

- La *colorimetria* spiega che un colore può essere ottenuto tramite combinazione di almeno tre colori base detti primari
- Se i tre colori base sono il Rosso, il Verde ed il Blu si ha lo spazio RGB

$$\text{Color} = a R + b G + c B$$

- Con 8 bit/colore base, per ogni colore si useranno 24 bit, ovvero circa 16 milioni di colori diversi



Esempio di Immagini Digitali



256 colori



64 colori



16 colori



4 colori



256 livelli grigio



16 livelli grigio



4 livelli grigio



bianco/nero

I formati BITMAP

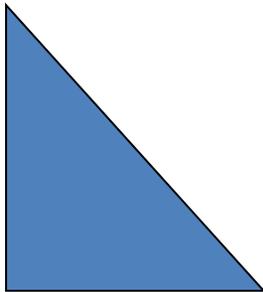
- Ciascuna immagine viene memorizzata con diversi formati bitmap alcuni dei quali prevedono forme di compressione
- Tra i formati più comuni,
 - Tagged Image File Format **TIFF**
 - Graphics Interchange Format **GIF**
 - Joint Photographers Expert Group **JPEG**
 - Microsoft Bit Map **BMP** e Device Independent BitMap **DIB**
 - PC Paintbrush **PCX**

Dimensione dei Bitmap: un esempio

Immagine	Definizione	Colori	Bit	Occupazione
Televisiva	720x625	256	8	440 Kb
SVGA	1024x768	65536	16	1,5 Mb
Fotografia	15000x10000	16 M	24	430MB

Le Immagini Vettoriali

- Una immagine viene descritta in modo astratto attraverso gli elementi grafici di alto livello (*linee, archi, colori*) che la costituiscono



Triangle 0, 0, 0, 100, 100, 100

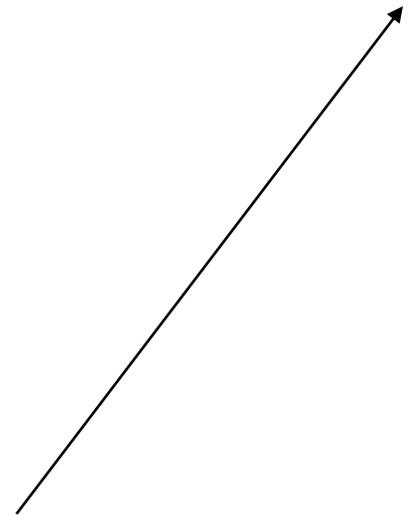
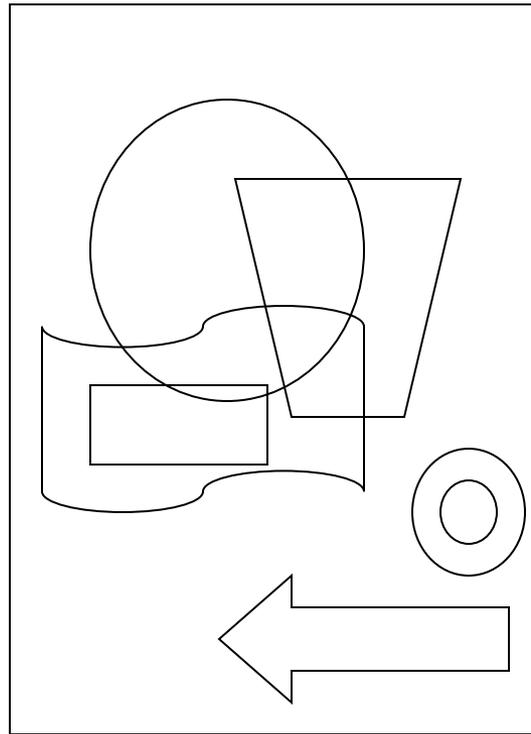
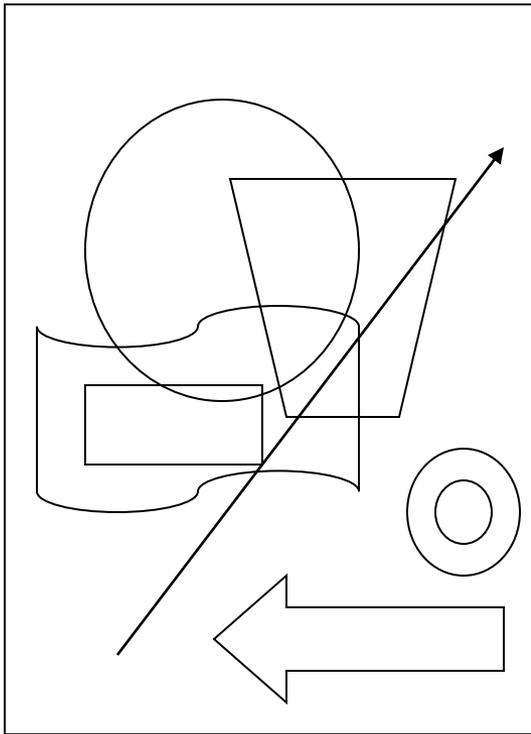
Caratteristiche della Grafica Vettoriale

- Ogni figura è codificata tramite un identificatore e alcuni parametri di posizione (punti, lunghezze di segmenti..)
- Pro
 - Indipendenza dal dispositivo di visualizzazione e dalla sua risoluzione
 - possibilità di modifiche ad alto livello
- Contro:
 - Una immagine reale non è sempre scomponibile in elementi primitivi
 - Per Visualizzare occorre avere lo stesso programma di Generazione del File Vettoriale

I formati Vettoriali

- Tra i formati grafici più diffusi, vanno ricordati
 - PostScript PS
 - EPS
 - PDF
 - Drawing eXchange Format (DXF)
 - Initial Graphics Exchange Specifications (IGES)

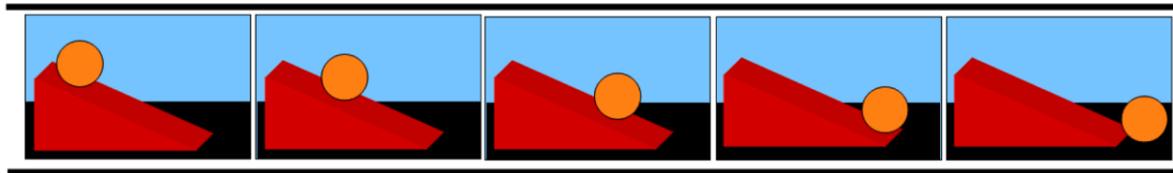
Un disegno astratto



Gli elementi non perdono la loro identità

Le immagini in Movimento

- L'occhio umano ricostruisce l'informazione di movimento se riceve una successione sufficientemente rapida di immagini fisse



- Cinema: 24 fotogrammi/sec
- TV: 25 o 30 fotogrammi/sec
- La sequenza continua di immagini della realtà viene quindi discretizzata ottenendo una serie di immagini (detti *frame*) che variano velocemente, ma a intervalli stabiliti.
- Il *frame-rate* è il numero di frame mostrati per secondo (fps).
- Lo standard MPEG (Moving Picture Expert Group) è sostanzialmente la codifica di ciascun frame fisso, oltre alla codifica di suoni, attraverso tecniche di Compressione Dei Dati.
 - senza compressione, 1 min. di filmato a 24 fotogrammi /sec occuperebbe 644MB

Compressione

- Per risolvere i problemi connessi con le dimensioni elevate sono stati introdotti *processi di compressione*
 - riducono lo spazio occupato mediante o la diminuzione del numero di bit necessari per codificare una singola informazione (*compressione entropica*)
 - ... oppure la diminuzione del numero di informazioni da memorizzare o trasmettere (*compressione differenziale, compressione semantica*). La compressione può conservare integralmente o no il contenuto della rappresentazione originale secondo due tecniche principali:
- la *compressione senza perdita di informazione (lossless, reversibile)* che sfrutta le ridondanze nella codifica del dato;
- la *compressione con perdita di informazione (lossy, irreversibile)* che invece sfrutta le ridondanze nella percezione dell'informazione.

Compressione Lossless

- La compressione lossless avviene tramite una classe di algoritmi che consentono di ricostruire tutta l'informazione iniziale partendo da quella compressa.
- Non sempre si ottengono riduzioni significative.
- Tra le tecniche di compressione lossless si ricordano:
 - la *Run-length encoding (RLE)* che codifica sequenze di valori uguali premettendo un indicatore di ripetizioni al valore codificato;
 - la codifica di *Huffman* che assegna un numero inferiore di bit alle sequenze più probabili attraverso un vettore di codifica;
 - la compressione *Lempel-Ziv-Welch (LZW)* che costruisce dinamicamente una tabella di codifica con numero variabile di bit sulla base delle sequenze incontrate;
 - la *codifica differenziale* in cui ogni dato è rappresentato come differenza rispetto al dato precedente.

Compressione Lossy

- I metodi lossy comportano riduzioni notevoli delle dimensioni, ma la ricostruzione dell'informazione da quella compressa non è però identica a quella iniziale.
- Tali metodi rimuovono parti che possono non essere percepite come avviene nel caso di immagini, video e suoni.
 - Ad esempio gli algoritmi di compressione usati nei formati *GIF* e *JPEG* per immagini fisse sfruttano la caratteristica dell'occhio umano di essere poco sensibile a lievi cambiamenti di colore in punti contigui, e quindi eliminano questi lievi cambiamenti appiattendolo il colore dell'immagine.
- Tra le tecniche di compressione lossy si ricordano:
 - la *compressione JPEG* per le immagini che applica una trasformata nel dominio delle frequenze (*Discrete Cosine Transform*) che permette di sopprimere dettagli irrilevanti riducendo il numero di bit necessari per la codifica;
 - la *compressione MPEG* per i video che codifica parte dei frame come differenze rispetto ai valori previsti in base ad una interpolazione;
 - la *compressione MP3* per l'audio che si basa alle proprietà psicoacustiche dell'udito umano per sopprimere le informazioni inutili.

Codifica dell'Audio

- Il suono è un segnale analogico funzione del tempo consistente in vibrazioni che formano un'onda, la cui ampiezza misura l'altezza dell'onda e il periodo è la distanza tra due onde.
- Anche il suono deve essere campionato e discretizzato per poter essere digitalizzato.
- qualità del suono:
 - il suono riprodotto è diverso da quello originale.
- L'operazione di campionamento discretizza il segnale con una frequenza dell'ordine delle decine di KHz (migliaia di campioni al secondo) perché è dimostrato che *l'orecchio umano percepisce fedelmente il suono originale se il suo campionamento è non inferiore a 30KHz.*

Frequenze di campionamento

Tipo	Frequenza di campionamento (Hz)	Profondità (bit)	Mono/stereo	Dimensione per un minuto
Telefono	8.000	8	mono	469,00 Kb
Parlato	11.025	8	mono	646,00 Kb
Radio mono	22.050	16	mono	2,52 Mb
Radio stereo	22.050	16	stereo	5,05 Mb
Audio Cassetta	44.100	16	stereo	10,10 Mb
Compact Disk	48.000	16	stereo	11,00 MB

Audio: Pulse Code Modulation

Campionamento (Hz)	Bit	Mono/Stereo	Occupazione Kb/s
8.000	8	8/ 16	
8.000	16	16/ 31	
11.025	8	11/ 22	
11.025	16	22/ 43	
22.050	8	22/ 43	
22.050	16	46/ 86	
44.100	8	43/ 86	
44.100	16	86/172	