Rappresentazione e Codifica dell'Informazione

Docente: Ing. Edoardo Fusella
Dipartimento di Ingegneria Elettrica e Tecnologie dell'Informazione
Via Claudio 21, 4° piano – laboratorio SECLAB
Università degli Studi di Napoli Federico II
e-mail: edoardo.fusella@unina.it

Il concetto di Informazione

- *Informazione*
  - cfr. lat. Informatio -onis «nozione, idea, **rappresentazione**»
  - Treccani: dato o elemento che consente di avere **conoscenza** di fatti, situazioni, ecc
  - Wikipedia: un'informazione è uno **scambio di conoscenza** tra due o più persone
  - … 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

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.
  - per es., la lingua italiana oppure unità di una scala di misura per una grandezza fisica

Rappresentazione Analogica

- Rappresentazione Analogica
  - può assumere un **numero infinito di valori**
    - ad esempio la distanza tra due punti nello spazio può assumere un numero infinito di valori.
  - … la rappresentazione **varia in analogia con la grandezza reale**:
    - ad ogni minima variazione della grandezza reale, si ottiene (o si dovrebbe ottenere) un'analoga variazione della rappresentazione (ad esempio, il termometro a mercurio).
    - una grandezza è rappresentata in modo continuo e la gran parte delle grandezze fisiche della realtà sono di tipo continuo.

Rappresentazione Discreta

- Rappresentazione Discreta
  - può assumere un **numero finito di valori**
    - ad esempio, un orologio digitale rappresenta il tempo a salti di minuti o di secondi o di frazioni più piccole.
  - la rappresentazione utilizza un insieme finito di rappresentazioni distinte che vengono messe in relazione con la grandezza reale da rappresentare
    - è un' approssimazione di quella analogica
    - ad ogni variazione della grandezza reale, non si ottiene necessariamente un'analoga variazione della rappresentazione
    - viene rappresentata con numeri e opera manipolando numeri

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).
      - le scale termometriche di Celsius, di Kelvin e di Fahrenheit sono usate per codificare la temperatura

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 , ... , v_k } l'insieme degli ***k*** valori diversi di una data informazione e A = {s_1 , s_2 , ... , 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 elemento di una data informazione v_i deve corrispondere una ed una sola parola codice
  - allora la lunghezza ***l*** deve essere scelta in modo che: $\boldsymbol{n^l \geq k}$

Ridondanza

- Nel caso di $n^l > k$, 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 (ad esempio i numeri delle carte di credito: 16 cifre).
- 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.

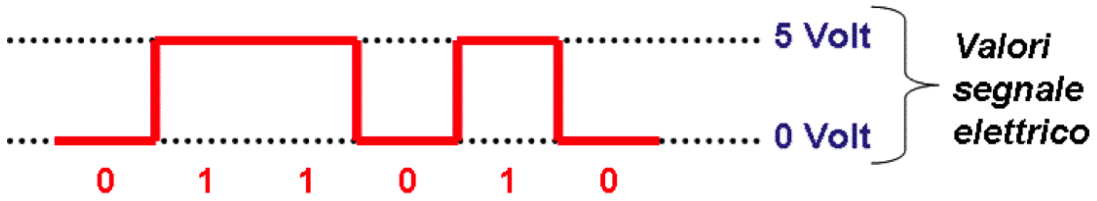
I calcolatori adottano codifiche a lunghezza fissata e definita

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**".
- 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.
- **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 $\mathbf{2}^{l}$ configurazioni (parole codice) diverse.
  - Viceversa se si devono rappresentare K informazioni diverse occorrono $\log_2 K$ bit per associare ad esse codici diversi.
    - $n^l \geq k$ -> $2^l \geq k$ -> $l \geq \log_2 K$

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 **1 byte** = 8 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 ed 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 nè 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* × (*b* - *c*)**	condizione	***a* × *b* – *a* × *c***	Condizione
200	90	88	200 × (90 – 88)	ok	200 × 90 – 200 × 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

intervallo	*periodo*	*Valore*	*divisione*	*resto*
[0,360]	360	1200	1200 : 360	120
[0,60]	60	61	61 : 60	1
[0,60]	60	55	55 : 60	55

- Ad esempio l'orologio con periodo di 12 ore.
  - Se sono le 11 e passano 3 ore l'orologio segnerà le ore 2 e non le ore 14

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
Peso 7	*Peso 6*	*Peso 5*	*Peso 4*	*Peso 3*	*Peso 2*	*Peso 1*	*Peso 0*	

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

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

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

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

- 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.

- …. 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 ..
  - $(1001101)_2 = =1 \times 2^6 + 0 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = 64 + 0 + 0 + 8 + 4 + 0 + 1 = (77)_{10}$

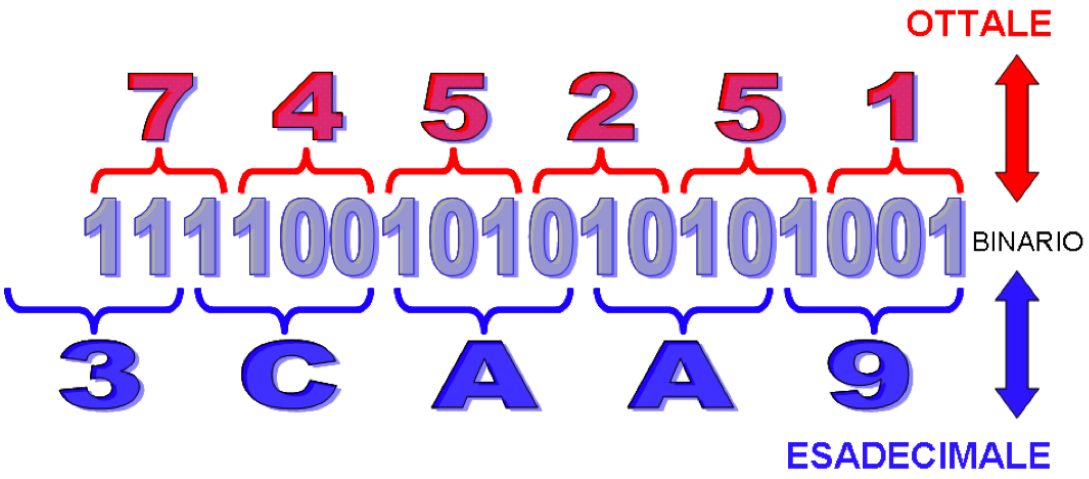
- Così, per evitare di dover trattare con stringhe di bit troppo lunghe, sono molto usati il sistema ***ottale*** **ed** ***esadecimale***.

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:

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

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

  con: $\mathbf{d = d_{pi} + d_{pf}}$
- se si divide la parte intera per 2:

  $\mathbf{d_{pi} /2 = c_i \times 2^{i-1} + c_{i-1} \times 2^{i-2} + \ldots\ldots + c_2 \times 2^1 + c_1 \times 2^0 + c_0 \times 2^{-1}}$, con $\mathbf{c_0}$ resto della divisione.
- Se ora si divide la parte intera ottenuta precedentemente ($\mathbf{d_{pi1}}$ **)**ancora per la base 2:

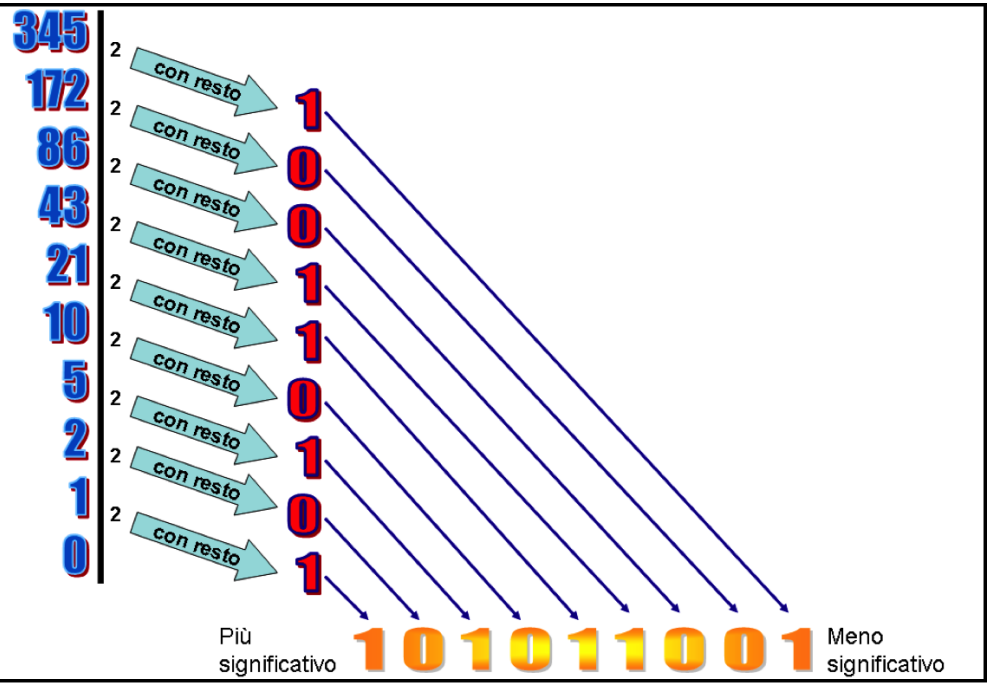
  $\mathbf{d_{pi1} /2 = c_i \times 2^{i-2} + c_{i-1} \times 2^{i-3} + \ldots\ldots + c_2 \times 2^0 + c_1 \times 2^{-1}}$, con $\mathbf{c_1}$ resto della divisione
- …. il procedimento deve essere ripetuto fino a quando si ottiene un quoziente uguale a 0.

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:

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

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

per spostare $\mathbf{c_{-1}}$ 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 $\mathbf{d_{pfi\text{-}esima}}$ non si annulla;

  la parte frazionaria $\mathbf{d_{pfi\text{-}esima}}$ si ripete con periodicità;

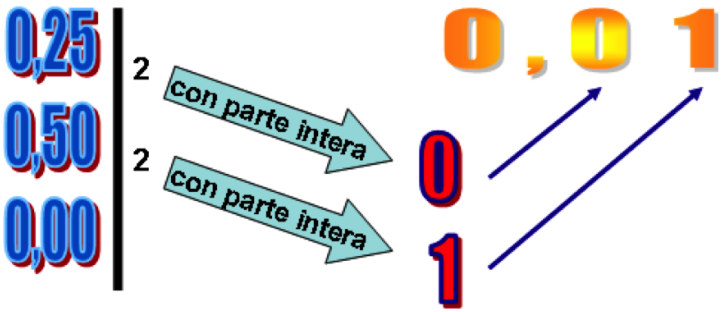
- …. ci si accontenta di una rappresentazione approssimata con un numero di bit inferiore a quello che andrebbero calcolati per raggiungere una delle condizioni precedenti.
  - solo la prima condizione garantisce una conversione senza approssimazione

… procedimento

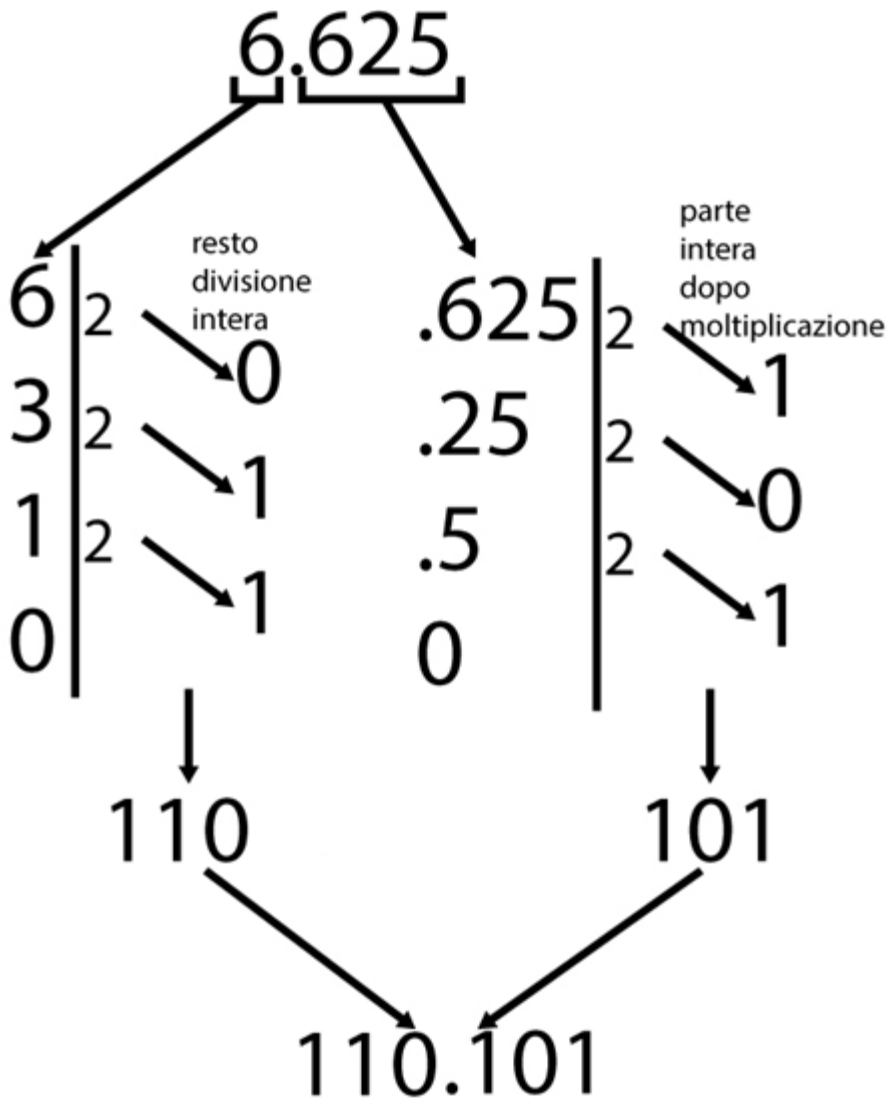
ALGORITMO

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.



Esempio


6.625
6
3
1
0
2
2
2
resto
divisione
intera
0
1
1
.625
.25
.5
0
2
2
2
parte
intera
dopo
moltiplicazione
1
0
1
110
101
110.101

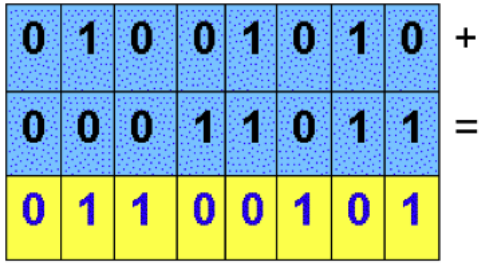
Operazioni sui binari

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

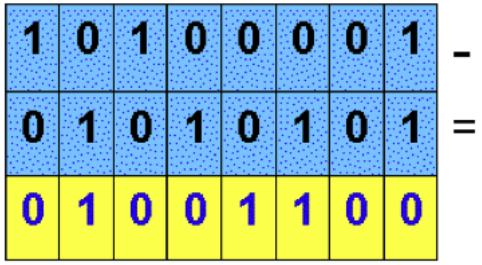
a	b	a+b	a*b
0	0	0	0
0	1	1	0
1	0	1	0
1	1	10	1

Si noti che 1+1 fa 0 con riporto 1

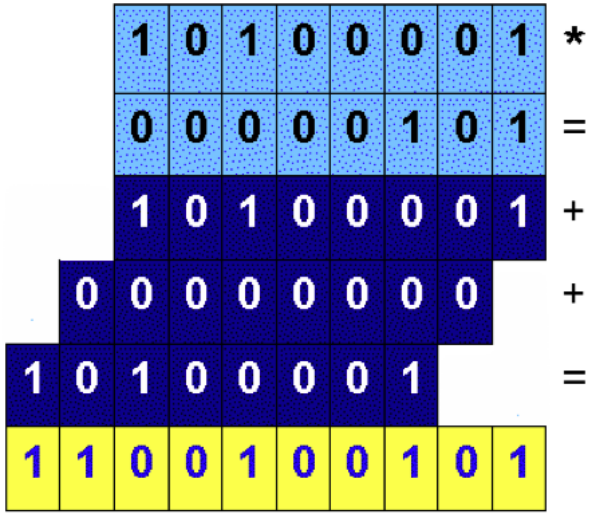
Esempi



Somma



Sottrazione



Prodotto

Sottrazione esempio

$0 - 0 = 0$
$1 - 0 = 1$
$1 - 1 = 0$
$0 - 1 = 1$ dopo essersi prestato 1 dalla colonna a sinistra

Sottrazione decimale

$$\begin{array}{r} {\scriptstyle 6}\ \ {\scriptstyle 3} \\ 7\ 4\ 5\ 9 - \\ {\scriptstyle 1}\ \ {\scriptstyle 1} \\ 7\ 8\ 9 = \\ \hline 6\ 6\ 7\ 0 \end{array}$$

Sottrazione binaria

$(10010)_2 - (1101)_2 = ?$

$$\begin{array}{r} {\scriptstyle 0}\ {\scriptstyle 1}\ \ \ {\scriptstyle 0} \\ 1\ 0\ 0\ 1\ 0 - \\ {\scriptstyle 1}\ \ {\scriptstyle 1}\ \ {\scriptstyle 1} \\ 1\ 1\ 0\ 1 = \\ \hline 0\ 0\ 1\ 0\ 1 \end{array}$$

Pertanto $(10010)_2 - (1101)_2 = (101)_2$

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

$$f(x) = \begin{cases} x & \text{se } x \geq 0 \\ 2^n - |x| & \text{se } x < 0 \end{cases}$$

Come si calcola

- Inversione bit + 1

19 -> -19

19 in complemento a 2	010011
Inverti i bit	101100
Somma 1	101101

-23 -> 23

-23 in complemento a 2	101001
Inverti i bit	010110
Somma 1	010111

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:

  $[0, 2^{l-1} - 1]$ per i valori positivi

  $[-2^{l-1}, -1]$ per i valori negativi.

- l'intervallo non è simmetrico:

  2^{l-1} valore assoluto del minimo

  $2^{l-1} - 1$ 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 *l* bit che rappresenta un numero intero rappresentato 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.

si vede se il MSB è 0/1	010011
il numero rappresentato è positivo, normale conversione binario-decimale	19

si vede se il MSB è 0/1	101100
il numero rappresentato è negativo, si complementa	010011
Somma 1	010100
normale conversione binario-decimale	-20

Interpretazione del complemento a due

- Dati *l* 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}:

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

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 a due (somma e sottrazione)

12-14=12+(-14)

Memorizza 14	001110
Inverti i bit	110001
Somma 1, ottieni -14	110010
Memorizza 12	001100
Somma -14 e 12	111110

-2

Complemento diminuito

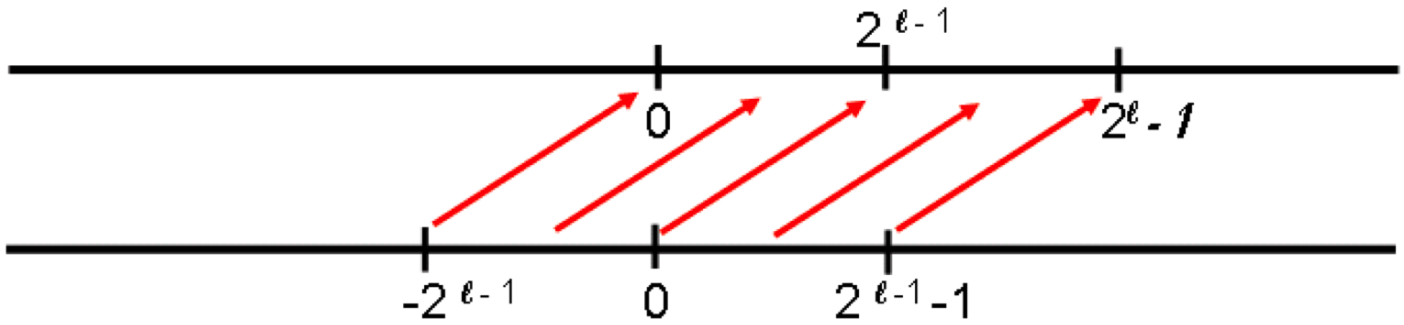
- Il *complemento a uno* (***x'***) del numero ***x*** si differenzia dal complemento a 2 (***x"***) dello stesso numero per una unità:

$$x' = x'' - 1$$

- 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 1/2

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

Overview 2/2

Rappresentazione di interi con 4 bit

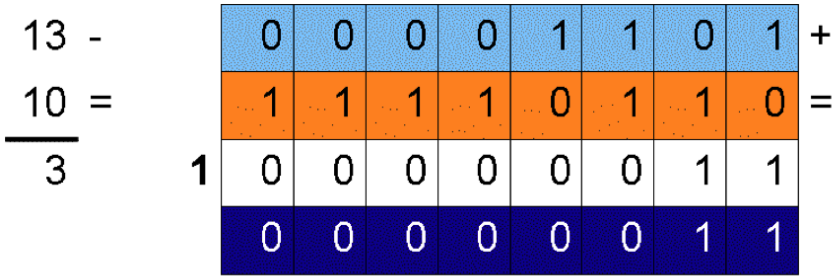
Decimale	Senza segno	Segno e modulo	Complemento a uno	Complemento a due	Eccesso 8
+8	1000	n/d	n/d	n/d	n/d
+7	0111	0111	0111	0111	1111
+6	0110	0110	0110	0110	1110
+5	0101	0101	0101	0101	1101
+4	0100	0100	0100	0100	1100
+3	0011	0011	0011	0011	1011
+2	0010	0010	0010	0010	1010
+1	0001	0001	0001	0001	1001
(+)0	0000	0000	0000	0000	1000
(−)0	n/d	1000	1111	n/d	n/d
−1	n/d	1001	1110	1111	0111
−2	n/d	1010	1101	1110	0110
−3	n/d	1011	1100	1101	0101
−4	n/d	1100	1011	1100	0100
−5	n/d	1101	1010	1011	0011
−6	n/d	1110	1001	1010	0010
−7	n/d	1111	1000	1001	0001
−8	n/d	n/d	n/d	1000	0000

Quali si usano?

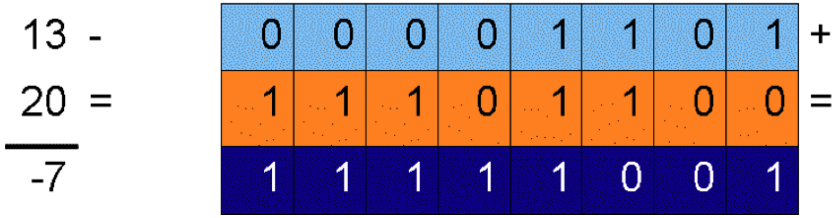
- La rappresentazione in complemento a due è la più efficiente per svolgere operazioni in aritmetica binaria poiché permette di trattare la sottrazione tra numeri come una somma tra numeri di segno opposto:

  (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

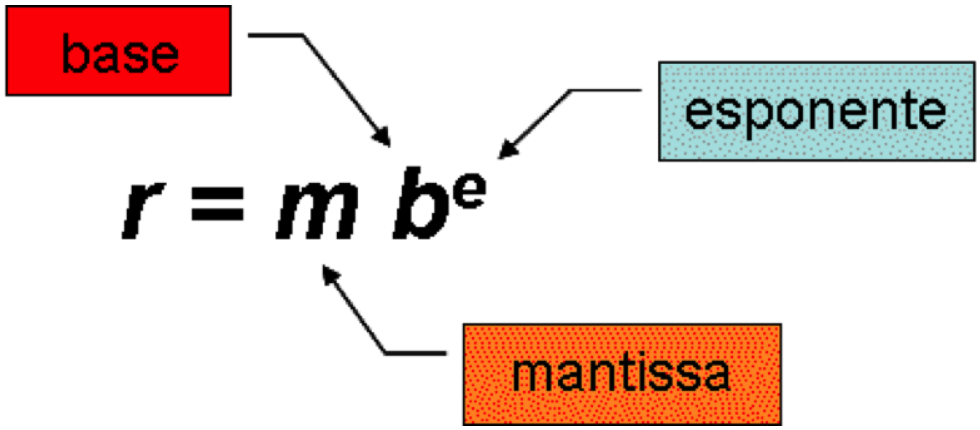


Si noti che il 9 bit di overflow si perde in quanto la rappresentazione si compone di soli 8 bit



Numeri Reali

- I numeri reali vengono rappresentati in binario attraverso la seguente notazione scientifica: $r=mb^{e,}$ 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

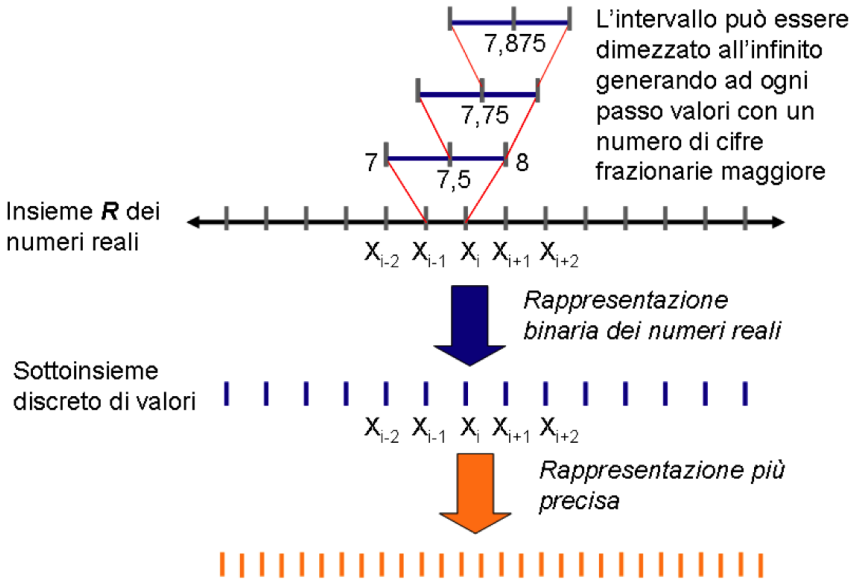


Notazione Scientifica

- 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;

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

- 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.
- Rappresentando un numero reale in binario, si commette un errore di rappresentazione dovuto al fatto che abbiamo a disposizione un numero limitato di cifre per la rappresentazione

$$\varepsilon_A \equiv \left| x - X \right| \qquad \varepsilon_R \equiv \frac{\left| x - X \right|}{\left| X_{j+1} - X_j \right|}$$

  - dove ε rappresenta l'errore (assoluto o relativo) che si commette sostituendo x (numero da rappresentare) con X (rappresentazione binaria), e :
    - $X = X_i$ se si approssima per difetto;
    - $X = X_{i+1}$ se si approssima per eccesso.
- Primo esempio:

$$x = 9.9, X = 10, X_j = 9, X_{j+1} = 10 \qquad \varepsilon_A = 0.1 \qquad \varepsilon_R = 0.1$$

- Secondo esempio:

$$x = 999.9, X = 1000, X_j = 900, X_{j+1} = 1000 \qquad \varepsilon_A = 0.1 \qquad \varepsilon_R = 0.001$$

Esempio errori di arrotondamento

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

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

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

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: $0{,}48 \times 10^3$ è uguale a 480×10^0, 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 $*10^0$ è equivalente 2,5000 * $10^{-3)}$.
- 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

$$0 < x < 1{,}0000 \times 10^{-99}$$

Osservazione

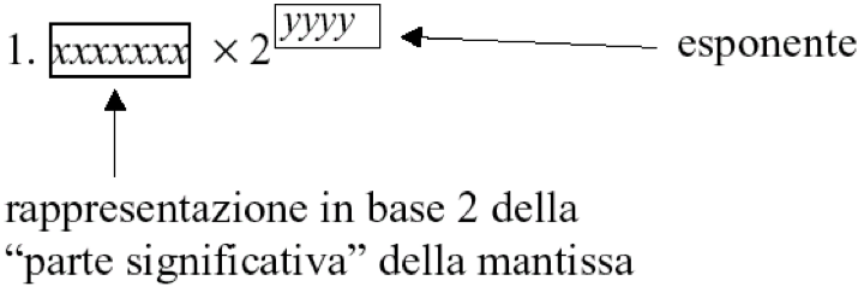
- … gli intervalli $[X_i, X_{i+1}]$ non hanno tutti la stessa ampiezza a causa della finitezza del numero di cifre della mantissa:
  - man mano che ci si avvicina alla condizione di overflow gli intervalli si fanno sempre più ampi
  - … mentre intorno alla condizione di underflow non solo si addensano 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: $100\times10^{0} + 100\times10^{-2} = 100\times10^{0} + 1\times10^{0} = 101\times10^{0}$
  - mentre per il prodotto e la divisione servono operazioni separate sulle mantisse e sugli esponenti: $100\times10^{0} * 100\times10^{-2} = (100 * 100)\times10^{(0-2)} = 10000\times10^{-2}$
- L'allineamento degli esponenti produce come effetto indesiderato quello di far scomparire alcune cifre rappresentative del numero.
  - Ad esempio la somma dei numeri seguenti: $1{,}9099\times10^{1} + 5{,}9009\times10^{4}$ nell'esempio precedente, diventa: $0{,}0001\times10^{4} + 5{,}9009\times10^{4}$,
  - … 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.

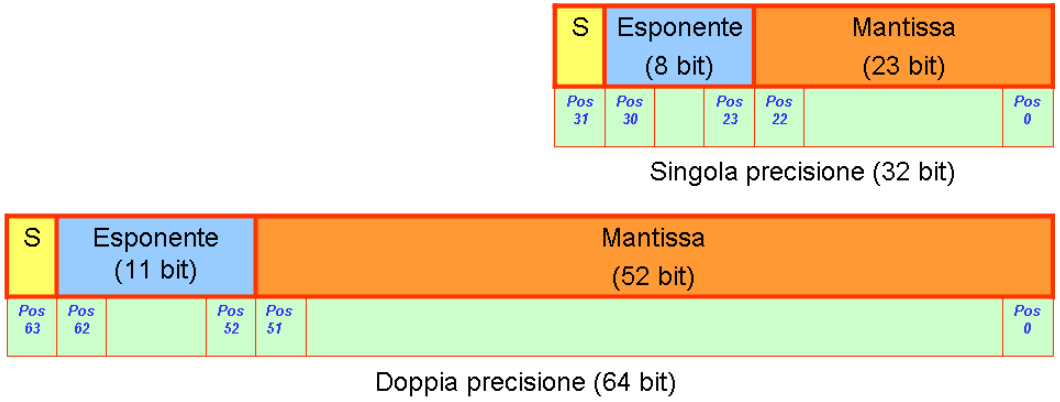
Perché rappresentare in virgola mobile?

- .. 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



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).



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]

IEEE 754 a 32 bit: esponente

- Esponente (8bit)
  - Rappresentato in eccesso 127 (polarizzazione o bias)
  - L'intervallo di rappresentazione è [-127, 128]
  - Le due configurazioni estreme sono riservate [-126, 127]
    - Numero più grande 11….11; più piccolo 00..00: per confrontare due interi polarizzati, per determinare il minore basta considerarli come interi senza segno
  - Esempio:
    - 10100011 = 163 in b=10; 163-127 = 36
    - 00100111 = 39 in b= 10; 39-127=-88

Esempi:

- 1 10000001 01000000000000000000000
  - Segno negativo;
  - Esponente: $2^7+2^0=129-127=2$
  - Mantissa: $1+2^{-2}=1.25$
  - Numero: $-1.25\ 2^2=-5$

Esempi

- 8.5
- Segno +
- 8.5 in binario: 1000 .1=1.0001 2^3
  - Mantissa: 00010000000000000000000
  - Esponente: 3+127=130=10000010
  - **NUMERO: 0 10000010 00010000000000000000000**

Configurazioni particolari

- **Esponente e mantissa tutti 0** : rappresentano 0
- **Mantissa tutti 0 ed esponente tutti 1**: rappresentano infinito
- **Mantissa diversa da 0 e esponente tutti 1**: rappresentato la situazione di **Not a Number** (NAN), cioè un valore indefinito (esempio il risultato di una divisione per 0 o la radice quadrata di un numero negativo)

Rappresentazione di caratteri

- La necessità di elaborare caratteri si presenta in moltissimi contesti
- Quando parliamo di caratteri ci riferiamo a :
  - Lettere maiuscole e minuscole dell'alfabeto inglese
  - Cifre decimali
  - Caratteri di interpunzione (!, ?, .,..)
  - Altri caratteri (più, meno, underscore..)

  Per un totale di quasi 100 caratteri (ci vogliono 7 bit)
- Esistono varie possibilità di costruire la tabella di rappresentazione dei caratteri
  - l'ente di standardizzazione americano ANSI ha definito nel 1968 la tabella ASCII (**A**merican **S**tandard **C**ode for **I**nformation **I**nterchange) usata come standard

Tabella ASCII (1/2)

carattere	codice ASCII	equivalente esadecimale	equivalente numerico	carattere	codice ASCII	equivalente esadecimale	equivalente numerico
NUL	00000000	00	0	*spazio*	00100000	20	32
SOH	00000001	01	1	!	00100001	21	33
STX	00000010	02	2	”	00100010	22	34
ETX	00000011	03	3	#	00100011	23	35
EOT	00000100	04	4	$	00100100	24	36
ENQ	00000101	05	5	%	00100101	25	37
ACK	00000110	06	6	&	00100110	26	38
BEL	00000111	07	7	’	00100111	27	39
BS	00001000	08	8	(	00101000	28	40
TAB	00001001	09	9	)	00101001	29	41
LF	00001010	0A	10	*	00101010	2A	42
VT	00001011	0B	11	+	00101011	2B	43
FF	00001100	0C	12	,	00101100	2C	44
CR	00001101	0D	13	-	00101101	2D	45
SO	00001110	0E	14	.	00101110	2E	46
SI	00001111	0F	15	/	00101111	2F	47
DLE	00010000	10	16	0	00110000	30	48
DC1	00010001	11	17	1	00110001	31	49
DC2	00010010	12	18	2	00110010	32	50
DC3	00010011	13	19	3	00110011	33	51
DC4	00010100	14	20	4	00110100	34	52
NAK	00010101	15	21	5	00110101	35	53
SYN	00010110	16	22	6	00110110	36	54
ETB	00010111	17	23	7	00110111	37	55
CAN	00011000	18	24	8	00111000	38	56
EM	00011001	19	25	9	00111001	39	57
SUB	00011010	1A	26	:	00111010	3A	58
ESC	00011011	1B	27	;	00111011	3B	59
FS	00011100	1C	28	<	00111100	3C	60
GS	00011101	1D	29	=	00111101	3D	61
RS	00011110	1E	30	>	00111110	3E	62
US	00011111	1F	31	?	00111111	3F	63

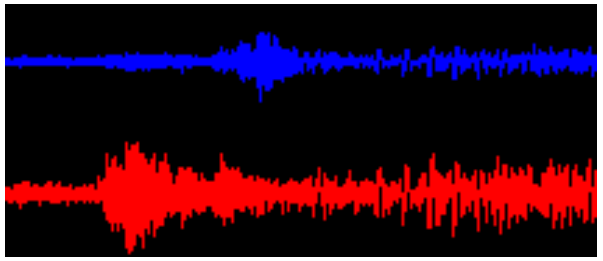
Tabella ASCII (2/2)

carattere	codice ASCII	equivalente esadecimale	equivalente numerico	carattere	codice ASCII	equivalente esadecimale	equivalente numerico
@	01000000	40	64	`	01100000	60	96
A	01000001	41	65	a	01100001	61	97
B	01000010	42	66	b	01100010	62	98
C	01000011	43	67	c	01100011	63	99
D	01000100	44	68	d	01100100	64	100
E	01000101	45	69	e	01100101	65	101
F	01000110	46	70	f	01100110	66	102
G	01000111	47	71	g	01100111	67	103
H	01001000	48	72	h	01101000	68	104
I	01001001	49	73	i	01101001	69	105
J	01001010	4A	74	j	01101010	6A	106
K	01001011	4B	75	k	01101011	6B	107
L	01001100	4C	76	l	01101100	6C	108
M	01001101	4D	77	m	01101101	6D	109
N	01001110	4E	78	n	01101110	6E	110
O	01001111	4F	79	o	01101111	6F	111
P	01010000	50	80	p	01110000	70	112
Q	01010001	51	81	q	01110001	71	113
R	01010010	52	82	r	01110010	72	114
S	01010011	53	83	s	01110011	73	115
T	01010100	54	84	t	01110100	74	116
U	01010101	15	85	u	01110101	75	117
V	01010110	56	86	v	01110110	76	118
W	01010111	57	87	w	01110111	77	119
X	01011000	58	88	x	01111000	78	120
Y	01011001	59	89	y	01111001	79	121
Z	01011010	5A	90	z	01111010	7A	122
[	01011011	5B	91	{	01111011	7B	123
\	01011100	5C	92	|	01111100	7C	124
]	01011101	5D	93	}	01111101	7D	125
^	01011110	5E	94	~	01111110	7E	126
_	01011111	5F	95	□	01111111	7F	127

Esempio


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

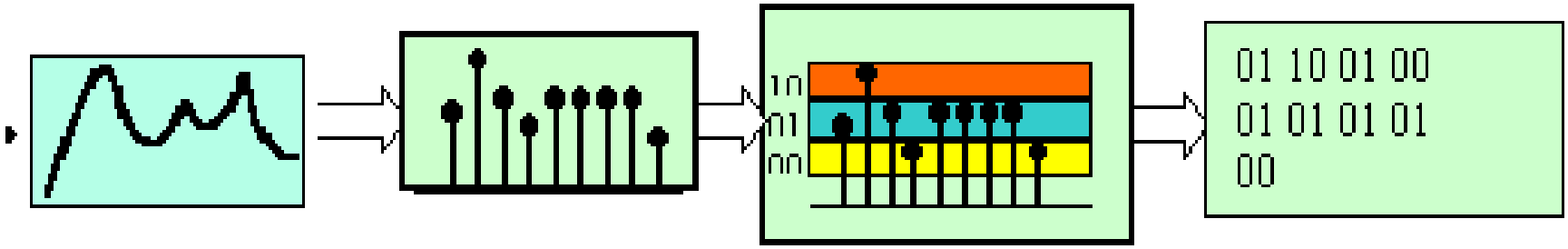
I dati multimediali



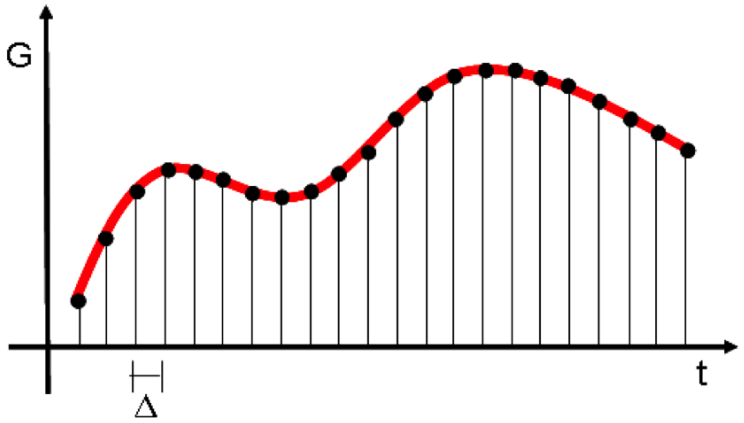
Da analogico a digitale

- 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

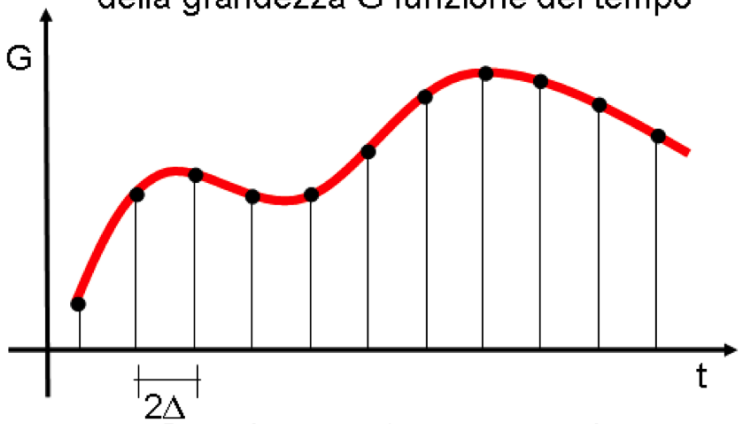
Convertitori Analogici-Digitali



Campionamento

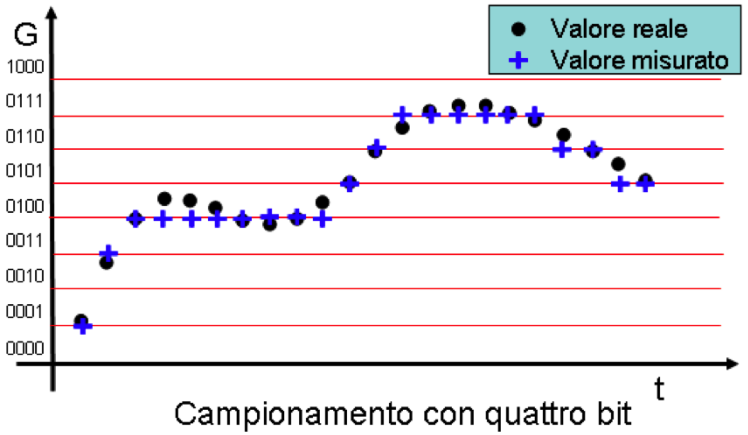


Campionamento con periodo Δ
della grandezza G funzione del tempo

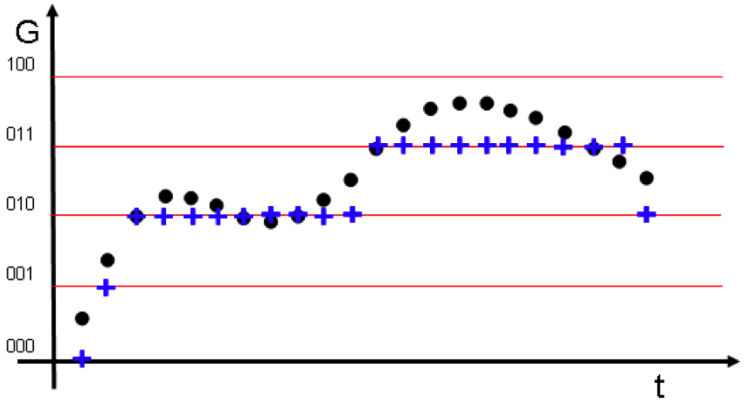


Campionamento meno preciso
con periodo 2Δ della stessa grandezza

Quantizzazione



Campionamento con quattro bit



Campionamento con tre bit
(più approssimato)

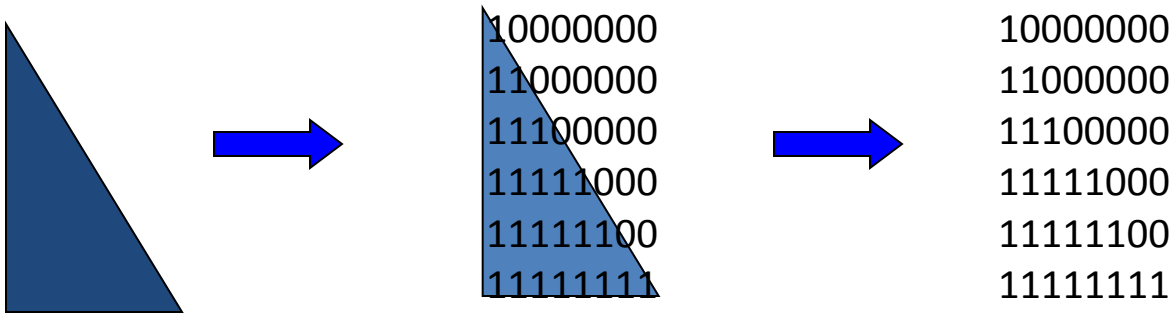
$G(t_1)$	0001	000
$G(t_2)$	0011	001
$G(t_3)$	0100	010
$G(t_4)$	0100	010
$G(t_5)$	0100	010
$G(t_6)$	0100	010
$G(t_7)$	0100	010
$G(t_8)$	0100	010
$G(t_9)$	0100	010
$G(t_{10})$	0100	010
$G(t_{11})$	0101	010
$G(t_{12})$	0110	011
$G(t_{13})$	0111	011
$G(t_{14})$	0111	011
$G(t_{15})$	0111	011
$G(t_{16})$	0111	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** (**pic**ture **el**ement)



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

Risoluzione

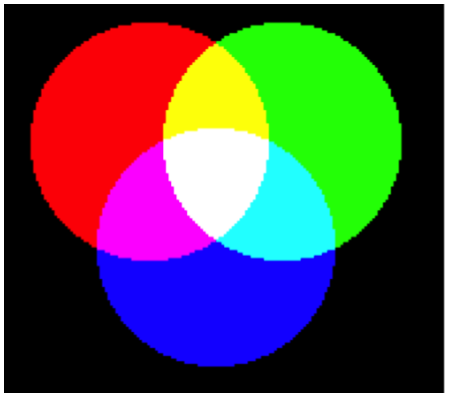
- Il concetto di risoluzione è legato a *quanto sono fitti i punti che visualizzano l'immagine*.
  - Maggiore è la risoluzione dell'immagine, maggiore è la possibilità di distinguere dettagli in essa presenti.
  - Tutti i pixel contenuti in una immagine digitale hanno dimensioni identiche. La loro dimensione 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.

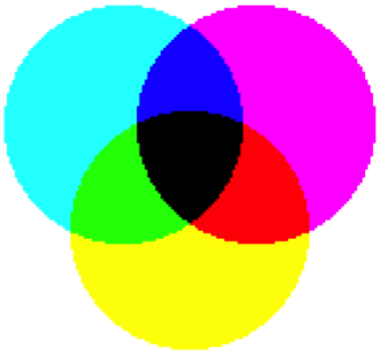
Le immagini a colori

- 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

  $$Color = a\,R + b\,G + c\,B$$

- 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

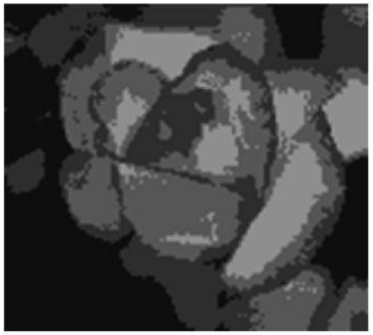
64 colori

16 colori

4 colori

256 livelli grigio

16 livelli grigio



4 livelli grigio

bianco/nero

I formati BITMAP

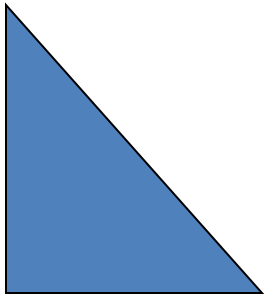
- Ciascuna immagine viene memorizzata con diversi formati bitmap alcuni dei quali prevedono forme di compressione
- Tra i formati più comuni,
  - Tagged Image File Format **TIFF**
  - Graphics Interchange Format **GIF**
  - Joint Photographers Expert Group **JPEG**
  - Microsoft Bit Map **BMP**

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



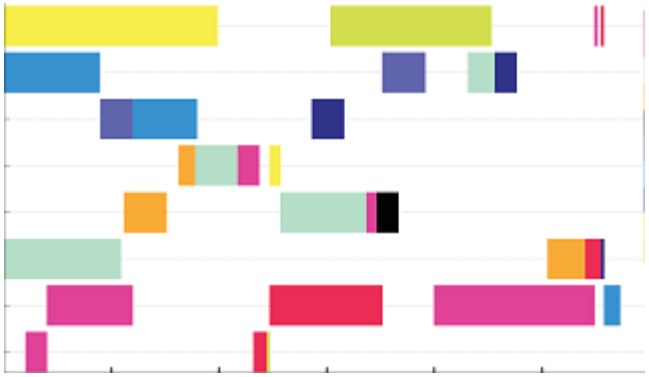
Triangle 0, 0, 0, 100, 100, 100

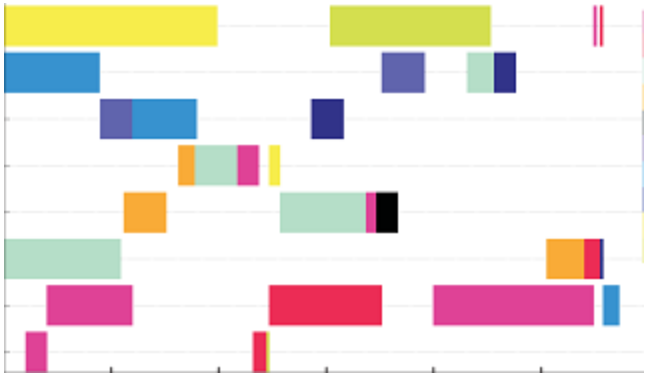
- Pro
  - Indipendenza dal dispositivo di visualizzazione e dalla sua risoluzione
- Contro:
  - Una immagine reale non è sempre scomponibile in elementi primitivi

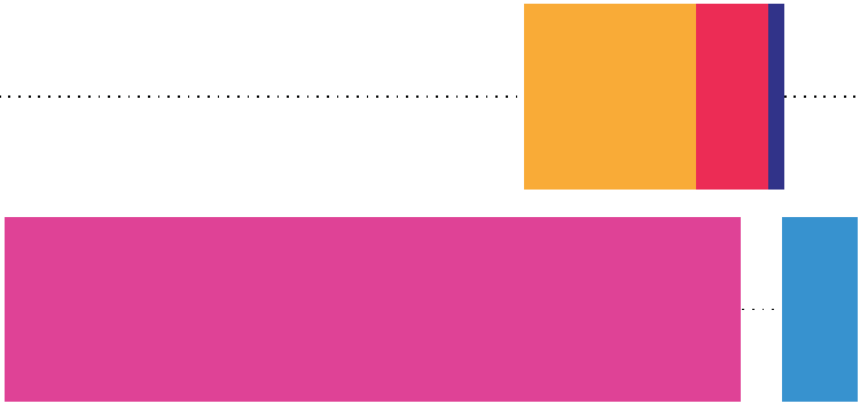
I formati Vettoriali

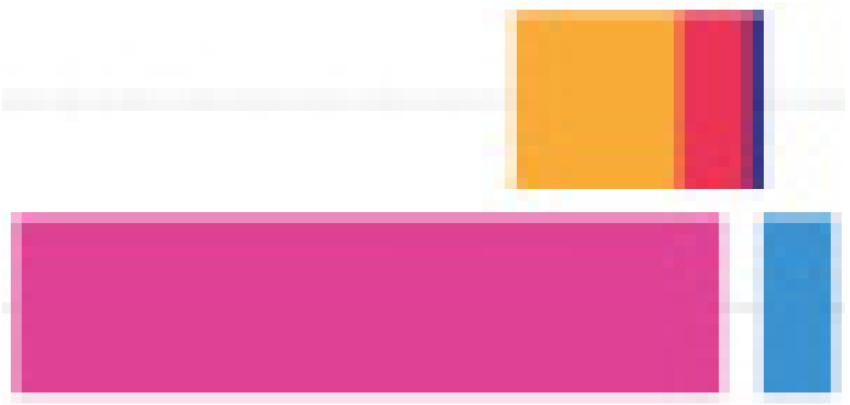
- Tra i formati grafici più diffusi, vanno ricordati
  - EPS
  - PDF

Vettoriale vs BITMAP



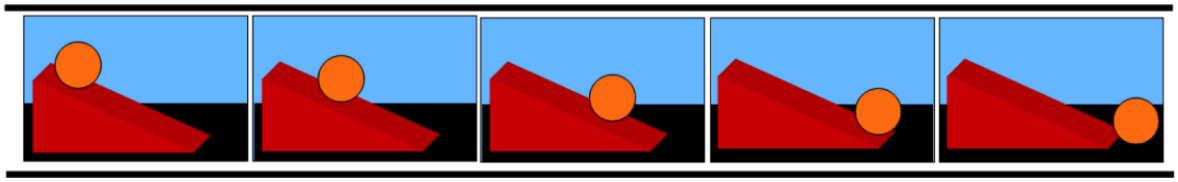






Le immagini in Movimento

- 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
  - … oppure la diminuzione del numero di informazioni da memorizzare o trasmettere
- 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.
- Esempio lossless:
  - la codifica di *Huffman* che assegna un numero inferiore di bit alle sequenze più probabili attraverso un vettore di codifica

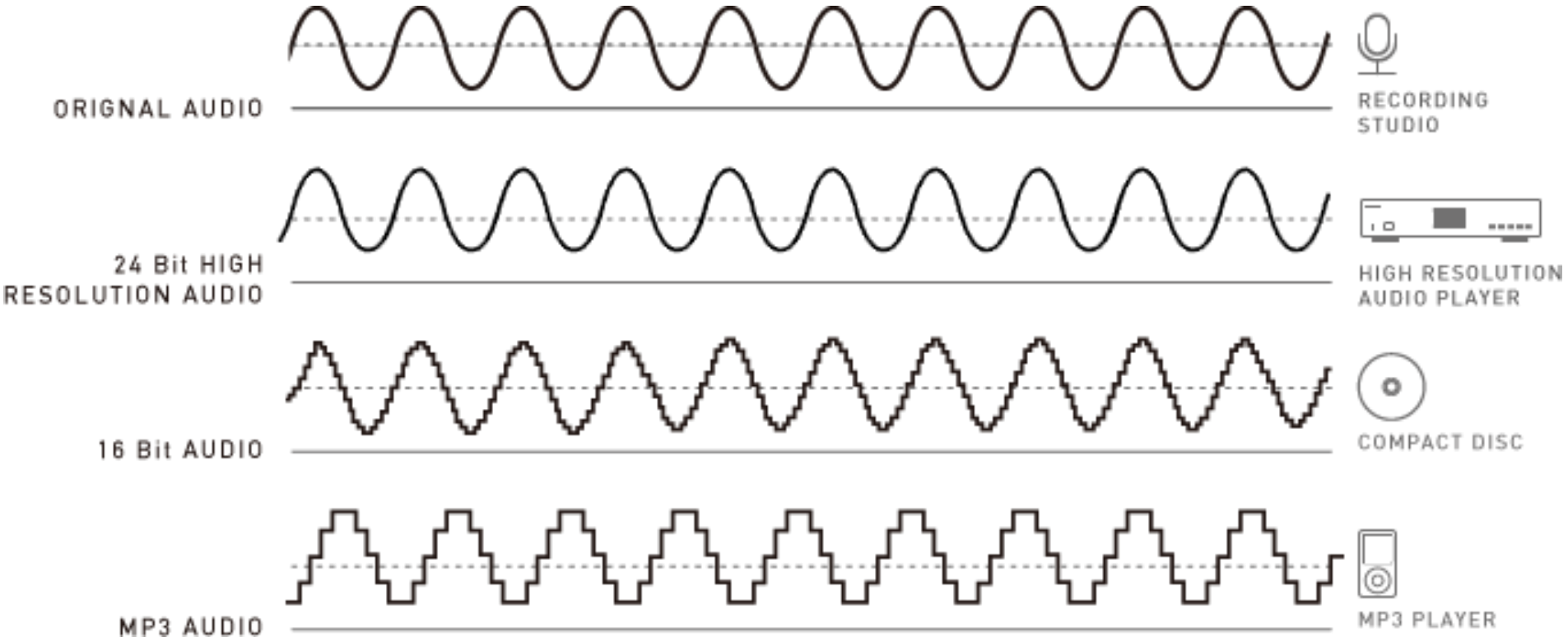
Compressione Lossy

- 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 appiattendo 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.*

Esempio


ORIGNAL AUDIO
RECORDING STUDIO
24 Bit HIGH RESOLUTION AUDIO
HIGH RESOLUTION AUDIO PLAYER
16 Bit AUDIO
COMPACT DISC
MP3 AUDIO
MP3 PLAYER

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

Esercizi di riepilogo (1/4)

1. Si vuole fare l'inventario di un laboratorio, in cui sono presenti: 10 pc, 5 scrivanie, 7 sedie, un attaccapanni e 2 armadi. A ciascun oggetto viene assegnato un identificativo binario univoco: quanti bit sono necessari per rappresentare l'identificativo?
2. Esprimere il numero binario 1101.1 in notazione posizionale e convertirlo in base 10
3. Convertire i seguenti numeri binari in decimale, ottale ed esadecimale
   - 11000000
   - 111010101100
   - 1110101
4. Convertire in binario i seguenti numeri
   - 2.45
   - 34.08
   - 101.55

Esercizi di riepilogo (2/4)

5. Convertire il numero binario 11001110 in base 5
6. Calcolare le seguenti somme algebriche
   - 1010 + 0011
   - 0111 + 0011
   - 1110 + 0100
   - 1010 – 0011
   - 1100 – 0001
   - 1001 – 0111
7. Eseguire le seguenti operazioni
   - 0110 x 0011
   - 0011 x 0010
   - 1010 : 11
   - 1101 : 10

Esercizi di riepilogo (3/4)

8. Si dia la rappresentazione in complemento a 1 e in complemento a 2 di -123 con parole di 8 bit.
9. Che numero rappresenta la sequenza 11100110 di 8 bit se il sistema di rappresentazione è in complemento a 2?
10. Che rappresentazione ha la sequenza di 8 bit in complemento a 2 11100110 se la parola viene allungata a 12 bit? E la sequenza 00010011?
11. Calcolare la seguenti somme tenendo conto che gli addendi sono numeri espressi in complementi a 2
    - 1000 + 0110
    - 0111 + 0001
    - 1110 + 1001

Esercizi di riepilogo (4/4)

12. Convertire il numero -30,375 in formato a virgola mobile IEEE 754 (precisione singola)
13. Che numero rappresenta la seguente congurazione binaria in formato IEEE 754 ? 01000110010001100000000000000000

Soluzione1

$$
\begin{aligned}
-30.375 &= (-11110.011)_{binario} \\
&= (-1.1110011)_{binario} \times 2^4 \\
&= (-1)1 \times (1 + 0.1110011) \times 2^{(131-127)}
\end{aligned}
$$

Ricordando che il formato IEEE 754 utilizza il seguente schema di

$(-1)^{segno} \times (1 + significando) \times 2^{(esponente-127)}$

Abbiamo:

$segno = 1$

$esponente = 131 = (10000011)_{binario}$

$significando = (11100110000000000000000)_{binario}$

e quindi:

$(-30.375)10 = (1\ 10000011\ 11100110000000000000000)_{binario}$

Soluzione2

Configurazione da convertire

0 10001100 10001100000000000000000

segno $0 \rightarrow$ segno +

esponente $10001100 \rightarrow 140$ decimale, a cui bisogna sottrarre la polarizzazione (127) per ottenere il vero esponente, cioè 13

significando $10001100000000000000000 \rightarrow 1+2^{-1}+2^{-5}+2^{-6}=1,546875$

Pertanto il numero è dato da

$$+1 \times 1,546875 \times 2^{13} = 12672,0$$