Corso di Laurea triennale in Ingegneria Chimica

in condivisione con

Corso di Laurea triennale in

Ingegneria Navale e Scienze dei Materiali

Elementi di Informatica

A.A. 2016/17

prof. Mario Barbareschi

Rappresentazione e codifica delle informazioni

Informazione

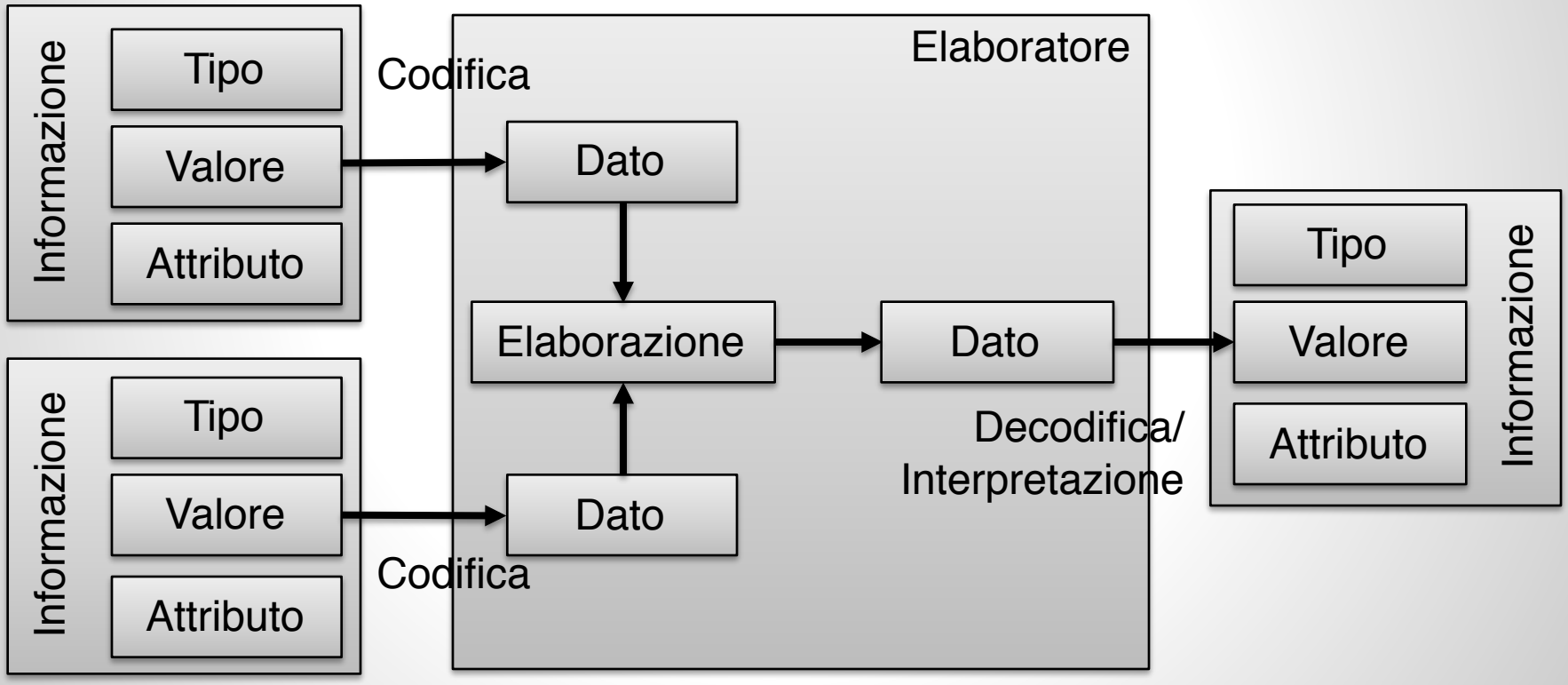
- **L'informazione** è legata al concetto di **scelta**: esse sono caratterizzate dalla tripla {tipo, valore, attributo}:
  - **Tipo**: definisce **l'insieme degli elementi** fra i quali si compie la scelta.
  - **Valore**: è il particolare **elemento scelto**.
  - **Attributo**: è un **identificatore** che **associa un significato al valore,** dando senso all'informazione.

Es.: "L'ora di New York è 12:00pm", "La capitale della Francia è Parigi", "Il semaforo di ingresso in galleria è rosso", "Mario ha 18 anni".

- Affinché una informazione possa essere utilizzata è necessario rappresentarla.

La trasformazione delle informazioni

I calcolatori elaborano **dati** ovvero **rappresentazioni dei valori delle informazioni!**
L'elaboratore opera sulle rappresentazioni dei valori delle informazioni e le trasforma in rappresentazione dei risultati!



Dati

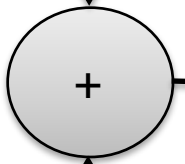
Le rappresentazioni dei valori (dati) devono essere scelte in un formato conveniente per l'elaborazione/elaboratore.

Es.: l'uomo moderno è più pratico nel sommare sommare numeri arabi che romani, i computer (elettronici e digitali) sono più efficienti con rappresentazioni in binario.

Informazione: Ho cento Euro nel portafoglio.
Tipo: Valùta EUR
Attributo: "Soldi nel portafoglio"
Valore:
- Cento (rappresentazione in italiano).
- 100 (rappresentazione in num. arabi)
- C (rappresentazione in num. romani)
- 1100100 (rappresentazione in binario)

Informazione: Ho mille Euro in banca.
Tipo: Valùta EUR
Attributo: "Soldi in banca"
Valore:
- Mille (rappresentazione in italiano).
- 1000 (rappresentazione in num. arabi)
- M (rappresentazione in num. romani)
- 1111101000 (rappresentazione in binario)

Elaborazione dei dati
Y=F(X)
X dati in input
Y dati in output



1100

Il dato in output è una rappresentazione di un valore. Interpretato (o decodificato), diventa informazione.

Decodifica

Informazione: Ho millecento Euro.
Tipo: Valùta EUR
Attributo: "Saldo"
Valore:
- Millecento
- 1100
- MC
- 10001001100

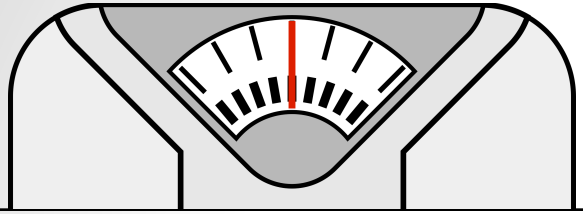
Rappresentazione Analogica e Discreta

- **La rappresentazione del valore** di una informazione (il dato) può essere **analogica** o **discreta (o numerica, o digitale: "a cifre")**.
- In una 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.
    - La codifica del valore è intensionale (implicita) data dal legame con la variazione della grandezza analoga.
- Una **rappresentazione discreta** usa **un insieme finito di rappresentazioni distinte che vengono messe in relazione con alcuni elementi dell'universo rappresentato**.
  - La codifica è estensionale (esplicita) definita tramite un alfabeto e regole di codifica.
  - Se il tipo dell'informazione non è finito, una rappresentazione digitale introdurrà necessariamente un errore di rappresentazione dei valori.

Esempio: la bilancia

Informazione: "Ieri il mio peso era di 82.340 Kg"; **Tipo**: $\mathbb{R}^+$; **Attributo**: "Peso di ieri";

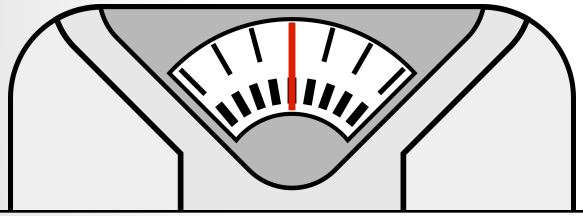
Valore rappresentato analogicamente:



Valore digitale rappresentato su 3 cifre:

Informazione: "Oggi il mio peso è 82.310 Kg"; **Tipo:** $\mathbb{R}^+$**; Attributo:** "Peso di oggi";

Valore rappresentato analogicamente:



Valore digitale rappresentato su 3 cifre:

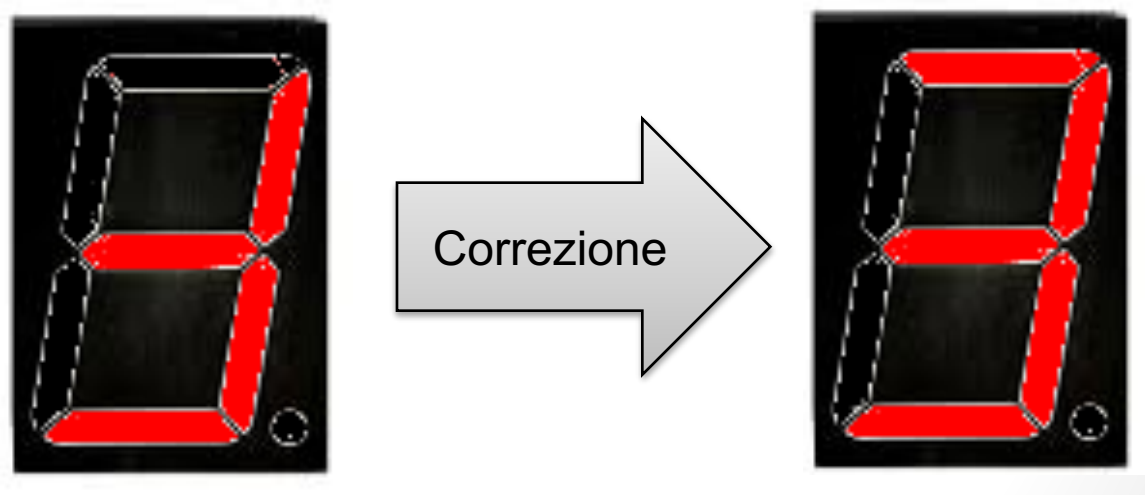
Informazione derivata da elaborazione: "Oggi sono dimagrito ?? Kg".

Come la rappresentazione del valore ha effetti sull'elaborazione dell'informazione?

Rappresentazioni digitali di informazioni di tipo non finito introducono errori che devono essere opportunamente gestiti nelle operazioni di calcolo!
Es.: sono dimagrito 0.03 Kg!

Perché una rappresentazione digitale?

- La **rappresentazione digitale** permette di conservare e trasmettere le informazioni **in maniera più robusta** di una rappresentazione analogica, perché i sistemi digitali possono facilmente **correggere interferenze** che modificherebbero irreversibilmente una rappresentazione analogica.



- I calcolatori digitali non operano su informazioni continue! Affinché un elaboratore possa usare queste informazioni è necessario prima trasformarle in discrete per permettere l'uso di rappresentazioni digitali.

Codifica

- La **codifica** è la tecnica adoperata per trasformare un'informazione in una sua rappresentazione (dato), la **decodifica** è l'operazione inversa.
- La **codifica** è definita da:
  - Un **alfabeto origine** T = (x_1, …, x_n) dei dati (valori) da codificare.
  - Un **alfabeto in codice** E = (a_1, …, a_k) che identifica un insieme di simboli.
  - **Un insieme** P **di parole-codice, ovvero di simboli o sequenze di due o più simboli (stringhe) dell'alfabeto in codice.**
  - **Una funzione chiamata tabella-codice che associa a ciascun elemento dell'alfabeto origine una parola codice:** T→ P
- Con il termine codice si intende qualche volta una parola-codice, qualche volta le relazioni di tutte le parole codice (ovvero la funzione tabella-codice).

Codifica: esempio

Codifichiamo i numeri interi tra 1 e 10 in numeri romani.

- **Alfabeto origine:** T = (1, 2, 3, 4, …, 10).
- **Alfabeto in codice:** E = (I, V, X, L, D, C, M)
- **Parole-codice:** P = (I, II, III, IV, V, VI, … IX, X)**.**
- **Tabella-codice:** rappresentiamo la funzione utilizzando una tabella.

	Parole-Codice (P)									
Alfabeto origine (T)		I	II	III	IV	V	VI	VII	VIII	IX	X
	1	•									
	2		•								
	3			•							
	4				•						
	5					•					
	6						•				
	7							•			
	8								•		
	9									•	
	10										•

Parole Codice (1/2)

- Con **cardinalità del codice** (*n*) si intende la **cardinalità dell'alfabeto in codice**, ovvero il numero di simboli distinti che possono essere adoperati per creare parole-codice.
- Un codice è detto a **lunghezza fissa** se le sue parole-codice hanno tutte la stessa lunghezza (*l*), altrimenti è detto a **lunghezza variabile**.
- **Per ogni coppia** (*n, l*) **è possibile generare al più un numero di parole-codice pari a** n^l, ottenute dall'applicare *l* volte il prodotto cartesiano (×) sui simboli dell'alfabeto in codice.
  - Es.: Alfabeto in codice: E=(a, b, c), n = |E| = 3
    - Con *l* = 1 posso al più avere tante parole-codice quanti sono i simboli dell'alfabeto in codice, ovvero *n*.
    - Con *l* = 2 posso al più generare tante parole codice quante sono le combinazioni con ripetizione degli *n* simboli nelle due posizioni, ovvero n^2: E × E = { (a, a), (a, b), (a, c), (b, a), (b, b), …, (c, c) }.
    - Con *l* = 3 al più ho n^3 parole codice (prod. cartesiano A×A×A).

Parole Codice (2/2)

- Dall'analisi, risulta che se l'alfabeto origine ha *m* elementi, per definire una tabella-codice necessariamente: $n^l \geq m$ (affinché le colonne della tabella-codice siano almeno quanto le righe, cioé l'alfabeto origine).
- Ne risulta che:

Per codificare un dato di cardinalità m mediante un alfabeto di n simboli è necessaria una stringa di lunghezza minima l_{min}, con:

$$l_{min} = \lceil \log_n m \rceil$$

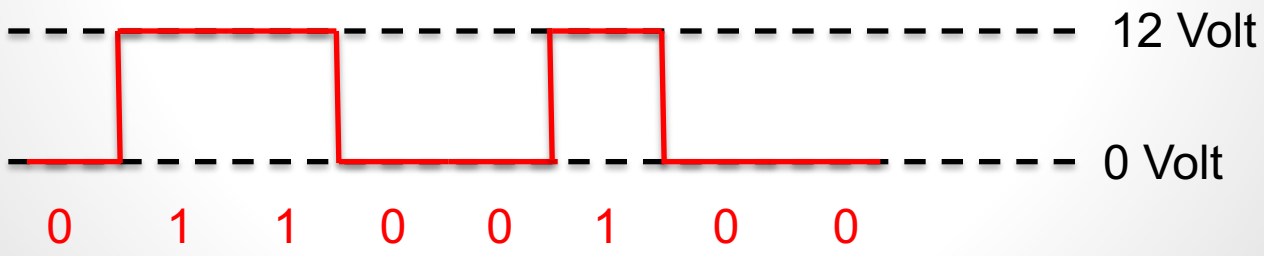
- Se $n^l > m$ è possibile generare parole-codice che non sono messe in relazione con l'alfabeto origine (più colonne che righe).

Codice binario 1/2

- Per le attuali architetture dei sistemi per l'elaborazione dell'informazione, il codice binario è di gran interesse.
- Il **codice binario** è una qualunque codifica che utilizza un alfabeto in codice composto da solamente due distinti simboli (rappresentati, ad esempio, come (0, 1), (OFF, ON), (FALSE, TRUE), (SPENTO, ACCESO), (A, B), (▼, ▲), etc.).
  - I due simboli rappresentano le unità minime di **rappresentazione** e **memorizzazione** digitale e sono identificate col nome bit (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: **digitale** assume il significato di informazione **codificata** in contrapposizione con **analogico** che invece descrive la realtà nelle sue infinite forme e varietà.

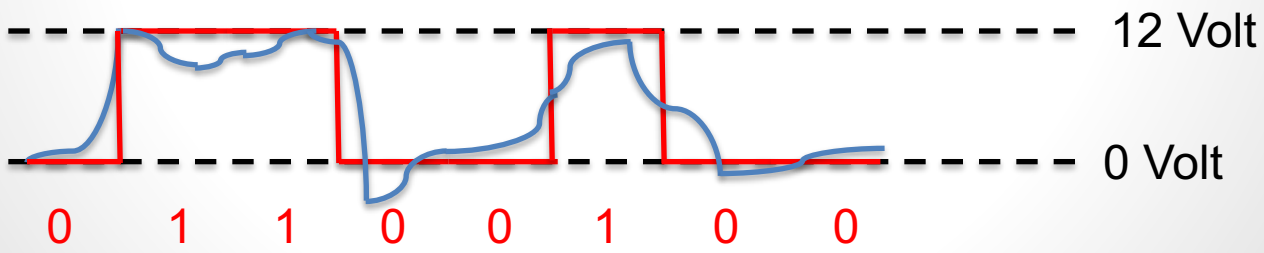
Codice Binario 2/2

- Le parole-codice di un codice binario sono rappresentate da stringhe composte esclusivamente dai due simboli.
  - Es.: 0011, 1100, 1010, 1111.
- La rappresentazione digitale semplifica la memorizzazione delle informazioni: come anticipato, essa rende i sistemi digitali meno soggetti ai disturbi elettrici rispetto ai sistemi analogici.
- I moderni calcolatori utilizzano segnali elettrici per rappresentare le cifre binarie.



Codice Binario 2/2

- Le parole-codice di un codice binario sono rappresentate da stringhe composte esclusivamente dai due simboli.
  - Es.: 0011, 1100, 1010, 1111.
- La rappresentazione digitale semplifica la memorizzazione delle informazioni: come anticipato, essa rende i sistemi digitali meno soggetti ai disturbi elettrici rispetto ai sistemi analogici.
- I moderni calcolatori utilizzano segnali elettrici per rappresentare le cifre binarie.



Esercizio sui codici binari (1/3)

- Creare un codice binario a lunghezza fissa *l* che permetta di rappresentazione la seguente informazione, scegliendo il minimo valore possibile per *l*:
- **Informazione:** "La prima nota della canzone è Sol".
  (Si trascurino le alterazioni e le durate delle note).
  - **Tipo**: note musicali, senza dettagli su alterazioni e durata.
  - **Valore:** "Sol".
  - **Attributo:** "Prima nota della canzone".

Esercizio sui codici binari (2/3)

Alfabeto origine: T = (Do, Re, Mi, Fa, Sol, La, Si),

Alfabeto in codice: E = (0, 1)

Lunghezza minima del codice: $l_{min} = \lceil \log_n m \rceil = \lceil \log_2 7 \rceil = \lceil 2.8\ldots \rceil = 3$

Parole-codice generabili per (n=2, l=3)**:** P = (000, 001, 010, 011, 100, 101, 110, 111); | P | = n^l = 8

Una possibile tabella-codice, rappresentata tabularmente:

	Parole-Codice (P)							
Alfabeto origine (T)		000	001	010	011	100	101	110	111
	Do	•							
	Re		•						
	Mi			•					
	Fa				•				
	Sol					•			
	La						•		
	Si							•	

Siccome | P | > | T | esistono parole-codice generabili che non sono messe in relazione con alcun simbolo dell'alfabeto origine, ovvero non rappresentano alcun valore!

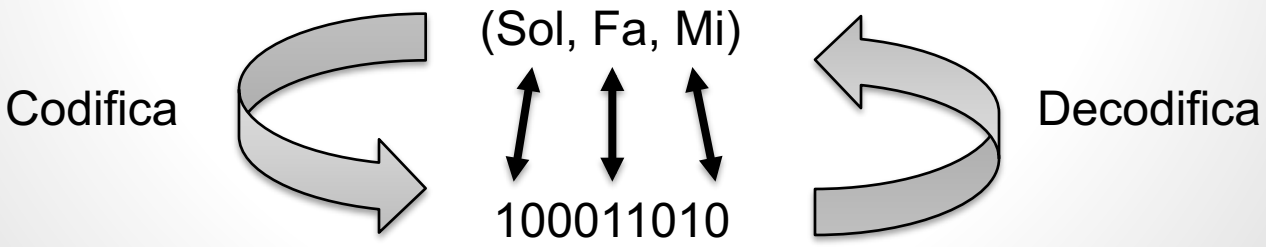
Esercizio sui codici binari (3/3)

Definito un codice, è possibile applicarlo in sequenza per rappresentare stringhe di valori dello stesso tipo!

Informazione: Le prime tre note della canzone sono “Sol, Fa, Mi”.

Tipo: Una sequenza di tre note musicali (trascurando le alterazioni e le durate).
Valore: (Sol, Fa, Mi).
Attributo: “prime tre note della canzone”.

Essendo il valore una stringa di valori elementari, possiamo applicare in sequenza la codifica che abbiamo precedentemente definito:



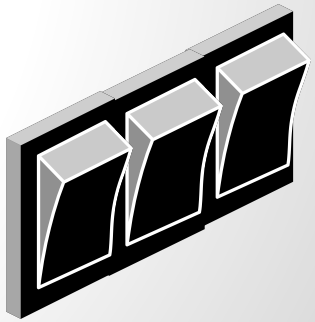
Convenendo un codice a lunghezza fissa sappiamo immediatamente separare le singole parole-chiave che compongono il nostro messaggio: (100, 011, 010).

Rappresentazione in macchina dei dati

- In un calcolatore i dati sono memorizzati in apparati chiamati **registri.**
- Un registro può assumere ***k*-stati** (ovvero, configurazioni) che sono messi in relazione a ***k* valori** che un dato può assumere.
- Componendo più registri, si compongono i loro stati. Ciò permette di memorizzare dati con cardinalità maggiore.
- La tecnologia elettronica che caratterizza i moderni calcolatori è capace di fabbricare **registri elementari bistabili** (chiamati **flip-flop**), che memorizzano informazioni a due soli valori (rappresentati da 0 e 1), i **bit (binary digit)**.

Registro a due stati
{ON, OFF}

Registro a quattro stati
{{ON, ON}, {ON, OFF},
{OFF, ON}, {OFF, OFF}}



Registro a otto stati
{{ON, ON, ON}, … }

La codifica binaria e i registri

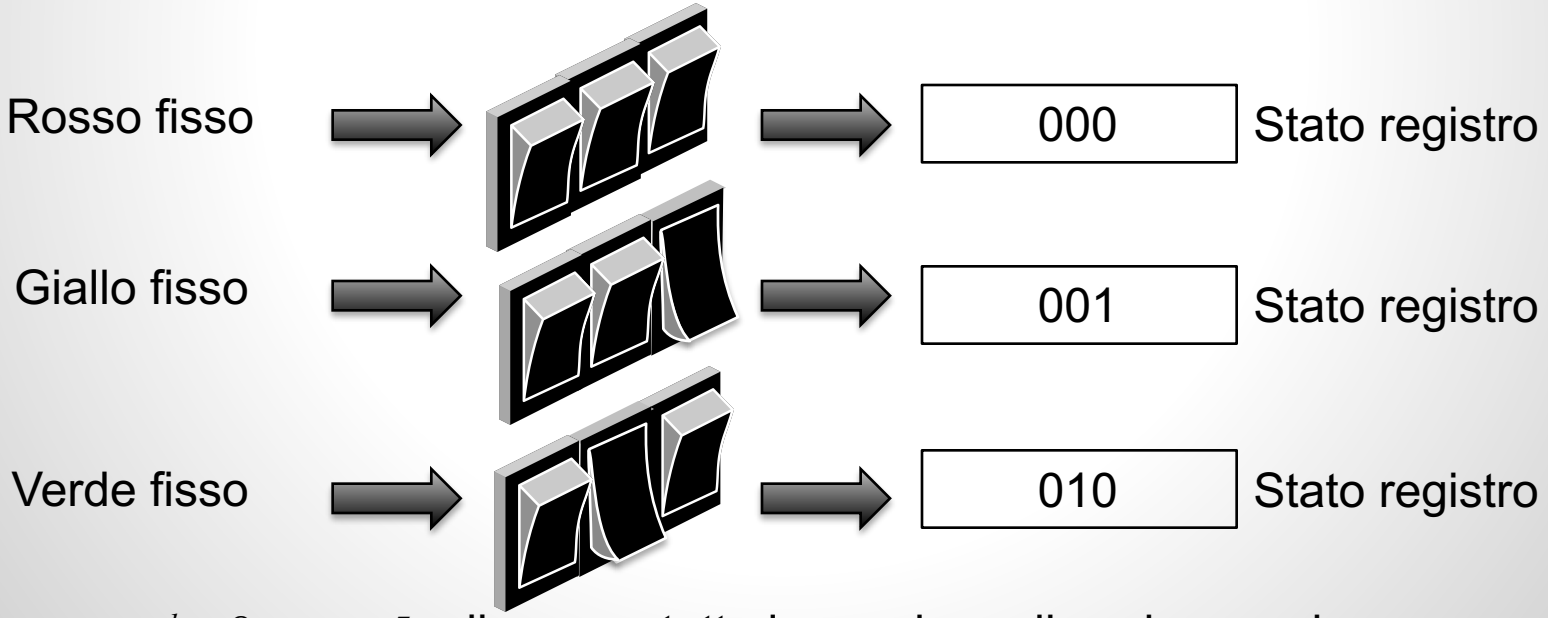
- Dal momento che il registro elementare dei calcolatori attuali è composto da elementi **bistabili**, i calcolatori necessariamente impiegano una codifica binaria per rappresentare le informazioni.
- I due simboli di cui sono composti i codici binari sono infatti messi in relazione con la coppia di stati che possono assumere i flip-flop.

Usando una codifica binaria siamo capaci di rappresentare qualsiasi informazione in un calcolatore, associando le stringhe binarie allo stato interno dei registri elementari!

Esempio codifica binaria: il tipo Semaforo

- **Tipo Semaforo:** {“Rosso fisso”, “Giallo fisso”, “Verde fisso”, “Giallo lampeggiante”, “Spento”}.
- Considerando un alfabeto in codice di due simboli (che possiamo rappresentare come {0, 1} o {  }), per codificare i valori di un semaforo abbiamo bisogno di un codice di lunghezza *l*:

$$l > l_{min} = \lceil \log_n m \rceil = \lceil \log_2 5 \rceil = \lceil \log_2 5 \rceil = \lceil 2.321\ldots \rceil = 3$$



- Siccome $n^l = 8 > m = 5$, allora non tutte le parole-codice che possiamo generare sono messe in relazione con l’alfabeto origine.

Misura dell'informazione

- Grazie alla codifica binaria, ogni flip-flop può rappresentare una informazione che varia tra due valori (appunto, 0 o 1): il **bit** è pertanto definibile come **un tipo di dato di cardinalità 2**.
- Dal momento che è possibile comporre i flip-flop in registri, possiamo anche definire il bit come **una unità di misura dell'informazione.**
  - **Maggiore è la quantità di informazione** (ovvero il numero di scelte tra le quali una informazione può assumere valore - cioé la cardinalità del tipo), **maggiore sarà il numero di bit che dovrò usare per codificare l'informazione stessa**.
  - Es.: Poiché l'informazione di tipo semaforo è codificabile al minimo con 3 bit, la sua quantità d'informazione è 3 bit perché in un calcolatore devo usare almeno 3 flip-flop per rappresentarla.

Byte, Word e Memorie

- Per ragioni legate alla costruzione dei moderni calcolatori, è d'uso fare riferimento a (stringhe di) 8 bit con il termine **byte**.
- I registri sono, in generale, multipli di una **word**, che dipende dalle caratteristiche del sistema, ma è un multiplo del byte: 8, 16, 32, 64 bit
- Organizzazioni di registri prendono il nome di **memorie**.

Misuriamo la quantità di informazione memorizzabile in un dispositivo di memoria usando come unità di misura il bit, i byte o i loro multipli.

Sigla	Nome	Numero di Byte	Numero di bit
B	Byte	1	8
KB	KiloByte	2^{10}=1024 (~10^3)	8192
MB	MegaByte	2^{20} = 1.048.576 (~10^6)	8.388.608
GB	GigaByte	2^{30}=1.073.741.824 (~10^9)	8.589.934.592
TB	TeraByte	2^{40}=1.099.511.627.776 (~10^{12})	8.796.093.022.208

Come funziona nei moderni calcolatori

- I processori dei calcolatori moderni lavorano con unità di dati maggiori o uguali ad un byte (8 bit).
  - Ad esempio, nonostante possa codificare l'informazione semaforo con 3 bit, essa sarà rappresentata in un calcolatore con 1 byte (8 bit).
- Per elaborare informazioni di dimensione maggiore si impiegano multipli del byte (o delle word).
- In maniera più dettagliata, i moderni calcolatori usano una codifica binaria a lunghezza fissa che adotta parole codice con lunghezza a multipli di 8.

Numero di byte	Numero di Bit	Configurazioni	
1	8	2^8	256
2	16	2^{16}=1024	65.536
3	24	2^{24} = 1.048.576	16.777.216
4	32	2^{32}=1.073.741.824	4.294.967.296

Codifiche in uso negli elaboratori

- Per elaborare informazioni numeriche, i calcolatori adottano codifiche binarie generalmente a lunghezza fissa, tra cui le seguenti:
  - Codifica di numeri interi $\mathbb{Z}$ in segno e modulo
  - Codifica di numeri interi $\mathbb{Z}$ in complementi alla base
  - Codifica di numeri reali $\mathbb{R}$ secondo lo standard IEEE 754
- Per elaborare informazioni testuali sono comuni le seguenti codifiche:
  - Codifica di caratteri secondo lo standard ASCII (7-8 bit)
  - Codifica di caratteri secondo lo standard Unicode (16 bit)
- Esistono poi numerose codifiche per elaborare immagini, video e suoni.

Errori di rappresentazione

- Adottando codifiche numeriche finite, i calcolatori possono rappresentare solamente sottoinsiemi degli insiemi numerici:

$$\mathbb{Z}^* \subseteq \mathbb{Z}; \mathbb{R}^* \subseteq \mathbb{R}$$

- L'errore di **round-off** è l'errore che si commette nell'approssimare (arrotondare) un elemento di un insieme numerico (es.: $r \in \mathbb{R}$) al valore più vicino rappresentabile (es.: $r^* \in \mathbb{R}^*$).
  - Gli algoritmi numerici devono opportunamente stimare e ridurre gli errori di round-off per garantire risultati accurati.
- Inoltre, in un insieme numerico finito, una operazione di calcolo può generare:
  - Una condizione di **overflow,** quando il risultato dell'operazione supera i limiti dell'intervallo rappresentabile (minore del minimo, o maggiore del massimo).
  - Una condizione di **underflow,** quando il risultato dell'operazione, a causa dell'approssimazione, viene rappresentato con lo zero pur non essendo nullo.

Esempio: Errori di rappresentazione

- Consideriamo una codifica che permette di rappresentare solamente i seguenti elementi di $\mathbb{R}$:

$$\mathbb{R}^* = \{-2, -1.5, -1, -0.5, 0, 0.5, 1, 1.5, 2\}.$$

- L'errore di **round-off** è quello commesso quando approssimo un numero al fine di trasformarlo in uno dei valori rappresentabili:
  - $1.55 \rightarrow 1.5$; Errore round-off $= | r - r^* | = 0.05$
  - $-0.666\ldots \rightarrow -0.5$; Errore round-off $= | r - r^* | = | -2/3 + 1/2 | = 1/6$
  - $1.5 \rightarrow 1.5$; Errore round-off $= | r - r^* | = 0$
- L'**overflow** si verifica quando il risultato di una operazione supera i limiti dell'intervallo rappresentabile:
  - $1.5 + 1.5 \rightarrow$ **overflow**
  - $-2 + -0.5 \rightarrow$ **overflow**
- L'**underflow** si verifica quando il risultato di una operazione rappresenta con lo zero un numero diverso dallo zero:
  - $-0.5 / 2 \rightarrow 0$ **underflow**

La matematica per gli insiemi finiti

- Dal momento che i calcoli avvengono su insiemi finiti,
  l'informatica fa ampio uso di due branche della matematica:
  - La **matematica discreta**, che studia a "livello teorico" strutture matematiche discrete, piuttosto che continue.
  - Il **calcolo numerico**, che studia a "livello applicativo" come approssimare modelli matematici continui per risolverli con insiemi numerici finiti.

Esempio

- Prendiamo in considerazione il caso di una 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
739+510	Overflow
-500-720	Overflow (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 non sempre vengono rispettate in base all'ordine con cui le operazioni vengono eseguite
- Proprietà:
  - proprietà associativa: ***a + (b - c) = (a + b) – c***
  - proprietà distributiva: ***a × (b - c) = a × b – a × c***

a	b	c	a+(b-c)	condizione	(a+b)-c	condizione
100	900	600	100+(900-600)	ok	(100+900)-600	Overflow

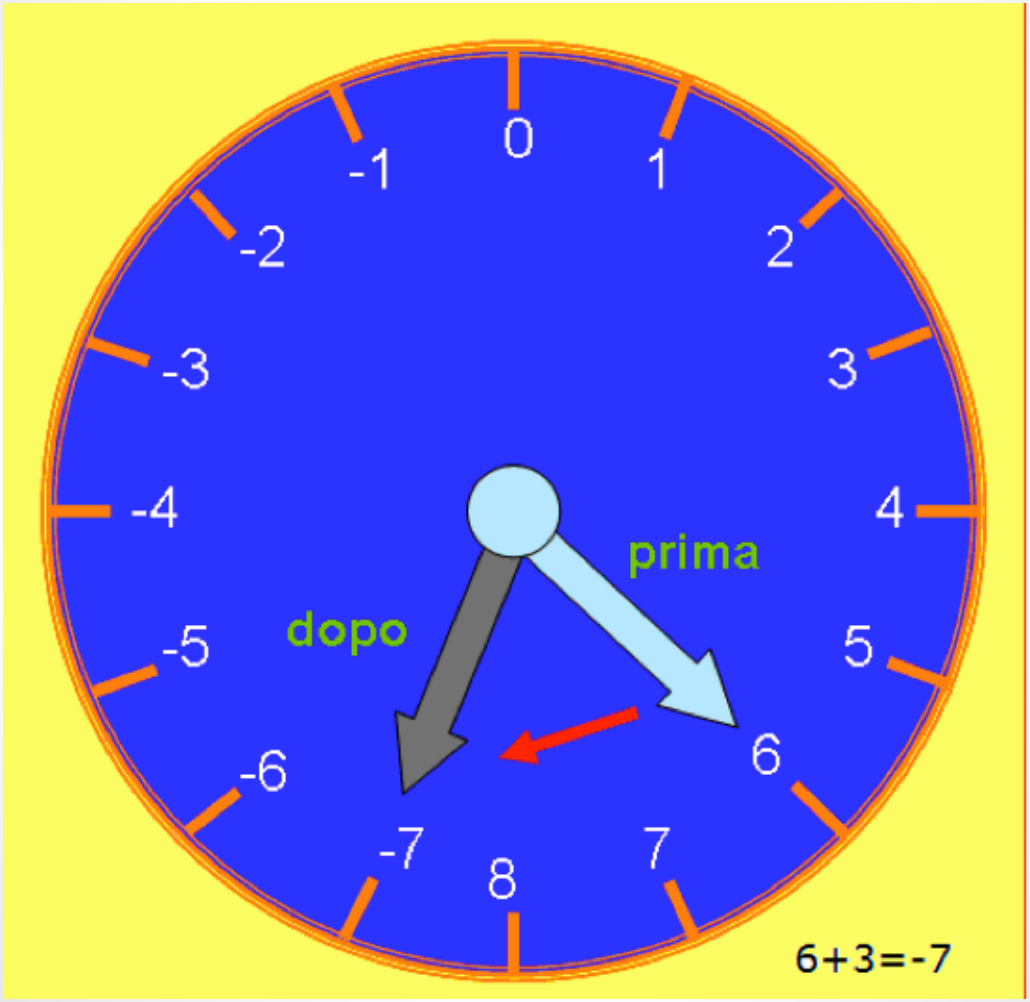
a	b	c	a×(b-c)	condizione	a×b-a×c	condizione
200	90	88	200×(90-88)	ok	200×90-200×88	Overflow

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, 359]	360	1200	1200:360	120
[0, 59]	60	61	61:60	1
[0, 59]	60	55	60:55	55
[0, 59]	60	60	60:60	0

Calcolatrice per Sistemi di Numerazione finiti



Sistemi di Numerazione

- Un sistema di numerazione può essere visto come un insieme di simboli (o 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

- Data una base **b** ed un insieme ordinato di cifre c_i, c_{i-1}, …, c_1, c_0, c_{-1}, …, il valore assunto in numerazione posizionale pesata è esprimibile come segue:

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

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

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

- 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

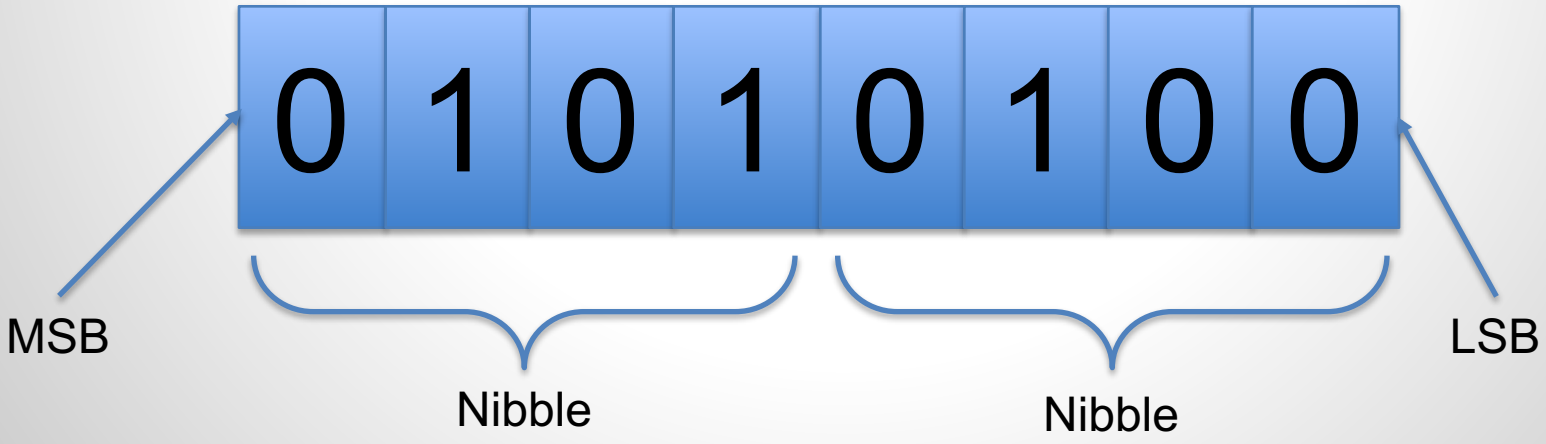
Sistema Binario

- 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.

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

Nibble, 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*)
- Il bit più a sinistra è quello più significativo
  - (posizione o peso 7, detto anche ***MSB*** da *Most Significant Bit*)
- Il nibble è la metà di un byte (ovvero 4 bit)



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

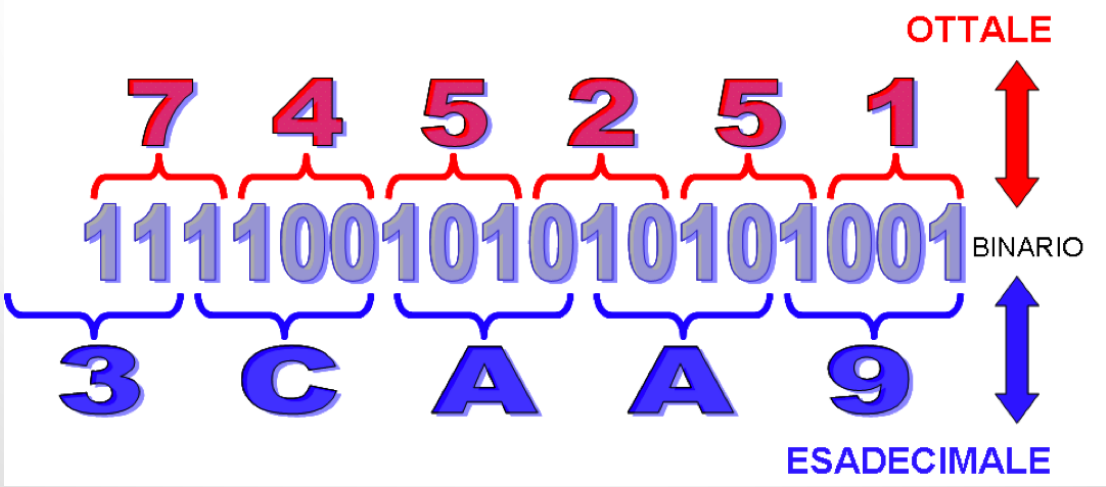
Esadecimale	Binario
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
A	1010
B	1011
C	1100
D	1101
E	1110
F	1111

Conversione Tra Sistemi di Numerazione

- Il cambio di base per una data rappresentazione numerica può effettuarsi:
  A. Mediante una procedura che prevede l'uso di divisioni e moltiplicazioni.
  B. Previo calcolo di potenze della base, come nella numerazione posizionale, adottando come base quella destinazione.
- Il metodo A è conveniente per convertire numeri decimali in qualunque altra base, mentre B per trasformare qualunque rappresentazione in base decimale.
- Fanno eccezione le basi 8 e 16 poiché, essendo multipli di 2, godono di proprietà più potenti delle altre basi ed esistono regole semplici per trasformare un numerp da una rappresentazione ad un'altra.

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:

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

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

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

- se si divide la parte intera per 2:

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

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

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

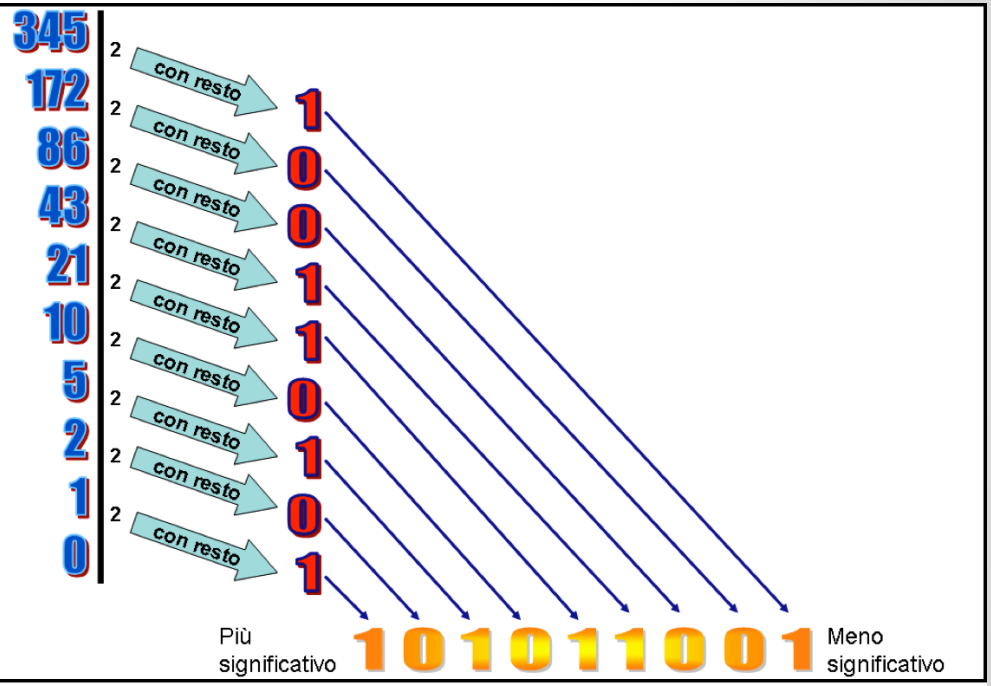
- …. il procedimento deve essere ripetuto fino a quando si ottiene un quoziente uguale a 0.

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.