UNIVERSITÀ DEGLI STUDI DELLA CAMPANIA
LUIGI VANVITELLI
SCUOLA POLITECNICA E DELLE SCIENZE DI BASE
DIPARTIMENTO DI INGEGNERIA CIVILE
DESIGN EDILIZIA E AMBIENTE

# Fondamenti di Informatica

Ing. Alba Amato, PhD

alba.amato@unina2.it

UNIVERSITÀ DEGLI STUDI DELLA CAMPANIA
LUIGI VANVITELLI
SCUOLA POLITECNICA E DELLE SCIENZE DI BASE
DIPARTIMENTO DI INGEGNERIA CIVILE
DESIGN EDILIZIA E AMBIENTE

Ing. Alba Amato, PhD
alba.amato@unina2.it

6 Crediti didattici(CFU)
**Ricevimento: Martedì 10:00 – 11:00**
**AULA DOCENTI**

Modalità d’esame:

Prova Scritta

Prova Orale

Solo gli studenti che hanno raggiunto la sufficienza nello scritto accedono all’orale

UNIVERSITÀ DEGLI STUDI DELLA CAMPANIA
LUIGI VANVITELLI
SCUOLA POLITECNICA E DELLE SCIENZE DI BASE
DIPARTIMENTO DI INGEGNERIA CIVILE
DESIGN EDILIZIA E AMBIENTE

- PROGRAMMA
  - L'INFORMAZIONE E LE SUE RAPPRESENTAZIONI
  - IL MODELLO ESECUTORE
  - ALGORITMI E PROGRAMMI
  - LA STRUTTURA DEI PROGRAMMI
  - I DATI
  - IL LINGUAGGIO C
  - IL LINGUAGGIO DELL'AMBIENTE MATLAB
  - LA TRADUZIONE DEI PROGRAMMI
  - INTRODUZIONE AI SISTEMI OPERATIVI
  - LE RETI DI COMUNICAZIONE
  - IL MONDO INTERNET

UNIVERSITÀ DEGLI STUDI DELLA CAMPANIA
LUIGI VANVITELLI
SCUOLA POLITECNICA E DELLE SCIENZE DI BASE
DIPARTIMENTO DI INGEGNERIA CIVILE
DESIGN EDILIZIA E AMBIENTE

# L'INFORMAZIONE E LE SUE RAPPRESENTAZIONI

*Lucidi tratti da:*

*Alla scoperta dei fondamenti dell'informatica. Un viaggio nel mondo dei bit*
di Angelo Chianese, Vincenzo Moscato, Antonio Picariello

capitolo 1 -par. 1.1, 1.2, 1.3

***Informatica:*** la *scienza della risoluzione dei problemi con l'aiuto degli elaboratori*

## *INFORMAZIONE E CODIFICA*

- perché persone o macchine possano utilizzare un'informazione hanno bisogno che essa sia appropriatamente "*rappresentata*"
- La *codifica* è l'insieme di convenzioni e di regole da adottare per trasformare un'informazione in una sua rappresentazione e viceversa.

# *Modalità di Rappresentazione*

- **Rappresentazione analogica**: le proprietà del fenomeno rappresentato sono omomorfe alla forma della rappresentazione (la rappresentazione varia in analogia con la grandezza reale, es. termometro).
- Rappresentazione ANALOGICA: una serie di distinzioni CONTINUE
  - Immagini, suoni, numeri reali
- **Rappresentazione discreta**: si utilizza un insieme finito di rappresentazioni distinte che vengono messe in relazione con alcuni elementi dell'universo rappresentato.
- Rappresentazione DISCRETA: ogni elemento chiaramente distinto dagli altri

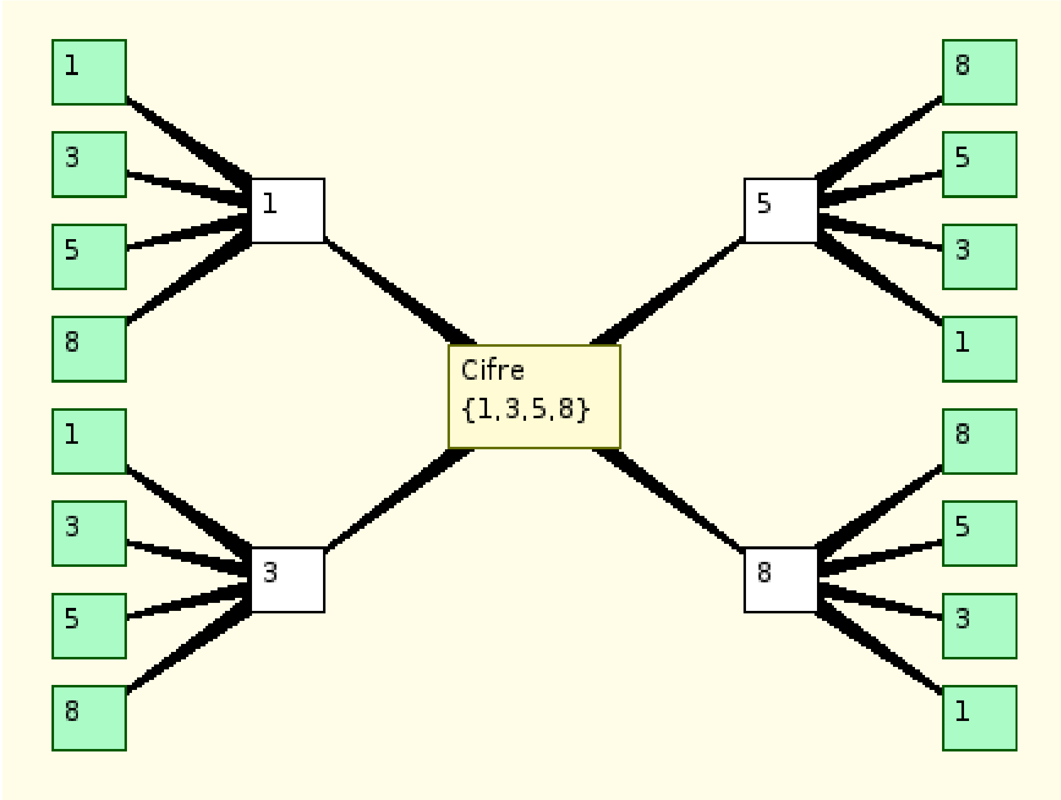
# *Codifica e Codice*

- 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
- 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.

# *Codifica e Codice*

- Siano allora **V = {$v_1$, $v_2$, ..., $v_m$ }** l'insieme degli ***m*** 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 $n^l$.

- Consideriamo l'insieme: A = {1,3,5,8}; vogliamo determinare quanti numeri a due cifre si possono scrivere con gli elementi di A, considerando che sono ammesse le ripetizioni.
- Le possibili combinazioni sono 16: 1,13,15,18,31,33,35,38,51,53,55, 58,81,83,85,88
- e si indicano con: $D'_{4,2} = 16 = 4^2$



UNIVERSITÀ DEGLI STUDI DELLA CAMPANIA
LUIGI VANVITELLI
SCUOLA POLITECNICA E DELLE SCIENZE DI BASE
DIPARTIMENTO DI INGEGNERIA CIVILE
DESIGN EDILIZIA E AMBIENTE

*L=lunghezza della parola*

# *Codifica e Codice*

- Ad esempio, fissato l'alfabeto **A = {-,.}** del codice Morse con ***n***=2 (2 simboli distinti), si hanno le diverse rappresentazioni riportate in tabella.

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

*numero di parole codice differenti è uguale a* $n^l$

# *Codifica e Codice*

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

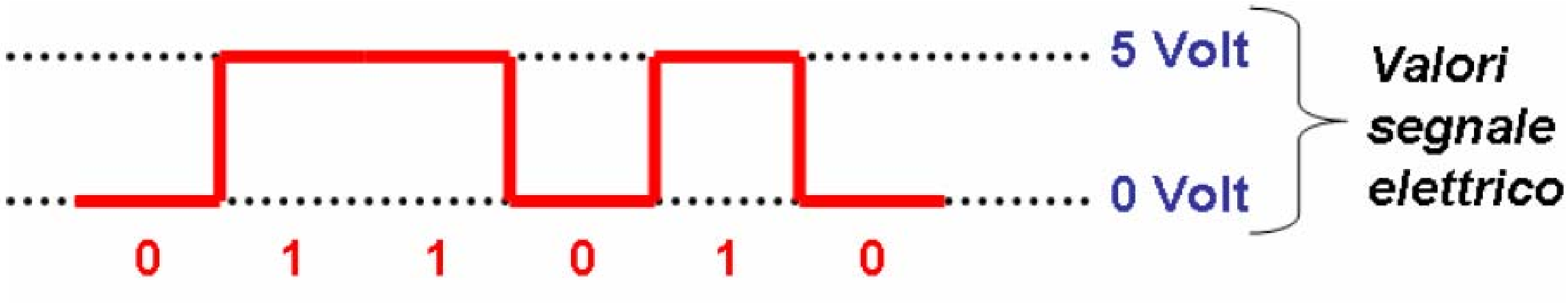
*numero di parole codice differenti* $n^l \geq m$ *l'insieme degli m valori diversi di una data informazione*

- Se $n^l > m$ allora la codifica è ridondante

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

# *Rappresentazione Binaria Digitale*

- alfabeto costituito da due soli simboli distinti: “0” e “1”.
- fenomeni diversi che possono essere facilmente associati ad un **bit:**
  - la presenza o assenza di tensione elettrica in un circuito elettrico;
  - le polarità positiva e negativa di un magnete
  - l’apertura o chiusura di una conduttura;
  - la condizione di acceso o di spento di un interruttore.




Esempio di rappresentazione digitale di un segnale di tensione

# *Rappresentazione Binaria Digitale*

- La rappresentazione digitale è più affidabile (probabilità di errore bassa), in quanto disturbi provenienti dall'ambiente o interferenze (rumore) indotte da altri componenti difficilmente possono far variare lo stato di un componente che memorizza i bit.
- Adottando due soli stati si può scegliere una separazione massima tra le corrispondenti grandezze indicative dello zero e dell'uno, per cui il rumore pur sommandosi non produce significative alterazioni.

# *Rappresentazione Binaria Digitale*

- La **trasformazione delle grandezze analogiche nella loro rappresentazione digitale** ha tra i tanti ***vantaggi*** quello della ***fedeltà della riproduzione e quello della trasmissione dell'informazione a diversi tipi di dispositivi elettronici.***
- Alcune informazioni nascono già in formato digitale grazie a strumenti che operano direttamente con tale rappresentazione (calcolatore elettronico, i telefoni cellulari, fotocamere digitali, ecc
- ***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.***

# *I numeri in 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 $\boldsymbol{2^{l}}$ configurazioni (parole codice) diverse.
  - Viceversa se si devono rappresentare ***K*** informazioni diverse occorrono $\boldsymbol{\log_2 K}$ bit per associare ad esse codici diversi.
- La ***precisione*** con cui i numeri possono essere espressi è ***finita e predeterminata***, poiché questi devono essere memorizzati con parole codice di lunghezza fissata.

# *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 dono 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 calcolatori*

- Con otto bit si rappresentano solo $2^8$ (256) valori diversi.
- Nel caso in cui un solo byte non fosse sufficiente per rappresentare i K valori dell'informazione, allora si individua il numero ***b*** di byte tale che: $\mathbf{2^{(b*8)} \geq K}$
- 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 né troppo piccolo.

# *Esempi*

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

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

# *Algebra con precisione finita*

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

| *a* | *b* | *C* | $a + (b - c)$ | condizione | $(a + b) - c$ | Condizione |
|---|---|---|---|---|---|---|
| 100 | 900 | 600 | 100 + (900 - 600) | ok | (100 + 900) - 600 | Overflow |
| *a* | *b* | *C* | $a \times (b - c)$ | condizione | $a \times b - a \times c$ | Condizione |
| 200 | 90 | 88 | 200 × (90 – 88) | Ok | 200 × 90 – 200 × 88 | Overflow |

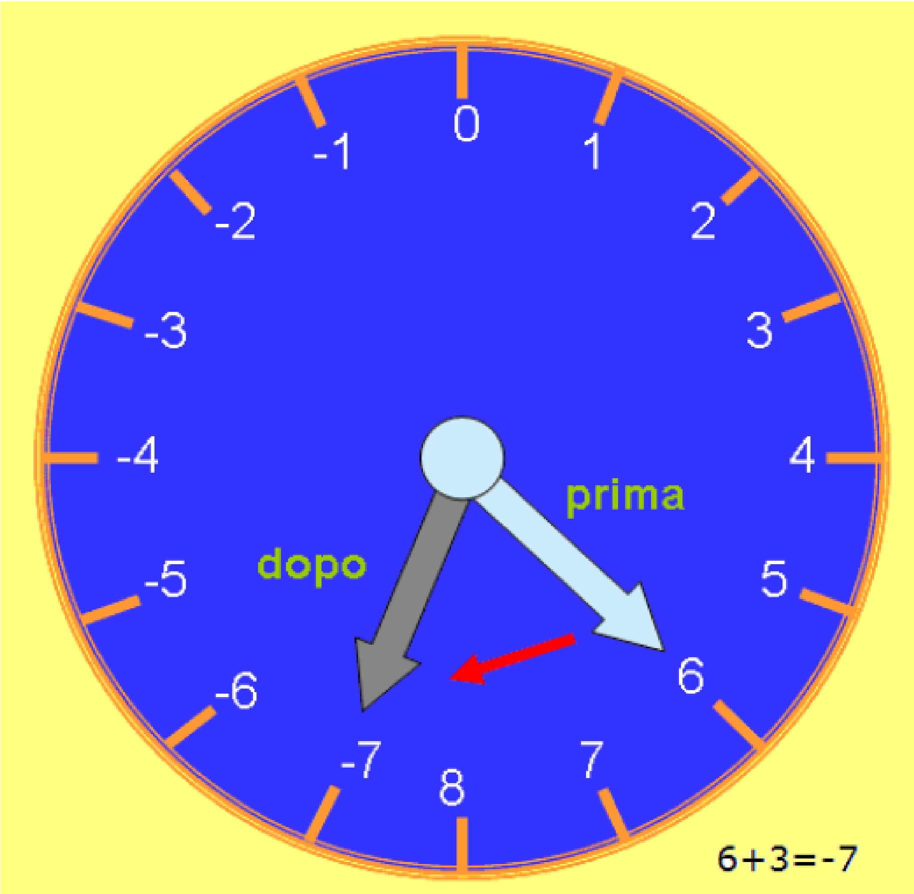
# *Sistemi Periodici*

- In caso di periodicità, i valori esterni all'intervallo di definizione vengono ricondotti ad esso prendendo il resto della divisione dei valori per il periodo
- Si parla di operazione di «modulo»

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

UNIVERSITÀ DEGLI STUDI DELLA CAMPANIA
LUIGI VANVITELLI
SCUOLA POLITECNICA E DELLE SCIENZE DI BASE
DIPARTIMENTO DI INGEGNERIA CIVILE
DESIGN EDILIZIA E AMBIENTE

# *Esempio*




- basta posizionare la lancetta sul valore del primo operando, spostarla di un numero di posizioni uguale al secondo operando in senso orario per la somma e in senso antiorario per la sottrazione, quindi leggere come risultato la sua posizione finale.
- 6+8=-2

# *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 MSB*

- Nel 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 finito di cifre (simboli)
  - Un insieme finito di regole che assegnano ad ogni sequenza di cifre uno ed un solo valore numerico.
    - **Non posizionali**: il valore di ciascuna cifra nella rappresentazione è indipendente dalla sua posizione (es. nel sistema romano il simbolo "L" esprime la quantità 50 indipendentemente dalla posizione)
    - **Posizionali:** ad ogni posizione della cifra all'interno della rappresentazione è associato un peso (es. nel sistema decimale la prima cifra a destra indica l'unità, la seconda le centinaia, etc)
- Verranno considerati solo i sistemi di codifica posizionali

# *Codifica di tipo Posizionale*

- una data stringa di bit può essere interpretata come una qualsiasi sequenza di cifre in un sistema di numerazione posizionale che associa alle cifre ***c*** un diverso peso in base alla posizione ***i*** occupata nella stringa che compone il numero, dove il peso dipende dalla base ***b*** di numerazione

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

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

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

# *Codifica di tipo Posizionale*

- Ad esempio nel sistema decimale (base b=10)

$$10.061.974 = 1x10^7 + 0x10^6 + 0x10^5 + 6x10^4 + 1x10^3 + 9x10^2 + 7x10^1 + 4x10^0$$

- 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 |

UNIVERSITÀ DEGLI STUDI DELLA CAMPANIA
LUIGI VANVITELLI
SCUOLA POLITECNICA E DELLE SCIENZE DI BASE
DIPARTIMENTO DI INGEGNERIA CIVILE
DESIGN EDILIZIA E AMBIENTE

# *Codifica di tipo Posizionale*

- Poiché basi diverse fanno uso delle stesse prime cifre, si rende necessario distinguere le rappresentazioni dei numeri con un pedice indicante la base dopo aver racchiuso la stringa tra parentesi.

$$(101111)_2 = (142)_5 = (47)_{10}$$

- Inoltre, poiché nel sistema decimale la prima cifra a destra indica le unità, la seconda indica le decine, la terza le centinaia, la quarta le migliaia, e così di seguito secondo le potenze del dieci, solo in esso è possibile leggere i numeri come *tremilacentouno*, *unmilioneetrenta*.
- Negli altri sistemi di numerazione devono essere scandite le cifre, da quella di peso maggiore fino a quella di minor peso, con indicazione della base (ad esempio “*unoquattrodue*” in base cinque).

# *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. Esempio:

$(101111)_2 = 1 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 = 32 + 8 + 4 + 2 + 1$
$=$

$(142)_5 = 1 \times 5^2 + 4 \times 5^1 + 2 \times 5^0 = 25 + 20 + 2 =$

$(47)_{10} = 4 \times 10^1 + 7 \times 10^0$

## *Proprietà delle Rappresentazioni*

- 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.
- Solitamente la stringa di cifre in bit è approssimativamente tre volte più lunga di quella decimale. Esempio:

$$(1001101)_2 = 1 \times 2^6 + 0 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 =$$

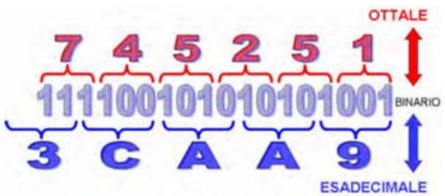
$$= 64 + 0 + 0 + 8 + 4 + 0 + 1 =$$

$$= (77)_{10}$$

# Sistema Ottale ed Esadecimale

- In informatica, per evitare di dover trattare con stringhe di bit troppo lunghe, sono stati introdotti il sistema ottale ed esadecimale.
- La tabella mostra la corrispondenza tra le cifre usate in tali rappresentazioni e i bit che le rappresentano.

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

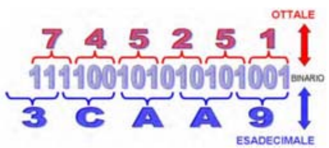



| Esadecimale | Binario |
|---|---|
| 0 | 0000 |
| 1 | 0001 |
| 2 | 0010 |
| 3 | 0011 |
| 4 | 0100 |
| 5 | 0101 |
| 6 | 0110 |
| 7 | 0111 |
| 8 | 1000 |
| 9 | 1001 |
| A | 1010 |
| B | 1011 |
| C | 1100 |
| D | 1101 |
| E | 1110 |
| F | 1111 |

# *Sistema Ottale ed Esadecimale*

- la trasformazione di rappresentazioni di valori tra esse si e la base 2 (e viceversa) è immediata.
  - 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.

| 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 |

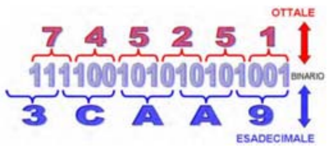
# *Sistema Ottale ed Esadecimale*

- Ad esempio: 1000 1010 1101 0101
  - 0101->5
  - 1101->D
  - 1010->A
  - 1000->8

- Diventa: 8AD5

| 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 Decimale-Binario*

Esempio: si converta in binario il numero 8

| | | |
|---|---|---|
| 8:2= 4 | resto 0 | Bit meno significativo (LSB) |
| 4:2= 2 | resto 0 | |
| 2:2= 1 | resto 0 | |
| 1:2= 0 | resto 1 | Bit più significativo (MSB) |

LSB

Il numero binario è: 1 0 0 0

MSB

## *Conversione Decimale-Binario*

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

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

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

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

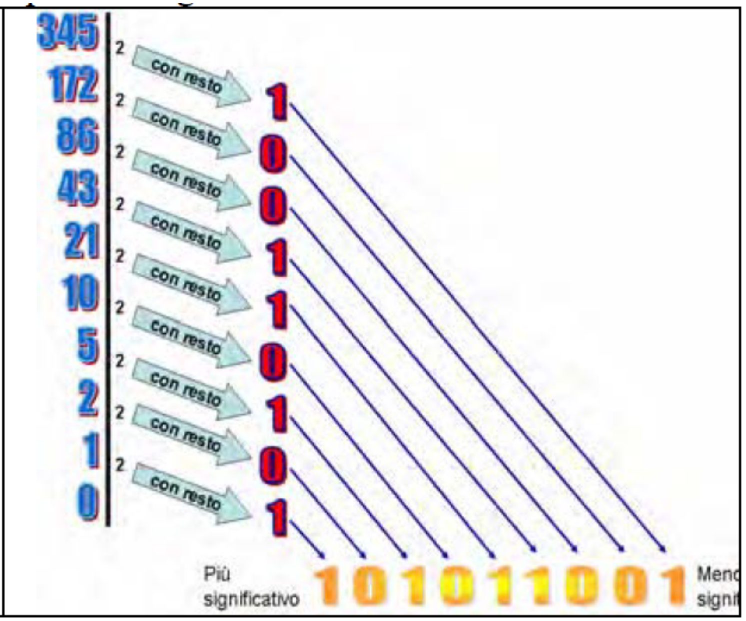
- …. il procedimento deve essere ripetuto fino a quando si ottiene un quoziente uguale a 0.

# *Conversione Decimale-Binario*

**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 Decimale-Binario*

Esempio: si converta in binario il numero 71

| | | |
|---|---|---|
| 71:2= 35 | resto 1 | Bit meno significativo (LSB) |
| 35:2= 17 | resto 1 | |
| 17:2= 8 | resto 1 | |
| 8:2= 4 | resto 0 | |
| 4:2= 2 | resto 0 | |
| 2:2 = 1 | resto 0 | |
| 1:2 = 0 | resto 1 | Bit più significativo (MSB) |

Il numero binario è: 1 0 0 0 1 1 1

## *Conversione Per Numeri Frazionari*

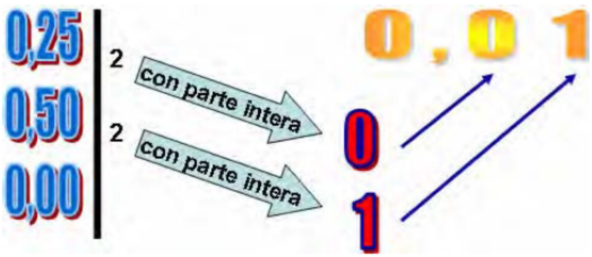
- Per la conversione della parte frazionaria si procede al contrario moltiplicando per 2:

$$\mathbf{d_{pf} \times 2 = c_{-1} \times b^{0} + c_{-2} \times b^{-1} + \ldots\ldots}$$
$$\mathbf{d_{pf1} = c_{-2} \times b^{-1} + \ldots\ldots}$$

- per spostare $\mathbf{c_{-1}}$ a sinistra della virgola che diventa parte intera.
- Anche in questo 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.
- Si noti che solo la prima condizione garantisce una conversione senza approssimazione.

**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.***

# *Conversione Per Numeri Frazionari*

Ad esempio, per convertire $0.6875_{10}$ in binario:

$$
\begin{array}{r|lcl}
 & \overset{*2}{} & & \\
0.6875 & 1.375 & & U=1 \\
0.375 & 0.75 & & U=0 \\
0.75 & 1.5 & & U=1 \\
0.5 & 1.0 & & U=1 \\
0 & & & \\
\end{array}
\Big\downarrow \text{ordine}
$$

da cui si ottiene che $0.6875_{10} = 0.1011_2$.

Se si volesse convertire il numero $57.6875_{10}$ sarebbe sufficente unire i due risultati (per la parte intera e per la parte frazionaria), ottenendo $111001.1011_2$.

## *Operazioni sui Binari*

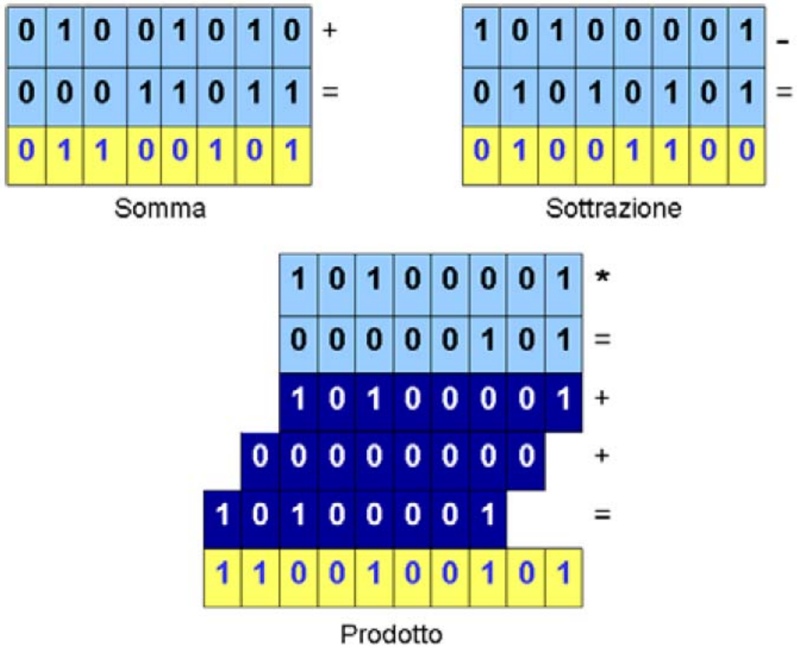
- Come nel decimale si definiscono la tavola dell'addizione e la tabellina del prodotto per le cifre binarie.

| 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

# *Operazioni sui Binari*

- si noti che nella somma si deve tener conto del riporto (che si propaga a sinistra così come nell'aritmetica decimale) mentre nella sottrazione, in presenza dell'operazione "0 – 1", si deve attivare il prestito dalle cifre più a sinistra (come nell'aritmetica decimale).



# *Esercizio*

- Convertire:

N = 57 da base 10 a base 2

# *Soluzione*

$$
\begin{array}{r|l}
\multicolumn{2}{c}{/2} \\
57 & 28 \quad \mathrm{R} = 1 \\
28 & 14 \quad \mathrm{R} = 0 \\
14 & 7 \quad \mathrm{R} = 0 \\
7 & 3 \quad \mathrm{R} = 1 \\
3 & 1 \quad \mathrm{R} = 1 \\
1 & 0 \quad \mathrm{R} = 1 \\
0 &
\end{array}
\quad \Big\uparrow \text{ordine}
$$

da cui si ottiene che $57_{10} = 111001_2$

UNIVERSITÀ DEGLI STUDI DELLA CAMPANIA
LUIGI VANVITELLI
SCUOLA POLITECNICA E DELLE SCIENZE DI BASE
DIPARTIMENTO DI INGEGNERIA CIVILE
DESIGN EDILIZIA E AMBIENTE

# *Esercizio*

- Convertire:

N = 101011,1011 da base 2 a base 10

## *Soluzione*

| $2^5$ | $2^4$ | $2^3$ | $2^2$ | $2^1$ | $2^0$ | $2^{-1}$ | $2^{-2}$ | $2^{-3}$ | $2^{-4}$ |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 |

da cui si ha: $N = 1{\cdot}2^5{+}0{\cdot}2^4{+}1{\cdot}2^3{+}0{\cdot}2^2{+}1{\cdot}2^1{+}1{\cdot}2^0{+}1{\cdot}2^{-1}{+}0{\cdot}2^{-2}{+}1{\cdot}2^{-3}{+}1{\cdot}2^{-4} = 43.6875_{10}$

## *Esercizio*

- Convertire:

0,625 da base 10 a binario accumulando al massimo 8 cifre

# Esercizi

1. Convertire in formato ***decimale*** i seguenti numeri ***binari***:
   11, 101011, 1100, 111111, 10101010
2. Convertire in ***decimale*** i seguenti numeri frazionari ***binari*** :
   0.111, 0.0101, 0.00011
3. Convertire in formato ***decimale*** i seguenti numeri ***ottali:***
   12, 23, 345, 333.14, 560.271
4. Convertire in formato ***decimale*** i seguenti numeri ***esadecimali:***
   12.5, DAB, 15D, FFFF, 51A
5. Convertire in ***binario*** i seguenti numeri ***decimali*** (considerando 6 bit per la parte frazionaria):
   45.226, 234.349, 67.712, 83.8123
6. Convertire in ***ottale*** e in ***esadecimale*** i ***numeri binari*** ottenuti dalla conversione dei numeri decimali di cui al punto precedente

UNIVERSITÀ DEGLI STUDI DELLA CAMPANIA
Luigi Vanvitelli
SCUOLA POLITECNICA E DELLE SCIENZE DI BASE
DIPARTIMENTO DI INGEGNERIA CIVILE
DESIGN EDILIZIA E AMBIENTE

# *Soluzioni*

$\mathbf{11}_{due} = (1x2^1 + 1x2^0)_{dieci} = (2 + 1)_{dieci} = 3_{dieci}$

$\mathbf{101011}_{due} = (1x2^5 + 0x2^4 + 1x2^3 + 0x2^2 + 1x2^1 + 1x2^0)_{dieci} = (32 + 8 + 2 + 1)_{dieci} = 43_{dieci}$

$\mathbf{1100}_{due} = (1x2^3 + 1x2^2 + 0x2^1 + 0x2^0)_{dieci} = (8 + 4 + 0 + 0)_{dieci} = 12_{dieci}$

$\mathbf{111111}_{due} = (1x2^5 + 1x2^4 + 1x2^3 + 1x2^2 + 1x2^1 + 1x2^0)_{dieci} = (32 + 16 + 8 + 4 + 2 + 1)_{dieci}$
$= 63_{dieci}$

$\mathbf{10101010}_{due} = (1x2^7 + 0x2^6 + 1x2^5 + 0x2^4 + 1x2^3 + 0x2^2 + 1x2^1 + 0x2^0)_{dieci} = (128 + 32$
$+ 8 + 2)_{dieci} = 170_{dieci}$

# *Soluzioni*

$\mathbf{0.111}_{due} = (1x2^{-1} + 1x2^{-2} + 1x2^{-3})_{dieci} = (0.5 + 0.25 + 0.125)_{dieci} = 0.875_{dieci}$

$\mathbf{0.0101}_{due} = (0x2^{-1} + 1x2^{-2} + 0x2^{-3} + 1x2^{-4})_{dieci} = (0.25 + 0.0625)_{dieci} = 0.3125_{dieci}$

$\mathbf{0.00011}_{due} = (0x2^{-1} + 0x2^{-2} + 0x2^{-3} + 1x2^{-4} + 1x2^{-5})_{dieci} = (0.0625 + 0.03125)_{dieci} =$
$= 0.09375_{dieci}$

# *Soluzioni*

$\mathbf{12}_{otto} = (1x8^1 + 2x8^0)_{dieci} = (8 + 2)_{dieci} = 10_{dieci}$

$\mathbf{23}_{otto} = (2x8^1 + 3x8^0)_{dieci} = (16 + 3)_{dieci} = 19_{dieci}$

$\mathbf{345}_{otto} = (3x8^2 + 4x8^1 + 5x8^0)_{dieci} = (3x64 + 32 + 5)_{dieci} = 229_{dieci}$

$\mathbf{333.14}_{otto} = (3x8^2 + 3x8^1 + 3x8^0 + 1x8^{-1} + 4x8^{-2})_{dieci} = (192 + 24 + 3 + 0.125 + 0.0625)_{dieci} = 219.1875_{dieci}$

$\mathbf{560.271}_{otto} = (5x8^2 + 6x8^1 + 0x8^0 + 2x8^{-1} + 7x8^{-2} + 1x8^{-3})_{dieci} = (320 + 48 + 0 + 0.25 + 0.109375 + 0.0019531)_{dieci} = 368.3613281_{dieci}$

# *Soluzioni*

$\mathbf{12.5}_{sedici} = (1x16^1 + 2x16^0 + 5x16^{-1})_{dieci} = (16 + 2 + 0.3125)_{dieci} = 18.3125_{dieci}$

$\mathbf{DAB}_{sedici} = (13x16^2 + 10x16^1 + 11x16^0)_{dieci} = (13x256 + 160 + 11)_{dieci} = 3499_{dieci}$

$\mathbf{15D}_{sedici} = (1x16^2 + 5x16^1 + 13x16^0)_{dieci} = (256 + 80 + 13)_{dieci} = 349_{dieci}$

$\mathbf{FFFF}_{sedici} = (15x16^3 + 15x16^2 + 15x16^1 + 15x16^0)_{dieci} = (15x4096 + 15x256 + 15x16 + 15)_{dieci} = (61440 + 3840 + 240 + 15)_{dieci} = 65535_{dieci}$

$\mathbf{51A}_{sedici} = (5x16^2 + 1x16^1 + 10x16^0)_{dieci} = (5x256 + 16 + 10)_{dieci} = 1306_{dieci}$

# *Soluzioni*

45.226 ➔ P.I. = 45 e P.F. = 0.226

**$45_{dieci}$**

45/2 = 22 con resto 1
22/2 = 11 con resto 0
11/2 = 5 con resto 1
5/2 = 2 con resto 1
2/2 = 1 con resto 0
1/2 = 0 con resto 1

**$0.226_{dieci}$**

| | |
|---|---|
| 0.226x2 = 0.452 | p.f. 0.452 p.i. 0 |
| 0.452x2 = 0.904 | p.f. 0.904 p.i. 0 |
| 0.904x2 = 1.808 | p.f. 0.808 p.i. 1 |
| 0.808x2 = 1.616 | p.f. 0.616 p.i 1 |
| 0.616x2 = 1.232 | p.f 0.232 p.i. 1 |
| 0.232x2 = 0.464 | p.f. 0.464 p.i. 0 |

45.226 = 101101.001110

# Soluzioni

234.349 ➔ P.I. = 234 e P.F. = 0.349

$234_{dieci}$

| | |
|---|---|
| 234/2 = 117 | con resto 0 |
| 117/2 = 58 | con resto 1 |
| 58/2 = 29 | con resto 0 |
| 29/2 = 14 | con resto 1 |
| 14/2 = 7 | con resto 0 |
| 7/2 = 3 | con resto 1 |
| 3/2 = 1 | con resto 1 |
| 1/2 = 0 | con resto 1 |

$0.349_{dieci}$

| | |
|---|---|
| 0.349x2 = 0.698 | p.f. 0.698 p.i. 0 |
| 0.698x2 = 1.396 | p.f. 0.396 p.i. 1 |
| 0.396x2 = 0.792 | p.f. 0.792 p.i. 0 |
| 0.792x2 = 1.584 | p.f. 0.584 p.i 1 |
| 0.584x2 = 1.168 | p.f 0.168 p.i. 1 |
| 0.168x2 = 0.336 | p.f. 0.336 p.i. 0 |

234.349 = 11101010.010110

## *Soluzioni*

67.712 → P.I. = 67 e P.F. = 0.712

**$67_{dieci}$**

| | |
|---|---|
| 67/2 = 33 | con resto 1 |
| 33/2 = 16 | con resto 1 |
| 16/2 = 8 | con resto 0 |
| 8/2 = 4 | con resto 0 |
| 4/2 = 2 | con resto 0 |
| 2/2 = 1 | con resto 0 |
| 1/2 = 0 | con resto 1 |

**$0.712_{dieci}$**

| | |
|---|---|
| 0.712x2 = 1.424 | p.f. 0.424 p.i. 1 |
| 0.424x2 = 0.848 | p.f. 0.848 p.i. 0 |
| 0.848x2 = 1.696 | p.f. 0.696 p.i. 1 |
| 0.696x2 = 1.392 | p.f. 0.392 p.i 1 |
| 0.392x2 = 0.784 | p.f 0.784 p.i. 0 |
| 0.784x2 = 1.568 | p.f. 0.568 p.i. 1 |

67.712 = 1000011.101101

# Soluzioni

83.8123 ➔ P.I. = 83 e P.F. = 0.8123

**$83_{dieci}$**

83/2 = 41 con resto 1
41/2 = 20 con resto 1
20/2 = 10 con resto 0
10/2 = 5 con resto 0
5/2 = 2 con resto 1
2/2 = 1 con resto 0
1/2 = 0 con resto 1

**$0.8123_{dieci}$**

0.8123x2 = 1.6246 p.f. 0.6246 p.i. 1
0.6246x2 = 1.2492 p.f. 0.2492 p.i. 1
0.2492x2 = 0.4984 p.f. 0.4984 p.i. 0
0.4984x2 = 0.9968 p.f. 0.9968 p.i. 0
0.9968x2 = 1.9936 p.f. 0.9936 p.i. 1
0.9936x2 = 1.9872 p.f. 0.9872 p.i. 1

83.8123 = 1010011.110011

# Soluzioni

101101.001110 → 101 101 . 001 110 = $55.16_{otto}$

11101010.010110 → 11 101 010 . 010 110 = $352.26_{otto}$

1000011.101101 → 001 000 011 . 101 101 = $103.55_{otto}$

1010011.110011 → 1 010 011 . 110 011 = $123.63_{otto}$

101101.001110 → 10 1101 . 0011 1000= $2D.38_{sedici}$

11101010.010110 → 1110 1010 . 0101 1000= $EA.58_{sedici}$

1000011.101101 → 100 0011 . 1011 0100= $43.B4_{sedici}$

1010011.110011 → 101 0011 . 1100 1100= $53.CC_{sedici}$

# *Esercizio*

- Sommare $110_2$ con $10_2$
- Sottrarre $11_2$ da $1110_2$

***RICORDIAMO CHE:***

$$
\begin{array}{llllll}
0 & + & 0 & = & 0 & \\
1 & + & 0 & = & 1 & \\
0 & + & 1 & = & 1 & \\
1 & + & 1 & = & 0 & \text{con riporto di 1 alla colonna immediatamente a sinistra}
\end{array}
$$

$$
\begin{array}{llllll}
0 & - & 0 & = & 0 & \\
1 & - & 0 & = & 1 & \\
1 & - & 1 & = & 0 & \\
0 & - & 1 & = & 1 & \text{dopo essersi prestato 1 dalla colonna a sinistra}
\end{array}
$$

## Soluzione

$$
\begin{array}{r c c}
110 & + & 6 \\
10 & = & 2 \\
\hline
1000 & & 8
\end{array}
$$

$$
\begin{array}{r c c}
1110 & - & 14 \\
11 & = & 3 \\
\hline
1011 & & 11
\end{array}
$$

# *Esercizio*

- $(110)_2 \times (10)_2 \; = \; ?$

- $(101)_2 \times (110)_2 \; = \; ?$

## *Soluzione*

- $(110)_2 \times (10)_2 \ = \ ?$

$$
\begin{array}{ccccccl}
 & 1 & 1 & 0 & \times & & \\
 & & 1 & 0 & = & & \\
\hline
 & 0 & 0 & 0 & + & & \rightarrow 110 \times 0 \\
1 & 1 & 0 & & = & & \rightarrow 110 \times 1 \\
\hline
1 & 1 & 0 & 0 & & &
\end{array}
$$

Quindi $(110)_2 \times (10)_2 \ = \ 1100$

- $(101)_2 \times (110)_2 \ = \ ?$

$$
\begin{array}{ccccccl}
 & & 1 & 0 & 1 & \times & \\
 & & 1 & 1 & 0 & = & \\
\hline
 & & 0 & 0 & 0 & + & \rightarrow 101 \times 0 \\
 & 1 & 0 & 1 & & + & \rightarrow 101 \times 1 \\
1 & 0 & 1 & & & = & \rightarrow 101 \times 1 \\
\hline
1 & 1 & 1 & 1 & 0 & &
\end{array}
$$

Ovvero $(101)_2 \times (110)_2 \ = \ (11110)_2$

## *La rappresentazione dei numeri relativi*

- Per i ***numeri relativ***i, ovvero tutti i numeri interi, positivi e negativi, incluso lo zero, si utilizzano invece altre tipologie di rappresentazioni.
- **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.

***La rappresentazione dei numeri relativi***


<u>Esempio:</u> si voglia convertire il numero -5 con 8 bit
Il primo bit (quello più significativo) viene posto a 1 perchè il numero è negativo

gli altri 7 bit si calcolano con il metodo visto prima che, applicato al numero 5, restituisce 0000101.
Dunque il numero binario che rappresenta -5 è:
10000101

## *Rappresentazione Segno e Modulo*

- La rappresentazione, detta per *segno e modulo*, consente di codificare tutti i numeri relativi appartenenti all'intervallo: $[-2^{l-1}+1, 2^{l-1}-1]$
- con $2^{l-1}$ valori positivi e altrettanti negativi: per un totale ***di*** $\boldsymbol{2^l}$ ***valori diversi***.
  - Es: Per $l$ =8 sono rappresentabili tutti i numeri relativi appartenenti all'intervallo [-127,127].
- **PROBLEMI:**
  - poiché 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.

| $\pm$ | $2^6$ | $2^5$ | $2^4$ | $2^3$ | $2^2$ | $2^1$ | $2^0$ |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| $\pm$ | $2^6$ | $2^5$ | $2^4$ | $2^3$ | $2^2$ | $2^1$ | $2^0$ |
|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

## *Rappresentazione Segno e Modulo*

**PROBLEMI: la rappresentazione per segno e modulo richiede un algoritmo complesso per effettuare somma e sottrazione in presenza delle diverse combinazioni dei segni degli operandi**.

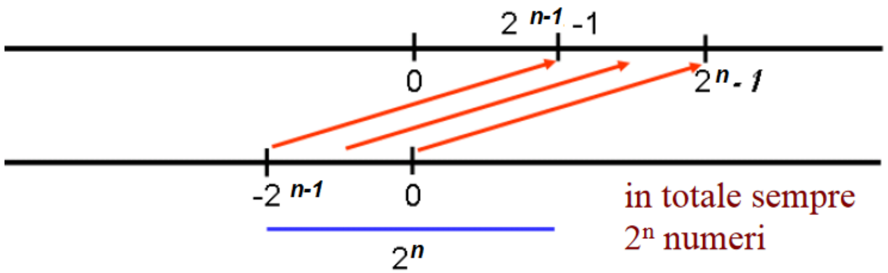
- Proviamo ad esempio ad eseguire -37+1

$$\begin{array}{cccccccccl} 1 & 0 & 1 & 0 & 0 & 1 & {}^{1}0 & 1 & + \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & = \\ \hline 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & = -38 \end{array}$$

- **DOVREMMO OTTENERE -36 MA OTTENIAMO -38**
- La somma fornisce risultati errati se gli addendi hanno segno diversi

# *Rappresentazione Complemento a 2*

- **Con la rappresentazione in *complemento a due*, somma e sottrazione si possono effettuare con lo stesso algoritmo, e quindi si possono affidare allo stesso circuito elettronico.**
- In complemento a 2 ***le configurazioni che hanno il bit più significativo uguale a zero***, cioè quelle comprese nell'intervallo $[0, 2^{n-1}-1]$, rappresentano se stesse (***numeri positivi***), mentre le configurazioni col bit più significativo uguale a uno, cioè quelle rientranti nell'intervallo $[2^{n-1}, 2^n - 1]$, rappresentano i numeri negativi che si ottengono traslando a sinistra l'intervallo di $2^n$, cioè l'intervallo $[-2^{n-1},-1]$.

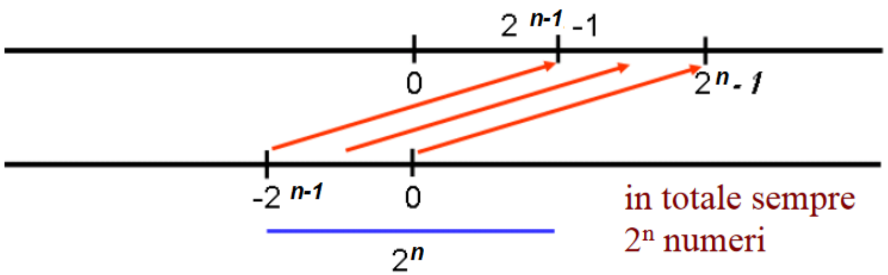



**0.......00 = 0**
**0.......01 = 1**
**...**
**01......11 = $2^{n-1}$-1 (massimo pos.)**
**10......00 = $-2^{n-1}$ (minimo neg.)**
**...**
**11......11 = -1**

## *Rappresentazione Complemento a 2*

- Pertanto, **i numeri positivi sono rappresentati normalmente** (rappresentazione binaria dei numeri positivi), con il bit più significativo pari a 0.
- **I numeri negativi si ottengono come complemento a 2 del numero positivo corrispondente**, ed hanno il bit più significativo pari a 1.
- Nella rappresentazione per complemento a 2, i valori rappresentati sono compresi nell'intervallo: $-(2^{n-1})$…….$+(2^{n-1}-1)$

**L'intervallo non è simmetrico!**




in totale sempre $2^n$ numeri

**0…….00 = 0**
**0…….01 = 1**
**…**
**01……11 = $2^{n-1}$-1 (massimo pos.)**
**10……00 = $-2^{n-1}$ (minimo neg.)**
**…**
**11……11 = -1**

## *Rappresentazione Complemento a 2*

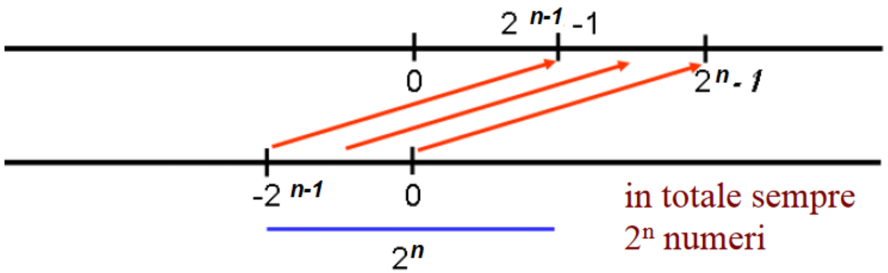
Con n bit, è possibile rappresentare numeri interi relativi il cui intervallo sarà:

$$-(2^{n-1}) \ldots\ldots +(2^{n-1}-1)$$

Se n=8 ---> -128,….,0,….,+127.

Se n=16 ---> -32.768,….,0,….,+32.767.

Se n=32 ---> -2.147.483.648,….,0,….,+2.147.483.647.




**0.......00 = 0**
**0.......01 = 1**
**...**
**01......11 = $2^{n-1}$-1 (massimo pos.)**
**10......00 = -$2^{n-1}$ (minimo neg.)**
**...**
**11......11 = -1**

## *Rappresentazione Complemento a 2*

- Come nella rappresentazione modulo e segno:
  - Positivi: cifra più significativa è 0 (rappresentati nella parte inferiore dell'intervallo)
  - Negativi: cifra più significativa è 1 (rappresentati nella parte superiore dell'intervallo)
- **Vantaggi** (della rappresentazione in complemento): Lo 0 ha una sola rappresentazione
- La somma si effettua **bit a bit**

# *Rappresentazione Complemento a 2*

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

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

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

$$c' = \boldsymbol{b} - 1 - \boldsymbol{c}$$

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

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

***OPPURE***

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 9 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| Si complementano le cifre | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 0 |
| Si somma 1 | | | | | | | | 1 |
| -9 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 |

## *Interpretazione Complemento a 2*

- Un'altra interpretazione del complemento a 2 riporta che il bit di segno, quello più significativo nella stringa di bit, contribuisca con peso negativo alla determinazione del valore nel sistema di numerazione posizionale pesato; in altri termini, con *l* bit, con la prima posizione che parte da zero $c_{l-1}$ è $-2^{l-1}$:

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

- come l'esempio che segue dimostra (sempre con *l* = 8).

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 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 |

## *Interpretazione Complemento a 2*

- l'interpretazione posizionale viene mantenuta e si modifica soltanto il suo peso, invertendolo.
- Il bit più significativo di una parola a 8 bit pesa $-2^7 = -128$

| $-2^7$ | $2^6$ | $2^5$ | $2^4$ | $2^3$ | $2^2$ | $2^1$ | $2^0$ |
|---|---|---|---|---|---|---|---|
| 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 |

- In complemento a 2 questa rappresentazione vale

$$-2^7 + 2^5 + 2^2 + 2^0 = -128 + 37 = -91.$$

# Esempio Codifica in Complemento a 2 su 4 bit

| Base 10 | Codifica su 4 bit del valore assoluto | Codifica su 3 bit del valore | Base 2 |
|---|---|---|---|
| -8 | 1000 | - | 1000 |
| -7 | 0111 | - | 1001 |
| -6 | 0110 | - | 1010 |
| -5 | 0101 | - | 1011 |
| -4 | 0100 | - | 1100 |
| -3 | 0011 | - | 1101 |
| -2 | 0010 | - | 1110 |
| -1 | 0001 | - | 1111 |
| 0 | - | 000 | 0000 |
| 1 | - | 001 | 0001 |
| 2 | - | 010 | 0010 |
| 3 | - | 011 | 0011 |
| 4 | - | 100 | 0100 |
| 5 | - | 101 | 0101 |
| 6 | - | 110 | 0110 |
| 7 | - | 111 | 0111 |

## *Rappresentazione Complemento a 2*

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

Esercizio: Rappresentare -35 in complemento a 2 su 8 bit

$$00100011_2 = +35_{10}$$

$$\downarrow$$

Inverti (complementa a 1) tutti i bit a sinistra del bit "1" meno significativo

$$11011101_2$$

## *Rappresentazione Complemento a 2*

- OPPURE
- complementando alla base tutte le cifre del valore assoluto del numero x e sommando poi 1 al valore ottenuto

Esercizio: Rappresentare $-35_{10}$ in complemento a 2 su 8 bit

**$00100011_2 = +35_{10}$**

↓ **Complemento a uno**

**11011100 +**
**1 =**
**------------**
**11011101**

## *Rappresentazione Complemento a 2*

Dato $N>0$, il numero $-N$ si rappresenta su $n$ bit con il numero unsigned $2^n - N$

$$-1 \Rightarrow 2^n - 1 \qquad (1........1)$$
$$-2^{n-1} \Rightarrow 2^n - 2^{n-1} = 2^{n-1} \qquad (10......0)$$

Esempio su 5 bit: rappresentare N = -15

$$N = 2^5 - 15 = 32 - 15 = 17 = 10001$$

- OPPURE

- Su 8 bit rappresentare N = -35
- $N = 2^8 - 35 = 256 - 35 = 221 = 11011101$

## *Rappresentazione Complemento a 2*

- Viceversa, se si ha una sequenza di l bit che rappresenta un numero intero con segno, con i numeri negativi rappresentati in complemento a 2, allora, per ottenere il numero rappresentato, si procede nel seguente modo.
  - Si esamina il bit di segno.
  - Se esso è zero, il numero rappresentato è non negativo e lo si calcola con la normale conversione binario-decimale.
  - Se invece il bit di segno è uno, allora si tratta di un numero negativo, per cui, per ottenerne il valore assoluto, si applica lo stesso procedimento visto in precedenza complementando tutti i bit e sommando 1 al risultato.
  - **11011101 → complemento i bit →00100010 sommo 1 → 00100011**
    **= 35 in valore assoluto**

## *Rappresentazione Complemento a 2*

- OPPURE

Data la rappresentazione di un numero negativo in complemento a due su n bit, il valore corrispondente si determina assegnando
- peso negativo al bit di segno ($-2^{n-1}$)
- peso positivo a tutti gli altri bit

Esempio su 5 bit:

$10011 = -2^4 + 2^1 + 2^0 = -13$

- Su 8 bit :
  - $11011101 = -2^7 + 2^6 + 2^4 + 2^3 + 2^2 + 2^0 =$
  - $-128 + 64 + 16 + 8 + 4 + 1 = -128 + 93 = -35$

## *Rappresentazione Complemento a 2*

- VANTAGGI:
- La somma bit a bit funziona sempre, non ci sono casi particolari (eccetto overflow)
- es. -37+41=4

$$
\begin{array}{ccccccccccc}
{}^{1} & {}^{1}1 & {}^{1}1 & {}^{1}0 & {}^{1}1 & 1 & {}^{1}0 & {}^{1}1 & 1 & + \\
 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & = \\
\hline
 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & = 4
\end{array}
$$

# *Esercizio*

- si determini il complemento a 2 del numero 10100.
- si determini il complemento a 2 del numero 01101001.

## *Soluzione*

- si determini il complemento a 2 del numero 10100.
- Tutti i bit 0 a partire dal bit meno significativo sono lasciati inalterati e così anche il primo bit 1.
- Tutti gli altri bit vengono invertiti, ottenendo: 01100.

- si determini il complemento a 2 del numero 01101001.
- In questo caso non esistono bit 0 a partire dal bit meno significativo.
- Solo il primo bit 1 viene lasciato inalterato.
- Gli altri vengono invertiti, ottenendo: 10010111.

## *Esercizio*

- si voglia convertire il numero 1 con 8 bit

- si voglia convertire il numero -1 con 8 bit

## *Soluzione*

si voglia convertire il numero 1 con 8 bit
Essendo il numero positivo:
Segno 0
Codifica binaria su 7 bit di 1: 0000001
Codifica: 00000001

si voglia convertire il numero -1 con 8 bit
Essendo il numero negativo:
Codifica binaria del valore assoluto (1) su 8 bits 00000001
complemento a 2 è 11111111.

## *Esempi di Conversione*

Si vuole convertire il numero **-12** in binario utilizzando una rappresentazione per *complementi a 2* su **5** bit

Si vuole convertire il numero **-10** in binario utilizzando una rappresentazione per *complementi a 2* su **4** bit.

Si vuole convertire il numero **111000** da *complementi a 2* in decimale.

| | |
|---|---|
| Si vuole convertire il numero **-12** in binario utilizzando una rappresentazione per *complementi a 2* su **5** bit | Poiché la rappresentazione per complementi a 2 consente la codifica di numeri relativi allora -12 è codificabile attraverso questa rappresentazione. Utilizzando 5 bit l'intervallo di rappresentazione è l'insieme dei numeri interi compresi in $[-2^{5-1}, 2^{5-1}-1]$, ovvero in [-16,15]. Poiché -12 è racchiuso in tale intervallo è possibile passare alla sua codifica.<br>Poiché -12 è negativo per ottenere il corrispettivo in binario si codifica il suo modulo su 5 bit, si complimentano le cifre ottenute e si aggiunge 1. Utilizzando l'algoritmo dell'esempio (1) si ha:<br>12:2 -> q1=6, r1=0<br>6:2 -> q2=3, r2=0<br>3:2 -> q3=1, r3=1<br>1:2 -> q4=0, r4=1<br>Considerando la successione dei resti invertita [r4,…,r1] e inserendo gli zeri a sinistra del bit più siginificativo, si ha che il numero in binario corrispondente è 01100.<br>Complementando le cifre ottenute si ha 10011, infine aggiungendo 1 al numero complementato si ha: **10100** |

## *Esempi di Conversione*

| | |
|---|---|
| Si vuole convertire il numero **-10** in binario utilizzando una rappresentazione per *complementi a 2* su **4** bit. | Poiché la rappresentazione per complementi a 2 consente la codifica di numeri relativi allora -10 è codificabile attraverso questa rappresentazione.. Utilizzando 4 bit l'intervallo di rappresentazione è l'insieme dei numeri interi compresi in $[-2^{4-1},2^{4-1}-1]$, ovvero in [-8,7]. Poiché -10 non è racchiuso in tale intervallo la sua codifica **non è possibile**. |

## *Esempi di Conversione*

| | |
|---|---|
| Si vuole convertire il numero **111000** da *complementi a 2* in decimale. | Innanzitutto si esamina il primo bit. Poiché esso è pari ad 1, il numero è negativo e quindi, per determinarne il modulo, si complementano ad 1 tutti i bit e si somma 1 al risultato. Il complemento è 000111, mentre sommando a tale numero 1, si ottiene 001000, il cui corrispettivo in decimale è $0*2^5 + 0*2^4 + 1*2^3 + 0*2^2 + 0*2^2 + 0*2^0 = 8$.<br>Il numero decimale corrispondente è quindi -**8**.<br>Lo stesso risultato poteva essere ottenuto, in via alternativa, aggiungendo a $-2^{l-1}$, ovvero a $-2^5 = -32$, la codifica in decimale di 11000, ovvero $1*2^4 + 1*2^3 + 0*2^2 + 0*2^2 + 0*2^0 = 24$.<br>Infatti -32+24=-**8**. |

## *Rappresentazione Complemento a 2*

Esempio: n=5, intervallo di rappresentazione: da $-2^4$ a $2^4-1$ (ovvero, da -16 a 15)

```
      00011           00111            11111
 +9   01001+     -9   10111+     -9    10111+
 +3   00011=     +3   00011=     +9    01001=
-----------     -----------     ------------
+12   01100      -6   11010       0    00000
```

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

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

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

Non è necessario preoccuparsi dei segni
Il risultato sarà corretto (in complemento a 2 se negativo) a meno di **trabocco (overflow)**.

# *Rappresentazione Complemento a 1*

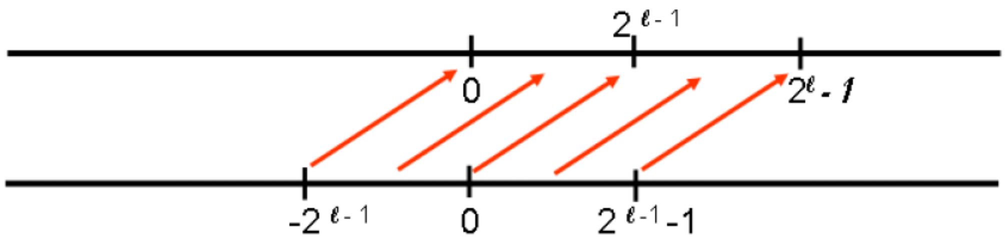
- Il *complemento a uno* (***x'***) del numero ***x*** si differenzia dal complemento a 2 (***x''***) dello stesso numero per una unità:

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

- Dalla definizione si comprende che il complemento a 1 di un numero, detto anche complemento diminuito o complemento alla base, si ottiene semplicemente complementando tutte le cifre del numero.
- complemento a 1 è stato usato in alcuni calcolatori, ma è stato abbandonato perché alla semplicità di determinazione dei numeri negativi (nel caso di base 2 basta sostituire ad ogni uno lo zero ed ad ogni zero l'uno) accompagna una doppia rappresentazione dello zero che complica le operazioni di somma e sottrazione.

## *Rappresentazione Per Eccessi*

- Nella rappresentazione per *eccesso* i numeri si rappresentano come somma di se stessi con $2^{l-1}$ dove $l$ è il numero di bit utilizzati.
- Si noti che 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 $[0, 2^{l}-1]$.



## *Rappresentazione Per Eccessi*

- 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).
  - Nel caso di n=4, k=$2^3$= 8, i numeri appartenenti a [–8,7] sono mappati nell'intervallo[0,15]
    - **-8 corrisponde a 0**
    - **0 corrisponde a 8**
    - **7 corrisponde a 15**

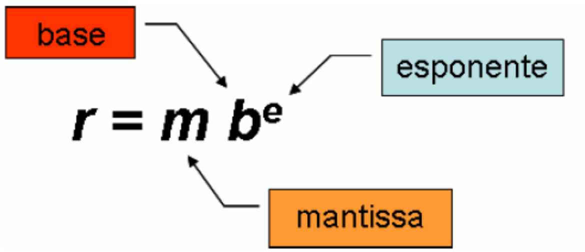
Rappresentazione di interi con 4 bit

| 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 |

***Overview***

## *Rappresentazione Reali*

- I numeri reali vengono rappresentati in binario attraverso la seguente notazione scientifica:
  - 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).



***Rappresentazione Reali***

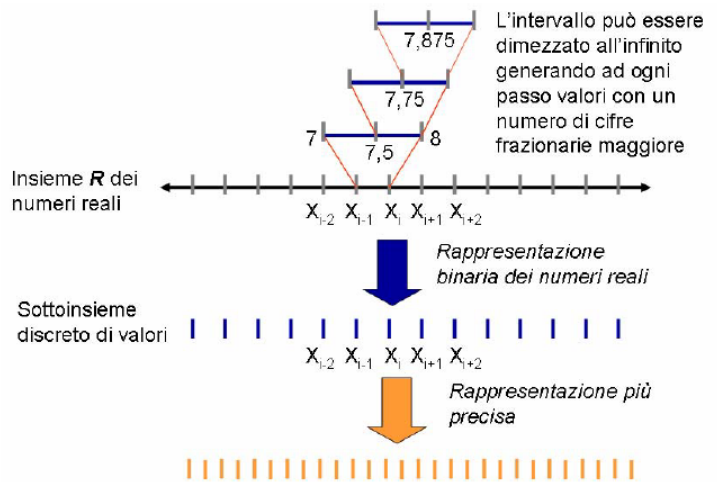
- Tale notazione scientifica viene adottata per diversi motivi:
  - la sua indipendenza dalla posizione della virgola;
  - la possibilità di trascurare tutti gli zeri che precedono la prima cifra significativa con la normalizzazione della mantissa;
  - la possibilità di rappresentare con poche cifre numeri molto grandi oppure estremamente piccoli;

## *Rappresentazione Finita e Discreta dei Reali*

- La rappresentazione in binario dei numeri reali si caratterizza rispetto alla notazione scientifica per alcune particolarità nel modo di rappresentare e utilizzare i numeri, dovute all'uso di ***rappresentazioni finite e definite sia per l'esponente che per la mantissa***.
- I ***due fattori*** limitano quindi sia ***l'intervallo dell'insieme dei numeri reali che è possibile rappresentare***, che ***il grado di precisione*** che essi avranno.
- Infatti in un intervallo reale comunque piccolo esistono infiniti valori (i numeri reali formano un continuo).

## *Rappresentazione Finita e Discreta dei Reali*

- I valori rappresentabili in binario appartengono 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, come la figura mostra, che ogni x appartenente all'intervallo [Xi, Xi+1[ viene sostituito con Xi.



## *Effetti delle Approssimazioni*

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

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

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

## *Errori di Arrotondamento*

- Il valore ε dipende dalla rappresentazione finita (numero finito di cifre) utilizzata per i numeri reali. Ad esempio disponendo di una calcolatrice con una aritmetica a quattro cifre decimali che applica le note regole di arrotondamento sull'ultima cifra, si ha:

| 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 | $13*10^{-6} = 0.13*10^{-4}$ |

- con un errore massimo sull'ultima cifra di 0.5 ($0.5 * 10^{-4}$) per le classiche regole dell'arrotondamento.

In generale se ***-m*** è il peso della cifra meno significativa, l'errore massimo che si commette è:

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

## *Underflow ed Overflow*

- L'insieme ***R*** è anche costituito da infiniti valori (è definito dall'intervallo ]-∞,∞[). I numeri reali rappresentabili sono invece definiti in un insieme limitato con estremi predefiniti *[-minreal, maxreal]*. Si definiscono:
  - l'overflow come la condizione che si verifica quando i valori o sono più piccoli di minreal o più grandi di maxreal;
  - l'underflow come la condizione che si verifica quando un valore, per effetto delle approssimazioni, viene confuso con lo zero.

## *Rappresentazione Normalizzata*

- La rappresentazione in virgola mobile, fissata la base, consente di esprimere lo stesso valore con infinite coppie (mantissa, esponente), ad esempio:
  - $48 \times 10^3$ è uguale a $4800 \times 10^1$, ma anche a $4,8 \times 10^4$
- 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 $2,5000 * 10^{-3}$). La mantissa scalata in questo modo prende il nome di *mantissa normalizzata* e il numero in virgola mobile, il nome di numero normalizzato.
- 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 Rappresentazione*

- Ad esempio disponendo di una calcolatrice con le seguenti caratteristiche:
  - 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 | $1,0000 \times 10^{-8}$ |

e la condizione di overflow quando:
$x > 9,9999 \times 10^{99}$
e di underflow quando:
$x < 1,0000 \times 10^{-99}$

## *Osservazione*

- Osservando la modalità di rappresentazione in virgola mobile, si può notare che 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 la calcolatrice precedente è facile osservare il fenomeno confrontando gli intervalli

$$[1.0000 \times 10^{-99}, 1.0001 \times 10^{-99}] \text{ e } [9.9998 \times 10^{99}, 9.9999 \times 10^{99}].$$

## *Operazioni in Virgola Mobile*

- Con la rappresentazione 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:
  - $1{,}9099\times10^{1} + 5{,}9009\times10^{4}$
  - con la calcolatrice considerata prima, diventa:
  - $0{,}0001\times10^{4} + 5{,}9009\times10^{4}$
- con il troncamento delle cifre 9099 del numero con esponente più piccolo.

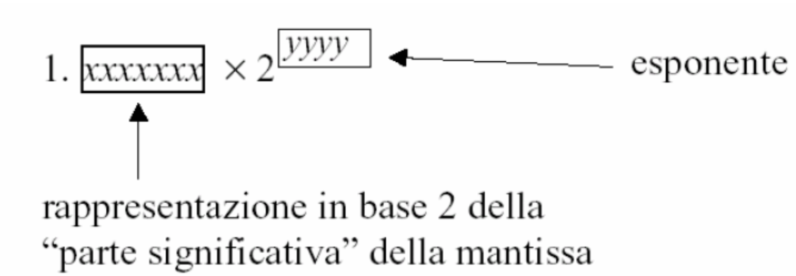
## *Operazioni in Virgola Mobile*

- Le operazioni che richiedono maggiore attenzione sono l'addizione e la sottrazione.
- Infatti, la causa principale degli errori di calcolo numerico risiede nella sottrazione di numeri di valore quasi uguale;
  - in tal caso le cifre più significative si eliminano fra loro e la differenza risultante perde un certo numero di cifre significative o anche tutte (fenomeno detto cancellazione).

Altra causa di errori è la divisione per valori molto piccoli, poichè il risultato può facilmente superare il valore di overflow: deve essere pertanto evitata non solo la divisione per lo zero ma anche per valori ad esso prossimi.

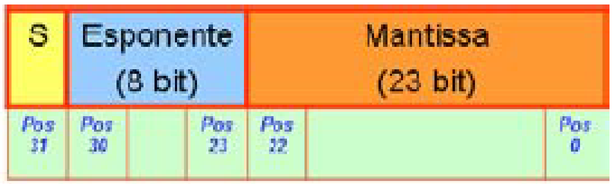
## *Perché la Virgola Mobile?*

- Il grande vantaggio della rappresentazione in virgola mobile è che, 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 × $10^{-9}$ viene rappresentato con la coppia di numeri interi (1230,-9) e gestito con operazioni interne che ne costruiscono l'effettivo valore.
- Nel caso binario la rappresentazione in virgola mobile normalizzata assume la forma:

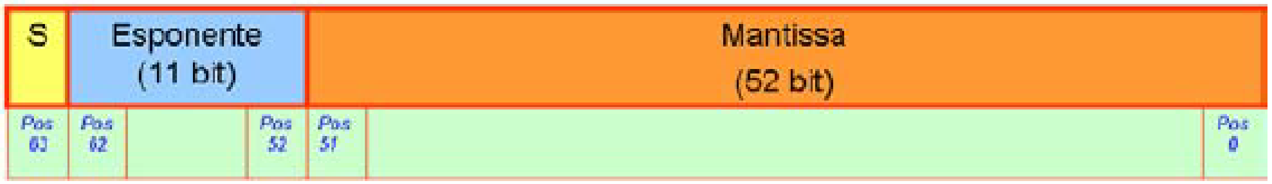


## *Standard Virgola Mobile*

- Nel 1985 divenne operativo lo Standard 754 IEEE per i numeri in virgola mobile che definisce principalmente tre formati numerici a virgola mobile:
  - singola precisione o precisione semplice (32 bit),
  - doppia precisione 64 bit),
  - precisione estesa (80 bit).




Singola precisione (32 bit)



## *Standard Virgola Mobile*

- Nella stringa di bit, si succedono nell'ordine da sinistra (MSB) a destra (LSB):
  - 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: l'insieme dell'uno implicito, della virgola implicita e delle cifre esplicite prende il nome di significante.

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

## *Operatori Booleani*

- Sulle stringhe di bit sono anche definiti operatori che lavorano bit a bit (*bitwise operator*). Essi sono detti *booleani* e sono:

  - ***AND***: dati due bit restituisce il valore 1 se e solo se i bit erano entrambi posti a 1, in tutti gli altri casi il risultato è 0; l'AND è detto anche *prodotto logico*.

  - ***OR***: dati due bit restituisce il valore 0 se e solo se i bit erano entrambi posti a 0, in tutti gli altri casi il risultato è 1; l'OR è anche detto *somma logica*.

  - ***NOT***: dato un bit restituisce il valore 0 se esso era posto a 1, restituisce invece 1 se il bit era posto a 0; il NOT viene anche detto operatore di *negazione* o di *complementazione*.

# *Tavole di Verità*

| a | b | *NOT* a | a *AND* b | a *OR* b |
|---|---|---|---|---|
| 0 | 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 0 | 1 |
| 1 | 1 | 0 | 1 | 1 |

***Si noti che la somma logica di 1 OR 1 fa 1 e non 0 con riporto 1 come nel caso della somma aritmetica.***

# *Esempi*

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | AND |
| 1 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | = |
| 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | OR |
| 1 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | = |
| 1 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | NOT |
| 1 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | |

## *Esempi di Conversione*

- codifichiamo il numero −118.5 nel sistema IEEE 754.
  - Dobbiamo determinarne il segno, l'esponente e la mantissa.
- Poiché è un numero negativo, il primo bit è "1".
- Poi scriviamo il numero in forma binaria: 1110110,1.
- Successivamente spostiamo la virgola verso sinistra, lasciando solo un 1 alla sua sinistra:1110110,1=1,1101101 x $2^6$
- La mantissa è la parte a destra della virgola, riempita con zeri a destra fino a riempire i 23 bit: 11011010000000000000000.
- L'esponente è pari a 6, ma dobbiamo convertirlo in forma binaria e adattarlo allo standard. Per la precisione singola, dobbiamo aggiungere 127. Quindi 6 + 127 = 133. In forma binaria: 10000101.

```
 1     8               23
+-+--------+-----------------------+
|S|  Esp   |  Mantissa             |
|1|10000101|11011010000000000000000|
+-+--------+-----------------------+
31 30      22                      0
```

# *Esempi di Conversione*

## Conversione da virgola mobile a decimale

Consideriamo il seguente numero rappresentato in formato IEEE 754:

| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 31 | | | | | | | | 23 | | | | | | | | | | | | | | | | | | | | | | | 0 |

Abbiamo che:

- $s = 1$
- $e = 2^7 + 2^0 = 129 \Rightarrow E = 129 - 127 = 2$
- $m = 1 + 2^{-2} = 1.25$, in cui l'1 sommato a $2^{-2}$ viene sommato poiché, essendo la mantissa normalizzata, l'1 è implicito.

Pertanto il numero rappresentato da questa parola di bit è dato da:

$$N = (-1)^s \cdot 2^E \cdot 1.m = (-1)^1 \cdot 2^2 \cdot 1.25 = -5$$

È dunque evidente che, nonostante il formato sia in virgola mobile, è anche possibile rappresentare numeri interi!

UNIVERSITÀ DEGLI STUDI DELLA CAMPANIA
LUIGI VANVITELLI
SCUOLA POLITECNICA E DELLE SCIENZE DI BASE
DIPARTIMENTO DI INGEGNERIA CIVILE
DESIGN EDILIZIA E AMBIENTE

# *Esercizio*

## Conversione da decimale a virgola mobile

numero decimale $N = -5,828125_{10}$.

# *Soluzione*

## Conversione da decimale a virgola mobile

Prendiamo in considerazione il numero decimale $N = -5,828125_{10}$. Procediamo attraverso i seguenti passi:

### a) Determinazione del segno

Poiché il numero è negativo, poniamo $s = 1$.

### b) Conversione della parte intera

Procediamo come nel caso di numero intero qualsiasi:

$$
\begin{array}{r|l}
\multicolumn{2}{c}{/2} \\
5 & 2 \qquad \mathrm{R} = 1 \\
2 & 1 \qquad \mathrm{R} = 0 \\
1 & 0 \qquad \mathrm{R} = 1 \\
0 &
\end{array}
\quad \uparrow \text{ordine}
$$

da cui si ottiene che $5_{10} = 101_2$.

# *Soluzione*

## c) Conversione della parte frazionaria

Anche qui, la conversione segue sempre la stessa regola:

$$
\begin{array}{r|lll}
 & \overset{*2}{\curvearrowright} & & \\
0.828125 & 1.65625 & \mathrm{U}=1 & \\
0.65625 & 0.3125 & \mathrm{U}=1 & \\
0.3125 & 0.625 & \mathrm{U}=0 & \\
0.625 & 1.25 & \mathrm{U}=1 & \text{ordine} \downarrow \\
0.25 & 0.5 & \mathrm{U}=0 & \\
0.5 & 1 & \mathrm{U}=1 & \\
0 & & &
\end{array}
$$

da cui si ottiene che $0.828125_{10} = 0.110101_2$.

# Soluzione

## d) Normalizzazione della mantissa

Il numero che abbiamo calcolato fin'ora è $101.110101_2$. Lo standard IEEE 754 richiede che, per essere rappresentato, il numero sia nella forma $1.m$, così da poter omettere l'unità 1 che risulta essere sempre implicita[1] . Pertanto vale l'uguaglianza:

$$101.110101 = 1,\underbrace{01110101}_{m} \cdot 2^2$$

In cui, avendo spostato la virgola verso sinistra di due posizioni, dobbiamo moltiplicare il numero per $2^2$. La mantissa $m$ è a questo punto determinata.

## e) Rappresentazione dell'esponente

Il nostro numero è ora nella forma $1.m * 2^E$. Dobbiamo rappresentare il nostro esponente $E = 2$. Dobbiamo prima di tutto applicare il bias:

$$e = E + 127 = 2 + 127 = 129$$

E possiamo ora procedere a convertire questo valore in binario, come sempre:

# Soluzione

E possiamo ora procedere a convertire questo valore in binario, come sempre:

$$
\begin{array}{r|lll}
\multicolumn{2}{c}{/2} & & \\
129 & 64 & \mathrm{R} = 1 & \uparrow \\
64 & 32 & \mathrm{R} = 0 & \\
32 & 16 & \mathrm{R} = 0 & \\
16 & 8 & \mathrm{R} = 0 & \text{ordine} \\
8 & 4 & \mathrm{R} = 0 & \\
4 & 2 & \mathrm{R} = 0 & \\
2 & 1 & \mathrm{R} = 0 & \\
1 & 0 & \mathrm{R} = 1 & \\
0 & & &
\end{array}
$$

e pertanto il nostro esponente sarà $e = 10000001$.

[1]La scelta di questa rappresentazione consente di aumentare di un bit l'espressività della mantissa, proprio perché l'1 prima della virgola viene omesso.

# *Soluzione*

## f) Rappresentazione complessiva

Abbiamo determinato i valori di $s$, $e$ ed $m$. Il numero in formato IEE 754 è dato dalla concatenazione dei tre campi:

| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | = -5,828125 |
| 31 | | | | | | | | 23 | | | | | | | | | | | | | | | | | | | | | | | 0 | |

## *Esempi di Conversione*

Si vuole codificare il numero reale **8.25** utilizzando la rappresentazione in virgola mobile nel formato a singola precisione.

## *Esempi di Conversione*

| | |
|---|---|
| Si vuole codificare il numero reale **8.25** utilizzando la rappresentazione in virgola mobile nel formato a singola precisione. | Innanzitutto si codificano in binario la parte intera e frazionaria della mantissa e l'esponente (utilizzando la rappresentazione per eccessi a $2^{l-1}$) ottenendo 0 111.01 * $2^{00000000}$. Normalizzando la mantissa si ottiene: 0 1.1101 * $2^{10000010}$ |

**scriviamo il numero in forma binaria: 1000,01.**

**Successivamente spostiamo la virgola verso sinistra, lasciando solo un 1 alla sua sinistra:= 1000,01 =1,00001 x $2^3$**

**3+127=130=10000010**

**0 10000010 00001000000000000000000**