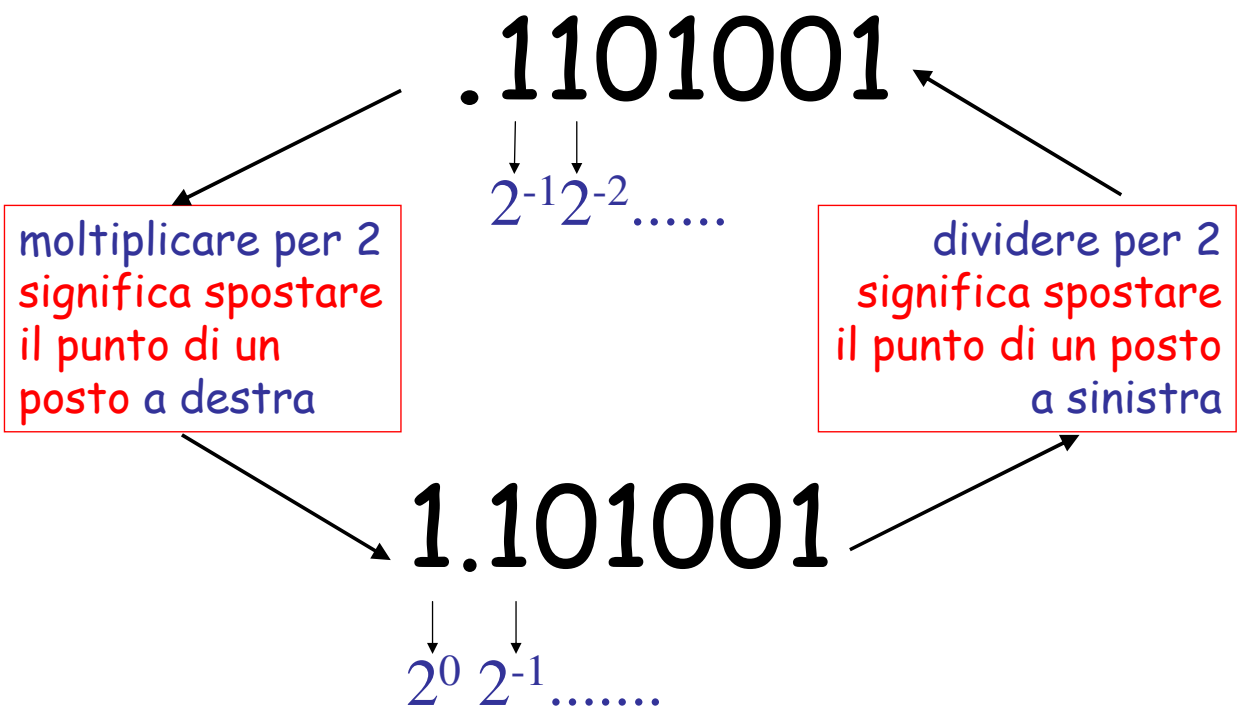
Numeri decimali

Cosa significa una parte decimale binaria?

.1101001

$2^{-1} + 2^{-2} + 2^{-4} + 2^{-7} = .5+.025+.0125+.00625$



- Il trucchetto di spostare la virgola x moltiplicare o dividere un numero funziona anche per le altre basi !
- Esempi:
  - $(1.234)_{10} / 10 = (.1234)_{10}$
  - $(.01234)_{10} \times 100 = (1.234)_{10}$
  - Ma anche $(463.427)_8 / 16 = (4.63427)_8$

  E così via..

Se abbiamo un valore decimale in base 10, ad esempio 0.99, come troviamo la sua rappresentazione in base 2? Ragioniamo come segue:

Supponiamo che

$(.99)_{10} = .b_1b_2b_3...b_k$ (in binario)

allora $2 \times .99 = \mathbf{1}.98 = \mathbf{b_1}.b_2...b_k$

quindi b_1 è 1

e .98 è rappresentato da $.b_2...b_k$

Quindi: per trovare la rappresentazione binaria di un valore decimale lo moltiplichiamo per 2 e osserviamo la cifra che appare nella parte intera.

Esempio: $(.59)_{10} = (?)_2$

.59×2= 1.18

.18×2= 0.36

.36×2= 0.72

.72×2= 1.44

.44x2= 0.88

.88x2= 1.76

...

$(0.59)_{10}=(.100101\ldots)_2$

E così via… dipende da quanti bit abbiamo a disposizione!

N.B. Mi posso fermare se moltiplicando per 2 ottengo una parte decimale = 0

Esempio Completo

$(18.59)_{10} = (?)_2$

$(18)_{10} = (10010)_2$

$(.59)_{10} = (.100101\ldots)_2$

Ne deriva che:

$(18.59)_{10} = (10010.100101\ldots)_2$

Osserviamo che spostando la virgola di h posti verso destra nella rappresentazione decimale di un numero N moltiplichiamo N per 10^h mentre spostandola verso sinistra di h posti N viene diviso per 10^h. Ad esempio:

$18.59 = 18.59 \times 10^0$
$= 1.859 \times 10^1$ rappresentazione normalizzata
$= 0.1859 \times 10^2$
$= 185.9 \times 10^{-1}$
$= 1859 \times 10^{-2}$

Si parla di rappresentazione in virgola mobile. Quindi in virgola mobile lo stesso numero decimale ha piu' rappresentazioni possibili. Solitamente si usa la rappresentazione normalizzata in cui la parte intera ha un'unica cifra decimale diversa da zero.

La rappresentazione in virgola mobile per i numeri binari e` del tutto analoga. Ad esempio:

$101.01 = 101.01 \times 2^0$
$= 10.101 \times 2^1$
$= 1.0101 \times 2^2$ Rappresentazione normalizzata
$= 1010.1 \times 2^{-1}$
$= 10101 \times 2^{-2}$

La rappresentazione in virgola mobile normalizzata ha sempre parte intera uguale a 1.

Per registrare un numero reale x in memoria si adotta la rappresentazione binaria in virgola mobile normalizzata:

$$x = 2^e \times 1.b_{-1}b_{-2}b_{-3}b_{-4}\ldots$$

Naturalmente non e` possibile memorizzare la sequenza possibilmente infinita di bit della parte frazionaria ma si memorizzano soltanto i primi bit. Anche per l'esponente si utilizzano un numero finito di bit. Generalmente per rappresentare un numero reale si usano 4 byte nel modo seguente:

+/-	$e_8e_7e_6e_5\ e_4e_3e_2e_1$	$b_{-1}b_{-2}b_{-3}b_{-4}\ \ldots\ b_{-21}b_{-22}b_{-23}$
1 bit	**8 bit**	**23 bit**

+/-	$e_8e_7e_6e_5\ e_4e_3e_2e_1$	$b_{-1}b_{-2}b_{-3}b_{-4}\ \ldots\ b_{-21}b_{-22}b_{-23}$

SEGNO **ESPONENTE** **MANTISSA**

Il primo bit rappresenta il segno del numero: 0 per +, 1 per -.
Gli otto bit successivi si usano per rappresentare l'esponente exp. Dovendo rappresentare sia esponenti positivi che negativi la sequenza di bit $e_8e_7e_6e_5\ e_4e_3e_2e_1$ e` la rappresentazione binaria di esp + 127. Con 8 bit si rappresentano gli interi compresi tra 0 e 255. Siccome le sequenze di bit 00000000 (0) e 11111111 (255) sono riservate (problema che non consideriamo), gli esponenti rappresentabili sono quelli per cui
$1 \leq esp + 127 \leq 254$ ovvero $-126 \leq esp \leq 127$.

Rappresentazione dei decimali

Quindi si rappresenta il numero

$$N = s\ 2^{e-127} \times 1.m$$