Appendice A

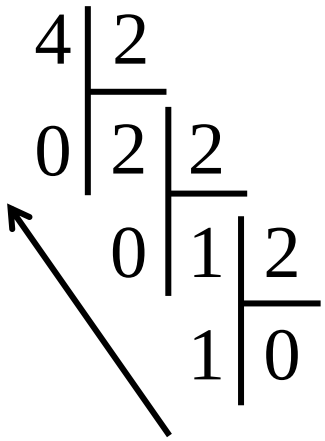
Conversione Decimale ↔ Binario

(numeri non negativi)

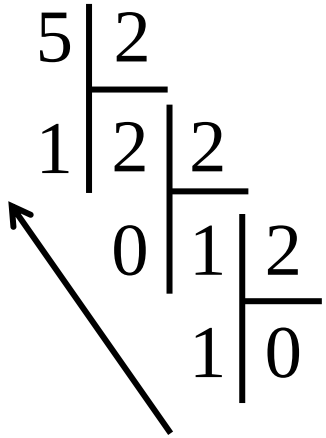
ε - macchina

Conversione Decimale → Binario (Numeri interi [≥1])

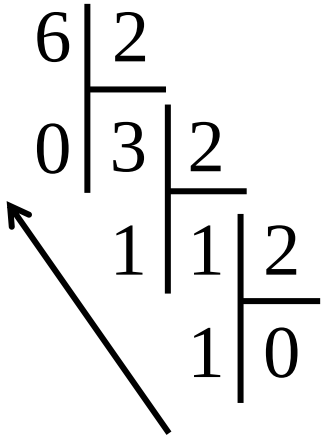
- Si applica la divisione intera per 2 fino a ottenere 0. I resti delle divisioni, letti al contrario, formano la rappresentazione in binario



$4_{10} = 100_2$



$5_{10} = 101_2$



$6_{10} = 110_2$

Conversione Binario → Decimale (Numeri interi [≥1])

- Il numero in Decimale è dato dal risultato della somma $b_{i-1}2^{i-1} + \cdots + b_1 2^1 + b_0 2^0$, dove *b* è l'*i*-esimo bit della rappresentazione in Binario

$$100_2 = 1 \cdot 2^2 + 0 \cdot 2^1 + 0 \cdot 2^0 = 4_{10}$$

$$101_2 = 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = 5_{10}$$

$$110_2 = 1 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0 = 6_{10}$$

Conversione Decimale → Binario (Numeri non interi [fra 0 e 1])

- Si applica la moltiplicazione per 2 fino a ottenere 1. Ogni volta che una parte intera è 1 si sottrae 1. Le parti intere delle moltiplicazioni, lette nell'ordine, formano la rappresentazione in binario

$0.5_{10} =.xxx_2?$

$0.5 \cdot 2 = 1 \rightarrow 1$

$0.5_{10} =.1_2$

$0.75_{10} =.xxx_2?$

$0.75 \cdot 2 = 1.5 \rightarrow 1$

$0.5 \cdot 2 = 1 \rightarrow 1$

$0.75_{10} =.11_2$

$0.6875_{10} =.xxx_2?$

$0.6875 \cdot 2 = 1.375 \rightarrow 1$

$0.375 \cdot 2 = 0.75 \rightarrow 0$

$0.75 \cdot 2 = 1.5 \rightarrow 1$

$0.5 \cdot 2 = 1 \rightarrow 1$

$0.625_{10} =.1011_2$

Conversione Decimale → Binario (Numeri non interi [fra 0 e 1])

- Caso particolare: se prima di ottenere 1 (e quindi concludere la conversione) si ottiene un decimale d già visto in un passo precedente allora si può concludere la conversione notando che la rappresentazione in binario del numero decimale iniziale conterrà una parte periodica, a partire dal decimale d.

$$0.6_{10} =. xxx_2?$$

$$\begin{aligned} 0.6 \cdot 2 &= 1.2 \rightarrow 1 \\ 0.2 \cdot 2 &= 0.4 \rightarrow 0 \\ 0.4 \cdot 2 &= 0.8 \rightarrow 0 \\ 0.8 \cdot 2 &= 1.6 \rightarrow 1 \end{aligned}$$

$$0.6\ gia'visto \rightarrow periodico$$

$$0.6_{10} =. \overline{1001}_2$$

Conversione Binario → Decimale (Numeri non interi [fra 0 e 1])

- Il numero in Decimale è dato dal risultato della somma $b_1 2^{-1} + b_2 2^{-2} + \cdots + b_i 2^{-i}$, dove *b* è l'*i*-esimo bit della rappresentazione in Binario

$$.100_2 = 1 \cdot 2^{-1} + 0 \cdot 2^{-2} + 0 \cdot 2^{-3} = \\ = 1 \cdot \frac{1}{2} + 0 \cdot \frac{1}{4} + 0 \cdot \frac{1}{8} = 0.5_{10}$$

$$.110_2 = 1 \cdot \frac{1}{2} + 1 \cdot \frac{1}{4} + 0 \cdot \frac{1}{8} = 0.75_{10}$$

$$.101_2 = 1 \cdot \frac{1}{2} + 0 \cdot \frac{1}{4} + 1 \cdot \frac{1}{8} = 0.625_{10}$$

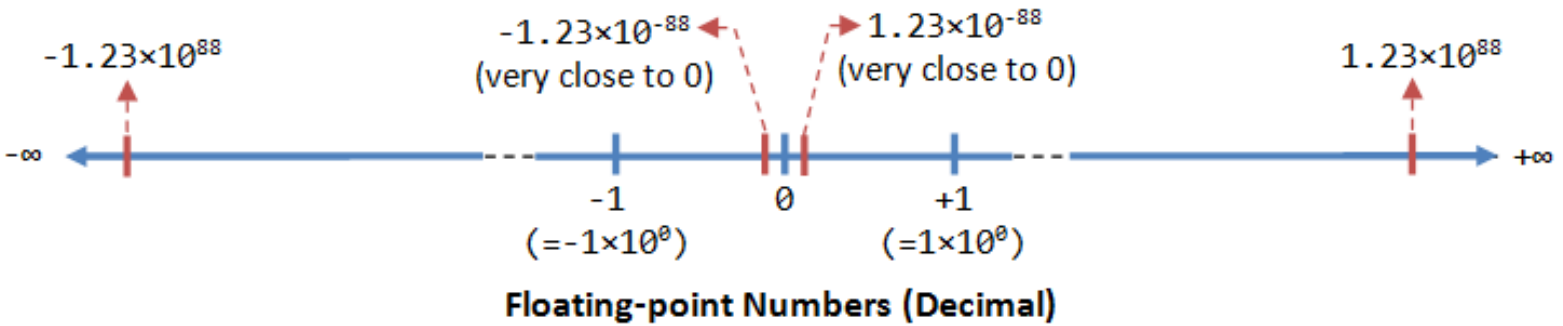
Conversione Decimale ↔ Binario (Numeri interi e non interi [>0])

- Si possono applicare gli algoritmi visti precedentemente combinandoli insieme sulla parte intera e su quella frazionaria.

- Esempio Binario → Decimale:
  - $1101.0011_2 =$
    $= 2^3 + 2^2 + 1 + \frac{1}{2^3} + \frac{1}{2^4} =$
    $= 8 + 4 + 1 + 0.125 + 0.0625 = 13.1875_{10}$

ε - macchina

- La rappresentazione dei numeri a virgola mobile tramite i moderni calcolatori è vincolata da una precisione limitata



Questo grafico mostra come i numeri rappresentabili siano *maggiormente raggruppati* intorno allo 0, mentre tendono a diventare più sparsi man mano che ci si allontana verso infinito

ε - macchina

- Definiamo ***ε – macchina*** il più piccolo numero in virgola mobile in valore assoluto diverso da 0 che sommato all'unità dia un risultato diverso da 1.
- In maniera alternativa, possiamo dire che se la differenza di due numeri *a* e *b* *distinti* risulta

$$a - b < \varepsilon$$

  allora $a - b = 0$, cioè $a = b$, contro l'hp iniziale.
- Questa caratteristiche potrebbe quindi essere responsabile di perdita di dati

ε - macchina

- E' possibile calcolare l'***ε – macchina***.
- Esistono già delle approssimazioni teoriche del suo valore:

IEEE 754 - 2008	Common name	C++ data type	Base b	Precision p	Machine epsilon[a] $b^{-(p-1)}/2$	Machine epsilon[b] $b^{-(p-1)}$
binary16	half precision	not available	2	11 (one bit is implicit)	2^{-11} = 4.88e-04	2^{-10} = 9.77e-04
binary32	single precision	float	2	24 (one bit is implicit)	2^{-24} = 5.96e-08	2^{-23} = 1.19e-07
binary64	double precision	double	2	53 (one bit is implicit)	2^{-53} = 1.11e-16	2^{-52} = 2.22e-16
binary80	extended precision	_float80[1]	2	64	2^{-64} = 5.42e-20	2^{-63} = 1.08e-19
binary128	quad(ruple) precision	_float128[1]	2	113 (one bit is implicit)	2^{-113} = 9.63e-35	2^{-112} = 1.93e-34
decimal32	single precision decimal	_Decimal32[2]	10	7	5×10^{-7}	10^{-6}
decimal64	double precision decimal	_Decimal64[2]	10	16	5×10^{-16}	10^{-15}
decimal128	quad(ruple) precision decimal	_Decimal128[2]	10	34	5×10^{-34}	10^{-33}