



Rappresentazione dell'informazione

Numeri in virgola fissa e
Numeri in virgola mobile

Numeri frazionari

- In una generica base B ,
il numerale

$$0, b_{-1}b_{-2} \dots b_{-m}$$

$$\text{con } b_i \in \beta = \{0, 1, \dots, B-1\}$$

si interpreta come

$$b_{-1} \cdot B^{-1} + b_{-2} \cdot B^{-2} + \dots + b_{-m} \cdot B^{-m}$$

Numeri frazionari

■ Esempi:

□ $B = 10, \beta = \{0, 1, 2, \dots, 9\}$ (*sistema decimale*)

$$0,356 = 3 \cdot 10^{-1} + 5 \cdot 10^{-2} + 6 \cdot 10^{-3}$$

□ $B = 2, \beta = \{0, 1\}$ (*sistema binario*)

$$0,101 = 1 \cdot 2^{-1} + 0 \cdot 2^{-2} + 1 \cdot 2^{-3}$$

Conversione binario - decimale

- Si ottiene valutando l'espressione $b_{-1} \cdot 2^{-1} + \dots + b_{-m} \cdot 2^{-m}$:

- $0,101_2 = 1 \cdot 2^{-1} + 0 \cdot 2^{-2} + 1 \cdot 2^{-3}$
$$= \frac{1}{2} + \frac{1}{8} = 0,5 + 0,125 = 0,625_{10}$$

- $0,1001_2 = 1 \cdot 2^{-1} + 0 \cdot 2^{-2} + 0 \cdot 2^{-3} + 1 \cdot 2^{-4}$
$$= \frac{1}{2} + \frac{1}{16} = 0,5 + 0,0625 = 0,5625_{10}$$

Conversione decimale - binario

- Si ottiene col metodo delle moltiplicazioni successive:

$$F = b_{-1} \cdot 2^{-1} + b_{-2} \cdot 2^{-2} + \dots + b_{-m} \cdot 2^{-m}$$



moltiplicando per due si ha:

$$2 \cdot F = b_{-1} + b_{-2} \cdot 2^{-1} + \dots + b_{-m} \cdot 2^{-(m-1)}$$



*si estrae
la parte intera*



*si itera il procedimento
sulla parte frazionaria*

Conversione decimale - binario

■ Esempio: $F = 0,78125$

$$\left. \begin{array}{l} 2 \cdot 0,78125 = \mathbf{1},5625 \\ 2 \cdot 0,5625 = \mathbf{1},125 \\ 2 \cdot 0,125 = \mathbf{0},25 \\ 2 \cdot 0,25 = \mathbf{0},5 \\ 2 \cdot 0,5 = \mathbf{1},0 \end{array} \right\} (0,78125)_{10} = (0,11001)_2$$



Il processo termina quando la parte frazionaria si annulla (oppure quando si è raggiunto il numero desiderato di cifre dopo la virgola)

Conversione decimale - binario

■ Esempio: $F = 0,9$

$$2 \cdot 0,9 = \mathbf{1},8$$

$$2 \cdot 0,8 = \mathbf{1},6$$

$$2 \cdot 0,6 = \mathbf{1},2$$

$$2 \cdot 0,2 = \mathbf{0},4$$

$$2 \cdot 0,4 = \mathbf{0},8$$

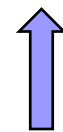
$$2 \cdot 0,8 = \mathbf{1},6$$

$$2 \cdot 0,6 = \mathbf{1},2$$

$$2 \cdot 0,2 = \mathbf{0},4$$

$$2 \cdot 0,4 = \mathbf{0},8$$

$$(0,9)_{10} = (0,11100\mathbf{1100})_2$$



periodico

Numeri in virgola fissa

- Un generico numero N è costituito da una parte intera e da una parte frazionaria, separate tra loro da una virgola:

$$b_{n-1}b_{n-2} \dots b_1b_0, b_{-1}b_{-2} \dots b_{-m}$$



$$N = b_{n-1} \cdot B^{n-1} + b_{n-2} \cdot B^{n-2} + \dots + b_1 \cdot B^1 + b_0 \cdot B^0 \\ + b_{-1} \cdot B^{-1} + b_{-2} \cdot B^{-2} + \dots + b_{-m} \cdot B^{-m}$$

Numeri in virgola fissa

- Nella rappresentazione binaria in virgola fissa (*“fixed point”*),

si utilizza un numero prestabilito di bit per rappresentare la parte intera, e un numero prestabilito di bit per rappresentare la parte frazionaria

(notazione in modulo e segno o notazione in complemento per i numeri con segno)

Numeri in virgola fissa

■ Esempio:

rappresentiamo il numero decimale 72,6
utilizzando 12 bit, di cui 4 riservati alla parte frazionaria

$$72,6_{10} = \underbrace{0100\ 1000}_{\text{parte intera}} \underbrace{1001}_2$$

parte intera

parte frazionaria

si ottiene col metodo
delle divisioni
successive

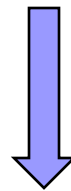
si ottiene col metodo
delle moltiplicazioni
successive

Numeri in virgola fissa

- Esempio:

rappresentiamo ora il numero decimale -72,6
utilizzando sempre 12 bit, di cui 4 per la parte frazionaria

$$72,6_{10} = 0100\ 1000\ 1001_2$$



complemento a 2

$$-72,6_{10} = 1011\ 0111\ 0111_2$$

Numeri in virgola mobile

- Nei problemi di calcolo tecnico e scientifico, i numeri vengono di solito espressi come prodotto di due fattori:
 - uno comprende le cifre significative del numero
 - l'altro è una potenza del 10
- Esempi:
 - $127000000 = 127 \cdot 10^6$
 - $0,0000015 = 15 \cdot 10^{-7}$

Numeri in virgola mobile

- In generale, un dato numerico ammette una rappresentazione approssimata del tipo:

$$\pm x_{n-1} \dots x_1 x_0 , y_{-1} y_{-2} \dots y_{-m} \cdot B^{\pm a_{k-1} \dots a_1 a_0}$$

dove:

- B è la base del sistema di numerazione
- $x_{n-1} \dots x_1 x_0 , y_{-1} y_{-2} \dots y_{-m} , a_{k-1} \dots a_1 a_0$ sono cifre dello stesso sistema

Numeri in virgola mobile

- In tale rappresentazione:

$$\pm x_{n-1} \dots x_1 x_0 , y_{-1} y_{-2} \dots y_{-m} \cdot B^{\pm a_{k-1} \dots a_1 a_0}$$

- il numero $\pm x_{n-1} \dots x_1 x_0 , y_{-1} y_{-2} \dots y_{-m}$
è detto ***mantissa***
- il numero $\pm a_{k-1} \dots a_1 a_0$
è detto ***esponente*** o ***caratteristica***

Numeri in virgola mobile

- Riprendendo gli esempi precedenti:

- $127000000 = 127 \cdot 10^6$



base = 10, mantissa = 127,
esponente (o caratteristica) = 6

- $0,0000015 = 15 \cdot 10^{-7}$



base = 10, mantissa = 15,
esponente (o caratteristica) = -7

Rappresentazione esponenziale normalizzata

- Tipicamente si preferisce rappresentare tutti i numeri in una stessa forma (*normalizzazione*), ad esempio con la prima cifra significativa immediatamente a destra della virgola.

- Esempi:

$$\square 127000000 = 127 \cdot 10^6 \quad \longrightarrow \quad 0,127 \cdot 10^9$$

$$\square 0,0000015 = 15 \cdot 10^{-7} \quad \longrightarrow \quad 0,15 \cdot 10^{-5}$$

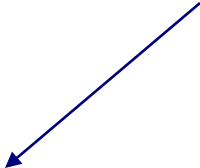
Ulteriori convenzioni

- Non si rappresentano i caratteri non necessari (lo zero che indica la parte intera della mantissa, la virgola, il segno di prodotto, il valore della base).
- La lunghezza della mantissa è fissata (costante).
- I valori dell'esponente, limitati entro un opportuno intervallo, vengono *polarizzati*: si somma all'esponente effettivo una costante (*bias*) al fine di rendere l'esponente da rappresentare sempre positivo, eliminando quindi la necessità di memorizzarne il segno.

Ulteriori convenzioni

- Nell'ordine si rappresentano:

s	esp	M
<i>(segno)</i>	<i>(esponente)</i>	<i>(mantissa)</i>



esponente *polarizzato*,
con valori compresi in
un intervallo predefinito



mantissa
di lunghezza
costante

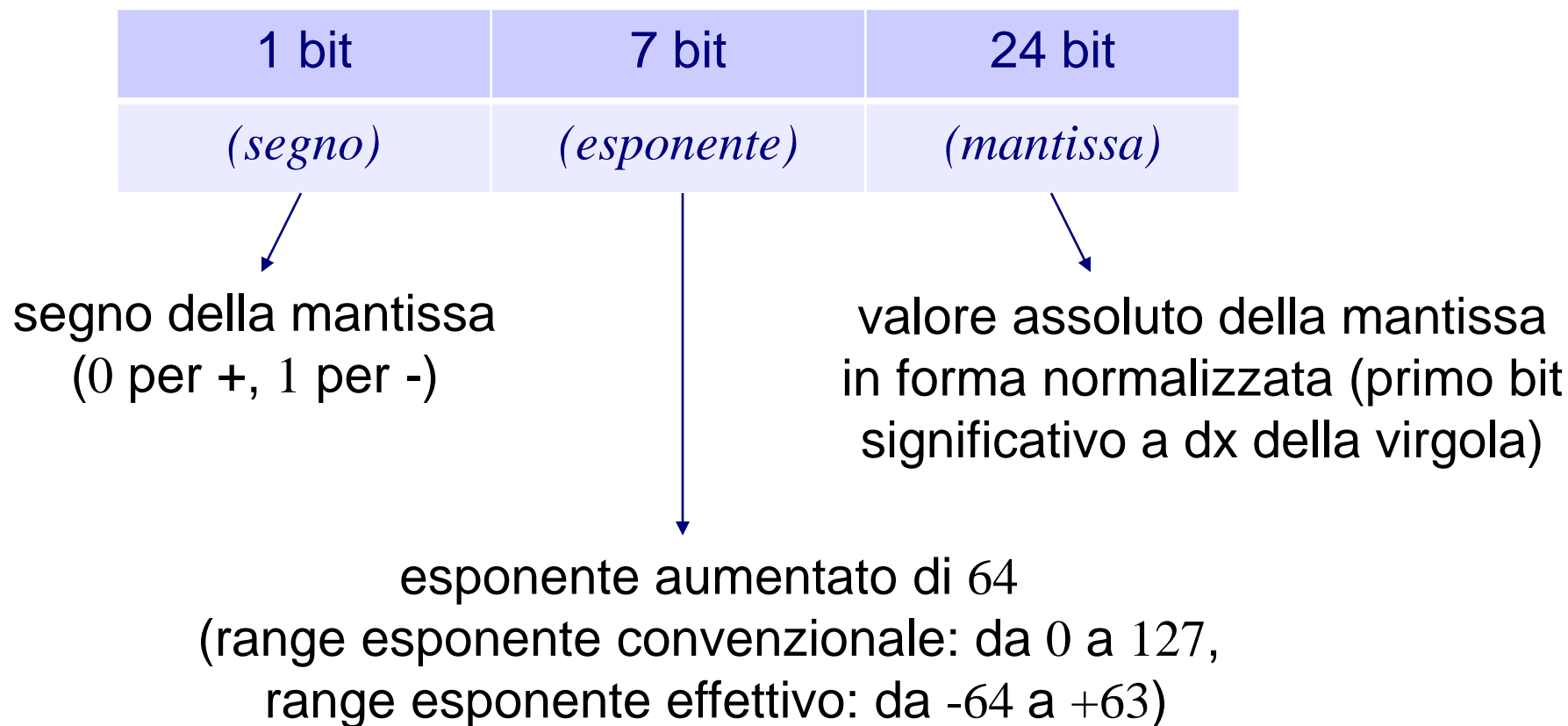
Esempio

- Assumiamo:
 - lunghezza mantissa: 8 cifre
 - valore effettivo dell'esponente: da -50 a +49
 - costante di polarizzazione: 50

	s	esp	M
$0,127 \cdot 10^9$	+	59	12700000
$0,15 \cdot 10^{-5}$	+	45	15000000

Rappresentazione floating point in binario

- Esempio di rappresentazione su 32 bit:



Rappresentazione floating point in binario

- Con tali convenzioni, il numero $204,17437_{10}$ sarebbe rappresentato come segue:

- rappresentazione binaria:

1100 1100, 0010 1100 1010 0011

- rappresentazione binaria normalizzata:

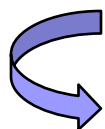
0, 1100 1100 0010 1100 1010 0011 · 10^{1000}



{
bit di segno: 0
esponente effettivo: 0001000
mantissa: 1100 1100 0010 1100 1010 0011

Rappresentazione floating point in binario

- All'esponente effettivo va poi sommata la costante di polarizzazione 64_{10} (ovvero 1000000_2)



Si ottiene quindi
la seguente rappresentazione:

1 bit	7 bit	24 bit
<i>0</i>	<i>1001000</i>	<i>1100 1100 0010 1100 1010 0011</i>

Osservazioni

- La lunghezza della mantissa definisce il numero di cifre significative rappresentabili cioè la *precisione*.
- Quando il valore dell'esponente è maggiore del massimo esponente consentito si verifica *overflow*.
- Quando invece il valore dell'esponente è minore del minimo consentito si verifica *underflow*.

(Entrambi i fenomeni di *overflow* e *underflow* possono verificarsi quando si eseguono le operazioni aritmetiche sui numeri *floating point*, e i risultati che si ottengono sono ovviamente privi di significato)

Osservazioni

- Non esiste una convenzione per la rappresentazione *floating point* che sia universalmente adottata da tutte le case costruttrici di elaboratori.
- Nel 1985 l'associazione IEEE ha definito uno standard di riferimento (*IEEE 754-1985*) che è andato progressivamente affermandosi.
- Nel 2008 è stata poi pubblicata una nuova versione dello standard che estende quella precedente (*IEEE 754-2008*).

Standard IEEE 754

- Quattro formati base:

- *mezza precisione* (*parola di 16 bit*)
- *precisione singola* (*parola di 32 bit*)
- *precisione doppia* (*parola di 64 bit*)
- *precisione quadrupla* (*parola di 128 bit*)

- Sono inoltre previsti ulteriori formati che estendono quelli base (*formati estesi*).

Formati base

- In riferimento a parole di 16 / 32 / 64 / 128 bit:

1 bit	5 / 8 / 11 / 15 bit	10 / 23 / 52 / 112 bit
s (segno)	exp (esponente)	M (mantissa)

↓
0 per +, 1 per -

↓
 $exp = E + bias$

↓
*normalizzata in modo che la parte intera
sia sempre 1: il corrispondente bit non
viene rappresentato (hidden bit)*

Formati base

- In caso di mezza precisione:

- l'esponente effettivo E varia tra -14 e 15
- la costante di polarizzazione (*bias*) vale 15
- l'esponente polarizzato exp varia tra 1 e 30

➡ Casi particolari:

- $E = -15$ ovvero $exp = 0$ (± 0 e denormalizzati)
- $E = 16$ ovvero $exp = 31$ ($\pm\infty$ e NaN)

Formati base

- In caso di precisione singola:

- l'esponente effettivo E varia tra -126 e 127
- la costante di polarizzazione (*bias*) vale 127
- l'esponente polarizzato exp varia tra 1 e 254

➡ Casi particolari:

- $E = -127$ ovvero $exp = 0$ (± 0 e denormalizzati)
- $E = 128$ ovvero $exp = 255$ ($\pm\infty$ e NaN)

Formati base

- In caso di precisione doppia:

- l'esponente effettivo E varia tra -1022 e 1023
- la costante di polarizzazione (*bias*) vale 1023
- l'esponente polarizzato exp varia tra 1 e 2046

➡ Casi particolari:

- $E = -1023$ ovvero $exp = 0$ (± 0 e denormalizzati)
- $E = 1024$ ovvero $exp = 2047$ ($\pm\infty$ e NaN)

Formati base

- In caso di precisione quadrupla:

- l'esponente effettivo E varia tra -16382 e 16383
- la costante di polarizzazione (*bias*) vale 16383
- l'esponente polarizzato exp varia tra 1 e 32766

➡ Casi particolari:

- $E = -16383$ ovvero $exp = 0$ (± 0 e denormalizzati)
- $E = 16384$ ovvero $exp = 32767$ ($\pm\infty$ e NaN)

Formati base

- E compreso tra E_{min} e E_{max} (*numeri normalizzati*):

$$v = (-1)^s \cdot (1, M) \cdot 2^E = (-1)^s \cdot (1, M) \cdot 2^{exp - bias}$$

- $E = E_{min} - 1$ e $M \neq 0$ (*numeri denormalizzati*):

$$v = (-1)^s \cdot (0, M) \cdot 2^{E_{min}}$$



valori più bassi del più piccolo
numero normalizzato

Formati base

- $E = E_{min} - 1$ e $M = 0$:

$$v = \pm 0$$

- $E = E_{max} + 1$ e $M = 0$:

$$v = \pm \infty$$

- $E = E_{max} + 1$ e $M \neq 0$:

$$v = \text{NaN}$$



Not-a-Number

Esempio

- Rappresentiamo il numero 5_{10} secondo lo standard IEEE 754, nel formato in singola precisione (ovvero su una parola di 32 bit):

$$5_{10} = 101_2 = (1, 01 \cdot 10^{10})_2$$

s (1 bit)	exp (8 bit)	M (23 bit)
0	10000001	01000000000000000000000000



all'esponente effettivo va sommata
la costante di polarizzazione (127_{10})

Esempio

- Rappresentiamo ora il numero $12,375_{10}$ secondo lo standard IEEE 754, nel formato in singola precisione (ovvero su una parola di 32 bit):

$$12,375_{10} = 1100,011_2 = (1, 100011 \cdot 10^{11})_2$$

s (1 bit)	exp (8 bit)	M (23 bit)
0	10000010	100011000000000000000000

↑
all'esponente effettivo va sommata
la costante di polarizzazione (127_{10})

Esempio

- Rappresentiamo ora il numero -2_{10} secondo lo standard IEEE 754, nel formato in mezza precisione (ovvero su una parola di 16 bit):

$$2_{10} = 10_2 = (1,0 \cdot 10^1)_2 \quad (\text{valore assoluto})$$

s (1 bit)	exp (5 bit)	M (10 bit)
<i>1</i>	<i>10000</i>	<i>0000000000</i>



all'esponente effettivo va sommata
la costante di polarizzazione (15_{10})

Esempio

- A quale numero decimale corrisponde la seguente rappresentazione *floating point* IEEE 754?

10111111111000000000000000000000



sequenza di 32 bit da interpretare
secondo lo schema:

s (1 bit)	exp (8 bit)	M (23 bit)
1	01111111	11000000000000000000000000

Esempio

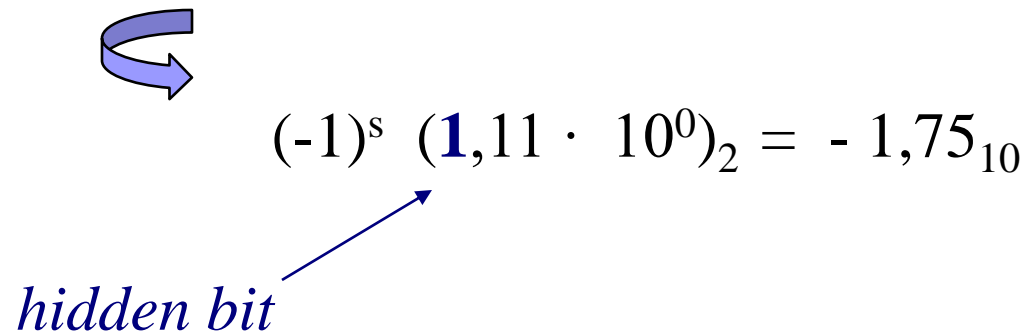
■ Abbiamo quindi:

□ $s = 1$ (numero negativo)

□ $\text{exp} = 01111111$

$$E = \text{exp} - \text{bias} = 01111111 - 01111111 = 0$$

□ $M = 1100000000000000000000000000$


$$(-1)^s (1,11 \cdot 10^0)_2 = -1,75_{10}$$

hidden bit

Operazioni in virgola mobile

- Diamo solo un cenno a come vengono eseguite, sui numeri in virgola mobile, le operazioni di:
 - somma/sottrazione
 - moltiplicazione
 - divisione

Somma/sottrazione

- È richiesta l'uguaglianza degli esponenti dei due operandi (a tal fine è necessario traslare opportunamente una mantissa rispetto all'altra).
- Passi:
 1. La mantissa del numero con l'esponente minore viene traslata a destra per un numero di bit pari alla differenza degli esponenti (in modo da rendere questi ultimi uguali).



Somma/sottrazione

2. Si pone l'esponente del risultato uguale all'esponente degli operandi.
3. Si effettua l'addizione o la sottrazione delle mantisse e si determina il segno del risultato.
4. Se necessario, si normalizza il risultato.

Moltiplicazione

■ Passi:

1. Si sommano gli esponenti e si sottrae la costante di polarizzazione (che viene raddoppiata nella somma).
2. Si moltiplicano le mantisse e si determina il segno del risultato.
3. Se necessario, si normalizza il risultato.

Divisione

■ Passi:

1. Si sottraggono gli esponenti e si somma la costante di polarizzazione.
2. Si dividono le mantisse e si determina il segno del risultato.
3. Se necessario, si normalizza il risultato.