Laboratorio di
Architetture degli Elaboratori I

Corso di Laurea in Informatica
A.A. 2024-2025
Università degli Studi di Milano

Turno A: Gabriella Trucco, Cognomi A-E, gabriella.trucco@unimi.it
Turno B: Massimo W. Rivolta, Cognomi F-M, massimo.rivolta@unimi.it
Turno C: Matteo Re, Cognomi N-Z, matteo.re@unimi.it

Riferirsi al sito Ariel del docente di teoria per i dettagli organizzativi

Informazioni

- Lezioni turno A: martedì 13:30 – 16:30
- Sito web del corso:

  https://myariel.unimi.it/course/view.php?id=3141
- Esame: prova in laboratorio
- Ricevimento: su appuntamento
- Materiale didattico: slide messe a disposizione sul sito web del corso

Calendario delle lezioni

Data	Argomento
8-10-2024	Presentazione del corso. Rappresentazione dei numeri
22-10-2024	Esercizi di logica Booleana. Introduzione a Logisim
29-10-2024	SOP, POS, cammino critico, mappe di Karnaugh
5-11-2024	Circuiti combinatori
12-11-2024	Memorie, LFSR
19-11-2024	Macchine a stati finiti, contatori
3-12-2024	Esercizi di riepilogo
10-12-2024	Esempio di tema d’esame

Introduzione alla rappresentazione dei numeri

Rappresentazione dei numeri: notazione posizionale

- Base B: numero di simboli usati per rappresentare i numeri nel sistema posizionale.
  - $B = 10$, simboli $\{0, 1, \ldots, 9\}$
  - $B = 16$, simboli $\{0, 1, \ldots, 9, A, B, C, D, E, F\}$
- Notazione posizionale:
  - ogni simbolo ha una posizione, descritta con un intero i
  - ad ogni posizione i si associa un peso p_i
  - al simbolo in posizione i viene associato il valore dato da:

$$\left[\text{VALORE DEL SIMBOLO IN POSIZIONE } i\right] \times p_i$$

  - di solito $p_i = B^i$
  - più è alto il peso associato a un simbolo, più significativo è quel simbolo

Rappresentazione dei numeri: notazione posizionale

Esempio: $(147)_{10}$

cifre	1	4	7
posizioni i	2	1	0
pesi p_i	10^2	10^1	10^0

$$(147)_{10} = 1 \cdot 10^2 + 4 \cdot 10^1 + 7 \cdot 10^0$$

In generale, dato un N intero non negativo scritto con n cifre come $(c_{n-1}c_{n-2}\ldots c_0)_B$, il valore rappresentato è:

$$\sum_{i=0}^{n-1} c_i \cdot B^i = c_0 \cdot B^0 + c_1 \cdot B^1 + \ldots + c_{n-2} \cdot B^{n-2} + c_{n-1} \cdot B^{n-1}$$

Da base B a base decimale

- PROBLEMA: Scrivere in base $B = 10$ un numero dato in base $B \neq 10$ ($B = 2$ o $B = 16$)
- SOLUZIONE: applicare il metodo **polinomiale**

Esempio 1:

$$(1010)_2 = (10)_{10}$$

Esempio 2:

$$(3AC)_{16} = (940)_{10}$$

Da base decimale a base B

- PROBLEMA: Scrivere in base $B \neq 10$ ($B = 2$ o $B = 16$) un numero dato in base $B = 10$
- SOLUZIONE: applicare il metodo **iterativo** (divisioni)

Procedimento

Abbiamo un numero $(N)_{10}$ da convertire nella base B:

1. dividere N per B (con una divisione intera);
2. il resto della divisione diventa la prima cifra meno significativa che resta da calcolare del numero in base B;
3. se il quoziente è 0 abbiamo finito;
4. se il quoziente è diverso da zero si torna al passo **1** considerando il quoziente come dividendo;

Da base decimale a base B

Esempio 1: $(13)_{10} = (?)_2$

$$\left.\begin{array}{llll} 13 : 2 = 6 & \text{resto} = 1 & \text{LSD} \\ 6 : 2 = 3 & \text{resto} = 0 & \\ 3 : 2 = 1 & \text{resto} = 1 & \\ 1 : 2 = 0 & \text{resto} = 1 & \text{MSD} \end{array}\right| \quad (13)_{10} = (1101)_2$$

Esempio 2: $(4021)_{10} = (?)_{16}$

Da base B = 2 a base B = 16 e vice versa

- PROBLEMA: Scrivere in base $B = 2$ ($B = 16$) un numero dato in base $B = 16$ ($B = 2$)
- SOLUZIONE: **raggruppamento** dei simboli e **lookup table**

Esempio 1: $(111001)_2 = (?)_{16}$
$16 = 2^4$, raggruppo i bit partendo da destra a gruppi di 4:

$$0011 \mid 1001$$

Con 4 bit ho 2^4 valori, uno per ogni simbolo della base $B = 16$.

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

Ispezionando la tabella:

$$(0011 \mid 1001)_2 = (39)_{16}$$

Da base B = 2 a base B = 16 e vice versa

Esempio 2: $(EA0)_{16} = (?)_2$

$$E \mid A \mid 0$$

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

$$(E \mid A \mid 0)_{16} = (1110\ 1010\ 0000)_2$$

Somma di interi non negativi

Come la somma in decimale, ricordare che quando sommiamo singole cifre binarie:

$$
\begin{array}{rl}
0+0=0 & \text{riporto} = 0 \\
0+1=1 & \text{riporto} = 0 \\
1+0=1 & \text{riporto} = 0 \\
1+1=0 & \text{riporto} = 1 \\
1+1+1=1 & \text{riporto} = 1
\end{array}
$$

Esempio 1: $(1101)_2 + (111)_2$
Esempio 2: $(111010)_2 + (1101110010)_2$

Interi con numero finito di bit

1	0	1

n=3

MSB		LSB
s_2	s_1	s_0
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

$2^3=8$ simboli

- Le architetture degli elaboratori lavorano con un numero finito di bit
- Dati n bit:
  - Ci sono 2^n simboli
  - I numeri positivi rappresentabili risiedono nell'intervallo $[0, 2^n-1]$

Interi: complemento a 2

- Rappresentazione in **complemento a 2** (C2)
  - Dati n bit, un numero positivo N è rappresentato in modo standard (come abbiamo visto per i non negativi)
  - $-N$, invece si rappresenta come $2^n - N$
- Metodo operativo per rappresentare $-N$:
  - Rappresentare il modulo N in modo standard
  - Complementare a 1 tutti i bit ($1 \leftarrow 0$, $0 \leftarrow 1$)
  - Sommare 1

Interi: complemento a 2

- In complemento a 2, con n bit, possiamo rappresentare gli interi nell'intervallo: $[-2^{n-1}; 2^{n-1} - 1]$
- Esempio: con 3 bit rappresentiamo i numeri in ... $[-4; 3]$

$N_{(10)}$	$N_{(C2)}$
-4	100
-3	101
-2	110
-1	111

$N_{(10)}$	$N_{(C2)}$
0	000
$+1$	001
$+2$	010
$+3$	011

- Il primo bit indica ancora il segno
- Lo zero ha una sola codifica

Interi: complemento a 2

Esempio 1: $(-18)_{10} = (?)_{C2}$
Esempio 2: $(9)_{10} = (?)_{C2}$
Esempio 3: $(-6)_{10} = (?)_{C2}$

Somma di interi

- Rappresentare i numeri in C2
- Effettuare la somma in modo standard
- Non considerare l'eventuale riporto oltre il bit di segno

Esempio 1: $(60)_{10} - (54)_{10} = (60)_{10} + (-54)_{10}$

Somma di interi: overflow

- Sommiamo due interi in C2 rappresentati con n bit, quindi appartenenti a $[-2^{n-1}; 2^{n-1} - 1]$
- Può succedere che il risultato cada al di fuori dell'intervallo
- Overflow: n bit in C2 bastano per rappresentare gli operandi, ma non per rappresentare il risultato
- Come riconoscerlo?
  - può succedere solo quando si sommano due operandi dello stesso segno: **se il segno del risultato è diverso da quello degli operandi** è avvenuto un overflow
  - **gli ultimi due riporti sono diversi** tra loro (01 o 10)

Somma di interi: esempio di overflow

Esempio: $(100)_{C2} + (101)_{C2}$

$$
\begin{array}{lcccccc}
\text{riporto} & 1 & 0 & & & \\
 & & 1 & 0 & 0 & + \\
 & & 1 & 0 & 1 & = \\
\hline
\text{somma} & (1) & 0 & 0 & 1 &
\end{array}
$$

- Sto sommando su 3 bit -4 e -3, il risultato sarebbe: $-7 < -2^{3-1}$, non rappresentabile in C2 su 3 bit.

Somma di interi: esempi

Esempio 1: $-(1101)_2 - (111)_2$
Esempio 2: $-(1101)_2 - (111)_2$ Stesso esempio di prima, ma con l'aggiunta di un bit: anziché lavorare su 5 bit, ne considero 6. Come cambia il risultato?

Interi: complemento a 2

Come passare da C2 a base 10?

- Procedimento inverso:
  - Sottrarre 1
  - Complementare a 1
  - Convertire da binario a decimale e aggiungere il segno
- Metodo facilitato di verifica:
  - convertire in decimale con l'algoritmo standard assegnando al bit più significativo peso negativo
  - Esempio: $(10100)_{C2} = -1 \times 2^4 + 1 \times 2^2 = (-12)_{10}$

Numeri frazionari

- Notazione posizionale e metodo polinomiale si estendono facilmente:

$$(I.c_{-1}c_{-2}\dots c_{-m})_B =$$
$$I + c_{-1}\cdot B^{-1} + c_{-2}\cdot B^{-2} + \dots + c_{-m}\cdot B^{-m} =$$
$$I + \sum_{i=-m}^{-1} c_i \cdot B^i$$

- Esempio: $(0.587)_{10} = 5\cdot 10^{-1} + 8\cdot 10^{-2} + 7\cdot 10^{-3}$
- Conversione da base 2 a base 10:
  $(0.1011)_2 = 1\cdot 2^{-1} + 0\cdot 2^{-2} + 1\cdot 2^{-3} + 1\cdot 2^{-4} = (0.6875)_{10}$

Rappresentazione dei numeri frazionari

- Come convertire la parte frazionaria da base 10 a base 2? Metodo **iterativo** (moltiplicazioni)

Procedimento

Abbiamo un numero $(I.F)_{10}$ da convertire nella base $B = 2$:

1. la parte intera I si converte come abbiamo visto in precedenza;
2. moltiplicare $.F$ per 2;
3. la parte intera del risultato diventa la cifra **più** significativa (la prima che resta da calcolare) del numero in base 2;
4. tornare a **2** considerando la parte frazionaria del risultato ottenuto al posto di $.F$

- Quando si finisce?
  - La parte frazionaria ottenuta è 0 (solo i numeri scrivibili come $N(2^{-z})$, con N e z interi, possono essere rappresentati su un numero finito di bit)
  - Il numero di bit ottenuti costituisce un'approssimazione che si ritiene sufficiente

Rappresentazione dei numeri frazionari

Esempio: $(0.587)_{10} = (?)_2$

$$
\begin{array}{lll}
0.587 \times 2 = 1.174 & \text{parte intera} = 1 & \text{MSD} \\
0.174 \times 2 = 0.348 & \text{parte intera} = 0 & \\
0.348 \times 2 = 0.696 & \text{parte intera} = 0 & \\
0.696 \times 2 = 1.392 & \text{parte intera} = 1 & \\
0.392 \times 2 = 0.784 & \text{parte intera} = 0 & \\
0.784 \times 2 = 1.568 & \text{parte intera} = 1 & \\
\quad \ldots & &
\end{array}
$$

$$(0.587)_{10} = (0.100101\ldots)_2$$

Rappresentazione approssimata dei numeri reali

Rappresentazione in **virgola fissa**

- La parte intera è rappresentata sempre su n bit mentre la parte frazionaria è rappresentata sempre su m bit
- La virgola può cadere in un'unica posizione, essendo implicita può essere omessa

Esempi: assumiamo base $B = 2$, $m = n = 4$ e notazione in $C2$

$(+4.25)_{10} = (0100.0100)_{C2}$ $(-4.25)_{10} = (1100.0100)_{C2}$

$(+3.35)_{10} = (0011.0101)_{C2}$ $(-3.35)_{10} = (1101.0101)_{C2}$

Rappresentazione approssimata dei numeri reali

Rappresentazione in **virgola mobile**

- Un numero razionale $(N)_B$ è espresso come:

$$N = (-1)^s \cdot m \cdot B^e$$

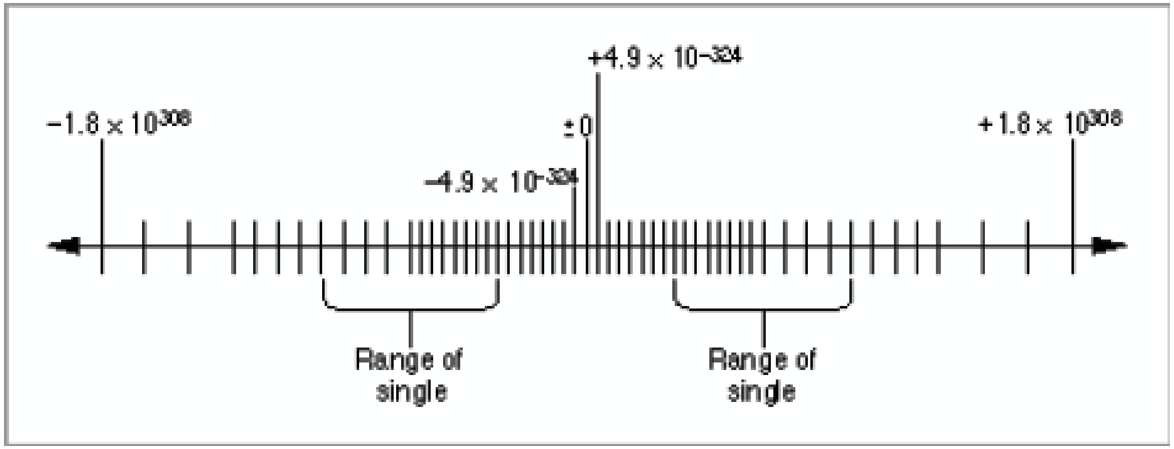
  - s è il **segno**, $(0 \rightarrow +, 1 \rightarrow -)$
  - m è la **mantissa**, numero frazionario su p bit in base B
  - e è l'**esponente**, numero intero
- Forma **normalizzata**: la parte intera della mantissa ha una cifra significativa, es. $5.79 \cdot 10^2$
- In base 2: $\pm 1.b_{-1}b_{-2}\dots b_{-p} \cdot 2^e$, rappresentiamo:
  - Segno: 0, 1
  - Mantissa: la parte frazionaria $b_{-1}b_{-2}\dots b_{-p}$ (la parte intera è implicita)
  - Esponente: e

Approssimare i numeri reali

Il numero: $\pm 1.b_{-1}b_{-2}\dots b_{-p} \cdot 2^e$ è associato al valore:

$$\pm(1 + b_{-1} \cdot 2^{-1} + b_{-2} \cdot 2^{-2} \cdots b_{-p} \cdot 2^{-p}) \cdot 2^e$$

Risoluzione variabile: il contributo del bit meno significativo (b_{-p}) dipende da e.



Virgola fissa e virgola mobile

Come confrontare le due diverse **rappresentazioni** ($R_{N,VF}$ e $R_{N,VM}$)?

- Minimo e massimo numero rappresentabile V_{min} e V_{max}

Dato un numero x che si vuole rappresentare:

- Errore assoluto $\epsilon_A(x)$: è la differenza tra il numero x e la sua più vicina rappresentazione
- Errore relativo $\epsilon_R(x)$: è la differenza tra il numero x e la sua più vicina rappresentazione in percertuale del valore x: $\frac{\epsilon_A(x)}{x}$

Virgola fissa	Virgola mobile
$R_{N,VF} = iii.fff$	$R_{N,VM} = i.fff \cdot 10^{ee}$
$V_{min} = 000.000,\ V_{max} = 999.999 \cong 10^3$	$V_{min} = 0,\ V_{max} = 9.999 \cdot 10^{99} \cong 10^{100}$
$\epsilon_A \leq 10^{-3}$	$\epsilon_A \leq 10^{-3} \cdot 10^{ee} \leq 10^{ee-3}$
$\epsilon_R \in \left[\frac{10^{-3}}{10^3} = 10^{-6}, \frac{0.001}{0}\right)$	$\epsilon_R = \frac{10^{ee-3}}{m \cdot 10^{ee}} \cong 10^{-3}$

- A parità di cifre utilizzate (nell'esempio sono 6), la rappresentazione in virgola mobile è in grado di rappresentare un numero più elevato di valori
- In virgola fissa: precisione assoluta costante e relativa variabile
- In virgola mobile: precisione assoluta variabile con N e relativa approssimativamente costante

Numeri reali: lo standard IEEE 754

- IEEE Stardard for Floating Point Arithmetic (IEEE 754), formato a precisione *singola*: 32 bit

s	E (8 bit)								m (23 bit)																						
31	30							23	22																						0

Quando rappresentiamo un **numero normalizzato**:

- i bit da 0 a 22 sono la mantissa i.e., la parte frazionaria del numero (la parte intera si intende implicitamente pari a 1)
- i bit da 23 a 30 sono l'esponente: un intero tra -126 e 127 memorizzato su 8 bit in eccesso 127 (significa che memorizziamo l'intero positivo $E = e + 127$ anzichè e)
- il bit 31 è il segno: 0 corrisponde a $+$, 1 corrisponde a $-$

$$R_{N,VM} = \pm I.F \cdot B^E$$

Esponente: rappresentazione in eccesso 127

- Per rappresentare il numero $e_{(10)}$ si somma ad esso $2^{8-1} - 1 = 127_{(10)}$
- Si rappresenta il valore risultante $E_{(10)} = e_{(10)} + 127$
- Intervallo rappresentato (numeri normalizzati): $-126 \leq e_{(10)} \leq 127 \quad \rightarrow \quad 0 < E_{(10)} < 255$

Esempio 1: $(-34)_{(10)} = (?)_{8bit,e127}$
$e_{(10)} = -34, E_{(10)} = -34 + 127 = +93_{(10)} = 01011101_{(2)}$

Esempio 2: $(11100101)_{8bit,e127} = (?)_{(10)}$
$E_{(10)} = +229, e_{(10)} = 229 - 127 = 102_{(10)}$

Mantissa

- La mantissa M viene rappresentata nella forma $1.c_{-1}c_{-2}\dots c_{-23}$
- Il bit c_0 corrisponde al peso $2^0 = 1$: è, per convenzione, sempre uguale a 1 e non si rappresenta
- Il punto decimale segue sempre il bit c_0 e, per convenzione, non si rappresenta; questo corrisponde alla **rappresentazione normalizzata** (il punto decimale è posto dopo l'unica cifra significativa della parte intera)
- i 23 bit di M rappresentano quindi l'intervallo $[1, 2)$

Numeri reali: lo standard IEEE 754

Esempio 1: convertiamo $(17.375)_{10}$, dobbiamo determinare s, m, E

- Il numero è positivo $\rightarrow s = 0$
- Converto la parte intera in binario: $(17)_{10} = (10001)_2$
- Converto la parte frazionaria $(.375)_{10}$ $\left(= \frac{3}{2^3}\right)$

$$\left.\begin{array}{rll} 0.375 \cdot 2 = 0.75 & \text{parte intera} = 0 & \text{MSD} \\ 0.75 \cdot 2 = 1.5 & \text{parte intera} = 1 & \\ 0.5 \cdot 2 = 1.0 & \text{parte intera} = 1 & \end{array}\right| \quad (.375)_{10} = (.011)_2$$

- Unisco i risultati: $(10001.011)_2$
- Normalizzazione: $(1.0001011)_2 \cdot (10)_2^4$
- Mantissa: $m = 0001\ 0110\ 0000\ 0000\ 0000\ 000$
- Esponente: $E = e + 127 = (4)_{10} + (127)_{10} = (131)_{10} = (10000011)_2$

Numeri reali: lo standard IEEE 754

Esempio 2: convertiamo $(-0.8)_{10}$, dobbiamo determinare s, m, E

- Il numero è negativo $\rightarrow s = 1$
- Converto la parte intera in binario: $(0)_{10} = (0)_2$
- Converto la parte frazionaria $(.8)_{10}$

$$\begin{array}{lll}
\mathbf{0.8} \cdot 2 = 1.6 & \text{parte intera} = 1 & \text{MSD} \\
0.6 \cdot 2 = 1.2 & \text{parte intera} = 1 & \\
0.2 \cdot 2 = 0.4 & \text{parte intera} = 0 & \\
0.4 \cdot 2 = \mathbf{0.8} & \text{parte intera} = 0 & \\
0.8 \cdot 2 = \ldots & \text{la sequenza 1100 si ripete} &
\end{array}$$

- Unisco i risultati: $(0.8)_{10} = (0.\overline{1100})_2$
- Normalizzazione: $(0.\overline{1100})_2 = (0.1100\,\overline{1100})_2 = (1.100\,\overline{1100})_2 \cdot (10)_2^{-1}$
- Mantissa: $m = 100\ 1100\ 1100\ 1100\ 1100\ 1100$
- Esponente:
  $E = e + 127 = (-1)_{10} + (127)_{10} = (126)_{10} = (1111110)_2 \rightarrow (01111110)_2$

Numeri reali: lo standard IEEE 754

Certi valori di m e di e sono utilizzati secondo diverse convenzioni, definite dallo standard ($E = e + 127$)

- Se $0 < E < 255$ ($e \in [-126, 127]$)$\rightarrow$ numero normalizzato
- Se $m = 0$, $E = 0 \rightarrow \pm 0$ (a seconda di s)
- Se $m = 0$, $E = 255 \rightarrow \pm\infty$ (a seconda di s)
- Se $m \neq 0$, $E = 255 \rightarrow$ NaN (Not a Number)
- Se $m \neq 0$, $E = 0 \rightarrow$ numero subnormalizzato

Standard IEEE 754: numeri subnormalizzati

Quale è il numero positivo normalizzato più piccolo che possiamo rappresentare?

- $s = 0,\ e = (-126)_{10} \to E = (0000\ 0001)_2,\ m = 0000\ldots000$
- $(1.0)_2 \cdot (10)_2^{-126} = 2^{-126} \approx 1.17 \cdot 10^{-38}$

Il successivo?

- $s = 0,\ e = (-126)_{10} \to (0000\ 0001)_2,\ m = 0000\ldots001$
- $(1.0\ldots01)_2 \cdot (10)_2^{-126} = 2^{-126} + 2^{-126-23} \approx 1.17 \cdot 10^{-38} + 1.4 \cdot 10^{-45}$

In prossimità dello zero non abbiamo risoluzione costante! Possiamo riempire questa regione in modo più "furbo" (precisione maggiore e localmente costante)? $\to$ I numeri subnormali hanno un ordine di grandezza minore di quello del più piccolo numero normalizzato

Standard IEEE 754: numeri subnormalizzati

- Ad E viene assegnato il codice 0000 0000 che si interpreta come esponente pari a -126 (non -127 !)
- la mantissa è un numero diverso da 0 e la parte intera, diversamente dai normalizzati, è implicitamente assunta essere 0
- Quindi un numero subnormalizzato è sempre fatto così :

$$(-1)^{s} \cdot (0.m) \cdot (10)_2^{-126}$$

Quale è il numero positivo subnormalizzato più piccolo che possiamo rappresentare?

- $s = 0$, $e = (-126)_{10}$ subnormalizzato $\rightarrow E = (0000\ 0000)_2$, $m = 0000 \ldots 001$
- $(0.0 \ldots 01)_2 \cdot (10)_2^{-126} = 2^{-23} \cdot 2^{-126} = 2^{-149} = 1.4012985 \cdot 10^{-45}$
- il successivo sarà ovviamente $2 \cdot 2^{-149} = 2.8025969 \cdot 10^{-45}$

In prossimità dello 0 abbiamo una precisione maggiore e localmente costante

Standard IEEE 754: numeri subnormalizzati

Quale è il numero positivo subnormalizzato più grande che possiamo rappresentare?

- $s = 0$, $e = (-126)_{10}$ subnormalizzato $\rightarrow E = (0000\ 0000)_2$, $m = 1111\ldots111$
- $(0.1\ldots11)_2 \cdot (10)_2^{-126} = (2^{-23} + 2^{-22} + \ldots + 2^{-1}) \cdot 2^{-126} = (1 - 2^{-23}) \cdot 2^{-126} = 1.1754942 \cdot 10^{-38}$
- il precedente sarà ovviamente $(1 - 2^{-22}) \cdot 2^{-126} = 1.1754941 \cdot 10^{-38}$

Anche qui abbiamo una precisione localmente costante

Standard IEEE 754: numeri subnormalizzati

Esempio 1: rappresentare in formato IEEE 754 il numero $-(0.125)_{10} \cdot 2^{-125}$

- Il numero è negativo $\rightarrow s = 1$
- Il numero è esprimibile come un subnormalizzato? Per esserlo devo poterlo scrivere come $(0.m) \cdot 2^{-126}$ (m ha 23 bit)
- $0.125 \cdot 2^{-125} = 0.125 \cdot 2^{1} \cdot 2^{-126} = 0.25 \cdot 2^{-126}$, quindi posso scriverlo come subnormalizzato
- converto 0.25 $\left(\frac{1}{2^2}\right)$ in base 2: $(0.25)_{10} = (0.01)_2$
- Mantissa $m = 01\,0\ldots0$
- Esponente $E = 0000\,0000$