Sperimentazioni di Fisica I mod. A – Lezione 3

Numeri Reali

Dipartimento di Fisica e Astronomia "G. Galilei",
Università degli Studi di Padova

Rappresentazione dei Numeri

Lezione III: Numeri Reali

1. Rappresentazione e Cambiamento di Base

Numeri Razionali e Irrazionali

I **numeri razionali** q $\in$ **Q**, sono dati dal **rapporto** tra **due numeri interi**. La loro rappresentazione è indicata con un **numero finito di cifre** oppure con una **sequenza di cifre che si ripete periodicamente**.

$$\frac{11_{10}}{2_{10}} = 5.5_{10} = 5 \bullet 10^{0} + 5 \bullet 10^{-1}$$

$$\frac{11_{10}}{3_{10}} = 3.\bar{6}_{10} = 3 \bullet 10^{0} + 6 \bullet 10^{-1} + \ldots + 6 \bullet 10^{-n}$$

I **numeri irrazionali** r $\in$ (**R\Q**) **non** sono riconducibili al **rapporto** di due numeri interi, hanno un **numero infinito di cifre decimali** e **non** è possibile individuare una **sequenza periodica**. π e $\sqrt{2}$ sono numeri irrazionali.

Cambiamento di Base

Un numero reale viene indicato in base-R tramite la sequenza

$$\mathbf{q = (q_n\, q_{n-1} q_{n-2} \cdots q_2 q_1 q_0 , q_{-1} q_{-2} \cdots q_{-m})_R}$$

dove $q_i \in \{0, 1, \ldots R\text{-}1\}$,

q_n = 0, se q è positivo

q_n = 1, se q è negativo

$$q = (1 - 2q_n) \sum_{i=-m}^{n-1} q_i R^i$$

La procedura di codifica della **parte intera** di un numero reale segue la **stessa procedura ricorsiva** già considerata.

Prendiamo in esame la conversione della **parte frazionaria** di un numero reale da una base R_1 ad una nuova base R_2.

$$q = \sum_{i=1}^{n} a_{-i} R_1^{-i} = \sum_{j=1}^{m} b_{-j} R_2^{-j}$$

Formula Ricorsiva (I)

$$q = q_0 = b_{-1}R_2^{-1} + b_{-2}R_2^{-2} + b_{-3}R_2^{-3} + \ldots + b_{-m+1}R_2^{-m+1} + b_{-m}R_2^{-m}$$

$$= R_2^{-1}(b_{-1} + b_{-2}R_2^{-1} + \ldots + b_{-m+1}R_2^{-m+2} + b_{-m}R_2^{-m+1}) \quad \mathbf{q_0}$$

$$= R_2^{-1}(b_{-1} + R_2^{-1}(b_{-2} + \ldots + b_{-m+1}R_2^{-m+3} + b_{-m}R_2^{-m+2})) \quad \mathbf{q_{-1}}$$

$$\mathbf{q_{-2}}$$

$$= R_2^{-1}(b_{-1} + R_2^{-1}(b_{-2} + R_2^{-1}(b_{-3}\ldots + b_{-m+1}R_2^{-m+4} + b_{-m}R_2^{-m+3})))$$

Possiamo individuare una **formula ricorsiva**:

$$q_{-j+1} = R_2^{-1}(b_{-j} + q_{-j})$$

con $j = 1, 2, \ldots, m-1$ e $q_{-m+1} = b_{-m}R_2^{-1}$

$$q = q_0 = R_2^{-1}(b_{-1} + q_{-1})$$

$$q = q_0 = R_2^{-1}(b_{-1} + R_2^{-1}(b_{-2} + q_{-2}))$$

$$q = q_0 = R_2^{-1}(b_{-1} + R_2^{-1}(b_{-2} + R_2^{-1}(b_{-3} + q_{-3})))$$

$$q = q_0 = \sum_{j=1}^{m} b_{-j}R_2^{-j}$$

Formula Ricorsiva (II)

$$q_{-j+1} = R_2^{-1}(b_{-j} + q_{-j})$$

Moltiplicando ambo i membri **per R_2**, si individua un metodo ricorsivo per calcolare i **parametri b_{-j}**, partendo da **$q_0 = q$**.

$$q_{-j+1} \bullet R_2 = b_{-j} + q_{-j}$$

Parte Intera

Parte Frazionaria

I coefficienti di q nella nuova base, **b_{-j}**, si ottengono **moltiplicando la parte frazionaria iniziale**, $q_0 = q$, per la **nuova base R_2** e associando ai **coefficienti** la **parte intera** del prodotto. Il processo si arresta quando si ottiene un moltiplicando, i.e. una **parte frazionaria q_i, nullo**.

$$q_0 \bullet R_2 = q \bullet R_2 = b_{-1} + q_{-1}$$

Formula Ricorsiva (III)

$$q_0 \bullet R_2 = q \bullet R_2 = b_{-1} + q_{-1}$$

b_{-1} è la **parte intera** del prodotto **$q * R_2$**,
q_{-1} è la **parte frazionaria** del prodotto **$q * R_2$**.

$$q_{-1} \bullet R_2 = b_{-2} + q_{-2}$$

b_{-2} è la **parte intera** del prodotto **$q_{-1} * R_2$**,
q_{-2} è la **parte frazionaria** del prodotto **$q_{-1} * R_2$**.

Si itera il procedimento finché la **parte frazionaria q_m è nulla**.

Esempio

Convertiamo il numero **0.828125_{10}** in **base-2**:

operazione					parte intera	coefficienti
0.828125_{10}	$\times$	2_{10}	$=$	1.65625_{10}	1	$b_{-1} = 1$
0.65625_{10}	$\times$	2_{10}	$=$	1.3125_{10}	1	$b_{-2} = 1$
0.3125_{10}	$\times$	2_{10}	$=$	0.625_{10}	0	$b_{-3} = 0$
0.625_{10}	$\times$	2_{10}	$=$	1.25_{10}	1	$b_{-4} = 1$
0.25_{10}	$\times$	2_{10}	$=$	0.5_{10}	0	$b_{-5} = 0$
0.5_{10}	$\times$	2_{10}	$=$	1.0_{10}	1	$b_{-6} = 1$
$0._{10}$	$\times$	2_{10}	$=$	0.0_{10}		

0.110101_2

Convertiamo il numero **0.110101_2** in **base-10**:

operazione					parte intera	coefficienti
0.110101_2	$\times$	1010_2	$=$	1000.010010_2	1000_2	$b_{-1} = 8_{10}$
0.01001_2	$\times$	1010_2	$=$	10.1101_2	10_2	$b_{-2} = 2_{10}$
0.1101_2	$\times$	1010_2	$=$	1000.001_2	1000_2	$b_{-3} = 8_{10}$
0.001_2	$\times$	1010_2	$=$	1.01_2	1_2	$b_{-4} = 1_{10}$
0.01_2	$\times$	1010_2	$=$	10.1_2	10_2	$b_{-5} = 2_{10}$
0.1_2	$\times$	1010_2	$=$	101.0_2	101_2	$b_{-6} = 5_{10}$
$0._2$	$\times$	1010_2	$=$	0.0_2		

0.828125_{10}

Errori di Troncamento

Spesso **non** si riesce ad **esprimere** il numero nella nuova base con un **numero finito di cifre**. Per esempio, il numero $\mathbf{0.3_{10}}$ convertito in **base-2**:

operazione					parte intera	*coefficienti*
0.3_{10}	$\times$	2_{10}	$=$	0.6_{10}	0	$b_{-1} = 0$
0.6_{10}	$\times$	2_{10}	$=$	1.2_{10}	1	$b_{-2} = 1$
0.2_{10}	$\times$	2_{10}	$=$	0.4_{10}	0	$b_{-3} = 0$
0.4_{10}	$\times$	2_{10}	$=$	0.8_{10}	0	$b_{-4} = 0$
0.8_{10}	$\times$	2_{10}	$=$	1.6_{10}	1	$b_{-5} = 1$
0.6_{10}	$\times$	2_{10}	$=$	1.2_{10}	1	$b_{-6} = 1$
...	...	...	...	...	...	...

$$0.3_{10} = 0.0\overline{1001}_2$$

Solitamente si ha a disposizione **un numero limitato e finito di bit** per la rappresentazione. Occorre pertanto definire **la precisione** con cui rappresentare il numero nella nuova base.

Precisione

Nel **sistema binario**, fissato con **m il numero di bit** a disposizione per la rappresentazione della **parte frazionaria**, il **numero più piccolo rappresentabile è**

$$\mathbf{x = \pm\ 0.0\ldots1 = \pm\ 2^{-m}}$$

Si definisce **"precisione"** la **distanza tra 1 ed il numero più grande rappresentabile** con m cifre dopo la virgola:

$$\boldsymbol{\varepsilon = 1 - 0.1\ldots1 = 2^{-m}}$$

Se $\mathbf{m = 6}$, $\boldsymbol{\varepsilon} = 1/2^6 = 1/64 =$ **0.015625**

Se $\mathbf{m = 7}$, $\boldsymbol{\varepsilon} = 1/2^7 = 1/128 =$ **0.0078125**

Rappresentazione dei Numeri

Lezione III: Numeri Reali

2. Notazione in Virgola Fissa

Rappresentazione

I **numeri reali** vengono rappresentati con un **numero fisso di bit** dopo (e spesso anche prima) della **virgola**.

Questa rappresentazione è **meno complicata della notazione in virgola mobile** e richiede processori meno potenti.

E' la rappresentazione più **utilizzata** in ambito **finanziario**.

$$x = \frac{1}{2^b} \sum_{i=0}^{N} x_i 2^i$$

I numeri vengono rappresentati come **interi moltiplicati per un fattore (1/2^{b})**, dove **b** indica il **numero di cifre frazionarie** ed **N** è il **numero complessivo di bit** (un bit può essere riservato per il segno).

Notazione Q-point

I formati utilizzati sono comunemente indicati con il formalismo

Qm.b

Dove **m** è il numero di **bit** dalla **parte intera** (escluso il segno, se presente) e **b** il numero di **bit** della **parte frazionaria**.

Q3.4 (**Q3.12**): **8** (**16**) **bit**, dei quali **3** per la **parte intera**, **4** (**12**) per la **parte frazionaria** ed **1** per il **segno**.

Q.15: **16 bit**, **15** per la **parte frazionaria** ed **1** per il **segno**.

Q.31: **32 bit**, **31** per la **parte frazionaria** ed **1** per il **segno**.

Gli ultimi due formati sono indicati per rappresentare numeri nell'intervallo [-1,+1]

Intervallo e Risoluzione

Fissato il numero di bit **N** = **m** + **1** + **b** della rappresentazione, l'intervallo dei numeri rappresentabili è:

$$\left[-\frac{2^{N-1}}{2^b},\frac{2^{N-1}-1}{2^b}\right]=\left[-2^m,2^m-\frac{1}{2^b}\right]$$

Per esempio **Q4.3** su **8** bit : $\left[-2^4,2^4-\dfrac{1}{2^3}\right]=[-16,+15.875]$

Q.15 su **16** bit: $\left[-2^0,2^0-\dfrac{1}{2^{15}}\right]=[-1,+0.999969482421875]$

Si definisce **"risoluzione"** la **più piccola differenza** tra due numeri della rappresentazione:

$$\varepsilon=\frac{1}{2^b}=\frac{1}{2^3}(Q4.3)=\frac{1}{2^{15}}(Q.15)$$

Aritmetica in Virgola Fissa

I calcoli in virgola fissa si riconducono all'**aritmetica** tra **numeri interi** nella rappresentazione a **complemento a due**.

Somme Algebriche

Dati **due numeri** in notazione **Qm.b**, per poterli sommare è necessario che entrambi i numeri siano nello **stesso formato**, altrimenti i numeri devono essere **allineati** con un'operazione di **shift** (a destra o a sinistra).

Il risultato può dare essere **rappresentato precisamente** o dare origine ad un **overflow**.

In caso di **overflow** sarebbe necessario usare un **numero di bit superiore** per il risultato.

Se si usa lo **stesso numero di bit**, il risultato potrebbe essere **radicalmente inaccurato**.

Esempio di Somma

Sommiamo e sottraiamo i numeri **u = 4.75$_{10}$** e **v = 2.5$_{10}$** con la notazione **Q4.3**.

```
u = 00100110 -> 4.75
v = 00010100 -> 2.5

u + v ->  00100110 +
          00010100 =
          --------
          00111010 -> 7.25

-v = C_2(00010100) = 11101100

u - v ->  00100110 +
          11101100 =
          --------
          00010010 -> 2.25
```

Moltiplicazione

Dati due numeri interi in notazione **Qm1.b1** e **Qm2.b2**, per rappresentare il loro prodotto sono necessari **m1+m2+1 bit** per la parte **intera** e **b1+b2 bit** per la parte **frazionaria**.

Consideriamo i numeri **u = 4.75_{10}** e **v = 2.5_{10}** in **Q4.3**.

```
u = 00100110 -> 4.75
v = 00010100 -> 2.5

u x v -> 00100110 x
         00010100 =
         --------
       0010011000
     001001100000
     ------------
     001011111000

uv = 01011111 -> 11.875
```

Troncamento e Arrotondamento

Il **numero finito di bit** per la rappresentazione della parte frazionaria può dare origine ad un **errore di troncamento**.

Vediamo un esempio in **Q3.2** con i numeri **u = 0.5_{10}** e **v = 0.75_{10}**.

```
U = 000010 -> 0.5
V = 000011 -> 0.75

U x V = 000010 x
        000011 =
        ------
        000010
       0000100
       -------
       0000110

U x V = 000001 -> 0.25
```

Diverso dal risultato decimale della somma **0.375_{10}**

Si potrebbe **sommare un bit** al risultato …

```
U x V + 1/2^b = 0000110 +
                0000010 =
                -------
                0001000
```

il numero riscalato è 000010 -> 0.5.

… ma **permane l'errore di arrotondamento!**

Esercizio 3 (I)

Dati i numeri in base-10 A = – 27.25 e B = +3.1, rappresentarli in base-2, complemento a due, in virgola fissa con notazione **Q5.2** ed effettuare le operazioni A + B e A – B.

Passo 1. Conversione di |A| in base-2

Parte intera di |A|:

27 / 2 = 13 + **1** / 2

13 / 2 = 6 + **1** / 2

6 / 2 = 3 + **0** / 2

3 / 2 = 1 + **1** / 2

1 / 2 = 0 + **1** / 2

Parte frazionaria di |A|:

0.25 * 2 = **0** + 0.5

0.5 * 2 = **1** + 0.0

→ |A| = **11011.01**

Esercizio 3 (II)

Passo 2. Scrittura di + |A| e – |A| in Q5.2

|A| = **11011.01**

+ |A| = **011011.01**

– |A| = **100100.11**

Passo 3. Calcolo di |B| (troncando alla seconda cifra significativa)

|B| = **11.00**

Passo 4. Scrittura di + |B| e – |B| in Q5.2

+ |B| = **000011.00**

– |B| = **111101.00**

Esercizio 3 (III)

Passo 5. Calcolo di A + B

100100.11 +

000011.00 =

100111.11

Passo 6. Calcolo di A – B

100100.11 +

111101.00 =

100001.11

Rappresentazione dei Numeri

Lezione III: Numeri Reali

3. Notazione in Virgola Mobile

Floating Point

La **notazione in virgola mobile** è l’analogo binario della **notazione scientifica in base-10**.

Un numero in virgola mobile è rappresentato da:

1. **M_R, significando (o mantissa)**
2. **E_R, esponente**
3. **S, segno (±)**

$$x = \pm M_R \bullet R^{E_R}$$

In base-2 la notazione diviene:

$$x = \pm M_2 \bullet 2^E$$

dove la mantissa M_2 è vincolata nell’intervallo $1 \le M_2 < 2$ può essere scritta:

$M_2 = m_0.\, m_1 m_2 m_3 m_4 m_5 \ldots m_{p-1}$, con $m_0 = 1$ (rapp. normalizzata)

(così definita la notazione **non prevede** la scrittura dello **zero**!)

Il numero di cifre **p** della **mantissa** è detto **“precisione”**.

Precisione Macchina

Vediamo un esempio. Scriviamo il numero **71_{10}** in **base-2**:

$$+71_{10} = +1000111_2 = +1.000111 * 2^6$$

Significante, **esponente** e **precisione** risultano

$$M_2 = 1.000111,\ E_2 = 6,\ p = 7$$

Se il numero x è il numero da rappresentare e **fl(x)** la sua rappresentazione **in virgola mobile**:

$$fl(x) = sgn\,(x)\,(m_0(=1).m_1m_2m_3m_4m_5\ldots m_p)\, b^E$$

$$b^E \leq |x| < b^{E+1}$$

L'**errore assoluto** è $|\, x - fl(x)\, | \leq b^{E+1}\, b^{-p} = b^{E+1-p}$

L'**errore relativo ε** $= |\, x - fl(x)\, | \,/\, |x| \leq (b^{E+1-p}) \,/\, (b^E) = b^{E-p-E+1} = b^{-p+1}$

ε, non dipende da x, è detto **"precisione macchina"**

Lo standard IEEE-754

Standard di rappresentazione dei numeri in virgola mobile, adottato ufficialmente nel 1985 e supervisionato nel 2008.

Definisce:

1. **i formati standard di rappresentazione**, tra cui precisione singola (32-bit) e precisione doppia (64-bit);
2. i formati di **+0, –0, +∞, –∞, NaN** (Not-a-Number) ed i **numeri subnormali**;
3. i formati di **codifica dei bit**;
4. cinque **algoritmi di arrotondamento**;
5. i set delle **operazioni applicabili**;
6. la **gestione** delle cinque **eccezioni** (divisioni per 0, overflow…)

Precisione Singola (32-bit)

```
 1     8              23                    lunghezza in bit
+-+--------+-----------------------+
 S   Exp.          Mantissa
+-+--------+-----------------------+
```

Esponente: **8-bit**, **offset-127**, utilizzando i valori nell'intervallo 1-254 (0 e 255 sono riservati per funzioni speciali);

Mantissa: **23-bit**, il **primo bit** è sempre **1** e viene **omesso** (bit nascosto) per guadagnare un bit di precisione (**$p = 24$**).

Segno: **1-bit**, 0 per numeri positivi, 1 per numeri negativi

[Esponente minimo, Esponente massimo] = [–126,+127]

Cifre significative per la parte frazionaria in base-10: 7.22_{10}

Esponente massimo in base-10: $+38.23_{10}$ ($\sim 2^{127}$)

Esponente minimo in base-10: -45_{10} (numero subnormale)

Esercizio 4 (I)

Scriviamo in **precisione singola** il numero -118.625_{10}

Passo 1. Convertiamo parte **intera** e **frazionaria** in base-2

Divisione	Quoziente	Resto
118 : 2	59	0
59 : 2	29	1
29 : 2	14	1
14 : 2	7	0
7 : 2	3	1
3 : 2	1	1
1 : 2	0	1

$\mathbf{1110110_2}$

Moltiplicazione	Prodotto	Parte Intera
0.625 * 2	1.25	1
0.25 * 2	0.5	0
0.5 * 2	1	1

$\mathbf{0.101_2}$

Esercizio 4 (II)

$$\mathbf{1110110.101_2}$$

Passo 2. **Normalizziamo** il numero ottenuto:

$$\mathbf{1.110110101_2 * 10_2^{6} (= 2_{10}^{6})}$$

Passo 3. L'**esponente** si esprime in **eccesso-127**, per cui convertiamo il numero 133 (= 127 + 6) in base.2

Divisione	**Quoziente**	**Resto**
133 : 2	66	1
66 : 2	33	0
33 : 2	16	1
16 : 2	8	0
8 : 2	4	0
4 : 2	2	0
2 : 2	1	0
1:2	0	1

$\mathbf{10000101_2}$

Esercizio 4 (III)

Segno: **1** (perché il segno è negativo) (1-bit)

Esponente: **10000101** (8-bit)

Infine la **mantissa** vale 1.110110101, ma **il primo bit** è sempre 1, per cui **viene omesso**.

Mantissa: **11011010-10000000-0000000** (23-bit)

Il numero **-118.625_{10}**, in notazione in virgola mobile e **precisione singola** si scrive:

11000010-11101101-01000000-00000000 (32-bit)

Numeri Speciali

Categoria	**Esponente**	**Mantissa**
Zeri	**0**	**0**

s 00000000 00000000-00000000-0000000

(il bit di segno, s, permette di distinguere tra +0 e -0),

Infiniti	**255**	**0**

s 11111111 00000000-00000000-0000000

(il bit di segno permette di definire sia +∞ che –∞),

Numeri Subnormali	**0**	**≠0**

s 00000000 xxxxxxxx-xxxxxxxx-xxxxxxx

(permettono di gestire un graduale underflow)

NaN	**255**	**≠0**

s 11111111 axxxxxxx-xxxxxxxx-xxxxxxx

(0/0, ∞/∞, 0*∞, +∞–∞, log(–1), radq(–1), $\sin^{-1}(x>1)$, $\cos^{-1}(x>1)$)

Precisione Doppia (64-bit)

```
 1   11                       52                    lunghezza in bit
+-+----------+------------------------------------------------+
 S   Exp.                  Mantissa
+-+----------+------------------------------------------------+
```

Esponente: **11-bit**, **offset-1023**, utilizzando i valori nell'intervallo 1-2046 (0 e 2047 sono riservati per funzioni speciali);

Mantissa: **52-bit**, il primo bit è sempre 1 e viene omesso (bit nascosto) per guadagnare un bit di precisione (**p = 53**).

Segno: **1-bit**, 0 per numeri positivi, 1 per numeri negativi

[Esponente minimo, Esponente massimo] = [–1022,+1023]

Cifre significative per la parte frazionaria in base-10: 15.95

Esponente massimo in base-10: 307.95_{10} ($\sim 2^{1023}$)

Esponente minimo in base-10: -324_{10} (numeri subnormali)

Esercizio 5 (I)

Interpretare la sequenza di 8 caratteri esadecimali **C48AE000** come la rappresentazione in virgola mobile a precisione singola secondo lo standard IEEE-754 di un numero. Indicare il corrispondente numero in base-10.

Passo 1. Conversione della sequenza dalla base-16 alla base-2

C-4-8-A-E-0-0-0

1100-0100-1000-1010-1110-0000-0000-0000

11000100100010101110000000000000

1-10001001-00010101110000000000000

Segno: **1** → numero negativo

Esponente: **10001001**

Mantissa: **00010101110000000000000**

Esercizio 5 (II)

Passo 2. Conversione in base-10 dell'Esponente: **10001001** = 137

Sottrazione dell'offset = 137 – 127 = **10**

Passo 3. Scrittura della Mantissa Normalizzata:

$$\mathbf{1.00010101110000000000000 * (10)^{10}}$$

$$\mathbf{10001010111.000000000000}$$

Passo 4. Conversione in base-10 del Numero: **10001010111**

$$\mathbf{2^{10}+2^{6}+2^{4}+2^{2}+2^{1}+2^{0}}$$

$$\mathbf{1024+64+16+4+2+1}$$

$$\mathbf{1111}$$

La sequenza **C48AE000** corrisponde al numero decimale: **– 1111**

Arrotondamento

Lo **standard IEEE** definisce **5 algoritmi di arrotondamento**:

Arrotondamento al valore più vicino:

- Arrotondamento per **difetto**, nel caso il numero sia esattamente a metà dell'intervallo;
- Arrotondamento per **eccesso**, nel caso il numero sia esattamente a metà dell'intervallo.

Arrotondamento diretto:

- Arrotondamento **a zero (troncamento)**;

1. Arrotondamento **verso +∞**;
2. Arrotondamento **verso –∞**.

Aritmetica in Virgola Mobile (I)

- **Addizione e Sottrazione:**
  1. Rappresentazione dei numeri con lo **stesso esponente**.
  2. **Esecuzione** delle operazioni.
  3. **Arrotondamento** del risultato.
- **Moltiplicazione e Divisione:**
  1. **Moltiplicazione/Divisione** delle **mantisse.**
  2. **Somma/Differenza** degli **esponenti.**
  3. **Arrotondamento** del risultato.
  4. **Normalizzazione.**

Nel caso dell'**addizione/sottrazione** possono verificarsi **errori di cancellazioni**, legati alla significatività dei bit.

Nel caso delle **moltiplicazioni/divisioni**, **piccoli errori possono propagarsi** in caso di operazioni in successione.

Aritmetica in Virgola Mobile (II)

- **Non è associativa**

$$(x+y)+z \neq x+(y+z)$$

$$(x \bullet y) \bullet z \neq x \bullet (y \bullet z)$$

- **Non è distributiva**

$$x \bullet (y+z) \neq (x \bullet y)+(x \bullet z)$$

- Esistono l'**elemento neutro** della moltiplicazione, dell'addizione e l'**opposto** ma non sono unici

$1.0 + (10^{100} – 10^{100}) = 1.0$

$(1.0 + 10^{100}) – 10^{100} = 0.0$ (**Assorbimento/Cancellazione**)

Alcuni numeri **non** sono **rappresentabili**, ad esempio: $\mathbf{0.1_{10}}$

In **campo finanziario** si preferisce l'aritmetica in **virgola fissa**.

Gestione delle Eccezioni

Nella processo **computazionale** si può incappare in **3 tipologie di problemi**:

1. un'operazione matematicamente **non lecita** (divisione per zero);
2. un'operazione matematicamente **lecita**, ma **non rappresentabile** (radice quadrata di un numero negativo);
3. un'operazione **lecita**, ma il cui **risultato** sia al di **fuori del range** del formato (overflow, underflow, perdita di precisione).

Lo standard **IEEE 754** definisce **5 flag di eccezione**:

1. **inesatto**, risultato **arrotondato** non matematicamente corretto;
2. **underflow**, restituisce un numero **subnormale**;
3. **overflow**, restituisce $\pm\infty$;
4. **divisione per zero**, restituisce $\pm\infty$ se l'operando è finito;
5. **invalido**, restituisce **NaN**, per esempio 0/0 o numero complesso.

Esercizio 6 (I)

Rappresentare il numero decimale A = – 7.55 in base-2, complemento a due, in virgola fissa con notazione **Q5**.**3**.
Rappresentare lo stesso numero in base-2 in virgola mobile, utilizzando **1** per il **segno**, **3** per l'**esponente** e **5** per la **mantissa**.

Passo 1. Conversione di |A| in base-2

Parte intera di |A|: $7_{10} = 111_2$

Parte frazionaria di |A|:

$0.55 * 2 = \mathbf{1} + 0.1$

$0.1 \quad * 2 = \mathbf{0} + 0.2$

$0.2 \quad * 2 = \mathbf{0} + 0.4$

$0.4 \quad * 2 = \mathbf{0} + 0.8$

$0.8 \quad * 2 = \mathbf{1} + 0.6$

$\rightarrow |A| = \mathbf{111.10001\ldots}$

Esercizio 6 (II)

Passo 2. Scrittura di – |A| in Q5.3

|A| = **111.10001…**

+ |A| = **000111.100**

– |A| = **111000.100**

Passo 3. Scrittura di A in virgola mobile

Segno (1 bit): numero negativo → **1**

|A| = **1.1110001 * (10)^{2}**

Esponente (3 bit): **2**, in offset $q = 2^{n-1} - 1 = 2^2 - 1 = 3 \rightarrow 2+3 = 5$

→ $\mathbf{5_{10} = 101_2}$

Mantissa (5 bit): **11100**

– |A| = **110111100**

Esercizio 7 (I)

Interpretare la sequenza di 8 caratteri binari **10011000** come la rappresentazione di un numero in base-2, complemento a due, in virgola fissa con notazione **Q4**.**3**. Interpretare successivamente la medesima sequenza come la rappresentazione in virgola mobile di un numero in base 2, utilizzando **1** per il **segno**, **3** per l'**esponente** e **4** per la **mantissa**. Scrivere i corrispondenti numeri in base-10.

Passo 1. Virgola fissa Q4.3: **10011.000**

Il bit di segno è 1, quindi il numero rappresentato è negativo.

Calcoliamo il complemento a due del numero, per determinarne il valore assoluto: **01101.000**

Parte intera: $\mathbf{1101_2} = \mathbf{13_{10}}$, Parte Frazionaria: $\mathbf{000_2} = \mathbf{0_{10}}$

La sequenza **10011.000** corrisponde al numero decimale: $\mathbf{-\ 13_{10}}$

Esercizio 7 (II)

Passo 2. Virgola mobile: **10011000**

Segno (1 bit): **1** $\rightarrow$ numero negativo

Esponente (3 bit): $\mathbf{001_2} = 1_{10}$

1 in offset $q = 2^{n-1} - 1 = 2^2 - 1 = 3 \rightarrow 1 - 3 = -2$

Mantissa (4 bit): **1000**

$\mathbf{1.1000 * (10)^{-2}} = \mathbf{0.011000} = 0.25 + 0.125 = 0.375$

La sequenza **10011000,** scritta in virgola mobile, utilizzando **1** bit per il **segno**, **3** bit per l'**esponente** e **4** bit per la **mantissa** corrisponde al numero decimale: $\mathbf{-\,0.375_{10}}$