Informatica Generale
02 - Rappresentazione numeri razionali

- **Cosa vedremo:**
  - Rappresentazione binaria dei numeri razionali
    - Rappresentazione in virgola fissa
    - Rappresentazione in virgola mobile

La rappresentazione dei numeri razionali: virgola *fissa*

- Un numero razionale ha una **parte intera** (prima della virgola) e una **parte frazionaria** (dopo la virgola), es: **`3.12`**, **`0.00074`**, **`5000.7`**, ... (in base 10)
- rappresentazione binaria solitamente su 4/8 byte
- **Rappresentazione in virgola fissa**: riservo un numero fisso di bit per parte intera e parte frazionaria;
- per semplicità consideriamo solo numeri positivi

Parte intera	Parte frazionaria

es: con **3 bit** per la **parte intera** e **2 bit** per quella **frazionaria** posso rappresentare numeri come:
`011.11` **`101.01`** **`000.01`**

La rappresentazione dei numeri razionali: virgola fissa (2)

- **Conversione di numeri frazionari da base 2 a base 10**: come per gli interi

  - $101.01 = 1 * 2^2 + 0 * 2^1 + 1 * 2^0 + 0 * 2^{-1} + 1 * 2^{-2} =$
    $= 4 + 0 + 1 + 0 + 0.25 =$
    $= 5.25$

perché $2^{-1} = 1/2 = 0.5$, $2^{-2} = 1/2^2 = 0.25$ e in generale $2^{-n} = 1/2^n$

La rappresentazione dei numeri razionali: virgola fissa (3)

- **Conversione da base 10 a base 2**:
  - parte intera: per divisioni successive
  - parte decimale: per ***moltiplicazioni successive***

es: `5.125` parte intera: `101` parte decimale:

```
0.125 * 2 = 0.25    => 0 riporto 0.25
0.25 * 2  = 0.5     => 0 riporto 0.5
0.5  * 2  = 1       => 1
```

Quindi `5.125 => 101.001`

- **Nota**: alcuni numeri frazionari con rappresentazione ***finita*** in base 10 sono ***periodici*** in base 2. Esempio:

$$0.6 \Rightarrow 0.1001100110011001... = 0.\underline{1001}$$

- La rappresentazione binaria può causare **troncamento**

Rappresentazione in virgola fissa: spreco di memoria e limiti di rappresentazione

- Spreco di bit per memorizzare zeri: es. in base 10, con 5 cifre per la parte intera e 2 cifre riservate alla parte frazionaria **`40000.00`** oppure **`00000.07`**
- Intervallo di numeri rappresentabili piccolo per molte applicazioni: in base 2, con **`N`** bit per parte intera e **`K`** per parte frazionaria, il numero max rappresentabile è $2^{N}-1/2^{K}$, il minimo numero positivo è $1/2^{K}$.
- Es: per **`N=4`** e **`K=3`**, il max è **`1111.111 = 15.875`**, mentre il minimo positivo è **`0000.001 = 0.125`**.
- La **notazione esponenziale** o **floating point** (**virgola mobile**) riduce entrambi i problemi: i bit vengono usati più efficientemente, per un intervallo di numeri più ampio.

Rappresentazione in virgola mobile (floating point, notazione esponenziale)

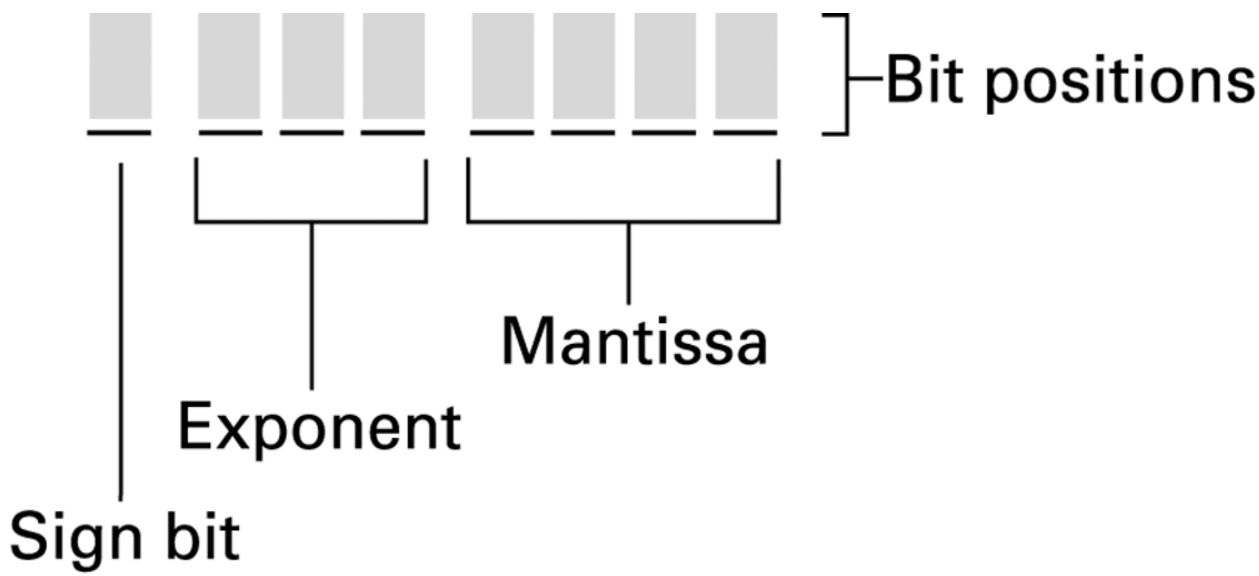
- Fissata una base **B**, un numero **N** può essere rappresentato da una coppia: (**mantissa** **M**, **esponente** **E**), con il seguente significato $\mathbf{N = M * B^E}$
- La **mantissa** rappresenta le **cifre significative** (cioè diverse da zero) del numero, l'**esponente** indica la posizione della virgola.
- Es: in **base 10**, il numero **`474.35`** ha varie rappresentazioni in virgola mobile, del tipo **(M, E)**, come:
  - **`(0.47435,+3)`**, cioè $\mathbf{0.47435*10^3}$ **[forma normalizzata]**
  - **`(0.047435,+4)`**, cioè $\mathbf{0.047435*10^4}$
  - **`(47435.0,-2)`**, cioè $\mathbf{47435.0*10^{-2}}$
- Nella **forma normalizzata**, la mantissa ha la prima cifra significativa (diversa da zero) subito dopo la virgola.

Rappresentazione in virgola mobile Esempio in base 2

- In base 2 la situazione è del tutto analoga.
- Es: in **base 2**, il numero positivo **`101.011`** (che vale 5.375 in base 10) ha varie rappresentazioni in virgola mobile, come:
  - **`(0.101011,+3)`**, cioè $\mathbf{0.101011*2^{3}}$ **[forma normalizzata]**
  - **`(0.0101011,+4)`**, cioè $\mathbf{0.0101011*2^{4}}$
  - **`(101011.0,-3)`**, cioè $\mathbf{101011.0*2^{-3}}$
- Nella **forma normalizzata**, la mantissa ha la prima cifra significativa (diversa da zero) subito dopo la virgola.

Rappresentazione in virgola mobile con numero fissato di bit

- Per rappresentare numeri in virgola mobile nel computer, dobbiamo fissare un numero di bit N_m per **il valore assoluto della mantissa**, e un numero di bit N_e per l'esponente **in complemento a 2** (o in **notazione in eccesso**, come nel libro).
- **Numeri negativi**: rappresentiamo il valore assoluto, mettendo **1** nel **bit del segno**.
- Se $N_m = 4$ e $N_e = 3$ abbiamo una rappr. su 8 bit come a destra.



Rappresentazione in virgola mobile: come estrarre il numero in base 10

- Fissiamo $\mathbf{N_m} = 4$ e $\mathbf{N_e} = 3$, come visto sopra. **Come si ottiene il numero razionale in base 10 corrispondente a un dato byte?** Es: **1 010 1010** (segno **1**, esponente **010**, mantissa **1010**)
  1) Converto l'esponente (in complemento a 2 su tre bit) in base 10: **010** vale **+2**;
  2) Aggiungo **0.** (0 virgola) prima della mantissa (che deve cominciare con **1**). Quindi **1010** diventa **0.1010**;
  3) Sposto la virgola di un numero di posizioni pari all'esponente verso destra se positivo, verso sinistra se negativo. Quindi poiché l'esponente è **+2**, **0.1010** diventa **10.10**;
  4) Converto il numero frazionario in base 10: **10.10** vale **2.5**;
  5) Se il bit del segno è **1**, prendo l'opposto: **1 010 1010** è **-2.5**.

Rappresentazione in virgola mobile: come estrarre il numero in base 10

- Altro esempio, con $\mathbf{N_m}$ = 6 e $\mathbf{N_e}$ = 4. Es: **0 1001 101111** (segno **1**, esponente **1101**, mantissa **101111**)
  1) L'esponente (in complemento a 2 su quattro bit) vale **-3**;
  2) La mantissa con virgola diventa **0.101111**;
  3) Sposto la virgola di **3** posizioni verso sinistra: diventa **0.000101111**; 1/16 + 1/64 + 1/128 + 1/256 + 1/ 512
  4) Converto in base 10: **0.000101111 =** 1/16 + 1/64 + 1/128 + 1/256 + 1/ 512 = **0.091796875**
  5) Il bit del segno è **0**, quindi **1 010 1010** vale **0.091796875**.

Rappresentazione in virgola mobile: come codificare un numero in base 10

- Fissiamo $\mathbf{N_m}$ = 4 e $\mathbf{N_e}$ = 3. **Come si ottiene la configurazione di bit corrispondente a un numero razionale in base 10?** Es: **-0.3125**
  1) Se il numero è negativo, metto **1** nel bit del segno e considero il valore assoluto. Quindi continuo con **0.3125**.
  2) Converto il numero razionale in base 2: **0.3125** diventa **0.0101** in base 2;
  3) Prendo come mantissa i primi $\mathbf{N_m}$ bit a partire da quello più significativo (il primo **1** da sinistra); aggiungo zeri se necessario; eventuali **1** dopo i primi $\mathbf{N_m}$ bit vengono persi con conseguenti errori di troncamento. Quindi la mantissa di **0.0101** è **1010**.

Rappresentazione in virgola mobile: come codificare un numero in base 10

4) Per l'esponente: conto di quante posizioni devo spostare la virgola ***verso sinistra*** per arrivare ***a sinistra*** del primo **1**. L'esponente di **0.0101** è **-1** perché devo spostare la virgola ***a destra*** di una posizione. In complemento a 2 su 3 bit, **-1** vale **111**.

5) Quindi **-0.3125** viene rappresentato come **1 111 1010**.