Rappresentazione dei numeri razionali

- Un numero razionale è un numero ottenibile come rapporto tra due numeri interi, il secondo dei quali diverso da 0
- In un intervallo [a,b] la quantità di numeri razionali è infinita
- Per rappresentare tutti i numeri razionali in un intervallo [a,b] sono necessarie un numero infinito di cifre.
- In termini di codici, fissato un intervallo e fissati l'alfabeto e la lunghezza delle parole di codice, è possibile rappresentare solo un sottoinsieme di tutti i valori razionali appartenenti all'intervallo scelto:
  - 7/2 = 3,5 sono richieste solo due cifre; rappresentabile in modo esatto
  - 5/3=1,666… parte frazionaria periodica; rappresentabile solo con un'approssimazione

Virgola fissa e virgola mobile

- Per la codifica dei numeri razionali si ricorre a due schemi:
  - Virgola fissa (o fixed point)
    - La posizione della virgola , e quindi la dimensione della parte frazionaria, è fissata
  - Virgola mobile (o floating point)
    - la posizione della virgola non è fissata a priori, ma è controllata mediante un fattore che dipende dalla codifica

- NB: Nel campo dei calcolatori i numeri razionali sono utilizzati come approssimazione per i numeri irrazionali ($\sqrt{3}, \pi, e$)

Codifica in virgola fissa

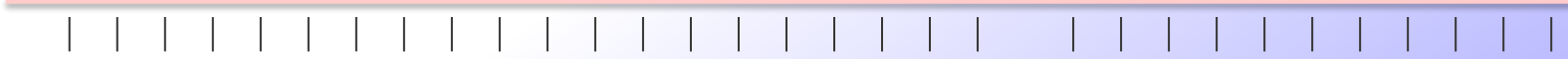
- Sia $X = [\alpha_{n-1} \dots \alpha_1 \, \alpha_0 \, \alpha_{-1} \dots \alpha_{-k}]$
- Il suo valore è dato dalla relazione:

$$v(X) = \alpha_{n-1}b^{n-1} + \cdots + \alpha_1 b^1 + \alpha_0 b^0 + \alpha_{-1}b^{-1} + \cdots + \alpha_{-k}b^{-k}$$

- Si nota che:
  1. I coefficienti $\alpha_{-1} \dots \alpha_{-k}$ rappresentano la parte frazionaria.
  2. Moltiplicando qualsiasi parola di codice per un fattore b^k otteniamo solo numeri interi
  3. È possibile utilizzare gli algoritmi visti per la codifica naturale a meno del fattore di scala b^k.
- NB: i valori adiacenti nella codifica differiscono di un fattore $\Delta = \mathbf{b^{-k}}$

$$(\alpha_{-k} + 1)b^{-k} - \alpha_{-k}b^{-k} = b^{-k}$$

  - Due parole adiacenti differiscono di una unità nel coefficiente α_{-k} :
    - I valori sono distribuiti in modo UNIFORME



Codifica in virgola mobile

- Si basa su tre elementi distinti:
  - segno s, mantissa m, esponente e
- Una parola di codice è rappresentata da
- $X = [s\ e\ m]$, con $v(X) = s \cdot m \cdot b^e$.
  - s = ±1 m = mantissa normalizzata, cioè $1 \leq m < b$
- NB: fissati n_e bit per l'esponente e n_m bit per la mantissa i valori adiacenti nella codifica differiscono di un fattore $\Delta = \mathrm{b}^{e-n_m}$
  - Due parole adiacenti differiscono di una unità nella cifra meno significativa della mantissa ed hanno lo stesso esponente: $\Delta = \mathrm{b}^{-\mathrm{n_m}} \cdot b^e = b^{e-n_m}$
    - I valori sono distribuiti in modo NON UNIFORME
      - La densità varia in funzione dell'esponente:
    - valori vicini con esponente piccolo, valori lontani con esponente grande

Lo standard IEEE 754

- Definisce i dettagli necessari a completare la specifica della rappresentazione in virgola mobile per la base 2

Precisione	Segno	Esponente	Mantissa
Singola	1 bit	8 bit	23 bit
Doppia	1 bit	11 bit	52 bit

- Le mantisse in realtà avrebbero 24 e 53 bit rispettivamente, ma essendo il primo sempre 1 $(1 \leq m < b)$, esso viene omesso
- L'esponente viene espresso in forma polarizzata
  - Semplifica l'ordinamento
  - Singola precisione: bias=127; doppia precisione: bias=1023;
- Riassumendo:

$$v(X) = (-1)^{segno} \times 1, mantissa \times 2^{esponente-bias}$$

Overflow ed Underflow

- Overflow:
  - Si verifica quando l'esponente è troppo grande per essere contenuto nel campo a lui riservato
  - Il numero è troppo grande per essere rappresentato

- Underflow
  - Si verifica quando un esponente negativo diventa troppo grande per essere contenuto nel campo a lui riservato
  - Il numero è troppo piccolo (in modulo) per essere rappresentato

Esempio conversione da decimale a IEEE754

- Sia $X_{10} = -28{,}75$
  - Segno negativo: s=1;
  - Parte intera: $28_{10} = 11100_{10}$
  - Parte decimale: $\mathbf{0,75_{10} = \frac{1}{2} + \frac{1}{4} = 2^{-1} + 2^{-2} = 0,11_2}$
  - Mantissa: 11100,11
  - Mantissa normalizzata: $1{,}110011 \times 2^4$
  - Esponente: $\mathbf{4_{10} \Rightarrow 4 + 127 = 131 = 10000011}$

$$X_{IEEE754} = \mathbf{1\ 10000011\ 11001100000000000000000}$$

segno esponente mantissa

Esempio conversione da IEEE754 a decimale

- Sia:

$$X_{IEEE754} = \underbrace{0}_{\text{segno}}\ \underbrace{10000100}_{\text{esponente}}\ \underbrace{11001101000000000000000}_{\text{mantissa}}$$

  - Segno: 0, numero positivo
  - Esponente: $10000100_{P,2} = 132 - 127 = 5_{10}$
  - Mantissa: $1,11001101 \times 2^5 = 111001,101$
  - Parte intera = $111001_2 = 57$
  - Parte Decimale = $0,101_2 = \frac{1}{2} + \frac{1}{8} = 0,625$

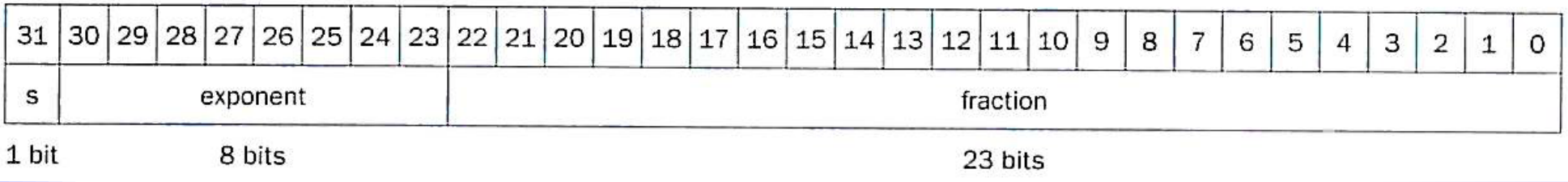
- $X_{10} = 57,625$

IEEE 754 rappresentazione dei numeri

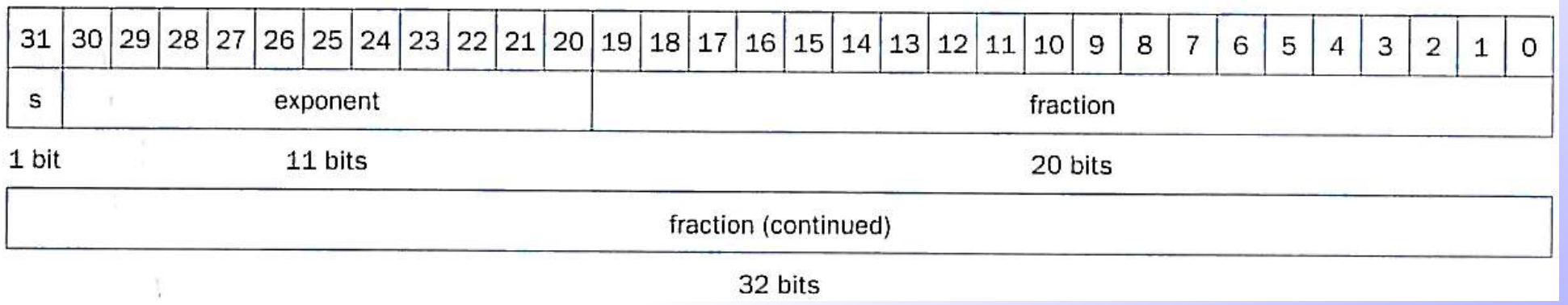
Single precision		Double precision		Object represented
Exponent	Fraction	Exponent	Fraction	
0	0	0	0	0
0	Nonzero	0	Nonzero	± denormalized number
1–254	Anything	1–2046	Anything	± floating-point number
255	0	2047	0	± infinity
255	Nonzero	2047	Nonzero	NaN (Not a Number)

Hardware per la codifica IEEE 754

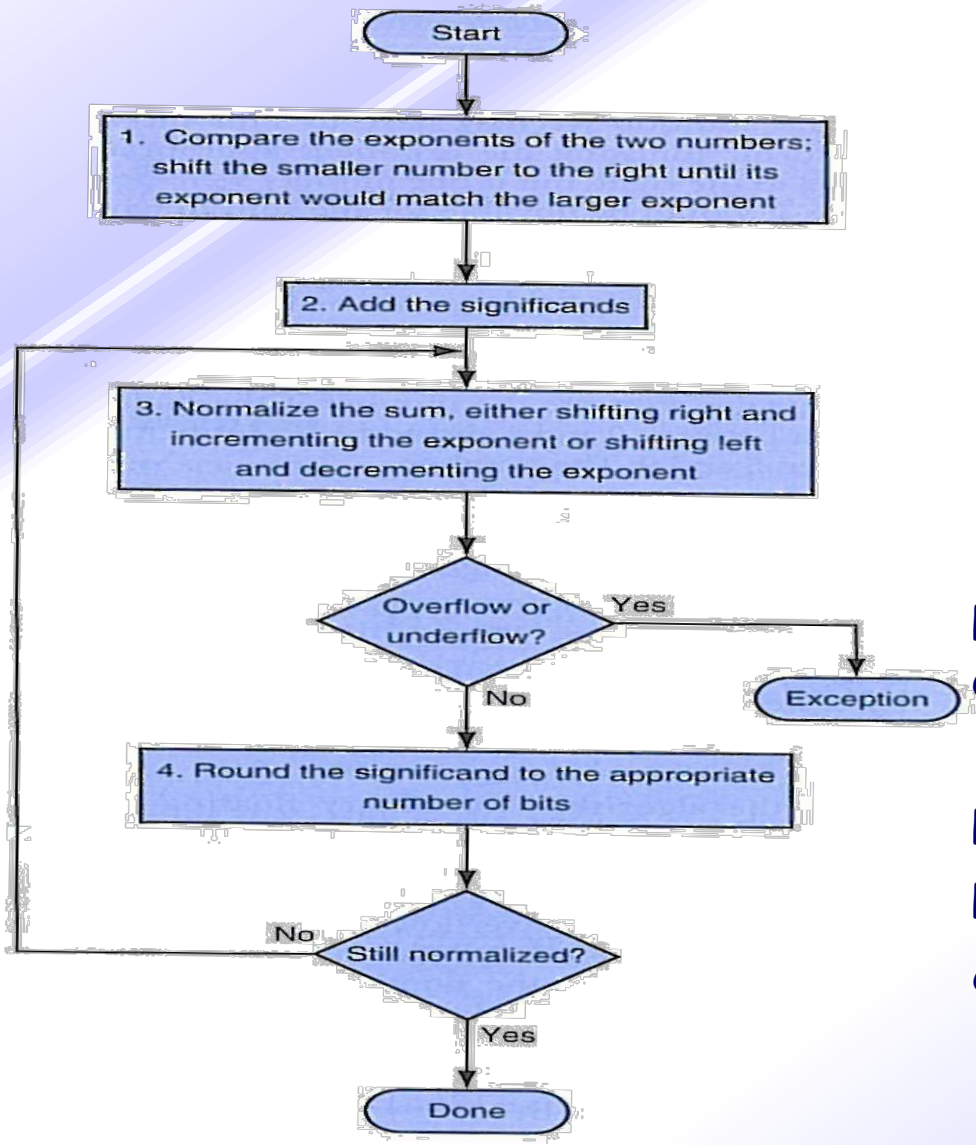
- Singola precisione: una parola



- Doppia precisione: due parole



Somma binaria in codifica IEEE 754



➢ Esempio

$$X_{IEEE754} = 1\ 10000011\ 11001100000000000000000 = -1,110011_2 \times 2^4$$

$$Y_{IEEE754} = 0\ 10000100\ 11001101000000000000000 = 1,11001101_2 \times 2^5$$

$$X + Y =$$

Passo1: si scorre a destra la mantissa del numero con esponente minore finché gli esponenti non sono uguali

$$X = -1,110011_2 \times 2^4 = -0,1110011_2 \times 2^5$$

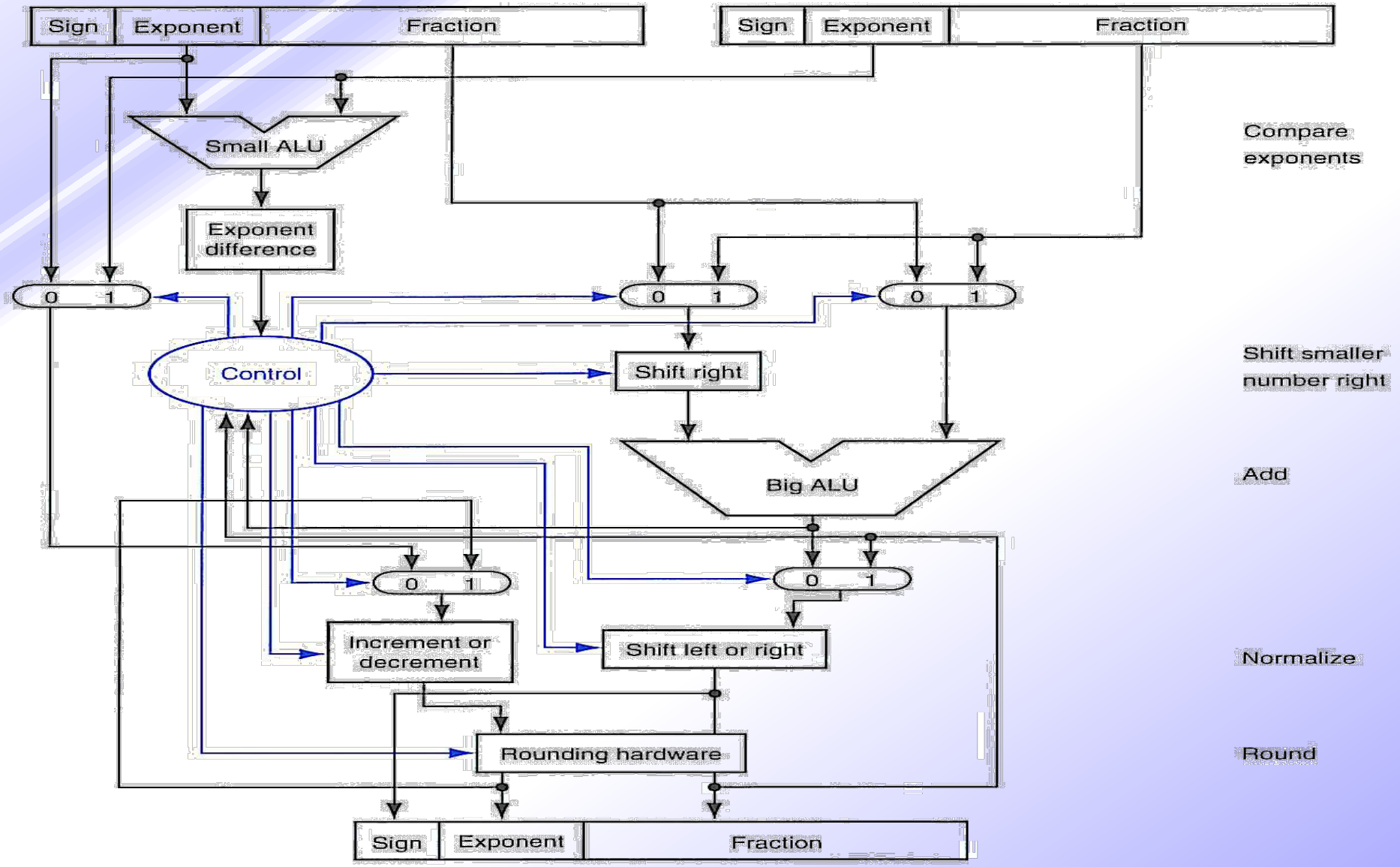
Passo2: si sommano le mantisse:

Passo3: Si normalizza la somma controllando eventuali overflow:

```
  11001101  -
  01100110  =
------------
  01100111
```

$$X + Y = 1,110011 \times 2^3$$

Hardware per la somma in IEEE754



Alcune complicazioni della virgola mobile

- Le operazioni sono più complicate
- Oltre all'overflow, possiamo avere l'underflow
- L'Accuracy può diventare un problema (V.Testo)
  - La codifica IEEE754 ha due bit extra: guard e round
  - Quattro modelli di arrotondamento
  - Un numero positivo diviso per zero fa «infinito»
  - Zero diviso per zero fa «Not A Number»
  - Altre difficoltà
- Implementare lo standard è COMPLESSO
- Non utilizzarlo potrebbe essere ANCHE PEGGIO
  - V. Testo per approfondimenti

Considerazioni conclusive

- In tutti i calcolatori l'aritmetica è limitata da una precisione finita
- Una sequenza di bit non ha significato senza uno standard che la interpreti
  - Complemento a 2
  - IEEE 754
- Computer instructions determine meaning of the bit patterns
- Le performance e l'accuracy sono importanti!!
  - Ci sono molte difficoltà negli elaboratori reali (i.e. algoritmi e implementazioni)

➢ siamo pronti per implementare

➢ IL PROCESSORE