Calcolatori Elettronici

Parte II: Sistemi di Numerazione Binaria

Prof. Riccardo Torlone
Università di Roma Tre

Unità di misura

Exp.	Explicit	Prefix	Exp.	Explicit	Prefix
10^{-3}	0.001	milli	10^{3}	1,000	Kilo
10^{-6}	0.000001	micro	10^{6}	1,000,000	Mega
10^{-9}	0.000000001	nano	10^{9}	1,000,000,000	Giga
10^{-12}	0.000000000001	pico	10^{12}	1,000,000,000,000	Tera
10^{-15}	0.000000000000001	femto	10^{15}	1,000,000,000,000,000	Peta
10^{-18}	0.000000000000000001	atto	10^{18}	1,000,000,000,000,000,000	Exa
10^{-21}	0.000000000000000000001	zepto	10^{21}	1,000,000,000,000,000,000,000	Zetta
10^{-24}	0.000000000000000000000001	yocto	10^{24}	1,000,000,000,000,000,000,000,000	Yotta

Attenzione però, se stiamo parlando di **memoria:**

- 1Byte = 8 bit
- 1K (KiB: KibiByte) = 2^{10} = 1.024 ~ 10^3
- 1M (MeB: MebiByte) = 2^{20} = 2^{10} 2^{10} =1.048.576 ~ 10^6
- 1G (GiB: GibiByte) = 2^{30} = 2^{10} 2^{10} 2^{10} =1.073.741.824 ~ 10^9
- 1T (TiB: TebiByte) = 2^{40} = ... =1.099.511.627.770 ~ 10^{12}

1 Mb = 1 Mega bit = 10^6 bit (misura di velocità)

4 GB = 4 Giga bytes = 2^{32} bytes (misura di memoria)

Ordini di grandezza

Le potenze di 2:

- $2^0 \ldots 2^9 = 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, ..$
- $2^{10} = 1.024$ $\sim 10^3$ 1K
- $2^{20} = 2^{10}\ 2^{10} = 1.048.576$ $\sim 10^6$ 1M
- $2^{30} = 2^{10}\ 2^{10}\ 2^{10} = 1.073.741.824$ $\sim 10^9$ 1G
- $2^{40} = \ldots = 1.099.511.627.770$ $\sim 10^{12}$ 1T
- $2^{50} = \ldots = 1.125.899.906.842.624$ $\sim 10^{15}$ 1P

ES $2^{26} = 2^6 \cdot 2^{20} = 64$ M

Il numero n di bit di un indirizzo binario determina le dimensioni della memoria (disposizioni con ripetizione di 0/1 su n posizioni):

CPU	*bit indirizzo*	*Memoria*
8080	16 bit	64 K
8086	20 bit	1 Mega
80286	24 bit	16 Mega
80486	32 bit	4 Giga
Pentium	32 bit	4 Giga

Un sistema di riferimento impreciso..

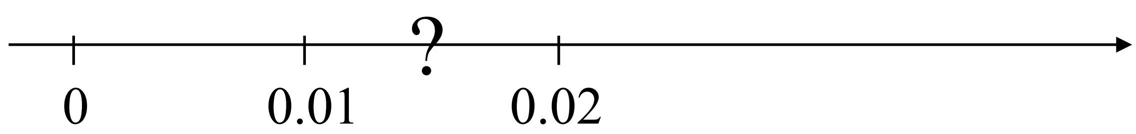
Numeri e numerali

- **Numero**: entità astratta
- **Numerale**: stringa di caratteri che rappresenta un numero in un dato sistema di numerazione

- Lo stesso numero è rappresentato da numerali diversi in sistemi di numerazione diversi
  - 156 nel sistema decimale - CLVI in numeri romani
- Lo stesso numerale rappresenta numeri diversi in sistemi di numerazione diversi
  - 11 vale *undici* nel sistema decimale *tre* nel sistema binario
- Il numero di caratteri del numerale determina l'intervallo di numeri rappresentabili
  - interi a 3 cifre con segno nel sistema decimale: [-999,+999]

Numeri a precisione finita

- Numero **finito** di cifre
- Si perdono alcune proprietà:
  - chiusura operatori (+ , −, ×)
  - proprietà associativa, distributiva,..
  - Esempio:
    - 2 cifre decimali e segno [−99,+99]
    - 78+36=114 (chiusura)
    - 60+(50−40) ≠ (60+50)−40 (associatività)
- Errori di arrotondamento
- Buchi nella rappresentazione dei reali
  - Esempio:
    - numerali decimali con due sole cifre frazionarie



Meccanismo di base: sistema posizionale

- Ciascuna cifra rappresenta il coefficiente di una potenza della base
- L'esponente è dato dalla posizione della cifra

b = base

$0 \leq a_i \leq b - 1$

$a_m \; .... \; a_1 \; a_0 \; . \; a_{-1} \; a_{-2} \; ... \; a_{-k}$

$$N = \sum_{i=-k}^{m} a_i \, b^i$$

ES

125.42

$10^2 \quad 10^1 \quad 10^0 \quad 10^{-1} \quad 10^{-2}$

Se la base è b occorrono b simboli:

- b = 10 {0,1,…,9}
- b = 2 {0,1}
- b = 8 {0,1,…,7}
- b = 16 {0,1,…,9,A,B,C,D,E,F}

Esempio in base binaria (virgola fissa)

1010.01

$2^3 \quad 2^2 \quad 2^1 \quad 2^0 \quad 2^{-1} \quad 2^{-2}$

Numero rappresentato in formato decimale:

$$1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0 + 0 \cdot 2^{-1} + 1 \cdot 2^{-2} = 10.25$$

Esempio in base ottale (virgola fissa)

2107.45

8^3 8^2 8^1 8^0 8^{-1} 8^{-2}

Numero rappresentato in formato decimale:

$$2 \cdot 8^3 + 1 \cdot 8^2 + 0 \cdot 8^1 + 7 \cdot 8^0 + 4 \cdot 8^{-1} + 5 \cdot 8^{-2} = 1095.578125$$

Conversione decimale-binario

- Si effettuano divisioni ripetute per 2
- Il resto delle divisioni fornisce le cifre del numerale binario (a partire dalla meno significativa)

ES $(26)_{10} = (11010)$

$26 \;/\; 2 \xrightarrow{\text{resto}} 0$ *cifra meno significativa*

$13 \;/\; 2 \xrightarrow{\text{resto}} 1$

$6 \;/\; 2 \xrightarrow{\text{resto}} 0$

$3 \;/\; 2 \xrightarrow{\text{resto}} 1$

$1 \;/\; 2 \xrightarrow{\text{resto}} 1$ *cifra più significativa*

0

- Altrimenti si determina ad occhio quali potenze di 2 sono contenute nel numero

ES $(26)_{10} = 1 \cdot 16 + 1 \cdot 8 + 0 \cdot 4 + 1 \cdot 2 + 0 \cdot 1$

Numeri naturali

- Rappresentando gli interi positivi in notazione binaria con n bit si copre l'intervallo $[0\ ,\ 2^n - 1]$
- Si sfruttano tutte le 2^n disposizioni

ES n=3 [0,7]

0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

NB *Anche gli 0 non significativi devono essere rappresentati*

Addizioni tra numeri naturali

- Le addizioni fra numerali si effettuano cifra a cifra (come in decimale) portando il riporto alla cifra successiva

0 + 0 = 0
0 + 1 = 1
1 + 0 = 1
1 + 1 = 0 con il riporto di 1

ES

$3 + 2 = 5$

$$\begin{array}{r} 0011 \; + \\ 0010 \; = \\ \hline 0101 \;\;\; \end{array}$$

Se il numero di cifre non permette di rappresentare il risultato si ha un trabocco nella propagazione del riporto

Moltiplicazioni fra numeri naturali

- La tabellina delle moltiplicazioni è molto semplice:

	0	1
0	0	0
1	0	1

- L'operazione fra numerali si effettua come in decimale: si incolonnano e si sommano i prodotti parziali scalandoli opportunamente:

$$
\begin{array}{lrl}
(11)_{10} & 1011 & \times \\
(5\ )_{10} & 101 & = \\
\hline
 & 1011 & \\
 & 0000 & \\
 & 1011 & \\
\hline
(55)_{10} & 110111 &
\end{array}
$$

- Notare che ciascun prodotto parziale è pari a zero o al moltiplicando

Numeri in virgola fissa senza segno

- Naturale estensione della rappresentazione dei numeri naturali
  - Si stabilisce il numero di bit
  - Viene fissata la posizione della virgola
  - Si interpreta con il meccanismo posizionale di base
- Esempio:
  - 6 cifre di cui due decimali
  - Numerale: 1010.01
  - Interpretazione: $1 \cdot 2^{3} + 0 \cdot 2^{2} + 1 \cdot 2^{1} + 0 \cdot 2^{0} + 0 \cdot 2^{-1} + 1 \cdot 2^{-2} = 10.25$

Addizioni tra numeri positivi in virgola fissa

- Si opera come in decimale

ES

3,5 + 2,75 = 6,25

$$\begin{array}{r} 0011.10 \; + \\ 0010.11 \; = \\ \hline 0110.01 \end{array}$$

Moltiplicazioni tra numeri positivi in virgola fissa

- Si opera come in decimale, tenendo conto del numero di cifre frazionarie e riposizionando il punto:

$$
\begin{array}{lr}
(2.75)_{10} & 10.11\ \times \\
(1.25)_{10} & 01.01\ = \\
\hline
 & 10\,11 \\
 & 0\,00\,0 \\
 & 10\,11 \\
\hline
(3.4375)_{10} & 11.01\,11
\end{array}
$$

$$
\boxed{
\begin{array}{lr}
(2.75)_{10} & 10.11\ \times \\
(2)_{10} & 10\ = \\
\hline
 & 00\,00 \\
 & 1\,01\,1 \\
\hline
(5.5)_{10} & 1\,01.10
\end{array}
}
$$

- Notare che:
  - *moltiplicare per 2^n equivale a spostare il punto di n posti a destra*
  - *moltiplicare per 2^{-n} equivale a spostare il punto di n posti a sinistra*

Moltiplicazione per potenze di due

- Moltiplicare per 2^n equivale a spostare il punto di n posti a destra

$(3.75)_{10}$ 011.11 x

$2^2 = (4)_{10}$ 100.00 =

```
       011.11  x
       100.00  =
   --------------
        00000
       00000
      00000
     00000
    01111
   -------------
    01111.0000
```

$(15)_{10}$ 01111.0000

Moltiplicazione per potenze di due

- Moltiplicare per 2^{-n} equivale a spostare il punto di n posti a sinistra

$$
\begin{array}{ll}
(3.75)_{10} & 11.11 \;\; \times \\
2^{-2} = (0.25)_{10} & \underline{00.01 \;\; =} \\
 & 1111 \\
 & 0000 \\
 & 0000 \\
 & \underline{0000} \\
(0{,}9375)_{10} & 000.1111
\end{array}
$$

Interi con segno

- Per rappresentare gli interi relativi, a parità di cifre si dimezza l'intervallo dei valori assoluti
- Si utilizzano varie rappresentazioni

<u>Modulo e segno</u>

- un bit per il segno 0 : + 1 : –
- n-1 bit per il modulo
- intervallo $[-2^{n-1}+1, +2^{n-1}-1]$

ES

n = 4 bit intervallo [–7,+7]

5 = 0101 –5 = 1101

NB

- *intervallo simmetrico*
- *doppia rappresentazione dello zero*

Complemento a 1

- Si aggiunge uno 0 a sinistra
- I numeri positivi si rappresentano con il sistema posizionale
- Per *cambiare di segno si complementa il numerale bit a bit*
- I numerali positivi iniziano per 0, i negativi per 1
- Con n bit: $[-2^{n-1}+1, +2^{n-1}-1]$

ES

n = 4 bit intervallo [−7, +7]

5 = 0101

−5 = 1010

- *Complementare = cambiare segno*
- *Doppia rappresentazione dello 0*

Complemento a 2

- I positivi hanno la stessa rappresentazione che in complemento a 1
- I negativi si ottengono sommando 1 alla loro rappresentazione in complemento a 1
- Intervallo con n bit: $[-2^{n-1}, +2^{n-1}-1]$
- Regola pratica per complementare (cambiare segno al numerale):
  - *Partendo da destra si lasciano invariati tutti i bit fino al primo uno compreso, e poi si complementa bit a bit*

ES

n = 4 bit intervallo [−8, +7]

5 = 0101

−5 = 1011

- *Intervallo più esteso*
- *Una sola rappresentazione dello 0*
- *Complementare (a 2) = cambiare segno*

Rappresentazioni in CP1 e CP2

- Se il numero è *positivo*:
  a) determinare il numero di bit n
  b) rappresentare il numero in notazione a n bit
- Se il numero è *negativo*:
  a) determinare il numero di bit n
  b) rappresentare il numero positivo corrispondente in notazione a n bit
  c) complementare il numerale così ottenuto

ES rappresentare $(-347)_{10}$ in CP2

- $2^8 = 256 < 347 < 512 = 2^9$
- intervallo con n bit: $[-2^{n-1}, +2^{n-1}-1]$
- pertanto $n_{min}=10$
- +347 in notazione a 10 bit:

512	**256**	**128**	**64**	**32**	**16**	**8**	**4**	**2**	**1**
0	1	0	1	0	1	1	0	1	1

- complementando a 2:

-512	**256**	**128**	**64**	**32**	**16**	**8**	**4**	**2**	**1**
1	0	1	0	1	0	0	1	0	1

Eccesso 2^{n-1}

- I numeri vengono rappresentati come somma fra il numero dato e una potenza di 2, detta eccesso
- Con n bit l'eccesso è tipicamente 2^{n-1}
- Intervallo come CP2: $[-2^{n-1}, +2^{n-1}-1]$
- I numerali positivi iniziano per 1, i negativi per 0
- Regola pratica:
  - *I numerali si ottengono da quelli in CP2 complementando il bit più significativo*

ES

n=4 bit: eccesso 8, intervallo [-8,+7]

$-3 \quad -3+8=5 \quad : \quad 0101$

$+4 \quad +4+8=12 \quad : \quad 1100$

- *Intervallo asimmetrico*
- *Rappresentazione unica dello 0*

Rappresentazioni in eccesso 2^{n-1}

- Dato un numero *m* (positivo o negativo) determinare il numero minimo di cifre n_{min} necessarie
- Determinare l'eccesso corrispondente
- Sommare *m* all'eccesso e rappresentare il numero ottenuto

ES rappresentare $(-347)_{10}$ in eccesso 2^{n-1}

- $2^8 = 256 < 347 < 512 = 2^9$
- intervallo con n bit: $[-2^{n-1}, +2^{n-1}-1]$
- pertanto $n_{min} = 10$
- $512 - 347 = 165$
- $165 = 128+32+4+1$
- $(-347)_{10}$ in eccesso 2^9 è:

512	**256**	**128**	**64**	**32**	**16**	**8**	**4**	**2**	**1**
0	0	1	0	1	0	0	1	0	1

Rappresentazioni a confronto

Decimale	M&S	CP1	CP2	Ecc 8
+ 7	0111	0111	0111	1111
+ 6	0110	0110	0110	1110
+ 5	0101	0101	0101	1101
+ 4	0100	0100	0100	1100
+ 3	0011	0011	0011	1011
+ 2	0010	0010	0010	1010
+ 1	0001	0001	0001	1001
+ 0	0000	0000	0000	1000
– 0	1000	1111	---	---
– 1	1001	1110	1111	0111
– 2	1010	1101	1110	0110
– 3	1011	1100	1101	0101
– 4	1100	1011	1100	0100
– 5	1101	1010	1011	0011
– 6	1110	1001	1010	0010
– 7	1111	1000	1001	0001
– 8	---	---	1000	0000

Addizioni in complemento

- In CP2 somme e sottrazioni tra numerali sono gestite nello stesso modo, ma si deve ignorare il trabocco:

$$\begin{array}{l} 4\ + \\ 2\ = \\ \hline 6 \end{array} \qquad\qquad \begin{array}{l} 0100\ + \\ 0010\ = \\ \hline 0110 \end{array}$$

- Se i due operandi hanno segno diverso il risultato è sempre corretto:

$$\begin{array}{r} 4\ + \\ -1\ = \\ \hline 3 \end{array} \qquad\qquad \begin{array}{r} 0100\ + \\ 1111\ = \\ \hline \cancel{1}0011 \end{array}$$

- Se i due operandi hanno lo stesso segno e il risultato segno diverso c'è errore

$$\begin{array}{l} 6\ + \\ 3\ = \\ \hline 9 \end{array} \qquad\qquad \begin{array}{l} 0110\ + \\ 0011\ = \\ \hline 1001 \end{array}$$

(9 non è compreso nell'intervallo)

Altre operazioni su numeri con segno

- Per fare la differenza si complementa il sottraendo e si somma:

$$\begin{array}{r} 6\ - \\ 2 = \\ \hline 4 \end{array} \qquad\qquad 0010 \longrightarrow \begin{array}{r} 0110\ + \\ 1110\ = \\ \hline 0100 \end{array}$$

- Le moltiplicazioni si fanno tra i valori assoluti e alla fine, se necessario, si complementa:

$$\begin{array}{l} (11)_{10}\ \text{x} \\ (-5)_{10} \end{array} \qquad\qquad \begin{array}{r} 01011\ \text{x} \\ 00101\ = \\ \hline 01011 \\ 00000 \\ 01011 \\ 00000 \\ 00000 \\ \hline 00110111 \end{array}$$

$$(-55)_{10} \qquad\qquad 11001001$$

Notazione in base 16

- Per i numerali esadecimali occorrono 16 cifre {0,1,…,9,A,B,C,D,E,F}
- Conversione esadecimale-binario:
  - *Si fa corrispondere a ciascuna cifra esadecimale il gruppo di 4 bit che ne rappresenta il valore*
- Conversione binario-esadecimale:
  - *Partendo da destra si fa corrispondere a ciascun gruppo di 4 o meno cifre binarie la cifra esadecimale che ne rappresenta il valore*

ES

F	5	7	A	3	1
1111	0101	0111	1010	0011	0001

- Si usano spesso stringhe esadecimali per rappresentare stringhe binarie in forma compatta

Numerali e numeri

- Un numerale è solo una stringa di cifre
- Un numerale rappresenta un numero solo se si specifica un sistema di numerazione
- Lo stesso numerale rappresenta diversi numeri in diverse notazioni

ES la stringa 110100 rappresenta:

- Centodiecimilacento in base 10
- $(+52)_{10}$ in binario naturale
- $(-11)_{10}$ in complemento a 1
- $(-12)_{10}$ in complemento a 2
- $(+20)_{10}$ in eccesso 32
- In esadecimale un numero grandissimo

Notazione in virgola mobile

- Estende l’intervallo di numeri rappresentati a parità di cifre, rispetto alla notazione in virgola fissa
- Numeri reali rappresentati tramite una coppia di numeri <m,e>

$$n = m \cdot b^e$$

  - m : *mantissa* (normalizzata tra due potenze successive della base)

$$b^{i-1} \leq | m | < b^i$$

  - e : *esponente* intero con segno
- Sia m che e hanno un numero <u>finito</u> di cifre:
  - *Intervalli limitati*
  - *Errori di arrotondamento*

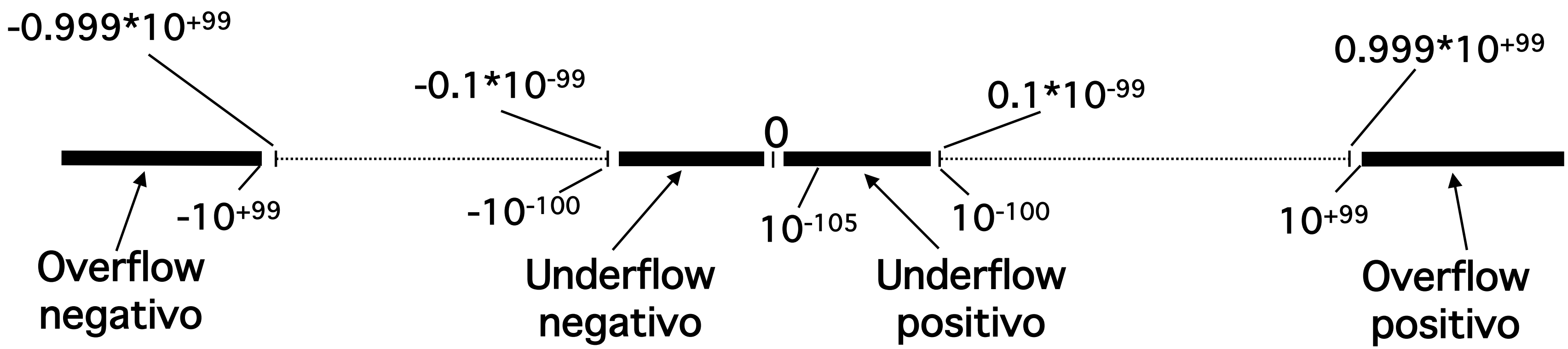
Esempio in base 10

- Numerali a 5 cifre ± .XXX ± EE
- Mantissa: 3 cifre con segno

$$0.1 \leq |m| < 1$$

- Esponente: 2 cifre con segno

$$-99 \leq e \leq +99$$



- Notare che con lo stesso numero di cifre in notazione a virgola fissa ± XXX .YY :
  - L'intervallo scende [-999.99,+999.99]
  - Ma si hanno 5 cifre significative invece di 3

Standard IEEE 754 (1985)

- Formato non proprietario cioè non dipendente dall'architettura
- Semplice precisione a 32 bit:

1	8	23
+/−	esp	mantissa

- Doppia precisione a 64 bit

1	11	52
+/−	esp	mantissa

- Notazioni con mantissa normalizzata e no
- Alcune configurazioni dell'esponente sono riservate

Standard IEEE 754 a 32 bit: numeri normalizzati

1	8	23
+/−	esp	mantissa

- Esponente: eccesso 127 [−127, +128] non si usano gli estremi, quindi:

$$-126 \leq e \leq 127$$

- Mantissa : rappresentata solo la parte frazionaria:

$$1 \leq m < 2$$

- **Intervallo numeri normalizzati** [2^{-126}, $\sim 2^{128}$]

- Uso delle configurazioni riservate:
  - *m* ed *e* tutti 0: rappresenta lo 0
  - *m* tutti 0 ed *e* tutti 1: *overflow*
  - $m \neq 0$ ed *e* tutti 1: *Not A Number*
  - $m \neq 0$ ed *e* tutti 0: *numero denormalizzato*

Standard IEEE normalizzati: estremi intervallo

- Più grande normalizzato $\sim 2^{128}$:

  X 11111110 11111111111111111111111

  $+/-$ 2^{127} $(1.11\ldots1)_2 \approx \sim 2$

- Più piccolo normalizzato 2^{-126}:

  X 00000001 00000000000000000000000

  $+/-$ 2^{-126} $(1.00\ldots0)_2 = 1$

Standard IEEE 754 a 32 bit: numeri denormalizzati

1	8	23
+/−	esp	mantissa

- Esponente
  - Uguale a 00000000
  - *e* vale convenzionalmente 2^{-126}
- Mantissa:
  - diversa da 0
  - $0 < m < 1$
- Intervallo di rappresentazione
  - $[2^{-126}\ 2^{-23} = 2^{-149}\ ,\ \sim 2^{-126}]$

Standard IEEE denormalizzati: estremi intervallo

- Più grande denormalizzato $\sim 2^{-126}$:

  X 00000000 11111111111111111111111

  +/- 2^{-126} $(0.11...1)_2 \approx \sim 1$

- Più piccolo denormalizzato 2^{-149}:

  X 00000000 00000000000000000000001

  +/- 2^{-126} $(0.00...1)_2 = 2^{-23}$

Addizioni in virgola mobile

- Per addizione e sottrazione occorre scalare le mantisse per eguagliare gli esponenti

ES

- $n_1 + n_2$

  n_1 : 0 10011001 00010111011100101100111

  n_2 : 0 10101010 11001100111000111000100

- $e_1 = (26)_{10}$, $e_2 = (43)_{10}$: occorre scalare n_1 di 17 posti

  n'_1: 0 10101010 00000000000000001000101 +

  n_2 : 0 10101010 11001100111000111000100

  0 10101010 11001100111001000001001

- Notare che l'addendo più piccolo perde cifre significative

Moltiplicazioni in virgola mobile

- Si moltiplicano le mantisse e si sommano algebricamente gli esponenti
- Se necessario si scala la mantissa per normalizzarla e si riaggiusta l'esponente

$\boxed{\textbf{ES}}$ $n_3 = n_1 \text{ x } n_2$

n_1 :   0 10011001 10010111011100101100111

n_2 :   1 10101010 10000000000000000000000

- $e_1 = (26)_{10}$, $e_2 = (43)_{10}$
- $e_1 + e_2 = (69)_{10}$ = 11000100
- $m_1 \text{ x } m_2$ = 10.0110001100101011110110101
- si scala la mantissa di un posto
- si aumenta di 1 l'esponente

n_3 :   1 11000101 00110001100101011101101

Errore assoluto e relativo

- Rappresentando un numero reale n in una notazione floating-point si commette un errore di approssimazione
- In realtà viene rappresentato un numero razionale n´ con un numero limitato di cifre significative
  - **Errore assoluto**: $e_A = n - n'$
  - **Errore relativo**: $e_R = e_A / n = (n - n') / n$

- *Se la mantissa è normalizzata l'errore relativo massimo è costante su tutto l'intervallo rappresentato ed è pari ad un'unità sull'ultima cifra rappresenta*

ES 10 cifre frazionarie $e_R = 2^{-10}$

- *Nelle notazioni non normalizzate l'errore relativo massimo non è costante*

Esempio 1: virgola mobile

- Rappresentazione binaria in virgola mobile a 16 bit:
  - 1 bit per il segno (0=positivo)
  - 8 bit per l'esponente, in eccesso 128
  - 7 bit per la parte frazionaria della mantissa normalizzata tra 1 e 2
- Calcolare gli estremi degli intervalli rappresentati, i numerali corrispondenti, e l'ordine di grandezza decimale assumendo che le configurazioni con tutti 0 e con tutti 1 siano riservate.
- Rappresentare in tale notazione:
  - il numero *m* rappresentato in compl. a 2 dai tre byte FF5AB9
  - il numero *n* rappresentato in compl. a 1 dai tre byte 13B472
- Calcolare l'errore relativo ed assoluto che si commette rappresentando i numero *m* ed *n* nella notazione data

Esempio 2: virgola mobile

- Rappresentazione binaria in virgola mobile a 16 bit:
  - 1 bit per il segno (0=positivo)
  - 8 bit per l'esponente, in eccesso 128 (configurazioni con tutti 0 e con tutti 1 riservate)
  - 7 bit per la parte frazionaria della mantissa normalizzata tra 1 e 2
- Dato il numero *m* rappresentato in tale notazione dai due byte C3A5, calcolare l’intero *n* che approssima *m* per difetto, e rappresentarlo in complemento a 2 con 16 bit.

Esempio 3: virgola mobile

- Rappresentazione binaria in virgola mobile a 16 bit:
  - 1 bit per il segno (0=positivo)
  - *e* bit per l'esponente, in eccesso 2^{e-1}
  - 15–*e* bit per la parte decimale della mantissa normalizzata tra 1 e 2
  - configurazioni dell'esponente con tutti 0 e con tutti 1 riservate
- Calcolare il valore minimo e_{min} di bit per l'esponente che consenta di rappresentare il numero *n* rappresentato in complemento a 2 dai tre byte FF5AB9

Esempio 4: virgola mobile

- Rappresentazione binaria in virgola mobile a 16 bit:
  - 1 bit per il segno (0=positivo)
  - 7 bit per l'esponente, in eccesso 64
  - 8 bit per la parte decimale della mantissa normalizzata tra 1 e 2
  - configurazioni dell'esponente con tutti 0 e con tutti 1 riservate
- Dati *m* e *n* rappresentati in tale notazione dalle stringhe esadecimali FC53 e F8F2
- Calcolare la somma di *m* e *n* e fornire la stringa esadecimale che la rappresenta nella notazione suddetta
- Indicare l'eventuale errore assoluto che si commette
- Provare anche con FC53 e 78F2
- Provare anche con 7C53 e F8F2

Esercizio 5: virgola mobile

Si considerino i numeri m ed n che, nel sistema di rappresentazione in eccesso a 2^7, sono rappresentati rispettivamente dalle le stringhe esadecimali 63 e 93.

A. Calcolare il valore di s = m + n e rappresentare s nel sistema di rappresentazione in complemento a due su 12 bit.

B. Individuare una rappresentazione in virgola mobile che consenta di rappresentare il suddetto numero s con il numero minimo possibile di bit ed indicare l'intervallo di rappresentazione della rappresentazione individuata tenendo conto del fatto che le configurazioni dell'esponente composte da tutti 0 e da tutti 1 sono riservate;

C. Rappresentare, nella notazione in virgola mobile definita al punto B, i numeri decimali 0, -2 e 1,25 indicando gli eventuali errori di rappresentazione commessi;

D. Individuare il numero e di bit dell'esponente e il numero m di bit della mantissa di una notazione in virgola mobile a 16 bit che sia in grado di rappresentare tutti i numeri rappresentabili nella definita al punto A e che abbia l'intervallo di rappresentazione più grande possibile.

Esercizio 6: virgola mobile

Si consideri una rappresentazione binaria in virgola mobile a 12 bit, di cui (nell'ordine da sinistra a destra) 1 bit per il segno (0=positivo), e per l'esponente, che è rappresentato in eccesso a 2^{e-1}, e i rimanenti bit per la parte frazionaria della mantissa m che è normalizzata tra 1 e 2.

A. Calcolare il valore di e che consente di rappresentare, con la massima precisione possibile, numeri compresi in valore assoluto tra 1000 e 0,001;

B. tenendo conto del fatto che le configurazioni dell'esponente composte da tutti 0 e da tutti 1 sono riservate, indicare il più piccolo e il più grande numero che è possibile rappresentare nella notazione in virgola mobile definita al punto A specificando i numerali corrispondenti;

C. rappresentare nella notazione individuata al punto A il numero 512 e il numero -1, indicando gli eventuali errori assoluti che si commettono;

D. dato il numero n rappresentato nella notazione definita al punto A dalla stringa esadecimale D85, rappresentarlo: (a) in complemento a 2 su 8 bit e (b) in eccesso 2 9 su 10 bit.

Esercizio 7: virgola mobile

Si consideri una rappresentazione binaria in virgola mobile a 20 bit, di cui si usa: 1 per il segno (0=positivo), 7 per l'esponente, che è rappresentato in eccesso a 64, e 12 per la parte decimale della mantissa. Con valori dell'esponente diversi da 0000000 la mantissa è normalizzata tra 1 e 2 ($1 \leq man < 2$). Con esponente pari a 0000000 si rappresentano invece numeri denormalizzati, con esponente uguale a -63 e mantissa compresa tra 0 e 1 ($0 < man < 1$).

A. Calcolare l'ordine di grandezza decimale del più piccolo numero positivo normalizzato e del più grande numero positivo denormalizzato, rappresentabili nella notazione suddetta.

B. Dato il numero n rappresentato in complemento a 2 dai tre byte FF323B, ricavare il numerale che approssima meglio nella notazione suddetta il numero $m = n \times 2^{-85}$, esprimendolo come stringa esadecimale.

C. Calcolare gli ordini di grandezza sia binari che decimali dell'errore assoluto che si commette rappresentando m nella notazione suddetta.