Rappresentazione dell'informazione
Esercitazione

Prof. Alberto Borghese
Dipartimento di Informatica
alberto.borghese@unimi.it

Università degli Studi di Milano

Riferimenti al testo: Paragrafi 2.4, 2.9, 3.1, 3.2, 3.5 (codifica IEEE754)

Sommario

Esercizi sulla codifica dei numeri binari

Esercizi sulle operazioni con i numeri binari

Codifica IEEE754 dei numeri reali anche in forma denormalizzata.

Conversione da base n a base 10

Consideriamo un numero in base 2 (2 cifre: 0, 1)

Convertiamo in decimale il numero 1101 in base binaria.

$$\mathbf{1} \cdot 2^3 + \mathbf{1} \cdot 2^2 + \mathbf{0} \cdot 2^1 + \mathbf{1} \cdot 2^0 = \mathbf{8} + \mathbf{4} + 0 + \mathbf{1} = \mathbf{13_{10}}$$

Consideriamo un numero in base 3 (3 cifre: 0, 1, 2)

Convertiamo in decimale il numero 2102 in base ternaria.

$$\mathbf{2} \cdot 3^3 + \mathbf{1} \cdot 3^2 + \mathbf{0} \cdot 3^1 + \mathbf{2} \cdot 3^0 = \mathbf{54} + \mathbf{9} + 0 + \mathbf{2} = \mathbf{65_{10}}$$

Conversione da base Hex a base decimale

Consideriamo un numero in base 16 (16 cifre: da 0 a F)

Convertiamo in decimale il numero 1BC8 in base 16.

$$\mathbf{1} \cdot 16^3 + \mathbf{11} \cdot 16^2 + \mathbf{12} \cdot 16^1 + \mathbf{8} \cdot 16^0 =$$
$$\mathbf{1} \cdot 4{,}096 + \mathbf{11} \cdot 256 + \mathbf{12} \cdot 16 + \mathbf{8} \cdot 1 =$$
$$4{,}096 + 2{,}816 + 192 + 8 = 7{,}112$$

Conversione da base Hex a binario

Consideriamo un numero in base 16
(16 cifre: da 0 a F)

Convertiamo in binario il numero 1BC8
in base 16.

Cifre esadecimali	Valori decimali	Equivalenti binari
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
A	10	1010
B	11	1011
C	12	1100
D	13	1101
E	14	1110
F	15	1111

Valori posizionali esadecimali	Valori decimali
16^{-3}	1/4096=0.000 244 140 625
16^{-2}	1/256=0.003 906 25
16^{-1}	1/16=0.062 5
16^{0}	1
16^{1}	16
16^{2}	256
16^{3}	4 096
16^{4}	65 536
16^{5}	1 048 576

1 B C 8

1 1011 1100 1000

0001 1011 1100 1000

16 cifre binarie (4x4)

1 B C 8

Conversione da base 10 a base 2 - I

N = 528

528 : 2 = 264 R = 0 (peso 2^0)

M = …. .0

Conversione da base 10 a base 2 - II

N = 528

528 : 2 = 264	R = 0 (peso 2^0)
264 : 2 = 132	R = 0 (peso 2^1)

M = …. 00

Conversione da base 10 a base 2 - III

N = 528

528 : 2 = 264	R = 0 (peso 2^0)
264 : 2 = 132	R = 0 (peso 2^1)
132 : 2 = 66	R = 0 (peso 2^2)

M = …. 000

Conversione da base 10 a base 2 - IV

N = 528

528 : 2 = 264	R = 0 (peso 2^0)
264 : 2 = 132	R = 0 (peso 2^1)
132 : 2 = 66	R = 0 (peso 2^2)
66 : 2 = 33	R = 0 (peso 2^3)

M = …. 0000

Conversione da base 10 a base 2 - V

N = 528

528 : 2 = 264	R = 0 (peso 2^0)
264 : 2 = 132	R = 0 (peso 2^1)
132 : 2 = 66	R = 0 (peso 2^2)
66 : 2 = 33	R = 0 (peso 2^3)
33 : 2 = 16	R = 1 (peso 2^4)

M = …. 10000

Conversione da base 10 a base 2 - VI

N = 528

528 : 2 = 264	R = 0 (peso 2^0)
264 : 2 = 132	R = 0 (peso 2^1)
132 : 2 = 66	R = 0 (peso 2^2)
66 : 2 = 33	R = 0 (peso 2^3)
33 : 2 = 16	R = 1 (peso 2^4)
16 : 2 = 8	R = 0 (peso 2^5)

M = …. 010000

Conversione da base 10 a base 2 - VII

N = 528

528 : 2 = 264	R = 0 (peso 2^0)
264 : 2 = 132	R = 0 (peso 2^1)
132 : 2 = 66	R = 0 (peso 2^2)
66 : 2 = 33	R = 0 (peso 2^3)
33 : 2 = 16	R = 1 (peso 2^4)
16 : 2 = 8	R = 0 (peso 2^5)
8 : 2 = 4	R = 0 (peso 2^6)

M = …. 0010000

Conversione da base 10 a base 2 - VIII

N = 528

528 : 2 = 264	R = 0 (peso 2^0)
264 : 2 = 132	R = 0 (peso 2^1)
132 : 2 = 66	R = 0 (peso 2^2)
66 : 2 = 33	R = 0 (peso 2^3)
33 : 2 = 16	R = 1 (peso 2^4)
16 : 2 = 8	R = 0 (peso 2^5)
8 : 2 = 4	R = 0 (peso 2^6)
4 : 2 = 2	R = 0 (peso 2^7)

M = …. 00010000

Conversione da base 10 a base 2 - IX

N = 528

528 : 2 = 264	R = 0 (peso 2^0)
264 : 2 = 132	R = 0 (peso 2^1)
132 : 2 = 66	R = 0 (peso 2^2)
66 : 2 = 33	R = 0 (peso 2^3)
33 : 2 = 16	R = 1 (peso 2^4)
16 : 2 = 8	R = 0 (peso 2^5)
8 : 2 = 4	R = 0 (peso 2^6)
4 : 2 = 2	R = 0 (peso 2^7)
2 : 2 = 1	R = 0 (peso 2^8)

M = …. 000010000

Conversione da base 10 a base 2 - X

N = 528

528 : 2 = 264	R = 0 (peso 2^0)
264 : 2 = 132	R = 0 (peso 2^1)
132 : 2 = 66	R = 0 (peso 2^2)
66 : 2 = 33	R = 0 (peso 2^3)
33 : 2 = 16	R = 1 (peso 2^4)
16 : 2 = 8	R = 0 (peso 2^5)
8 : 2 = 4	R = 0 (peso 2^6)
4 : 2 = 2	R = 0 (peso 2^7)
2 : 2 = 1	R = 0 (peso 2^8)
1 : 2 = 0	R = 1 (peso 2^9)

M = 10 0001 0000 = 1 x 2^4 + 1 x 2^9 = = 16 + 512 = 528

Conversione da base 10 a base 3

- N = 528

528 : 3 = 176	R = 0 (peso 3^0)
176 : 3 = 58	R = 2 (peso 3^1)
58 : 3 = 19	R = 1 (peso 3^2)
19 : 3 = 6	R = 1 (peso 3^3)
6 : 3 = 2	R = 0 (peso 3^4)
2 : 3 = 0	R = 2 (peso 3^5)

M = 201120 = 2 x 3^5 + 1 x 3^3 + 1 x 3^2 + 2 x 3^1 =

2x243 + 1x27 + 1x9 + 2x3 = 486 + 27 + 9 + 6 = 528

Conversione numeri decimali

I numeri decimali sono i numeri che hanno una parte frazionaria. Sono un (piccolissimo) sottoinsieme dei numeri reali

Viene convertita separatamente la parte intera e la parte frazionaria.

Conversione da base 10 a base 2 – parte intera

N = **528**,125

528 : 2 = 264	R = 0 (peso 2^0)
264 : 2 = 132	R = 0 (peso 2^1)
132 : 2 = 66	R = 0 (peso 2^2)
66 : 2 = 33	R = 0 (peso 2^3)
33 : 2 = 16	R = 1 (peso 2^4)
16 : 2 = 8	R = 0 (peso 2^5)
8 : 2 = 4	R = 0 (peso 2^6)
4 : 2 = 2	R = 0 (peso 2^7)
2 : 2 = 1	R = 0 (peso 2^8)
1 : 2 = 0	R = 1 (peso 2^9)

M = 10 0001 0000 = 1 x 2^4 + 1 x 2^9 = = 16 + 512 = 528

Conversione da base 10 a base 2 – parte decimale

N = **528**,125

0,125 * 2	= 0 + 0.25	0 (peso 2^{-1})
0,25 * 2	= 0 + 0.50	0 (peso 2^{-2})
0,5*2	= 1 + 0	1 (peso 2^{-3})

M = 10 0001 0000,001

Errori di approssimazione

Esempio: $10{,}75_{10} = 1010{,}11_2$

10 : 2 => 0
5 : 2 => 1
2 : 2 => 0
1 : 2 => 1
1010,

0,75 *2 => 1
(1),50 *2 => 1
(1),00 =>
11

Esempio: $10{,}76_{10} = 1010{,}1100001\ldots_2$

0,76 *2 => $1x2^{-1}$
(1),52*2 => $1x2^{-2}$
(1),04*2 => $0x2^{-3}$
(0),08 *2 => $0x2^{-4}$
(0),16 *2 => $0x2^{-5}$
(0),32 *2 => $0x2^{-6}$
(0),64 *2 => $1x2^{-7}$ (2^{-7} = 0,0078125)
(1),28 ……….

Errori di approssimazione: arrotondamento e troncamento.

Con 7 bit di parte fraz, rappresento:
0,5+0,25+0,0078125 = 0,7578125
Errore = 0,0011875

Sottrazione in complemento a 2

N = (a-b) = (18-21) in base 10 su 8 bit

a = 10010

18 : 2 = 9   R = 0
9 : 2 = 4   R = 1
4 : 2 = 2   R = 0
2 : 2 = 1   R = 0
1 : 2 = 0   R = 1

b= 10101

Scrivo a e b su 8 bit. Sono numeri positivi, copro le cifre alla sinistra del numero con 0 e ottengo:

a = 10010 => su 8 bit => 0001 0010

b = 10101 => su 8 bit => 0001 0101

Sottrazione in complemento a 2

N = (a-b) = (18-21) in base 10 su 8 bit

N = (a-b) = a + (-b) => calcolo –b in complemento a 2:

- a = 18, su 8 bit => 0001 0010
- Complemento a 1   1110 1101
- Complemento a 2   11101111   = -18

Eseguo la somma: 0001 0010 +
1110 1011=

1111 1101 = -3

1111 1101 => compl a 1 = 0000 0010 => compl a 2: +1 = 0000 0011

Sommario

Esercizi sulla codifica dei numeri binari

Esercizi sulle operazioni con i numeri binary

Codifica IEEE754 dei numeri reali anche in forma denormalizzata

Numeri decimali rappresentazione in fixed point

Numeri reali per il computer non sono i numeri reali per la matematica!!
E' meglio chiamarli float (numeri decimali), sono in numero finito.

Dato un certo numero di bit (stringa) per codificare un numero float, esistono due tipi di codifiche possibili:

Rappresentazione in fixed point.
La virgola è in posizione fissa all'interno della stringa.

Supponiamo di avere una stringa di 8 cifre, con virgola in 3a posizione:
27,35 = + | 27,35000
-18,7 = - | 18,70000
0,001456 = + | 00,00145(6)
928 = + | 28,00000 - perdo il 9 delle centianaia

Numeri decimali rappresentazione floating point

Rappresentazione come mantissa + esponente. $E = \sum_k c_k b_k = \sum_{k=-M}^{N} c_k B^k$

Esempio di **rappresentazioni equivalenti**:
627,35 = 62,735 x 10^1 = **6,2735 x 10^2 =** 0,62735 x 10^3 =
10^2 x (6x10^0+ 2x10^{-1}+7x10^{-2}+3x10^{-3}+5x10^{-4})

In grassetto viene evidenziata la rappresentazione normalizzata (una cifra prima della virgola).

Mantissa (significando)

+6,2735 x 10^2 **è costutuita da 3 componenti**

Esponente

Segno

Vengono rappresentati numeri molto grandi e molto piccoli.

Standard IEEE 754 (1980)

http://stevehollasch.com/cgindex/coding/ieeefloat.html

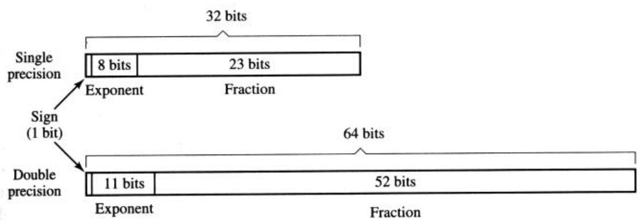


Figure 2-10 **Single-precision and double-precision IEEE 754 floating point formats.**

Rappresentazione **polarizzata** dell'esponente:

Polarizzazione pari a 127 per singola precisione =>
1 viene codificato come 1000 0000.

Polarizzazione pari a 1023 in doppia precisione.
1 viene codificato come 1000 0000 000.

Codifica mediante lo standard IEEE 754

Esempio: $N = -10{,}75_{10} = -1010{,}11_2$

Mantissa (significando)

Normalizzazione: ±1,xxxxxx

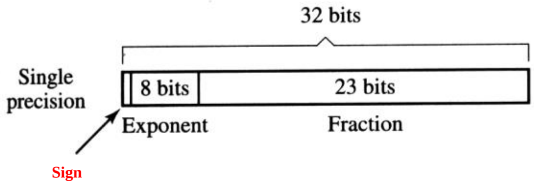
Esempio: -1,01011 x 2^3

Segno

Esponente

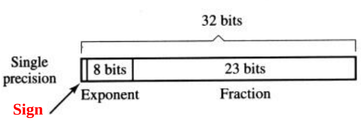
Parte frazionaria
(cifre dopo la virgola)

Devo inserire le informazioni di:
- Segno
- Mantissa
- Esponente



legate alla **codifica binaria**

Codifica mediante lo standard IEEE 754



Esempio: $N = -10{,}75_{10} = -1010{,}11_2$

1) Normalizzazione: ±1,xxxxxx Esempio: -1,01011 x 2^3

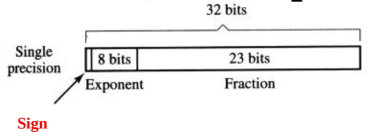
2) Codifica del segno 1 = - 0 = +

3) Calcolo dell'esponente, exp.

4) Completamento della scrittura della parte frazionaria su 23 bit

s	Exp+127	Parte Frazionaria
1	130	0101100000000000 …………00

Calcolo dell'esponente in notazione polarizzata



Esempio: $N = -10{,}75_{10} = -1010{,}11_2$
$= -1{,}01011_2 \times 2^3$

Calcolo dell'esponente, e, in rappresentazione polarizzata (si considerano solo 254 esponenti sui 256 possibili, compreso tra -126 e +127):

	Codifica	**Exp effettivo del numero**
	1111 1111 = 255 →	*Codifica riservata*
	1111 1110 = 254 →	+127
	1000 0010 = 130	**+3**
	1000 0001 = 129 →	+2
	1000 0000 = 128 →	+1
Polarizzazione:	0111 1111 = 127 →	0
	0111 1110 = 126 →	-1
	0000 0001 = 1 →	-126
	0000 0000 = 0 →	*Codifica riservata*

N = 1 | 1000 0010 | 0101 1000 0000 0000 0000 000

Configurazioni notevoli nello Standard IEEE 754

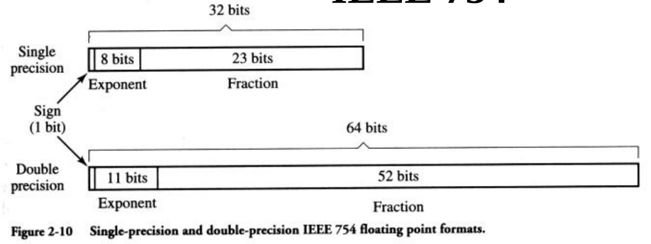


Figure 2-10 Single-precision and double-precision IEEE 754 floating point formats.

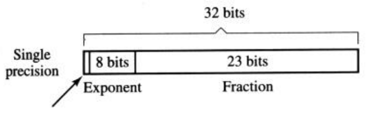
Configurazioni notevoli:

0	Mantissa: 0	Esponente: 00000000
$+\infty$	Mantissa: 0	Esponente: 11111111.
NaN	Mantissa: $\neq 0$.	Esponente: 11111111.

Range degli esponenti (8 bit): 1--254 => $-126 \leq exp \leq +127$.

Numeri float: $1.0 \times 2^{-126} = 1.175494351 \times 10^{-38} \div 3.402823466 \times 10^{38} = 1.1....11 \times 2^{127}$

Capacità dello Standard IEEE 754



Range degli esponenti (8 bit): 1--254 => $-126 \le exp \le +127$.
Minimo float (in valore assoluto!): 1.0×2^{-126}
Massimo float: $1.1111\ 1111\ 1111\ 1111\ 1111\ 111 \times 2^{+127}$
Capacità di rappresentazione di una variabile float in decimale:

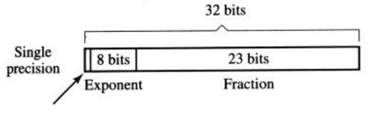
Minimo: $1.175494350822288 \times 10^{-38}$ (1.175494350822288e-038)
Massimo: $3.402823466385289 \times 10^{38}$ (3.402823466385289e+038)

Distanza tra Minimo_float e il float successivo è 2^{-149}
Distanza tra Minimo_float e 0 è 2^{-126} ($1.175494350822288 \times 10^{-38}$)
Discontinuità tra Minimo_float e zero.

Si può fare di meglio?
Numero denormalizzato Mantissa: $\neq 0$ Esponente: 00000000

Denormalizzazione nello Standard IEEE 754

Esempio di numero denormalizzato: $0{,}000001 \times 2^{-126}$



Range degli esponenti (8 bit): 1--254 => $-126 \le exp \le +127$.
Minimo float (in valore assoluto!): $1.0 \times 2^{-126} = 1.175494350822288 \times 10^{-38}$

Tuttavia abbiamo anche la mantissa a disposizione. Se troviamo una codifica per cui possiamo scrivere 0,0000 0000 0000 0000 00000 001, otteniamo un numero più piccolo (in valore assoluto!) pari a : $2^{(-23-126)} = 2^{-149}$ ($1.401298464324817 \times 10^{-45}$)
Discontinuità tra Minimo_float e 0 diventa 2^{-149}, la stessa che è presente tra Minimo_float e il float successivo.

Configurazioni notevoli:

0	Mantissa: 0	Esponente: 00000000
$+\infty$	Mantissa: 0	Esponente: 11111111.
NaN	Mantissa: $\neq 0$.	Esponente: 11111111.
Numero denormalizzato	Mantissa: $\neq 0$	Esponente: 00000000

Risoluzione della codifica dei reali

Distanza tra due numeri vicini.

Fixed point: Risoluzione fissa, pari al peso del bit meno significativo.

Esempio su 8 bit: +1111,101 la risoluzione per tutti i numeri sarà: $1 \times 2^{-3} = 0{,}125$

Floating point: Risoluzione ***relativa*** fissa, pari al peso del bit meno significativo.

Il bit meno significativo è in 23a posizione in singola precisione => 2^{-23}, ne consegue che la risoluzione sarà 2^{-23} volte il numero descritto.

Esempi:

$100,\ldots\ldots = 1{,}000 \times 2^2$ => La risoluzione sarà $2^{-23} \times 2^2 = 2^{-21}$

1.0×2^{-126} => La risoluzione sarà $2^{-23} \times 2^{-126} = 2^{-149}$

....

Conversione in IEEE754

N = 140,25

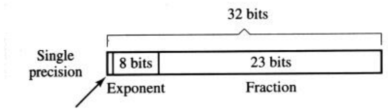
Converto in binario: 1000 1100,01

Normalizzo: $1{,}000110001 \times 2^7$

Detemino il segno (+) -> bit di segno = 0

Calcolo l'esponente polarizzato su 9 bit (127+7) = 1000 0100

Determino la parte frazionaria su 23 bit = 000110001 0000 ….000



Sommario

Sistema di numerazione binario

Rappresentazione binaria dell'Informazione

Conversione in e da un numero binario

Operazioni elementari su numeri binari: somma, sottrazione

I numeri decimali

Codifica IEEE754 dei numeri reali anche in forma denormalizzata.