Università degli Studi di Cassino

Corso di Fondamenti di Informatica

Rappresentazione dei dati numerici

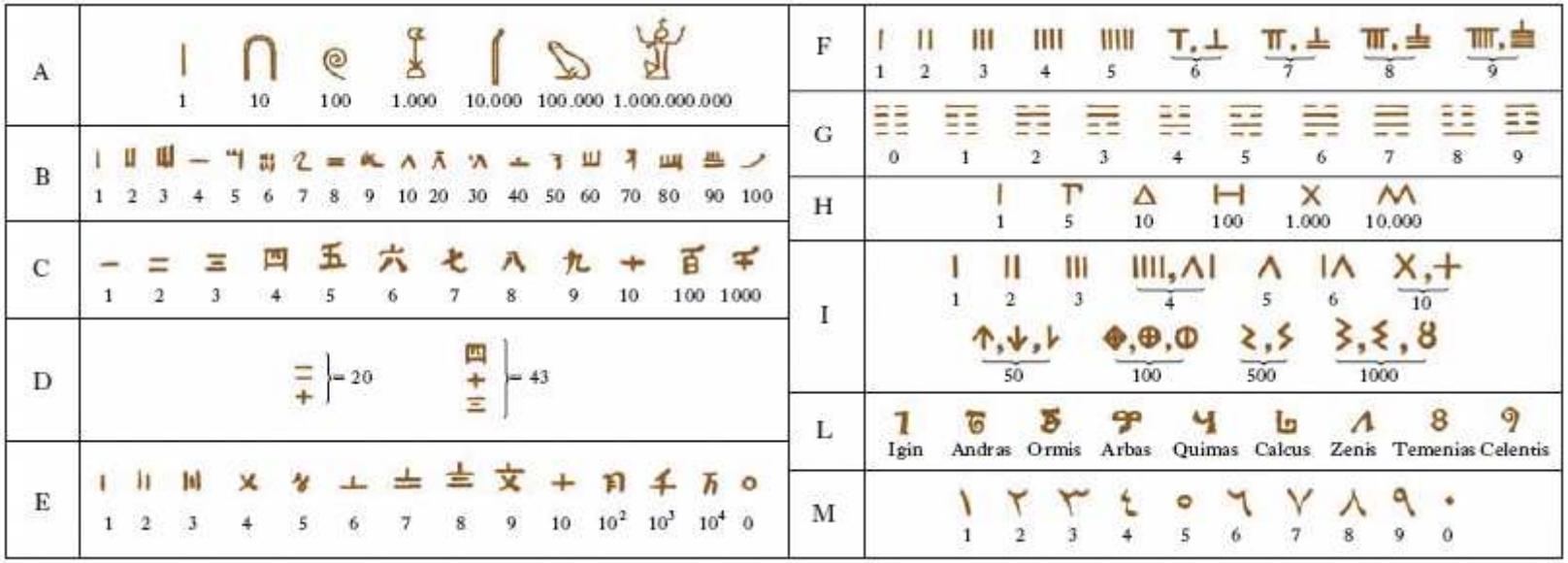
Aritmetica dei registri

Anno Accademico 2011/2012

Francesco Tortorella

Numero e rappresentazione

- Spesso si confonde il numero con la sua rappresentazione
- In quanti modi si possono rappresentare i numeri interi ?



Come rappresentiamo i numeri ?

- Rappresentazione ***posizionale***
- Base di numerazione: dieci
  - **Cifre: 0 1 2 3 4 5 6 7 8 9**
- Rappresentazione posizionale
  - **possibile per la presenza dello zero**

Esempio:

3201 =

$(3x10^3) + (2x10^2) + (0x10^1) + (1x10^0)$

In generale …

- Rappresentazione in base B → B cifre
  - 0 1 2 … B-1

- Rappresentazione dei numeri:
  - $d_{31}d_{30}$ ... $d_2d_1d_0$ è un numero a 32 cifre
  - valore =
    $d_{31} \times B^{31} + d_{30} \times B^{30} + \ldots + d_2 \times B^2 + d_1 \times B^1 + d_0 \times B^0$

Esempio: 'trentanove'

Base	Rappresentazione
10	39
8	47
4	213
2	100111
16	27

Quale base usare ?

- Decimale (base 10)
  - naturale per gli esseri umani.

- Binaria (base 2)
  - rappresentazione ottimale per il calcolatore

Conversione base 10 → base 2 (interi)

Come ottenere la rappresentazione in base 2 di un numero intero T rappresentato in base 10 ?

Supponiamo:

$$\mathbf{T=c_{n-1}x2^{n-1} + c_{n-2}x2^{n-2} + \ldots + c_2x2^2 + c_1x2^1 + c_0x2^0}$$

$$\mathbf{c_i \in \{0,1\}}$$

Non conosciamo:

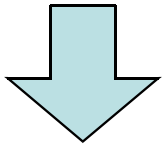
- le cifre c_i
- il numero di cifre n

Conversione base 10 → base 2 (interi)

$$T=c_{n-1}x2^{n-1} + c_{n-2}x2^{n-2} + \ldots + c_2x2^2 + c_1x2^1 + c_0x2^0 =$$

$$(c_{n-1}x2^{n-2} + c_{n-2}x2^{n-3} + \ldots + c_2x2^1 + c_1)\,x2 + c_0 =$$

$$Q_0x2 + c_0$$



$$Q_0 = T \text{ div } 2 \qquad c_0 = T \bmod 2$$

$$Q_0 = (c_{n-1}x2^{n-3} + c_{n-2}x2^{n-4} + \ldots + c_2)x2 + c_1 = Q_1x2 + c_1$$

$$Q_1 = Q_0 \text{ div } 2 \qquad c_1 = Q_0 \bmod 2$$

Aritmetica in base 2

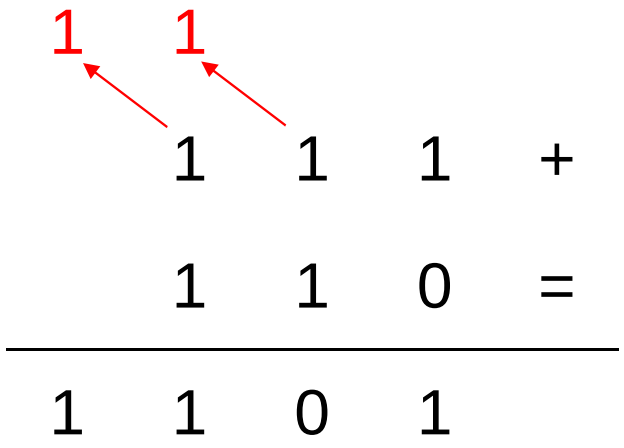
Le operazioni aritmetiche si svolgono in maniera analoga a quanto si fa in base 10.

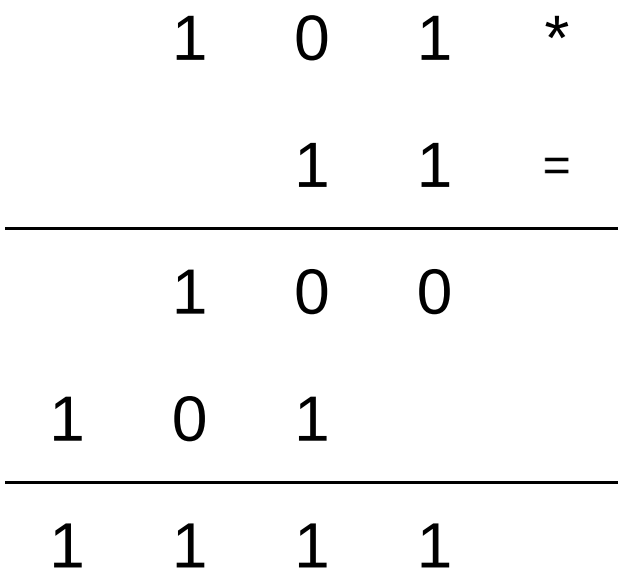
+	0	1
0	0	1
1	1	10

*	0	1
0	0	0
1	0	1

"tavola pitagorica" in base 2

Aritmetica in base 2





Aritmetica dei registri

- I registri di memoria sono supporti di lunghezza finita
- Ciò impone delle restrizioni all'insieme di numeri rappresentabili e, di conseguenza, dei vincoli all'aritmetica
- Registro a N bit → 2^N valori diversi rappresentabili
  - Es.: 8 bit → 256 valori
    possibile rappresentare l'intervallo [0,255]

Aritmetica dei registri

Non ci sono problemi nel caso in cui l'operazione produce un risultato rappresentabile nel registro

0	1	1	0	1	1	0	1	+
1	0	0	0	1	0	0	1	=
0	1	1	1	1	0	1	1	0

1	0	9	+
1	3	7	=
2	4	6	

Aritmetica dei registri

Se l'operazione fornisce un risultato R non rappresentabile, si produce un riporto uscente dal registro, mentre all'interno rimane una parte della rappresentazione del risultato (R mod 2^N)

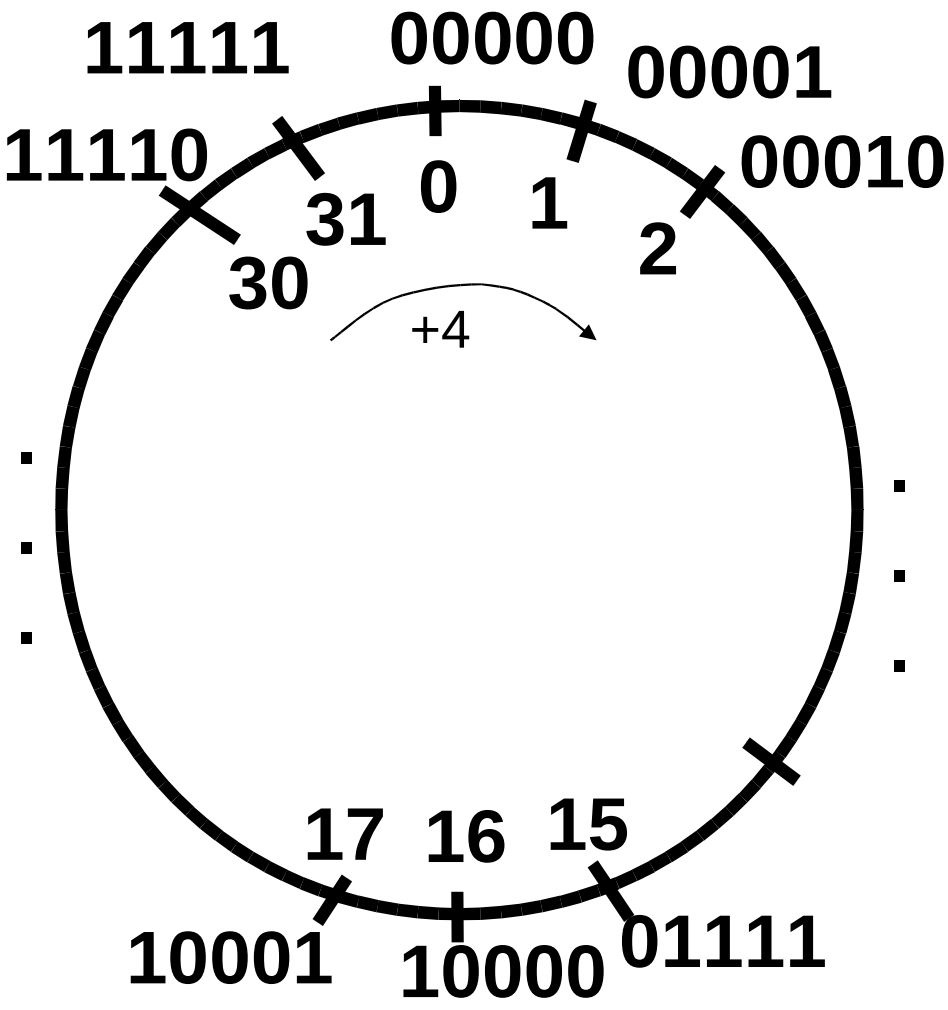
	1	1	1	0	1	1	0	1	+
	1	0	0	0	1	0	0	1	=
1	0	1	1	1	0	1	1	0	

2	3	7	+
1	3	7	=
1	1	8	
3	7	4	

Aritmetica dei registri

- L'aritmetica dei registri a N bit è caratterizzata da una congruenza mod 2^N

- Quindi, per N=5:
  - 30+4=2 !



Aritmetica dei registri

- Il riporto uscente dal registro, generato da un'addizione tra numeri interi, si definisce *carry*
- Il prestito uscente dal registro, generato da una sottrazione tra numeri interi, si definisce *borrow*

4	00100	10	01010	
+ 2	+ 00010	+ 26	11010	
6	0|00110	4	1|00100	carry

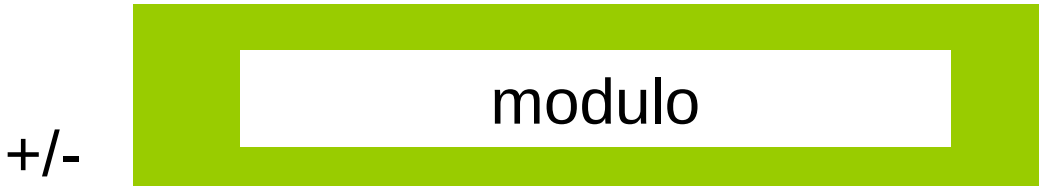
4	00100	10	01010	
- 2	- 00010	- 26	11010	
2	0|00010	16	1|10000	borrow

Rappresentazione dei numeri relativi

- Rappresentazione in segno e modulo
- Rappresentazione in complementi alla base
- Rappresentazione per eccessi

Rappresentazione dei numeri negativi

- Soluzione più immediata: segno + modulo



Possibile convenzione:

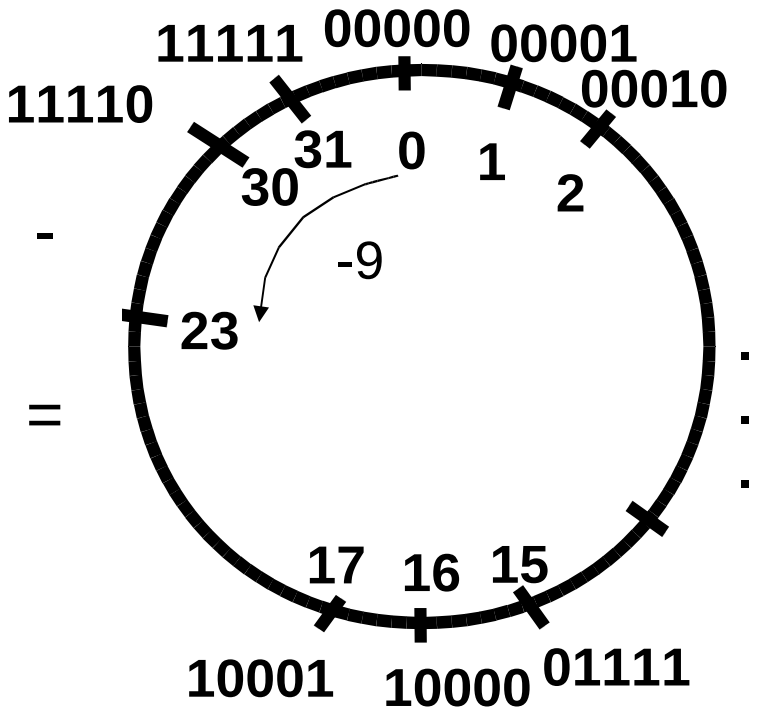
0 → + 1 → -

- Problemi
  - dove mettere il segno ?
  - doppia rappresentazione per lo zero (+0, -0)
  - operazioni alquanto complicate

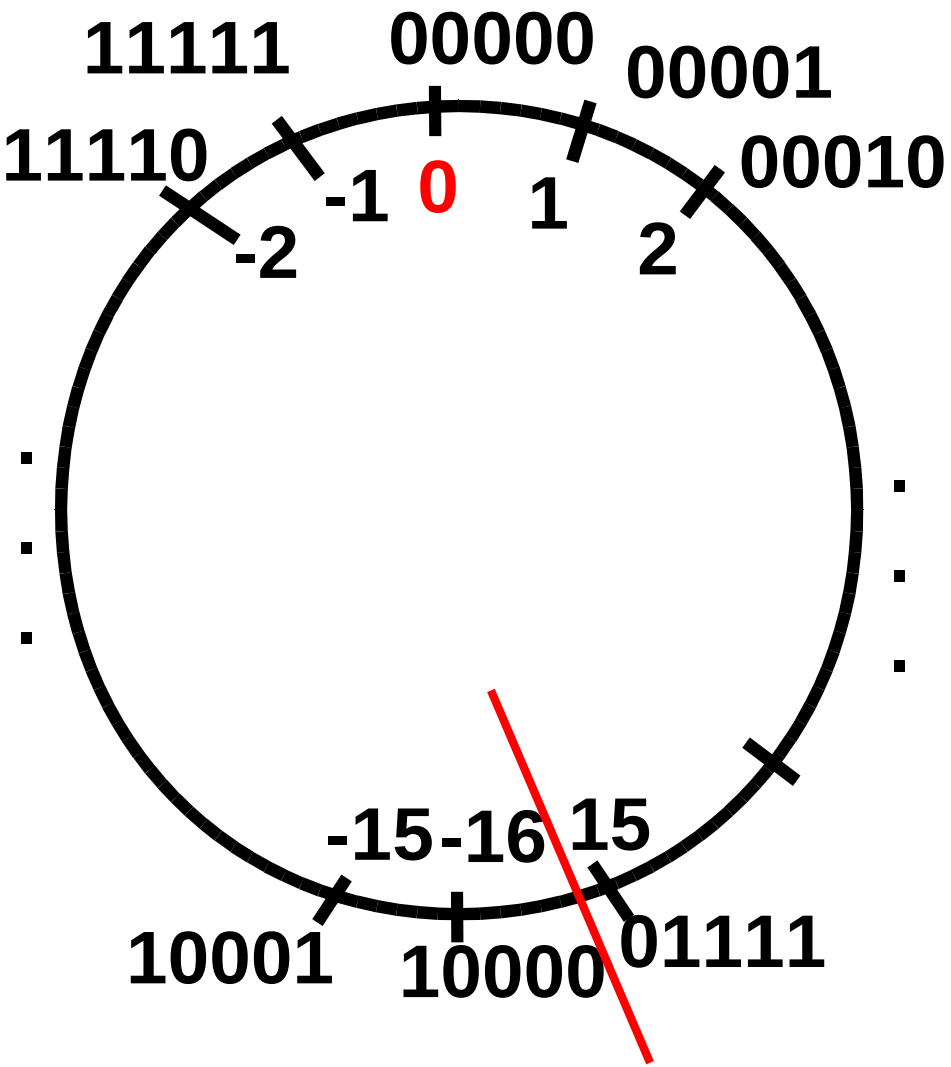
Rappresentazione dei numeri negativi

- Soluzione alternativa
  - Che cosa succede in un registro a N bit quando si sottrae un numero da 0 ?

	0	0	0	0	0
-					
	0	1	0	0	1
=					
1	1	1	0	1	1



Complementi alla base



Caratteristiche:

- 2^{N-1} non-negativi
- 2^{N-1} negativi
- uno zero
- quanti positivi ?
- confronto ?
- rappr. dello zero

Complementi alla base

- L'intervallo di numeri rappresentati è $[-2^{N-1}\ \ +2^{N-1}-1]$
- La rappresentazione di un numero x nell'intervallo è data da $R(x)=(x+2^N) \bmod 2^N$
- Il bit più significativo è indicativo del segno ("bit di segno")

00000	0
00001	+1
00010	+2
00011	+3
.	.
.	.
01110	+14
01111	+15
10000	-16
10001	-15
10010	-14
.	.
.	.
11101	-3
11110	-2
11111	-1

Calcolo rapido del complemento alla base

- Per ottenere rapidamente la rappresentazione in complemento alla base di un numero negativo su N bit
  - si estrae la rappresentazione del valore assoluto del numero su N bit
  - si complementano le cifre ad una ad una
  - si aggiunge 1
- Es.: complemento alla base su 8 bit di -33

  33_{10} =00100001 11011110+1=11011111

Operazioni in complemento alla base

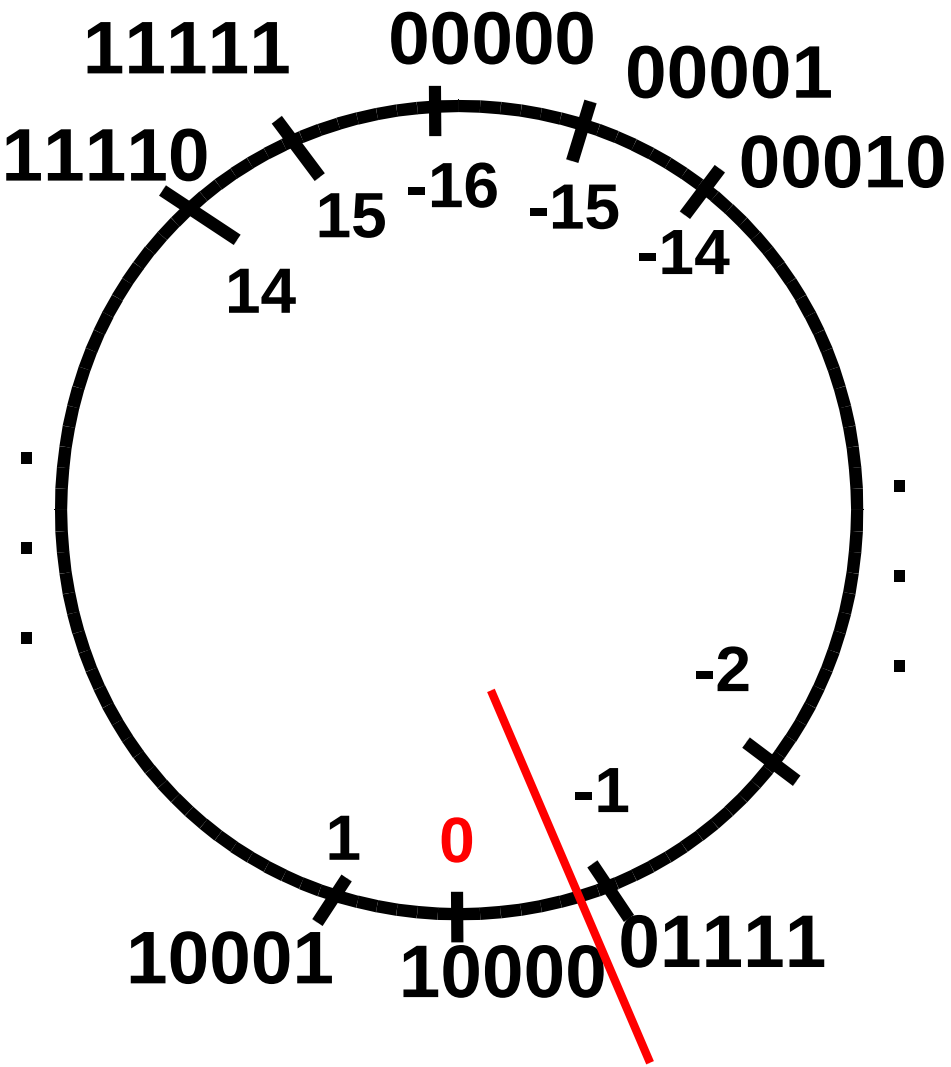
- Le addizioni si realizzano direttamente sulle rappresentazioni in quanto R(x+y)=R(x)+R(y)
- Anche le sottrazioni si valutano tramite addizioni, ponendo x-y come x+(-y); di conseguenza R(x-y)=R(x)+R(-y)
- Nel caso in cui l'operazione produce un numero al di fuori dell'intervallo di rappresentazione si ha un *overflow*

Operazioni in complemento alla base

+4	0100		+4	0100	
+2	+ 0010		-2	+1110	
+6	0|0110		+2	1|0010	
-4	1100		+5	0101	-6	1010
-2	+ 1110		+4	+0100	-3	+1101
-6	1|1010		-7	0|1001	+7	1|0111

overflow

Rappresentazione per eccessi (polarizzata)



Caratteristiche:

- 2^{N-1} non-negativi
- 2^{N-1} negativi
- uno zero
- quanti positivi ?
- confronto ?
- rappr. dello zero

Eccessi

- L'intervallo di numeri rappresentati è $[-2^{N-1} \; +2^{N-1}-1]$
- La rappresentazione di un numero x nell'intervallo è data da $R(x)=x+2^{N-1}$
- Il bit più significativo è indicativo del segno ("bit di segno")

00000	-16
00001	-15
00010	-14
00011	-13
.	.
.	.
01110	-2
01111	-1
10000	0
10001	+1
10010	+2
.	.
.	.
11101	+13
11110	+14
11111	+15

Numeri reali in base 2

- La rappresentazione dei numeri reali in base 2 è completamente analoga a quella in base 10:
  - Parte intera + parte frazionaria, separate da un punto
- La parte frazionaria è formata da cifre che pesano le potenze di 2 a esponente negativo.
  - Esempio: 110.0101_2 → $2^{+2}+2^{+1}+2^{-2}+2^{-4} = 6.3125$
- Conversione: si convertono separatamente la parte intera e quella frazionaria.
- Come si converte la parte frazionaria ?

Conversione base 10 → base 2 (frazionari)

Consideriamo un numero F minore di 1.

$$F = c_{-1}x2^{-1} + c_{-2}x2^{-2} + \ldots + c_{-n}x2^{-n} \qquad c_i \in \{0,1\}$$

$$Fx2 = c_{-1} + (c_{-2}x2^{-1} + \ldots + c_{-n}x2^{-(n-1)}) = c_{-1} + P_1 \quad P_1<1$$

$$P_1x2 = c_{-2} + (c_{-3}x2^{-1} + \ldots + c_{-n}x2^{-(n-2)}) = c_{-2} + P_2$$

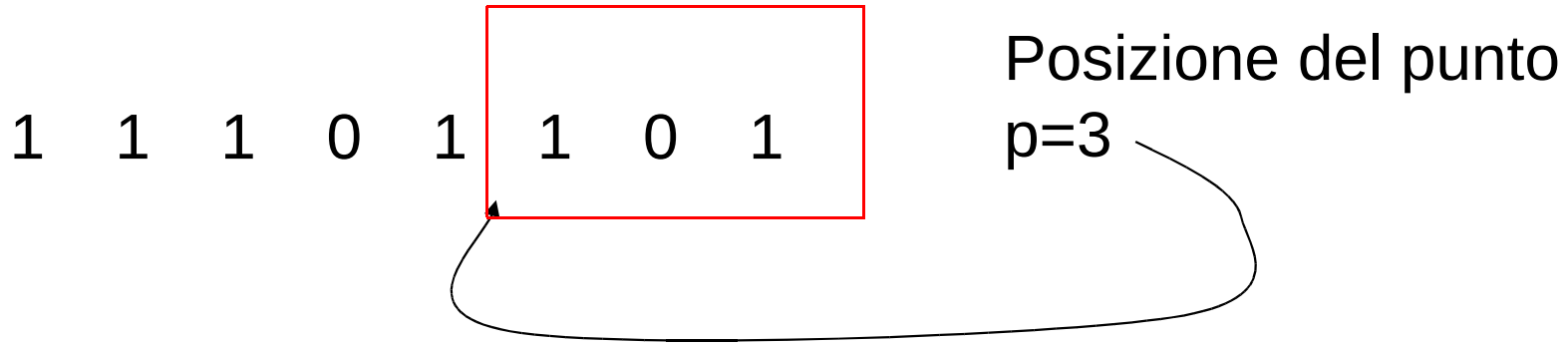
Rappresentazione nei registri dei numeri reali

- Come rappresentiamo 22.315 ?
- A differenza dei numeri interi, per rappresentare i numeri reali è necessario codificare la posizione del punto frazionario

- Due soluzioni:
  - Codifica esplicita $\rightarrow$ virgola fissa
  - Codifica implicita $\rightarrow$ virgola mobile

Rappresentazione in virgola fissa

- Con la codifica implicita, si assume prefissata la posizione del punto all'interno del registro → Rappresentazione in virgola fissa (fixed point)
- Esempio:



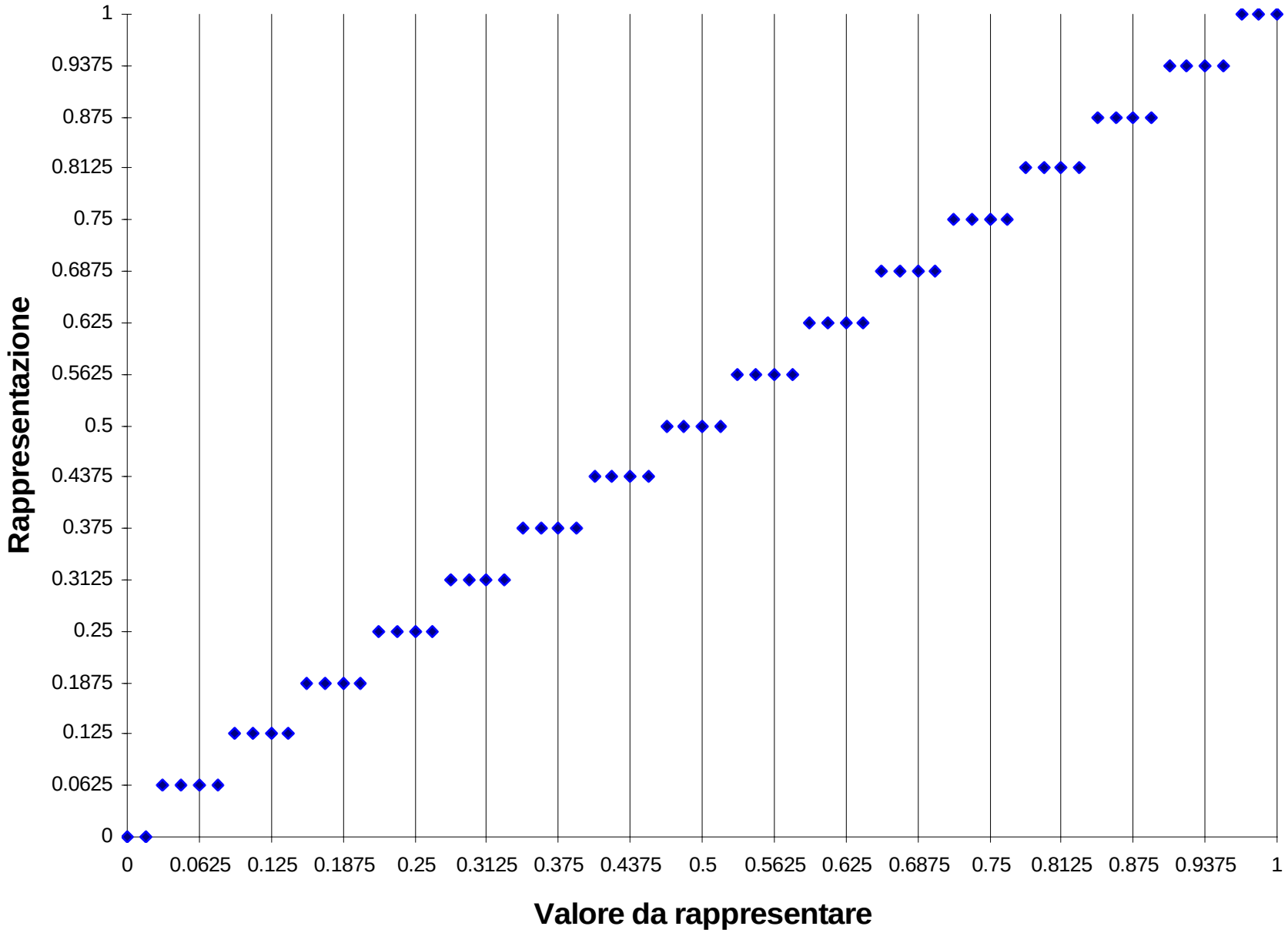
il numero rappresentato è 11101.101

Rappresentazione in virgola fissa

- Con questa convenzione, il valore X rappresentato nel registro è $K*2^{-p}$, dove K è il valore che otterremmo se interpretassimo come un intero il contenuto del registro.
- Qual è l'insieme dei valori rappresentabili su un registro a N bit ?

  K: $0,1,2,\ldots,2^N-1$ → X: $0, 2^{-p}, 2*2^{-p}, \ldots, (2^N-1)*2^{-p}$
- Esempio: N=8, p=4

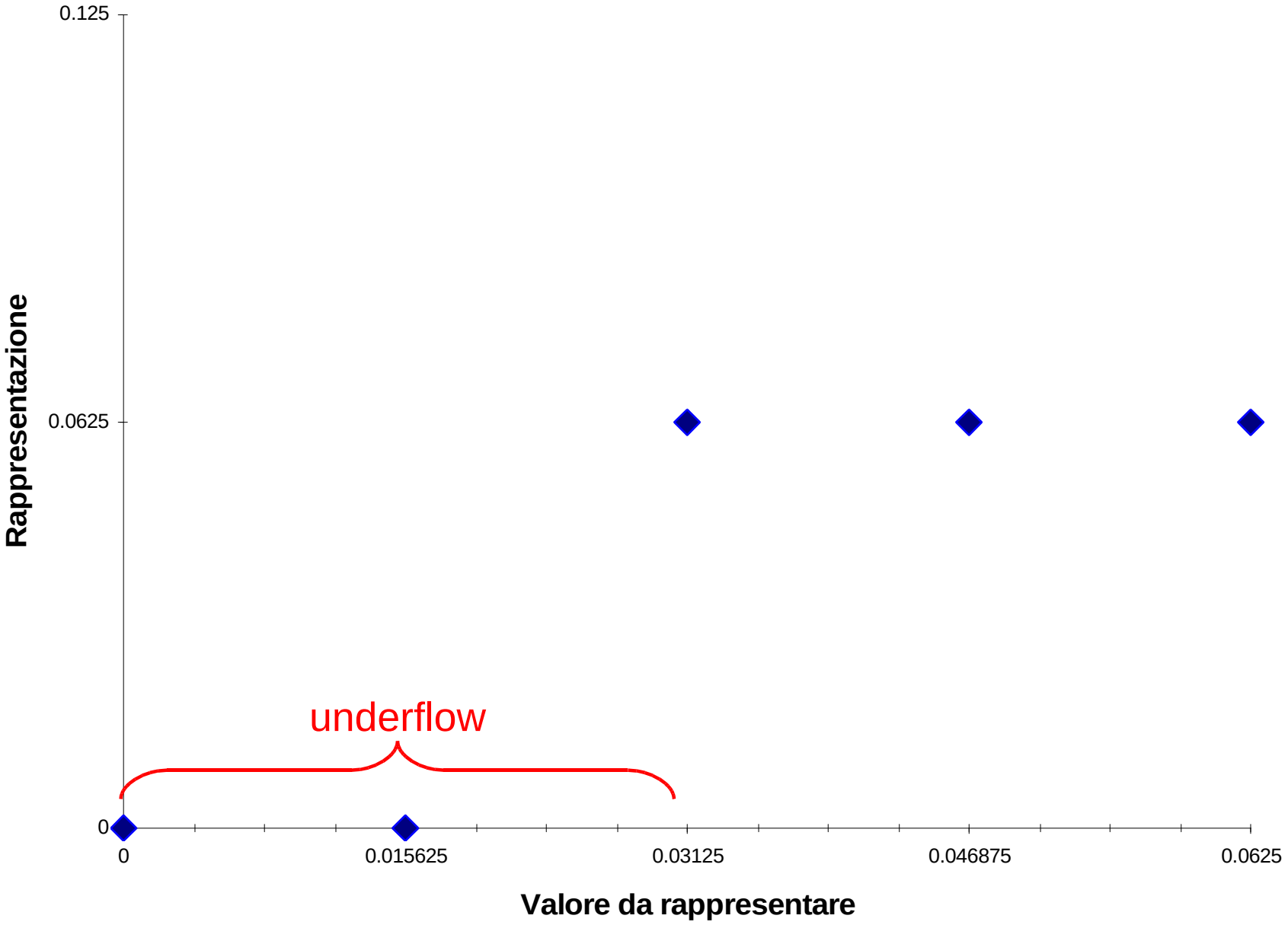
  X= 0, 0.0625, 0.125, 0.1875,…, 15.9375


Rappresentazione
1
0.9375
0.875
0.8125
0.75
0.6875
0.625
0.5625
0.5
0.4375
0.375
0.3125
0.25
0.1875
0.125
0.0625
0
0
0.0625
0.125
0.1875
0.25
0.3125
0.375
0.4375
0.5
0.5625
0.625
0.6875
0.75
0.8125
0.875
0.9375
1
Valore da rappresentare

Rappresentazione in virgola fissa

- I numeri sono rappresentati con una certa approssimazione
  - Esempio: tutti i valori compresi tra 0.03125 e 0.09375 sono rappresentati da 0.0625

- Tutti i valori compresi tra 0 e 0.03125 sono rappresentati da 0.0000 → *underflow*


0.125
0.0625
0
Rappresentazione
underflow
0
0.015625
0.03125
0.046875
0.0625
Valore da rappresentare

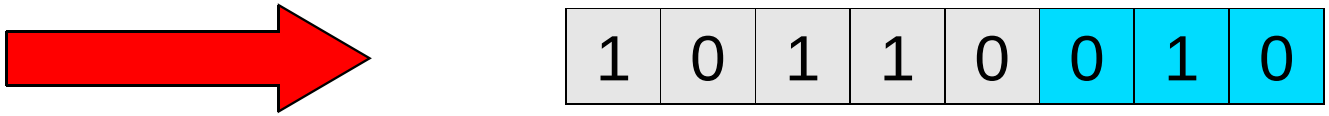
Rappresentazione di un numero in virgola fissa

Supponiamo di voler rappresentare il numero 22.315 in virgola fissa in un registro ad 8 bit con p=3.

Separiamo parte intera e parte frazionaria:

$22_{10} \rightarrow 10110_2$

$0.315_{10} \rightarrow 0.010100\ldots_2$



Riassumendo…

- La rappresentazione in virgola fissa ha innegabili vantaggi:
  - Semplicità
  - Piena compatibilità con la rappresentazione degli interi e possibilità di usare circuiti aritmetici comuni.
- Ma ha anche grossi problemi:
  - Errore relativo elevato per x →0
  - Compromesso range/precisione
  - Entrambi legati al fatto che il fattore di scala è fisso.

La virgola è mobile…

- Si potrebbero mitigare i problemi andando a rappresentare esplicitamente il fattore di scala.
- In questo modo la virgola non è più “fissa”, ma diventa “mobile”.
- Rappresentazione in virgola fissa → Rappresentazione in virgola mobile (floating point)

Rappresentazione in virgola mobile

- Fissata la base b, il valore viene considerato nella forma $M*b^E$ (notazione scientifica) ed è rappresentato tramite la coppia (M,E)

  Esempio: $22.315=0.22315*10^2$ → (0.22315,2)

  $10110.010=10.110010*2^3$ → (10.110010,11)

- Nel registro saranno quindi prefissate zone diverse per la mantissa e per l'esponente

Rappresentazione in virgola mobile

Come si rappresentano M ed E ?

- M
  - numero reale
  - segno e modulo
  - virgola fissa
- E
  - numero intero con segno
  - eccessi
- La disposizione nel registro facilita il confronto

Intervallo di numeri rappresentabili

- M rappresentato su m bit con p cifra frazionarie
  M: 0, 2^{-p}, $2*2^{-p}$, …, $(2^m-1)*2^{-p}$

- E rappresentato su e bit
  E: -2^{e-1},…,$+2^{e-1}-1$

- $N_{min}=M_{min}*2^{Emin} = 2^{-p}*2^{-2^{e-1}}$

- $N_{max}=M_{max}*2^{Emax} = (2^m-1)*2^{-p}*2^{+2^{e-1}-1}$

Intervallo di numeri rappresentabili

- Esempio:
  - m=23 p=23
  - e=8
- $N_{min} = 2^{-23} * 2^{-128} \cong 3.5 * 10^{-46}$

- $N_{max} = (2^{23}-1) * 2^{-23} * 2^{127} \cong 1.7 * 10^{+38}$

Esempio

Rappresentazione in FP di –12.6:

$12.6_{10} = 1100.\overline{1001}_2 = 0.11001\overline{1001} * 2^4$

Segno: 1

Mantissa: 0.11001001100110011001100

Esponente: $4+128 = 132_{10} = 10000100_2$

Rappresentazione normalizzata

- Con la virgola mobile non c'è unicità di rappresentazione:
  $N = M*2^E = (M*2)*2^{E-1} = (M*4)*2^{E-2} = (M/2)*2^{E+1}$

- Quale scegliere ? Quella che massimizza la precisione:
  prima cifra della mantissa diversa da 0
  → *rappresentazione normalizzata*

Rappresentazione normalizzata

- Esempio: N = 0.0003241892
  mantissa a 5 cifre decimali
- Diverse rappresentazioni possibili:

  $0.00032*10^{0}$

  $0.00324*10^{-1}$

  $0.03241*10^{-2}$

  $0.32418*10^{-3}$ ← normalizzata

Rappresentazione normalizzata

- L'intervallo di rappresentazione si modifica :
  $N_{min} = 2^{m-1} * 2^{-p} * 2^{-2^{e-1}}$

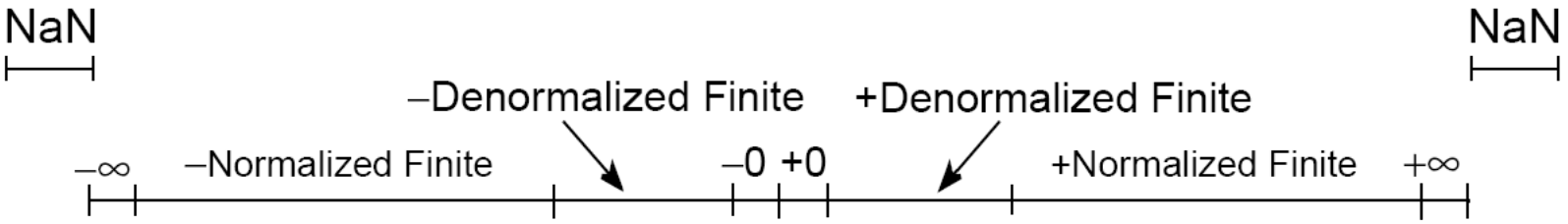
- Esempio:
  - m=23 p=23
  - e=8

- $N_{min} = 2^{-23} * 2^{-128} \cong 3.5 * 10^{-46}$ (non normalizzata)
- $N_{min} = 2^{22} * 2^{-23} * 2^{-128} \cong 1.5 * 10^{-39}$ (normalizzata)

Lo standard IEEE754

- Due formati
  - 32 bit: 23 bit mantissa + 8 bit esp. + 1 bit segno, bias=127
  - 64 bit: 52 bit mantissa + 11 bit esp. + 1 bit segno, bias=1023
- Mantissa con *hidden bit*
  $N = (-1)^s * (1.M) * 2^{E-bias}$
- Esponente polarizzato
  - Sono riservate le rappresentazioni dell'esponente 00…0 e 11…1
- Underflow graduale, denormalizzazione

Lo standard IEEE754



Real Number and NaN Encodings For 32-Bit Floating-Point Format

S	E	F		S	E	F
1	0	0	–0	0	0	0
1	0	$0.XXX^2$	–Denormalized Finite	0	0	$0.XXX^2$
1	1...254	Any Value	–Normalized Finite	0	1...254	Any Value
1	255	0	–∞	0	255	0
X^1	255	$1.0XX^2$	–SNaN	X^1	255	$1.0XX^2$
X^1	255	1.1XX	–QNaN	X^1	255	1.1XX

+0	+Denormalized Finite
+∞	+Normalized Finite
+SNaN	
+QNaN	

NOTES:
1. Sign bit ignored.
2. Fractions must be non-zero.

Lo standard IEEE754

	Range denormalizzato	Range normalizzato	Decimale
32 bit	Min: 2^{-149} Max:$(1-2^{-23})\times2^{-126}$	Min: 2^{-126} Max: $(2-2^{-23})\times2^{127}$	1.4×10^{-45} 3.4×10^{38}
64 bit	Min: 2^{-1074} Max:$(1-2^{-52})\times2^{-1022}$	Min: 2^{-1022} Max: $(2-2^{-52})\times2^{1023}$	4.9×10^{-324} 1.8×10^{308}

Lo standard IEEE 754

- Esistono rappresentazioni riservate (definite "numeri speciali") che permettono l'estensione dell'aritmetica a casi particolari:
  - NaN (0/0, sqrt(-2^k))
  - $+\infty$, $-\infty$

denormalizzato 

E	M	N
255	**≠0**	**NaN**
255	**=0**	**$(-1)^s \infty$**
1-254	**qualunque**	+/- numero fp
0	**0**	**0**
0	**≠0**	**$(-1)^s * 2^{-126} * (0.M)$**