Rappresentazione dei dati

Rappresentazione dei dati

Rappresentazione in base 2 e base 16
Aritmetica dei registri

Come rappresentiamo i numeri ?

- 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 …

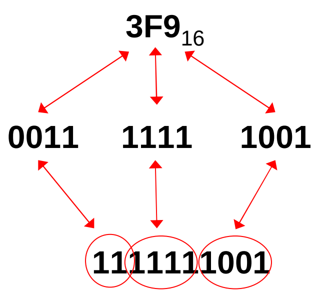
- Rappresentazione in base B → B-1 cifre
  - 0 1 2 … B-1

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

Altre basi

- B=2 :
  - cifre: 0 1
  - 1011010 →
    $1x2^6 + 0x2^5 + 1x2^4 + 1x2^3 + 0x2^2 + 1x2 + 0x1 =$
    $64 + 16 + 8 + 2 = 90$
    7 cifre binarie → 2 cifre decimali
- B=16 :
  - cifre: 0 1 2 3 4 5 6 7 8 9 A B C D E F
  - 524 →
    $5x16^2 + 2x16 + 4x1 = 1316$
    3 cifre esadecimali → 4 cifre decimali

Siccome $16=2^4$, il passaggio tra le rappresentazioni in base 2 e in base 16 è molto semplice:



base		
10	16	2
00	0	0000
01	1	0001
02	2	0010
03	3	0011
04	4	0100
05	5	0101
06	6	0110
07	7	0111
08	8	1000
09	9	1001
10	A	1010
11	B	1011
12	C	1100
13	D	1101
14	E	1110
15	F	1111

Quale base usare ?

- Decimale
  - naturale per gli esseri umani.
- Esadecimale
  - utile (agli esseri umani) per esaminare lunghe stringhe di bit
- Binaria
  - 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:

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

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

Non conosciamo:

- le cifre c_i
- il numero di cifre n

Conversione base 10 → base 2 (interi)

$T=c_{n-1}x2^{n-1} + c_{n-2}x2^{n-2} + \dots + c_2x2^2 + c_1x2^1 + c_0x2^0 =$
$(c_{n-1}x2^{n-2} + c_{n-2}x2^{n-3} + \dots + c_2x2^1 + c_1)\ x2 + c_0 =$
$Q_0x2 + c_0$

Q_0= T div 2 $\quad$ c_0 = T mod 2

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

$Q_1= Q_0$ div 2 $\quad$ $c_1 = Q_0$ mod 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	9	+
1	0	0	0	1	0	0	1	=		1	3	7	=
1	1	1	1	0	1	1	0			2	4	6	

F. Tortorella **Corso di Elementi di Informatica** **Università degli Studi di Cassino**

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)

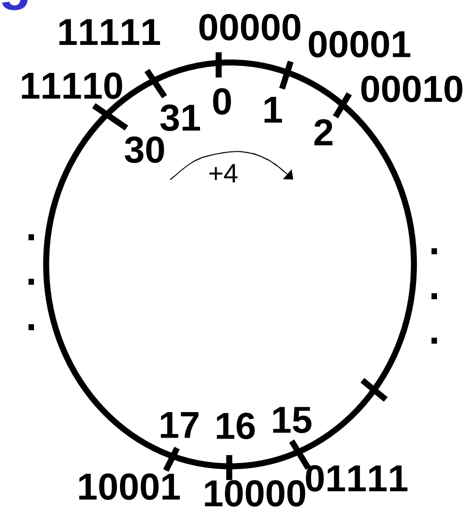
	1	1	1	0	1	1	0	1	+		2	3	7	+
	1	0	0	0	1	0	0	1	=		1	3	7	=
1	0	1	1	1	0	1	1	0			3	7	4	
											1	1	8	

F. Tortorella **Corso di Elementi di Informatica** **Università degli Studi di Cassino**

Aritmetica dei registri

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

- Quindi:
  - 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*

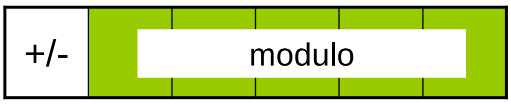
4	00100	10	01010	
+ 2	+ 00010	+ 26	11010	
6	0|00110	4	1|00100	carry
4	00100	10	01010	
- 2	- 00010	- 26	11010	
2	0|00010	16	1|10000	
borrow				

Rappresentazione dei dati

Rappresentazione in segno e modulo
Rappresentazione in complementi alla base

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

F. Tortorella | **Corso di Elementi di Informatica** | **Università degli Studi di Cassino**

Rappr. segno e modulo

- L'intervallo di numeri rappresentati è $[-2^{N-1} \;\; +2^{N-1}]$
- Lo zero è rappresentato due volte

00000	0
00001	+1
00010	+2
00011	+3
.	.
.	.
01110	+14
01111	+15
10000	0
10001	-1
10010	-2
.	.
.	.
11101	-13
11110	-14
11111	-15

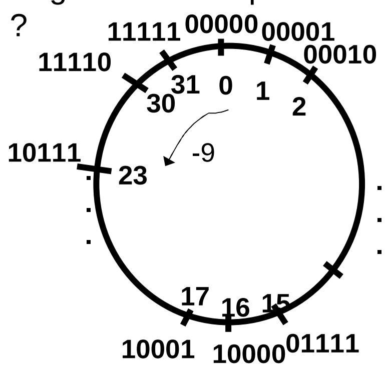
Rappresentazione dei numeri negativi

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

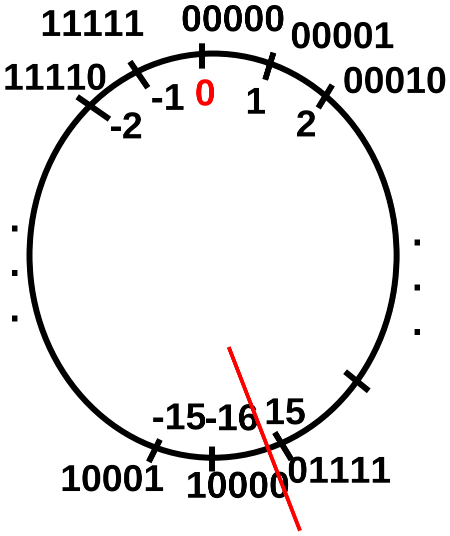
0 0 0 0 0 -

0 1 0 0 1 =

1 1 0 1 1 1



Complementi alla base



Caratteristiche:

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

F. Tortorella **Corso di Elementi di Informatica** **Università degli Studi di Cassino**

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

F. Tortorella **Corso di Elementi di Informatica** **Università degli Studi di Cassino**

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
+2	+ 0010
+6	0|0110

+4	0100
-2	+1110
+2	1|0010

-4	1100
-2	+ 1110
-6	1|1010

+5	0101
+4	+0100
-7	0|1001

-6	1010
-3	+1101
+7	1|0111

overflow

F. Tortorella **Corso di Elementi di Informatica** **Università degli Studi di Cassino**

Rappresentazione dei dati

Rappresentazione in virgola fissa

Conversione base 10 → base 2 (frazionari)

Consideriamo un numero F minore di 1.

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

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

$P_1<1$

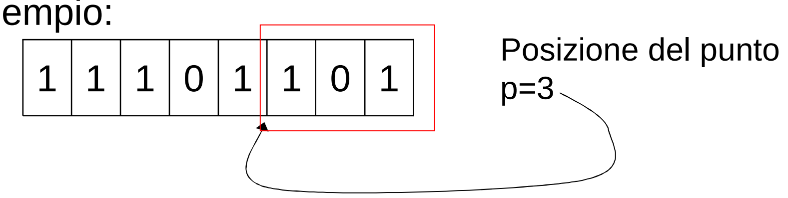
$P_1x2=c_{-2}+(c_{-3}x2^{-1} + \ldots + c_{-n}x2^{-(n-2)}) = c_{-2} + P_2$

Rappresentazione 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
  - Codifica implicita
- Con la codifica esplicita dovremmo rappresentare sia il numero che il suo fattore di scala → antieconomico e complicato

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:



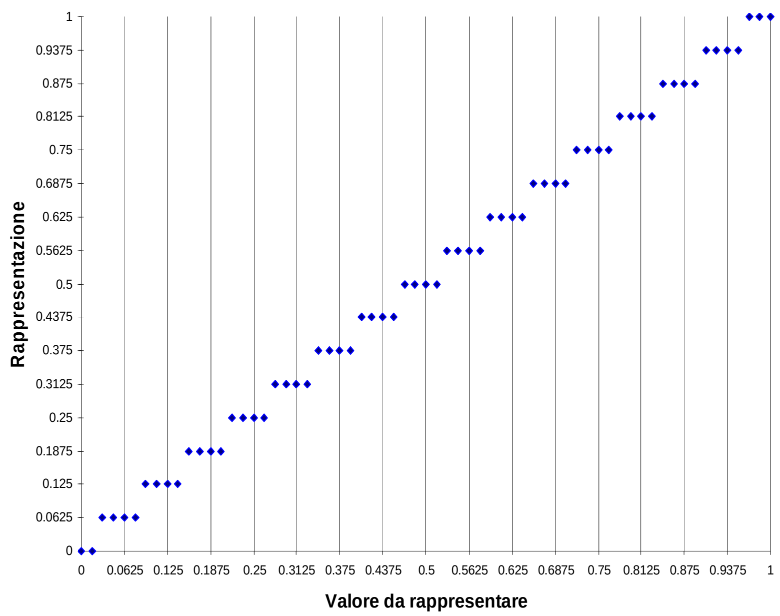
il numero rappresentato è 11101.101

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

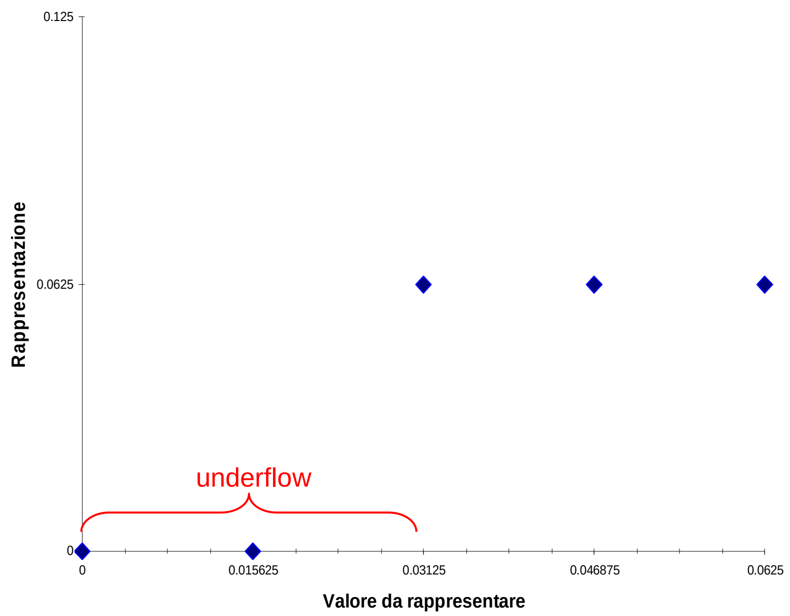
  X= 0, 0.0625, 0.125, 0.1875,…, 15.9375



Rappresentazione 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*



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:

$22_{10} \rightarrow 10110_2$

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

1	0	1	1	0	0	1	0

Precisione della virgola fissa

- Quantifichiamo l'errore assoluto:

  $Err_{max} = 2^{-p}/2 \rightarrow$ per p=4 $Err_{max} = 0.03125$

- Come fare per diminuire l'errore ?

  basta aumentare p, ma qual è l'effetto sul range dei numeri rappresentabili ?

  → compromesso tra range e precisione

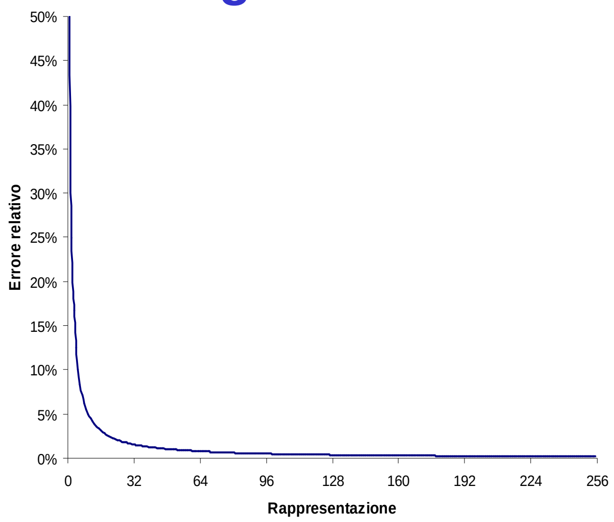
- Ricordiamo che X: $0, 2^{-p}, 2*2^{-p}, \ldots, (2^N-1)*2^{-p}$

Precisione della virgola fissa

Il problema vero è legato all'errore relativo:

$E_{rel}=Err_{max}/x$

Alternative ?



Rappresentazione dei dati

Rappresentazione in virgola mobile

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: $22.315=0.22315*10^2 \rightarrow (0.22315,2)$
  $10110.010=10.110010*2^3 \rightarrow (10.110010,11)$

- 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

+/-	**E**	**M**

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
- $N_{min} = 2^{-23}*2^{-128} \cong 3.5*10^{-46}$
- $N_{max} = (2^{23}-1)*2^{-23}*2^{127} \cong 1.7*10^{+38}$

Lo standard IEEE754

- Due formati
  - 32 bit: 23 bit mantissa + 8 bit esp. + 1 bit segno
  - 64 bit: 52 bit mantissa + 11 bit esp. + 1 bit segno
- Mantissa con *hidden bit*
  $N = (-1)^s * (1.M) * 2^{E-127}$
- Esponente polarizzato
  - I valori 0 e 255 sono riservati
- Intervallo di rappresentazione
  $1.8*10^{-38}$, $3.4*10^{38}$
- Underflow graduale, denormalizzazione

Lo standard IEEE 754

- Permette la rappresentazione di casi particolari:
  - NaN (0/0, sqrt(-2^k))
  - $+\infty$, $-\infty$

denormalizzato

E	M	N
255	$\neq$**0**	**NaN**
255	**=0**	$(-1)^s \infty$
0	**0**	**0**
0	$\neq$**0**	$(-1)^s * 2^{-126} * (0.M)$