Rappresentazioni numeriche

Introduzione

- Un calcolatore elettronico dispone di uno spazio finito per memorizzare le cifre che esprimono un valore numerico

- Esempio: disponiamo di p=3 cifre decimali
  L'insieme S di valori rappresentabili è S={0,..,999}

- Quali sono le differenze fra S e l'insieme dei numeri interi?
  - In generale si perdono le proprietà di chiusura delle operazioni
    - Ad esempio se a,b sono interi → a+b è un intero, ma non è detto che sia rappresentabile in S

Perdite di proprietà

Esempio: p=3 cifre decimali, valori rappresentabili
S={0,..,999}

- di chiusura dovuta ad overflow (risultato maggiore del valore massimo rappresentabile)
  - 600 + 600 = 1200 (1200 $\notin$ S)
  - 50 x 50 = 2500 ($\notin$ S)
- di chiusura dovuta ad underflow (risultato minore del valore minimo rappresentabile)
  - 3-5 = -2 ($\notin$ S)
- Perdita proprietà associativa
  - a+(b-c) $\neq$ (a+b)-c

    700 + (400-300) $\neq$ (700+400)-300; 700+400→ **Overflow**!
- Perdita proprietà distributiva
  - a x (b-c) $\neq$ a x b – a x c

Sistema di Numerazione Posizionale

- E' definito da una coppia (A,B)
  dove $B > 1$ è un intero, detto ***base*** del sistema,
  ed A un insieme di simboli distinti, le ***cifre***, con $|A| = B$,

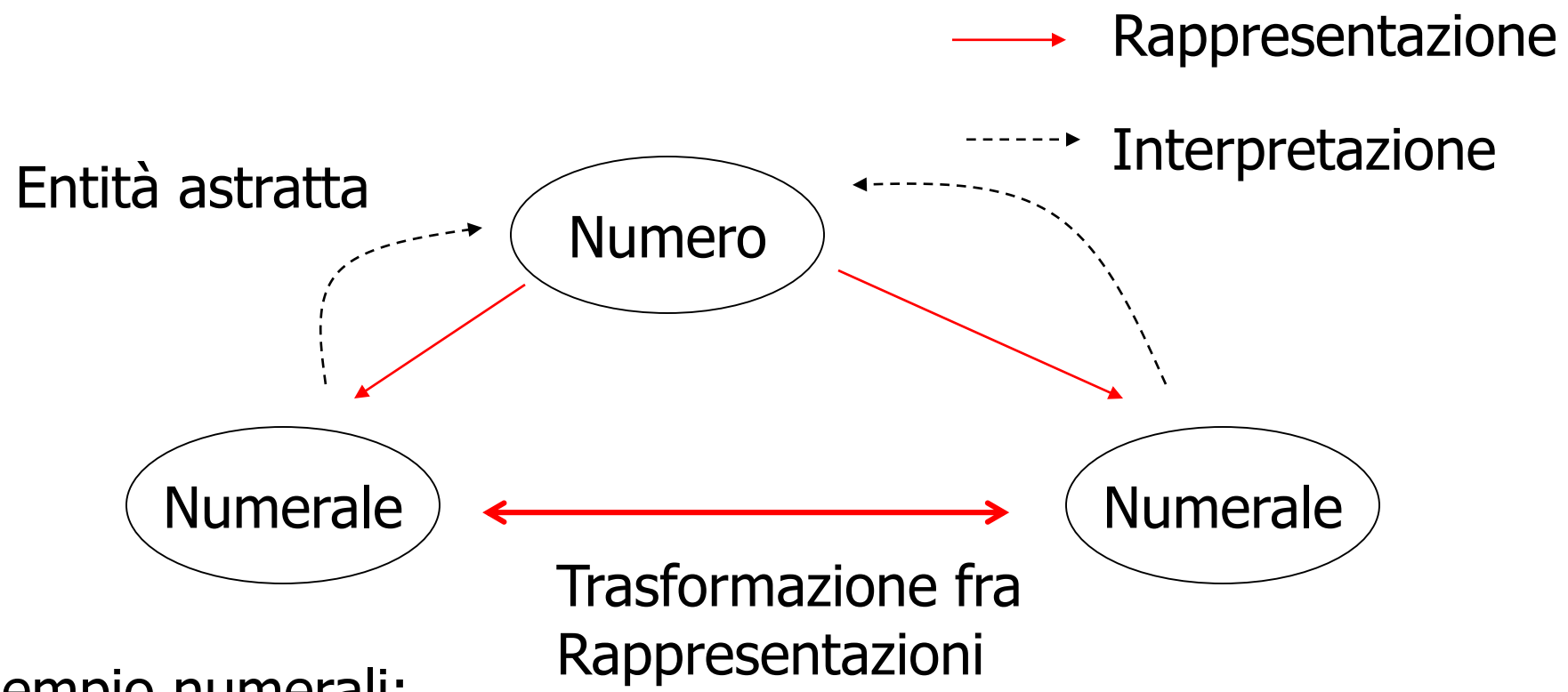
  Esempi di sistemi:

  - ***decimale,*** B =10, A={0,1,2,3,4,5,6,7,8,9}
  - ***binario,*** B =2, A={0,1}
  - ***ottale,*** B =8, A={0,1,2,3,4,5,6,7}
  - ***esadecimale,*** B =16, A ={0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F}

- Ogni cifra rappresenta un numero distinto compreso fra 0 e B -1

  Es: B=16:
  1→ "uno", 2→"due",.., A →"dieci", .., F →"quindici",

Numeri e numerali



Esempio numerali:
"dodici", 12, XII,

Analogia: *gatto* e *cat* denotano la stessa "entità"
in due lingue differenti

Sistema Numerazione Posizionale

- Un valore numerico è rappresento da una sequenza di cifre (rappresentazione o allineamento) appartenenti ad *A*

$$d_{k-1} \ldots d_2 d_1 d_0 \, . \, d_{-1} d_{-2} \ldots d_{-p}$$

- L'indice associato alla cifra denota la posizione della cifra che esprime il peso della cifra
  - Valore di d_i= $V(d_i) = d_i \times B^i$

PARTE-INTERA . PARTE-FRAZIONARIA

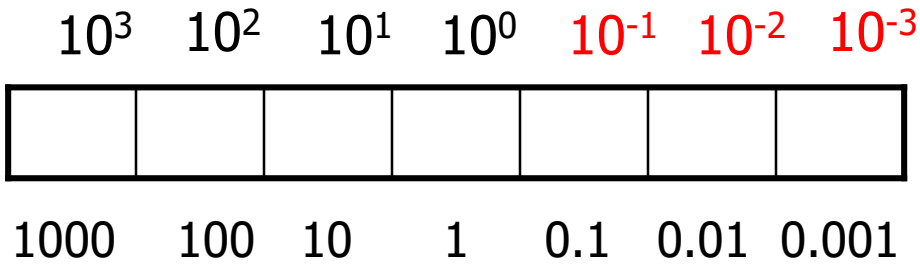
$$V(d_{k-1}..d_0.d_{-1}..d_{-p}) = \sum_{i=-p}^{i=k-1} d_i B^i$$

Esempio: sistema decimale (base 10)

$A=\{0,1,2,3,4,5,6,7,8,9\}$

Esempio: 743.234

- $d_2=7$, $d_1=4$, $d_0=3$, $d_{-1}=2$, $d_{-2}=3$, $d_{-3}=4$
- $V(734) = 7 \times 10^2 + 4 \times 10 + 3$
- $V(0.234) = 2 \times 10^{-1} + 3 \times 10^{-2} + 4 \times 10^{-3}$



Notazione

- Per evidenziare la base B del sistema di numerazione si usa la seguente notazione

  $(X)_B$ (si legge «X in base B»)

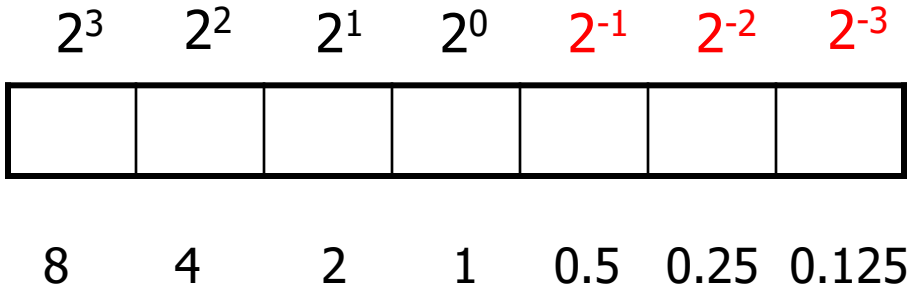
- Negli esempi seguenti, se omessa vale 10
- La cifra più a sinistra è detta cifra **più significativa**, quella a destra cifra **meno significativa**
  - Se B=2 si usano gli acronimi MSB (Most Significant Bit) ed LSB (Least Significant Bit)

Sistema Binario (base 2)

- Utilizzato dai circuiti elettronici dei calcolatori, 2 cifre (bit), $d \in A = \{0,1\}$

$$V(N) = d_{k-1} \times 2^{k-1} + d_{k-2} \times 2^{k-2} + \ldots\ldots + d_1 \times 2^1 + d_0 \times 2^0 + d_{-1} \times 2^{-1} + \ldots\ldots + d_{-p} \times 2^{-p}$$

$$(1010.101)_2 = 1 \times 2^3 + 1 \times 2^1 + 1 \times 2^{-1} + 1 \times 2^{-3} = (10.625)_{10}$$



Potenze di 2

- $2^2=4$
- $2^3=8$
- $2^4=16$
- $2^5=32$
- $2^6=64$
- $2^7=128$
- $2^8=256$
- $2^9=512$
- $2^{10}=1024$ (K) K=Kilo
- 2^{20} = 1024K (M) M=Mega
- 2^{30} = 1024M (G) G=Giga
- 2^{40} = 1024G (T) =Tera
- 2^{50} = 1024T (P) =Peta

- 2^{16} =65536 = 2^6 2^{10} = 64 K
- 2^{32}= 2^2 2^{30} = 4 G

osservazione : 1 Kb > 10^3bit,

tuttavia le **bande** dei bus-link di comunicazione vengono misurate in bits/sec in base decimale:

p.e. 1 Kb/s = 1000 b/s

ciò proviene dalla tradizione del mondo della trasmissione analogica

Base ottale ed esadecimale

Base ottale (8 o O)

$A=\{0,1,2,3,4,5,6,7\}$

Base esadecimale (16 o H)

$A=\{0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F\}$

Conversione da una base ad un'altra

- **Problema:** ***dato un valore rappresentato dall'allineamento N in base B1 trovare la rappresentazione N' in base B2***

$$(N)_{B1} \rightarrow (N')_{B2}$$

- Nel seguito, se chiaro dal contesto, N denota il valore
- Bisogna convertire separatamente le parti intera (N_I) e frazionaria (N_F)
  - $(N)_{B1} = (N_I \,.\, N_F)_{B1}$
  - $(N_I)_{B1} \rightarrow (N'_I)_{B2}$
  - $(N_F)_{B1} \rightarrow (N'_F)_{B2}$

Conversione

- Casi notevoli
  - B1≠10 e B2=10
  - B1=10 , B2≠10
- Poiché si ha familiarità con la base B=10 quando le due basi sono diverse da 10 conviene (più intuitivo) fare due trasformazioni successive:
  - da B1 a base 10
  - da base 10 a B2

Conversione da ***B*** (2, 8, 5) a Decimale

$(1010.101)_2 = 1 \times 2^3 + 1 \times 2^1 + 1 \times 2^{-1} + 1 \times 2^{-3} = (10.625)_{10}$

$(721)_8 \rightarrow 7 \times 8^2 + 2 \times 8^1 + 1 \times 8^0 = 7x64 + 16 + 1 = 448 + 17 = (465)_{10}$

$(134)_5 \rightarrow 1 \times 5^2 + 3 \times 5^1 + 4 \times 5^0 = 25 + 15 + 4 = = (44)_{10}$

Conversione da base 10 a *B* (N_I>0, intero)

Sia $(N_I)_{10}$ il valore in decimale dell'intero che vogliamo convertire in altra base, tale valore nella nuova base è pari a :

$$(N_I)_{10} = d_{k-1}B^{k-1} + .. + d_1B + d_0$$

Obiettivo: dobbiamo trovare i valori d_i nella nuova base *B*

$$(N'_I)_B = d_{k-1}B^{k-1} + .. + d_1B + d_0 = B(d_{k-1}B^{k-2} + .. + d_1) + d_0$$

quindi **dividendo** $(N_I)_{10}$ per B abbiamo che

d_0= è il resto e $d_{k-1}B^{k-2} + .. + d_1$ = è il quoziente

cioè

$d_0 = (N_I)_{10} \bmod B$, e $d_{k-1}B^{k-2} + .. + d_1 = (N_I)_{10} / B$

Conversione da base 10 a *B* (N_I>0, intero)... cont

- notare che:

$$d_{k-1}B^{k-2} + \ .. \ + d_1$$

è un intero, pertanto il resto della sua divisione con B ci fornisce d_1, cioè

$d_1 = ((N_I)_{10} / B) \bmod B$

- le altre cifre si identificano in modo analogo

Algoritmo di conversione da base 10 a B (N intero)

N intero in base 10 da convertire,
B base di arrivo

$i \leftarrow 0$;
while N<>0 **do**
1. $d_i \leftarrow$ N mod B
2. $N \leftarrow N/B$
3. $i \leftarrow i+1$

endwhile

Esempio: $(25)_{10} = (??)_2$

N	N / 2	N mod 2	Cifra
25	12	1	$d_0=1$
12	6	0	$d_1=0$
6	3	0	$d_2=0$
3	1	1	$d_3=1$
1	0	1	$d_4=1$

$(25)_{10} = (11001)_2$

Esempio

$(30)_{10} = (??)_{16}$

N	N / 16	N mod 16	Cifra
30	1	14	d_0=E
1	0	1	d_1=1

$(30)_{10} = (1E)_{16}$

N	N/2	N mod 2	Cifra
30	15	0	d_0=0
15	7	1	d_1=1
7	3	1	d_2=1
3	1	1	d_3=1
1	0	1	d_4=1

$(30)_{10} = (11110)_2$

Conversione da base 10 a B (Parte frazionaria)

Sia $(N_F)_{10}$ il valore in decimale della parte frazionaria che vogliamo convertire in altra base, tale valore è pari a :

$$(N_F)_{10} = d_{-1}B^{-1} + d_{-2}B^{-2} + . + d_{-m}B^{-m} + .. + \ldots$$

Moltiplicando per B

$$(N_F)_{10}\ B = d_{-1}B^{0} + d_{-2}B^{-1} .. + d_{-m}B^{-m+1} + \ldots + \ldots =$$

Quindi: $\mathbf{d_{-1}}$ = parte intera di N_FB (= trunc(N_FB))

$$N' = N_FB - d_{-1} = (d_{-2}B^{-1} .. + d_{-m}B^{-m+1} + .. + ..)$$

- Le altre cifre si identificano in modo analogo:
  d_{-2} = parte intera di N'B

- Finché precisione voluta *oppure* N=0

Algoritmo di conversione
da base 10 a *B* (0<N<1)

Esempio: $(0.8125)_{10} = (??)_2$

N<1 valore frazionario
da convertire,
B base di arrivo,
m cifre (precisione)

i←1;
while N<>0 **and** i≤m **do**
1. d_{-i} ← trunc (NB);
2. N←NB - d_{-i};
3. I←i+1

endwhile

N	2N	Trunc(2N)	Cifra
0.8125	1.625	1	$d_{-1}=1$
0.625	1.25	1	$d_{-2}=1$
0.25	0.5	0	$d_{-3}=0$
0. 5	1.0	1	$d_{-4}=1$
0			

$(0.8125)_{10} = (0.1101)_2$

Esempio: $(12.25)_{10} \rightarrow (..)_2$

- 12/2 = 6 resto 0 $\rightarrow d_0=0$
- 6/2 = 3 resto 0 $\rightarrow d_1=0$
- 3/2 = 1 resto 1 $\rightarrow d_2=1$
- 1/2 = 0 resto 1 $\rightarrow d_3=1$

- 0.25 x 2 = 0.50, parte intera 0 $\rightarrow d_{-1}=0$
- 0.50 x 2 = 1.0, parte intera 1 $\rightarrow d_{-2}=1$

$$(12.25)_{10} \rightarrow (1100.01)_2$$

Esempio numeri periodici

N	2N	Trunc(2N)	Cifra
0.2	0.4	0	$d_{-1}=0$
0.4	0.8	0	$d_{-2}=0$
0.8	1.6	1	$d_{-3}=1$
0.6	1.2	1	$d_{-4}=1$
0.2			

Esempio: $(0.\ 2)_{10} = (??)_2$

$(0.2)_{10} = (0.\overline{0011})_2$

Se un numero è periodico in base 10 allora lo è anche in base 2

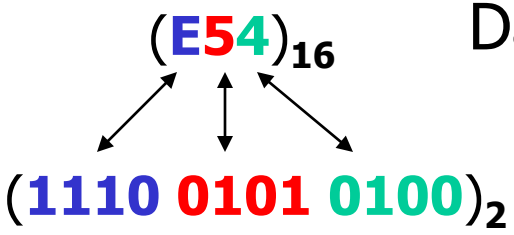
L'affermazione opposta non è vera

Altre basi notevoli

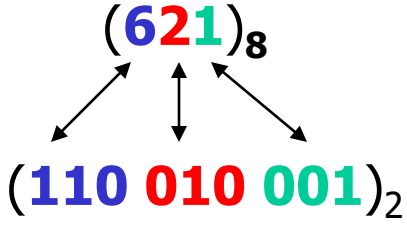
Basi 8 e 16

- Esempio:
  - $(721)_8 \rightarrow 7 \times 8^2 + 2 \times 8^1 + 1 \times 8^0 = 7x64 + 16 + 1 = 448 + 17 = (465)_{10}$
  - $(0.1)_8 \rightarrow 1/8 = (0.125)_{10}$
- Esempio:
  - $(721)_{16} \rightarrow 7x16^2 + 2x16^1 + 1x16^0 = 7x\ 256 + 32 + 1 = 1792 + 33 = (1825)_{10}$
  - $(0.1)_{16} \rightarrow 1/16 = (0.0625)_{10}$
- Nota: nel caso rappresentazioni esadecimali è prassi anteporre 0x, oppure il suffisso H
  - Ex: 0x721, 721H

Relazione fra le basi 2/8/16



Da base 16(2) a 2(16)



Da base 8(2) a 2(8)

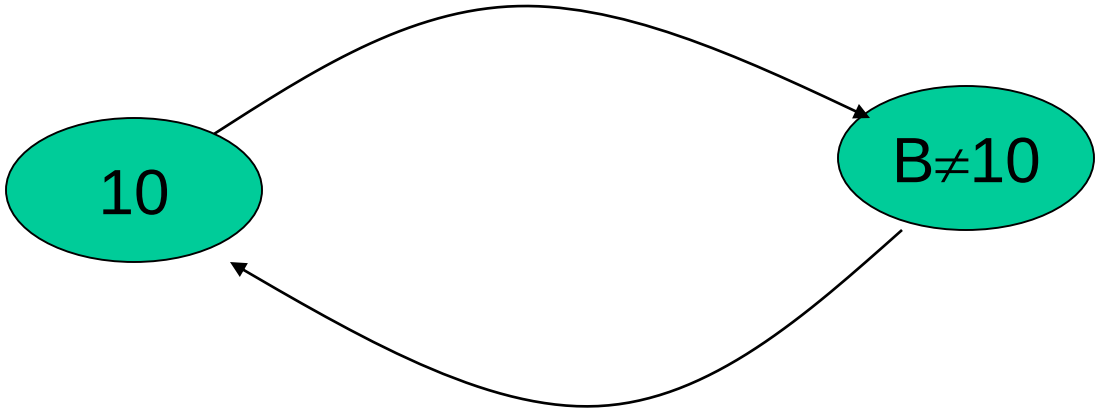
$(E54)_{16} \leftrightarrow (1110\ 0101\ 0100)_2 \leftrightarrow (111\ 001\ 010\ 100)_2 \leftrightarrow (7124)_8$

Da base 16(8) a 8(16)

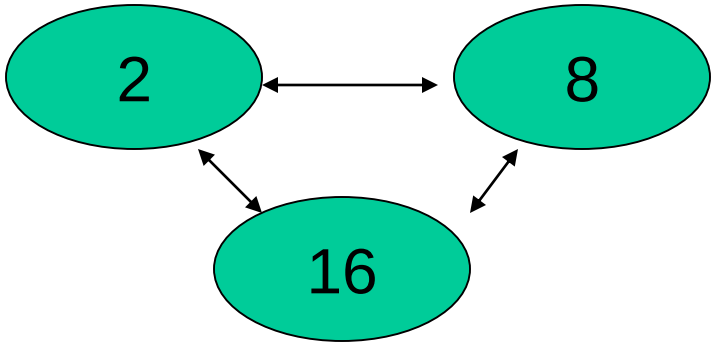
Riepilogo

divisioni successive (N intero)

prodotti successivi (N<1)



sviluppo del polinomio



Rappresentazione valori interi negativi

Esistono diversi metodi

- Modulo e segno
- Complemento a uno (obsoleto)
- Complemento a due
- Eccesso 2^{m-1}

Modulo e segno

- E' il più immediato da comprendere
  - si dedica un bit al segno ed i rimanenti bit al modulo
  - di norma 1 denota il segno "-"

- Esempio (quattro bit di cui tre per il numero e uno per il segno)

  - $-7 \rightarrow |7| = (111)_2 \rightarrow -7 = (1111)_2$
  - $7 \rightarrow (0111)_2$

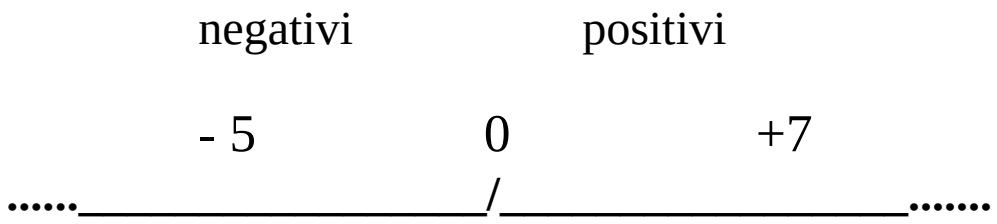
- Con k bit l'intervallo di dei valori rappresentabili è

$$S=[-2^{k-1}-1,..,2^{k-1}-1]$$

  - Doppia rappresentazione di 0

Modulo e segno

normale rappresentazione grafica che conosciamo



Rappresentazione con segno e modulo (caso con numero di bit limitato: 3 + 1 nell'esempio sottostante)

	positivi			negativi		
num dec	0	...	7	0	-1	....	-7
bit segno	0	...	0	1	1	...	1
	0	...	1	0	0	...	1
	0	...	1	0	0	...	1
	0	...	1	0	1	...	1
	+			-			

Complemento a 2 (complemento alla base – binaria nel caso)

Fissato un numero k>1 di cifre binarie, il complemento a 2 su k bit di un intero N, $N \in S=\{-2^{K-1}, ..2^{K-1}-1\}$, è

$$C(k,N)=\begin{cases} N & \text{per } 0 \leq N \leq 2^{K-1}-1 \text{ (num. positivi)} \\ 2^k-|N| & \text{per } -2^{K-1} \leq N \leq -1 \text{ (num. negativi)} \end{cases}$$

- Una definizione alternativa è $C(k,N) = (N + 2^k) \bmod 2^k$

Rappresentazione dei numeri in complemento a due (caso con k = 5)

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	-16	-15	-14	-13	-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1
MSB																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
LSB																															

Proprietà

Perché usare la rappresentazione in complemento?

Semplifica le operazioni aritmetiche

La **differenza** X – Y può essere calcolata mediante la **somma dei complementi**:

$$C(x-y)=C(x)+C(-y)$$

- In generale la somma algebrica diventa somma aritmetica
- Semplificazione dei circuiti elettronici che eseguono le operazioni (solo addizioni)

Calcolo del complemento a 2

Primo metodo:

- rappresentare il valore assoluto di N in base binaria
- invertire tutti i bit ed aggiungere 1

Esempio: rappresentare N=–25 in complemento su k=8 bit.

|-25| = 25 = 16+8+1

00011001		(25)
11100110	+	(Inverto i bit)
1	=	(sommo 1)
11100111		(231)

Secondo metodo:

- Rappresentare il valore assoluto di N in base 2
- Partendo da destra, lasciare invariati tutti i bit fino al primo bit 1, poi invertire gli altri

Valore espresso in base 2

- Il valore della stringa di bit $S=(b_{k-1}..b_2b_1b_0)$, supposto che essa esprima un numero in complemento a 2 su k bit, è

$$V(S)= - b_{k-1}2^{k-1} + \sum_{i=0}^{k-2} b_i 2^i$$

  - Pertanto
    - $b_{K-1} = 0 \Leftrightarrow$ numero positivo
    - $b_{K-1} = 1 \Leftrightarrow$ numero negativo
  - Attenzione, MSB non è un bit di segno!
    - Per ottenere il corrispondente valore di segno opposto non e' sufficiente invertire solo MSB

Altri esempi

Esempio: k = 4 bit

-2^3	2^2	2^1	2^0 ← Peso in decimale		
-8	4	2	1		
0	1	0	1	= 4+1 = 5	
1	1	0	1	=-8+4+1= -3	
0	0	0	1	= 1	più piccolo positivo
0	1	1	1	=4+2+1 = 7	più grande positivo
1	1	1	1	=-8+4+2+1=-1	più piccolo negativo
1	0	0	0	=-8	più grande negativo

Altri esempi

- k=8 bit, pesi=<-128,64,32,16,8,4,2,1>
- 11110000, rappresenta –128+64+32+16 = -16
- 10000000, rappresenta –128
- 11111111, rappresenta -128+64+32+16+8+4+2+1= -1
- 00000000, rappresenta 0

Rappresentazione dei numeri decimali in complemento alla base

2 digit a disposizione, quindi delle 100 configurazioni metà rappresentano numeri positivi (incluso zero) e metà negativi

Se N **positivo** C(N) = N

Se N **negativo** C(N)= 10^2 - /N/

N	**0**	**1**	**2**	**..............**	**48**	**49**	**–50**	**–49**	**–48**	**-1**
C(N)	**0**	**1**	**2..... ... ...**		**48**	**49**	**50**	**51**	**52 ....**	**99**

Differenza di numeri in complemento alla base (caso decimale)

La **differenza** X – Y può essere calcolata mediante la **somma dei complementi**:

$$C(x-y)=C(x)+C(-y)$$

Esempio: X-Y; X=21 e Y=23, con **k=2** cifre decimali a disposizione:

- C(21)=21, C(-23)=100-23=77
- C(21) + C(-23) = 21+77 = 98 = C(-2)

Ciò vale in generale :

- Se $Y>X$, ossia (**X-Y<0**), allora: $C(X-Y)=^{(def)} B^k - |X-Y| = B^k -(-(X-Y)) = B^k -Y+X$, ma per definizione ciò è uguale a C(X)+C(-Y)
- Il caso $Y \leq X$ verrà trattato fra breve
- Nota: In questo caso non può mai verificarsi overflow

Differenza di numeri in complemento

Eseguiamo ora la differenza fra X=23, Y=21, con **k=2** cifre decimali

- C(23)=23,
- C(-21)=100-21=79
- 23+79 = 1**02** = 2 + 100 = C(2) + 100,
    - ma essendoci solo due cifre il numero diventa **02**

Ciò vale in generale:

- Se $X \geq Y$, ossia $(X-Y \geq 0)$, allora: $C(X-Y) =^{(def)} X-Y$

- d'altra parte $C(X)+C(-Y) = X+B^k - Y \geq B^k$

- Pertanto $C(X-Y)=C(X)+C(-Y)$.. a meno di un fattore B^k

Esempio di calcolo del complemento alla base decimale

Fissiamo Base B=10, numero di digit k=2 → 10^2= 100

- X=23, Y=21
- -X-Y = ?

Algoritmo

1. Calcolo complemento di X, X'=C(-23)=100-23=77;
2. Calcolo complemento di Y, Y'=C(-21)=100-21=79
3. Eseguo la somma, X'+Y'=156
4. Sottraggo 100 se la somma è > 100: 156-100 = 56
5. Il risultato (56) è il complemento di -X-Y,

56 = C(-44)

Rappresentazione eccesso 2^{m-1}

- Il valore N viene rappresentato da **$N+2^{m-1}$**
- Si tratta di una traslazione dell'intervallo di rappresentabilità verso destra.
- Range di valori $[-2^{m-1}...2^{m-1}-1]$

Esempio di codifica eccesso $4= 2^2$ dei valori [- 4, 3]

-4 -3 -2 -1 0 1 2 3 (valore rappresentato in decimale)

0 1 2 3 4 5 6 7 (eccesso $4 = 2^2$)

Rappresentazione eccesso 2^{m-1} (cont.)

- Per passare dalla rappresentazione eccesso 2^{m-1} al **complemento** a **2** su m bit si deve invertire solo il bit MSB

Esempio precedente

-4	-3	-2	-1	0	1	2	3

rapp. eccesso 4 (rappresentato in binario)

0	0	0	0	1	1	1	1	(MSB)
0	0	1	1	0	0	1	1	
0	1	0	1	0	1	0	1	

rapp. compl. a 2

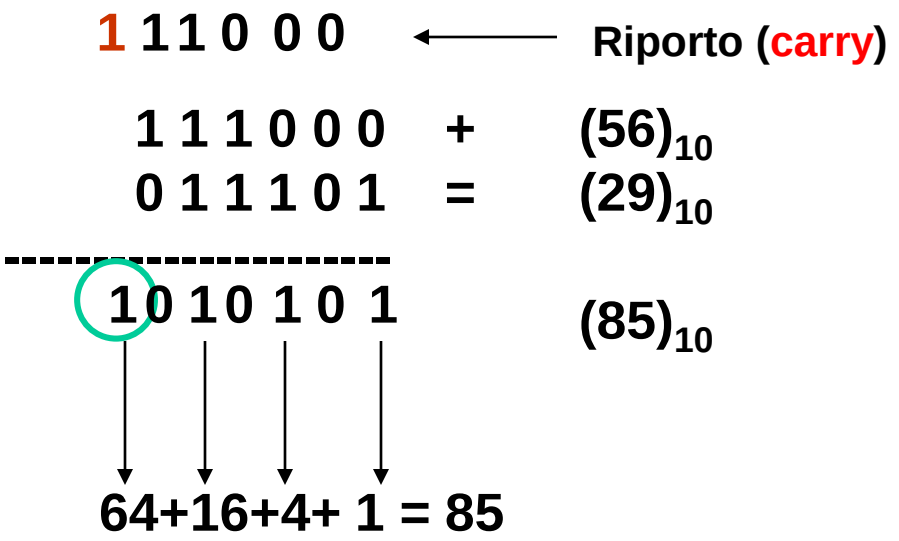
1	1	1	1	0	0	0	0	(MSB)
0	0	1	1	0	0	1	1	
0	1	0	1	0	1	0	1	

Operazioni aritmetiche

- Somma
- Sottrazione
- Prodotto
- Divisione

Somma binaria

- BASE B=2
- 0+0=0
- 0+1=1
- 1+0=1
- 1+1=10 $=(2)_{10}$
- 1+1+1=11 $=(3)_{10}$



La somma di due numeri a k bit e' rappresentabile al piu' con k+1 bit
Se abbiamo a disposizione k bit ed il risultato richiede k+1 bit si ha overflow

Regole per la somma

Somma di due bit A e B

Cin	A	B	Cout	Si	
0	0	0	0	0	
0	0	1	0	1	
0	1	0	0	1	
0	1	1	1	0	←
1	0	0	0	1	←
1	0	1	1	0	
1	1	0	1	0	
1	1	1	1	1	

Cout Cin +
A +
B =

Si

la coppia di valori (Cout,Si) indica il numero di “uno”, espresso in base 2

attenzione a due queste configurazioni …. sono le uniche in cui Cin<>Cout

In generale:
Cin e’ il il riporto (carry) generato dalla somma dei bit di peso i-1
Cout è il riporto generato dalla somma dei 2 bit A,B di peso i

Somma algebrica in complemento

- Esprimere gli operandi in complemento alla base
  - La rappresentazione in complemento differisce solo per i valori negativi
- Eseguire la somma
- Trascurare l'eventuale riporto
- Se non si è verificato overflow, allora la somma rappresenta il risultato espresso in complemento
- Si verifica overflow quando gli operandi hanno lo stesso segno ed il risultato ha segno opposto

Overflow, esempio

- Eseguire su k=4 bit la differenza: –3-6

|-3| → 2+1 → 0011 → 1101

|-6| → 4+2 → 0110 → 1010

1000	(riporti)
1101 +	(-3)
1010 =	(-6)
(1)0111	(7!)

Rilevazione overflow

Si verifica OVERFLOW se:

1) **i due operandi hanno lo stesso segno**
2) **Il risultato ha il segno diverso dagli operandi**

ma…….c'è anche un modo alternativo (usato nei circuiti addizionatori)

```
        1000                          0100
-3 +    1101          +5 +    0101
-6 =    1010          +6 =    0110
------- => ----------- ------- => -----------
-9      10111 => 7    +11     01011 => -5
```

……il verificarsi dell'overflow implica la disuguaglianza del riporto in ingresso e quello in uscita dalla posizione MSB (Cin<>Cout)

L'overflow si può rilevare testando la condizione "Cin<>Cout" di MSB

Estensione del segno

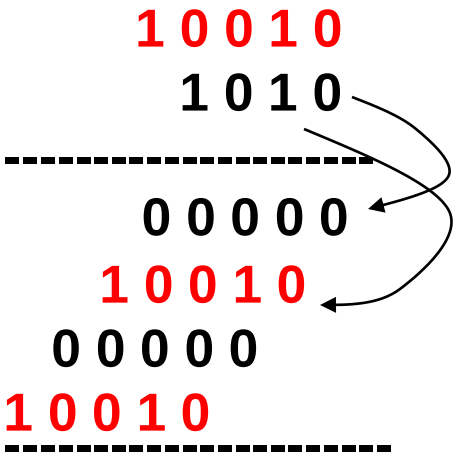
- Problema:
  - Sia dato un intero N, rappresentato in complemento mediante k bit
  - Rappresentare N usando k+q bit (q>0)
- Soluzione:
  - Fare q copie di MSB

- Dimostrazione (banale per N positivo)
  - Sia N<0 (N=1bb…b , dove b è una cifra binaria)
  - Per induzione: Sia N_q la stringa con estensione di q bit
    - q=1: Poiché $-2^{K-1}=-2^K+2^{K-1}$, allora $V(N)=V(N_1)$.
    - q>1: estendere di un bit la stringa ottenuta da N con estensione di q-1 bit → $V(N_q)=V(N_{q-1})$
- Esempio
  - -2 = $(110)_2$ con 3 bit diventa $(111110)_2$ su 6 bit

Moltiplicazione numeri senza segno

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

Esempio (operandi senza segno)



1 0 1 1 0 1 0 0 => $2^7 + 2^5 + 2^4 + 2^2 = 128 + 32 + 16 + 4 =$ **180**

Prodotto e divisione per 2^k

- Il prodotto di N per 2^k si ottiene postando di k posizioni le cifre a sinistra ed inserendo k bit pari a zero
- La divisione di N per 2^k si ottiene postando di k posizioni le cifre a destra ed inserendo k bit pari al valore di MSB (shift aritmetico)

- Esempio : -128/8 = -16 ($8=2^3$)

  1000 0000 → (3 posizioni a destra)

  1111 0000 = $(-16)_{10}$

- Esercizio: verificare tale regola

Prodotto e divisione per 2^k

- Se N è un numero senza segno, allora il prodotto (divisione) per 2^k si ottiene spostando (shift) le cifre a sinistra (destra) di k posizioni ed introducendo 0 nelle posizioni lasciate libere

Esempio: 15 x 4= 60 (4=2^2,shift 2 posizioni)

0000 1111 ←

0011 1100

Esempio: 128 / 2= 64 (2=2^1, shift 1 posizione)

1000 0000 →

0100 0000

Attenzione: nel caso di rappresentazioni con segno questa regola non vale..

Esercizi di riepilogo

- Eseguire le seguenti conversioni
- $(-16)_{10} = (??)_2$ [complemento a 2, minimo numero di cifre]
- $(-16)_{10} = (??)_2$ [complemento a 2, k=10 cifre binarie]
- $(-126)_{10} = (??)_2$ [complemento a 2, minimo numero di cifre]
- $(27)_{10} = (??)_2$
- $(1/3)_{10} = (??)_3$
- $(128)_{10} = (??)_{16} = (??)_8 = (??)_2$
- $(11.111)_2 = (??)_{10}$

Esercizi di riepilogo

Esprimere in base 10 il numero periodico $(0,10)_2$

Eseguire le operazioni

- 16 - 23, in complemento (k=7 bit)
- 16 + 23 in complemento (k=7 bit, k=6 bit)
- -16 - 23 in complemento (k=7 bit e k=6 bit)
- 11101 x 11
- 10101011 / 10