

Sistema binario

base = 2 cifre: 0, 1

$N = 101011.1011_2$

2^5	2^4	2^3	2^2	2^1	2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}
(32)	(16)	(8)	(4)	(2)	(1)	(1/2)	(1/4)	(1/8)	(1/16)
1	0	1	0	1	1	1	0	1	1

$$N = 1 \cdot 2^5 + 0 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 + 1 \cdot 2^{-1} + 0 \cdot 2^{-2} \\ + 1 \cdot 2^{-3} + 1 \cdot 2^{-4} = 43.6875_{10}$$

Avendo c cifre a disposizione si possono rappresentare 2^c numeri interi, per esempio da 0 a $(2^c - 1)$

Per rappresentare un certo numero N:

$$c \geq \log_2(N + 1)$$

Esempi importanti (n=8):

1	1	1	1	1	1	1	1
---	---	---	---	---	---	---	---

$$2^n - 1 = 255$$

0	1	1	1	1	1	1	1
---	---	---	---	---	---	---	---

$$2^{n-1} - 1 = 127$$

1	0	0	0	0	0	0	0
---	---	---	---	---	---	---	---

$$2^{n-1} = 128$$

Aritmetica binaria: Addizione

+	0	1
0	0	1
1	1	(1) 0

Riporto: $1 + 1 = 2_{10} = 10_2$

Esempio:

$$\begin{array}{r} 110 + 6 \\ 10 = 2 \\ \hline 1000 \quad 8 \end{array}$$

Aritmetica binaria: Sottrazione

-	0	1
0	0	(1) 1
1	1	0

Prestito (borrow): $10_2 - 1_2 (= 2_{10} - 1_{10}) = 01_2$

Esempio:

$$\begin{array}{r}
 1110 - 14 \\
 \quad 11 = 3 \\
 \hline
 1011 \quad 11
 \end{array}$$

Aritmetica binaria: Moltiplicazione

*	0	1
0	0	0
1	0	1

Esempio:

$$\begin{array}{r} 111010 * \quad 58 \\ 1011 = \quad 11 \\ \hline 111010 \\ 111010 \\ 111010 - \\ \hline 100111110 \quad 638 \end{array}$$

Aritmetica binaria: Divisione

/	0	1
0	0	0
1	--	1

Esempio:

$$\begin{array}{r}
 \overbrace{111000} : 1000 = 111 \qquad 56 : 8 = 7 \\
 1000 \\
 \hline
 1100 \\
 1000 \\
 \hline
 1000 \\
 1000 \\
 \hline
 0000
 \end{array}$$

Conversione binario \Rightarrow decimale

$$(a_n a_{n-1} \dots a_1 a_0 . a_{-1} a_{-2} \dots) =$$

$$(a_n * 2^n + a_{n-1} * 2^{n-1} + \dots + a_0 + a_{-1} * 2^{-1} + a_{-2} * 2^{-2} \dots)$$

Es: binario \Rightarrow decimale

$$1011_2 \Rightarrow 1*2^3 + 0*2^2 + 1*2^1 + 1*2^0 = \\ 11_{10}$$

Conversione decimale \Rightarrow binario

Dato un numero decimale, è necessario distinguere la parte intera dalla parte frazionaria: $N = I.F$

Regola pratica per la conversione della parte intera

$$I = C_n * 2^n + C_{n-1} * 2^{n-1} + \dots + C_1 * 2^1 + C_0$$

$$I/2 = C_n * 2^{n-1} + C_{n-1} * 2^{n-2} + \dots + C_2 * 2^1 + C_1 \quad \text{con resto } C_0$$

$$I/4 = C_n * 2^{n-2} + C_{n-1} * 2^{n-3} + \dots + C_2 \quad \text{con resto } C_1$$

.....

Per convertire la parte intera I di un numero N in una nuova base b basta dividerla ripetutamente per la nuova base e scrivere ordinatamente i valori dei resti ottenuti, a partire dalla posizione meno significativa

Es: decimale \Rightarrow binario

57

28 1 

14 0

7 0

3 1

1 1


0 1

$$57_{10} = 111001_2$$

ERRORE TIPICO:

considerare la prima cifra ottenuta come la più significativa:

81	
40	1
20	0
10	0
5	0
2	1
1	0
0	1



otterrei **1000101** che vale 69!

NB: conviene sempre “fare la prova”!!!

Metodo “pratico” di conversione da decimale intero a binario

128	64	32	16	8	4	2	1
7	6	5	4	3	2	1	0

ES: convertire in binario 137

$$\begin{aligned}137 &= 128 + 8 + 1 \\ &= 10001001_2\end{aligned}$$

Regola pratica per la conversione della parte frazionaria

Dato un numero binario, sia F la sua parte frazionaria.

$$F = \overset{?}{C_{-1}} * 2^{-1} + \overset{?}{C_{-2}} * 2^{-2} + \dots + \overset{?}{C_{-n}} * 2^{-n}$$

$$F * 2 = C_{-1} + C_{-2} * 2^{-1} + \dots + C_{-n} * 2^{-(n-1)}$$

la parte intera è C_{-1}

$$(F * 2 - C_{-1}) * 2 = C_{-2} + \dots + C_{-n} * 2^{-(n-2)}$$

la parte intera è C_{-2}

.....

Regola pratica per la conversione della parte frazionaria

Per convertire la parte frazionaria F di un numero decimale N in binario basta:

- moltiplicare la parte frazionaria per 2**
- scrivere il valore della parte intera ottenuto nella posizione più significativa**
- ripetere il procedimento sulla parte frazionaria ottenuta**

La conversione ha termine quando si ottiene una parte frazionaria nulla (ciò non è garantito in caso di numeri periodici)

Esempio: convertire 43.6875_{10} in binario


43				0.6875			
21		1	↑	1.375		1	↓
10		1		0.75		0	
5		0		1.5		1	
2		1		1		1	
1		0					
0		1					

$$\Rightarrow 43.6875_{10} = 101011.1011_2$$

ERRORE TIPICO:

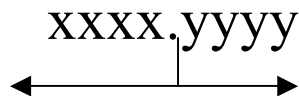
considerare la prima cifra ottenuta come la meno significativa:

0.6875		
1.375		1
0.75		0
1.5		1
1		1



$$0.1101 = 0.5 + 0.25 + \dots > 0.75$$

COME FARE PER RICORDARSELO?



ESERCIZIO

Convertire in binario il numero decimale 0.025

0.025		
0.05		0
0.1		0
0.2		0
0.4		0
0.8		0
1.6		1
1.2		1

$$0.025 = 0.000 \overline{0011}$$

CONVERSIONE BINARI-ESADECIMALI-OTTALI

Oltre al sistema binario e decimale, vi sono altri sistemi “convenienti” per la rappresentazione dell’informazione (anche se in ogni caso nel calcolatore le informazioni sono codificate in binario):

- Sistema ottale ($b = 2^3 = 8$)
- Sistema esadecimale ($b = 2^4 = 16$)

Sistema ottale

$$N_8 = 573.671 =$$

$$5 * 8^2 + 7 * 8^1 + 3 * 8^0 + 6 * 8^{-1} + 7 * 8^{-2} + 1 * 8^{-3}$$

base = 8

cifre = 0,1,2,3,4,5,6,7

Esercizio

Convertire in notazione ottale il numero binario 111010.010

$$\begin{array}{ccc} \boxed{111} & \boxed{010} & \boxed{010} \\ \underline{\quad} & \underline{\quad} & \underline{\quad} \\ 7 & 2 & . 2 \end{array}$$

$$72.2_8$$

ERRORE TIPICO:

Convertire in notazione ottale il numero binario 10111010.11

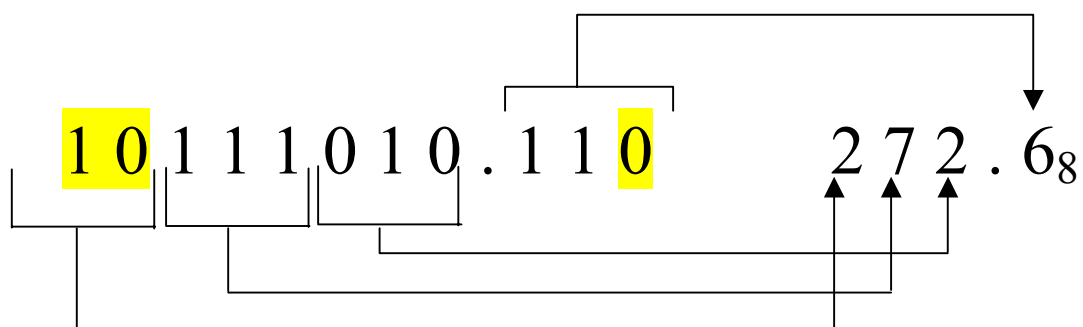
101	110	10	11
5	6	2	3

Invece $562.3_8 = 5*64 + 6*8 + 2 + 3/8 = 370.375$ che sicuramente non può essere rappresentato con una parte intera di soli 8 bit!!!

PARTIRE SEMPRE DAL PUNTO, EVENTUALMENTE COMPLETANDO LE CIFRE CON DEGLI ZERI PER OTTENERE LE TERNE:

xxx xxx . yyy yyy ...

L'esercizio quindi va risolto così:

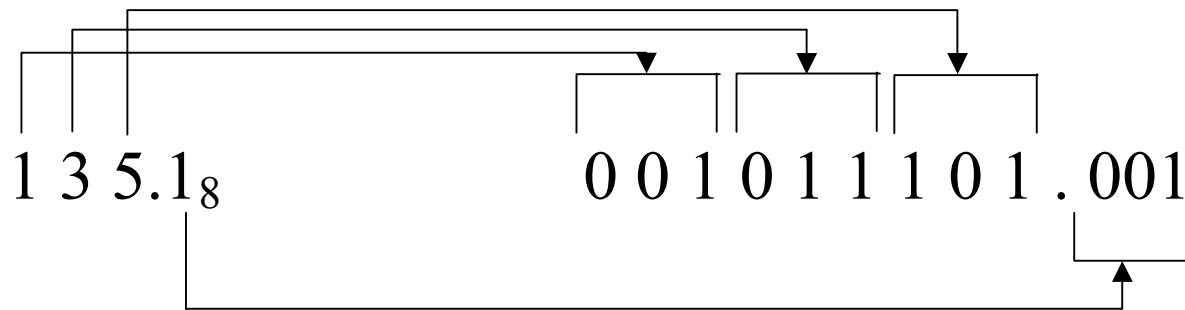


infatti risulta $272.6_8 = 2*64 + 7*8 + 2 + 6/8 = 186.75$

e $10111010.11_2 = 128 + 32 + 16 + 8 + 2 + 0.5 + 0.25 = 186.75$

ESERCIZIO

Convertire in binario il numero in notazione ottale 135.1_8



ERRORE TIPICO: CONVERTIRE IN

001 011 101 . **1**

infatti $0.1_8 = 1/8 = 0.125$ mentre $0.1_2 = 1/2 = 0.5$

Sistema esadecimale

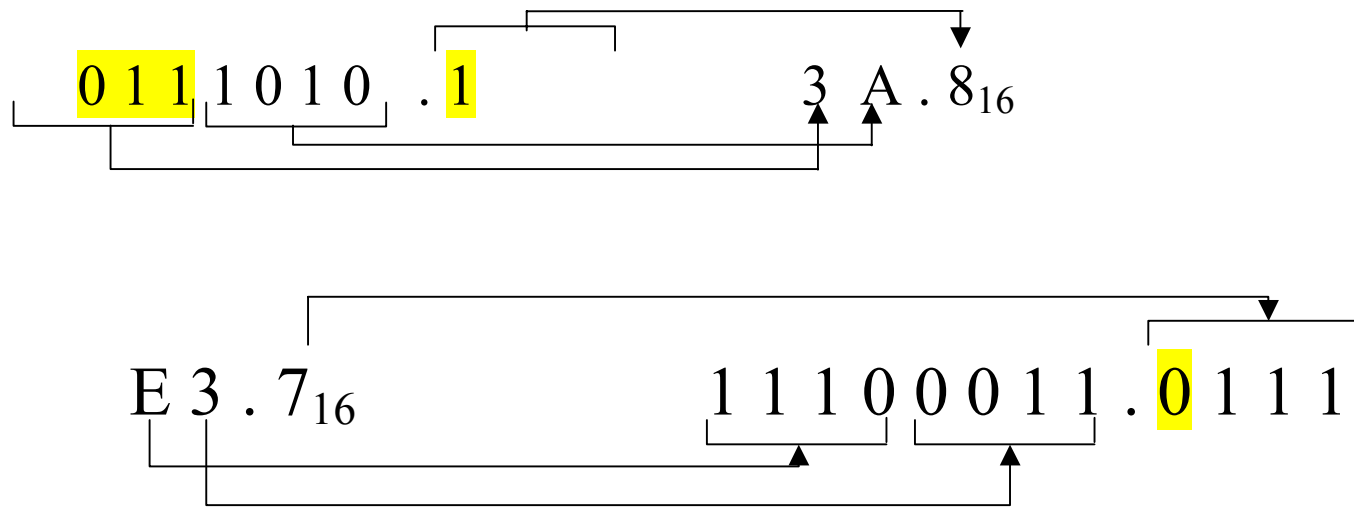
base = 16

cifre = 0, 1, 2, 3, 4, 5, 6, 7, 8, 9,
A, B, C, D, E, F
[10] [11] [12] [13] [14] [15]

$N_{16} = 97A.6E1 =$

$$9 * 16^2 + 7 * 16^1 + 10 * 16^0 + 6 * 16^{-1} + 14 * 16^{-2} + 1 * 16^{-3}$$

Conversione esadecimale-binario



ATTENZIONE!!! PARTIRE SEMPRE DAL PUNTO DECIMALE!!!
XXXX . XXXX XXXX

RAPPRESENTAZIONE DI NUMERI INTERI

I numeri interi (con segno) sono rappresentati un una o più celle di memoria: sia n il numero di bit a disposizione.

Esistono vari tipi di rappresentazione:

- **In valore assoluto e segno**
- **In complemento a 2**
- **In complemento a 1**
- **Polarizzazione o “eccesso”**

Rappresentazione binaria in valore assoluto e segno

Data una parola di lunghezza n:

1 bit viene usato per rappresentare il segno (0 = segno + 1 = segno -)

n-1 bit vengono usati per rappresentare il valore assoluto

1	0	0	0	1	0	1	1
---	---	---	---	---	---	---	---

Esempio: - 11 con 8 bit

Esistono due codifiche per il valore 0

0	0	0	0	0	0	0	0
---	---	---	---	---	---	---	---

0^+

1	0	0	0	0	0	0	0
---	---	---	---	---	---	---	---

0^-

I valori rappresentabili vanno:

da $-(2^{n-1} - 1)$ a $+(2^{n-1} - 1)$

segno	*	*	*	*	*	*	*
-------	---	---	---	---	---	---	---

n-1 bit: da 0 a $2^{n-1} - 1$

Inconveniente nell'uso della rappresentazione in valore assoluto e segno: difficoltà nelle operazioni di addizione e sottrazione (due operazioni distinte, decisione presa analizzando i segni degli operandi).

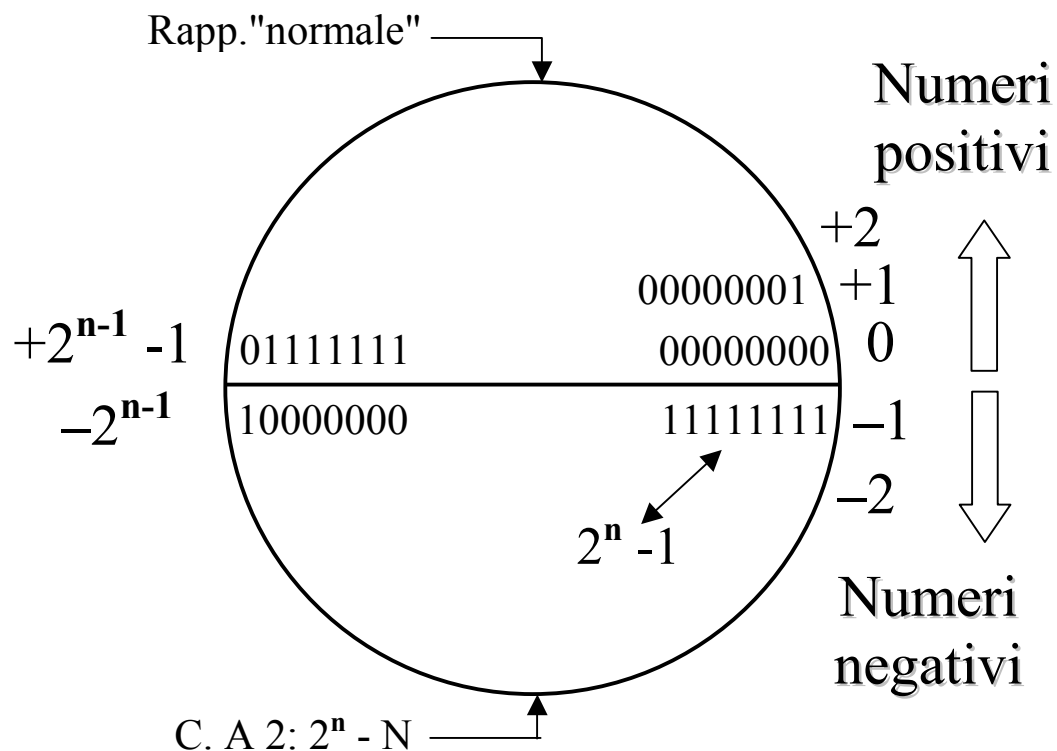
Sottrazione a-b

Segno a	Segno b		segno(a-b)	a-b
+	+	$ a > b $	+	$a - b$
+	+	$ b > a $	-	$b - a$
-	+		-	$ a + b$
+	-		+	$a + b $
-	-	$ a > b $	-	$ a - b $
-	-	$ b > a $	+	$ b - a $

Si vuole una rappresentazione per la quale esista un unico semplice metodo per l'addizione e la sottrazione...

Rappresentazione in Complemento a 2

Idea: con n cifre $\Rightarrow 2^n$ numeri, p.es. da 0 a $2^n - 1$



Data una parola di lunghezza n:

**i numeri positivi sono rappresentati nello stesso modo della rappresentazione in segno e valore assoluto;
il bit più significativo è pari a 0**

**i numeri negativi si ottengono come complemento a 2 del numero positivo corrispondente;
il bit più significativo è pari a 1**

I valori rappresentabili vanno:

da -2^{n-1} a $+2^{n-1} - 1$

in pratica da $-(2^{n-1} - 1)$ a $+2^{n-1} - 1$

Complemento a 2

Dato un numero N rappresentato in base 2 da n cifre il suo complemento a 2 è dato da:

$$C_2 = 2^n - N$$

Esempio:

$$N_2 = 10010100$$

$$C_2 = \begin{array}{r} 100000000 - \\ \quad 10010100 \\ \hline \quad 01101100 \end{array}$$

Regola pratica: partendo dal bit meno significativo si lasciano immutati tutti i bit fino al primo 1 compreso, poi si invertono gli altri

ESEMPI

0	0	0	0	0	0	0	0
---	---	---	---	---	---	---	---

$$N=0$$

NB: c'è un'unica rappresentazione per lo 0

0	1	1	1	1	1	1	1
---	---	---	---	---	---	---	---

$$N=2^{n-1} - 1$$

1	1	1	1	1	1	1	1
---	---	---	---	---	---	---	---

$$N= - 1$$

1	0	0	0	0	0	0	0
---	---	---	---	---	---	---	---

$$N= - 2^{n-1} \text{ [non usato]}$$

1	0	0	0	0	0	0	1
---	---	---	---	---	---	---	---

$$N= - (2^{n-1} - 1)$$

ESERCIZIO

Rappresentare in complemento a due con 8 bit il numero decimale -67 .

quindi

$$67_{10} = 01000011_2$$
$$-67 = 10111101$$

ERRORE TIPICO: dimenticarsi che la rappresentazione in C.a 2 è relativa ad un numero di bit fissati!!!

Es. Rappresentare in C. a 2 con 8 bit il numero decimale -3 .

$$3_{10} = 11_2$$

**quindi $-3 = 01$
ma questo risulta un numero positivo!!!**

Svolgimento corretto:

$$3 = 00000011 \quad (8 \text{ bit})$$

$$-3 = 11111101$$

ALTRO ERRORE TIPICO: complementare sempre e comunque, anche i numeri positivi!

ES: Rappresentare il complemento a due con 6 bit del numero decimale 15.

$$15 = 001111 \quad \text{anche in compl. a 2!!!}$$

AVVERTENZA:

“RAPPRESENTAZIONE IN COMPLEMENTO A DUE” VS.
“COMPLEMENTO A DUE”.

Appello del 9 dicembre 2002:

Rappresentare i numeri 96 e 69 (in base 10) in notazione binaria in complemento a due con 8 bit. Eseguire la somma algebrica dei numeri così ottenuti e commentare il risultato. [4].

Appello del 7 gennaio 2003:

Progettare una Macchina di Turing che ricevendo in ingresso un numero binario di lunghezza arbitraria lo trasformi nel suo complemento a due. [5]

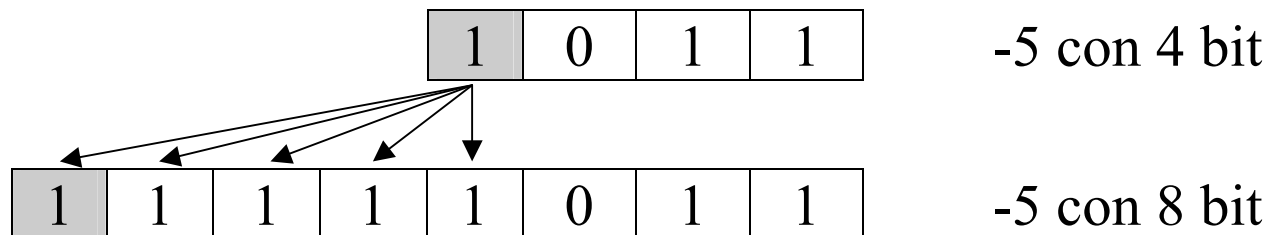
Simmetria dell'operazione di complemento a 2

Dato $p > 0$ rappresentato in C. A 2, per ottenere $-p$ ne faccio il C. A 2

Dato $p < 0$ rappresentato in C. A 2, per ottenere $-p > 0$ ne faccio il C. A 2

Es. $+3: 0011$ $-3: 1101$

Estensione del segno da n a n+d bit

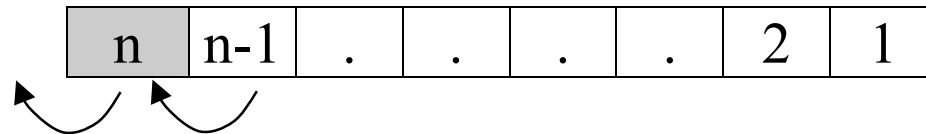


Aritmetica in complemento a 2

Addizione:

L'addizione di due numeri rappresentati in complemento a 2 dà il risultato corretto trascurando il riporto (a patto che il risultato sia entro il range dei numeri rappresentabili).

Si ha overflow se i riporti generati nelle posizioni n e $n-1$ sono diversi.



Sottrazione: semplice somma del complemento a 2.

Esempi (n=6: $-32 \leq x \leq 31$)

$$26 - 13 = 26 + (-13) = +13$$

$$\begin{array}{r}
 011010 + \quad 26 \\
 110011 = \quad -13 \quad [13: 001101] \\
 \hline
 \cancel{1}001101 \quad +13 \\
 \swarrow \searrow \downarrow
 \end{array}$$

$$-25 - 13 = -25 + (-13) = -38 \text{ [overflow]}$$

$$\begin{array}{r}
 100111 + \quad -25 \quad [25: 011001] \\
 110011 = \quad -13 \quad [13: 001101] \\
 \hline
 \cancel{1}011010 \quad \text{"+26"} \\
 \swarrow \searrow \downarrow
 \end{array}$$

Esercizio (Appello del 23 set 2003)

Rappresentare i numeri -51 e -98 (in base 10) in notazione binaria in complemento a due con 8 bit. Eseguire la somma algebrica dei numeri così ottenuti e commentare il risultato. [3]

Soluzione

Conversione nella notazione binaria dei numeri (valori assoluti)

$$51 = 00110011_2 \\ (32 + 16 + 2 + 1)$$

$$98 = 01100010_2 \\ (64 + 32 + 2)$$

Rappresentazione in complemento a due:

$$-51 = 11001101 \\ -98 = 10011110$$

Somma algebrica:

$$\begin{array}{r} 11001101 + \\ 10011110 = \\ 101101011 \end{array}$$

Commento:

Poiché i riporti generati nei bit di posizione 7 e 8 sono diversi (0 e 1 rispettivamente) ho un caso di overflow [cosa evidente anche dal fatto che dalla somma di due numeri negativi ottengo un numero positivo]. In effetti con 8 bit posso esprimere in complemento a due i numeri da -128 a $+127$ [nella pratica da -127 a $+127$], mentre si ha $-51 - 98 = -149$.

NB: è possibile vedere che a 9 bit la somma sarebbe giusta:

$$\begin{array}{r} 111001101 + \\ 110011110 = \\ 1\ 101101011 \\ \swarrow \searrow \end{array}$$

ovvero risulta $-010010101_2 = -(128 + 16 + 4 + 1) = -149$,
pari infatti a $-51 - 98$.

Esercizio (Appello del 23 set 2003)

Rappresentare i numeri -54 e -44 (in base 10) in notazione binaria in complemento a due con 8 bit. Eseguire la somma algebrica dei numeri così ottenuti e commentare il risultato. [3]

Soluzione

- Conversione nella notazione binaria dei numeri (valori assoluti)

$$54 = 00110110_2$$

(32+16+4+2)

$$44 = 00101100_2$$

(32+8+4)

- Rappresentazione in complemento a due dei numeri (con 8 bit)

$$-54 = 11001010$$

$$-44 = 11010100$$

- Somma algebrica:

$$\begin{array}{r}
 11001010 + \\
 \underline{11010100} = \\
 1\ 10011110 \\
 \swarrow \searrow
 \end{array}$$

e quindi il risultato è 10011110.

Commento:

Non c'è overflow perché i riporti generati nelle posizioni 7 e 8 sono uguali (1 e 1). In effetti risulta un numero negativo pari a $-01100010_2 = -98_{10}$, che è proprio $-54 - 44$

ERRORE COMMESSO DA ALCUNI

Dimenticarsi del numero di bit (8):

$$\begin{array}{ll} 54 = 110110_2 & -54 = 001010 \\ 44 = 101100_2 & -44 = 010100 \end{array}$$

e sommando:

$$\begin{array}{r} 001010+ \\ 010100= \\ 011110 \end{array}$$

risultato che non ha alcun senso!!!

(NB: in questo caso l'esercizio è valutato 0 punti!)

Esercizio (Appello del 9 dic 2002)

Rappresentare i numeri 96 e 69 (in base 10) in notazione binaria in complemento a due con 8 bit. Eseguire la somma algebrica dei numeri così ottenuti e commentare il risultato. [4]

Soluzione

Rappresentazione in complemento a due con 8 bit:

$$96 = 01100000 \qquad 69 = 01000101$$

Somma algebrica:

$$\begin{array}{r} 01100000+ \\ 01000101= \\ \hline 10100101 \\ \blacktriangledown \end{array}$$

Commento:

Ho overflow perchè i riporti sono diversi (ho soltanto il riporto nel bit 7), cosa deducibile anche dal fatto che il risultato ha il “bit di segno” a 1 (rappresenta un numero negativo ottenuto dalla somma di due positivi!).

ERRORE TIPICO: complementare i due numeri che in realtà sono positivi (NB: in questo caso l'esercizio è valutato 0 punti).

Esercizi proposti (soluzioni: pagina WEB)

- 1) Convertire il seguente numero, dato in complemento a due, come numero relativo in base 8: 1111000110
- 2) Rappresentare il numero -748_{10} in complemento a due con il minimo numero di bit possibili:

Per esercitarsi su somme in complemento a due + conversioni:

<http://www.brunel.ac.uk/~castjg/hndcfund/material/java/tca/tca.html> anche in
<http://www.cs.princeton.edu/courses/archive/spr01/cs126/demo/numbers/twos-complement.html>

Calcolatrice in complemento a due a 8 bit. Consente di convertire un numero decimale da/a complemento a due (8 bit), eseguire la somma e segnalare l'eventuale condizione di overflow. Richiede java (jre).

Rappresentazione in Complemento a 1

Dato un numero N rappresentato in base 2 da n cifre il suo complemento a 1 è dato da:

$$C_1 = (2^n - 1) - N$$

Esempio:

$$N_2 = 10010100$$

$$C_1 = \begin{array}{r} 11111111 - \\ 10010100 \\ \hline 01101011 \end{array}$$

Regola pratica: si inverte il valore di tutti i bit

Data una parola di lunghezza n:

**i numeri positivi sono rappresentati nello stesso modo della rappresentazione in segno e valore assoluto;
il bit più significativo è pari a 0**

**i numeri negativi si ottengono come complemento a 1 del numero positivo corrispondente;
il bit più significativo è pari a 1**

I valori rappresentabili vanno:

da $-(2^{n-1} - 1)$ a $+2^{n-1} - 1$

ESEMPI:

Esistono due rappresentazioni per lo zero:

0	0	0	0	0	0	0	0
---	---	---	---	---	---	---	---

 0^+

1	1	1	1	1	1	1	1
---	---	---	---	---	---	---	---

 0^- [rapp: $2^n - 1 - 0$]

1	1	1	1	1	1	1	0
---	---	---	---	---	---	---	---

 -1

1	0	0	0	0	0	0	0
---	---	---	---	---	---	---	---

 $-(2^{n-1} - 1)$

NB: anche nel caso del complemento a 1 valgono simmetria del complemento a 1 + regola sull'estensione del segno

Aritmetica in complemento a 1

Addizione:

L'addizione di due numeri rappresentati in complemento a 1 dà il risultato corretto sommando al risultato ottenuto il riporto (a patto che il risultato sia entro il range dei numeri rappresentabili)

Si ha overflow se i riporti generati nelle posizioni n e $n-1$ [tenendo conto anche di quelli generati nella somma del riporto!] sono diversi

Sottrazione:

Per sottrarre un numero basta sommare il suo complemento a 1

Esempi (n=6: $-31 \leq x \leq 31$)

$$26 - 13 = 26 + (-13) = +13$$

	0 1 1 0 1 0 +	26	
	1 1 0 0 1 0 =	-13	[13: 0 0 1 1 0 1]
(1)	0 0 1 1 0 0		
	0 0 1 1 0 1	+13	

$$-25 - 6 = -25 + (-6) = -31$$

	1 0 0 1 1 0 +	-25	[25: 0 1 1 0 0 1]
	1 1 1 0 0 1 =	-6	[6: 0 0 0 1 1 0]
(1)	0 1 1 1 1 1		
	1 0 0 0 0 0	-31	

Altra regola pratica:

il complemento a 2 si può ottenere sommando 1 al complemento a 1

infatti $C. A 2 = 2^n - N$, $C. A 1 = (2^n - 1) - N$

Esempio (n=5)

7: 00111

-7 in C. A 1: 11000

-7 in C. A 2: 11001

Esercizio

Rappresentare i numeri -27 e -9 in complemento a uno con 8 bit e con 6 bit. Eseguire la somma in entrambe le rappresentazioni e commentare i rispettivi risultati.

Soluzione

$$27 = 11011$$

$$9 = 1001$$

- a 8 bit:

$$27 = 00011011$$

$$9 = 00001001$$

$$-27 = 11100100$$

$$-9 = 11110110$$

somma:

$$\begin{array}{r}
 11100100+ \quad -27 \\
 11110110= \quad -9 \\
 \hline
 1 \quad 11011010+ \\
 \quad \quad \quad \quad \quad 1= \\
 \hline
 11011011
 \end{array}$$

Non c'è overflow perché i due riporti sono uguali.

Prova: il risultato è $-(00100100) = -36 = -27 -9$

- a 6 bit:

$$\begin{array}{ll}
 27 = 011011 & 9 = 001001 \\
 -27 = 100100 & -9 = 110110
 \end{array}$$

Somma:

$$\begin{array}{r} 100100 + \\ 110110 = \\ \hline 1 \quad 011010 + \\ \quad \quad \quad 1 = \\ \hline 011011 \end{array}$$

Ho overflow perché i riporti sono diversi (oppure: perché sommando due negativi ottengo un positivo)

NB: la differenza tra i due casi si spiega perché con 8 bit sono rappresentabili numeri da $-(2^7-1)$ a $+(2^7-1) = -127$ a $+127$,
con 6 bit da $-(2^5-1)$ a $+(2^5-1) = -31$ a $+31$.