CODIFICA NUMERI INTERI

NOTE ALLE ESERCITAZIONI 14.12.2018

A.A 2018-2019

Conversione binario $\Rightarrow$ decimale

La conversione è immediata, secondo la definizione stessa di numero binario:

$(a_n\ a_{n-1} \ldots a_1\ a_0) = (a_n * 2^n + a_{n-1} * 2^{n-1} + \ldots + a_0)$

Es: binario $\Rightarrow$ decimale

$$101011 \Rightarrow 1\cdot 2^5 + 0\cdot 2^4 + 1\cdot 2^3 + 0\cdot 2^2 + 1\cdot 2^1 + 1\cdot 2^0 = 43$$

Conversione decimale $\Rightarrow$ binario

Regola pratica per la conversione

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

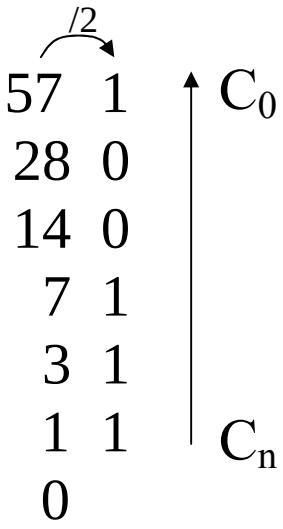
$I/2 = C_n*2^{n-1} + C_{n-1}*2^{n-2} + \ldots\ldots + C_2*2^1 + C_1$ con resto C_0

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

.....

Per convertire la parte intera I di un numero N decimale in notazione binaria basta dividerla ripetutamente per 2 e scrivere ordinatamente i valori dei resti ottenuti, a partire dalla posizione meno significativa

Es: decimale ⇒ binario



$57_{10} = 111001_2$

ERRORE COMUNE:

considerare la prima cifra ottenuta come la più significativa:

Metodo “pratico” di conversione da decimale a binario

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

ES: convertire in binario 137

$137 = 128 + 8 + 1 \qquad [= 2^7 + 2^3 + 2^0]$

$= 10001001_2$

NB: almeno per valori non troppo elevati, è utile usare questo metodo

Notazioni ottale ed esadecimale

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

base = 8 cifre = 0,1,2,3,4,5,6,7

ES: $573_8 = 5 * 8_2 + 7 * 8_1 + 3 * 8_0$

Esercizio Convertire in notazione ottale il numero binario 111010

111 010 = 72

7 2

ERRORE COMUNE:

Convertire in notazione ottale il numero binario 10111010

10111010

5 6 2

=> ERRORE: per raggruppare le cifre si parte da quella meno significativa

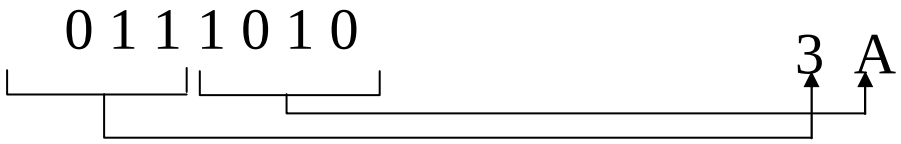
Sistema esadecimale

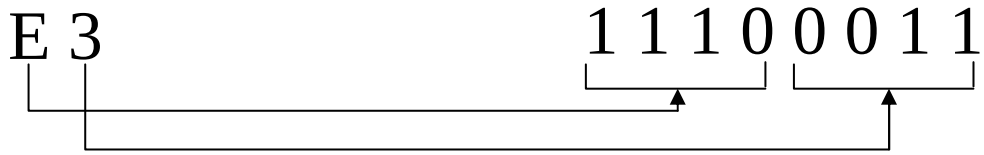
base = 16

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

ES: $97A_{16} = 9 * 16^2 + 7 * 16^1 + 10 * 16^0$

Conversione binario-esadecimale e viceversa





Codifica in complemento a 2

Dato un numero N rappresentato in base 2 da n cifre il suo complemento a 2 è dato da: $\mathbf{C_2 = 2^n - N}$

Esempio:

$\mathbf{N_2 = 10010100}$

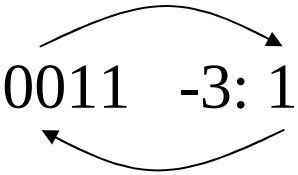
$$\begin{array}{rl} \mathbf{C_2 =} & \mathbf{100000000 -} \\ & \mathbf{10010100} \\ \hline & \mathbf{01101100} \end{array}$$

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

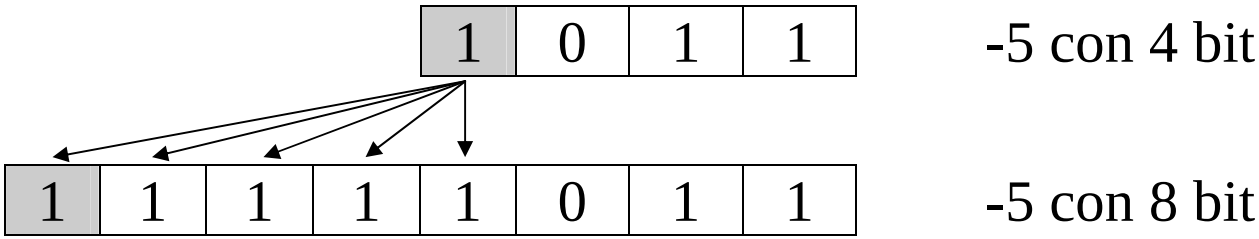
Simmetria dell'operazione di complemento a 2

Dato p > 0 rappresentato in C_2, per ottenere -p ne faccio il C_2
Dato p < 0 rappresentato in C_2, per ottenere -p ne faccio il C_2

Es. +3: 0011 -3: 1101



Estensione del segno da n a n+d bit



-5 con 4 bit

-5 con 8 bit

ERRORE COMUNE: dimenticarsi che la rappresentazione in C_2 è relativa ad un numero di bit fissati.

Es. Rappresentare in C_2 con 8 bit il numero decimale –3.

$3_{10} = 11_2$

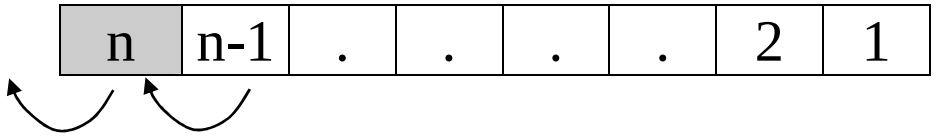
quindi –3 = 01
ma questo risulta un numero positivo
=> ERRORE

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.

Codifica in complemento a 1

Dato un numero N rappresentato in base 2 con n cifre il suo complemento a 1 è dato da: $\mathbf{C_1 = (2^n - 1) - N}$

Esempio:

$$\mathbf{N_2 = 10010100}$$

$$\begin{array}{rl} \mathbf{C_1 =} & \mathbf{11111111} \\ & \mathbf{-\ 10010100} \\ \hline & \mathbf{01101011} \end{array}$$

Regola pratica: si complementano (invertono) i valori di tutti i bit

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 ≤ x ≤ 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
```

Regola pratica:

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

infatti $C_2 = 2^n - N$ **e** $C_1 = (2^n - 1) - N$

Esempio (n=5)

7: 00111

-7 in C_1: 11000

-7 in C_2: 11001