Sistemi numerici

Il sistema che utilizza il pc per elaborare le istruzioni è il sistema binario, esso è composto da due sole cifre 0 e 1.

Utilizzando il sistema binario e avendo n cifre a disposizione possiamo rappresentare 2^n numeri, questo per via del calcolo combinatorio:

$$2 \cdot 2_1 \ldots . 2_n$$

- **Numeri decimali**

$$5374_{10} = 5 \cdot 10^3 + 3 \cdot 10^2 + 7 \cdot 10^1 + 4 \cdot 10^0$$

- **Numeri binari**

$$1101_2 = 1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = 13_{10}$$

Potenze di 2

- $2^0 = 1$
- $2^1 = 2$
- $2^2 = 4$
- $2^3 = 8$
- $2^4 = 16$
- $2^5 = 32$
- $2^6 = 64$
- $2^7 = 128$
- $2^8 = 256$
- $2^9 = 512$
- $2^{10} = 1024$
- $2^{11} = 2048$
- $2^{12} = 4096$
- $2^{13} = 8192$
- $2^{14} = 16384$
- $2^{15} = 32768$

Conversazione numerica

- **Conversione da binario a decimale:**

un esempio di conversione in decimale può essere 10011_2 che in decimale diventa:

$$\overset{4}{1}\overset{3}{0}\overset{2}{0}\overset{1}{1}\overset{0}{1}_2 = 2^4 \cdot 1 + 2^3 \cdot 0 + 2^2 \cdot 0 + 2^1 \cdot 1 + 2^0 \cdot 1 =$$

$$= 16 \cdot 1 + 8 \cdot 0 + 4 \cdot 0 + 2 \cdot 1 + 1 \cdot 1 = 19_{10}$$

- **Conversione da decimale a binario:**

un esempio di conversione in decimale può essere 47_{10} che in decimale diventa:

47	1
23	1
11	1
5	1
2	0
1	1
0	

SI LEGGE DAL BASSO VERSO L'ALTO

$= 101111_2$

Per la conversione da decimale a binario ci sono 2 metodi:

- **Metodo 1:**
  - trova la più grande potenza di 2 più adatta;
  - sottrai;
  - se non esce 0 ripeti.

Esempio:

53_{10}

$53-32=21$	$32 \cdot 1$
$21-16=5$	$16 \cdot 1$
$5-4=1$	$4 \cdot 1$
	$1 \cdot 1$

$\longrightarrow 110101_2$

- **Metodo 2:**
  - dividi per 2;
  - metti il resto a sinistra della rappresentazione binaria.

Esempio:

53	1
26	0
13	1
6	0
3	1
1	1
0	

110101_2

Altri esempi dei due metodi possono essere:

$75_{10} = 64+8+2+1 = 1001011_2$

oppure

75	1
37	1
18	0
9	1
4	0
2	1
1	1
0	

$= 1001011_2$

Da x a y

Quando invece abbiamo una base x e vogliamo convertirla ad una base y prima convertiamo da x a decimale e poi da decimale a y.

Esempio:

Abbiamo 25_{20} e vogliamo convertirlo in base 4:

1) Convertiamo prima in base 10 quindi:

$$\overset{1}{2}\overset{0}{5}_{20} = 2 \cdot 20^1 + 5 \cdot 20^0 = 40 + 5 = 45$$

2) convertiamo 45 in base 4 e per fare questo basterà dividere 45 finchè non esce 0 e ad ogni divisione metteremo da parte i resti:

RESTI DIVISIONI

45	1
11	3
2	2
0	

$= 231_4$

Valori binari e intervallo

- **Numero decimale a N cifre**
  - Quanti valori? 10^N
  - Range? $[0, 10^N - 1]$

    Esempio con un numero decimale a 3 cifre:
    - $10^3 = 1000$ possibili valori;
    - Range: $[0, 999]$
- Numero binario a N cifre
  - Quanti valori? 2^N
  - Range $[0, 2^N - 1]$

    Esempio con un numero binario a 3 cifre:
    - $2^3 = 8$ possibili valori
    - Intervallo: $[0, 7] = [000_2, 111_2]$

Numeri esadecimali

Hex Digit	Decimal Equivalent	Binary Equivalent
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
A	10	1010
B	11	1011
C	12	1100
D	13	1101
E	14	1110
F	15	1111

Le caratteristiche principali dei numeri esadecimali sono 2:

- ha come base 16
- abbreviazione binaria

Conversione da esadecimale a binario

Proviamo per esempio a convertire 4AF (anche scritto 0x4AF) in binario:

$$\underbrace{0\,1\,0\,0}_{4}\,\underbrace{1\,0\,1\,0}_{A}\,\underbrace{1\,1\,1\,1}_{F}\,{}_2$$

Conversione da esadecimale a decimale

Per la conversione da esadecimale a decimale invece proviamo a convertire sempre $4AF_{16}$ in decimale:

$$16^2 \cdot 4 + 16^1 \cdot 10 + 16^0 \cdot 15 = 1199_{10}$$

Bits, Bytes, Nibble...

Un bit è una singola unità di un numero binario ad esempio 0 o 1.

Invece un byte sono 8 bit ed un nibble 4 bit, quindi 2 nibble formano un byte.

- Un esempio di bits può essere:

1 0 0 1 0 1 1 0

bit più significativo — bit meno significativo

- Esempio di Bytes & Nibble invece può essere:

byte

$\overbrace{1\ 0\ 0\ 1\ \underbrace{0\ 1\ 1\ 0}_{\text{Nibble}}}^{}$

- Infine un esempio di Bytes può essere.

CE B F 9 A D7

byte più significativo — byte meno significativo

Grandi potenze di 2

Esempi di grandi potenze di 2 sono:

- 2^{10} = 1 Kilo ≅ 1000 (1024)
- 2^{20} = 1 mega ≅ 1 milione (1048576)
- 2^{30} = 1 giga ≅ 1 miliardo (1073741824)

Addizioni

Le addizioni con numeri decimali e binari si fa in due modi diversi:

- decimale

RIPORTI

```
  1 1
 3 7 3 4
+5 1 6 8
--------
 8 9 0 2
```

• binario

RIPORTI

$$\begin{array}{rcccc} & ^{1}1 & 0 & ^{1}1 & 1 \\ + & 0 & 0 & 1 & 1 \\ \hline & 1 & 1 & 1 & 0 \end{array}$$

In binario le cose che cambiano nell'addizione sono:

- 1+1 fa 0 con riporto di 1
- 0+0 fa 0
- 1+0 fa 1
- 0+1 fa 1

Esempi di operazioni esadecimali

Facciamo 2 esempi di operazione con numeri esadecimali:

• Addizione:

$$\begin{array}{rcccc} & ^{1}3 & ^{0}A & ^{1}0 & 9 \\ + & 1 & B & 1 & 7 \\ \hline & 5 & 5 & 2 & 0 \end{array}$$

9
A
B
C
D
E
F
0 con riporto di 1

• Sottrazioni:

$$\begin{array}{rcccc} & 3 & D & ^{1}0 & 9 \\ - & 1 & B & 2 & 7 \\ \hline & 2 & 1 & E & 2 \end{array}$$

Overflow

I sistemi digitali operano su un numero fisso di bit ed overflow si riferisce a quando il risultato è troppo grande per adattarsi al numero di bit disponibile.
Ciò avviene se avendo una nibble a disposizione proviamo a computare per esempio 12+11 che fa 23 e sarà

maggiore rispetto a 2^4-1 ovvero 15

Numeri Sign/Magnitude

Hanno 1 bit di segno e N-1 magnitude bits.
Il bit più a sinistra va ad identificare il segno del bit

- numero positivo il bit varrà 0;
- numero negativo il bit varrà 1.

$$A=(-1)^{a_{N-1}} \sum_{i=0}^{N-2} a_i 2^i$$

Per esempio la rappresentazione in sign/mag a 4 bit di ± 6 è

$+6 = 0110$

$-6 = 1110$

Il range di un numero N bit sign/mag è:

$$[-(2^{N-1}-1), 2^{N-1}-1]$$

Numeri Sign/Magnitude

Esistono vari problemi con i numeri sign/mag per esempio le addizioni non funzionano, tipo $-6+6$

$$\begin{array}{r} 1110 \\ +\ 0110 \\ \hline 10100 \end{array}$$ (SBAGLIATO!)

Un altro problema è che esistono 2 rappresentazioni di ±

- $-0 = 1000$
- $+0 = 0000$

Numeri in complemento a due

Nel complemento a due non abbiamo lo stesso problema che abbiamo nei numeri sign/magnitude quindi troveremo:

- addizioni funzionanti
- una sola rappresentazione dello 0

Nel complemento a due abbiamo

- il bit più significativo che ha come valore -2^{N-1}

$$A = a_{N-1}(-2^{N-1}) + \sum_{i=0}^{N-2} a_i 2^i$$

- numero a 4 bit più positivo è: 0 1 1 1
- numero a 4 bit più negativo è: 1 0 0 0

- Il bit più significativo indica ancora il segno quindi 1=negativo e 0=positivo
- Il range di un numero in complemento a due di N bit è:

$$[(2-1), 2N-1-1]$$