Sia $N = 1574$ (valore)

$= 1 \cdot 1000 + 5 \cdot 100 + 7 \cdot 10 + 4 \cdot 1$

migliaia, centinaia, decina, unità

$= 1 \cdot 10^3 + 5 \cdot 10^2 + 7 \cdot 10^1 + 4 \cdot 10^0$

PESI

Sistema POSIZIONALE PESATO

cifre $\{\underbrace{0, 1, \dots 9}_{D}\} =$

Ogni cifra assume significato a seconda della POSIZIONE che occupa (e.g. 5 unità o centinaia)

Indicando la cifra generica con d_i

$N = d_{k-1} \cdot 10^{k-1} + d_{k-2} \cdot 10^{k-2} + \dots + d_1 \cdot 10^1 + d_0 \cdot 10^0$

$(d_{k-1}\, d_{k-2} \dots d_1 d_0)$

(Nell'esempio $N = d_3 d_2 d_1 d_0 = 1574$)

Sia $B = \{0, 1\}$ (cifre binarie)

$N = b_{n-1} b_{n-2} \ldots b_1 b_0$ $\qquad b_{n-1} \cdot 2^{n-1} + b_{n-2} \cdot 2^{n-2} + \ldots + b_1 \cdot 2^1 + b_0 \cdot 2^0$

PESI

Domande:

1) Quanti valori posso rappresentare con n bit. Dagli esempi

2 bit → 4, 3 bit → 8 4 bit → 16

In generale n bit → 2^n

Perché? Idea:

0 00	1 00
0 01	1 01
0 10	1 10
0 11	1 11

oppure

Con ogni bit che aggiungo, raddoppio i valori rappresentabili

Esempi di conto in binario

00 → 0
01 → 1
10 → 2
11 → 3

000 → 0
001 → 1
010 → 2
011 → 3
100 → 4
101 → 5
110 → 6
111 → 7

$1 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0$

$4 + 2 + 1 = 7$

2) Qual è il massimo rappresentabile?

2 bit ⇒ 3 , 3 bit ⇒ 7 , 4 bit ⇒ 15

In generale n bit ⇒ $2^n - 1$

Perché? Idea

Se con $n-1$ bit il max rapp. è $2^{n-1}-1$, aggiungendo l'n-esimo bit, affinché incrementi il valore deve essere 1. Ma ha peso 2^{n-1}. Pertanto

$$V = \underbrace{2^{n-1}}_{1} + \underbrace{2^{n-1} - 1}_{\text{MAX su } n-1 \text{ bit}} = 2 \cdot 2^{n-1} - 1 = 2^n - 1$$

3) Quanti bit occorrono per rappresentare N espresso in notazione decimale?

Idea: con n bit il MAX rappresentabile è $2^n - 1$

Pertanto n deve essere tale che

$$2^n - 1 \geq N$$

$$\iff \quad 2^n \geq N+1 \quad \iff \quad n \geq \log_2(N+1)$$

e poiché n deve essere intero

$$n = \lceil \log_2(N+1) \rceil$$

$N = 8$

$n = \lceil \log_2(8+1) \rceil$

$= \lceil \log_2 9 \rceil \to 3$

$= 4$

e.g. $N = 7$

$n = \lceil \log_2(7+1) \rceil$

$= \lceil \log_2 8 \rceil$

$= 3$

Osservazione: più è piccolo l'insieme delle cifre tanto più sono lunghe le rappresentazioni

$N_{10} = 7$ (in decimale una cifra) $\quad N_2 = 111$ (in binario 3 cifre)

Come passiamo da una rappresentazione all'altra in modo efficiente?

Non efficiente binario decimale

$$\begin{array}{cccccccc} 7 & 6 & 5 & 4 & 3 & 2 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \end{array} \longrightarrow 2^7 + 2^5 + 2^4 + 2^2 + 2^1 =$$

$$128 + 32 + 16 + 4 + 2 \ldots$$

$$N_{10} = d_{k-1} d_{k-2} \dots d_1 d_0 = b_{n-1} b_{n-2} \dots b_1 b_0 = N_2$$

rapp. decimale

rapp. binaria

$$N_{10} = b_{n-1} \cdot 2^{n-1} + b_{n-2} \cdot 2^{n-2} + \dots + b_1 \cdot 2^1 + b_0 \cdot 2^0$$

$$= b_0 \cdot 2^0 + b_1 \cdot 2^1 + \dots \quad b_{n-2} \cdot 2^{n-2} + b_{n-1} \cdot 2^{n-1}$$

$$= b_0 + 2 \left(b_1 + b_2 \cdot 2^1 + \dots + b_{n-2} \cdot 2^{n-3} + b_{n-1} \cdot 2^{n-2} \right)$$

Osservazione $N_{10} = b_0 + 2 X$ se divido N_{10} per 2 il resto della divisione è b_0!

$$= b_0 + 2 \left(b_1 + 2 \left(b_2 + b_3 \cdot 2^1 + \dots + b_{n-2} \cdot 2^{n-4} + b_{n-1} \cdot 2^{n-3} \right) \right)$$

Osservazione $X = b_1 + 2 Y$ se divido X (quoziente della divisione precedente) per 2, il resto della divisione è b_1!

Posso iterare fino a b_{n-2} ...

$$N_{10} = b_0 + 2\left(b_1 + 2\left(b_2 + \ldots \quad (b_{n-2} + 2\, b_{n-1})\right)\ldots\right)$$

Da questa relazione posso derivare due algoritmi di conversione DEC $\rightarrow$ BIN e BIN $\rightarrow$ DEC efficienti

$b_0 = \overset{S_0}{\overline{N_{10}}} \% 2$ $\quad$ (sia $N_{10} = S_0$ e sia $\overset{S_0}{N_{10}} = \underset{\text{resto}}{b_0} + 2\,\underset{\text{quoziente}}{S_1}$)

$b_1 = S_1 \% 2$ $\quad$ $(S_1 = b_1 + 2 S_2)$

$b_2 = S_2 \% 2$

$\vdots$

$b_{n-1} = S_{n-1} \% 2$ $\quad$ $(S_{n-1} = b_{n-1} + 2 \cdot \underset{\text{quoziente}}{0})$

l'algoritmo si arresta nel momento in cui il quoziente è zero

Nota 1: l'algoritmo di conversione decimale binario genera le cifre della rappresentazione binaria dalla meno significativa alla più significativa!

VANNO ordinate all'inverso

$b_0 b_1 \ldots b_{n-1} \longrightarrow b_{n-1} b_{n-2} \ldots b_1 b_0$

Nota 2: l'algoritmo effettua tante divisioni quante sono le cifre binarie

Conversione BIN - DEC ?

Facile: prima dividevo per 2 e "prelevavo il resto"
Ora moltiplico per 2 e aggiungo il resto

$$b_0 + 2(b_1 + 2(b_2 + \ldots\ldots (b_{n-2} + 2b_{n-1}))\ldots)$$

$$S_{n-1} = b_{n-1}$$
$$S_{n-2} = b_{n-2} + 2S_{n-1}$$
$$S_{n-3} = b_{n-3} + 2S_{n-2}$$
$$\vdots$$
$$S_0 = b_0 + 2S_1$$

$\downarrow$

N_{10}

$\Leftarrow$ L'aritmetica che utilizzo (moltipl. e somme) è quella decimale

Notazioni Ottale ed Esadecimale

$D = \{0, 1, 2 \ldots 9\}$ $B = \{0, 1\}$ alfabeti decimale e binario

$O = \{0, 1, 2, \ldots 6, 7\}$ $EX = \{0, 1, \ldots, 9, \underset{10}{A}, \underset{11}{B}, \underset{12}{C}, \underset{13}{D}, \underset{14}{E}, \underset{15}{F}\}$

Ottale - 8 cifre

Esadecimale - 16 cifre

$$N_8 = O_{k-1} O_{k-2} \cdots O_1 O_0 \longrightarrow O_{k-1} \cdot 8^{k-1} + O_{k-2} \cdot 8^{k-2} + \ldots + O_0 8^0$$

$$N_{16} = e_{k-1} e_{k-2} \ldots e_1 e_0 \longrightarrow e_{k-1} \cdot 16^{k-1} + e_{k-2} + 16^{k-2} + \ldots + e_0 8^0$$

Sistemi posizionali pesati in base 8 e 16

Perché sono utili?

Permettono di rappresentare velocemente lunghe stringhe binarie. Le conversioni OCT $\leftrightarrow$ BIN e DEC $\rightarrow$ EX sono immediate!

$$N_2 = b_5\, b_4\, b_3\, b_2\, b_1\, b_0$$

$$\rightarrow \underbrace{b_5 \cdot 2^5 + b_4 \cdot 2^4 + b_3 \cdot 2^3}_{} + \underbrace{b_2 \cdot 2^2 + b_1 \cdot 2^1 + b_0 \cdot 2^0}_{} \quad \begin{cases} 000 \\ 8 \text{ valori} \\ 111 \end{cases}$$

$$\rightarrow (b_5 \cdot 2^2 + b_4 \cdot 2^1 + b_3 \cdot 2^0) \cdot 2^3 + (b_2 \cdot 2^2 + b_1 \cdot 2^1 + b_0 2^0)$$

$$\rightarrow \underbrace{(b_5 \cdot 2^3 + b_4 \cdot 2^1 + b_3 \cdot 2^0)}_{O_1} \underline{\underline{8^1}} + \underbrace{(b_2 \cdot 2^2 + b_1 \cdot 2^1 + b_0 \cdot 2^0)}_{O_0} \underline{\underline{8^0}}$$

$$N_8 = O_1\, O_0$$

Algoritmi di conversione

BIN → OCT

- Raggruppo i bit in terzine (estendo eventualmente a sinistra con qualche bit uguale a ∅ per ottenere una lunghezza multipla di 3)
- Sostituisco ogni terzina con la corrispondente cifra ottale

Es. 011111 ⟶ 3 7

11111_2

↓

37_8

OCT $\rightarrow$ BIN

Costruisco la rappresentazione binaria sostituendo ogni cifra ottale con la corrispondente rappresentazione binaria

Es. $754 \longrightarrow 111\ 101\ 100$

Per le conversioni EX $\leftrightarrow$ BIN procedo allo stesso modo raggruppando i bit in QUARTINE

Nota 2 : gli algoritmi di conversione

BIN <—> OCT e BIN <—> EX

si estendono naturalmente a

QUALSIASI sistema di rappresentazione

basato su un insieme di cifre X

tale che $|X| = 2^K$

per qualche K

→ Potenza di 2