1^ Esercitazione : Rappresentazione di numeri

A) RAPPRESENTAZIONE DEI NATURALI

Sistema posizionale (in base b ≥ 2)

$$c_{m-i}\, c_{m-2} \ldots c_1\, c_0 \quad = \quad \sum_{i = 0,\ldots,m-1} c_i\, b^i$$

con c_i { 0 , … , b-1 }

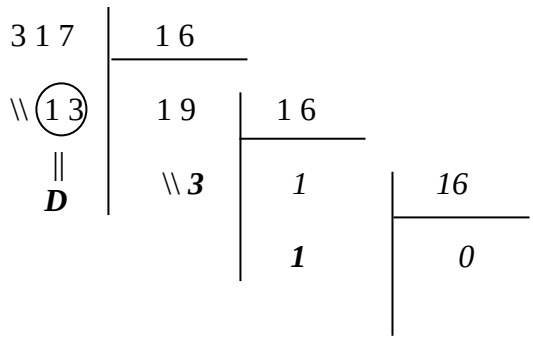
A.1) CAMBIAMENTI DI BASE

Problema : passare da N_a a N_b (con N un naturale ed a e b le basi desiderate)

Algoritmo : 1. dividi N_a ripetutamente per b_a finchè il quoziente non è 0

2. i resti di queste divisioni (convertite da base a a base b) danno le cifre dalla meno alla più significativa di N_b

Esempio 1 : esprimere in base 16 il numero 317



Pertanto $317_{10} = 13D_{16}$

Esempio 2 : convertire il numero 102202 da base 3 a base 5

Due strade :

a) eseguire $102202 /_3 12$ (cioè effettuare la divisione in base 3) => DIFFICILE

b) convertire 102202 in base 10 e poi convertire il risultato in base 5
- $102202_3 = 3^5 + 2\,3^3 + 2\,3^2 + 2 = 317_{10}$
- col procedimento dell'esempio precedente $317_{10} = 2232_5$

Prop. : *lavorando il aritmetica in base b si ha che*

1. $n_{m-1} \dots n_1 n_0$ *DIV* $b^i = n_{m-1} \dots n_i$ $r = n_{i-1} \dots n_0$
2. $n_{m-1} \dots n_1 n_0$ *MOD* $b^i = n_{i-1} \dots n_0$

Es: 353_{10} DIV 10^0 =35 (r=3) 1011_2 DIV 10_2 (cioè 2^1) = 101 r=1

Da ciò :

1. *Conversione da base 2 a base 4 :* considera i bit a coppie partendo dal meno significativo e traducile in base 4

   Esempio 3 : convertire 100111101 da base 2 a base $4=2^2$

   $1\ 00\ 11\ 11\ 01_2 = 1\ 0\ 3\ 3\ 1_4$

2. *Conversione da base 2 a base 8 :* considera i bit a triple partendo dal meno significativo e traducile in base 8

   Esempio 4 : convertire 100111101 da base 2 a base $8=2^3$

   $100\ 111\ 101_2 = 4\ 7\ 5_8$

3. *Conversione da base 2 a base 16=2^4 :* considera i bit a quadruple partendo dal meno significativo e traducile in base 16

   Esempio 5 : convertire 100111101 da base 2 a base 16

   $1\ 0011\ 1101_2 = 1\ 3\ D_{16}$

Vale anche il viceversa; in particolare :

1. *Conversione da base 8 a base 2 :* considera ogni cifra ottale e riscrivila come numero binario di 3 bit

2. *Conversione da base 16 a base 2 :* considera ogni cifra esadecimale e riscrivila come numero binario di 4 bit

A.2) ARITMETICA IN BASE b

Tutte le operazioni come in base 10 , ma fatte *modulo b* (Es.: $(1 + 1)_2 = 10_2$)
e quindi anche i riporti e i prestiti agiscono *modulo b*

Esempio 6 : sommare in base 2 i numeri 1001 e 111

```
  1 0 0 1  +
    1 1 1  =
¾¾¾¾¾¾¾¾¾¾
  1 0 0 0 0
```

Elaboratore lavora con parole di lunghezza fissa (diciamo n); quindi
1. se un numero è codificato con meno di n bit : inserimento in testa di zeri non significativi
2. se un numero è codificato con più di n bit : ne considera solo le n cifre meno significative (situazione di errore detta *overflow* e gestita con un'eccezione del processore)

Esempio 6 (continua) : usando parole lunghe quattro bit avremmo

1 0 0 1 +
1 1 1 =
¾¾¾¾¾¾¾¾
0 0 0 0

che è chiaramente un errore!!

B) RAPPRESENTAZIONE DEGLI INTERI

B.1) COMPLEMENTO A DUE

Ci sono vari standard per rappresentare interi in base b : con bit di segno, in complemento a uno o in complemento a due. I primi due rendono le operazioni di somma e sottrazione delicate (sono necessari controlli preliminari sul segno e sui valori assoluti degli operandi); col secondo, invece, la sottrazione si esegue semplicemente come somma dell'opposto (a patto di ignorare l'eventuale overflow derivante dalla somma di numeri negativi).

In complemento a due si ha che: $A = -c_{m-1} \times b^{m-1} + \sum_{i=0}^{m-2} c_i \times b^i$ (b= base, come al solito)

$$\overline{A} = \overline{c_{m-1} \ldots c_0} = \overline{c_{m-1}} \ldots \overline{c_0} + 1$$

dove $\overline{c_i} = 1 - c_i$.

Osservazioni:

1. il "range" dei numeri rappresentabili va da $-b^{m-1}$ *a* $b^{m-1} - 1$
2. lo zero è $\underbrace{0 \ldots 0}_{m}$
3. non viene considerata la sequenza $1\,\underbrace{0 \ldots 0}_{m-1}$
4. il bit più significativo NON E' un bit di segno (nel senso che per complementare un numero non basta complementarne il bit più significativo) ma è un bit indicatore del segno; infatti

$$c_{m-1} = \begin{cases} 1 & \text{se } c < 0 \\ 0 & \text{altrimenti} \end{cases}$$

Negli esempi seguenti supporremo sempre di lavorare **con parole di 8 bit.**

Esempio 7 : complementare i seguenti numeri binari : 10010 e 11011000

a) 1. complementa bit a bit il numero (00010010 diventa 11101101)
2. somma 1 (11101101 diventa 11101110)

b) 1. complementa bit a bit il numero (11011000 diventa 00100111)
2. somma 1 (100111 diventa 101000)

Mostriamo ora come le operazioni aritmetiche in questo caso siano molto semplici.

Esempio 8 : eseguire in base 2 le seguenti operazioni espresse in base 10 :
a) 6 + 8 b) – 6 + 8 c) 6 – 8 d) – 6 – 8

Anzitutto traduciamo 6 , – 6 , 8 e – 8 in base 2

. $6_{10} = 110_2$
. $-6_{10} = 11111010_2$
. $8_{10} = 1000_2$
. $-8_{10} = 11111000_2$

a) 110 + 1000 = 1110 (cioè 14_{10})
b) 11111010 + 1000 = 10 (cioè 2_{10})
c) 110 + 11111000 = 11111110 (cioè -2_{10})
d) 11111010 + 11111000 = 11110010 (cioè -14_{10})

C) RAPPRESENTAZIONE DEI REALI

Sistema posizionale (in base b ≥ 2)

$$c_{s-i}\, c_{s-2} \ldots c_1\, c_0\, c_{-1} \ldots c_{-t} \quad = \quad \sum_{i = -t,\ldots,s-1} c_i\, b^i$$

con $c_i \quad \{ 0 , \ldots , b\text{-}1 \}$

C.1) VIRGOLA FISSA

Riserva h bit per la parte intera (P.I.) e k bit per la parte frazionaria (P.F.) (h e k fissati)

Cambiamento di base : trasformare A , B_a in A , B_b

1. converti A_a in A_b (normale conversione per naturali)

2. moltiplica B_a per b_a
- la P.I. del risultato convertita in base b è la 1^ cifra di B_b

- itera il procedimento sulla P.F. del risultato
- finché la P.F. è 0, oppure sono state determinate tutte le cifre disponibili per esprimere la P.F. di B_b

Esempio 1 : convertire 17,416 in base 2 con 8 bit sia per P.I. che per P.F.

1. $17_{10} = 10001_2$
2. - 0,416 * 2 = 0,832 da cui P.I. = 0 P.F. = 0,832
   - 0,832 * 2 = 1,664 da cui P.I. = 1 P.F. = 0,664
   - 0,664 * 2 = 1,328 da cui P.I. = 1 P.F. = 0,328
   - 0,328 * 2 = 0,656 da cui P.I. = 0 P.F. = 0,656
   - 0,656 * 2 = 1,312 da cui P.I. = 1 P.F. = 0,312
   - 0,312 * 2 = 0,624 da cui P.I. = 0 P.F. = 0,624
   - 0,624 * 2 = 1,248 da cui P.I. = 1 P.F. = 0,248
   - 0,248 * 2 = 0,496 da cui P.I. = 0 P.F. = 0,496

   Perciò $0{,}416_{10} = 0{,}01101010_2$

Pertanto $17{,}416_{10} = 00010001\ ,\ 01101010_2$

N.B.: la versione binaria è un'approssimazione del numero decimale originale.
Infatti :

$$10001{,}0110101_2 = 2^4 + 1 + 2^{-2} + 2^{-3} + 2^{-5} + 2^{-7} = 17{,}4140625_{10}$$

Probl.: l'intervallo dei reali rappresentabile è piccolo e con approssimazioni grossolane

Esempio 2 : avendo a disposizione 32 bit e dandone 20 per la P.I. e 12 per la P.F. si ha
a) P.I. $\{ 2^{-19} + 1 , \dots , 2^{19} - 1 \} = \{ -524287 , \dots , 524287 \}$
b) si hanno a disposizione solo 3 o 4 cifre frazionarie (infatti $2^{12} = 4096$)

C.2) VIRGOLA MOBILE

Un reale r è rappresentato dalla terna $< s , m , e >$ dove

$$r = (-1)^s \cdot m \cdot b^e$$

e gli elementi della terna sono chiamati rispettivamente *bit di segno* , *mantissa* (o *significante*) e *esponente* . S indica il segno.
Tipicamente si adotta una forma normalizzata (tranne che per lo zero) in cui la mantissa è tale che :
1. la sua parte intera è nulla
2. la sua parte frazionaria inizia con una cifra non nulla

Banalmente $< s , m , e >$ soddisfa ciò se e solo se $b^{-1} \leq m < 1$.

Quindi, adottando la rappresentazione normalizzata,

$$r = (-1)^S \cdot 0{,}m \cdot b^e$$

dove
1. s è il bit di segno della mantissa
2. m (**N**) rappresenta la parte frazionaria del numero normalizzato (quindi la mantissa è un intero rappresentato con bit di segno)
3. e è l'esponente, rappresentato in complemento alla base (se b=2, in complemento a 2)

C.2.1) CAMBIAMENTO di BASE

Trasformare $< s , m_a , e_a >$ in $< s , m_b , e_b >$

1. applica il procedimento per numeri in virgola fissa a $(0,m_a \cdot a_e)_a$ ottenendo h , k_b
2. m_b e e_b sono la mantissa e l'esponente normalizzati di h , k_b

Nel seguito assumeremo di avere 1 bit per il segno, 8 per la mantissa e 4 per l'esponente.

Esempio 3 : convertire in base 2 il numero $0{,}09375_{10}$

1. applico il procedimento per P.F., ottenendo $0{,}09375_{10} = 0{,}00011_2 = 0{,}11 \times 2^{-3}$
2. la rappresentazione cercata è $< 0 , 11000000_2 , 1101_2 >$
(-3 in complemento a 2 è appunto 1101)

Esempio 4 : convertire in decimale i seguenti numeri in virgola mobile :

a) $< 0 , 10010000_2 , 0101_2 > = + (2^{-1} + 2^{-4}) \cdot 2^5 = 18_{10}$
b) $< 1 , 11001000_2 , 0111_2 > = - (2^{-1} + 2^{-2} + 2^{-5}) \cdot 2^7 = -100_{10}$
c) $< 1 , 10001000_2 , 1101_2 > = - (2^{-1} + 2^{-5}) \cdot 2^{-3} = -0{,}06640625_{10}$

N.B. : in base 2, intervallo rappresentato dando M bit alla mantissa e E all'esponente :

1. i numeri positivi sono $[0{,}1 \cdot 2^{-2^{E-1}+1} , 0{,}\underbrace{1\ldots1}_{M} \cdot 2^{2^{E-1}-1}]$
2. i numeri negativi sono $[-0{,}\underbrace{1\ldots1}_{M} \cdot 2^{2^{E-1}-1} , -0{,}1 \cdot 2^{-2^{E-1}+1}]$

C.2.2) OPERAZIONI tra REALI

1. Moltiplicazione (in base *b*)

$$< s_1 , m_1 , e_1 > * < s_2 , m_2 , e_2 > = < s , m , e >$$

dove 1. $s = \begin{cases} 1 & \text{se } s_1 = s_2 \\ 0 & \text{altrimenti} \end{cases}$

2. m ed e sono la mantissa e l'esponente normalizzati di $m_1 \cdot m_2 \cdot b^{e1+e2}$

N.B. : attenzione all'overflow degli esponenti!!

Analoga è la formula per la divisione.

Esempio 5 : eseguire in base 2 18 * 0,06640625

$< 0 , 10010000 , 0101 > * < 1 , 10001000 , 1101 > = < 1 , 10011001 , 0001 >$

Il risultato, convertito in base 10, è correttamente $-1,1953125$

2. Somma

$$< s_1 , m_1 , e_1 > + < s_2 , m_2 , e_2 > = < s , m , e >$$

- se $e_1 = e_2$

- $s = \begin{cases} s_1 & \text{se } (-1)^{s1} \cdot m_1 \geq (-1)^{s2} \cdot m_2 \\ s_2 & \text{altrimenti} \end{cases}$

- m ed e sono le normalizzazioni di m' ed e' definiti come :

(i) $e = e_1$

(ii) $m = \begin{cases} m_1 + m_2 & \text{se } s_1 = s_2 \\ | m_1 - m_2 | & \text{altrimenti} \end{cases}$

Es: $-0,3\times10^{-2} +0,4\times10^{-2} = +0,1\times10^{-2}$

- altrimenti (sia $e_1 < e_2$)

  - shift a destra di m_1 di $e_2 - e_1$ posizioni (inserendo 0 a sinistra)
  - porta così i numeri allo stesso esponente
  - procede come al punto precedente

Esempio 6 : eseguire in base 2 $18 - 100$

$< 0 , 10010000 , 0101 > - < 1 , 11001000 , 0111 > =$

(e1=+5, e2=+8, $e_2 - e_1$=2)

$= < 0 , \mathbf{00}100100 , 0111 > - < 1 , 11001000 , 0111 > =$

$= < 0 , 10100100 , 0111 >$

che corrisponde a -82

N.B. : nell'operazione di shift ci può essere perdita di cifre significative !!

Esempio 7 : si esegua in base 2 $18 - 0,06640625$

$< 0 , 10010000 , 0101 > - < 1 , 10001000 , 1101 > =$

= < 0 , 10010000 , 0101 > – < 1 , 00000000 , 0101 > =
= < 0 , 10010000 , 0101 >

Nello shift si è addirittura perso il secondo operando !!!!

D) ESERCIZI DA SVOLGERE

1. Convertire il numero decimale 1342 nelle basi 2 , 3 , 5 , 6 , 7 , 9 , 16

2. Convertire il numero binario 1011100 nelle basi 3 , 4 , 5 , 8 , 10 , 16

3. Dimostrare la proposizione data (*Sugg. : usare il teorema fondamentale dell'algebra*)

4. Convertire in base 2 i seguenti numeri ottali : 742 , 1176 , 253 , 1064

5. Convertire in base 2 i seguenti numeri esadecimali : A231 , 1BC , 1045 , FA2

6. Convertire in base 2 e poi sommare le seguenti coppie di naturali espressi in base 10. Quali di queste somme danno problemi di overflow rappresentando i numeri con 8 bit (= 1 byte) ?
   a) 115 , 64 b) 83 , 12 c) 197 , 94 d) 241 , 16 e) 230 , 25 f) 107 , 44

7. Rappresentare in complemento a due gli opposti dei seguenti interi e convertirli in base 10 :
   a) 1001 b) 11110010 c) 11010010 d) 10010
   e) 111000 f) 11001011 g) 1000111 h) 11101110

8. Eseguire in base 2 con numeri rappresentati come byte le seguenti operazioni decimali :
   a) 16 – 4 b) 18 – 47 c) – 49 + 54 d) – 4 –101 e) 92 + 23

9. Convertire in binario i seguenti numeri razionali, usando 8 bit sia per la P.I. Che per la P.F.
   a) 27,311 b) 8,92c) 0,511 d) 107,88 e) 49,266

10. Convertire in decimale i seguenti numeri razionali binari espressi con le convenzioni dell'esercizio precedente
    a) 11001,11100011 b) 100011,00100111 c) 110,00010111 d) 111011,00001111

11. Convertire in binario i seguenti numeri razionali, usando 1 bit di segno, 8 bit per la mantissa e 4 bit per l'esponente
    a) 27,311 b) - 8,92 c) 0,511 d) – 107,88
    e) 49,266 f) 0,000615 g) – 0,00215 h) 1500,615

12. Convertire in decimale i seguenti numeri razionali binari espressi con le convenzioni dell'esercizio precedente
    a) < 0 , 11100000 , 0111 > b) < 1 , 01100100 , 1110 >
    c) < 1 , 10010000 , 0011 > d) < 0 , 01001000 , 0100 >

13. Moltiplicare e sommare tutte le possibili coppie di numeri dell'esercizio precedente (cioè a+b , a+c , a+d , b+c , b+d , c+d e analogo per *) segnalando eventuali problemi di overflow e di perdita di cifre significative.