

FONDAMENTI DI INFORMATICA

Corsi di Laurea in Ingegneria Elettrica,
Ingegneria Gestionale, Ingegneria Meccanica

Universit degli Studi di Firenze

A. A. 2001-2002

APPUNTI DELLE LEZIONI

Seconda parte

Prof. Alessandro Fantechi

1. I sistemi di numerazione modulo n

In un *sistema di numerazione modulo n* si dice che $X \bmod n = q$ (cioè, X è equivalente modulo n a q) se $X = n \cdot p + q$ e $q < n$: divido cioè X per n e considero il resto. Ovviamente le operazioni tra due numeri in questi sistemi devono venire ridefinite in modo da dare dei risultati modulo n. Per esempio, la somma modulo n è definita come

$$(X+Y) \bmod n = q \text{ se } X+Y = n \cdot p + q \text{ e } q < n.$$

Consideriamo un sistema di numerazione *modulo 5*: questo sistema di numerazione è composto dai numeri da 0 a 4 e la somma di due numeri deve dare come risultato sempre un numero dello stesso insieme, calcolato nel seguente modo:

$$\begin{aligned} 0+3 &= 3 \\ 1+3 &= 4 \\ 2+3 &= 0 \\ 3+3 &= 1 \\ 4+3 &= 2 \\ &\dots \end{aligned}$$

Consideriamo ora il *sistema di numerazione modulo 100*: l'insieme dei valori di questo sistema è $\{0, 1, 2, \dots, 99\}$, cioè l'insieme dei numeri di al più due cifre. La somma di due numeri in questo sistema dovrà sempre dare come risultato un numero di due cifre che può essere ottenuto nel seguente modo:

$$\begin{array}{r} 73 + \\ 35 = \\ \hline 108 \\ \downarrow \\ 08 = (73+35) \bmod 100 \end{array}$$

Cioè, basta considerare le due cifre meno significative della somma aritmetica (tralasciando quindi l'ultimo riporto).

2. La rappresentazione dei numeri naturali.

Si vogliono illustrare in questo paragrafo le convenzioni per rappresentare i numeri naturali, cioè gli elementi dell'insieme \mathbb{N} caratterizzato nel seguente modo:

- $0 \in \mathbb{N}$
- se $n \in \mathbb{N}$, anche il successore di n appartiene ad \mathbb{N} (**succ**(n) $\in \mathbb{N}$)
- \mathbb{N} non contiene altri elementi diversi da quelli ottenibili tramite le due proprietà precedenti.

Il concetto fondamentale è la distinzione tra l'oggetto in quanto tale e la sua rappresentazione; gli algoritmi che mostreremo (somma, prodotto, ecc.) sono algoritmi che, data la rappresentazione degli operandi, producono la rappresentazione del risultato. Quindi tutti gli algoritmi sono dipendenti

dalla rappresentazione impiegata.

2.1. La rappresentazione posizionale in una base qualsiasi.

In generale ogni rappresentazione è una funzione che associa ad ogni elemento dei naturali una sequenza di simboli: è auspicabile per ogni rappresentazione che oggetti (numeri) distinti abbiano differenti rappresentazioni e che la rappresentazione di ogni oggetto sia unica. Le rappresentazioni usate sui calcolatori impiegano tutte sequenze finite di simboli, tali quindi da rappresentare insiemi finiti di naturali. Per ogni rappresentazione deve essere definita una base: tale base è costituita dal numero di simboli diversi impiegato dalla rappresentazione. La base in cui siamo abituati a pensare è la base 10.

Un numero naturale n è rappresentato in una base B da una stringa di simboli c_i tali che

$$= \sum_{i=0}^{p-1} c_i B^i \quad (1)$$

ovverosia

$$n = c_{p-1} B^{p-1} + c_{p-2} B^{p-2} + \dots + c_1 B + c_0 \quad \text{con } 0 \leq c_i < B$$

Il problema, dato un naturale da rappresentare, è individuare i c_i ; viceversa, la formula dà un modo per calcolare il valore rappresentato, una volta noti i coefficienti. Per risolvere il problema del calcolo dei coefficienti posso riscrivere n in un altro modo, cioè come:

$$n = B \cdot q + r$$

Noti i due naturali n e B , esiste una sola coppia di naturali r e q (il resto e il quoziente della divisione intera di n per B) che soddisfano l'equazione, con $r < B$. Quindi la cifra meno significativa della rappresentazione in base B di n si ottiene come resto della divisione di n per B ; iterando il procedimento, cioè rieseguendo la divisione del quoziente già ottenuto per B , si ottengono via via nuovi resti che costituiscono i vari coefficienti (ad es. se consideriamo il quoziente di n diviso B , che risulta essere

$c_{p-1} B^{p-2} + c_{p-2} B^{p-3} + \dots + c_1$, dividendolo ancora per B otterrò come resto il coefficiente c_1). Il procedimento termina quando si ottiene un quoziente uguale a 0. In questo modo viene automaticamente determinata la lunghezza della sequenza di simboli della rappresentazione.

Poniamo, ad esempio, di voler calcolare la rappresentazione in base 2 del numero naturale 75 (mod sta ad indicare l'operazione di modulo vista nel paragrafo 1, mentre div sta ad indicare la divisione intera):

$$c_0 = 75 \text{ mod } 2 = 1$$

$$q_0 = 75 \text{ div } 2 = 37$$

$$c_1 = 37 \bmod 2 = 1$$

$$c_2 = 18 \bmod 2 = 0$$

$$c_3 = 9 \bmod 2 = 1$$

$$c_4 = 4 \bmod 2 = 0$$

$$c_5 = 2 \bmod 2 = 0$$

$$c_6 = 1 \bmod 2 = 1$$

$$q_1 = 37 \operatorname{div} 2 = 18$$

$$q_2 = 18 \operatorname{div} 2 = 9$$

$$q_3 = 9 \operatorname{div} 2 = 4$$

$$q_4 = 4 \operatorname{div} 2 = 2$$

$$q_5 = 2 \operatorname{div} 2 = 1$$

$$q_6 = 1 \operatorname{div} 2 = 0$$

quindi

$$\rho_2(75) = 1001011_2$$

dove $\rho_B(x)$ è una funzione che produce la rappresentazione posizionale del naturale x in una certa base

B . Nel seguito, prevalentemente ometteremo l'identificatore di funzione ρ e indicheremo con il simbolo

= il fatto che il naturale n è rappresentato da una stringa di simboli, cioè

$$75 = 1001011_2$$

Questo algoritmo consente di ricavare la rappresentazione in base B a partire dalla rappresentazione in base 10 del numero; la sommatoria (1) costituisce invece l'algoritmo di conversione in senso inverso, cioè da base B a base 10. In generale, i due algoritmi convertono da base C a base B e da base B a base C rispettivamente, dove C è la base in cui si compiono le operazioni: nel nostro caso, $C = 10$ perchè noi sappiamo eseguire operazioni solo in base 10.

La rappresentazione posizionale gode di alcune proprietà:

a) La potenza n -sima di una base B è sempre rappresentata da 1 seguito da n zeri.

Ad esempio:

$$2^4 = 10000_2$$

$$5^2 = 100_5$$

$$10^2 = 100_{10}$$

$$11^2 = 100_{11}$$

b) Il massimo numero rappresentabile con una sequenza di m cifre è $B^m - 1$; con tale sequenza si rappresentano quindi B^m oggetti diversi. In altre parole una sequenza di m cifre rappresenta i naturali in un sistema di numerazione *modulo* B^m e quindi tutte le operazioni daranno risultati *modulo* B^m .

c) La moltiplicazione di un numero naturale rappresentato in base B per B stesso è ottenibile trasladando verso sinistra tutte le cifre della rappresentazione ed inserendo uno 0 come cifra meno significativa. Quindi se il numero n è rappresentabile con p cifre, $n \cdot B$ è rappresentabile con $p+1$ cifre. Allo stesso modo si ottiene $n \cdot B^s$ trasladando di s passi la rappresentazione verso sinistra ed inserendo s zeri da destra.

d) La divisione intera di un numero naturale rappresentato in base B per B stesso è ottenibile traslando verso destra tutte le cifre della rappresentazione ed inserendo uno 0 come cifra più significativa. Quindi se il numero n è rappresentabile con p cifre, $n \text{ div } B$ è rappresentabile con $p-1$ cifre. Allo stesso modo si ottiene $n \text{ div } B^s$ traslando di s passi la rappresentazione verso destra ed inserendo s zeri da sinistra.

3. La rappresentazione dei numeri relativi.

La rappresentazione dei numeri naturali è la cosiddetta rappresentazione *unsigned*, cioè priva di segno; passiamo ora al problema di usare sequenze finite di simboli per rappresentare numeri con segno (o interi relativi). Bisogna introdurre necessariamente nuovi concetti. Dato un numero relativo n si può considerare il suo opposto \bar{n} definito come

$$\bar{n} = -n$$

cioè tale che $n + \bar{n} = 0$ (l'operatore $+$ è l'operatore aritmetico definito sui numeri relativi). L'operazione che trasforma un numero nel suo opposto si dice *complementazione*. Se siamo in un sistema di numerazione modulo B^p , l'opposto di n è quel numero \bar{n} tale che $(n + \bar{n}) \text{ mod } B^p = 0$ (dove l'operatore $+$ è l'operatore di somma modulo B^p). In questo caso, esistono infiniti numeri che rispettano tale condizione e

$$\bar{n} = k * B^p - n, \text{ per } k = 0, 1, 2, \dots$$

Se consideriamo $k=0$ riavremo la definizione di opposto sui numeri relativi.

Partendo dal concetto di opposto, si attua una suddivisione dei B^p numeri del nostro sistema di numerazione in due parti, una sui cui elementi sono mappati i numeri positivi, e l'altra su cui sono mappati i negativi. Una possibile suddivisione per $B=2$ è quella che porta ad avere la prima cifra della sequenza uguale a 0 per i numeri positivi, ed uguale ad 1 per i negativi.

Secondo questo criterio, si possono individuare almeno tre modi di rappresentare i numeri relativi, che mantengano delle proprietà aritmetiche significative: in *modulo e segno*, in *complemento alla base* e in *complemento alla base meno 1*. Di questi ne vedremo in dettaglio i primi due.

Modulo e segno.

Per ogni numero relativo n , la sequenza di simboli che rappresenta $|n|$ (il valore assoluto di n) viene estesa aggiungendo un nuovo simbolo s nella posizione più significativa, con $s \in \{+, -\}$. Quindi, chiamata r la rappresentazione di $|n|$, n è rappresentato dalla sequenza $\langle s, r \rangle$.

Ad esempio, per $B=2$ e $p=4$

$$+5 = 0101$$

$$-5 = 1101$$

Lo 0 viene rappresentato come 00..00, la cifra del segno è sempre 0, quindi la configurazione 100..00 è ridondante e non è associata ad alcun numero. I numeri relativi rappresentabili in questo modo con p cifre in modulo e segno sono quelli compresi nell'intervallo $[-(2^{p-1}-1), 2^{p-1}-1]$

Complemento alla base.

La rappresentazione dei numeri in complemento alla base è la seguente: supponiamo che i numeri relativi da rappresentare siano in tutto M e $M=B^p$, allora

$$n=r, \text{ se } n \geq 0, \text{ dove } r \text{ è la rappresentazione di } |n| \text{ su } p \text{ cifre;}$$

$$n=r', \text{ se } n < 0, \text{ dove } r' \text{ è la rappresentazione del naturale } B^p - |n|.$$

Quindi l'opposto di n in questo caso è B^p-n e lavoriamo in un sistema di numerazione modulo B^p e con $k=1$, infatti la somma di un numero e del suo opposto produce come risultato la rappresentazione di B^p (ovviamente su $p+1$ cifre).

Se consideriamo $B=2$, tutti i numeri positivi hanno 0 nella posizione più significativa della rappresentazione, mentre i negativi hanno 1. Ad esempio, su 4 cifre

$$+5 = 0101$$

$$-5 = 1011$$

Infatti $1011_2=11$ e $2^4-5=16-5=11$.

I numeri rappresentabili con p cifre sono quelli compresi nell'intervallo $[-2^{p-1}, 2^{p-1}-1]$: la rappresentazione 100...00 (1 seguito da $p-1$ zeri) è associata a -2^{p-1} .

Ogni numero ha un numero minimo di cifre necessario per essere rappresentato, tuttavia accade che sia fissata a priori la lunghezza della stringa della rappresentazione (dalla lunghezza della parola di memoria, ad esempio) e sorge quindi il problema di come portare su p cifre le rappresentazioni di numeri che ne richiedono s , con $s < p$. Come si rappresentano 5 e -5 su 16 cifre? Tutto quello che bisogna fare è propagare la cifra del segno.

$$+5 = 0000000000000101$$

$$-5 = 1111111111111011$$

4. Le operazioni sui numeri relativi.

Consideriamo ora le varie rappresentazioni e descriviamo l'algoritmo delle operazioni aritmetiche in ogni caso. Per gli esempi si considererà d'ora in avanti solo la base 2.

Modulo e segno.

L'operazione di complementazione con $n = \langle s, r \rangle$ produce una nuova coppia $\langle s', r' \rangle$, con $s'=0$ se $s=1$, $s'=1$ se

$s=0$.

L'operazione di somma aritmetica del tipo $n+n'$ (con $n=\langle s,r \rangle$ e $n'=\langle s',r' \rangle$) è calcolata per le rappresentazioni nel seguente modo

$$\langle s, |r| + |r'| \rangle \text{ se } s=s'$$

$$\langle s, |r| - |r'| \rangle \text{ se } s \neq s' \text{ e } |r| > |r'|$$

$n+n' =$

$$\langle s', |r'| - |r| \rangle \text{ se } s \neq s' \text{ e } |r'| > |r|$$

$$\langle +, 0 \rangle \text{ se } s \neq s' \text{ e } |r'| = |r|$$

In modo analogo viene definita la rappresentazione della differenza $n-n'$.

Gli operatori $+$ e $-$ usati nella parte destra sono gli operatori aritmetici che lavorano sulle rappresentazioni dei naturali. Le operazioni di moltiplicazione e divisione per potenze di 2 vengono eseguite traslando a sinistra o a destra la sola parte r della coppia $\langle s,r \rangle$.

Rappresentazioni in complemento

Se lavoriamo in un sistema di numerazione modulo B^P , abbiamo che

$$C-A = (C-A)_{\text{mod } B^P} = (C+(-A))_{\text{mod } B^P} = (C + (B^P-A))_{\text{mod } B^P},$$

dove B^P-A è l'opposto di A .

Quindi, se avessimo a disposizione un modo semplice di calcolare B^P-A , avremmo ridotto il calcolo di una sottrazione al calcolo di un'addizione. Il fatto che si lavori modulo una potenza della base in cui si rappresentano i numeri rende appunto semplice questo calcolo. Come esempio supponiamo di lavorare con i numeri rappresentati in base 10 in un sistema modulo 10^4 : avremo quindi che

$$B^P = 10^4 = 10000 = 9999+1.$$

Di conseguenza (B^P-A) può essere riformulato come

$$B^P-A = 10^4-A = (9999-A) + 1;$$

poichè stiamo considerando numeri modulo B^P , avremo che $A < 10^4$ (cioè A è un numero rappresentabile in base 10 con 4 cifre), quindi, per calcolare, ad esempio, B^P-3456 eseguiremo

$$\begin{array}{r} 9999 - \\ \underline{3456} \\ 6543 + \\ \underline{\quad 1} \\ 6544 \end{array}$$

Si può notare come sia estremamente semplice effettuare la prima sottrazione, che non comporta mai riporti e utilizza la semplice sostituzione delle cifre del sottraendo secondo la seguente corrispondenza:

$$0 \rightarrow 9, 1 \rightarrow 8, 2 \rightarrow 7, 3 \rightarrow 6, 4 \rightarrow 5, 5 \rightarrow 4, 6 \rightarrow 3, 7 \rightarrow 2, 8 \rightarrow 1, 9 \rightarrow 0$$

Il numero ottenuto con questa regola (nell'esempio, 6543) viene detto *complemento a 9* (alla base -1) di

3456; il numero ottenuto sommandoci 1, cioè $B^p - A$, viene detto *complemento a 10* (alla base).

Il calcolo sopra descritto diventa ancora più semplice per la base 2: infatti, la regola di sostituzione delle cifre diventa uno scambio di 0 con 1 e viceversa (ciò che viene detto anche *complemento bit a bit*, cioè ogni cifra può essere trattata separatamente indipendentemente dalla sua collocazione nella sequenza). Questa operazione è molto semplice anche dal punto di vista circuitale.

A. Complemento alla base.

Data la rappresentazione r del numero relativo n , l'operazione di complementazione di n produce il numero opposto \bar{n} , la cui rappresentazione r' si può definire anche come segue:

$$\bar{n} = B^p - r = B^p - r + 1 - 1 = (B^p - 1) - \sum_{i=0}^{p-1} c_i \cdot B^i + 1$$

e quindi

$$\bar{n} = \left(\sum_{i=0}^{p-1} (B-1) \cdot B^i \right) - \sum_{i=0}^{p-1} c_i \cdot B^i + 1.$$

Se la cifra significativa (cioè uguale a 1) più a destra nella rappresentazione di n è quella di indice k , ovvero $c_k=1$, per $i=0, \dots, k-1$, possiamo scrivere anche che

$$\bar{n} = \sum_{i=k+1}^{p-1} [(B-1) - c_i] \cdot B^i + [B - c_k] \cdot B^k + \sum_{i=0}^{k-1} 0 \cdot B^i$$

In altre parole, se $r = a_{p-1} \dots a_k 0000$, cioè le k cifre meno significative sono uguali a 0, r' è ottenuta lasciando invariati i k zeri, complementando alla base la cifra successiva e complementando alla base meno 1 tutte le altre cifre.

Ad esempio,

$$\begin{aligned} +10 &= 01010_2 \\ -10 &= 10110_2 \end{aligned}$$

Si riconsiderino le proprietà c) e d) della notazione posizionale: per i numeri relativi in complemento a 2 la moltiplicazione di un numero per 2^s consiste nel traslare verso sinistra la rappresentazione di s posizioni (mantenendo invariato il segno) e nell'inserire s zeri a destra. La divisione del numero per 2^s consiste invece nello spostamento verso destra delle cifre della rappresentazione di s posizioni e nell'inserire a sinistra s zeri o s uno, a seconda che il numero sia positivo o negativo, rispettivamente.

Poichè la caratteristica fondamentale delle rappresentazioni che usano il complemento alla base è quella di eseguire l'operazione di somma senza esaminare il segno degli addendi (infatti la seguente proprietà dei numeri relativi viene conservata $n - n' = n + \bar{n}$), come si deve fare invece per la rappresentazione in modulo e segno, dovremo definire un operatore sulle rappresentazioni tale $r + r'$ rappresenti la somma $n + n'$ modulo B^p .

A questo punto resta da vedere che differenza c'è tra il + aritmetico sui numeri relativi e quello sulla loro rappresentazione. Dati due numeri relativi n ed n' , la loro somma non sarà necessariamente rappresentata dalla somma delle rappresentazioni. Questo è il punto fondamentale di differenza tra l'operatore sui numeri e quello sulle rappresentazioni: siccome l'insieme di numeri rappresentabili è finito, non è detto che la somma di due numeri rappresentabili appartenga all'insieme. Il risultato dell'operatore corrisponde al troncamento del risultato sulle cifre usate nella rappresentazione; in realtà quello che si calcola è la rappresentazione (in complemento a 2 o a 1) di $(n+n') \bmod B^P$. Se il risultato è un numero più grande di B^{P-1} , la rappresentazione non è esatta.

Ad esempio, in complemento a 2, abbiamo:

$$10 = 01010_2 \text{ e } 11 = 01011_2$$

$$10 + 11 = 21 \text{ mentre}$$

$$01010_2 + 01011_2 = 10101_2 = -11$$

Il risultato corretto si avrebbe su sei cifre, cioè $010101_2 = 21$; in questi casi si dice che si è avuta una situazione di *overflow*.

4.1 Un sommatore

Proviamo a definire il comportamento, cioè la risposta, di una rete che calcola la somma di due qualsiasi cifre della base 2; come succede nella somma per le rappresentazione in base 10, si considera anche la possibilità di avere un riporto in ingresso (dalla somma delle due cifre precedenti della rappresentazione nel caso che si debba iterare il comportamento per effettuare la somma tra due sequenze di cifre) e in uscita (per la somma delle due cifre successive). Il comportamento atteso della rete si può descrivere mediante una tabella, detta *tabella di verità*. Da tabelle di questo tipo si possono ricavare espressioni i cui operatori sono direttamente realizzati da operatori elementari della rete.

Gli operatori base realizzati direttamente da circuiti elettronici sono la somma logica (componente OR), il prodotto logico (componente AND), e il complemento a 1 (componente NOT). I componenti logici AND, OR e NOT sono essi stessi definibili mediante tabelle di verità nel modo che segue; le colonne indicate con a e b rappresentano gli ingressi della rete, l'altra colonna indica il risultato.

<u>a</u>	<u>NOT</u>	<u>a</u>	<u>b</u>	<u>AND</u>	<u>a</u>	<u>b</u>	<u>OR</u>
0	1	0	0	0	0	0	0
1	0	0	1	0	0	1	1
		1	0	0	1	0	1
		1	1	1	1	1	1

La tabella di verità dà una definizione astratta del comportamento della rete, dalla quale è possibile ottenere automaticamente una descrizione eseguibile. Il suo significato è intuitivo: per ogni combinazione di valori in ingresso, si specifica qual è il risultato fornito dalla rete descritta. Reti ricavabili da tabelle di verità sono dette reti *combinatorie*: il loro funzionamento è indipendente dal

tempo, cioè per ogni combinazione di valori degli ingressi viene prodotto sempre lo stesso risultato. Definiamo ora la tabella di verità di una rete che prende tre cifre binarie in ingresso (a, b ed r), e produce il valore di due variabili in uscita (s, che rappresenta il valore della somma, rappresentato su una sola cifra, ed r', che rappresenta il valore dell'eventuale riporto).

a	b	r	s	r'
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

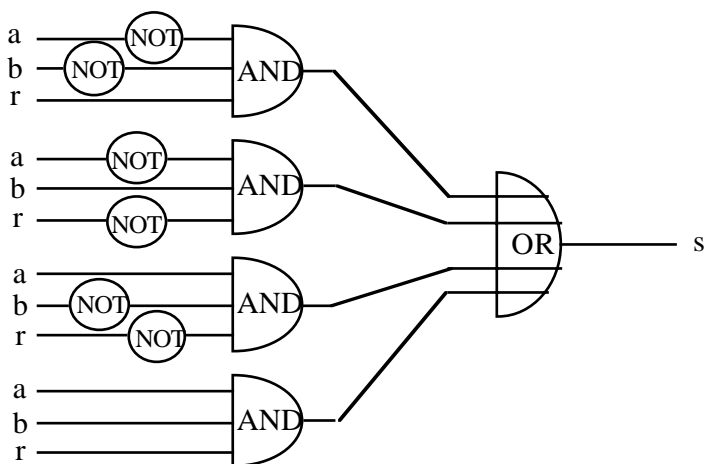
In base a questa tabella di verità posso costruire una rete (usando componenti AND, OR e NOT) il cui comportamento esterno è quello descritto. Per realizzare tale rete occorre per prima cosa ricavare dalla tabella di verità le seguenti espressioni, una per uscita, che sono costituite dalla somma di tanti termini prodotto quanti sono gli 1 nella tabella della funzione corrispondente; in ogni termine prodotto è riportato ogni ingresso, complementato se il corrispondente valore nella riga considerata è 0:

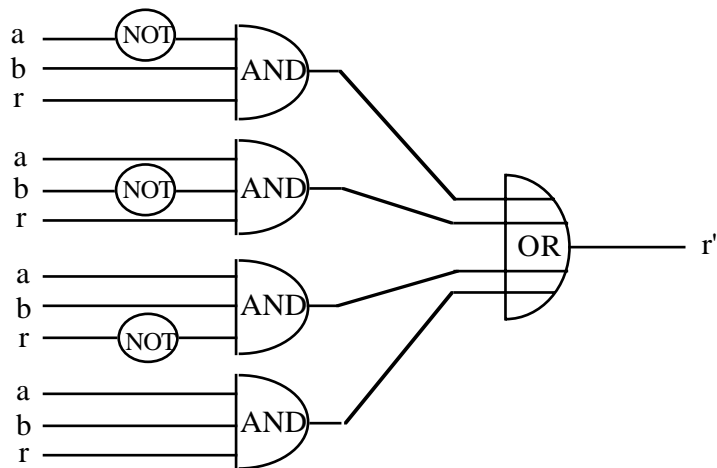
$$s_i = \bar{a}_i \cdot \bar{b}_i \cdot r_{i-1} + \bar{a}_i \cdot b_i \cdot \bar{r}_{i-1} + a_i \cdot \bar{b}_i \cdot \bar{r}_{i-1} + a_i \cdot b_i \cdot r_{i-1}$$

$$r_i = \bar{a}_i \cdot b_i \cdot r_{i-1} + a_i \cdot \bar{b}_i \cdot r_{i-1} + a_i \cdot b_i \cdot \bar{r}_{i-1} + a_i \cdot b_i \cdot r_{i-1}$$

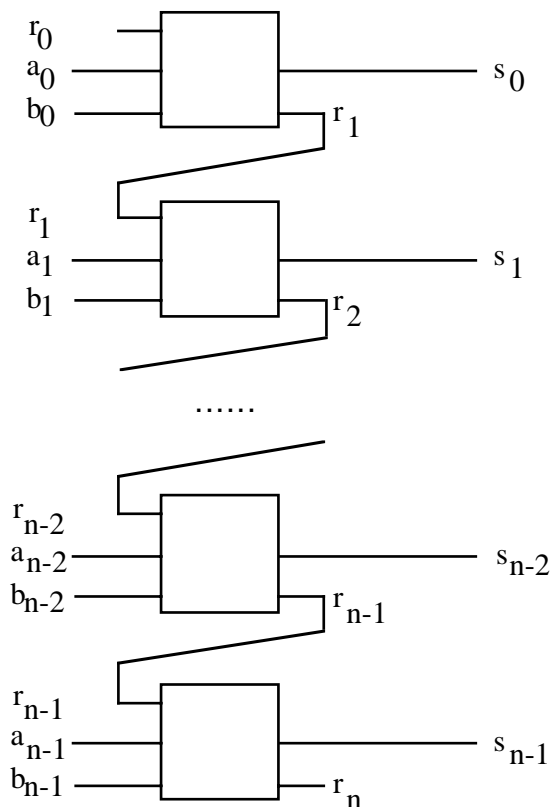
L'operatore + indica la somma logica, * indica il prodotto logico, mentre, ad esempio, \bar{a} , indica il complemento a 1 di a.

Le reti che si ricavano dalle espressioni r_i e s_i sono mostrate di seguito come esempio:





La somma di due numeri rappresentati con sequenze di n cifre si calcola collegando iterativamente secondo lo schema n di queste reti (a_i , b_i e r_i rappresentano rispettivamente le cifre di uguale posizione nelle due sequenze A e B ed il riporto dello stadio $(i-1)$ della rete, mentre s_i ed r_{i+1} rappresentano la cifra i -esima della rappresentazione della somma ed il riporto per lo stadio $(i+1)$ della rete; l'indice 0 indica l'elemento relativo alle cifre meno significative delle sequenze).



La rete ottenuta (*sommatore* su n bit) può essere quindi utilizzata per sommare due numeri unsigned di n bit. Se il riporto r_n è uguale a 1, allora si ha una situazione di overflow, altrimenti il risultato della somma è rappresentabile su n bit. Una rete diversa, detta *sottrattore*, realizzerà la differenza tra due numeri senza segno.

Se si vogliono sommare due numeri in modulo e segno, occorrerà utilizzare sia il sommatore che il sottrattore, a seconda che i segni dei due addendi siano concordi o discordi. Entrambe le reti lavoreranno sugli $(n-1)$ bit dei moduli. Il bit di riporto r_{n-1} indicherà la situazione di overflow.

Se si vogliono sommare due numeri rappresentati in complemento a due si può direttamente utilizzare il sommatore a n bit. La condizione per verificare che la somma eseguita da questa rete abbia dato un risultato corretto (cioè un risultato rappresentabile su n cifre, senza overflow) è in questo caso la seguente:

se $r_n=r_{n-1}$ allora la somma è corretta,

se $r_n \neq r_{n-1}$ allora la somma ha causato un supero di capacità.

Una stessa rete sommatore, essendo l'operatore di somma che realizza indifferente ai segni degli operandi, può eseguire anche la differenza di due numeri per le rappresentazioni in complemento a 2. Infatti, date due rappresentazioni di n cifre dello stesso tipo, A e B , relative a due numeri a e b , possiamo fare le seguenti asserzioni. Siano A , B (o $B' = 2^{n-1} - B$, cioè il complemento ad 1 di B , che si ottiene negando tutti i bit di B) ed r_0 i tre ingressi della rete completa, avremo che il risultato fornito dal sommatore per

$A + B + 0$, (cioè con $r_0 = 0$) è la rappresentazione di $(a+b)_{\text{mod } 2^n}$, cioè la somma in complemento a 2 di a e b , mentre

$A + B' + 1$, (cioè con $r_0 = 1$) è la rappresentazione di $(a-b)_{\text{mod } 2^n}$, cioè la sottrazione in complemento a 2 di a e b .

Da un punto di vista circuitale avremo bisogno di alcune reti aggiuntive per realizzare la differenza di due rappresentazioni: si deve infatti realizzare il complemento a 1 della rappresentazione B . Questa consisterà semplicemente in una serie di operatori NOT, ciascuno applicato ad una cifra della sequenza B (ovverosia $B' = \text{NOT}(B)$). Se vogliamo realizzare con lo stesso sommatore *entrambe* le operazioni aritmetiche sugli interi, servirà un'altra rete per selezionare il particolare ingresso del sommatore che serve per eseguire la somma o la differenza (cioè, B o $\text{NOT}(B)$), e 0 od 1 sulla cifra del riporto iniziale, nel caso del complemento a 2).

4.2. Moltiplicazione e divisione

La moltiplicazione tra due numeri unsigned può essere effettuata in base 2 usando lo stesso algoritmo che siamo soliti usare in base 10: si moltiplica il moltiplicando per ogni cifra del moltiplicatore, traslando a sinistra ogni risultato di tanti posti quanto è il numero d'ordine della cifra; si sommano poi i risultati intermedi. Si può notare che in base 2 la moltiplicazione di un numero per una cifra, che in base 10 richiede la conoscenza delle "tabelline", si riduce a dare come risultato il numero stesso se la cifra è 1, e il numero 0 se la cifra è 0.

La moltiplicazione di due numeri rappresentati su n bit dà un risultato che in generale è rappresentabile su $2n$ bit; per questo nei calcolatori si hanno spesso istruzioni di moltiplicazione che prelevano gli operandi da due registri e ritornano il risultato in un doppio registro. In questo caso non si possono avere situazioni di overflow. Tuttavia, moltiplicazioni per una potenza della base, se vengono eseguite sul registro stesso che contiene il moltiplicando (di n bits) con l'algoritmo delle traslazioni mostrato in precedenza, possono ovviamente dare overflow.

Anche per la divisione si può applicare il consueto algoritmo che usiamo in base 10: l'operazione elementare, che consiste nel vedere se e quante volte il divisore "sta" in una porzione del dividendo o in un risultato intermedio, in base due consisterà semplicemente nel ritornare il valore 0, se il divisore è maggiore di tale porzione o risultato intermedio, e il valore 1 altrimenti. Una divisione di un numero di n bit per un altro numero di n bit darà sempre un risultato rappresentabile in n bit; non si avranno quindi situazioni di overflow, a parte il caso di "divisione per zero".

Poichè in molti calcolatori l'operazione di prodotto di due numeri di n bit fornisce un risultato rappresentato su $2n$ bit, l'operazione inversa di divisione viene effettuata con un dividendo di $2n$ bit e un divisore di n bit, producendo un quoziente e un resto entrambi di n bit. In questo caso si può avere una situazione di overflow: si pensi ad es. al caso in cui il divisore valga 1, in cui perciò il quoziente risulta uguale al dividendo. Un modo semplice per verificare la situazione di overflow consiste nel vedere se il divisore sta più di una volta nel numero formato dagli $n+1$ bit più significativi del dividendo: infatti se chiamiamo tale numero p , il dividendo D sarà uguale a $p \cdot 2^{n-1+r}$. Se il divisore d sta più di una volta in p , cioè $d \leq p/2$, avremo: $D/d \geq (p \cdot 2^{n-1+r}) / (p/2)$, cioè $D/d \geq 2^n + 2r/p$, e quindi $D/d \geq 2^n$. Questo significa che il quoziente non è rappresentabile su n bit.

Le operazioni di moltiplicazione e divisione tra numeri relativi si effettueranno operando sui moduli dei numeri e considerando separatamente il segno, con la consueta regola che il segno del risultato è positivo se i segni degli operandi sono concordi e negativo altrimenti.

Gli algoritmi proposti per la risoluzione di moltiplicazioni e divisioni non necessariamente sono i più efficienti; in pratica vengono spesso usati altri algoritmi, che non tratteremo in questa sede.

5. La rappresentazione dei reali.

Mediante rappresentazioni posizionali si possono trattare anche i numeri reali: un qualunque numero reale può essere rappresentato in una base qualsiasi con un numero infinito di cifre della parte frazionaria. Solo alcuni numeri reali possono essere rappresentati in modo finitario, cioè con un numero finito di cifre della parte frazionaria: tali numeri sono un sottoinsieme dei numeri razionali. Siamo abituati a considerare che tutti i razionali siano rappresentabili in modo finitario, utilizzando l'artificio della definizione del periodo: questo artificio richiede una particolare simbologia e non rientra nella notazione posizionale "pura".

Inoltre, dovremo necessariamente considerare solo rappresentazioni finite, con un numero di cifre limitato a priori, e quindi saremo in grado di rappresentare un sottoinsieme dei razionali ancora più ristretto; in pratica, saremo in grado di rappresentare solo approssimazioni dei numeri reali.

5.1 Virgola fissa

Nella rappresentazione in *virgola fissa* si usa una approssimazione del numero su $p+f$ cifre (p per la parte intera ed f per quella frazionaria). In tal modo il numero reale r è rappresentato su $p+f$ cifre dai coefficienti c_i tali che

$$r = \sum_{i=-f}^{-1} c_i B^i$$

Ad esempio la stringa binaria 00101011 per $p=5$ ed $f=3$ rappresenta il numero

$$r = 1 \cdot 2^2 + 1 \cdot 2^{-2} + 1 \cdot 2^{-3} = 4 + 1/4 + 1/8 = 5 + 3/8 = 5.375.$$

Viceversa si può ottenere la stringa binaria che rappresenta il numero 5.375 con l'algoritmo seguente (si tiene conto che $4 < 5.375 < 8$ e quindi servono almeno 3 cifre per la parte intera, mentre poniamo $f=3$) che fornisce le cifre della rappresentazione a partire dalla più significativa.

$c_2 = 5.375 \text{ div } 2^2 = 1$	$q_2 = (5.375 \text{ mod } 2^2) \cdot 2 = 2.750$
$c_1 = 2.750 \text{ div } 2^2 = 0$	$q_1 = (2.750 \text{ mod } 2^2) \cdot 2 = 5.5$
$c_0 = 5.5 \text{ div } 2^2 = 1$	$q_0 = (5.5 \text{ mod } 2^2) \cdot 2 = 3.0$
$c_{-1} = 3.0 \text{ div } 2^2 = 0$	$q_{-1} = (3.0 \text{ mod } 2^2) \cdot 2 = 6.0$
$c_{-2} = 6.0 \text{ div } 2^2 = 1$	$q_{-2} = (6.0 \text{ mod } 2^2) \cdot 2 = 4.0$
$c_{-3} = 4.0 \text{ div } 2^2 = 1$	$q_{-3} = (4.0 \text{ mod } 2^2) \cdot 2 = 0.0$

In questo caso le tre cifre frazionari sono sufficienti per rappresentare precisamente il numero. Questo metodo può essere impiegato anche per ottenere la rappresentazione dei numeri naturali.

Un algoritmo alternativo è il seguente:

- la rappresentazione della parte intera del numero viene calcolata come già visto nel paragrafo 2; quindi avrò i coefficienti da c_0 a c_2 ;

- la rappresentazione della parte frazionaria si ottiene a partire dalla cifra più significativa nel seguente modo (consideriamo ancora l'esempio precedente e calcoliamo la rappresentazione di 0.375)

$$c_{-1} = \text{trunc}(0.375 * 2) = 0 \quad q_{-1} = \text{fract}(0.375 * 2) = 0.750$$

$$c_{-2} = \text{trunc}(0.750 * 2) = 1 \quad q_{-2} = \text{fract}(0.750 * 2) = 0.5$$

$$c_{-3} = \text{trunc}(0.5 * 2) = 1 \quad q_{-3} = \text{fract}(0.5 * 2) = 0.0$$

L'operatore *fract* è un operatore che dato un numero del tipo *i.f* produce il risultato *0.f*, mentre l'operatore *trunc* produce come risultato *i* (notare che in questo algoritmo il risultato di *trunc* può essere solo 0 o 1); in altre parole $\text{fract}(x) = x - \text{trunc}(x)$.

Si noti che lo stesso algoritmi può essere applicato per una qualsiasi base, moltiplicando successivamente per la base, invece che per 2.

La rappresentazione dei reali appena mostrata è usata dagli elaboratori fissando a priori *p* ed *f* invece di introdurre un simbolo (.) per separare le due parti del numero. In pratica equivale a rappresentare numeri con un fattore implicito di scala di B^{-f} : infatti si può anche considerare che

$$101011 = (2^5 + 2^3 + 2^1 + 2^0) * 2^{-3}.$$

Anche nel caso dei reali si possono voler rappresentare numeri con segno; la tecnica di rappresentazione usata è quella di scegliere un partizionamento degli interi rappresentabili nel sistema di numerazione B^p a cui far corrispondere un sottoinsieme dei reali. Ad esempio se il numero reale 5.375 è rappresentato in base 2 su 8 cifre come 00101011, il numero -5.375 è rappresentato dalla sequenza 11010100 in complemento a 1 e dalla sequenza 11010101 in complemento a 2.

Applicando l'algoritmo visto per calcolare la rappresentazione in base 2 del numero 0.3_{10} , si vede che, qualsiasi numero di cifre frazionarie si consideri, non si arriva mai ad un valore *fract* pari a 0; cioè, il numero non è rappresentabile in modo finitario in base 2 (ma lo è in modo periodico). D'altra parte, se consideriamo il numero 0.12_3 , vediamo che la sua rappresentazione in base 10 sarà data da:

$1 * 1/3 + 2 * 1/9$, cioè $5/9 = 0.55555555...$, che non è rappresentabile in modo finitario in base 10, mentre lo è in base 3.

Attraverso semplici calcoli, si può vedere che in una base *B* qualsiasi posso rappresentare in modo

finitario solo quei numeri che sono ottenibili come una frazione il cui denominatore è una potenza dei fattori primi di B.

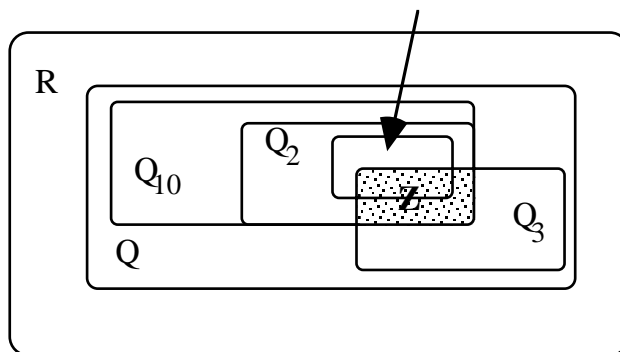
Se indichiamo con Q_B l'insieme di tali numeri, e con $FP(B)$ l'insieme dei fattori primi di B, ho che:

1) $Q_B \subseteq Q_C \Leftrightarrow FP(B) \subseteq FP(C)$

2) $Q_B \cap Q_C = Z \Leftrightarrow FP(B) \cap FP(C) = \Phi$ (cioè non ci sono numeri con parte frazionaria non nulla a comune tra i due insiemi Q_B e Q_C , se e solo se B e C sono primi tra loro), dove Z è l'insieme degli interi.

I numeri che siamo in grado di rappresentare in virgola fissa con p bit sono un sottoinsieme di Q_2 (si veda la seguente figura, dove R è l'insieme dei numeri reali).

numeri rappresentabili esattamente in virgola fissa in base 2 su m bit, con m fissato



5.2 Virgola mobile

La rappresentazione in virgola fissa non permette di agire sul fattore di scala per rendere più accurata l'approssimazione; si può pensare invece di rendere esplicito questo fattore introducendo una rappresentazione detta in *virgola mobile*, in base alla quale un numero reale r è visto come $m \cdot B^e$ e quindi può venire individuato tramite la coppia $\langle e, m \rangle$, dove m (la *mantissa*) è un numero reale in virgola fissa, mentre e (l'*esponente*) è un intero. Di queste coppie ne esistono più di una, ma, date due potenze successive della base, ne esiste solo una con la mantissa che cade tra queste due potenze. In generale il procedimento consiste nel trovare la rappresentazione binaria migliore del numero reale, come visto in precedenza, e poi introdurre un fattore di scala a seconda del tipo di risultato (si sceglie il troncamento migliore).

Al fine di presentare alcune considerazioni che valgono, in modo più o meno analogo, per tutte le rappresentazioni in virgola mobile, anche in base diversa da due, consideriamo la rappresentazione standard IEEE 754-1985 (in realtà consideriamo una semplificazione di questa rappresentazione standard). In questa rappresentazione si considerano solo numeri *normalizzati*, cioè in cui la parte intera è costituita da una sola cifra binaria uguale ad 1 (mantissa cade quindi tra 2^0 e 2^1). I numeri sono rappresentati da triple $\langle s, e, f \rangle$, dove:

- s codifica il segno;
- e rappresenta l'esponente;
- f rappresenta la parte frazionaria della mantissa (poiché la parte intera, essendo sempre uguale ad 1, non ha bisogno di essere rappresentata).

Il reale r rappresentato dalla tripla $\langle s, e, f \rangle$ è il numero

$$r = (1 - 2^s) \cdot (1 + f \cdot 2^{-m}) \cdot 2^{e-P}$$

Il primo fattore calcola il segno a partire da s ; il secondo fattore calcola la mantissa a partire da f e considerando il fattore di scala (f è rappresentato con m cifre); il terzo calcola l'esponente a partire da e . In quest'ultima espressione, P , che vale 2^{k-1} se e è rappresentato con k cifre, viene detto fattore di polarizzazione e permette di rappresentare anche esponenti negativi.

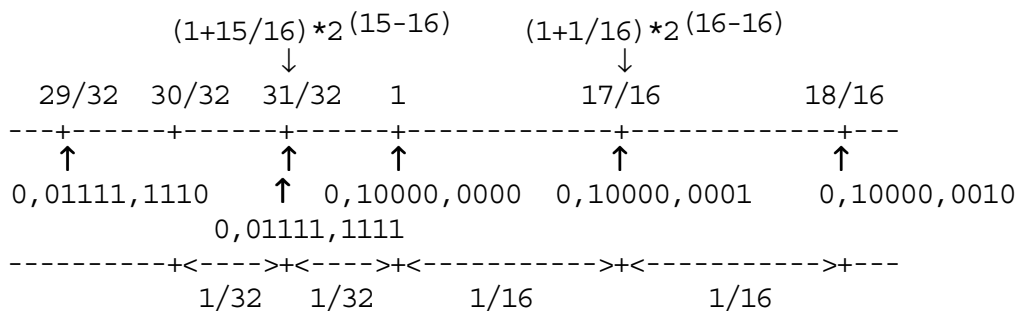
Lo standard IEEE- 754-1985 prevede due forme di rappresentazione:

- precisione semplice (o short real), con rappresentazioni su 32 cifre di cui $k=8$ e $m=23$;
- doppia precisione (o long real), con rappresentazione su 64 cifre di cui $k=11$ e $m=52$.

Consideriamo nel seguito tale rappresentazione, per semplicità, in configurazione ridotta, con $k=5$, $m=4$; avremo perciò $P=16$, i numeri rappresentati avranno esponente compreso tra -16 e $+15$ e mantissa compresa tra 1 e $1+15/16$ (cioè strettamente minore di 2). La rappresentazione dei numeri sarà di 10 cifre. Prendiamo l'insieme dei numeri rappresentabili in questo modo, raffigurati come punti sulla retta dei reali, in alcuni intervalli dove si presentano singolarità. Al crescere dei numeri

rappresentabili, al passaggio del numero 1 si nota un raddoppio della distanza tra due numeri rappresentabili successivi, e perciò un dimezzamento della precisione assoluta; in realtà, questo avviene al passaggio di ogni potenza di due. La precisione relativa è da considerare però costante, ed è data dal numero di cifre della mantissa. Intorno allo zero stanno i numeri rappresentabili minimi in valore assoluto. E' da notare come la distanza tra due valori positivi (o negativi) rappresentabili consecutivi in questo intorno costituisca la migliore precisione assoluta. Tra il minimo valore positivo rappresentabile e il massimo valore negativo rappresentabile la distanza non è pari a tale precisione, ma è molto maggiore; esiste quindi un "buco" intorno allo zero, cioè un intervallo in cui non ci sono numeri rappresentabili e la cui grandezza è molto maggiore della distanza tra valori rappresentabili

INTORNO di 1:

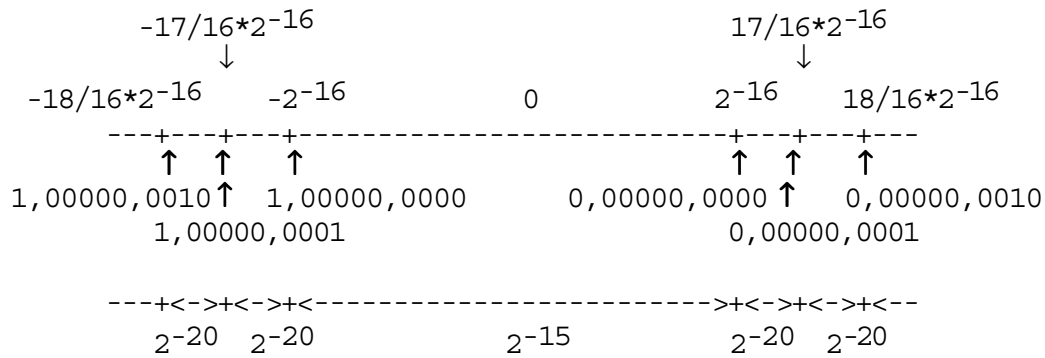


nell'intorno dello zero. Nella rappresentazione standard IEEE 754-1985, inoltre, lo zero stesso non viene rappresentato¹.

Per essere precisi, lo zero non è rappresentato nella rappresentazione semplificata che consideriamo, mentre nella rappresentazione standard vera e propria esiste un artificio che permette di rappresentarlo, come pure altri numeri appartenenti al suo intorno. L'artificio consiste nel restringere l'intervallo di rappresentabilità agli estremi (minimo e massimo numero rappresentato) in modo da liberare alcune configurazioni di bit che vengono usate per rappresentare tali numeri.

Il fenomeno del buco intorno allo zero è comunque comune a tutte le rappresentazioni in virgola mobile, anche quelle che ammettono una rappresentazione dello zero. Di ciò occorre tenere conto in particolari programmi in cui si eseguono calcoli numerici con una notevole accuratezza, che potrebbero avere comportamenti anomali derivanti dall'approssimarsi di certi valori allo zero.

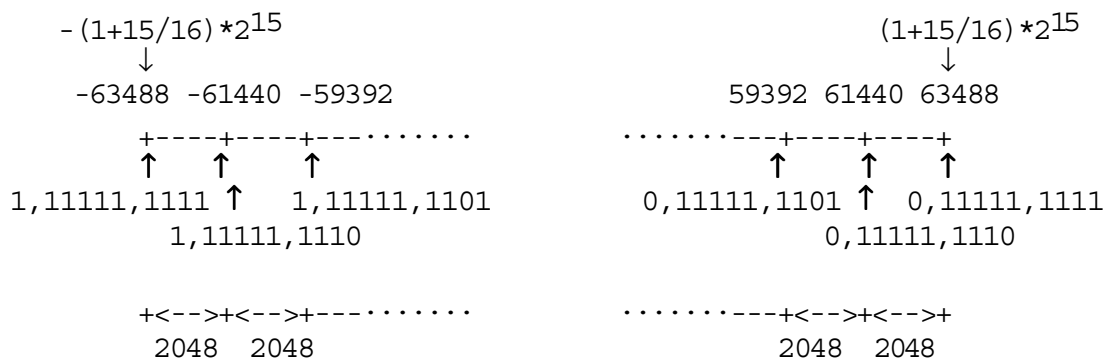
INTORNO di 0:



Da notare come la precisione assoluta sia molto bassa negli intervalli estremi della rappresentazione, come si vede dal grafico che segue.

Gli schemi presentati ci suggeriscono anche che la codifica secondo lo standard IEEE 754-1985 possiede una interessante proprietà: considerando le rappresentazioni dei reali positivi, il loro ordine, quando la tripla venga considerata nel suo complesso come la rappresentazione binaria unsigned di un unico valore, coincide con l'ordine dei numeri reali rappresentati. Ulteriori osservazioni sulla rappresentazione dei reali sono presentate in Appendice.

ESTREMI SUPERIORE ED INFERIORE DEI VALORI RAPPRESENTABILI



6. Rappresentazione di testi e figure.

Qualunque informazione si può vedere come un testo, cioè come sequenza anche illimitata di caratteri da interpretare (lo spazio è un carattere). Per ragioni di comprensione, nello scambio di informazioni tra elaboratori diversi, sono stati definiti dei codici standard per rappresentare qualunque carattere proveniente da una tastiera. I più diffusi sono il codice ASCII (American Standard Code for Information Interchange) e il codice EBCDIC (Extended Binary Coded Decimal Interchange Code); quest'ultimo è diventato uno standard di fatto perchè usato dall'IBM.

Nella tabella seguente è rappresentato il codice ASCII, che usa una stringa di 7 bit per carattere. Molto spesso i caratteri sono però rappresentati su 8 bit, dei quali il più significativo è sempre 0; sono molto diffusi anche codici ASCII "estesi", che usano 8 bit e danno significati particolari (ad es. caratteri "grafici") alle configurazioni di bit in cui il bit più significativo è 1.

Si sta inoltre diffondendo negli ultimi anni, in particolare in riferimento al linguaggio JAVA che lo adotta, il codice UNICODE che rappresenta ogni carattere con 16 bit, rendendo possibile anche la codifica di caratteri di altri alfabeti (greco, cirillico, arabo, ebraico, cinese, giapponese...).

Essendo l'input ad un elaboratore fornito premendo in sequenza i caratteri di una tastiera, anche un numero risulta rappresentato in ingresso come una sequenza di caratteri, ciascuno dei quali è una cifra rappresentata a sua volta da una sequenza di bit. Ad esempio, il numero naturale 125 è rappresentato come

0110001 0110010 0110101

nel codice ASCII. Tale sequenza deve poi venire convertita, calcolando il numero in base 2 corrispondente, in modo da poter eseguire le operazioni con gli algoritmi visti. Tali operazioni di conversione hanno un peso notevole (considerando anche che è necessaria una riconversione prima della stampa dei risultati) ed è quindi possibile anche pensare di definire algoritmi che eseguono le operazioni sulle rappresentazioni binarie cifra per cifra del numero (si parla in questo caso di rappresentazioni indirette dei numeri). Questi algoritmi sono in generale di complessità maggiore di quelli visti finora e vengono quindi impiegati in situazioni in cui si hanno grosse moli di dati con poca quantità di calcolo da eseguire; risulta allora conveniente evitare la doppia conversione per ognuno di questi dati a costo di una complessità maggiore del calcolo.

Per quanto riguarda le figure, si procede ad una loro linerizzazione in questo modo:

- si fissa un insieme di colori (al limite solo due: bianco e nero);
- ogni colore viene codificato con una sequenza di cifre (0 o 1 per il bianco e nero);
- si divide il piano della figura in tanti quadrati elementari detti *pixel*;

- ogni pixel viene associato a un colore.

Quindi si ha un campionamento della figura (un punto per ogni pixel) che è tanto più accurato quanto più grande è il numero di pixel usato; la sequenza di codici associati ai colori dei vari pixel costituisce la rappresentazione lineare della figura.

0	NUL	32		64	@	96	`
1	SOH	33	!	65	A	97	a
2	STX	34	"	66	B	98	b
3	ETX	35	#	67	C	99	c
4	EQT	36	\$	68	D	100	d
5	ENQ	37	%	69	E	101	e
6	ACK	38	&	70	F	102	f
7	BEL	39	'	71	G	103	g
8	BS	40	(72	H	104	h
9	HT	41)	73	I	105	i
10	LF	42	*	74	J	106	j
11	VF	43	+	75	K	107	k
12	FF	44	,	76	L	108	l
13	CR	45	-	77	M	109	m
14	SO	46	.	78	N	110	n
15	SI	47	/	79	O	111	o
16	DLE	48	0	80	P	112	p
17	XON	49	1	81	Q	113	q
18	DC2	50	2	82	R	114	r
19	XOFF	51	3	83	S	115	s
20	DC4	52	4	84	T	116	t
21	NAK	53	5	85	U	117	u
22	SYN	54	6	86	V	118	v
23	ETB	55	7	87	W	119	w
24	CAN	56	8	88	X	120	x
25	EM	57	9	89	Y	121	y
26	SUB	58	:	90	Z	122	z
27	ESC	59	;	91	[123	{
28	FS	60	<	92	\	124	
29	GS	61	=	93]	125	}
30	RS	62	>	94	^	126	~
31	US	63	?	95	_	127	DEL

Codice ASCII