La Rappresentazione dei numeri e le operazioni aritmetiche

Sommario

Introduzione

Pressochè tutte le applicazioni, per realizzare le loro elaborazioni, utilizzano calcoli numerici, indici, contatori, ecc.
Così come qualsiasi altro tipo di dato, anche i numeri, per essere immagazzinati nella memoria di un calcolatore, devono essere tradotti in sequenze di bit .
Questa operazione è a tutti gli effetti analoga ad una generica operazione di codifica.
In particolare:
$X = r(x)$
Dove X e' la rappresentazione, r e' la legge di codifica, x è il numero.

Strategia di codifica

Tipicamente, si adotta una codifica a lunghezza fissa. Il numero di bit a vari a a seconda della cardinalità dell'insieme dei numeri che si desidera rappresentare. *Nella pratica resta comunque pari ad un multiplo di 8 bit (tipicamente 8,16,24,32).*
L'associazione di un numero alla parola codice viene:

- ✓ Realizzata differentemente a seconda della tipologia di numeri che si desidera rappresentare
  - Naturali, relativi, razionali, ecc..
- ✓ Influenzata da aspetti che mirano a preservare la facile manipolazione delle rappresentazioni da parte del calcolatore
  - Operazioni aritmetiche, confronti logici, ecc..

Overflow e Underflow

Overflow – Definizione

Sia la dimensione che il numero dei registri in un calcolatore sono finiti. La cardinalità degli insiemi numerici che si è rappresentato è, invece, infinita.
È inevitabile dunque che in un insieme di cardinalità infinita solo un sotto-insime finito di elementi possa essere rappresentato. Gli operatori aritmetici, pur essendo talvolta chiusi rispetto all'intero insieme, quasi certamente non lo sono rispetto al sotto-insieme di cardinalità finita.
Quando accade che per effetto di operazioni, si tenta di rappresentare un numero non contenuto nel sotto-insieme si parla di overflow.

Overflow – Esempio

Si assuma di rappresentare i numeri naturali mediante un'operazione di codifica sul sottoinsieme da 0 a 127 (7 bit).

- La somma 100+100 genera overflow, essendo il numero 200 non rappresentabile mel sottoinsieme.
- La differenza 5-10 genera un overflow essendo il numero -5 non rappresentabile nel sottoinsieme.

Errore di approssimazione

Nel caso della rappresentazione di numeri razionali sussiste il problema dell'overflow, ma subentra anche il problema dei numeri razionali di costruire un insieme denso.
A causa di ciò, anche se si sceglie per la rappresentazione un intervalli limitato, non tutti i numeri nell'interno di esso potranno essere rappresentati.
Quando si tenta di rappresentare un numero per cui non è stata prevista una rappresentazione, spesso si associa a tale numero la rappresentazione del numero "più vicino", cioè quello che lo approssima meglio.
In questo caso si parla dei errore di approssimazione.

Underflow

Quando in un'elaborazione si tenta di rappresentare un numero "troppo vicino" allo zero, l'errore di approssimazione può fa si che la rappresentazione scelta sia quello dello zero.
Questa evenienza può condizionare pesantemente i calcoli successivi nel seguito dell'elaborazione a causa delle peculiarità dello zero (che, ad esempio, non può essere utilizzata al denominatore di una frazione).
Si parla allora di underflow.
Questa problematica e' molto sentita nell'ambito del calcolo numerico, specialmente nella applicazioni che, per loro natura, tendono a lavorare con quantità molto piccole (per esempio applicazioni per il calcolo differenziale).

Rappresentazione dei Numeri Naturali

Si voglia rappresentare attraverso una sequenza di bit di lunghezza costante pari ad n l'insieme dei numeri naturali:

- Il numero degli elementi rappresentabili è pari a 2^n.
- Tipicamente, volendo rappresentare sempre anche lo zero si rappresentano i numeri compresi tra 0 e $2^n - 1$.

L'associazione tra ogni numero e la propria rappresentazione avviene, nei casi pratici, nella maniera più intuitiva:

- Ad ogni numeri viene associata la stringa di bit che lo rappresenta in un sistema di numerazione binario posizionale.

L'overflow avviene quando si tenta di rappresentare un numero esterno all'intervallo [0 , $2^n - 1$].

Esempio

Rappresentazione dei numeri naturali su 4 bit.

n=4
Intervallo: [0,15]
Codifica: X=x

Operazione sui numeri naturali su 4 bit

Per realizzare le operazioni, il calcolatore può lavorare direttamente sulle rappresentazioni. La correttezza dei calcoli è garantita dalle leggi dell'aritmetica binaria posizionale (analoghe a quelle della classica aritmetica decimale).
L'overflow può essere facilmente rilevato attraverso la valutazione del riporto (o del prestito) sull'ultima cifra.

```
6+          0110+
8=  ──────► 1000=
----        ------
14          1110

14+         1110+
 3=──────►  0011=
---         ----------
17         10001r
           ↗
        Overflow
```

Rappresentazione dei Numeri Relativi

Rappresentazione in Segno e Modulo

Mentre nel caso dei numeri naturali l'associazione di un numero con una stringa di bit è alquanto intuitiva, nel caso dei numeri relativi si pone il problema di associare una rappresentazione a ciascun numero negativo.
Una possibilità è:

- Associare un singolo bit (per esempio quello più significativo) al segno
- Utilizzare i restanti bit per rappresentare il modulo (che è un numero naturale)

Questo tipo di codifica va sotto il nome di "codifica segno e modulo".

$$s=sign(x)=\begin{cases}1 \text{ per } x<0\\ 0 \text{ per } x>=0\end{cases}$$

Se si utilizzano n bit:

- La legge di codifica X = r(x) è: $X = |x| + 2^{n-1} * sign(x)$.
- Si possono rappresentare numeri relativi compresi nell'intervallo $[-(2^{n-1} - 1); 2^{n-1} - 1]$.
- In totale i numero rappresentati sono $2^n - 1$.
    - Ciò dipende dal fatto che, a causa della natura di questa codifica, lo zero ammette doppia rappresentazione dette: *zero positivo* e *zero negativo*.

Esempio

Rappresentazione in segno e modulo su 4 bit

n=4
Intervallo: [-7;7]
Codifica:
X=|x|+8 * sign(x)

Operazioni in segno e modulo su 4 bit

Diversamente dalla rappresentazione dei numeri naturali, questa volta non è possibile lavorare direttamente sulla rappresentazione dei numeri per realizzare le operazioni aritmetiche. È necessario, invece, lavorare separatamente sul segno e sul modulo.
Quando ad esempio si sommano due numeri di segno discorde, bisogna determinare quello con modulo maggiore e sottrarre ad esso il modulo dell'altro. Il segno risulterà quello dell'addendo maggiore in modulo.
Tale caratteristica, insieme con il problema della doppia rappresentazione dello zero, rende i calcoli particolarmente laboriosi e, per questo motivo, non è molto utilizzata nella pratica.

Rappresentazione in complementi alla base

Una seconda tecnica per la rappresentazione dei numeri relativi consiste nell'associare a ciascun numero il suo **resto Modulo M=2^n,** definito come:

$$|x|_M = x - [x/M] * M$$

Questo tipo di codifica, su n bit, è equivalente ad associare:

- Il numero stesso (cioè X = x), ai numeri positivi compresi tra 0 e $2^{n-1} - 1$;
- I numeri rappresentati sono quelli compresi nell'intervallo:
  $[-2^{n-1};2^{n-1}-1]$
- In altri termini, un insieme dei numeri interi relativi x è rappresentato de un insieme di numeri naturali X, ciascuno dei quali è il resto modulo-M di x.

Esempio

Rappresentare in complementi alla base su 4 bit

n=4
Intervallo: [-8 ; 7]
Codifica:
X = x; 0<= x<= 7
X = 2^n - |x|; -8<=x<=-1

Complementi alla base: proprietà

Questa rappresentazione ha il fondamentale vantaggio di permettere, nell'ambito di operazioni aritmetiche, di **lavorare direttamente sulle rappresentazioni**.
La regola sulla quale questa affermazione si basa è la seguente:

la rappresentazione della somma (algebrica) di x e y si ottiene come la somma (modulo-M) delle rappresentazioni di x e y; analoghe sono le proprietà della differenza del prodotto.

Questo tipo di codifica conserva, inoltre, la proprietà delle rappresentazioni di avere il primo bit alto se (e solo se) il corrispondente numero è negativo.

Complementi alla base: la complementazione

In complementi alla base, a partire dalla rappresentazione di un numero, è anche **particolarmente semplice ottenere la rappresentazione del suo opposto**.
È infatti sufficiente ***complementare tutti i bit a partire da sinistra, tranne l'uno più a destra ed eventuali zero successivi.***
Questa ulteriore caratteristica consente di **realizzare le sottrazioni attraverso la composizione di una complementazione (nel senso suddetto) ed un'addizione**.
Nell'aritmetica in complementi alla base, di conseguenza, l'addizione e il complementare rappresentano i componenti fondamentali per la realizzazione di tutte le operazioni.

Rappresentazione in complementi diminuiti

La rappresentazione in complementi diminuiti costituisce un'ulteriore alternativa per la codifica dei numeri relativi. Concettualmente è analoga alla rappresentazione in complementi alla base.
La differenza rispetto ad essa è che la legge di codifica dei numeri negativi è leggermente differente:

- $X = 2^n - |x|;$ (complementi alla base)
- $X = 2^n - 1 - |x|;$ (complementi alla base diminuiti)

I numeri rappresentabili, se si utilizzano n bit, sono quelli compresi nell'intervallo $[-(2^{n-1}-1); 2^{n-1}-1]$.
Anche in questo caso, quindi, le cifre rappresentabili sono 2^n-1 e ancora una volta è lo zero ad avere una doppia rappresentazione.

Esempio

Rappresentazione in complementi alla base diminuiti su 4 bit

n = 4
Intervallo: [-7 ; 7]
Codifica:
$X = x;$ $0<=x<=7$
$X = 2^n - 1 - |x|;$ $-7<=x<=-1$

Complementi diminuiti: perché?

Maggiora la semplicità cin cui è possibile calcolare la rappresentazione dell'opposto di un numero, a partire dalla rappresentazione del numero stesso: ***basta semplicemente complementare tutti i bit della rappresentazione indistintamente.***
Esempio:
La rappresentazione in complementi alla base diminuiti su 4 bit di 4 è 0100;
complementando tutti i bit si ottiene 1011;
1011 è la rappresentazione in complementi alla base diminuiti su 4 bit di -4.

La rappresentazione in complementi alla base diminuiti su 4 bit di -6 è 1001;
complimentando tutti i bit si ottiene 0110;
0110 è la rappresentazione in complementi diminuiti su 4 bit di 6.

Rappresentazione Eccesso-K

La rappresentazione in eccesso-k costituisce un metodo diverso da quello dei resti in modulo per ricondurre i numeri negativi a positivi.
IN particolare tutti i numeri sono traslati verso l'alto di k, che viene scelto maggiore o uguale al numero più piccolo da rappresentare: $X = r(x) = x+k$. È anche detta rappresentazione *biased* (polarizzata) e k è detto *bias* (costante di polarizzazione).

Rappresentazione eccesso-k - Proprietà

Analogamente al caso dei complementi diminuiti, la somma va corretta aggiungendo o sottraendo la costante k, e quindi in maniera sufficientemente semplice. Le moltiplicazioni e le divisioni risultano invece più complesse. Il vantaggio di tale codifica è che viene conservata la proprietà della disuguaglianza sulle rappresentazioni: $x_1 > x_2$ se e solo se $X_1 > X_2$. Questa rappresentazione perciò è utilizzata solamente laddove siano richieste fondamentalmente somme algebriche e confronti logici fra gli operandi. Tipicamente si utilizza per rappresentare gli esponenti nella rappresentazione in virgola mobile.

Esempio

Rappresentazione in eccessp-8 su 4 bit

n=4
Intervallo: [-8;7]
Codifica: X=x+k;

Rappresentazione dei numeri frazionari

Numeri reali in virgola fissa

Quando di un numero frazionario si rappresentano separatamente la parte intera e la parte frazionaria si parla di rappresentazione in *virgola fissa (fixed point).*
La rappresentazione dei due contributi (che sono numeri interi) può essere realizzata secondo una delle tecniche viste in precedenza. In questo caso la posizione della virgola è fissa e resta sottintesa.

Numeri reali in virgola mobile

Il numero reale x può essere rappresentato dalla coppia (m,e) tale che: $x = m*b^e$ dove:

- *m* è detta *mantissa*;
- *e* è detto *esponente*;
- *b* è la *base di numerazione adottata*.

Il metodo in questo caso prende il nome di *codifica in virgola mobile (floating point).*

Normalizzazione

Per ciascun numero esistono infinite coppie che lo rappresentano.
Esempio (b=10):
346.09801 è rappresentato da
» (346.09801, 0) oppure
» (346098.01, -3) oppure
» (0.034609801, 4) etc…

Allo scopo di uniformare le rappresentazioni, si fissa convenzionalmente la posizione della virgola subito dopo la prima cifra significativa, ottenendo così l'unico rappresentante: (m,e) = (3.4609801, 2).

Esempio: intervallo di rappresentazione

Con b=10, usando 4 cifre per m e 2 per e (più due bit per i relativi segni), l'insieme rappresentabile (usando solo rappresentazioni normalizzate) è:
$[-9.999\times1099,-1.000\times10\text{-}99]\cup\{0\}\cup[+1.000\times10\text{-}99,+9.999\times1099]$

Approssimazione

Come è facile verificare, in questo tipo di rappresentazione l'approssimazione non è costante. In particolare la precisione assoluta è molto spinta in prossimità dello zero e va diminuendo progressivamente a mano a mano che il numero aumenta (in valore assoluto).
Si commettono quindi "errori piccoli" su "numeri piccoli" ed "errori grandi" su "numeri grandi".
Quello che resta inalterato è invece l'errore relativo, costante su tutto l'asse di rappresentabilità.

Overflow e Underflow

L'errore relativo dipende dal numero di cifre della mantissa. Gli estremi dell'intervallo di rappresentazione dipendono dal numero di cifre dell'esponente.
Nel caso precedente di 2 cifre per l esponente, si ha overflow per numeri maggiori (in modulo) di 10^{99} e si ha underflow per numeri minori (in modulo) di 10^{-99}.

Rappresentazione in macchina di interi

La classe dei numeri interi in numerazione binaria è quella fondamentale dei sistemi numerici:

- *Unsigned,* corrispondente alla classe dei numeri naturali;
- *Signed,* corrispondenti alla rappresentazione in complementi

Data una lunghezza di riferimento (definita dal parallelismo del processore oppure dalla storia della produzione della specifica architettura), si adoperano classi di numeri in singola, doppia, quadrupla e in qualche caso ottupla lunghezza.

Il riferimento per l'indirizzamento in memoria di un dato è legato alla lunghezza di riferimento (è ad esempio il byte o la parola di 32 bit) e si pone allora il problema di definire come i componenti successivi del dato si susseguono in memoria:

- dalla parte più significativa a quella meno significativa o viceversa. In merito esistono e sono entrambi adoperati, due distinti ordinamenti: diretto o big endian e inverso o little endian.

Diretto o big endian (a): l'indirizzo del dato è quello della componente più significativa, le componenti meno significative sono memorizzate ad indirizzi crescenti.

Inverso o little endian (b): l'indirizzo del dato è quello della componente meno significativa, il dato poi si sviluppa in memoria "alla rovescia", con le parti più significative ad indirizzi crescenti.

La soluzione diretta è quella naturale, quella inversa si adotta per semplificare la realizzazione della aritmetica, che elabora le cifre a partire da quella di minor peso.