Corso di Informatica

Modulo 1

2.1-Rappresentazione dei numeri naturali

Prerequisiti

- Aritmetica elementare
- Numeri naturali

Introduzione

In questa Unità iniziamo a vedere come nel computer sono rappresentati i numeri che si imparano per primi: i **numeri naturali**.

Dei numeri naturali trattiamo le **operazioni** e le rispettive **proprietà**.

Successivamente, trattiamo **l'aritmetica modulare**, che ci obbliga a tenere conto del fatto che per rappresentare le informazioni nel computer, abbiamo a disposizione uno spazio limitato; si descrive, di conseguenza, il fenomeno dell'**overflow**.

Successivamente, si trattano le conversioni di base, illustrando le conversioni da base 2, 8 e 16 a base 10 e viceversa, i casi particolari di conversione da base 2 a base 8 e viceversa e da base 2 a base 16 e viceversa e si dà un cenno al codice BCD.

I numeri naturali

L'**insieme N dei numeri naturali** è formato da tutti i numeri interi a partire da 0.

N = {0, 1, 2,...}

È un **insieme infinito**, sul quale abbiamo:

- **operazioni dirette**:
  - **addizione**
  - **moltiplicazione**
  - **elevamento a potenza**
- **relazioni**
- **M.C.D**
- **m.c.m.**

Dato un naturale x, esiste:

- il **successivo** (x+1)
- il **precedente** (x-1) (esiste solo se x è diverso da 0

Operazioni sui numeri naturali

- **Proprietà dell'addizione**:
  - COMMUTATIVA: a + b = b + a;
  - ASSOCIATIVA: (a + b) + c = a + (b + c)
  - ESISTENZA ELEMENTO NEUTRO: a + 0 = 0 + a = a
- **Proprietà della moltiplicazione:**
  - COMMUTATIVA: a * b = b * a;
  - ASSOCIATIVA: (a * b) * c = a * (b * c)
  - ESISTENZA ELEMENTO NEUTRO: a * 1 = 1 * a = a
  - DISTRIBUTIVA RISPETTO ALLA SOMMA: a * (b + c) = a * b + a * c
- **Proprietà dell'elevamento a potenza**:
  - PRODOTTO DI POTENZE CON LA STESSA BASE: $a^b * a^c = a^{b+c}$;
  - POTENZA DI POTENZA: $(a^b)^c = a^{b*c}$
  - QUOZIENTE DI POTENZE CON LA STESSA BASE: $a^b / a^c = a^{b-c}$ (b>c)

Aritmetica modulare

In un elaboratore le informazioni (numeriche e non numeriche), sono rappresentate da sequenze di 0 e 1 di **lunghezza finita e fissa** (8, 16, 32 bit), dipendente dal particolare tipo di computer.

Questo può creare problemi poiché nelle operazioni dirette (addizione e moltiplicazione) c'è il rischio che il risultato non possa essere rappresentato.

Ad esempio, in un sistema a 8 bit (per semplicità), il risultato di un calcolo che fosse maggiore di 255 (che vale 11111111_2), non potrebbe essere rappresentato, con conseguenze negative in tutta la successiva elaborazione.

Aritmetica modulare

La parte della matematica che si occupa dei calcoli in casi in cui si ha un limite superiore ai valori, si dice **aritmetica modulare** (Gauss).

In questo tipo di aritmetica, detta anche aritmetica dell'orologio, nelle operazioni occorre immaginare che i numeri "si avvolgano su se stessi": ogni volta che raggiungono i multipli di un determinato numero *n*, detto **modulo**, il numero riparte da 0.

Come conseguenza, la rappresentazione di un numero che va oltre il valore di *n* dà luogo ad una situazione di errore detta **overflow** (ossia *traboccamento*).

Aritmetica modulare

Ad esempio, se consideriamo un'aritmetica su 3 bit, si ha:

0 = 000
1 = 001
2 = 010
3 = 011
4 = 100
5 = 101
6 = 110
7 = 111
8 = (1) 000 ???

Si rappresentano solo le sequenze di binarie di 3 cifre: 000, 001, ..., 111)

Esempio di aritmetica modulare **modulo 2^3**.

Il numero 8 richiede un quarto bit per essere rappresentato correttamente.

Overflow

Questo impone limiti alla rappresentazione dei numeri all'interno del computer.

Conversioni numeriche

La trasformazione di un numero nel suo equivalente in un altro sistema di numerazione si dice **conversione**.

Per effettuare una conversione da una base **b** ad un'altra **B**, occorre conoscere le regole per manipolare i numeri.

In particolare:

- in ogni conversione intervengono **regole dell'aritmetica** di **b** e di **B**;
- è sufficiente conoscere le regole in una sola delle due basi, **b** o **B**.

Conversioni numeriche

Vediamo le conversioni:

- binario, ottale, esadecimale → decimale
- decimale → binario (o ottale o esadecimale)

La conversione da una base qualsiasi (binario, ottale e esadecimale) a quella decimale si effettua applicando direttamente la definizione di notazione posizionale, poiché conosciamo l'aritmetica della base di arrivo (10).

Conversioni numeriche

Se si conosce l'aritmetica della base **B** di arrivo, ma non si conosce quella della base **b** di partenza, conversione può essere ottenuta utilizzando l'aritmetica **B**.

La conversione da una base qualsiasi (binario, ottale e esadecimale) a quella decimale si effettua applicando direttamente la definizione di notazione posizionale, poiché conosciamo l'aritmetica della base di arrivo (10).

Conversioni numeriche
Da base 2, 8, 16 a base 10

Esempi:

- Da base 2 a base 10: $1101_2=1\cdot2^3+1\cdot2^2+0\cdot2^1+1\cdot2^0=13_{10}$
- Da base 8 a base 10: $1101_8=1\cdot8^3+1\cdot8^2+0\cdot8^1+1\cdot8^0=577_{10}$
- Da base 10 a base 10: $1101_{10}=1\cdot10^3+1\cdot10^2+0\cdot10^1+1\cdot10^0=1101_{10}$
- Da base 16 a base 10: $1101_{16}=1\cdot16^3+1\cdot16^2+0\cdot16^1+1\cdot16^0=4353_{10}$

Pertanto, la conversione da base **b** (2, 8, 16) a base **B** (10), è possibile perchè sappiamo svolgere i calcoli nella base **B**.

Conversioni numeriche
Da base 2, 8, 16 a base 10

Regola pratica per conversione binario → decimale:

- si raddoppia il bit più significativo (**MSB**, **M**ost **S**ignificant **B**it) e si aggiunge il successivo bit
- si raddoppia la somma e si aggiunge il successivo bit
- si procede raddoppiando le somme e aggiungendo i bit successivi fino al bit meno significativo (**LSB**, **L**east **S**ignificant **B**it); l'ultima somma rappresenta il numero decimale

Conversioni numeriche
Da base 2, 8, 16 a base 10

Esempio: $100101_2 = ?_{10}$

Regola con il sistema posizionale:
$100101_2 = 1 * 32 + 0 * 16 + 0 * 8 + 1 * 4 + 0 * 2 + 1 * 1 = 37_{10}$

Regola pratica:
1 * 2 + **0** = 2; 2 * 2 + **0** = 4; 4 * 2 + **1** = 9; 9 * 2 + **0** =18; 18 * 2 + **1** = 37_{10};

In pratica si fa solo una moltiplicazione per due in binario (che corrisponde ad uno shift a sinistra) e una somma, al posto di moltiplicazione delle cifre per le potenze di 2.

Conversioni numeriche
Da base 10 a base 2, 8, 16

Se non si conosce l'aritmetica della base **B** di arrivo, ma si conosce quella della base **b** di partenza, la conversione può essere ottenuta utilizzando l'aritmetica **b**.

La conversione in una base qualsiasi (binario, ottale e esadecimale) da quella decimale si effettua usando l'aritmetica della base di partenza (10).

Conversioni numeriche
Da base 10 a base 2, 8, 16

Regola delle divisioni successive:

Il numero da convertire viene diviso (con divisione intera!!) per la base di arrivo B.

Si ripete la divisione per tutti i quozienti via via ottenuti, terminando il procedimento quando si ottiene quoziente uguale a zero.

I resti ottenuti, convertiti nel sistema di arrivo, costituiscono le cifre della numero convertito.

La sequenza dei resti va presa in ordine inverso: l'ultimo resto è la cifra più significativa, mentre il primo resto ottenuto è la cifra meno significativa).

Conversioni numeriche

Da base 10 a base 2, 8, 16

Esempio: $123_{10} = ?_2$

123_{10} : 2 = 61 (r = 1)
61_{10} : 2 = 30 (r = 1)
30_{10} : 2 = 15 (r = 0)
15_{10} : 2 = 7 (r = 1)
7_{10} : 2 = 3 (r = 1)
3_{10} : 2 = 1 (r = 1)
1_{10} : 2 = 0 (r = 1)

$123_{10} = 1111011_2$

Esempio: $123_{10} = ?_8$

123_{10} : 8 = 15 (r = 3)
15_{10} : 8 = 1 (r = 7)
1_{10} : 8 = 0 (r = 1)

$123_{10} = 173_8$

Casi particolari di conversioni

La conversione da una base **b** ad una base **B** risulta semplice quando:

$b = B^k$ Es. $8 = 2^3$, $16 = 2^4$

$b^{k'} = B$ Es. $2^3 = 8$, $2^4 = 16$

In questo caso, se occorre convertire un numero da base **b** a base $\mathbf{B^k}$, ogni cifra in base **b** va convertita in una sequenza di *k* cifre in base **B**, secondo una tabella di conversione costituita da b elementi, come vediamo tra breve.

Casi particolari di conversioni
Da base 8 a base 2

Esempio: ottale → binario

b = 8, B = 2 → k = 3

Ad ogni cifra in base **b** (8)corrispondono 3 cifre in base **B** (2).

Es.: $25_8 = 010101_2$

$\boxed{010}$ $\boxed{101}$

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

Casi particolari di conversioni
Da base 2 a base 8

Esempio: binario → ottale

b = 2, B = 8 → k' = 3

Ogni 3 cifre in base **b** (2) corrisponde 1 cifra in base **B** (8).

Es.: $01101_2 = 001\ 101 = 15_8$

$\boxed{1}$ $\boxed{5}$

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

Regola pratica:
- si suddivide il numero binario dato in gruppi di tre bit partendo da destra;
- si completa con zeri l'eventuale gruppo di sinistra con meno di tre bit
- si associa a ciascun gruppo il corrispondente simbolo ottale

Casi particolari di conversioni

Da base 16 a base 2

Esempio: esadecimale → binario

b = 16, B = 2 → k = 4

Ad ogni cifra in base **b** (16)corrispondono 4 cifre in base **B** (2).

Es.: $2F_{16} = 00101111_2$

0010 1111

0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
A	1010
B	1011
C	1100
D	1101
E	1110
F	1111

Casi particolari di conversioni

Da base 2 a base 16

Esempio: binario → esadecimale

b = 2, B = 16 → k' = 4

Ogni 4 cifre in base **b** (2) corrisponde 1 cifra in base **B** (16).

Es.: $011101_2 = 0001\ 1101\ = 15_8$

1 5

Regola pratica:
- si suddivide il numero binario dato in gruppi di quattro bit partendo da destra;
- si completa con zeri l'eventuale gruppo di sinistra con meno di quattro bit
- si associa a ciascun gruppo il corrispondente simbolo esadecimale (v. tabella)

Rappresentazione BCD

In questo tipo di notazione ciascuna cifra del numero è rappresentata da un gruppo di quattro bit.

Esempio:

4510 = → 0100 0101 0001 0000

Vantaggi: non introduce errori, spesso usata per scambi valutari.

Svantaggi: spreco di bit (delle 16 configurazioni disponibili, se ne usano solo 10)

Rappresentazione BCD

Quando nella programmazione ad alto livello si fa riferimento a rappresentazioni decimali, il computer effettua sempre una conversione in binario.

La rappresentazione BCD semplifica in parte la conversione poiché ciascuna cifra decimale è rappresentata dal suo equivalente binario.

In realtà alcune considerazioni pratiche (spreco di bit, difficoltà ad effettuare operazioni algebriche) fanno preferire altri tipi di rappresentazione negli attuali μp.

Argomenti

- I numeri naturali
- Operazioni sui numeri naturali
- Aritmetica modulare
- Conversioni numeriche
  - da base 2, 8, 16 a base 10
  - da base 10 a base 2, 8, 16
- Casi particolari di conversioni
  - da base 8 a base 2
  - da base 2 a base 8
  - da base 16 a base 2
  - da base 2 a base 16
- Rappresentazione BCD

Altre fonti di informazione

- P.Gallo, F.Salerno – Informatica Generale 1, ed. Minerva Italica
- G.Callegarin – Corso di Informatica 1, ed. CEDAM