Rappresentazione dell’Informazione

NUMERICA

Rappresentazione degli Interi

$N = 0, +1, -1, +2, -2, +3, -3, \ldots$

Come possiamo rappresentare il *segno* di un numero?

Aggiungiamo un ulteriore bit che poniamo a 1 se il numero è negativo!

Con k bit si possono codificare tutti gli interi

$$-2^{k-1}+1 \leq N \leq 2^{k-1}-1$$

Esistono due codifiche dello 0

0000…0 1000…0

Esempio

$N_{10} = +14$ $N_2 = 01110$

$N_{10} = -14$ $N_2 = 11110$

Rappresentazione degli Interi

(Rappresentazione in complemento a 2)

Supponiamo di avere a disposizione k bit

La rappresentazione di -N si ottiene facendo la conversione in binario del numero 2^k-N

Con k bit si possono codificare tutti gli interi

$$-2^{k-1} \leq N \leq 2^{k-1}-1$$

$0000...0_2 \equiv 0_{10}$

$1000...0_2 \equiv -2^k{}_{10}$

Esempio (con 5 bit)

$N_{10} = +14$ $\quad N_2 = 01110$

$N_{10} = -14$ $\quad 2^k-14 = 18$ $\quad N_2 = 10010$

Rappresentazione di interi in complemento a due

Supponiamo di avere a disposizione una cella di otto bit compreso il bit di segno. Il bit di segno può assumere due soli simboli: **0** quando il numero è **positivo** e **1** quando il numero è **negativo**.

Regola del complemento a due?

A partire da destra verso sinistra il primo uno che incontriamo non si complementa invece tutti gli altri bit si complementano.

Premettendo che su un byte il più piccolo numero rappresentabile è –128, e il più grande è + 127.

Rappresentiamo prima +15:

(0)	0	0	0	1	1	1	1

Rappresentiamo adesso –15 con la tecnica del complemento a 2:

(1)	1	1	1	0	0	0	1

Rappresentazione di interi in complemento a uno

Supponiamo di avere a disposizione una cella di otto bit compreso il bit di segno. Il bit di segno può assumere due soli simboli: **0** quando il numero è **positivo** e **1** quando il numero è **negativo**.

Regola del complemento a uno?

Si complementano tutti i bit, lo 0 diventa 1 e1 diventa 0.

In questo caso con un byte il più piccolo numero rappresentabile è -127, e il più grande è + 127 otteniamo quindi una rappresentazione in meno

Rappresentiamo prima +15:

(0)	0	0	0	1	1	1	1

Rappresentiamo adesso -15 con la tecnica del complemento a 2:

(1)	1	1	1	0	0	0	0

VANTAGGI/SVANTAGGI

COMPLEMENTO A 1

- Facilità nella implementazione della complementazione
- Effettuare degli aggiustamenti nei risultati delle operazioni matematiche

COMPLEMENTO A 2

- Complessità nell’implementazione della complementazione
- Non bisogna fare nessuna correzione

Regola generale della somma di tre bit:

a_j	B_j	c_j	S_j	C_{j+1}
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

Esercizio n.1: Rappresentare in complemento a 1 e a 2 la somma di A=+100 e B=-47

Complemento a 1

	1	1	0	0	0	0	0	0	0
A=+100		(0)	1	1	0	0	1	0	0
B=-47		(1)	1	0	1	0	0	0	0
+53		(0)	0	1	1	0	1	0	0

Regola:
Se A+B>0 allora effettuare la correzione aggiungendo 1 al risultato

Complemento a 2

	1	1	0	0	0	0	0	0	0
A=+100		(0)	1	1	0	0	1	0	0
B=-47		(1)	1	0	1	0	0	0	1
+53		(0)	0	1	1	0	1	0	1

Esercizio n.2: Rappresentare in complemento a 1 e a 2 la somma di A=-100 e B=+47

	0	0	1	1	1	1	1	1	0
A=-100		(1)	0	0	1	1	0	1	1
B=+47		(0)	0	1	0	1	1	1	1
-53		(1)	1	0	0	1	0	1	0
+53		(0)	0	1	1	0	1	0	1

	0	0	1	1	1	1	0	0	0
A=-100		(1)	0	0	1	1	1	0	0
B=+47		(0)	0	1	0	1	1	1	1
-53		(1)	1	0	0	1	0	1	1
+53		(0)	0	1	1	0	1	0	1

OVERFLOW E UNDERFLOW

Esercizio n.3: Rappresentare in complemento a 1 e a 2 la somma di A=+100 e B=+28

Complemento a 1

	0	1	1	1	1	1	0	0	0
A=+100		(0)	1	1	0	0	1	0	0
B=+28		(0)	0	0	1	1	1	0	0
+128 ?		(1)	0	0	0	0	0	0	0
OVERFLOW									

Complemento a 2

	0	1	1	1	1	1	0	0	0
A=+100		(0)	1	1	0	0	1	0	0
B=+28		(0)	0	0	1	1	1	0	0
+128 ?		(1)	0	0	0	0	0	0	0
OVERFLOW									

Esercizio n.4: Rappresentare in complemento a 1 e a 2 la somma di A=-100 e B=-28

	1	0	0	0	0	0	1	1	0
A=-100		(1)	0	0	1	1	0	1	1
B=-28		(1)	1	1	0	0	0	1	1
-128 ?		(0)	1	1	1	1	1	1	0
UNDERFLOW									

	1	1	1	1	1	1	0	0	0
A=-100		(1)	0	0	1	1	1	0	0
B=-28		(1)	1	1	0	0	1	0	0
-128		(1)	0	0	0	0	0	0	0
é il più piccolo rappresentabile in C2, è l’unico che non ha il suo complemento									

Rappresentare numeri razionali

- Il numero di bit utilizzati per rappresentare i numeri in un computer è fissato
  - Con 8 bit si rappresentano i numeri da 0 a 255 (oppure da –128 a 127)

- Rappresentare i numeri razionali con la virgola in posizione fissa è un inutile spreco di bit (34.56 = 0000034.5600000000)
- La precisione varia a seconda del valore del numero

Rappresentazione in Virgola Mobile

- I numeri si devono esprimere utilizzando il maggior numero di cifre significative → virgola mobile
- I numeri si rappresentano in notazione esponenziale con mantissa ed esponente
- Si assegnano pochi bit all'esponente e il resto (m) alla mantissa
- La mantissa è compresa tra 0 e $1\text{-}2^{-m}$

Rappresentazione in virgola mobile

$$\overbrace{\boxed{0}}^{\text{Segno della Mantissa}}\ \underbrace{\boxed{00000000}}_{\text{Esponente}}\ \underbrace{\boxed{00000000000000000000000}}_{\text{Mantissa}}$$

virgola mobile: un esempio

2

Segno della Mantissa

Segno della Mantissa	Esponente	Mantissa
0	00000010	01100000000000000000000

$$1*2^{-2}+1*2^{-3}=0.25+0.125=0.375$$

virgola mobile: un esempio

2

Segno della
Mantissa

0	00000010	01100000000000000000000

Esponente

Mantissa

0.375

$0.375*2^2=1.5$