Aritmetica binaria

Moreno Marzolla

moreno.marzolla@pd.infn.it

Copyright © 2006 Moreno Marzolla

This work is licensed under the Creative Commons Attribution-Noncommercial-Share Alike 2.5 Italy License. To view a copy of this license, visit http://creativecommons.org/licenses/by-nc-sa/2.5/it/ or send a letter to Creative Commons, 171 Second Street, Suite 300, San Francisco, California, 94105, USA.

Ricordiamo...

- Il docente
  - Moreno Marzolla
  - Email: **moreno.marzolla@pd.infn.it**
  - Web: **http://www.dsi.unive.it/~marzolla**
  - Ufficio: Dipartimento di Fisica, via Marzolo 8
- Orario delle lezioni
  - Mercoledì 11:30—13:15, Aula A
  - Giovedì 11:30—13:15, Aula A

Ricordiamo...

- Testi di riferimento
  - Dispense *"Informatica di Base"* (Colussi, Filé, Rossi), presso la libreria Progetto
  - Donatella Sciuto, Giacomo Buonanno, Luca Mari, *"Introduzione ai sistemi informatici"*, terza edizione, McGraw-Hill, 2005, ISBN 88-386-6269-X
- Altri testi consigliati
  - J. Glenn Brookshear, *"Informatica—una panoramica generale"*, Addison-Wesley, 2004, ISBN 88-7192-184-4
  - Stefano Ceri, Dino Mandrioli, Licia Sbattella, *"Informatica: arte e mestiere"*, McGraw-Hill, 2004, ISBN 88-386-6140-5
- Lucidi delle lezioni e ulteriori risorse
  http://www.dsi.unive.it/~marzolla/teaching

Circuiti aritmetici

- Rappresentiamo gli interi positivi come sequenze di bit
- Quanto vale 101110_2?

$$101110_2 = 1\times2^5+0\times2^4+1\times2^3+1\times2^2+1\times2^1+0\times2^0$$
$$= 32 + 8 + 4 + 2$$
$$= 46$$

Conversione binario→decimale

- Esiste un modo molto semplice per convertire un numero binario in decimale
- Esempio: quanto vale 00110101_2?

128	64	32	16	8	4	2	1
0	0	1	1	0	1	0	1

- Risposta: 32 + 16 + 4 + 1 = 53
- Si sommano i numeri corrispondenti alle cifre binarie che valgono 1

Conversione decimale→binario

- Si può procedere così
  - Si divide il numero decimale ripetutamente per 2.
  - I resti della divisione danno le cifre della rappresentazione binaria, a partire dalla cifra meno significativa
- Es: come si scrive 74 in binario?
  - 74 / 2 = 37 resto 0
    37 / 2 = 18 resto 1
    18 / 2 = 9 resto 0
    9 / 2 = 4 resto 1
    4 / 2 = 2 resto 0
    2 / 2 = 1 resto 0
    1 / 2 = 0 resto 1

Cifra più a destra

1001010_2

Cifra più a sinistra

Conversione decimale→binario

- Come si scrive 197 in binario?
  - 197 / 2 = 98 resto 1
    98 / 2 = 49 resto 0
    49 / 2 = 24 resto 1
    24 / 2 = 12 resto 0
    12 / 2 = 6 resto 0
    6 / 2 = 3 resto 0
    3 / 2 = 1 resto 1
    1 / 2 = 0 resto 1

11000101_2

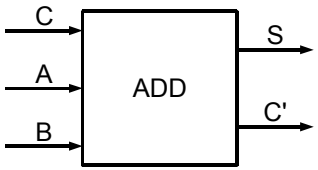
Somma binaria

- Quanto vale $010101_2 + 001100_2$?

Riporto	1	1	1	0	0	
	0	1	0	1	0	1_2 +
	0	0	1	1	0	0_2 =
Somma	1	0	0	0	0	1_2

Sommatore a 1 bit

- Vogliamo costruire un circuito che accetta in input tre bit A, B e C (Carry, ossia "riporto")
- In output produce la somma binaria S=A+B+C e il nuovo riporto C'



A	B	C	S	C'
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

Calcolare S

- La somma S vale 1 se tutti e tre gli input sono 1, oppure esattamente un input vale 1
- Cioè:

  S ≡ (A AND B AND C) OR
  (A AND (NOT B) AND (NOT C)) OR
  (B AND (NOT A) AND (NOT C)) OR
  (C AND (NOT A) AND (NOT B))
- Oppure

  S ≡ (A AND B AND C) OR
  (A AND (B NOR C)) OR
  (B AND (A NOR C)) OR
  (C AND (A NOR B))

A	B	C	S	C'
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

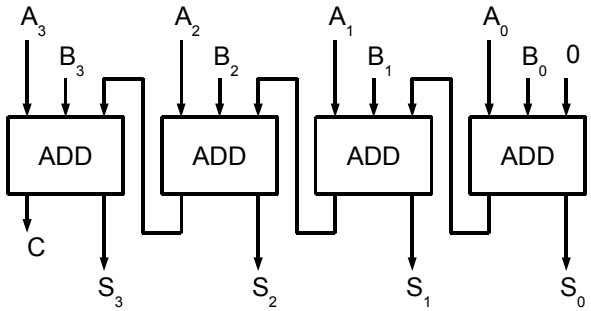
Calcolare C'

- C' vale 1 se almeno due input sono 1
- Cioè

  C' = (A AND B) OR
  (A AND C) OR
  (B AND C)
- Esercizio:
  - Disegnare il circuito per calcolare S e C'

Circuito sommatore

- Se vogliamo sommare due numeri di *N* bit, basta utilizzare *N* circuiti ADD.
  - Ad esempio, *N*=4
    $A_3A_2A_1A_0 + B_3B_2B_1B_0 = S_3S_2S_1S_0$ con riporto C



...e i numeri negativi?

- Fin qui abbiamo considerato solo l'aritmetica binaria con soli numeri positivi
- Come rappresentare numeri negativi, ad es -1?
- Si utilizza la rappresentazione in complemento a due
  - Supponendo di avere *N* bit in totale a disposizione, il numero *x* viene codificato in binario allo stesso modo di 2^N+x
  - Nota: 0 (decimale) in complemento a due si scrive $00...0_2$ (una stringa di *N* zeri)

Esempio / 1

- Supponiamo di avere *N*=4 bit, e di voler codificare il numero *x*=6
  - $2^N=2^4=16$
  - $2^N+x = 16+6 = 22$
  - 22 in binario diventa 10110_2
  - Però abbiamo a disposizione solo 4 bit, quindi scartiamo quello più a sinistra: rimane 0110_2
- Nota: la rappresentazione in complemento a 2 di *x*=6 coincide con la sua normale rappresentazione in base 2
  - Questo è valido per tutti i numeri positivi

Esempio / 2

- Supponiamo di avere ancora *N*=4 bit, e di voler codificare il numero *x*=-7
  - $2^N=2^4=16$
  - $2^N+x = 16-7 = 9$
  - 9 in binario si scrive 1001_2
- Osservazione 1
  - Con N bit è possibile rappresentare, in complemento a due, i numeri interi compresi tra -2^{N-1} e $2^{N-1}-1$ (estremi inclusi)
- Osservazione 2
  - Una stringa binaria rappresenta (in complemento a due) un numero positivo se e solo se il bit più a sinistra vale 0

Somma in complemento a due

- Caratteristica molto utile dell'aritmetica in complemento a due è che la somma tra due interi si può ancora fare con le solite regole dell'aritmetica binaria
- Vediamo un esempio

Esempio / 3

- Calcolare la somma 5+2, usando la notazione in complemento a due, con *N*=4 bit
  - $5 = 0101_2$
  - $2 = 0010_2$
  - Sommando $0101_2 + 0010_2$ con le solite regole per la somma binaria, si ha $0111_2 = 7$
- Calcolare la somma 5-7
  - $5 = 0101_2$
  - -7 si rappresenta come 2^4-7 = 16-7 = 9, ossia 1001_2
  - Sommando $0101_2 + 1001_2$ con le solite regole si ha 1110_2

Esempio / 4

- Quanto vale 1110_2 in complemento a due?
  - Ha il bit più a sinistra uguale a uno, quindi deve trattarsi di un numero negativo
  - $1110_2 = 2^4 + x$
  - quindi $x = 1110_2 - 16 = 14 - 16 = -2$
- Quanto vale 1110_2 come rappresentazione binaria di un intero non negativo?
  - $1110_2 = 8 + 4 + 2 = 14$

Valori rappresentabili

- Data una stringa binaria di *N* bit, quanti (e quali) numeri si possono rappresentare in complemento a due?
- Es: *N*=4

Comp. a due	Decimale	Comp. a due	Decimale
0000	0	1000	-8
0001	1	1001	-7
0010	2	1010	-6
0011	3	1011	-5
0100	4	1100	-4
0101	5	1101	-3
0110	6	1110	-2
0111	7	1111	-1

Errore di overflow / 1

- Supponiamo di sommare due interi *x* e *y* rappresentati da *N* bit in complemento a due
- Se *x* e *y* hanno segno diverso:
  - Il risultato sarà ancora un numero in complemento a due rappresentabile con *N* bit
  - Infatti: supponiamo che *x* sia positivo e *y* negativo

    $0 \le x \le 2^{N-1}-1 \qquad -2^{N-1} \le y \le 0$
  - Quindi:

    $-2^{N-1} \le x+y \le 2^{N-1}-1$
  - Dunque *x*+*y* sta dentro all'intervallo degli interi rappresentabile in complemento a due con *N* bit

Errore di overflow / 2

- Se invece *x* e *y* hanno lo stesso segno, potrebbe verificarsi *overflow*
- Esempio: calcoliamo -2-7, usando *N*=4 bit

Riporto	1	0	0	0	
		1	1	1	0_2 +
		1	0	0	0_2 =
Somma	~~1~~	0	1	1	0_2

- -2-7 = 6 !!!!

Errore di overflow / 3

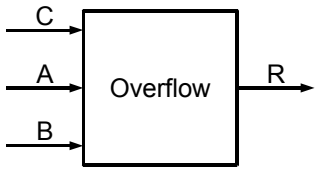
- Altro esempio: calcoliamo 3+6 usando *N*=4 bit

Riporto	0	1	1	0	
		0	0	1	1_2 +
		0	1	1	0_2 =
Somma		1	0	0	1_2

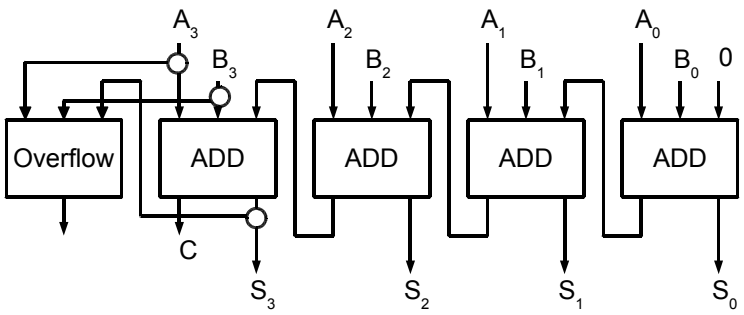
- 3+6 = -7 !!!!
- In generale: si ha overflow quando entrambe le seguenti condizioni si verificano
  - Gli operandi hanno lo stesso segno
  - Il segno del risultato è diverso da quello degli operandi

Esercizio

- Costruire un circuito logico che dati in input A, B (segni degli operandi, 0 indica positivo, 1 negativo) e C (segno del risultato) determina se nella somma si è verificato overflow
  - Suggerimento: partire dalla tabella di verità e costruire il circuito a partire da essa
  - R=1 se e solo se si è verificato overflow



Circuito sommatore con controllo di overflow



Codifica dei numeri reali / 1

- Come rappresentiamo un generico numero reale, ad es. 34,765?
- Normalmente i numeri si considerano in rappresentazione scientifica:
  - $34{,}765 = 0{,}34765 \times 10^{2}$
  - $0{,}007653 = 0{,}7653 \times 10^{-2}$
- Osserviamo
  - $0{,}34765 =$
    $3\times10^{-1} + 4\times10^{-2} + 7\times10^{-3} + 6\times10^{-4} + 5\times10^{-5}$

Codifica dei numeri reali / 2

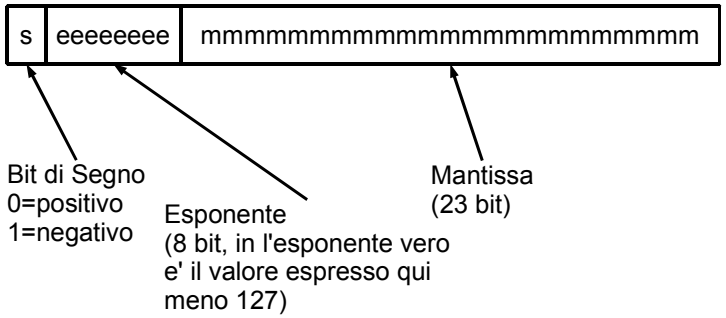
- Lo stesso si può applicare anche per la base 2
  - $0{,}1101_2 = 1\times2^{-1} + 1\times2^{-2} + 0\times2^{-3} + 1\times2^{-4}$
- In generale possiamo scrivere un numero reale in base 2 come

$$\boxed{+/-0{,}m_1m_2m_3\ldots \times 2^{e_1e_2e_3\ldots}}$$

- Dove:
  - $m_1m_2m_3\ldots$ sono cifre binarie della *mantissa*
  - $e_1e_2e_3\ldots$ rappresenta *l'esponente*
  - Poiché l'esponente potrebbe essere negativo, $e_1e_2e_3\ldots$ deve rappresentare un intero, ad esempio, usando la rappresentazione in complemento a 2

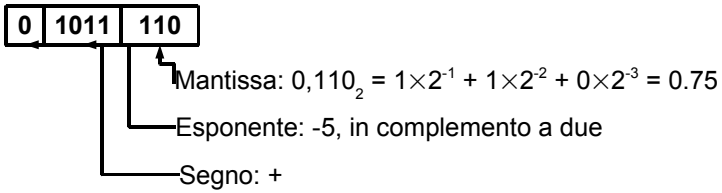
Codifica dei numeri reali / 3

- Solitamente si usa un numero fisso di cifre per la mantissa e per l'esponente
- Es: standard IEEE 754 singola precisione



Codice dei numeri reali / 4

- Esempio semplice:
  - 1 bit segno
  - 4 bit esponente (in complemento a due)
  - 3 bit mantissa
- Quanto vale



- Risposta: $+0.75 \times 2^{-5} = 0.0234375$