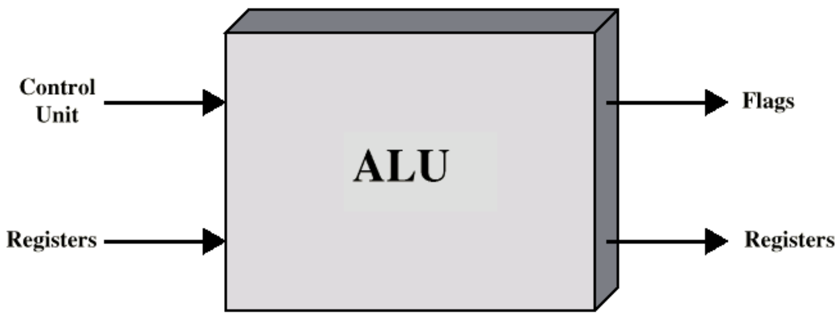
Aritmetica del calcolatore

Capitolo 9

Unità aritmetica e logica

- Esegue le operazioni aritmetiche e logiche
- Ogni altra componente nel calcolatore serve questa unità
- Gestisce gli interi
- Può gestire anche i numeri reali

Input e output della ALU



Rappresentazione degli interi

- Possiamo solo usare 0 e 1 per rappresentare tutto
- I numeri positivi sono scritti in binario come sappiamo
  - e.g. 41=00101001
- Non c’è bisogno del segno

Rappresentazione in modulo e segno

- Segno: bit più a sinistra
  - 0 significa positivo
  - 1 significa negativo
- Esempio:
  - +18 = 00010010
  - -18 = 10010010
- Problemi
  - Per eseguire operazioni aritmetiche bisogna considerare sia i moduli che i segni
  - Due rappresentazioni per lo 0: +0 and -0

Rappresentazione in complemento a due

- Segno nel bit più a sinistra
- Per n bit: possiamo rappresentare tutti i numeri da -2^{n-1} a $+2^{n-1} - 1$
- Per i numeri positivi, come per modulo e segno
  - n zeri rappresentano lo 0, poi 1, 2, … in binario per rappresentare 1, 2, … positivi
- Per i numeri negativi, da n uni per il -1, andando indietro

Rappresentazione in complemento a due

- +3 = 00000011
- +2 = 00000010
- +1 = 00000001
- +0 = 00000000
- -1 = 11111111
- -2 = 11111110
- -3 = 11111101

Complemento a due su 3 e 4 bit

a. Using patterns of length three

Bit pattern	Value represented
011	3
010	2
001	1
000	0
111	-1
110	-2
101	-3
100	-4

b. Using patterns of length four

Bit pattern	Value represented
0111	7
0110	6
0101	5
0100	4
0011	3
0010	2
0001	1
0000	0
1111	-1
1110	-2
1101	-3
1100	-4
1011	-5
1010	-6
1001	-7
1000	-8

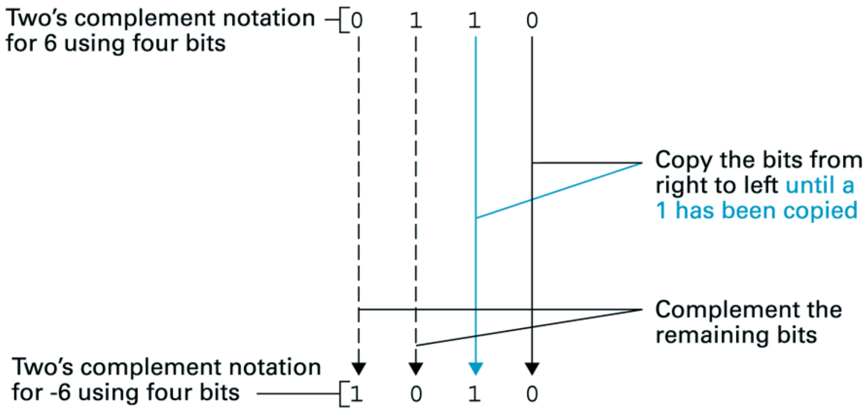
Complemento a due: numeri negativi

- Confrontiamo le rappresentazioni di k e –k
  - da destra a sinistra, uguali fino al primo 1 incluso
  - poi una il complemento dell'altra
- Esempio (su 4 bit): 2=0010, -2=1110

Complemento a due: decodifica

- Se bit di segno =0 ➔ positivo, altrimenti negativo
- Se positivo, basta leggere gli altri bit
- Se negativo, scrivere gli stessi bit da destra a sinistra fino al primo 1, poi complementare, e poi leggere
- Es.: 1010 è negativo, rappresenta 0110 (6), quindi -6

Da k a -k



Complemento a due: altro metodo

- Data la rappresentazione di k (positivo), -k si può anche ottenere così:
  - Complemento bit a bit della rappresentazione di k
  - Somma di 1 al risultato
- Esempio:
  - 2=0010
  - Complemento: 1101
  - 1101 +1 = 1110
  - -2=1110

Complemento a due: in generale

- Positivi: da 0 (n zeri) a 2^{n-1} -1 (uno zero seguito da n-1 uni)
- Negativi:
  - Bit di segno a 1
  - I restanti n-1 bit possono assumere 2^{n-1} configurazioni diverse, quindi da -1 a -2^{n-1}
- Se sequenza di bit $a_{n-1}\ a_{n-2} \ldots a_1\ a_0$,

  numero = $-2^{n-1} \times a_{n-1} + \sum_{(i=0, \ldots, n-2)} 2^i \times a_i$
- Numeri positivi: $a_{n-1} = 0$
- Numeri negativi: positivo – 2^{n-1}

Benefici

- Una sola rappresentazione dello zero
- Le operazioni aritmetiche sono facili
- La negazione è facile
  - 3 = 00000011
  - Complemento Booleano 11111100
  - Somma di 1 11111101

Numeri rappresentabili

- Complemento a 2 su 8 bit
  - Numero più grande: +127 = 01111111 = 2^7 -1
  - Numero più piccolo: -128 = 10000000 = -2^7
- Complemento a 2 su 16 bit
  - +32767 = 011111111 11111111 = 2^{15} - 1
  - -32768 = 100000000 00000000 = -2^{15}

Esercizi

- Da complemento a 2 a base 10:
  - 00011, 01111, 11100, 11010, 00000, 10000
- Da base 10 a complemento a 2 su 8 bit:
  - 6, -6, 13, -1, 0
- Numero più grande e più piccolo per la notazione in complemento a 2 su 4, 6, 8 bit

Conversione tra diverse lunghezze

- Da una rappresentazione su n bit ad una rappresentazione dello stesso numero su m bit (m > n)
- Modulo e segno: facile
  - Bit di segno nel bit più a sinistra
  - M-n zeri aggiunti a sinistra
  - Esempio (da 4 a 8 bit): 1001 ➔ 10000001

Conversione tra diverse lunghezze

- Complemento a 2: stessa cosa del modulo e segno per numeri positivi
- Per numeri negativi: replicare il bit di segno dalla posizione attuale alla nuova
- Esempi:
  - +18 (8 bit) = 00010010
  - +18 (16 bit) = 00000000 00010010
  - -18 (8 bit) = 10010010
  - -18 (16 bit) = 11111111 10010010

Negazione su numeri in complemento a 2

- Due passi:
  - Complemento
  - Somma 1

Negazione: caso speciale 1

- 0 = 00000000
- Complemento: 11111111
- Somma 1: +1
- Risultato: 1 00000000
- L'uno più a sinistra è un overflow, ed è ignorato. Quindi - 0 = 0

Negazione: caso speciale 2

- -128 = 10000000
- Complemento: 01111111
- Somma 1: +1
- Risultato: 10000000
- Quindi, -(-128) = -128 !
- 2^n stringhe su n bit, un numero positivo in più di quelli negativi: -2^n si può rappresentare, ma $+2^n$ no ➔ -2^n non può essere complementato

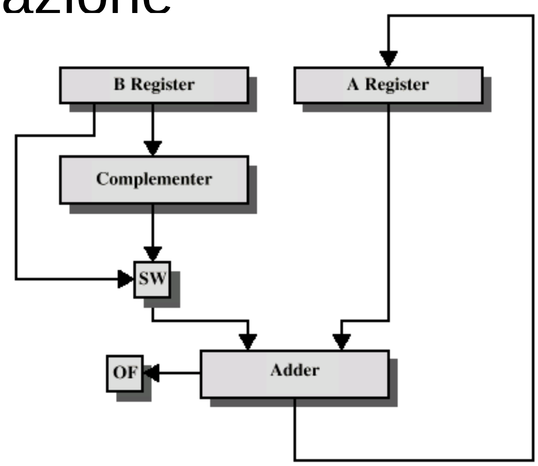
Somma e sottrazione

- Per la somma: normale somma binaria
  - Controllare il bit di segno per l'overflow
- Per la sottrazione: basta avere i circuiti per somma e complemento
  - Es. (4 bit): 7-5 = 7 +(-5) = 0111 + 1011 = 0010
  - 5 = 0101 ➔ -5 = 1011

Esempi di somme

Problem in base ten	Problem in two's complement	Answer in base ten
3 + 2	0011 + 0010 = 0101	5
−3 + −2	1101 + 1110 = 1011	−5
7 + −5	0111 + 1011 = 0010	2

Hardware per somma e sottrazione



OF = overflow bit
SW = Switch (select addition or subtraction)

Overflow

- Overflow: quando si sommano due numeri positivi tali che il risultato è maggiore del massimo numero positivo rappresentabile con i bit fissati (lo stesso per somma di due negativi)
- Se la somma dà overflow, il risultato non è corretto
- Come si riconosce? Basta guardare il bit di segno della risposta: se 0 (1) e i numeri sono entrambi negativi (positivi) ➔ overflow

Esempi di somme

- -4 (1100) + 4 (0100) = 10000 (0)
  - Riporto ma non overflow
- -4 (1100) - 1 (1111): 11011 (-5)
  - Riporto ma non overflow
- -7 (1001) -6 (1010) = 10011 (non è -13, ma 3)
  - Overflow
- + 7 (0111) + 7 (0111) = 1110 (non è 14, ma -2)
  - Overflow

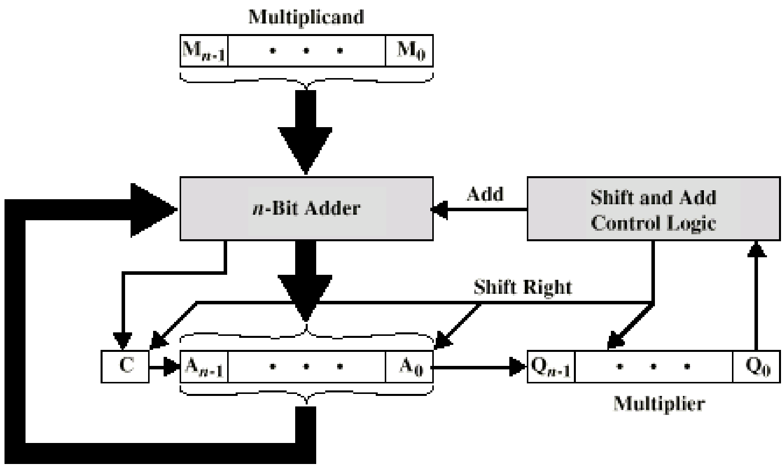
Moltiplicazione

- Più complessa
- Calcolare il prodotto parziale per ogni cifra
- Sommare i prodotti parziali

Esempio di moltiplicazione

- 1011 Moltiplicando (11 decimale)
- x 1101 Moltiplicatore (13 decimale)
- 1011 Prodotto parziale 1
- 0000 Prodotto parziale 2
- 1011 Prodotto parziale 3
- 1011 Prodotto parziale 4
- 10001111 Prodotto (143 decimale)
- Nota: da due numeri di n bit potremmo generare un numero di 2n bit

Moltiplicazione di interi senza segno



(a) Block Diagram

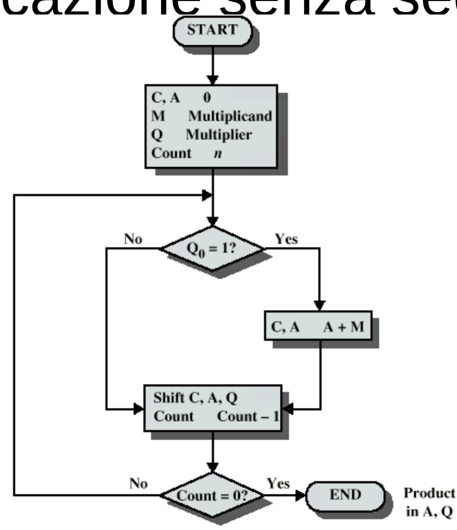
Implementazione

- Se Q_0 = 0, traslazione di C, A e Q
- Se Q_0 = 1, somma di A e M in A, overflow in C, poi traslazione di C, A, e Q
- Ripetere per ciascun bit di Q
- Prodotto (2n bit) in A e Q

Un esempio

C	A	Q	M		
0	0000	1101	1011	Initial Values	
0	1011	1101	1011	Add	First
0	0101	1110	1011	Shift	Cycle
0	0010	1111	1011	Shift	Second Cycle
0	1101	1111	1011	Add	Third
0	0110	1111	1011	Shift	Cycle
1	0001	1111	1011	Add	Fourth
0	1000	1111	1011	Shift	Cycle

Diagramma di flusso pr la moltiplicazione senza segno



Moltiplicare numeri in complemento a 2

- Per la somma, i numeri in complemento a 2 possono essere considerati come numeri senza segno
- Esempio:
  - 1001 + 0011 = 1100
  - Interi senza segno: 9+3=12
  - Complemento a 2: -7+3=-4

Moltiplicare numeri in complemento a 2

- Per la moltiplicazione, questo non funziona!
- Esempio: 11 (1011) x 13 (1101)
  - Interi senza segno: 143 (10001111)
  - Se interpretiamo come complemento a 2: -5 (1011) x -3 (1101) dovrebbe essere 15, invece otteniamo 10001111 (-113)
- Non funziona se almeno uno dei due numeri è negativo