

CAPITOLO 5: Rappresentazione di dati numerici

The Architecture of Computer Hardware and Systems Software & Networking: An Information Technology Approach

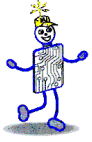
5th Edition, Irv Englander

John Wiley and Sons ©2013

Diapositive realizzate da Angela Clark, University of South Alabama

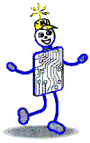
Diapositive per la 4° edizione realizzate da Wilson Wong, Bentley University

Diapositive per il CLEll tradotte e adattate da Gianluca Amato, Univ. CH-PE



Rappresentazione dei numeri

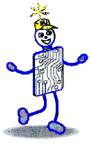
- I numeri possono essere rappresentati con una combinazione di
 - Valore assoluto
 - Segno (più o meno)
 - Cifre decimali (se necessarie)



Numeri senza segno

- Binario: Numero intero senza segno in binario
- BCD: Rappresentazione binaria cifra per cifra del numero
 - 4 bits: 0 to 9
 - 8 bits: 0 to 99
 - 16 bits: 0 to 9,999
 - 32 bits: 0 to 99,999,999

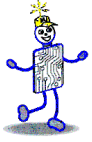
Decimale	Binario	BCD (binary coded decimal)		
68	= 0100 0100 = $2^6 + 2^2 = 64 + 4 = \mathbf{68}$	= 0110	1000	
		= $2^2 + 2^1 = \mathbf{6}$	$2^3 = \mathbf{8}$	
99 (largest 8-bit BCD)	= 0110 0011 = $2^6 + 2^5 + 2^1 + 2^0 =$ = $64 + 32 + 2 + 1 = \mathbf{99}$	= 1001	1001	
		= $2^3 + 2^0 =$	$2^3 + 2^0 =$	
		= 9	9	
255 (largest 8-bit binary)	= 1111 1111 = $2^8 - 1 = \mathbf{255}$	= 0010	0101	0101
		= 2^1	$2^2 + 2^0$	$2^2 + 2^0$
		= 2	5	5



Intervallo rappresentabile

- Intervallo dei valori rappresentabili in BCD < intervallo valore rappresentabili in binario tradizionale
 - Binario: 4 bits possono rappresentare 16 differenti valori (da 0 a 15)
 - BCD: 4 bits possono rappresentare solo 10 valori differenti (da 0 a 9)

No. of Bits	BCD Range		Binary Range	
4	0-9	1 digit	0-15	1+ digit
8	0-99	2 digits	0-255	2+ digits
12	0-999	3 digits	0-4,095	3+ digits
16	0-9,999	4 digits	0-65,535	4+ digits
20	0-99,999	5 digits	0-1 million	6 digits
24	0-999,999	6 digits	0-16 million	7+ digits
32	0-99,999,999	8 digits	0-4 billion	9+ digits
64	0-(10 ¹⁶ -1)	16 digits	0-16 quintillion	19+ digits



Binario vs BCD

- Generalmente la rappresentazione binaria è preferita
 - Intervallo di valori rappresentabili più ampio a parità di bit
 - Calcoli più semplici
- Il BCD è usato spesso in applicazioni aziendali per non avere problemi di arrotondamento e conversione dal decimale
 - ci torneremo quando parleremo di numeri con la virgola



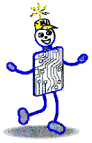
Interi con segno

- Non c'è una maniera univoca di rappresentare il segno in binario
- Opzioni:
 - Rappresentazione *modulo e segno*
 - Complemento a 1
 - Complemento a 2 (la più comune)



Modulo e segno

- Usare il bit più a sinistra per il segno
 - 0 = più; 1 = meno
- Intervallo di interi rappresentabili:
 - Metà degli interi sono positivi, metà negativi
 - Valore assoluto dell'intero più grande è (circa) la metà del caso senza segno
- Esempio usando 8 bit:
 - Senza segno: $1111\ 1111 = +255$
 - Con segno: $0111\ 1111 = +127$
 $1111\ 1111 = -127$
 - Nota: 2 valori per 0:
 $+0\ (0000\ 0000)$ e $-0\ (1000\ 0000)$



Algoritmi di calcolo complessi

- Gli algoritmi di calcolo per la rappresentazione modulo e segno sono complessi e difficile da implementare in hardware
 - Bisogna controllare i 2 valori of 0
 - Si esegue in realtà addizione o sottrazione sulla base dei segni e della grandezza dei numeri coinvolti
 - Utile per la rappresentazione BCD
- Esempio: Algoritmo di addizione in base 10

Addizione:
2 Positive Numbers

$$\begin{array}{r} 4 \\ +2 \\ \hline 6 \end{array}$$

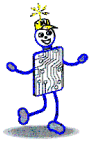
Addizione:
1 Signed Number

$\begin{array}{r} 4 \\ -2 \\ \hline 2 \end{array}$	$\begin{array}{r} 2 \\ -4 \\ \hline -2 \end{array}$	$\begin{array}{r} 12 \\ -4 \\ \hline 8 \end{array}$
--	---	---



Rappresentazione in complemento

- Il segno dei numeri non va trattato in maniera speciale
- Operazioni svolte in maniera consistente per tutte le combinazioni di segno e valore degli addendi
- Due metodi
 - Alla base / Alla base diminuita
 - Nel sistema decimale
 - il complemento alla base è il complemento a 10
 - Il complemento alla base diminuita è il complemento a 9



Complemento a 9

- *Fare il complemento di v*: sottrarre v da un valore standard che dipende dalla base
- Base 10: complemento a base diminuita = complemento a 9
 - Esempio con 3 cifre: valore standard da cui sottrarre = 999
 - L'intervallo di possibili valore da 000 a 999 si divide arbitrariamente a 500
 - Due numeri per lo zero

Numberi	Negativo		Positivo	
Rappresentazione	Complemento		Il numero stesso	
Range of decimal numbers	-499	-000	+0	499
Calculation	999 - number		nessuno	
Representation example	500	999	0	499

$999 - 499$ **-** Valori crescenti **+**



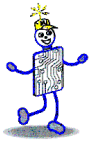
Complemento a 9 (base 10)

- È necessario specificare il numero di cifre, detto anche *dimensione della parola*
- Esempio: rappresentazione con 3 cifre
 - Prima cifra = da 0 a 4 → numero positivo
 - Prima cifra = da 5 a 9 → numero negativo
- Conversione da complemento a 9 in rappresentazione usuale
 - **321** rimane **321**
 - **521**: fare il complemento - $(999 - 521) = -478$



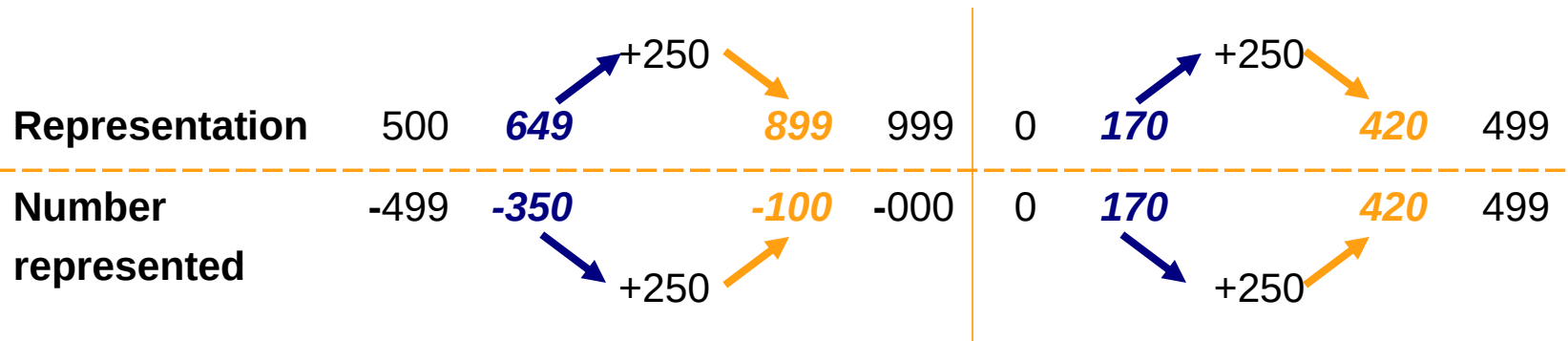
Complemento e negazione

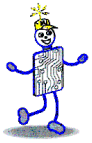
- L'operazione di complemento ci consente di passare dalla rappresentazione di un numero alla rappresentazione del suo opposto
- Esempio
 - $999 - 478 = 521$
 - $999 - 521 = 478$
- Se complementiamo un numero due volte, si torna al valore iniziale
 - Complement = basis – value
 - Complement twice
 - Basis – (basis – value) = value



Addizione modulare

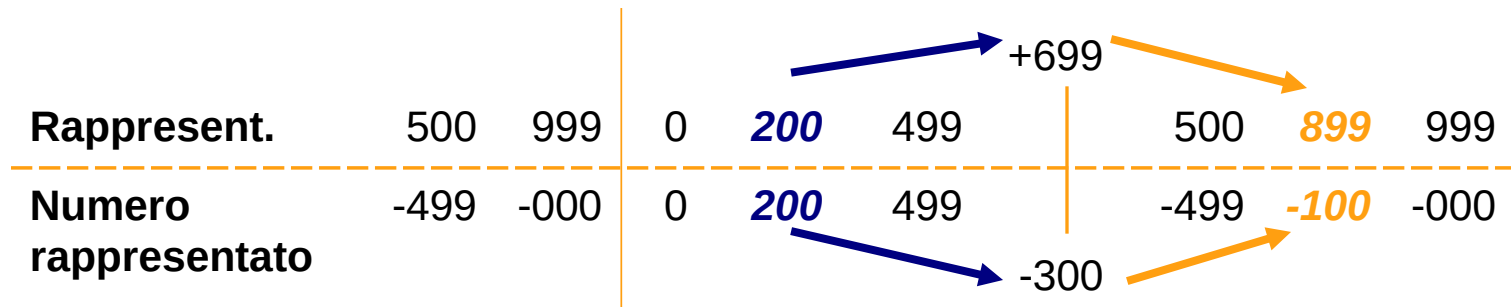
- Muoversi verso l'alto nella scala corrisponde all'addizione
- Esempio in complemento a 9: non si attraversa il valore standard 999



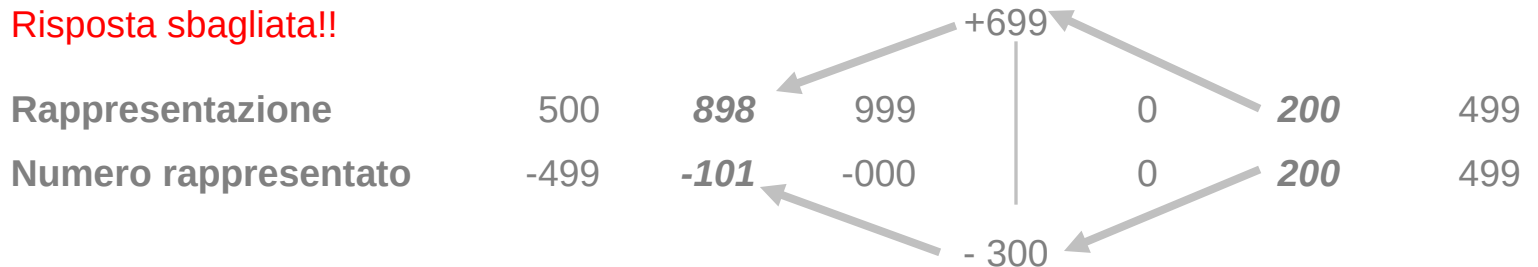


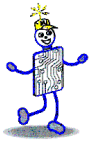
Addizione con wraparound

- Contare verso destra per aggiungere un valore negativo
- Contare verso sinistra attraverserebbe il modulo e darebbe una risposta scorretta perché ci sono due valori di 0 (+0 e -0)



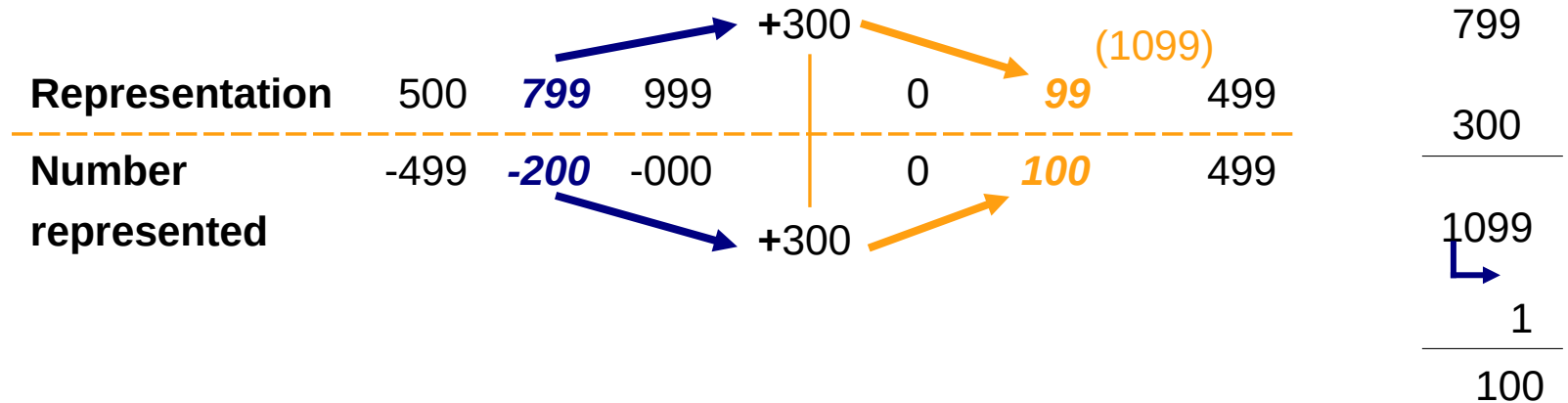
Risposta sbagliata!!





Addizione con riporto

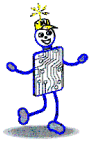
- Contare a destra può attraversare 999
 - Il risultato è sbagliato perché si attraversano due zeri
- Addizione con end-around carry:
 - Sommare 2 numeri in complemento a 9 normalmente
 - Se si genera un riporto, sommarlo al risultato





Overflow

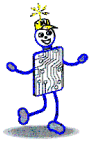
- Una parola di dimensione fissa ha un intervallo di valori rappresentabili limitato
- *Overflow*: combinazione di numeri che, sommati, generano un risultato fuori dall'intervallo consentito
 - Nella notazione in complemento, l'overflow non si verifica mai se si sommano numeri di segno opposto
 - Se si sommano numeri dello stesso segno ma il risultato è di segno opposto, si è verificato un overflow.



Complemento a 1 (base 2)

- *Fare il complemento di v*: sottrarre v da un valore standard che dipende dalla base
 - Base 2: complemento a base diminuita = base 1
 - Esempio con 3 cifre: $v \rightarrow 111 - v$
- *Inversione: si cambiano gli 1 in zero e viceversa*
 - Numeri che iniziano con 0 sono positivi
 - Numeri che iniziano con 1 sono negativi
 - 2 valori per lo zero
- Esempio con numeri a 8 bit

Numero	Negativo		Positivo	
Rappresentazione	Complemento		Numero stesso	
Intervallo numerico	-127_{10}	-0_{10}	$+0_{10}$	127_{10}
Calcolo	Inversione		Nessuno	
Esempio rappresentazione	10000000	11111111	00000000	01111111



Addizione

- Sommare 2 numeri positivi a 8 bit

$$0010\ 1101 = 45$$

$$0011\ 1010 = 58$$

$$\hline 0110\ 0111 = 103$$

- Sommare 2 numeri a 8 bit di segno diverso

$$0010\ 1101 = 45$$

- Prendere il complemento a 1 di 58 (invertire)

$$\underline{1100\ 0101} = -58$$

$$1111\ 0010 = -13$$

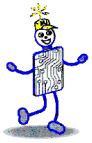
0011 1010

1100 0101

Inverti per
ottenere il
valore assoluto

0000 1101

$$8 + 4 + 1 = 13$$



Addizione con riporto

- Numero a 8 bit

- Inversione

0000 0010 (2_{10})

1111 1101

- Addiziona

- 9 bit

Aggiunta riporto

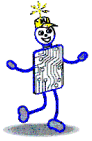
0110 1010 = 106

1111 1101 = -2

10110 0111

+1

0110 1000 = 104



Sottrazione

- Numero a 8 bit

$$0110\ 1010 = 106$$

- Inversione

0101 1010 (90_{10})

$$\underline{-0101\ 1010 = 90}$$

1010 0101

$$0110\ 1010 = 106$$

- Addizione

$$\underline{-1010\ 0101 = 90}$$

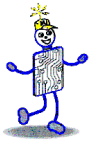
- 9 bit

$$\textcircled{1}0000\ 1111$$

Aggiunta riporto

$$\begin{array}{l} \text{└─} \\ \text{└─} \end{array} +1$$

$$\underline{0001\ 0000 = 16}$$



Overflow

- Numeri a 8 bit
 - 256 combinazioni diverse
 - Numeri positivi:
0 to 127

- Addizione
 - Controllare gli *overflow*
 - 2 numeri positivi producono un risultato negativo → *overflow!*
 - **Risposta sbagliata!**

- Attenzione: la maggior parte dei linguaggi non controlla il verificarsi di overflow!

$$0100\ 0000 = 64$$

$$0100\ 0001 = 65$$

$$1000\ 0001 = -126$$

$$0111\ 1110$$

$$126_{10}$$



Inverti per
ottenere il
valore
assoluto

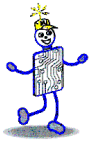




Complemento a 10

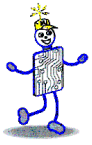
- Uso la base per l'operazione di complementazione
 - Su 3 cifre, il valore da cui sottrarre è $10^3 = 1000$
 - L'intervallo di possibili valore da 000 a 999 si divide arbitrariamente a 500
 - Una sola rappresentazione per 0

Numero	Negativo		Positivo	
Rappresentazione	Complemento		Numero stesso	
Intervallo numerico	-500	-001	0	499
Calcolo	1000 - il numero		nessuno	
Esempio rappresentazione	500	999	0	499



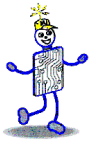
Esempi con numeri di 3 cifre

- Esempio 1:
 - Rappresentazione in complemento a 10 di 247
 - 247 (positive number)
 - Complemento a 10 di 247
 - $1000 - 247 = 753$ (negative number)
- Esempio 2:
 - Numero corrispondente al valore 17 in rappresentazione con complemento a 10
 - Positivo perché prima cifra è 0 (su 3 cifre, $17 = 017$)
 - $1000 - 017 = 983$
- Esempio 3:
 - Numero corrispondente al valore 777 in rappresentazione con complemento a 10
 - Numero negativo perché la prima cifra è 7
 - $1000 - 777 = 223$
 - Valore in notazione convenzionale = -223



Metodo alternativo per il complemento a 10

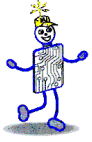
- Basato sul complemento a 9
- Esempio usando numeri di 3 cifre
 - Notare: $1000 = 999 + 1$
 - Complemento a 9 di $v = 999 - v$
 - Riscrivendo
 - Complemento a 10 di $v = 1000 - v = 999 + 1 - v$
 - Complemento a 10 = complemento a 9 + 1
- Più facile da effettuare, specialmente quando si lavora con numeri binari



Complemento a 2

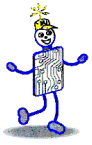
- Modulo = Un 1 seguito dal numero di zeri che dipende dalla dimensione della parola
 - Per 8 bit, sottrarre da $2^8 = 100000000_2$ (modulo)
- Due modo per trovare il complemento
 - Sottrarre dal modulo o invertire e sommare 1

Numero	Negativo		Positivo	
Rappresentazione	Complemento		Il numero stesso	
Intervallo numerico	-128_{10}	-1_{10}	$+0_{10}$	127_{10}
Calcolo	Complemento		Nessuno	
Esempio rappresentazione	10000000	11111111	00000000	01111111



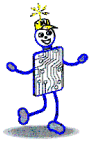
Overflow e riporto

- *Riporto (carry)*: si verifica quando una addizione o sottrazione supera il numero di cifre stabilite
- *Overflow*: il risultato della addizione o sottrazione altera il segno del risultato



Notazione esponenziale

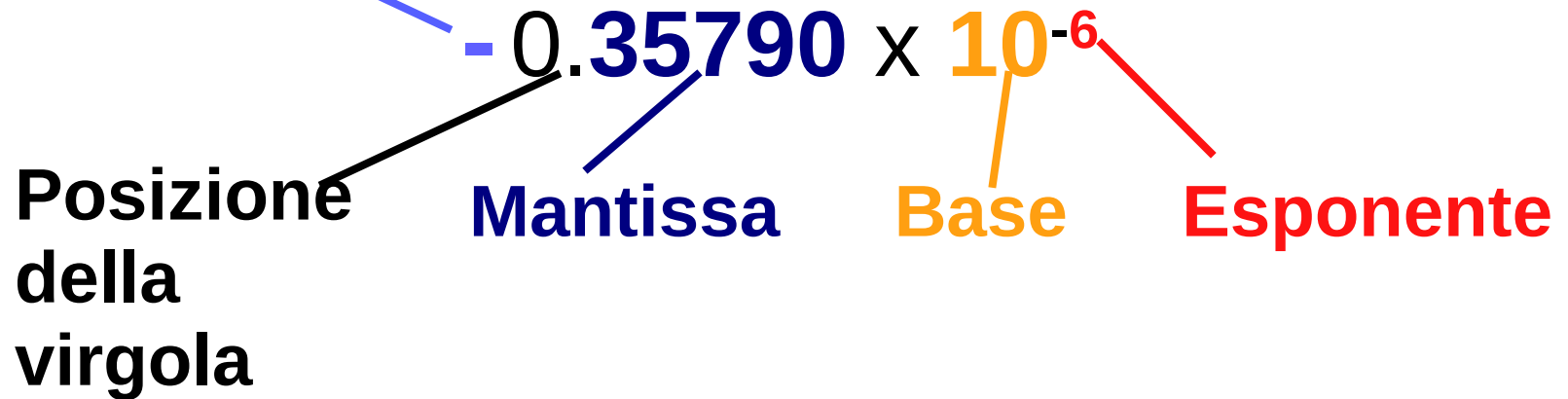
- Anche chiamata *notazione scientifica*
 - 12345
 - 12345×10^0
 - 0.12345×10^5
 - 123450000×10^{-4}
- 4 informazioni richieste per un numero
 1. Segno (“+” nell’esempio)
 2. Valore assoluto o *mantissa* (12345)
 3. Segno dell’esponente (“+” in 10^5)
 4. Valore assoluto dell’esponente (5)
- E in più
 5. Base dell’esponente (10)
 6. Posizione della virgola

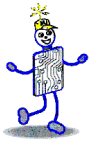


Riepilogo

Segno mantissa

Segno esponente





Esempio formato con 8 cifre

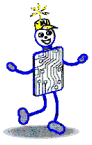
- Decidiamo di rappresentare un numero con 8 cifre decimali
 - Tradeoff tra precisione mantissa e intervallo di valori possibili

Segno della mantissa

SEEMMMM

2 cifre per esponente

5 cifre per mantissa

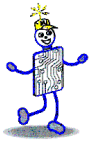


Formato

- Mantissa: nel formato *modulo e segno*
- Assumiamo che la virgola sia all'inizio della mantissa.
- Notazione eccesso-N: simile alla notazione in complemento
 - Considerare N come lo 0, N+1 come 1, N+2 come 2, N-1 come -1, etc....

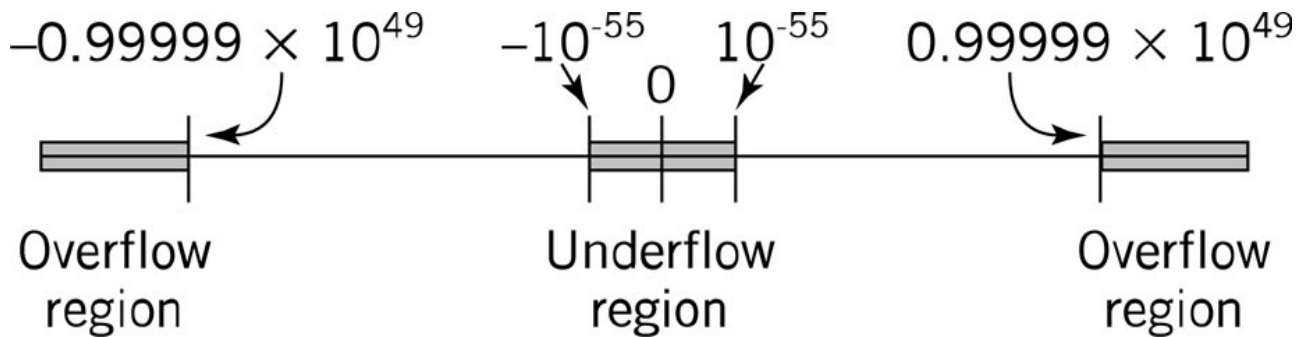
Representation	0	49	50	99
Exponent being represented	-50	-1	0	49

- Increasing value +
5-34



Overflow e Underflow

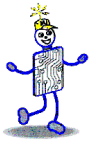
- È possibile che il numero sia troppo grande o troppo piccolo per la rappresentazione





Calcoli in virgola mobile

- Addizione e sottrazione
 - Esponente e mantissa trattati separatamente
 - Gli esponenti dei numeri devono essere uguali
 - Allineare i punti decimali
 - Cifre meno significative possono essere perse
 - L'overflow della mantissa richiede la modifica dell'esponente.



Addizione e sottrazione

Somma 2 numeri in virgola mobile

05199520

\pm 04967850

Allinea gli esponenti

05199520

0510067850

Somma le mantisse; (1) indica riporto

(1)0019850

Il riporto richiede traslazione a destra

05210019(850)

Arrotondamento

05210020

Controllo dei risultati

$$05199520 = 0.99520 \times 10^1 = 9.9520$$

$$04967850 = 0.67850 \times 10^1 = \underline{0.06785}$$

$$= 10.01985$$

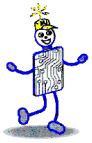
In forma esponenziale

$$= 0.1001985 \times 10^2$$



Moltiplicazione e divisione

- Mantisse: moltiplicate o divise
- Esponenti: sommate o sottratte
 - Normalizzazione necessaria per
 - Rimettere a posto la posizione della virgola
 - Mantenere la precisione del risultato
 - Sistemare il valore di eccesso perché è stato aggiunto due volte
 - Esempio: 2 numeri con esponente= 3 rappresentati in notazione eccesso-50
 - $53 + 53 = 106$
 - Poiché 50 è stato aggiunto due volte, sottrarlo per ottenere il valore corretto: $106 - 50 = 56$



Moltiplicazione e divisione

- Mantenere la precisione

- Normalizzazione e arrotondamento

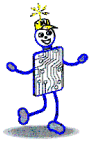
- Moltiplicare 2 numeri 05220000
 $\times \quad 04712500$
 - Somma esponenti, sottrai
eccesso $52 + 47 - 50 = 49$
 - Moltiplica mantisse $0.20000 \times 0.12500 = 0.025000000$
 - Normalizza il risultato 04825000
 - Arrotonda il risultato 04825000
 - Controlla il risultato

$$05220000 = 0.20000 \times 10^2$$

$$04712500 = 0.125 \times 10^{-3}$$

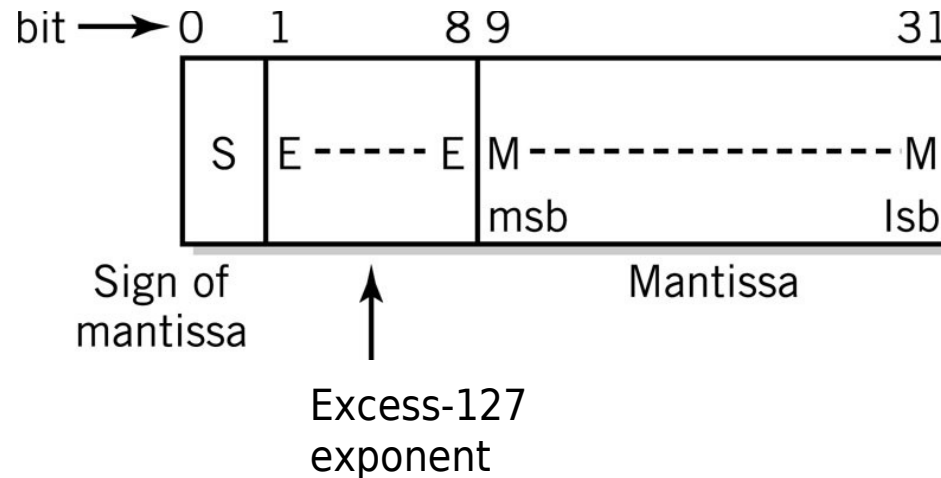
$$= 0.0250000000 \times 10^{-1}$$

- Normalizzando e arrotondando 0.25000×10^{-2}



Virgola mobile nel computer

- Tipico formato in virgola mobile
 - 32 bits forniscono un intervallo da $\sim 10^{-38}$ to 10^{+38}
 - Esponente ad 8-bit = 256 valori possibili
 - Notazione eccesso-127
 - 23 bit per mantissa: circa 7 cifre decimali di precisione





Standard IEEE 754

- Definizione dei valori virgola mobile a 32 bit

Exponent	Mantissa	Value
0	± 0	0
0	Not 0	$\pm 2^{-126} \times 0.M$
$1-254$	Any	$\pm 2^{-127} \times 1.M$
255	± 0	$\pm \infty$
255	not 0	special condition



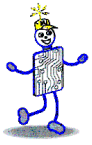
Considerazione per la programmazione

- Vantaggi degli interi
 - Più facile lavorarci sopra per i computer
 - Potenziale per una precisione maggiore
 - Esecuzione più rapida
 - Minore occupazione di memoria
- La maggior parte dei linguaggi ad alto livello fornisce 2 o più formati
 - Short integer (16 bits)
 - Long integer (64 bits)



Considerazione per la programmazione

- Numeri in virgola mobile
 - Quando usarli?
 - Variabili o costanti hanno una parte frazionale
 - Numeri molto grandi al di fuori dell'intervallo rappresentabile con gli interi
 - I programmi dovrebbero usare la precisione minima necessaria per il compito che si prefiggono
 - La notazione BCD a virgola fissa è una alternativa per le applicazioni aziendali



Copyright 2013 John Wiley & Sons

All rights reserved. Reproduction or translation of this work beyond that permitted in section 117 of the 1976 United States Copyright Act without express permission of the copyright owner is unlawful. Request for further information should be addressed to the Permissions Department, John Wiley & Sons, Inc. The purchaser may make back-up copies for his/her own use only and not for distribution or resale. The Publisher assumes no responsibility for errors, omissions, or damages caused by the use of these programs or from the use of the information contained herein.”