UNIVERSITÀ DEGLI STUDI DI TRIESTE

Operazioni nel sistema binario

Prof.ssa Giulia Cisotto

giulia.cisotto@units.it

Trieste, 26 febbraio 2026

Conversioni tra sistemi numerici diversi (2, 10, 16)

Posso rappresentare il numero 41 con *n=5* bit?

Overflow!

1	0	0	1	0	1
b_5	b_4	b_3	b_2	b_1	b_0

MSB = Most Significant Bit

LSB = Least Significant Bit

RAPPRESENTAZIONE NUMERI INTERI NEGATIVI	RANGE DI VALORI RAPPRESENTABILI
«Modulo e segno»	$[-(2^{N-1}-1), 2^{N-1}-1]_{10}$
CA1	$[-(2^{N-1}-1), 2^{N-1}-1]_{10}$
CA2	$[-2^{N-1}, 2^{N-1}-1]_{10}$
MSB = 0 → POSITIVO, MSB = 1 → NEGATIVO	

METODO DI RAPPRESENTAZIONE	RANGE DI VALORI RAPPRESENTABILI
Virgola fissa *unsigned*	$[\mathbf{0}, 2^{L}-2^{-D}]$
Virgola fissa *signed*	$[-2^{L-1}, 2^{L-1}-2^{-D}]_{10}$
Virgola mobile (IEEE 754) – single precision	$[\pm 1.4\times10^{-45}, \pm 3.4\times10^{38}]$
Virgola mobile (IEEE 754) – double precision	$[\pm 4.9\times10^{-324}, \pm 1.8\times10^{308}]$

Virgola fissa unsigned

L-1	L-2	…	0	-1	-2	…	-D
PARTE INTERA				PARTE FRAZIONARIA			

Per essere più specifici su quante cifre si usano per parte intera/frazionaria esistono i **FORMATI UQ e Q**. Esempio: UQ3.5 è numero in virgola fissa unsigned con 3 cifre intere e 5 frazionarie.

$$\mathbf{N} = d_{n-1}\cdot r^{n-1}+d_{n-2}\cdot r^{n-2}+\ldots+d_1\cdot r^1+d_0\cdot r^0+d_{-1}\cdot r^{-1}+\ldots+d_{-m}\cdot r^{-m} = \sum_{i=-m}^{n-1} d_i\cdot r^i$$

$\mathbf{N} = 3.625_{10}$ → UQ3.5

$3_{10} = 011_2$ Esteso a 3 bit

$(0.625)_{10} = (0.10100)_2$ Esteso a 5 bit

→ $(3.625)_{10} = (011.10100)_2$

$(011.10100)_2$ Formato UQ3.5 →

$(011.10100)_2 = 1\cdot 2 + 1 + 1\cdot 2^{-1} + 1\cdot 2^{-3} = 3 + 0.5 + 0.125 = 3.625$

Virgola fissa SIGNED

Assumiamo di avere **N bit** a disposizione.

Usa CA2!

- L < N bit per rappresentare la parte intera del numero, incluso il segno
- D=N-L bit per rappresentare la parte decimale del numero

L'intervallo di numeri interi rappresentabile è quindi: $[-2^{L-1}-1, 2^{L-1}-1]$

Mentre quello della parte decimale è: $[0, 1-2^{-D}]$

Algoritmo per la codifica in Q-format per virgola fissa SIGNED:

- Scrivere il valore assoluto in binario con virgola (modo standard)
- Adattare la parte intera e la frazionaria al numero di bit desiderati
- Scrivere la sequenza di bit completa (parte intera | parte frazionaria)
- Fare il CA2
- Posizionare la virgola nel punto desiderato

Esempio:

$$\mathbf{N} = -5.25_{10}$$

Q5.3

Esteso a 5 bit

$$5_{10} = 00101_2$$

$$0.25_{10} = 0.010_2$$

Esteso a 3 bit

$$00101.010_2$$

CA2

$$11010101_2 + 1 = 11010110_2$$

$$11010.110_2$$

$$\mathbf{N} = -d_{n-1} \cdot r^{n-1} + d_{n-2} \cdot r^{n-2} + \ldots + d_1 \cdot r^1 + d_0 \cdot r^0 + d_{-1} \cdot r^{-1} + \ldots + d_{-m} \cdot r^{-m}$$

$$11010.110_2 = -1 \cdot 2^4 + 1 \cdot 2^3 + 1 \cdot 2 + 1 \cdot 2^{-1} + 1 \cdot 2^{-2} = -16 + 8 + 2 + 0.75 = -5.25_{10}$$

SOMMA BINARIA NEL SISTEMA BINARIO

Somma: si esegue la somma tra i bit di pari ordine

La somma è definita su 3 elementi:

- due addendi
- il riporto (carry)

La somma di 2 unità (valore 1) di un dato ordine, creano 1 unità dell'ordine immediatamente superiore (carry).

ADDENDO #1	ADDENDO #2	RIPORTO	SOMMA
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

- 0 + 0 = 0
- 0 + 1 = 1
- 1 + 0 = 1
- 1 + 1 = 0 con riporto 1 sul bit di ordine superiore
- 1 + 1 + 1 = 1 con riporto 1 sul bit di ordine superiore

Serve un bit in più!

SOMMA BINARIA NEL SISTEMA BINARIO: ESEMPIO

Si esegua la somma tra 010011 e 010001.

$010011_2 = 19_{10}$

$010001_2 = 17_{10}$

RIPORTO						
PRIMO ADDENDO	0	1	0	0	1	1
SECONDO ADDENDO	0	1	0	0	0	1
SOMMA						

SOMMA BINARIA NEL SISTEMA BINARIO: ESEMPIO

Si esegua la somma tra 010011 e 010001.

$010011_2 = 19_{10}$

$010001_2 = 17_{10}$

RIPORTO					1	
PRIMO ADDENDO	0	1	0	0	1	1
SECONDO ADDENDO	0	1	0	0	0	1
SOMMA						0

SOMMA BINARIA NEL SISTEMA BINARIO: ESEMPIO

Si esegua la somma tra 010011 e 010001.

$010011_2 = 19_{10}$

$010001_2 = 17_{10}$

RIPORTO				1	1	
PRIMO ADDENDO	0	1	0	0	1	1
SECONDO ADDENDO	0	1	0	0	0	1
SOMMA					0	0

SOMMA BINARIA NEL SISTEMA BINARIO: ESEMPIO

Si esegua la somma tra 010011 e 010001.

$010011_2 = 19_{10}$

$010001_2 = 17_{10}$

RIPORTO			0	1	1	
PRIMO ADDENDO	0	1	0	0	1	1
SECONDO ADDENDO	0	1	0	0	0	1
SOMMA				1	0	0

SOMMA BINARIA NEL SISTEMA BINARIO: ESEMPIO

Si esegua la somma tra 010011 e 010001.

$010011_2 = 19_{10}$

$010001_2 = 17_{10}$

RIPORTO		0	0	1	1	
PRIMO ADDENDO	0	1	0	0	1	1
SECONDO ADDENDO	0	1	0	0	0	1
SOMMA			0	1	0	0

SOMMA BINARIA NEL SISTEMA BINARIO: ESEMPIO

Si esegua la somma tra 010011 e 010001.

$010011_2 = 19_{10}$

$010001_2 = 17_{10}$

RIPORTO	1	0	0	1	1	
PRIMO ADDENDO	0	1	0	0	1	1
SECONDO ADDENDO	0	1	0	0	0	1
SOMMA	1	0	0	1	0	0

$100100_2 = 36_{10}$

SOTTRAZIONE BINARIA NEL SISTEMA BINARIO

Sottrazione: si esegue la differenza tra i bit di pari ordine

Anche la sottrazione opera su gruppi di 3 bit:
- minuendo e sottraendo
- prestito (borrow) proveniente dalla cifra di ordine immediatamente superiore

MINUENDO	SOTTRAENDO	PRESTITO	RESTO
0	0	0	**0**
0	1	1	**1**
1	0	0	**1**
1	1	0	**0**

- 0 – 0 = 0
- 0 – 1 = 1 con prestito dal bit di ordine immediatamente superiore
- 1 – 0 = 1
- 1 – 1 = 0

***Nota**: il prestito viene chiesto alla **cifra di ordine superiore che vale due unità** della cifra di ordine inferiore*

SOTTRAZIONE BINARIA NEL SISTEMA BINARIO: ESEMPIO

Si esegua la differenza tra 11101 e 01110.

$11101_2 = 29_{10}$

$01110_2 = 14_{10}$

PRESTITO						
MINUENDO		1	1	1	0	1
SOTTRAENDO		0	1	1	1	0
DIFFERENZA						

SOTTRAZIONE BINARIA NEL SISTEMA BINARIO: ESEMPIO

Si esegua la differenza tra 11101 e 01110.

$11101_2 = 29_{10}$

$01110_2 = 14_{10}$

PRESTITO						
MINUENDO		1	1	1	0	1
SOTTRAENDO		0	1	1	1	0
DIFFERENZA						1

SOTTRAZIONE BINARIA NEL SISTEMA BINARIO: ESEMPIO

Si esegua la differenza tra 11101 e 01110.

$11101_2 = 29_{10}$

$01110_2 = 14_{10}$

Prestito di una unità -> il minuendo sarà 0 e non 1

PRESTITO					2	
MINUENDO		1	1	**0**	0	1
SOTTRAENDO		0	1	1	1	0
DIFFERENZA					1	1

SOTTRAZIONE BINARIA NEL SISTEMA BINARIO: ESEMPIO

Si esegua la differenza tra 11101 e 01110.

$11101_2 = 29_{10}$

$01110_2 = 14_{10}$

Prestito di una unità -> il minuendo sarà 0 e non 1

PRESTITO				2	2	
MINUENDO		1	**0**	**0**	0	1
SOTTRAENDO		0	1	1	1	0
DIFFERENZA				1	1	1

SOTTRAZIONE BINARIA NEL SISTEMA BINARIO: ESEMPIO

Si esegua la differenza tra 11101 e 01110.

$11101_2 = 29_{10}$

$01110_2 = 14_{10}$

Prestito di una unità -> il minuendo sarà 0 e non 1

PRESTITO				2	2	
MINUENDO		1	**0**	1	0	1
SOTTRAENDO		0	1	1	1	0
DIFFERENZA				1	1	1

SOTTRAZIONE BINARIA NEL SISTEMA BINARIO: ESEMPIO

Si esegua la differenza tra 11101 e 01110.

$11101_2 = 29_{10}$

$01110_2 = 14_{10}$

Prestito di una unità -> il minuendo sarà 0 e non 1

PRESTITO			2	2	2	
MINUENDO		**0**	**0**	1	0	1
SOTTRAENDO		0	1	1	1	0
DIFFERENZA			1	1	1	1

SOTTRAZIONE BINARIA NEL SISTEMA BINARIO: ESEMPIO

Si esegua la differenza tra 11101 e 01110.

$11101_2 = 29_{10}$

$01110_2 = 14_{10}$

Prestito di una unità -> il minuendo sarà 0 e non 1

PRESTITO			2	2	2	
MINUENDO		**0**	1	1	0	1
SOTTRAENDO		0	1	1	1	0
DIFFERENZA			1	1	1	1

SOTTRAZIONE BINARIA NEL SISTEMA BINARIO: ESEMPIO

Si esegua la differenza tra 11101 e 01110.

$11101_2 = 29_{10}$

$01110_2 = 14_{10}$

Prestito di una unità -> il minuendo sarà 0 e non 1

PRESTITO			2	2	2	
MINUENDO		**0**	1	1	0	1
SOTTRAENDO		0	1	1	1	0
DIFFERENZA		0	1	1	1	1

$01111_2 = 15_{10}$

ALTRI ESEMPI DI SOMME E SOTTRAZIONI

Supponendo di **avere a disposizione 6 bit**, si calcoli la somma e la sottrazione in binario dei seguenti numeri: 010101_2 110100_2

SOMMA

RIPORTO						
PRIMO ADDENDO	0	1	0	1	0	1
SECONDO ADDENDO	1	1	0	1	0	0
SOMMA						

ALTRI ESEMPI DI SOMME E SOTTRAZIONI

Supponendo di **avere a disposizione 6 bit**, si calcoli la somma e la sottrazione in binario dei seguenti numeri: 010101_2 110100_2

SOMMA

Overflow!

RIPORTO	1		1			
PRIMO ADDENDO	0	1	0	1	0	1
SECONDO ADDENDO	1	1	0	1	0	0
SOMMA	0	0	1	0	0	1

il numero ottenuto non è rappresentabile in quanto il risultato è su 7 bit!

ALTRI ESEMPI DI SOMME E SOTTRAZIONI

Supponendo di **avere a disposizione 6 bit**, si calcoli la somma e la sottrazione in binario dei seguenti numeri: 010101_2 110100_2

SOTTRAZIONE

PRESTITO						
MINUENDO	0	1	0	1	0	1
SOTTRAENDO	1	1	0	1	0	0
DIFFERENZA						

ALTRI ESEMPI DI SOMME E SOTTRAZIONI

Supponendo di **avere a disposizione 6 bit**, si calcoli la somma e la sottrazione in binario dei seguenti numeri: 010101_2 110100_2

SOTTRAZIONE

PRESTITO						
MINUENDO	0	1	0	1	0	1
SOTTRAENDO	1	1	0	1	0	0
DIFFERENZA	???	0	0	0	0	1

Non è possibile eseguire l'operazione

ALTRI ESEMPI DI SOMME E SOTTRAZIONI

Perché non è possibile effettuare l'operazione?

① Per poter eseguire l'operazione avrei bisogno di chiedere in prestito un bit da un ordine superiore, ma avrei bisogno di 7 bit e non 6 bit.

② L'operazione $(010101 - 110100)_2$ sarebbe $(21 - 52)_{10} = -31_{10}$. Un numero negativo non è rappresentabile nel sistema binario puro.

SOTTRAZIONE

PRESTITO						
MINUENDO	0	1	0	1	0	1
SOTTRAENDO	1	1	0	1	0	0
DIFFERENZA	???	0	0	0	0	1

Non è possibile eseguire l'operazione

ALTRI ESEMPI DI SOMME E SOTTRAZIONI

Con il sistema numerico binario «base» non è possibile fare operazioni su numeri negativi.

Ma ci sono 3 modi rappresentazione dei numeri negativi:

- Modulo e Segno (MS)
- *Complemento a 1 (CA1)*
- Complemento a 2 (CA2)