UNIVERSITÀ DEGLI STUDI DI TRIESTE

Operazioni nel sistema binario

Prof.ssa Giulia Cisotto

giulia.cisotto@units.it

Trieste, 26 febbraio 2026

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)

OPERAZIONI CON NUMERI IN “MODULO E SEGNO”

Come si esegue la **somma** di due valori in Modulo e Segno?

- Confronto i bit di segno dei due numeri:

a) Se i bit di segno sono uguali:
  - Il bit di segno risultante sarà il bit di segno dei due addendi
  - Eseguo la somma bit a bit (a meno di overflow)

b) Se i bit di segno sono diversi:
  - Confronto i valori assoluti dei due addendi
  - Il bit di segno risultante sarà il bit di segno dell’addendo con valore assoluto maggiore
  - Eseguo la differenza bit a bit

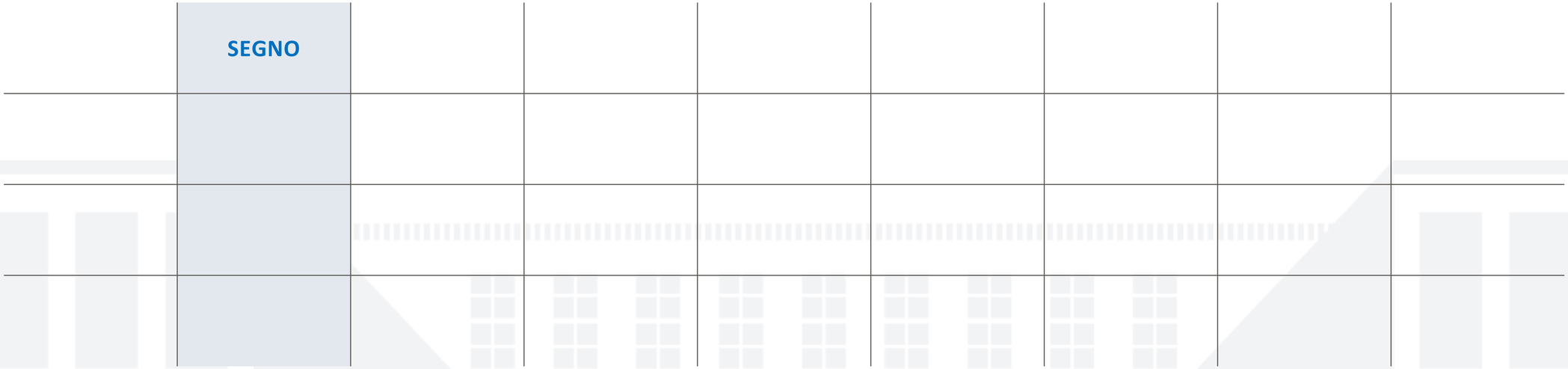
SOMMA CON NUMERI IN “MODULO E SEGNO”: ESEMPIO

Supponiamo di voler sommare **+42_{10} e -25_{10}** in **modulo e segno su 8 bit**.

	SEGNO							

SOMMA CON NUMERI IN “MODULO E SEGNO”: ESEMPIO

Supponiamo di voler sommare **+42_{10} e -25_{10}** in **modulo e segno su 8 bit**.



Rappresentare il modulo degli addendi → Estendere la rappresentazione a 8 bit, includendo il segno

42_{10} in binario (bastano 6 bit) → 101010

25_{10} in binario (bastano 5 bit) → 11001

+42_{10} → 0 **01**101010 ★

-25_{10} → 1 **00**11001

1) I bit di segno sono diversi

2) Il bit di segno risultante sarà il bit di segno dell’addendo con valore assoluto maggiore ★

3) Eseguo la differenza bit a bit

SOMMA CON NUMERI IN “MODULO E SEGNO”: ESEMPIO

Supponiamo di voler sommare **$+42_{10}$ e -25_{10}** in **modulo e segno su 8 bit**.

PRESTITO	SEGNO	0	-	2	-	-	-	2
MINUENDO		0	1	0	1	0	1	0
SOTTRAENDO		0	0	1	1	0	0	1
RESTO	0	**0**	**0**	**1**	**0**	**0**	**0**	**1**

→ **$+17_{10}$**

Rappresentare il modulo degli addendi → Estendere la rappresentazione a 8 bit, includendo il segno

42_{10} in binario (bastano 6 bit) → 101010

25_{10} in binario (bastano 5 bit) → 11001

$+42_{10}$ → 0 **01**01010★

-25_{10} → 1 **00**11001

1) I bit di segno sono diversi

2) Il bit di segno risultante sarà il bit di segno dell’addendo con valore assoluto maggiore★

3) Eseguo la ***differenza*** *bit a bit*

OPERAZIONI CON NUMERI IN “MODULO E SEGNO”

Come si esegue la **sottrazione** di due valori in Modulo e Segno?

- Confronto i bit di segno dei due numeri:

a) Se i bit di segno sono uguali:
  - Il bit di segno risultante sarà uguale al bit di segno dell’operando a modulo maggiore
  - Il risultato avrà modulo pari al modulo della differenza dei moduli degli operandi

b) Se i bit di segno sono diversi:
  - Il bit di segno risultante sarà *uguale al bit di segno del minuendo*
  - Il risultato avrà modulo pari alla somma dei moduli dei due operandi

Osservazione: $A - B = A + (-B)$

OPERAZIONI CON NUMERI IN “MODULO E SEGNO”

Osservazioni

Si può avere overflow solo quando:

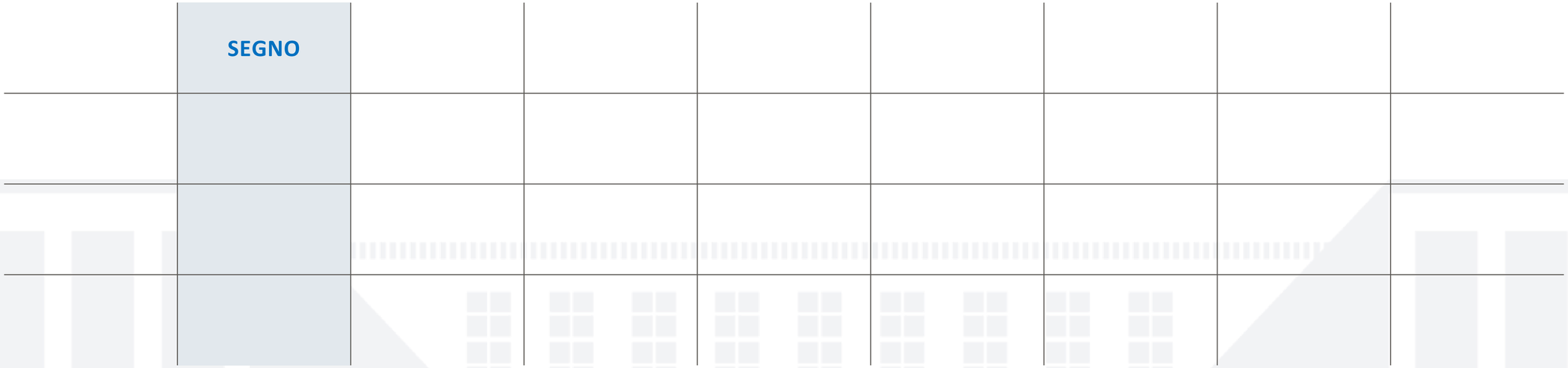
- si sommano due operandi con segno concorde
- si sottraggono due operandi con segno discorde

L’overflow si verifica quando c’è un riporto dalla cifra più significativa del modulo, cioè non si è nella condizione di rappresentare il risultato ottenuto.

Nelle operazioni tra valori rappresentati in MS, gli operandi devono essere rappresentati con lo stesso numero di cifre (si aggiungono gli zeri necessari a sinistra del modulo, prima del bit di segno).

SOTTRAZIONE CON NUMERI IN “MODULO E SEGNO”: ESEMPIO

Supponiamo di voler sottrarre **19_{10}** dal numero **-37_{10}** in **modulo e segno su 8 bit**.



Rappresentare il modulo degli operandi → Estendere la rappresentazione a 8 bit, includendo il segno

37_{10} in binario (bastano 6 bit) → 100101

19_{10} in binario (bastano 5 bit) → 10011

-37_{10} → 1 **0**100101★

19_{10} → 0 **00**10011

1) I bit di segno sono diversi

2) Il bit di segno risultante sarà *uguale al bit di segno del minuendo*★

3) Il risultato avrà modulo pari alla ***somma*** *dei moduli dei due operandi*

SOTTRAZIONE CON NUMERI IN "MODULO E SEGNO": ESEMPIO

Supponiamo di voler sottrarre **19_{10}** dal numero **-37_{10}** in **modulo e segno su 8 bit**.

RIPORTO	SEGNO	0	0	0	1	1	1	
ADDENDO 1	1	0	1	0	0	1	0	1
ADDENDO 2	0	0	0	1	0	0	1	1
SOMMA	**1**	**0**	**1**	**1**	**1**	**0**	**0**	**0**

→ **-56_{10}**

Rappresentare il modulo degli operandi → Estendere la rappresentazione a 8 bit, includendo il segno

37_{10} in binario (bastano 6 bit) → 100101

19_{10} in binario (bastano 5 bit) → 10011

-37_{10} → 1 **0**100101 ★

19_{10} → 0 **00**10011

1) I bit di segno sono diversi

2) Il bit di segno risultante sarà *uguale al bit di segno del minuendo* ★

3) Il risultato avrà modulo pari alla ***somma*** *dei moduli dei due operandi*

SOMMA CON NUMERI IN CA2

Si scarta il riporto perché stiamo lavorando in aritmetica ***modulo 2^N***

Come si esegue la **somma** di due valori in CA2 con N bit?

① Si esegue la somma su tutti i bit degli addendi, segno compreso

② Un eventuale riporto (carry) oltre il bit di segno (MSB) viene scartato

③ Nel caso gli operandi siano di segno concorde (entrambi positivi o entrambi negativi) occorre verificare la presenza o meno di overflow (il segno del risultato non è concorde con quello dei due addendi)

- *L'overflow non si presenta mai quando si sommano operandi di segno opposto.*
- L'overflow si presenta se il risultato ha segno discorde dagli operandi:

(+A)+(+B)=-C oppure (-A) + (-B) = +C

- C'è overflow se i riporti nelle ultime due posizioni più significative sono diversi.

NOTA IMPORTANTE

Gli operandi devono essere rappresentati con lo stesso numero di bit.

Nell'ipotesi di avere un valore X in CA2 su n bit (segno incluso) e di volerne ricavare la rappresentazione, sempre in CA2, su m bit (m>n), si attua l'**estensione del segno**:

si replica l'MSB negli (m – n) bit più a sinistra

Per i numeri positivi si aggiungono 0 nella parte più significativa

+18 = 00010010

+18 = 00000000 00010010

Per i numeri negativi si aggiungono 1 nella parte più significativa

-18 = 101110

-18 = 1111 101110

SOMMA CON NUMERI IN CA2: ESEMPIO

ESTENSIONE DEL SEGNO IN CA2

Si esegua la somma tra 3 e -8 su 8 bit.

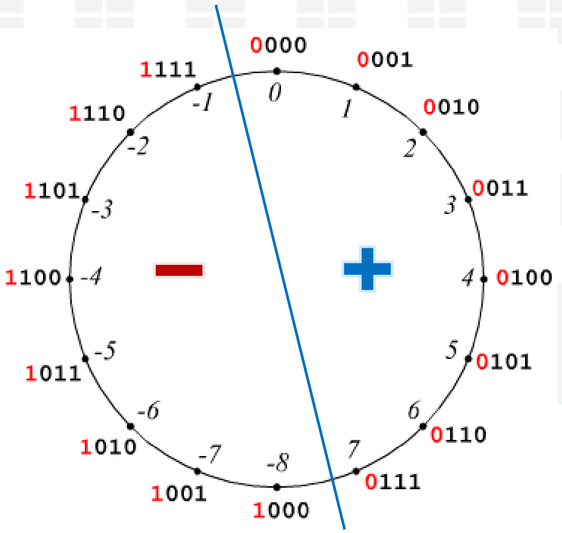
```
 (+3)   0000 0011
+(−8)   1111 1000
-----------------
```

Si esegua la somma tra -2 e -5 su 8 bit.

```
 (−2)   1111 1110
+(−5)   1111 1011
-----------------
    1   1111 1001  : scarto il carry
```

E' FINITA L'OPERAZIONE?

NOTA: **Non c'è overflow**, perché il risultato ha lo stesso segno (MSB=1) degli operandi.



7 bit → max modulo rappresentabile = $2^7 = 128$

$111\ 1001_2 = 64+32+16+8+1 = 121$

$1111\ 1001_2$ (CA2) = 128-121 = 7 → -7

Alternativa:

$1111\ 1001_2$ (CA2) = -128+64+…+1 = -7

SOTTRAZIONE CON NUMERI IN CA2

La sottrazione tra due numeri in CA2 viene trasformata in somma applicando la regola:

$$A - B = A + (-B)$$

ovvero:

$$A - B = A + \text{CA2}(B)$$

Per assicurarsi della correttezza del risultato di un'operazione di sottrazione in CA2 bisogna verificare l'assenza di overflow:

- *non* si ha overflow se gli operandi hanno segno concorde
- si ha overflow *se gli operandi hanno segno discorde* e il segno del risultato è discorde con essi

SOTTRAZIONE CON NUMERI IN CA2: ESEMPIO

Si esegua la sottrazione tra 8 e 5 su 8 bit

```
 (+8) 0000 1000                      0000 1000
-(+5) 0000 0101 -> Complementa -> +1111 1011
-----                               -----------
 (+3)                             X 0000 0011
```

Scartiamo il riporto perché lavoriamo in modulo 2^8

SOTTRAZIONE CON NUMERI IN CA2: ESEMPIO

Si esegua la sottrazione tra -6 e +3 su 4 bit

$6_{10} = 0110$

$\mathrm{CA2}(+6) = 1010$

$3_{10} = 0011$

$\mathrm{CA2}(+3) = 1101$

```
  1010   (-6)
+ 1101   (-3)
------------
 10111
```

$\mathrm{CA2}(0111) = +7$

OVERFLOW!

Il risultato dovrebbe essere **-9**, numero *non rappresentabile in CA2 con soli 4 bit.*

***Nota**: Se **controlliamo i bit di segno** dei due addendi (concordi) vediamo che il risultato ha segno diverso.*

RAPPRESENTAZIONE NUMERI INTERI NEGATIVI	RANGE DI VALORI RAPPRESENTABILI
CA2	$[-2^{N-1}, 2^{N-1} -1]_{10}$

Con 4 bit posso rappresentare [-8, +7] in CA2

OPERAZIONE DI SHIFT

Esiste un’ulteriore operazione detta shift:

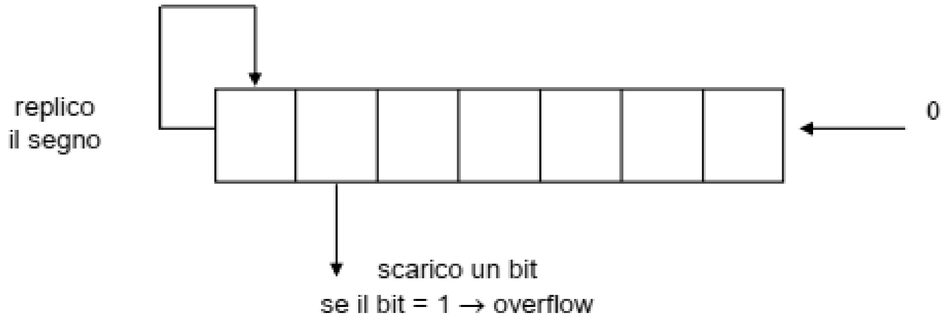
- Consiste nello spostare (shift) verso destra (right) o verso sinistra (left) la posizione delle cifre di un numero, espresso in una base qualsiasi, inserendo uno zero nelle posizioni lasciate libere.

- Left : equivale a moltiplicare il numero per la base

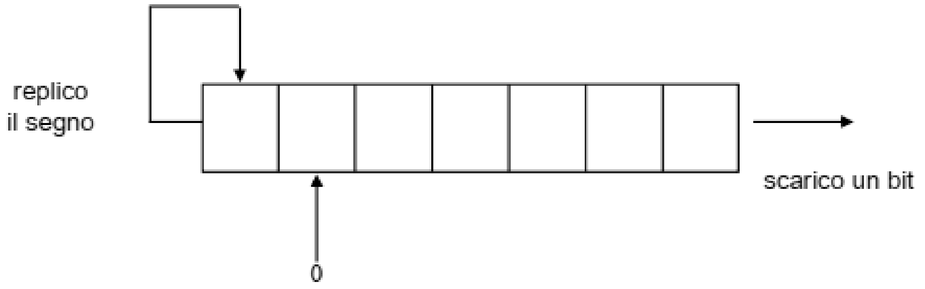
- Right : equivale a dividere il numero per la base

OPERAZIONE DI SHIFT – CON RAPPRESENTAZIONE “MS”

Left: equivale a moltiplicare il numero per la base.

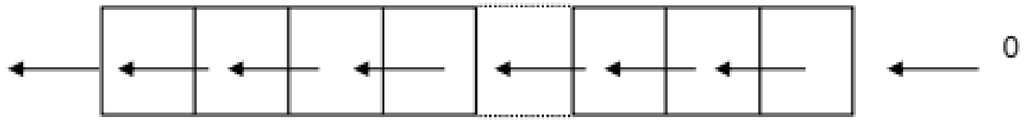


Right: equivale a dividere il numero per la base.



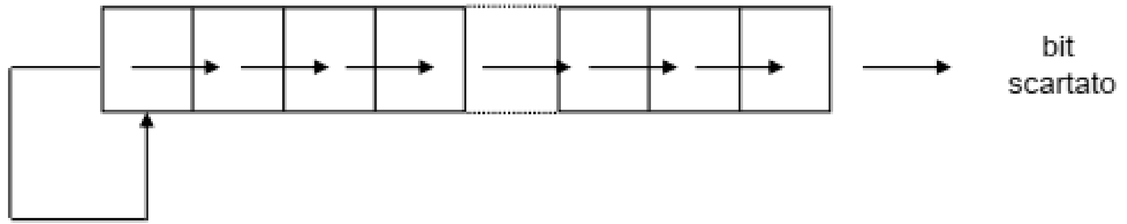
OPERAZIONE DI SHIFT – CON RAPPRESENTAZIONE “CA2”

Left: equivale a moltiplicare il numero per la base.



se il nuovo MSB è diverso
dal precedente c’è overflow

Right: equivale a dividere il numero per la base.



OPERAZIONE DI SHIFT: ESEMPIO

Si esegua uno shift verso destra di 1 (equivalente a /2) del numero in CA2:

$$0100\ 0010_2 = 66_{10}$$

$$00100\ 001_2 = 33_{10}$$

Materiale per la lezione

- Come per le lezioni precedenti

Prossima lezione: 3 ***marzo, h.14:00, aula 4C***