Numeri interi (+/-)

- **Alfabeto binario**
  - il segno è rappresentato da **0** (**+**) oppure **1** (**-**)
  - è indispensabile indicare il numero **k** di bit utilizzati
- **Modulo e segno**
  - **1** bit di segno (**0 positivo**, **1 negativo**)
  - **k – 1** bit di modulo
    - **Esempio:** $+6_{dieci} = 0110_{ms}$ $-6_{dieci} = 1110_{ms}$
  - si rappresentano i valori da $-2^{k-1}+1$ a $2^{k-1}-1$
    - con **4 bit** i valori vanno da $-7 (= -2^{4-1}+1)$ a $+7 (= 2^{4-1}-1)$
    - con **8 bit** i valori vanno da $-127 (= -2^{8-1}+1)$ a $+127 (= 2^{8-1}-1)$

Problema: ci sono due rappresentazioni dello 0 (ambiguità!)

con 4 bit avremo: $+0_{dieci} = 0000_{ms}$ **e** $-0_{dieci} = 1000_{ms}$

Diverse codifiche/interpretazioni

Codice	Nat	MS
0000	0	**+0**
0001	1	+1
0010	2	+2
0011	3	+3
0100	4	+4
0101	5	+5
0110	6	+6
0111	7	+7

Codice	Nat	MS
1000	8	**-0**
1001	9	-1
1010	10	-2
1011	11	-3
1100	12	-4
1101	13	-5
1110	14	-6
1111	15	-7

Numeri interi in “Complemento a 2”

Il complemento a due è il metodo più diffuso per la rappresentazione dei numeri interi (+/-) in informatica.

E’ stato inventato intorno al 1960 per evitare la presenza di “due zeri” (uno positivo e uno negativo) e per utilizzare tutte le possibili 2^n combinazioni che si possono ottenere con n bit.

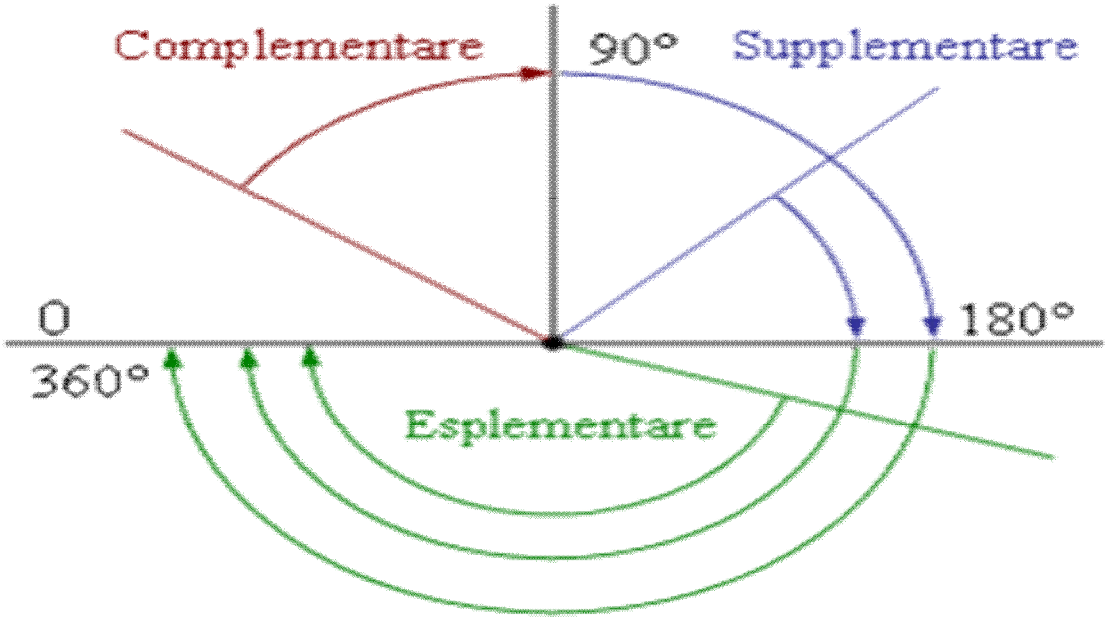
Il bit iniziale (più a sinistra) del numero indica il segno

+ (0) - (1)

mentre la parte restante indica il modulo

Il complemento in mat e geo

In geometria: si dice ***complementare*** di un angolo di ampiezza α ogni angolo avente l'ampiezza la ß "mancante" per ottenere un angolo retto:



In matematica: il numero X che manca a un altro numero Y per ottenere un terzo numero Z è detto "**complemento a Z di Y"**:
Esempio: il complemento a 7 di 4 é 3 (essendo 4+3=7) o, in altre parole, "**4 complemento a 7 è uguale a 3"**.

Complemento a 1 e Complemento a 2

C_1: Per complemento a 1 di un numero binario X, si intende quel numero (binario) Y che, sommato al numero (binario) di partenza X dia 1 come risultato:

per ottenere Y basta invertire o negare i bit di X

C_2: Per complemento a 2 di un numero binario X, si intende quel numero (binario) Y che, sommato al numero (binario) di partenza X dia 0 come risultato:

per ottenere Y basta aggiungere 1 al Complemento a 1 di X

Numeri interi in Complemento a 2

Esempio: rappresentiamo -5 con 4 bit in "Complemento a 2".
HINT: possiamo vedere -5 come il Complemento a 2 di +5

Si scrive innanzitutto la rappresentazione binaria del numero +5:

$$+5_{10} = 0101_2$$

Si scrive il "**complemento a 1**" del binario così ottenuto
si invertono cioè i bit (0 diventa 1 e 1 diventa 0), quindi:

$$1010_{C1}$$

Per ottenere il "**complemento a due**" di +5 aggiungiamo 1 al C1:

$$1011_{C2} = -5$$

Osserva: $(0101)_{C2} + (1011)_{C2} = 10000$ (il primo bit a sx è cancellato)

complemento a 2: un altro esempio

calcola la rappresentazione di –X a partire da quella di X e viceversa

Dato X procedere al complemento di ogni bit di X (cioè il C1 di X) , poi aggiungere 1 (cioè il C2 di X)

- rappresentazione di $+6_{dieci} = 0110_{C2}$
  (NB ci vogliono 4 bit!!)
- Complemento (a 1) di tutti i bit $\Rightarrow 1001_{C2}$
  (corrisponderebbe a -7_{dieci})
- aggiungere 1 $\Rightarrow 1010_{C2}$
  (che corrisponde a -6_{dieci})

Numeri interi in *Complemento a 2*

- **Alfabeto binario**
  - il segno è rappresentato sempre da 0 (+) o 1 (-)
  - è indispensabile indicare il numero **k** di bit utilizzati
- **Complemento a 2 (metodo diretto)**
  - **X** corrisponde alla versione binaria del naturale $2^k + X$

    $+6_{dieci} \Rightarrow 2^4+6 = 22 \Rightarrow [1]0110 \Rightarrow 0110_{C2}$ **(cancelliamo [1])**

    $-6_{dieci} \Rightarrow 2^4- 6 = 10 \Rightarrow [0]1010 \Rightarrow 1010_{C2}$ **(cancelliamo [0])**
  - si rappresentano i valori che vanno da -2^{k-1} a $2^{k-1}-1$
    - con 4 bit i valori vanno da –8 a +7
    - con 8 bit i valori vanno da –128 a +127
    - Con 32 bit i valori vanno da –2'147'483'648 fino a +2'147'483'647
  - **Attenzione: c'è una sola rappresentazione dello 0**
    - con 4 bit : $+0_{dieci} = 0000_{C2}$ mentre $1000_{C2} = -8_{dieci}$

Il complemento a 2

Ancora un altro metodo indiretto per calcolare

la rappresentazione di -X a partire da quella di X e viceversa

Partendo da destra e andando verso sinistra, si lasciano invariati tutti i bit fino al primo 1 compreso, poi si complementano tutti gli altri bit.

Esempio:

- rappresentazione di $+6_{dieci}$ = $\mathbf{0110_{C2}}$ (NB ci vogliono 4 bit!!)
- gli ultimi due bit (_ _ **1 0**) rimangono invariati
- gli altri due bit vengono complementati:

$\mathbf{1010_{C2}}$ che corrisponde a -6_{dieci}

Complemento a 2 => Decimale

- Se si considera la notazione posizionale, si può notare che nella rappresentazione binaria in complemento a due il valore (decimale) si può ottenere associando alla cifra più significativa (a sinistra) un peso negativo mentre tutte le altre cifre mantengono il peso originario positivo.

- **Il valore decimale di un numero $c_{n-1}c_{n-2}...c_1c_0$ scritto in complemento a due è :**

$$-c_{n-1}\times 2^{n-1} + c_{n-2}\times 2^{n-2} + \ldots + c_1\times 2^1 + c_0\times 2^0$$

- **Esempi:**
  - $0101_{C2} = -0\times 2^3+1\times 2^2+0\times 2^1+1\times 2^0 = +5_{dieci}$
  - $1011_{C2} = -1\times 2^3+0\times 2^2+1\times 2^1+1\times 2^0 = -5_{dieci}$

Complemento a 2: alcune osservazioni

- **I valori positivi iniziano con 0, quelli negativi con 1**
- **Data la rappresentazione di un numero su k bit, la rappresentazione dello stesso numero su k+1 bit si ottiene aggiungendo (a sinistra) un bit uguale al primo ("estensione del segno")**
  - Rappresentazione di –6 su 4 bit = **1010**
  - Rappresentazione di –6 su 5 bit = **11010**
  - Rappresentazione di –6 su 8 bit = **11111010**
- **la sottrazione si effettua come somma algebrica**
  - 4 – 6 = +4 + (–6) = 0100 + 1010 = 1110 = –2
  - 9 – 6 = +9 + (–6) = 01001 + 11010 = [1]00011 = +3

C2: rappresentazioni con diversi #bit

Naturale in base dieci	Complemento a due (# bit)					
	2	3	4	5	8	16
–13	NA	NA	NA	1 0011	1111 0011	1111 1111 1111 0011
–8	NA	NA	1000	1 1000	1111 1000	1111 1111 1111 1000
–7	NA	NA	1001	1 1001	1111 1001	1111 1111 1111 1001
–4	NA	100	1100	1 1100	1111 1100	1111 1111 1111 1100
–1	11	111	1111	1 1111	1111 1111	1111 1111 1111 1111
+1	01	001	0001	0 0001	0000 0001	0000 0000 0000 0001
+4	NA	NA	0100	0 0100	0000 0100	0000 0000 0000 0100
+7	NA	NA	0111	0 0111	0000 0111	0000 0000 0000 0111
+8	NA	NA	NA	0 1000	0000 1000	0000 0000 0000 1000
+13	NA	NA	NA	0 1101	0000 1101	0000 0000 0000 1101

C2: operazioni

- Operazioni di somma di numeri binari in complemento a 2
- Con gli **8 bit** utilizzati negli esempi qui riportati si possono rappresentare i numeri interi **da -128_{dieci} fino a $+127_{dieci}$**

$$\begin{array}{r} {\scriptstyle 1\,1} \\ 00011001_{C2}\,+ \\ 00111100_{C2}\,= \\ \hline 01010101_{C2} \end{array} \qquad \begin{array}{r} +25_{dieci}\,+ \\ +60_{dieci}\,= \\ \hline +85_{dieci} \end{array}$$

$$\begin{array}{r} {\scriptstyle 1} \\ 00101100_{C2}\,+ \\ 10110010_{C2}\,= \\ \hline 11011110_{C2} \end{array} \qquad \begin{array}{r} +44_{dieci}\,+ \\ -78_{dieci}\,= \\ \hline -34_{dieci} \end{array}$$

$$\begin{array}{r} {\scriptstyle 1\,11} \\ 11100111_{C2}\,+ \\ 11000100_{C2}\,= \\ \hline \cancel{1}10101011_{C2} \end{array} \qquad \begin{array}{r} -25_{dieci}\,+ \\ -60_{dieci}\,= \\ \hline -85_{dieci} \end{array}$$

$$\begin{array}{r} {\scriptstyle 1\,11\,1\,1} \\ 11010100_{C2}\,+ \\ 01001110_{C2}\,= \\ \hline \cancel{1}00100010_{C2} \end{array} \qquad \begin{array}{r} -44_{dieci}\,+ \\ +78_{dieci}\,= \\ \hline +34_{dieci} \end{array}$$

C2: operazioni e overflow

Operazioni di somma di numeri binari in complemento a 2
Con gli 8 bit utilizzati negli esempi qui riportati si possono rappresentare i numeri interi da -128_{dieci} fino a $+127_{dieci}$

- **Se due operandi dello stesso segno danno un risultato di segno opposto vuol dire che è stata superata la capacità di calcolo** (*overflow*).
- Banalmente, con due operandi di segno opposto l'overflow non può mai verificarsi

Diverse codifiche/interpretazioni

Dieci	Due	MS	C2
+0	0000	0000	0000
+1	0001	0001	0001
+2	0010	0010	0010
+3	0011	0011	0011
+4	0100	0100	0100
+5	0101	0101	0101
+6	0110	0110	0110
+7	0111	0111	0111
+8	1000	NA	NA

Dieci	Due	MS	C2
–0	0000	1000	0000
–1	NA	1001	1111
–2	NA	1010	1110
–3	NA	1011	1101
–4	NA	1100	1100
–5	NA	1101	1011
–6	NA	1110	1010
–7	NA	1111	1001
–8	NA	NA	1000

Diverse codifiche/interpretazioni

Codice	Nat	MS	C2
0000	0	0	0
0001	1	1	1
0010	2	2	2
0011	3	3	3
0100	4	4	4
0101	5	5	5
0110	6	6	6
0111	7	7	7

Codice	Nat	MS	C2
1000	8	-0	-8
1001	9	-1	-7
1010	10	-2	-6
1011	11	-3	-5
1100	12	-4	-4
1101	13	-5	-3
1110	14	-6	-2
1111	15	-7	-1