Rappresentazione in complemento a 2: caratteristiche generali

- La rappresentazione non è completamente posizionale, ma in parte sì.
- Guardando il bit più significativo (MSB) si capisce se il numero è positivo (o nullo) o negativo.
- Lo zero ha un'unica rappresentazione (costituita da tutti bit nulli).

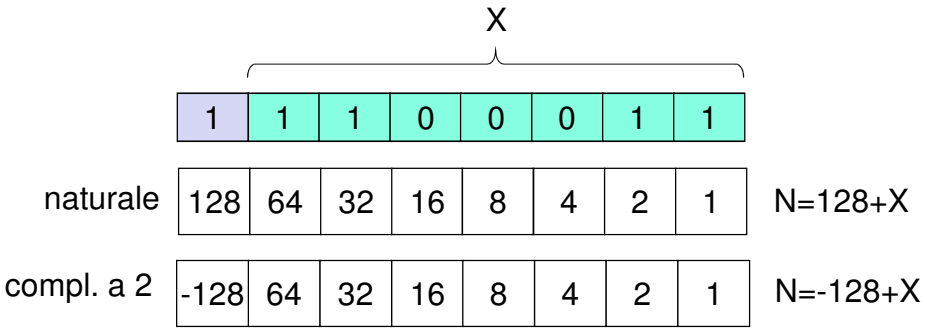
Rappresentazione in complemento a 2 di un numero positivo dato

- Rappresentiamo su k bit il numero N:
- Se $N \geq 0$ basta assegnare al MSB il valore 0 e codificare N sui rimanenti k-1 bit.
  NB: questo implica che sono rappresentabili i numeri fino a $2^{k-1}-1$
- Esempio su 8 bit (k=8):
  N= +96; Cp_2 (+96) = 01100000
  Prova: $01100000_2 = (64+32)_{10} = 96_{10}$

	0	1	1	0	0	0	1	1
base 2 naturale	**128**	64	32	16	8	4	2	1
complemento a 2	**-128**	64	32	16	8	4	2	1

Rappresentazione in complemento a 2 di un numero negativo dato

- Un numero negativo rappresentato in complemento a due comincia sempre per 1
  - Di posizione con valore negativo ce n'è una sola: se in corrispondenza c'è uno zero, il numero non può essere negativo.



Rappresentazione in complemento a 2 di un numero negativo dato

- Dato N≥0, vogliamo rappresentare –N in complemento a 2 mediante una stringa S di 8 bit.
- La stringa S interpretata come numero in compl. a 2 vale -128+X = -N
  - Quindi, X=128-N
- La stringa S interpretata come numero naturale vale 128+X, cioè 128+128-N = 256-N
- In conclusione, la stringa S che denota –N in complemento a 2 è la stessa che denota 256-N in binario.
- Esempio:

  -N= -56

  256 – 56 = 200 = 128+64+8 → 11001000

  Cp_2 (-56) = 11001000

  Prova: l'interpretazione in compl. a 2 di 11001000 è -128+64+8 = -56

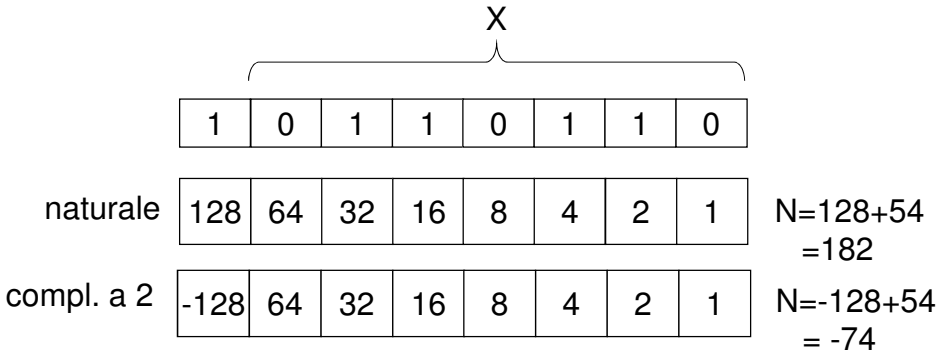
Rappresentazione in complemento a 2 di un numero negativo dato

-N = -74

-128+X = -74

X = 128-74 = 54 = 0110110

256-74 =182



Notazione in complemento a 2 caratteristiche ed esempi

- MSB = 0 → numero positivo
  - stesso valore che si avrebbe in binario puro: il diverso peso del MSB non ha influenza
- MSB = 1 → numero negativo
  - il valore si ottiene sommando il contributo negativo del MSB con i contributi positivi degli altri bit
- Esempi
  - 11010111 → -128 + 64 + 16 + 4 + 2 + 1 = -41
  - 00110011 → 32 + 16 + 2 + 1 = 51
  - 10000000 → -128 + 0 -128
  - 11111111 → -128+64+32+16+8+4+2+1 = -1
  - 00000000 denota il valore 0, cioè 0
  - 01111111 → 64+32+16+8+4+2+1 = 127.

Notazione in complemento a 2: proprietà

- Il massimo intero positivo rappresentabile è di uno inferiore al minimo negativo rappresentabile perché lo 0 viene considerato parte dei positivi.
- Lo zero ha ora un'unica rappresentazione → efficienza, semplicità

Complemento a 2: altri esempi

- Con k = 8 la rappresentazione in complemento a 2 di -1 si ottiene rappresentando in binario su 8 bit il valore 255, infatti:
  $Cp_2(-1) = 2^8 - 1 = 256-1 = 255 \rightarrow 11111111$
- In generale il valore -1 si rappresenta sempre con tutti i bit uguali a uno, indipendentemente da k.

Notazione in complemento a 2 caratteristiche ed esempi

- Valori opposti, come 17 e -17, hanno rappresentazioni diverse

  17 → 00010001

  -17 → 11101111

- Rappresentazioni identiche a meno del MSB denotano valori interi completamente diversi

  01010101 → 85

  11010101 → -128+85 = -43

Notazione in complemento a 2: intervallo di valori rappresentabili

- MSB=0 → k-1 bit usabili come in binario puro → intervallo da 0 a 2^{k-1}-1
  MSB=1 → stesso intervallo traslato di -2^{k-1} → intervallo da -2^{k-1} a -1
- Complessivamente è rappresentabile l'intervallo da -2^{k-1} a 2^{k-1}-1
- Esempio:
- k=8 bit → 2^8 = 256 combinazioni → intervallo da -2^7 a 2^7-1 cioè -128..127
  - anziché, come in binario puro, 0..255
- 256 combinazioni allocate metà ai positivi e metà ai negativi, anziché tutte ai positivi

Notazione in complemento a 2: proprietà

- Le somme algebriche si possono eseguire con le stesse regole dell'aritmetica binaria
  - La sottrazione si può eseguire banalmente complementando e sommando: a-b = a+(-b)
- È verificata la proprietà X + (-X) = 0 (trascurando il riporto)
- Non è necessario alcun circuito particolare per trattare i negativi → semplicità e basso costo.

Rappresentazione in complemento a 2: facilità d'uso

- Il problema principale della rappresentazione in modulo e segno è dato dalla non applicabilità degli algoritmi che funzionano per i numeri interi positivi.
- La rappresentazione in complemento a 2 permette di utilizzare direttamente gli algoritmi dei numeri naturali per eseguire le operazioni.
- In particolare è verificata la proprietà X + (-X) = 0 (usando le regole dell'aritmetica binaria senza segno).

$$X + (-X) = X + (2^k - X) = 2^k$$

- Ma 2^k non è rappresentabile con k bit: grazie all'overflow si ottenono k zeri.
- Per questo motivo la rappresentazione in complemento a 2 è molto diffusa.

Rappresentazione in complemento a 2 di un numero negativo dato

- Generalizzando alla rappresentazione su k bit:
  - per i negativi: $Cp_2(-N) = \text{binario}(2^k - N)$
  - per i positivi: $Cp_2(+N) = \text{binario}(N)$

  (con N>0)
- Es, con k = 8 bits:
  $Cp_2(-39) = \text{binario}(256 - 39) = \text{binario}(217) \rightarrow 11011001$
- Metodo alternativo: sottrarre da 0.

  -39 = 0 - (+39)

```
-1 -1 -1 -1 -1 -1 -1 -1
 0  0  0  0  0  0  0  0 -
 0  0  1  0  0  1  1  1 =
---------------------------
 1  1  0  1  1  0  0  1
```


Rappresentazione in complemento a 2 di un numero negativo dato

- Altro esempio (8 bit)
- Cp2(-47) = 256-47 = 209 = 11010001

47 +	00101111 +
(-47) =	11010001 =
0	(1)00000000 = 0

Rappresentazione in complemento a 2: operazione cambio di segno

- Dato un numero X (pos o neg) in Cp2, come posso cambiare di segno?
- cioè: dato Cp2(X), voglio trovare Cp2(-X)

- Se X è positivo:
  - Per definizione Cp2 (-X) = binario(2^n-X) = binario((2^n-1) – (X-1))
  - ma (2^n-1) è una sequenza di n bit : 11111….
  - e la differenza tra n uni e un numero binario (X-1) più piccolo (rappresentato con n bit) equivale a invertire tutti i bit della rappresentazione di (X-1) -- (provare)
  - Dopo di che basta aggiungere 1 per ottenere 2^n-X = Cp2(-X)

- Quindi: algoritmo: (1) inverto tutti i bit (2) faccio +1
- funziona anche se X è negativo! (provare)

Rappresentazione in complemento a 2: operazione cambio di segno

- Il procedimento funziona anche al contrario, cioè se si applica ad un numero negativo (-X) invertendo i bit e sommando 1 si ottiene la rappresentazione del numero positivo (+X).

- Il procedimento funziona anche per lo zero: sommando uno a una parola di tutti uni si ottiene 0 (trascurando il riporto).

Rappresentazione in complemento a 2: operazione cambio di segno

- Esempio (con 8 bits)

rappresentazione in cp2 di +92

0 1 0 1 1 1 0 0
→ 1 0 1 0 0 0 1 1 (flip di tutti i bit)
→ 1 0 1 0 1 1 0 0 (somma di +1)

rappresentazione in cp2 di -92

- E ritorno, con lo stesso algoritmo:

1 0 1 0 1 1 0 0
→ 0 1 0 1 0 0 1 1 (flip di tutti i bit)
→ 0 1 0 1 1 1 0 0 (aggunta di +1)

Rappresentazione in complemento a 2 di un numero negativo dato

Il metodo piú semplice per trovare la cp(-X) di un numero negativo -X (X positivo):

1. Trovare cp(X) , cioè binario(X)
   …e invertirne il segno, cioè:
2. Invertire tutti i bit di tale rappresentazione,
   («farne il complemento a 2», è questo che dà il nome alla notazione!)
3. Aggiungere 1 al risultato così ottenuto.

- Esempio: determinazione della rappresentazione di -25 (su k=8 bit)
  1) 25 → 00011001
  2) flip di segno → 11100110
  3) incremento di 1 → 11100111
- Verifica: -128 + 64 + 32 + 4 + 2 + 1 = -128 + 103 = -25

Notazione in complemento a 2
Overflow

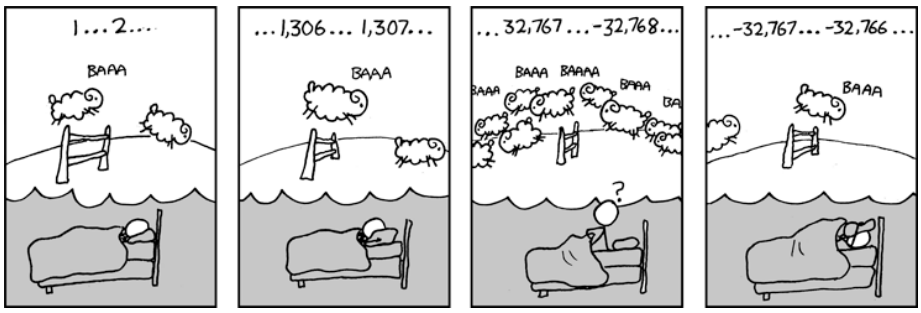
- Consideriamo i casi seguenti:

-73	10110111	+73	01001001
-73	10110111	+73	01001001
------	----------	------	----------
-146	(1)01101110	+146	10010010

- Il risultato è sbagliato: la prima stringa denota +110, la seconda –110.
- Il motivo è che –146 e +146 sono fuori dall'intervallo rappresentabile con 8 bit (-128 ... 127).
- L'overflow si riconosce dal fatto che sommando due numeri positivi otteniamo un negativo e viceversa.

Overflow

- Sommando due valori che danno come risultato un valore esterno all'intervallo rappresentabile $[-2^{n-1}... 2^{n-1}-1]$ si provoca l'invasione del bit di segno da parte del risultato
- In questo caso si ha overflow: il numero di bit non è sufficiente per la rappresentazione del numero.
- Una regola pratica per vedere se si ha overflow:
  - operazioni tra numeri di segno diverso sono sempre corrette in quanto non si può uscire dal range
  - operazioni tra numeri dello stesso segno sono corrette se il risultato mantiene il segno degli addendi.



https://xkcd.com/571/