

CODIFICA IN COMPLEMENTO A 2 (CA2)

Consente di codificare i numeri interi sia positivi che negativi.

In matematica il segno meno è una convenzione che indica un numero negativo. In una memoria digitale si possono memorizzare solamente dei valori 1 o 0 quindi bisogna trovare una convenzione fra noi ed il calcolatore in modo che entrambi sappiamo quando un numero è negativo o positivo. Bisogna cercare una convenzione. La convenzione è quella specificata nella **codifica (o rappresentazione) in complemento a due**. Questa codifica si basa su due cose, una convenzione e una regola.

CONVENZIONE:

IN CA2 (COMPLEMENTO A 2) UN NUMERO NEGATIVO HA SEMPRE IL BIT PIU' SIGNIFICATIVO = 1

REGOLA:

DATO UN NUMERO RAPPRESENTATO IN CA2, IL SUO OPPOSTO SI OTTIENE INVERTENDO GLI 1 CON GLI 0 E POI SOMMANDO 1 AL NUMERO OTTENUTO.

Esempio:

$$\begin{array}{rcl} 00000101 & \rightarrow & 5 \\ 11111010+ & & \\ 1 & = & \\ \hline 11111011 & \rightarrow & -5 \end{array}$$

Esercizio: Indicare il negativo dei seguenti numeri binari in CA2

00011101	(29)	→	Soluzione:	11100011	(-29)
00011110	(30)	→	Soluzione:	11100010	(-30)
01000001	(65)	→	Soluzione:	10111111	(-65)

Quando mi trovo davanti un numero binario in CA2, COME SI PUO' SAPERE SE UN NUMERO E' POSITIVO O NEGATIVO?

Soluzione:

SE IL BIT PIU' SIGNIFICATIVO VALE 0 ALLORA E' POSITIVO ALTRIMENTI E' NEGATIVO

COME SI FA A CAPIRE QUANTO E' IL VALORE DECIMALE RAPPRESENTATO IN CA2?

Soluzione, si seguono i seguenti due passi

1. SI GUARDA IL BIT PIU' SIGNIFICATIVO, ESSO CI DA L'INFORMAZIONE CHE IL NUMERO E' POSITIVO (SE VALE 0) O NEGATIVO (SE VALE 1)
2. PER TROVARE IL MODULO (OSSIA IL VALORE A MENO DEL SEGNO)
 - a. SE IL NUMERO E' POSITIVO SI CONVERTE LA SEQUENZA BINARIA IN DECIMALE
 - b. SE IL NUMERO CA2 E' NEGATIVO SI DETERMINA L'OPPOSTO IN E SUCCESSIVAMENTE SI CONVERTE LA SEQUENZA BINARIA IN DECIMALE.

Esempio :

Dire se i seguenti numeri rappresentati in CA2 ad 8 bit sono positivi o negativi

00011001

10010010

Esempio :

Convertire in decimale i numeri binari dell'esempio precedente

00011001 è positivo, basta fare la conversione BINARIO/DECIMALE → 25

10010010 è negativo:

 prima scrivo il segno meno -

 poi faccio il CA2 del numero di partenza 01101110

 Converto in decimale così trovo il valore assoluto 110₍₁₀₎

Quindi risultato: -110₁₀

Esercizio:

Convertire in decimale i seguenti numeri binari rappresentati in CA2 (negativi)

11100011	(Risultato:	CA2→ 00011101	DECIMALE→ - 29)
11100010	(Risultato:	CA2→00011110	DECIMALE→ -30)
10111111	(Risultato:	CA2→01000001	DECIMALE → -65)

Caratteristiche della rappresentazione in CA2:

CONSIDERANDO DI AVERE A DISPOSIZIONE N BIT:

- c. IL MASSIMO NUMERO POSITIVO RAPPRESENTABILE VALE $(2^{N-1} - 1)_{10}$
- d. IL MINIMO NUMERO NEGATIVO RAPPRESENTABILE VALE $(-2^{N-1})_{10}$
- e. QUINDI LA QUANTITA' DI NUMERI RAPPRESENTABILI E' $(2^N)_{10}$

Esercizio

1. Qual'è il l'estremo inferiore dei numeri in base dieci (negativo) rappresentabile con 8 bit in CA2?
2. Qual'è il l'estremo superiore in base dieci (positivo) rappresentabile con 8 bit in CA2?
3. Rappresentare tali valori in CA2

soluzione

1. l'estremo inferiore è: $-2^{(8-1)} = -128$
2. l'estremo superiore è: $2^{(8-1)} - 1 = 127$

3. Per rappresentare l'estremo inferiore si fa il CA2 di 128

128 → 10000000
CA2 → 01111111+
 1
 10000000

Per rappresentare l'estremo superiore si converte semplicemente in binario il numero 127_{10}

127 → 01111111

Somma e sottrazione in CA2

La rappresentazione in CA2 consente di eseguire le operazioni di somma e di sottrazione. Per eseguire una somma si seguono le solite regole della somma binaria (siamo già capaci).

Per eseguire la sottrazione fra due numeri si utilizza il "trucco" di eseguire la somma fra il primo ed il CA2 del secondo.

E' la stessa cosa che si fa con i numeri decimali, infatti $10-5$ si ottiene come somma fra 10 e (-5) . Quindi anche la sottrazione viene svolta con le stesse regole della somma binaria.

Esempio:

Eseguire l'operazione $10-5_{(10)}$ in binario in CA2 ad 8 bit

l'operazione viene eseguita in binario convertendo in CA2 il numeri 10 e -5 e poi sommandoli con la somma binaria.

10 → 00001010
5 → 00000101 quindi -5 → 11111011

00001010 +
11111011 =
00000101 verifico convertendo in base 10 il risultato → 5 esatto!

Esercizi:

eseguire in binario in CA2 ad 8 bit le seguenti operazioni

$32 - 12$ (risultato 20 → 00010100)

$(-12) + 10$ (risultato 2 → 00000010)

OSSERVAZIONE IMPORTANTE: NELLE OPERAZIONI DI ADDIZIONE IN CA2 SI TRASCURA UN EVENTUALE RIPORTO FRA I DUE BIT PIU' SIGNIFICATIVI (QUELLO FRA I DUE MSB)

Esempio: eseguire in binario la seguente operazione in CA2 ad 8 bit $18_{10} - 12_{10}$

$$\begin{array}{rcl} 18 \rightarrow & 00010010 & \\ 12 \rightarrow & 00001100 & -12 \rightarrow 11110011 + 1 = 11110100 \\ & & \\ & 00010010+ & \\ & 11110100= & \\ \hline & 00000110 & \rightarrow 6_{10} \end{array}$$

Abbiamo visto che nelle memorie ci sono dei limiti ai numeri rappresentabili (con 8 bit in CA2 il massimo è 127).

Cosa succede se in una memoria a 8 bit cerco di memorizzare il risultato della somma di due numeri il cui risultato è maggiore dei numeri rappresentabili?

In tal caso il risultato "non ci sta" nella memoria, si dice che si è verificato un **OVERFLOW (TRABOCCAMENTO)** della memoria e non è quindi possibile rappresentare il risultato.

Esempio:

Eseguire la seguente operazione in binario in CA2 ad 8 bit $100 + 50_{10}$

Osserviamo cosa succede: la somma dovrebbe essere 150 ma poichè 150 "non ci sta" in 8 bit (perché il massimo numero rappresentabile con 8 bit in CA2 è 127) si otterrà un risultato non corretto a causa dell'overflow.

$$\begin{array}{rcl} 100 & \rightarrow & 01100100 + \\ 50 & \rightarrow & 00110010 = \\ & & \hline & & 10010110 \end{array} \quad \begin{array}{l} \text{(MSB=1, quindi è un numero negativo! Impossibile! C'è un errore! Il} \\ \text{risultato è sbagliato! Infatti si è verificato un } \mathbf{OF}) \end{array}$$

Prova con calcolatrice windows (2^{33219} OK, $2^{33220} \rightarrow \text{overflow}$)

OSSERVAZIONI IMPORTANTI

1. QUANDO IL RISULTATO DI UN'OPERAZIONE SUPERA I LIMITI DEI VALORI RAPPRESENTABILI SI DICE CHE SI E' VERIFICATO UN **OVERFLOW** (TRABOCCAMENTO, lo indichiamo con OF)
2. **NELLE OPERAZIONI** DI ADDIZIONE SVOLTE IN CA2 IL FATTO CHE GLI ULTIMI DUE RIPORTI SIANO DIVERSI **E' UN SEGNALE CHE SI E' VERIFICATO UN OVERFLOW**. Quindi ogni volta che viene eseguita una somma in CA2, il calcolatore verifica se gli ultimi due riporti sono diversi e se lo sono, poichè **SICURAMENTE** i bit ottenuti dell'operazione **non rappresentano il risultato corretto**, il calcolatore deve mostrare come risultato dell'operazione **"OF"**.

Esempi. Si eseguano le seguenti operazioni fra numeri binari in CA2 ad 8 bit:

- | | | | | |
|----|-----------------------|---------------|------------------------|--------------------------------------|
| 1. | $11001110 + 10111000$ | $\rightarrow$ | $(-50) + (-70) = -120$ | 10001000 |
| 2. | $11000010 + 10100111$ | $\rightarrow$ | $(-62) + (-89) = -151$ | OF ultimi due riporti diversi |

Esercizi.

1. RAPPRESENTARE IN COMPLEMENTO A 2 AD 8 BIT I SEGUENTI NUMERI DECIMALI

$127_{(10)}$	$\rightarrow$	01111111
$-12_{(10)}$	$\rightarrow$	11110100
$-127_{(10)}$	$\rightarrow$	10000001
$-128_{(10)}$	$\rightarrow$	10000000
$200_{(10)}$	$\rightarrow$	non rappresentabile

2. Date le seguenti coppie di numeri:

- a. Convertire ciascuna coppia di numeri in CA2 ad 8 bit
- b. eseguire la somma in CA2 indicando mostrando eventualmente l' **OF** (per mostrarlo si evidenzi che gli ultimi due riporti sono diversi)

50; -45

-20; 90

-20; -120 **(OF)**

CODIFICA IN ECCESSO N

Quando si vuole codificare i numeri relativi in eccesso a N si fa questo ragionamento:

- Contiamo quanti bit ci mette a disposizione il calcolatore per rappresentare i numeri e chiamiamo questo numero n (ad esempio 4 bit, quindi $n=4$)
- Contiamo quanti possibili combinazioni di 1 e 0 possiamo avere con n bit, tale valore è sempre 2^n (nel nostro esempio $2^4=16$)
- La metà del valore precedentemente calcolato è il numero "N" dell'"eccesso N", quindi nel nostro esempio

$$N=16/2=8$$

la nostra rappresentazione dei numeri relativi è in **eccesso 8**

Ricordiamoci N=8

Questo significa che, delle 2^n (16 nel nostro esempio) possibili combinazioni di bit che possiamo avere, le prime N (ossia le prime 8) le utilizziamo per rappresentare i numeri negativi, l' N-esima (l'ottava nel nostro esempio) rappresenta il valore 0₁₀, le successive N (ossia le successive 8) le utilizziamo per rappresentare i numeri positivi.

Vediamo tutte le combinazioni che possiamo avere nel nostro esempio:

0000	negativi	→	rappresenta in decimale il numero -8
0001		→	rappresenta in decimale il numero -7
0010		→	rappresenta in decimale il numero -6
0011		→	rappresenta in decimale il numero -5
0100		→	rappresenta in decimale il numero -4
0101		→	rappresenta in decimale il numero -3
0110		→	rappresenta in decimale il numero -2
0111		→	rappresenta in decimale il numero -1
1000	zero	→	rappresenta in decimale il numero 0
1001	positivi	→	rappresenta in decimale il numero 1
1010		→	rappresenta in decimale il numero 2
1011		→	rappresenta in decimale il numero 3
1100		→	rappresenta in decimale il numero 4
1101		→	rappresenta in decimale il numero 5
1110		→	rappresenta in decimale il numero 6
1111		→	rappresenta in decimale il numero 7

Si osservi che con la codifica eccesso N i numeri negativi hanno MSB=0 mentre i positivi hanno MSB=1, contrariamente alla codifica in CA2

Minimo valore decimale rappresentabile:	-N	(nell'esempio: -8)
Massimo valore rappresentabile:	N-1	(nell'esempio: 7)
Numero di valori rappresentabili	$2*N$	(nell'esempio: 16)

Algoritmo (procedura) per passare da numero decimale a codifica ECCESSO N:

1. Si aggiunge al numero decimale il valore N
2. Si converte il numero ottenuto in binario utilizzando tutti i bit a disposizione

esempio: per rappresentare il numero decimale -3 in ECCESSO 8

1. si aggiunge a -3 ad 8 → $-3+8=5$
2. Si converte il numero ottenuto (5) in binario → 0101

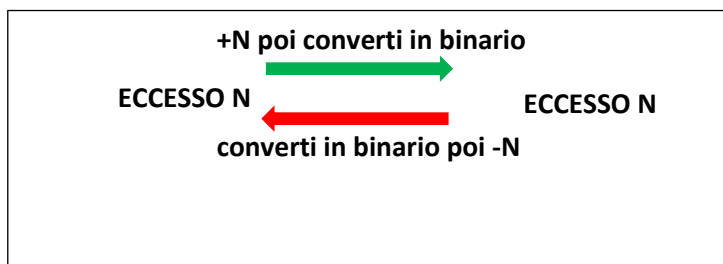
Algoritmo (procedura) per passare da numero codificato in ECCESSO N a numero decimale:

1. Si converte in decimale il numero in ECCESSO N
2. Si sottrae N al numero ottenuto

esempio: per convertire in decimale il numero 1001 in ECCESSO 8

1. Si converte in decimale il numero in 1001 $\rightarrow 9_{(10)}$
2. Si sottrae 8 al numero ottenuto $\rightarrow 9-8=1_{(10)}$

Schema mnemonico per ricordarsi:



Esercizi: fai esercizi p. 33 libro nuovo

Esercizi: Determinare l'opposto in Eccesso 128 dei seguenti numeri binari in eccesso 128 con 8 bit

- | | | | | | | |
|----|----------|-------|---------------|------------|----------|-------|
| 1) | 00011101 | (-99) | $\rightarrow$ | Soluzione: | 11100011 | (99) |
| 2) | 00110110 | (-74) | $\rightarrow$ | Soluzione: | 11001010 | (-74) |
| 3) | 10000111 | (7) | $\rightarrow$ | Soluzione: | 01111001 | (-7) |

Procedimento per esercizio 1:

Calcolo il valore decimale del numero in ECCESSO 128 applicando l'algoritmo:

00011101 $\rightarrow$ converto in decimale : 29 $\rightarrow$ sottraggo 128: -99

L'opposto di -99 è 99

Rappresento 99 in ECCESSO 128 applicando l'algoritmo

99 + 128 $\rightarrow$ 227 $\rightarrow$ converto in binario $\rightarrow$ 1110 0011

Risultato dell'esercizio: 11100011