Rappresentazioni

Ver. 1.6

Numeri relativi

- I valori binari senza segno sono espressi nella rappresentazione “binario puro”
- L’informazione relativa al segno in un numero binario deve essere comunque memorizzata nei suoi stessi bit
- Basta un solo bit: i segni sono 2
- Viene normalmente utilizzato il MSB

Rappresentazione in Modulo e Segno

- Per avere un insieme di valori negativi, il modo più semplice è cambiare il segno ai valori positivi corrispondenti
- Questo è il sistema utilizzato normalmente in notazione decimale
- L’insieme dei valori negativi è ottenuto “ribaltando” rispetto allo 0 l’insieme dei valori positivi
- In Inglese: Sign and Magnitude (SM)

Rappresentazione in Modulo e Segno

- Nella rappresentazione in Modulo e segno (MS) gli *n* bit che compongono il valore sono divisi in due parti:
  1. 1 bit per il segno
  2. *n* –1 bit per il modulo (il valore assoluto, ossia un numero in binario puro)

S	*Modulo*

- Il segno è posto nel MSB e:
  - 0 è usato per valori ≥ 0
  - 1 è usato per valori ≤ 0

Rappresentazione in Modulo e Segno

- Ci sono due differenti sequenze di bit per rappresentare lo 0, da considerare equivalenti nei calcoli
- Esempi
  - +5 in MS su 6 bit: **0**00101
  - +5 in MS su 8 bit: **0**0000101
  - –5 in MS su 6 bit: **1**00101
  - –5 in MS su 8 bit: **1**0000101
  - 0 in MS su 6 bit: **0**00000 **1**00000

Intervallo di rappresentabilità in Modulo e Segno

- Da un numero in MS di *n* bit:
  - se si ignora il bit di segno, si ottiene un numero in binario puro di $n-1$ bit, quindi il suo intervallo di rappresentabilità (range) è: $0 \rightarrow (2^{n-1} - 1)$
  - Aggiungendo il segno + (0) si ha: $+0 \rightarrow +(2^{n-1} - 1)$ [2^{n-1} valori positivi]
    mentre aggiungendo il segno – (1) si ha: $-(2^{n-1} - 1) \rightarrow -0$ [2^{n-1} valori negativi]
  - Componendo i due intervalli: $-(2^{n-1} - 1) \rightarrow -0, +0 \rightarrow +(2^{n-1} - 1)$ [2^n val.]

Intervallo di rappresentabilità in Modulo e segno

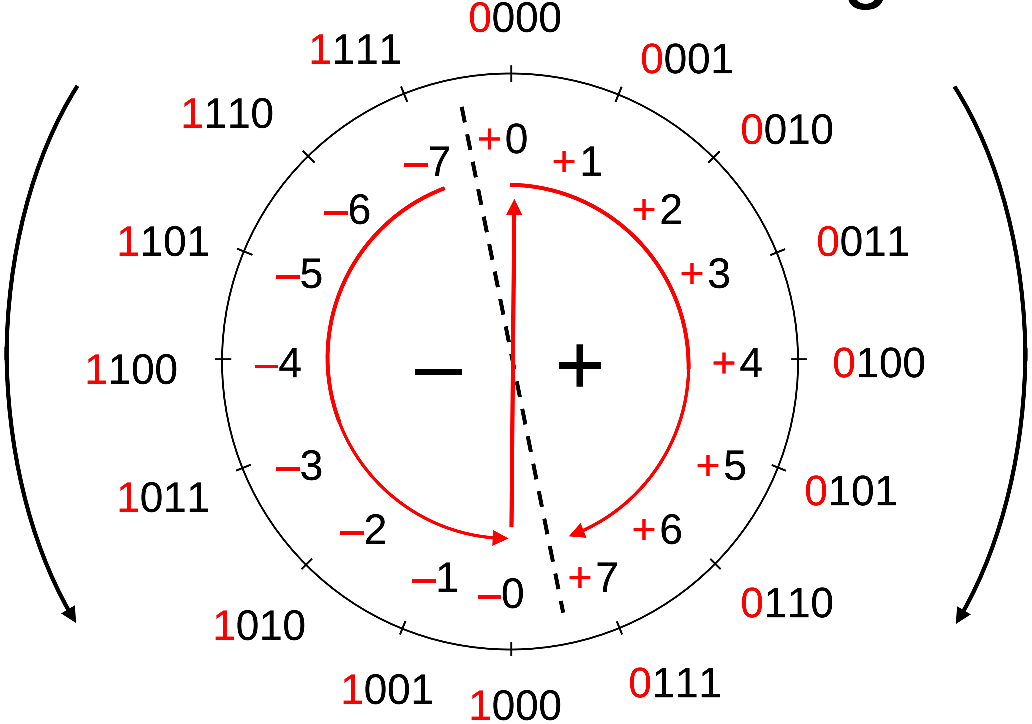
- Esempi
  - Un valore in MS su 8 bit può assumere valori da –127 a +127 (con 2 zeri)
  - Un valore in MS su 16 bit può assumere valori da –32767 a + 32767 (con 2 zeri)

Intervallo di rappresentabilità in Modulo e segno

- Confrontando i valori in MS con quelli in binario puro (es. su 8 bit) si nota che:
  - I valori in binario puro con MSB=0 costituiscono (e coincidono con) la metà positiva dei valori in MS
    (i valori 0→127 diventano +0→+127)
  - I valori in binario puro con MSB=1 vengono ribaltati e diventano la metà negativa dei valori in MS
    (i valori 128→255 diventano –0 → –127)

Intervallo di rappresentabilità in Modulo e segno

- Corrispondenza tra valori in binario puro e MS: discontinuità tra negativi e positivi



Pro e contro della rappresentazione in MS

- Facile da capire, usata normalmente in base 10 (con un simbolo per il segno)
- Facile da convertire
- Ci sono 2 diverse sequenze di bit che rappresentano lo stesso valore (0):
  - calcoli più complessi e lenti
  - richiedono circuito elettronici più complessi

Esercizi

- Convertire i valori come indicato
  - +12 → MS su 8 bit
  - –12 → MS su 8 bit
  - +23 → MS su 6 bit
  - –127 → MS su 8 bit
  - 10010001_{MS} → $()_{10}$
  - 010011_{MS} → $()_{10}$
  - 0000000_{MS} → $()_{10}$
  - 11111111_{MS} → $()_{10}$

Esercizi

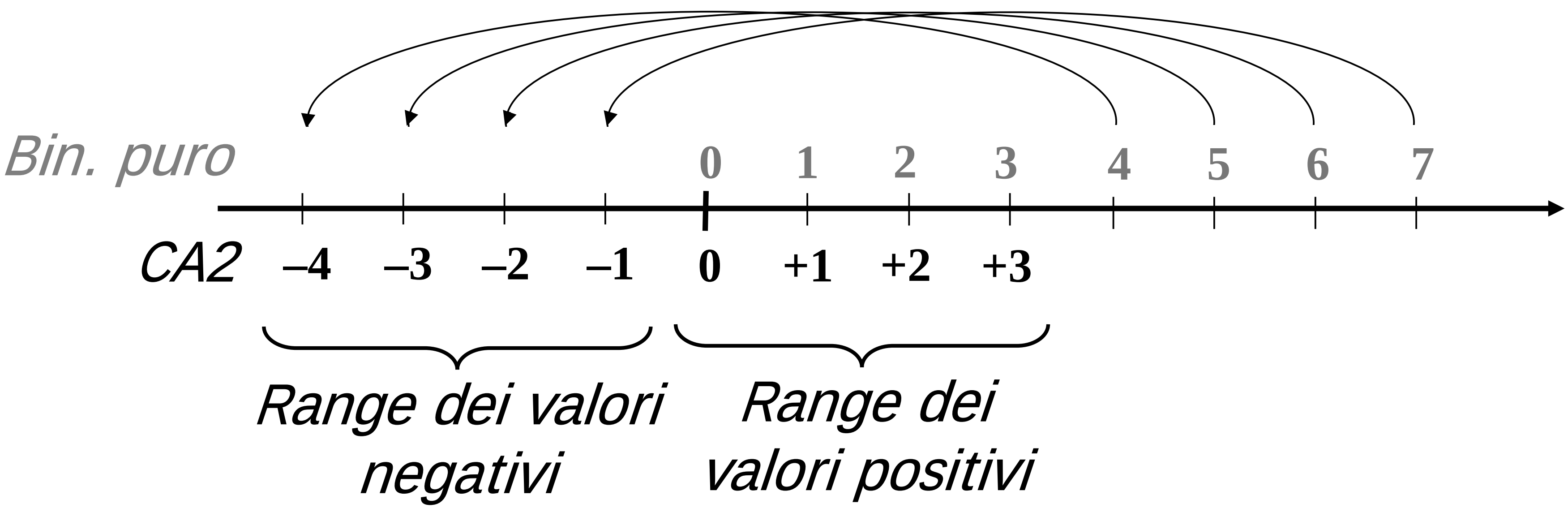
- Soluzioni
  - +12 → **0**0001100
  - –12 → **1**0001100
  - +23 → **0**10111
  - –127 → **1**1111111
  - 10010001_{MS} → –17
  - 010011_{MS} → +19
  - 0000000_{MS} → +0
  - 11111111_{MS} → –127

Rappresentazione in Complemento a 2

- La notazione in Complemento a 2 (CA2, in Inglese: 2’s Complement - 2C) è la più usata dai calcolatori attuali per memorizzare valori interi con segno
- Non ha il problema del doppio zero
- I calcoli sono più semplici e veloci di quelli in MS

Rappresentazione in Complemento a 2

- I valori in binario puro con MSB=1 vengono "traslati" sommando una costante negativa per rappresentare il range dei valori negativi in CA2



Rappresentazione in Complemento a 2

- I valori negativi allora non sono rappresentati con una sistema di numerazione posizionale

  7 6 5 4 3 2 1 0

  10001010 = 1 * 2^7 + 1 * 2^3 + 1 * 2^2 **NO**

- Ma il peso del primo bit (2^{n-1}) è proprio il valore sottratto ai positivi per cambiarli di segno (è pari alla metà del range dei numeri in binario puro 2^n)

Rappresentazione in Complemento a 2

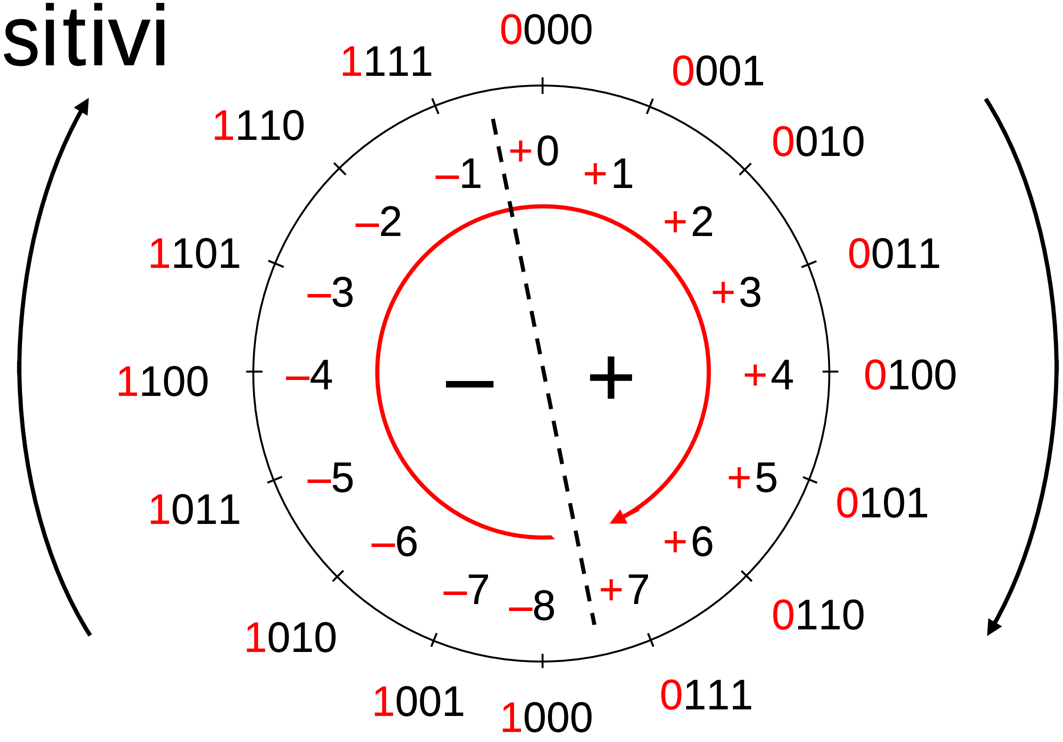
- Allora un numero negativo in CA2 può ancora essere considerato una somma di potenze della base, ma quella del MSB è però **negativa**:

$$\begin{array}{l} \scriptstyle 7\,6\,5\,4\,3\,2\,1\,0 \\ \mathbf{1}0010100 = -1 \cdot 2^7 + 1 \cdot 2^4 + 1 \cdot 2^2 = \end{array}$$

$$= -128+16+4 = -108$$

Rappresentazione in Complemento a 2

- Corrispondenza tra valori in binario puro e CA2: continuità tra negativi e positivi



Rappresentazione in Complemento a 2

- **Valori ≥ 0**

  Rappresentazione identica alla MS, il bit di segno è 0

0	*Modulo*

- **Valori < 0**

  Gli *n* bit (segno 1 incluso) sono il risultato di un calcolo

Rappresentazione in Complemento a 2

- L'*operazione di complemento a 2* può essere usata al posto della definizione (sommare -2^{n-1} al valore in binario puro del modulo) per calcolare il valore di un numero negativo dal positivo corrispondente
- Più in generale, l'operazione calcola l'opposto di un valore, quindi se questo è negativo si ottiene il positivo corrispondente

Rappresentazione in Complemento a 2

- Il termine “complemento a 2” è spesso usato senza ulteriori specificazioni, ma:
  - *la rappresentazione in complemento a 2* descrive come interpretare i bit che costituiscono un valore
  - *l’operazione di complemento a 2* è un calcolo che produce un valore di segno opposto ad un altro (già rappresentato in CA2)

Rappresentazione in Complemento a 2

- Operazione di CA2 (1º metodo)
  - invertire tutti i bit (ossia 0 ↔ 1)
  - aggiungere 1 al LSB
- Esempio
  - Dato (+88): 01011000_{CA2}

    complemento a 1: 10100111 +

    somma 1: 1 =

    risultato (–88): 10101000_{CA2}
  - Molte CPU hanno operazioni veloci di inversione dei bit e di incremento unitario

Rappresentazione in Complemento a 2

- Operazione di CA2 (2º metodo)
  - da DESTRA a SINISTRA copiare tutti i bit a 0 (se ce ne sono) fino al primo 1
  - copiare questo bit a 1
  - invertire i restanti bit
- Esempio

dato (88): 01011000_{CA2}

risultato (–88): 10101000_{CA2}

Rappresentazione in Complemento a 2

- Esempi
  - +5 in CA2 su 6 bit: **0**00101
  - +5 in CA2 su 8 bit: **0**0000101
  - –5 in CA2 su 6 bit: **1**11011
  - –5 in CA2 su 8 bit: **1**1111011
  - 0 in CA2 su 6 bit: **0**00000

Rappresentazione in Complemento a 2

- Esempio

  Convertire **0**0000101$_{CA2}$ in decimale.
  - Il valore è positivo, quindi il modulo è 0000101
  - Essendo un modulo in binario puro, vale 5
  - Risultato: +5

Rappresentazione in Complemento a 2

- Esempio

  Convertire $\mathbf{1}0101101_{CA2}$ in decimale.

  - Dalla definizione si calcola:

    $\mathbf{1}0101101 = -\mathbf{1}*2^{7}+1*2^{5}+1*2^{3}+1*2^{2}+1*2^{0}$

    $= -128+45 = -83$

Rappresentazione in Complemento a 2

- Esempio (metodo alternativo)

  Convertire **1**0101101$_{CA2}$ in decimale.
  - Si calcola il corrispondente valore positivo con l'operazione di complemento a 2:
    10101101$_{CA2}$ → **0**1010011$_{CA2}$
  - Nel valore positivo il modulo è un binario puro:
    1010011 → 83
  - Risultato: –83

Rappresentazione in Complemento a 2

- I valori della forma $1000...000_{CA2}$ calcolati con la definizione valgono -2^{n-1}
- L'operazione di complementazione su di essi porta allo stesso risultato

Intervallo di rappresentabilità del complemento a 2

- Considerando un numero X in CA2 su *n* bit:
  - se $X \geq 0$, il range è lo stesso del MS:
    $+0 \rightarrow +(2^{n-1} - 1)$ [2^{n-1} numeri positivi]
  - se $X \leq 0$, X può essere ottenuto dal corrispondente valore positivo con l'operazione di CA2, quindi si ha:
    $-(2^{n-1} - 1) \rightarrow -0$ [2^{n-1} numeri negativi]
  - ma –0 non esiste, esiste solo +0, quindi:
    $-(2^{n-1} - 1) \rightarrow -1$ [2^{n-1}-1 numeri negativi]

Intervallo di rappresentabilità del complemento a 2

- *(continuazione)*
  - ma il numero 100…0 vale a -2^{n-1} ed è posizionato a sinistra di $-(2^{n-1} - 1)$, quindi:
    $-2^{n-1} \rightarrow -1$ [2^{n-1} numeri negativi]
  - allora l'intervallo completo è:
    $-(2^{n-1}) \rightarrow 0 \rightarrow +(2^{n-1} - 1)$ [2^n valori]
- Esempi
  - Un numero in CA2 su 8 bit può assumere valori da –12**8** a +127
  - Un numero in CA2 su 16 bit può assumere valori da –3276**8** a +32767

Esercizi di conversione

- Convertire i seguenti valori in MS e CA2 (quando possibile)

VALORE	MS	CA2
+27	011011	011011
–10		
–3		
0		
+14		
–17		
–16		
+32		
–32		

Esercizi di conversione

- Soluzioni:

VALORE	MS	CA2
+27	011011	011011
–10	101010	110110
–3	100011	111101
0	$^{0}/_{1}$00000	000000
+14	001110	001110
–17	110001	101111
–16	110000	110000
+32	IMPOSS	IMPOSS
–32	IMPOSS	100000

Esercizi di conversione

- Convertire in decimale i seguenti valori considerandoli la prima volta in MS e la seconda in CA2

VALORE	MS	CA2
010101	+21	+21
110101		
10000		
11000		
1111		
10		
00000		
1		

Esercizi di conversione

- Soluzioni:

VALORE	MS	CA2
010101	+21	+21
110101	–21	–11
10000	–0	–16
11000	–8	–8
1111	–7	–1
10	–0	–2
00000	+0	0
1	IMP	–1

Estensione del segno

- Per convertire un valore in CA2 a *n* bit in uno a *m* bit in CA2 ($m > n$):
  - se il numero è positivo, vengono aggiunti *m-n* bit pari a 0 a sinistra del modulo (dopo il bit di segno):

    00101010 → **0**<u>00</u>0101010

Estensione del segno

- se il numero è negativo ciò non è corretto:

~~**1**0101010 → **1**<u>00</u>0101010~~ Errore!

- se per i valori negativi si passa al corrispondente positivo, si aggiungono gli 0 e si ripassa al negativo, si nota che i bit aggiunti sono 1

10101010 → **1**<u>11</u>0101010

Estensione del segno

- In entrambi i casi si ottiene lo stesso effetto aggiungendo *m-n* bit con il valore del segno *a sinistra del segno* stesso (da qui *estensione del segno*)
  - **0**0101010 → <u>00</u>**0**0101010
  - **1**0101010 → <u>11</u>**1**0101010
- In effetti tutti i primi bit uguali costituiscono il segno del numero
  - <u>**0000**</u>101010   <u>**111**</u>0101010

Estensione del segno

- Si può accorciare un numero in CA2 eliminando (ripetutamente) il MSB *se il risultato non cambia di segno*
  - **0**0**0**0101010 → 00101010 *OK*
  - **0**000**1**01010 → 101010 *NO!*
  - **11**10101010 → 110101010 *OK*
  - **1**1**1**0101010 → 10101010 *OK*
  - **1**11**0**101010 → 0101010 *NO!*

Esercizi di confronto

- Confrontare (*senza convertirli in decimale*) i due valori presenti ciascuna riga, considerandoli in MS e in CA2, mettendo un simbolo >, < o = nella colonna centrale

VALORE 1	MS	CA2	VALORE 2
00101	<	<	01101
01001			10001
10110			11010
1111			10001
101			11101

Esercizi di confronto

- Soluzioni:

VALORE 1	MS	CA2	VALORE 2	
00101	<	<	01101	
01001	>	>	10001	❶
10110	>	<	11010	❷
1111	<	>	10001	❷
101	>	=	11101	❸

- Note
  - ❶ i valori positivi sono sempre > dei negativi
  - ❷ Per i valori negativi, i confronti in CA2 e in MS sono sempre opposti (a meno che non sia lo stesso valore ❸ estendendo il segno)

Somma in Complemento a 2

- Sommando due valori in CA2 (con lo stesso numero di bit), il risultato è un numero in CA2 (positivo o negativo, con lo stesso numero di bit)
- La *sottrazione* è ottenuta sommando l'opposto del secondo valore:
  A – B = A + (– B) = A + CA2(B)
- Un eventuale riporto oltre il MSB è scartato e non è indicazione di overflow

Somma in Complemento a 2

- Si ha un ***overflow*** quando, sommando due valori concordi, il risultato ha segno opposto (il risultato perde di significato):
  - positivo + positivo = negativo
  - negativo + negativo = positivo
- Ovviamente la somma di due numeri discordi non dà mai overflow essendo il risultato un valore compreso tra i due

Somma in Complemento a 2

- Esempi di somma su 4 bit
  - 3 + 2

    0011+ → +3

    0010= → +2

    0101 → +5 → *OK*
  - 3 + 5

    0011+ → +3

    0101= → +5

    1000 → –8 → *OVERFLOW*

Somma in Complemento a 2

- Esempi di somma su 4 bit
  - –3 – 5

Riporto	1 **1**101+	→ –3	
	1011=	→ –5	
	1000	→ –8	→ *OK*

  - 3 + 4

0011+	→ +3	
0100=	→ +4	
0111	→ +7	→ *OK*

Somma in Complemento a 2

- Esempi di somma su 4 bit
  - 5 – 4

1	**0**101+	→	+5	
	1100=	→	–4	
	0001	→	+1	→ *OK*

Riporto

  - –5 + 3

1011+	→	–5	
0011=	→	+3	
1110	→	–2	→ *OK*

Somma in Complemento a 2 Esercizi

- Convertire i seguenti valori in CA2 su 5 bit (se possibile) ed effettuare i calcoli
  - 5 + 15
  - 4 + 7
  - 7 – 10
  - – 7 – 5
  - 9 – 19
  - – 10 – 15
  - 16 – 16

Somma in Complemento a 2 Esercizi

- Soluzioni:
  - 5 + 15 = ~~10100~~ *Overflow NoRiporto*
  - 4 + 7 = 01011 *OK NoRiporto*
  - 7 – 10 = 11101 *OK NoRiporto*
  - – 7 – 5 = 10100 *OK Riporto*
  - 9 – 19 = IMPOSS *(19 on 5 bit?)*
  - – 10 – 15 = ~~00111~~ *Overflow Riporto*
  - 16 – 16 = IMPOSS *(+16 on 5 bit?)*

Shift in Complemento a 2

- L’operazione di shift su un numero in CA2 produce un numero in CA2 (salvo overflow)
- Shift a destra (divisione per 2)
  - ogni bit è spostato a destra, anche il segno
  - il bit di segno viene duplicato (estensione)
    - $\mathbf{0}101010\underline{1}_{CA2}$ »1 = $\mathbf{00}101010_{CA2}$
    - $\mathbf{1}101010\underline{1}_{CA2}$ »1 = $\mathbf{11}101010_{CA2}$
  - La parte a destra della virgola è scartata:
    - shift ripetuti di valori positivi portano a 0
    - shift ripetuti di valori negativi portano a –1

Shift in Complemento a 2

- Shift a sinistra (moltiplicazione per 2)
  - ogni bit è spostato a sinistra
  - un bit 0 bit viene inserito a destra
  - il bit di segno viene eliminato
    - $\underline{0}0101010_{CA2}$ «1 = $0101010\mathbf{0}_{CA2}$
    - $\underline{1}1101001_{CA2}$ «1 = $1101001\mathbf{0}_{CA2}$
  - Poiché il valore aumenta, può esserci un overflow, si ha quando cambia il segno:
    - $\mathbf{1}0010110_{CA2}$ «1 = $\mathbf{0}0101100_{CA2}$ *OVERFLOW*

Shift in Complemento a 2

(Continuazione)

- In uno shift di più bit, ogni shift intermedio (di 1 bit) *deve mantenere il segno*, ossia non è corretto valutare solo i segni prima e dopo uno shift multiplo per identificare l'ov.
  - $00101010_{CA2} \cdot 8$ (ossia $\cdot 8 \rightarrow \cdot 2^3 \rightarrow$ »3):

00101010$_{CA2}$ »1 = **0**1010100$_{CA2}$ *1º SHIFT: OK*

01010100$_{CA2}$ »1 = **1**0101000$_{CA2}$ *2º SHIFT: OVERF*

10101000$_{CA2}$ »1 = ~~**0**1010000$_{CA2}$~~

c'è già stato overflow per cui il risultato è non corretto

Shift in Complemento a 2 Esercizi

- Convertire in CA2 su 5 bit i valori a sinistra ed effettuare i calcoli con shift
  - 6 · 2 00110«1 → 0110<u>0</u> *OK*
  - 10 · 4
  - –2 · 8
  - 4 · 4
  - –4 · 4
  - 4 / 4
  - –4 / 4
  - 4 / 8
  - –4 / 8
  - –4 · 8

Shift in Complemento a 2 Esercizi

- Soluzioni:
  - 6 · 2 — $\underline{0}0110$«1 → $0110\underline{0}$ *OK*
  - 10 · 4 — $\underline{01}010$«2 → $\cancel{010\underline{00}}$ *OV (1º shift)*
  - –2 · 8 — $\underline{111}10$«3 → $10\underline{000}$ *OK*
  - 4 · 4 — $\underline{00}100$«2 → $\cancel{100\underline{00}}$ *OV (2º shift)*
  - –4 · 4 — $\underline{11}100$«2 → $100\underline{00}$ *OK*
  - 4 / 4 — $001\underline{00}$»2 → 00001 *OK*
  - –4 / 4 — $111\underline{00}$»2 → 11111 *OK (–1)*
  - 4 / 8 — $00\underline{100}$»3 → 00000 *OK*
  - –4 / 8 — $11\underline{100}$»3 → 11111 *OK (–1)*
  - –4 · 8 — $\underline{111}00$«3 → $\cancel{00\underline{000}}$ *OV (3º shift)*

Operazioni in CA2
Esercizi

- Calcolare le seguenti espressioni dopo aver convertito i valori decimali in CA2 su 8 bit verificando eventuali overflow. Utilizzare opportuni shift per moltiplicazioni e divisioni.
  1. $A=5$, $B=-2$ $\quad$ $A/4 - B \cdot 2$
  2. $A=-16$, $B=7$ $\quad$ $(A/2 - B \cdot 4) / 4$
  3. $A=123$, $B=-16$ $\quad$ $(A \cdot 2 - B/8) \cdot 4$

CA2 Operation Esercizi

- Soluzione (A=5, B=–2 A/4 – B·2)

  A/4 – B·2 = A/4 + CA2(B·2)

  +5 = 00000101 → A = +5= 00000101

  +2 = 00000010 → B = –2 = 11111110

  A/4 = A»2 = 00000001

  B·2 = B«1= 11111100

  –B·2= CA2(11111100) = 00000100

  A/4+ → 00000001+

  –B·2= → 00000100=

  00000101 → +5 (NO OVERF)

CA2 Operation Esercizi

- Soluzione (A=–16, B=7 (A/2 – B·4) / 4)

(A/2 – B·4) / 4 = (A/2 + CA2(B·4)) / 4

+16 = 00010000 → A = –16 = 11110000

+7 = 00000111 → B = +7 = 00000111

A/2 = A»1 = 11111000

B·4 = B«2= 00011100

–B·4= CA2(00011100) = 11100100

	1 1 1	
A/2+	11111000 +	
–B·4=	11100100 =	(NO OV)
	11011100 / 4 → 11110111 (-9)	

CA2 Operation Esercizi

- Soluzione (A=123, B=–16 (A·2–B/8)·4)

  (A·2 – B/8) · 4 = (A·2 + CA2(B/8)) · 4

  +123= 01111011 → A = +123 = **0**1111011

  +16 = 00010000 → B = –16 = 11110000

  A·2 = ~~**1**1110110~~ (OVERFLOW)

  Inutile continuare dato che la prima operazione ha già dato overflow

Moltiplicazione e divisione in Complemento a 2

- La moltiplicazione e la divisione di valori in CA2 viene effettuata dai processori sui corrispondenti valori positivi, poi al risultato viene assegnato il segno corretto (così –1 / 2 = 0 e non –1)

Complemento a 2 e Modulo e segno in esadecimale

- Per facilità di scrittura, possono essere scritti con cifre esadecimali raggruppando i bit a 4 a 4
  - $4C2_{CA2} = 010011000010_{CA2} = +1218_{10}$
  - $A2E_{CA2} = 101000101110_{CA2} = -1490_{10}$
  - $4C2_{MS} = 010011000010_{MS} = +1218_{10}$
  - $A2E_{MS} = 101000101110_{MS} = -558_{10}$
- Non sono numeri esadecimali perché non moltiplicano sempre potenze di 16