Note sull’esecuzione delle somme in complemento a 2

Conversione in complemento a 2

- Per convertire da 2 a C2...
- Se il numero è positivo la rappresentazione è la stessa...
- La regola “inverti i bit e aggiungi 1” permette di passare da $+X_2$ a $-X_{C2}$.
- Esempio a 8 bit:

  $(32)_{10} = (00010000)_2 = (00010000)_{C2}$

  $(-32)_{10} = -(00010000)_2 = (11101111+1)_{C2}$

Nel caso in cui non venga specificato il numero di bit...

- Quanti bit dobbiamo usare per eseguire la somma di due numeri in complmento a 2?
- Siano b1 e b2 due numeri binari composti rispettivamente da m e n cifre (rappresentazione in base 2).
- Il numero di bit da utilizzare per la somma in complemento a 2 è, nel caso peggiore, pari a max(m,n)+2.
- I due bit “aggiuntivi” servono per:
  - Codificare il segno di ciascun numero;
  - Evitare l’overflow.

Nel caso in cui non venga specificato il numero di bit (esempio)

- b1 = $+15_{10}$, b2 = $+7_{10}$;
- b1 = 1111_2, b2 = 111_2;
- quindi m=4, n=3;
- Per eseguire l'operazione in complemento a due ho bisogno di max(m,n)+2=max(4,3)+2=6 bit.

Nel caso in cui non venga specificato il numero di bit (esempio)

- Proviamo ad eseguire l'operazione utilizzando solo 5 bit (estendo i numeri con i bit di segno):

  01111_{C2} +

  00111_{C2} =

  10110_{C2}
- Il risultato è negativo (bit di segno pari a 1!). Sto utilizzando un numero di bit troppo basso, quindi vado in overflow.
- Ciò accade perchè, per rappresentare il risultato dell'operazione (15+7=22), ho bisogno di 5 bit per rappresentare il valore 22 + 1 bit per rappresentare il segno!

Nel caso in cui non venga specificato il numero di bit (esempio)

- Se eseguiamo invece l'operazione con 6 bit...

  001111_{C2} +

  000111_{C2} =

  010110_{C2}

- Il risultato è in questo caso corretto. Abbiamo utilizzato 5 bit per codificare il valore assoluto del numero + 1 bit per il segno.

Nel caso in cui non venga specificato il numero di bit (esempio)

- Cosa succede se utilizziamo un numero più alto di bit per eseguire l'operazione? Es. Utilizziamo 8 bit...

  00001111_{C2} +

  00000111_{C2} =

  00010110_{C2}
- Il risultato è ancora corretto. Il bit di segno viene replicato all'infinito a sinistra...
- => E' quindi possibile "tagliare" il numero a sinistra quando il bit di segno si replica all'infinito!

- Esercizio -> provare a sommare due numeri negativi per verificare la correttezza di quanto esposto.
- Esercizio -> provare a sommare un numero positivo ed uno negativo.

Nel caso in cui venga specificato il numero di bit

- In questo caso (è il caso reale! => i registri del calcolatore hanno dimensione finita) il numero di bit viene fissato a priori.
- Il calcolo viene effettuato utilizzando il numero di bit a disposizione, es. 8.
- I bit in eccesso (es. Il nono bit) vengono scartati.
- E' necessario verificare l'overflow: sommando due numeri positivi, il risultato deve essere positivo; sommando due numeri negativi, il risultato deve essere negativo.
- Sommando un numero positivo ed uno negativo, l'overflow non si verifica.