Risoluzione degli esercizi proposti dall'1 al 6

TOSTO FRANCESCA
MATRICOLA 171693

1. Calcolare le seguenti conversioni di base:

a) $(263)_8 \rightarrow (\ )_3$

Prendo il numero in base 8 e lo calcolo in base 10 così posso fare la conversione dal sistema decimale a quello in base 3 operando ripetutamente la divisione del numero per la base 3:

$(263)_8 = (((2*8)+6)*8)+3 = 179$

$179:3 = 59\ resto\ 2$
$59:3 = 19\ resto\ 2$
$19:3 = 6\ resto\ 1$
$6:3 = 2\ resto\ 0$
$2:3 = 0\ resto\ 2$

Quindi il numero è $(20122)_3$.

Verifica: $(((((2*3)*3)+1)*3)+2)*3)+2 = 179$

b) $(242)_5 \rightarrow (\ )_7$

Seguo lo stesso procedimento di prima e vedo che il numero in base 8 corrisponde al numero in base 10 72.

$72:7 = 10\ resto\ 0$
$10:7 = 1\ resto\ 3$
$1:7 = 0\ resto\ 1$

Quindi il numero è $(132)_7$.

Verifica: $((7+3)*7)+2 = 72$

2. Si valuti per quale base x sono vere le seguenti relazioni:

a) $(45)_x \rightarrow (100101)_2$

Deve essere una base $\geq$ 6 in quanto presenta cifre fino a 4 e 5.

Opero la conversione da binario a decimale e ottengo il numero 37 in base 10.

Quindi $(45)_x = 37$

E per la regola che usiamo per convertire a base 10 si ha: $4x+5 = 37 \quad x = 8$ che è la base cercata.

b) $(431)_x \rightarrow (71)_{13}$

Deve essere una base $\geq$ 5. Il numero 71 in base 13 diventa 92 in base 10 sempre con il medesimo procedimento.

Per la definizione di numero i base fissa, dobbiamo porre: $4X^2+3x+1 = 92$, da cui $4X^2+3x-91 = 0$.
$x = \frac{-3 \pm \sqrt{9+4*4*91}}{2} = \frac{-3 \pm \sqrt{1463}}{2} = \frac{-3 \pm 38.249}{2}$. Nessuna radice è intera, quindi non esiste una base che verifichi la corrispondenza. Metodo intuitivo:
per la regola di conversione in base 10 si ha: $(4x+3)x+1=92$ che non dà risultati accettabili. Si può concludere che questa uguaglianza non è valida per alcuna base, anche perché già con base 5 il numero in questione varrebbe 116 > 92, aumentando ulteriormente la base il numero superebbe sempre di più 92.

3. Valutare le seguenti operazioni assumendo una rappresentazione in complemento a 2 su 4 bit. Per ciascuna di esse spiegare se il risultato è esprimibile su 4 bit.

a) -3+5

Il modulo di 3 in complemento a due è 11 perciò per trovare -3 e rappresentarlo su 4 bit faccio il complemento a due e ottengo 1101. Per +5 so che il modulo è 101 quindi su 4 bit il suo valore sarà 0101.

```
 1101 +
 0101 =
1 0010
```

Nota: il carry finale cade fuori dalla rappresentazione e si scarica $0010_2 = 2_{10}$ risultato corretto in 4 bit e questo si poteva già osservare prima in quanto i due operandi hanno segno discorde quindi non si può avere overflow nella somma CA2.

-9+3

non si può fare perché -9 non può essere rappresentato in 4 bit. Infatti il modulo di 9 è 1001 e -9 sarebbe 10111 su 5 bit.

b) +2+6= 0010+0110

Qui c'è possibilità di overflow poiché i due operandi sono concordi.

```
 0010 +
 0110 =
 1000    overflow: cambia il segno!
```

Infatti +8 in CA2 si scrive 01000 e quindi si può rappresentare solo in 5 bit. Inoltre in 4 bit posso rappresentare solo numeri da -2^3 a 2^3-1 quindi da -8 a +7 quindi +8 non avrei comunque potuto rappresentarlo.

-4+4

Il modulo di 4 è 100 quindi +4 in CA2 su 4 bit sarà 0100 e -4 sarà 1100.

```
 1100 +
 0100 =
1 0000
```

Nota: il carry finale cade fuori dalla rappresentazione e si scarica $0000_2 = 0_{10}$ risultato corretto su 4 bit, inoltre i due operandi erano discordi quindi non poteva esserci overflow.

c)-7+8

Il modulo di 7 in CA2 è 111 quindi -7= 1001 su 4 bit, +8 non si può rappresentare su 4 bit, quindi l'operazione non si può fare. Un matematico, ma non una macchina, potrebbe osservare che -8 si può rappresentare su 4 bit (è il numero più negativo, 1000), e che possiamo vedere -7+8 come

-(+7-8). Inoltre avendo gli addendi segni opposti non ci saranno problemi di overflow.

```
 0111 +
 1000 =
 1111
```

Per ottenere il risultato corretto, occorre complementare a 2 il risultato ottenuto:

1111 → 0001= 1 risultato richiesto

4+4

Il modulo di 4 come detto precedentemente è 100 quindi +4 sarà 0100. Ci potrà essere overflow in quanto i due operandi sono concordi e ciò si avrà se il risultato sarà discorde.

```
0100 +
0100 =
1000   = -8 overflow perché su 4 bit non si può rappresentare +8 ma solo su 5
```

(01000).

4. Si eseguano le seguenti operazioni su numeri in complemento a 2, espressi su 5 bit, indicando una eventuale situazione di overflow:

a) 10111 + 10011

```
10111 +  → -16+4+2+1= -9
10011 =  → -16+2+1= -13
 01010   → overflow perché cambia il bit del segno. Infatti -22 può essere rappresentato
```

solo su 6 bit. Ciò si poteva capirlo dal fatto che su 5 bit si possono rappresentare numeri solo da -16 a +15, inoltre i due operandi erano concordi quindi c'era possibilità di ovreflow.

b) 00011 + 10001

```
00011 +  → +3
10001 =  → -15
10100   = -12 → risultato corretto, i due operandi sono discordi.
```

c) 01101 – 10011

Per effettuare una sottrazione in c.a 2 occorre sommare al minuendo il c.a 2 del sottraendo:

```
01101 +  → +13
01101 =  → +13
11010  → overflow perché cambia il segno. Infatti il risultato sarebbe +26 ma non è
```

rappresentabile su 5 bit.

5. Si considerino i numeri in modulo e segno espressi su 8 bit. Si valuti l'intervallo di rappresentazione (min e max) nel caso di numeri interi e di numeri con virgola binaria tra il secondo e terzo bit da destra (es 101010.00)

Su 8 bit i numeri espressi in modulo e segno possono andare da $-(2^7-1)$ a $+(2^7+-1)$, cioè da -127 (11111111) a +127 (01111111).

Questi saranno anche i valori massimo e minimo dell'intervallo dei numeri interi. Nel caso, invece, dei numeri con la virgola, essi saranno del tipo: xxxxxx.yy dove yy può essere 00, 11,10,01.

Inoltre dopo la virgola il numero massimo che possiamo avere è 11 cioè 0.75, il minimo è 00 cioè 0.00.

La parte intera è perciò costituita da 6 bit che potranno variare in un intervallo da $-(2^5-1)$ a $+(2^5-1)$, cioè da -31 a +31.

A questo punto si può concludere che il minimo numero con la virgola è -31.75, cioè 111111.11 mentre il massimo è +31.75, cioè 011111.11.

6. Calcolare il risultato delle seguenti operazioni su numeri assoluti, considerando operandi e risultato rappresentati in binario su 5 bit. Per ciascuna di esse verificare preventivamente se l'operazione è possibile, spiegando in caso contrario la ragione

a) 01011+00111

Operazioni su numeri assoluti, cioè senza segni, quindi è come se fosse un'operazione in binario puro.

01011+ → 11

00111= → 7

L'operazione può avvenire senza problemi in quanto nel posto dell'MSB c'è, in entrambi gli operandi, la cifra 0. Quindi è come se fossero due numeri da 4 bit che sommati tra di loro possono dare al massimo un numero da 5 bit, non di più.

Perciò

```
01011 +
00111 =
10010     = 18
```

b) 11011+10001

11011+ → 27

10001= → 17

La somma di questi due operandi darebbe 44 che non può esser rappresentato su soli 5 bit, ma ne 6 (44= 101100). Inoltre si può notare che le due MSB dei due operandi sono uguali a 1 e solamente la somma di queste due richiede già un sesto bit, quindi ci sarà sicuramente overflow.

```
 11011 +
 10001 =
 01100
```

1 → nella somma di numeri assoluti la presenza di carry dopo il bit più significativo indica overflow

c) 01011+10000

01011+ → 11

10000= → 16

Il risultato è 27 quindi non ci sarà overflow, inoltre tutte le cifre 1 vengono sommate a cifre uguali a 0 quindi non si avranno mai riporti e non si potrà "sforare" nel sesto bit.

```
01011 +
10000 =
11011  = 27
```

d) 01000+01101

01000+ → 8

01101= → 13

Anche in questo caso non ci sarà overflow poiché gli unici 1 che si sommano tra loro e danno un riporto uguale a 1 sono nel quarto bit e nel quinto bit ci sono solo zeri.

```
01000 +
01101 =
10101  = 21
```