Esercitazione Programmazione Procedurale con Laboratorio

Basi e Rappresentabilità

Esercizio 1 Conversione e modulo a segno

Tradurre in base 10 i seguenti numeri rappresentati in base due e modulo a segno (sign-module), considerando un ordinamento *big endian*. In seguito, rappresentare il complementare rispetto a 0 ($-x$). Supporre che tutti questi caratteri siano di tipo *char*, cioè occupino un byte (8 bit). Qual è il range di numeri rappresentabili con queste caratteristiche di tipo?

- 11111111_2
- 00011101_2
- 00111001_2
- 10000000_2
- 11011000_2
- 10101010_2

Soluzione dell'esercizio 1

- $11111111_2 \equiv -127_{10}$, $-x = 01111111_2$
- $00011101_2 \equiv 29_{10}$, $-x = 10011101_2$
- $00111001 \equiv 57_{10}$, $-x = 10111001_2$
- $10000000_2 \equiv -0_{10}$, $-x = 00000000_2$
- $11011000_2 \equiv -88_{10}$, $-x = 01011000_2$
- $10101010_2 \equiv -42_{10}$, $-x = 00101010_2$

Si possono rappresentare tutti i valori interi nell'intervallo (range) $[-127, 127]$.

Esercizio 2 Conversione e complemento a due

Tradurre in base 10 i seguenti numeri rappresentati in base due e complemento a due (two's complement), considerando un ordinamento *big endian*. In seguito, rappresentarne il complementare rispetto a 0 ($-x$). Supporre che tutti questi caratteri siano di tipo *char*, cioè occupano 8 bit. Qual è il range di numeri rappresentabili con queste caratteristiche di tipo?

- 01111111_2
- 00101010_2
- 11111111_2
- 01011111_2
- 10011001_2

- 00000001_2
- 10000000_2

Soluzione dell'esercizio 2

- $01111111_2 \equiv 127_{10}$, $-x = 10000001_2$
- $00101010 \equiv 42_{10}$, $-x = 11010110_2$
- $11111111_2 \equiv -1_{10}$, $-x = 00000001_2$
- $01011111_2 \equiv 95_{10}$, $-x = 10100001_2$
- $10011001_2 \equiv -103_{10}$, $-x = 01100111_2$
- $00000001_2 \equiv 1_{10}$, $-x = 11111111_2$
- $10000000_2 \equiv -128_{10}$, $-x$ non è ottenibile perché il massimo valore rappresentabile con 8 bit in complemento a due è 127 (vedi ultima risposta).

Si possono rappresentare tutti i valori nell'intervallo (range) $[-128, 127]$.

Esercizio 3 Conversione e little endian

Convertire i seguenti numeri in base 2, utilizzando la rappresentazione in complemento a due e solamente 6 bit: per esempio, $7 \equiv 000111$. Rappresentare i numeri sia in *big endian* che in *little endian*.

- -16_{10}
- 13_{10}
- -3_{10}
- -10_{10}
- 26_{10}
- -31_{10}

Soluzione dell'esercizio 3

- 110000 (big endian), 000011 (little endian)
- 001101 (big endian), 101100 (little endian)
- 111101 (big endian), 101111 (little endian)
- 110110 (big endian), 011011 (little endian)
- 011010 (big endian), 010110 (little endian)
- 100001 (big endian), 100001 (little endian)

Esercizio 4 Rappresentabilità

Quale intervallo di numeri è possibile rappresentare utilizzando:

- 11 bit, solo interi positivi (0, 1, 2, etc)
- 10 bit, modulo a segno
- 12 bit, in complemento a due

Soluzione dell'esercizio 4

- $[0, 2^{11} - 1] = [0, 2047]$
- $[-2^9 + 1, 2^9 - 1] \equiv [-511, 511]$
- $[-2^{11}, 2^{11} - 1] \equiv [-2048, 2047]$

Esercizio 5 Conversione tra basi differenti

Convertire i seguenti numeri dalla loro base alla base richiesta. I seguenti numeri sono espressi come letterali in linguaggio C. Il prefisso *0x* indica un valore in base esadecimale, mentre il prefisso *0* indica un valore in base ottale.

- $0xfa$ a base 10
- 017 a base 10
- $0xaBc$ a base 10
- 0128 a base 10
- 844 a base 16
- 245 a base 8

Soluzione dell'esercizio 5

- 250
- 15
- 2748
- 0128 non è un numero valido in base 8, dato che possono essere utilizzate solo le cifre da 1 a 7
- $0x34c$
- 0365

Esercizio 6 Addizione/sottrazione in binario

Eseguire direttamente in binario le seguenti operazioni, considerando ciascun valore rappresentato con tipo *int*, cioè con 4 byte (complemento a due):

- $69 + 12$
- $69 + (-12)$
- $12 + (-69)$

Soluzione dell'esercizio 6

Le seguenti operazioni sono state effettuate come addizioni, eventualmente invertendo il seguente addendo.

```
                                                    1  1      Carry Row
  0000 0000 0000 0000 0000 0000 0100 0101 (69)
+ 0000 0000 0000 0000 0000 0000 0000 1100 (12)
-------------------------------------------
  0000 0000 0000 0000 0000 0000 0101 0001 (81)
```

```
  1111 1111 1111 1111 1111 1111 1      1     Carry Row
  0000 0000 0000 0000 0000 0000 0100 0101 (69)
+ 1111 1111 1111 1111 1111 1111 1111 0100 (-12)
-------------------------------------------
  0000 0000 0000 0000 0000 0000 0011 1001 (57)
                                  111       Carry Row
  0000 0000 0000 0000 0000 0000 0000 1100 (12)
+ 1111 1111 1111 1111 1111 1111 1011 1011 (-69)
-------------------------------------------
  1111 1111 1111 1111 1111 1111 1100 0111 (-57)
```

Esercizio 7 Addizione/sottrazione in binario

Come esercizio precedente, supponendo questa volta che ciascun addendo sia rappresentato come *char* (1 byte).

- $127 + 1$
- $104 + 45$
- $-103 + (-69)$

Soluzione dell'esercizio 7

Le soluzioni possono essere trovate all'indirizzo: `http://sandbox.mc.edu/~bennet/cs110/tc/add.html`

Esercizio 8 Rappresentazione in virgola mobile

Per questo esercizio, considerare il seguente formato semplificato per i numeri in virgola mobile (*floating point*). Il più a sinistra di una stringa rappresenta il segno (s), i secondi 4 bit rappresentano l'esponente (q). I rimanenti 11 bit rappresentano la mantissa c (o significando). Sia q che c sono rappresentati in modulo a segno e in little endian. Calcolare il valore in base 10 delle seguenti stringhe, utilizzando la formula $(-1)^s \times c \times 2^q$.

$$1|1000|00110000000 \tag{1}$$

$$0|1001|00010000000 \tag{2}$$

$$0|1011|01000000000 \tag{3}$$

Soluzione dell'esercizio 8

(1) $(-1)^1 \times 12 \times 2^1 = -24$

(2) $(-1)^0 \times 8 \times 2^{-1} = 4$

(3) $(-1)^0 \times 2 \times 2^{-5} = 0.0625$