Tutorato Architettura degli Elaboratori Modulo 1
Lezione 1

Francesco Pelosin

16 Ottobre 2019

1 Conversione di base

1.1 Base $b \rightarrow$ Base 10

Dato un numero di n cifre espresso in una qualsiasi base b:

$$d_{n-1}, d_{n-2}, \ldots, d_1, d_0$$

Possiamo esprimerlo in base 10 nel seguente modo:

$$d_{n-1} \cdot b^{n-1} + d_{n-2} \cdot b^{n-2} + \cdots + d_1 \cdot b^1 + d_0 \cdot b^0$$

Esercizi

Tradurre i seguenti numeri in base 10:

(a) 11310_5

(b) 147_{12}

(c) $3A7_{16}$

(d) 1703_8

Soluzioni

(a) $11310_5 = 1 \cdot 5^4 + 1 \cdot 5^3 + 3 \cdot 5^2 + 1 \cdot 5^1 + 0 \cdot 5^0 = 625 + 125 + 75 + 5 = 830_{10}$

(b) $147_{12} = 1 \cdot 12^2 + 4 \cdot 12^1 + 7 \cdot 12^0 = 144 + 48 + 7 = 199_{10}$

(c) $3A7_{16} = 3 \cdot 16^2 + 10 \cdot 16^1 + 7 \cdot 16^0 = 768 + 160 + 7 = 935_{10}$

(d) $1703_8 = 1 \cdot 8^3 + 7 \cdot 8^2 + 0 \cdot 8^1 + 3 \cdot 8^0 = 512 + 448 + 3 = 963_{10}$

1.2 Base 10 $\rightarrow$ Base b

Dato un numero n espresso in base 10 è possibile tradurlo in base b applicando il seguente algoritmo.

Algorithm 1 $\text{Base}_{10} \rightarrow \text{Base}_b$ (Conversione Inversa)

procedure ConversioneInversa(n, b)
  $Q \leftarrow n$
  $i \leftarrow 0$
  while $Q > 0$ **do**
    $d_i = Q \mod b$ ▷ Resto della divisione per b
    print d_i
    $Q = Q/b$ ▷ Divisione intera
    $i = i + 1$

Esercizi

Tradurre i seguenti numeri nella base richiesta:

(a) $310_{10} \rightarrow$ in Base 6

(b) $321_{10} \rightarrow$ in Base 2

(c) $519_{10} \rightarrow$ in Base 5

(d) $1006_{10} \rightarrow$ in Base 9

Soluzioni

N.B.: la notazione es.: $6|310$ significa "310/6".

(a) 310_{10} :

$$\left.\begin{array}{r|l} 6\lfloor\underline{310} & 4 \\ 6\lfloor\underline{51} & 3 \\ 6\lfloor\underline{8} & 2 \\ 6\lfloor\underline{1} & 1 \end{array}\right\} = \texttt{1234}_6$$

(b) 321_{10} :

$$\left.\begin{array}{r|l} 2\lfloor\underline{321} & 1 \\ 2\lfloor\underline{160} & 0 \\ 2\lfloor\underline{80} & 0 \\ 2\lfloor\underline{40} & 0 \\ 2\lfloor\underline{20} & 0 \\ 2\lfloor\underline{10} & 0 \\ 2\lfloor\underline{5} & 1 \\ 2\lfloor\underline{2} & 0 \\ 2\lfloor\underline{1} & 1 \end{array}\right\} = \texttt{101000001}_2$$

(c) 519_{10} :

$$\left.\begin{array}{r|l} 5\lfloor\underline{519} & 4 \\ 5\lfloor\underline{103} & 3 \\ 5\lfloor\underline{20} & 0 \\ 5\lfloor\underline{4} & 4 \end{array}\right\} = \mathtt{4034}_5$$

(d) 1006_{10} :

$$\left.\begin{array}{r|l} 9\lfloor\underline{1006} & 7 \\ 9\lfloor\underline{111} & 3 \\ 9\lfloor\underline{12} & 3 \\ 9\lfloor\underline{1} & 1 \end{array}\right\} = \mathtt{1337}_9$$

Esercizi

Convertire i seguenti numeri da Base 10 a Base 2:

(a) 136_{10}

(b) 192_{10}

(c) 255_{10}

(d) 35_{10}

(e) 63_{10}

Convertire i seguenti numeri da Base 2 a Base 10:

(a) 100011_2

(b) 101100_2

(c) 111111_2

(d) 10001000_2

(e) 11000000_2

(f) 11111111_2

Soluzioni

N.B.: la notazione es.: $2|136$ significa “136/2”.

(a) 136_{10} :

$$\left.\begin{array}{r|l} 2\lfloor\underline{136} & 0 \\ 2\lfloor\underline{68} & 0 \\ 2\lfloor\underline{34} & 0 \\ 2\lfloor\underline{17} & 1 \\ 2\lfloor\underline{8} & 0 \\ 2\lfloor\underline{4} & 0 \\ 2\lfloor\underline{2} & 0 \\ 2\lfloor\underline{1} & 1 \end{array}\right\} = \texttt{10001000}_2$$

(b) 192_{10} :

$$\left.\begin{array}{r|l} 2\lfloor\underline{192} & 0 \\ 2\lfloor\underline{96} & 0 \\ 2\lfloor\underline{48} & 0 \\ 2\lfloor\underline{24} & 0 \\ 2\lfloor\underline{12} & 0 \\ 2\lfloor\underline{6} & 0 \\ 2\lfloor\underline{3} & 1 \\ 2\lfloor\underline{1} & 1 \end{array}\right\} = \texttt{11000000}_2$$

(c) 255_{10} :

$$\left.\begin{array}{r|l} 2\lfloor\underline{255} & 1 \\ 2\lfloor\underline{127} & 1 \\ 2\lfloor\underline{63} & 1 \\ 2\lfloor\underline{31} & 1 \\ 2\lfloor\underline{15} & 1 \\ 2\lfloor\underline{7} & 1 \\ 2\lfloor\underline{3} & 1 \\ 2\lfloor\underline{1} & 1 \end{array}\right\} = \texttt{11111111}_2$$

(d) 35_{10} :

$$\left.\begin{array}{r|l} 2\lfloor\underline{35} & 1 \\ 2\lfloor\underline{17} & 1 \\ 2\lfloor\underline{8} & 0 \\ 2\lfloor\underline{4} & 0 \\ 2\lfloor\underline{2} & 0 \\ 2\lfloor\underline{1} & 1 \end{array}\right\} = \texttt{100011}_2$$

(e) 63_{10} :

$$\left.\begin{array}{r|l} 2\underline{\lfloor 63} & 1 \\ 2\underline{\lfloor 31} & 1 \\ 2\underline{\lfloor 15} & 1 \\ 2\underline{\lfloor 7} & 1 \\ 2\underline{\lfloor 3} & 1 \\ 2\underline{\lfloor 1} & 1 \end{array}\right\} = \texttt{111111}_2$$

(a) $100011_2 = 1 \cdot 2^5 + 0 \cdot 2^4 + 0 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 = 32 + 2 + 1 = 35_{10}$

(b) $101100_2 = 1 \cdot 2^5 + 0 \cdot 2^4 + 1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 0 \cdot 2^0 = 32 + 8 + 4 = 44_{10}$

(c) $111111_2 = 1 \cdot 2^5 + 1 \cdot 2^4 + 1 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 = 32+16+8+4+2+1 = 63_{10}$

(d) $10001000_2 = 1 \cdot 2^7 + 0 \cdot 2^6 + 0 \cdot 2^5 + 0 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 0 \cdot 2^0 = 128 + 8 = 136_{10}$

(e) $11000000_2 = 1 \cdot 2^7 + 1 \cdot 2^6 + 0 \cdot 2^5 + 0 \cdot 2^4 + 0 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 0 \cdot 2^0 = 128 + 64 = 192_{10}$

(f) $11111111_2 = 1 \cdot 2^7 + 1 \cdot 2^6 + 1 \cdot 2^5 + 1 \cdot 2^4 + 1 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 = 128 + 64 + 32 + 16 + 8 + 4 + 2 + 1 = 255_{10}$

1.3 Base 2 → Base 8 (o Base 16)

Ricorda: Dato un sistema numerico in base b e date n cifre diverse, possiamo rappresentare b^n numeri diversi.

Considerando il sistema numerico binario a 3 cifre, possiamo rappresentare $2^3 = 8$ numeri diversi, tuttavia gli stessi sono rappresentabili attraverso una sola cifra nel sistema numerico ottale (Base 8). Lo stesso principio si estende a 4 cifre in relazione alla base esadecimale (Base 16). Infatti, considerando il sistema numerico binario a 4 cifre, possiamo rappresentare $2^4 = 16$ numeri diversi, gli stessi possono essere codificati con un unico simbolo in base esadecimale.

Di conseguenza dovendo trasformare un numero binario in un numero ottale (o esadecimale) possiamo semplicemente raggruppare i bit a gruppi di 3 (o 4) e scrivere il loro corrispettivo valore in Base 8 (o Base 16). Possiamo sfruttare questa proprietà per velocizzare la conversione. Volendo ora applicare il procedimento inverso, ossia passare da Base 8 (o Base 16) a Base 2, basterà trasformare ogni cifra del numero nel corrispettivo binario ed esprimerla su 3 (o 4) bit.

Esercizi

Tradurre i seguenti numeri da ottale a binario:

(a) 1242_8

(b) 364_8

(c) 673_8

(d) 371_8

(e) 536_8

(f) 7325_8

Tradurre i seguenti numeri da esadecimale a binario:

(a) $E3D_{16}$

(b) $AB4_{16}$

(c) $F8C_{16}$

(d) 962_{16}

(e) $34A_{16}$

(f) $BF4_{16}$

Tradurre i seguenti numeri da binario a ottale:

(a) $100011111_2 \rightarrow$ Base 8

(b) $011110011_2 \rightarrow$ Base 8

(c) $101010111_2 \rightarrow$ Base 8

Tradurre i seguenti numeri da binario a esadecimale:

(a) $111101011010_2 \rightarrow$ Base 16

(b) $101111100100_2 \rightarrow$ Base 16

(c) $011010010010_2 \rightarrow$ Base 16

Soluzioni

Soluzioni da ottale a binario:

(a) 1242_8=($1_{\underline{001}}2_{\underline{010}}4_{\underline{100}}2_{\underline{010}}$)=001 010 100 010_2

(b) 364_8=($3_{\underline{011}}6_{\underline{110}}4_{\underline{100}}$)=011 110 100_2

(c) 673_8=($6_{\underline{110}}7_{\underline{111}}3_{\underline{011}}$)=110 111 011_2

(d) 371_8=($3_{\underline{011}}7_{\underline{111}}1_{\underline{001}}$)=011 111 001_2

(e) 536_8=($5_{\underline{101}}3_{\underline{011}}6_{\underline{110}}$)=101 011 110_2

(f) 7325_8=($7_{\underline{111}}3_{\underline{011}}2_{\underline{010}}5_{\underline{101}}$)=111 011 010 101_2

Soluzioni da esadecimale a binario:

(a) $E3D_{16}$=$(E_{\underline{1110}}3_{\underline{0011}}D_{\underline{1101}})$=$1110\ 0011\ 1101_2$

(b) $AB4_{16}$=$(A_{\underline{1010}}B_{\underline{1011}}4_{\underline{0100}})$=$1010\ 1011\ 0100_2$

(c) $F8C_{16}$=$(F_{\underline{1111}}8_{\underline{1000}}C_{\underline{1100}})$=$1111\ 1000\ 1100_2$

(d) 962_{16}=$(9_{\underline{1001}}6_{\underline{0110}}2_{\underline{0010}})$=$1001\ 0110\ 0010_2$

(e) $34A_{16}$=$(3_{\underline{0011}}4_{\underline{0100}}A_{\underline{1010}})$=$0011\ 0100\ 1010_2$

(f) $BF4_{16}$=$(B_{\underline{1011}}F_{\underline{1111}}4_{\underline{0100}})$=$1011\ 1111\ 0100_2$

Soluzioni da binario a ottale:

(a) 100011111_2=$(100_{\underline{4}}011_{\underline{3}}111_{\underline{7}})$=$4\ 3\ 7_8$

(b) 011110011_2=$(011_{\underline{3}}110_{\underline{6}}011_{\underline{3}})$=$3\ 6\ 3_8$

(c) 101010111_2=$(101_{\underline{5}}010_{\underline{2}}111_{\underline{7}})$=$5\ 2\ 7_8$

Soluzioni da binario a esadecimale:

(a) 111101011010_2=$(1111_{\underline{15=F}}0101_{\underline{5}}1010_{\underline{10=A}})$=$F\ 5\ A_{16}$

(b) 101111100100_2=$(1011_{\underline{11=B}}1110_{\underline{14=E}}0100_{\underline{4}})$=$B\ E\ 4_{16}$

(c) 011010010010_2=$(0110_{\underline{6}}1001_{\underline{9}}0010_{\underline{2}})$=$6\ 9\ 2_{16}$

2 Somma tra numeri binari senza segno

Le operazioni di somma sono eseguite in colonna bit a bit applicando un procedimento analogo a quello per le somme in decimale su carta. La somma di numeri binari senza segno produce overflow se e soltanto se il risultato è troppo grande per essere rappresentato nel numero finito di bit messo a disposizione (ossia se il riporto più significativo è a 1).

Esercizi

Eseguire le seguenti somme tra numeri binari senza segno su 8 bit. Discutere l'overflow motivando la risposta. Se i numeri sono espressi in decimale eseguire prima la traduzione in binario.

(a) 11010110_2+00111010_2

(b) 11011100_2+00011111_2

(c) 00111111_2+11000000_2

(d) 136_{10}+192_{10}

(e) 136_{10}+35_{10}

(f) 63_{10}+44_{10}

Soluzioni

N. B.: Ogni somma bit a bit scrive sopra di essa il riporto generato.

(a) 11010110_2+00111010_2=00010000_2 ⇒ ultimo riporto uguale a 1 ⇒ overflow

$$
\begin{array}{ccccccccc}
\scriptstyle 1 & \scriptstyle 1 & \scriptstyle 1 & \scriptstyle 1 & \scriptstyle 1 & \scriptstyle 1 & \scriptstyle 1 & \scriptstyle 0 & \\
 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 \\
+ & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 \\
\hline
 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0
\end{array}
$$

(b) 11011100_2+00011111_2=11111011_2 ⇒ ultimo riporto uguale a 0 ⇒ no overflow

$$
\begin{array}{ccccccccc}
\scriptstyle 0 & \scriptstyle 0 & \scriptstyle 0 & \scriptstyle 1 & \scriptstyle 1 & \scriptstyle 1 & \scriptstyle 0 & \scriptstyle 0 & \\
 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\
+ & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\
\hline
 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1
\end{array}
$$

(c) 00111111_2+11000000_2=11001000_2 ⇒ ultimo riporto uguale a 0 ⇒ no overflow

$$
\begin{array}{ccccccccc}
\scriptstyle 0 & \scriptstyle 0 & \scriptstyle 0 & \scriptstyle 0 & \scriptstyle 0 & \scriptstyle 0 & \scriptstyle 0 & \scriptstyle 0 & \\
 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\
+ & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
\hline
 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1
\end{array}
$$

(d) 136_{10}+192_{10}=10001000_2+11000000_2=11001000_2 ⇒ ultimo riporto uguale a 1 ⇒ overflow

$$
\begin{array}{ccccccccc}
\scriptstyle 1 & \scriptstyle 0 & \scriptstyle 0 & \scriptstyle 0 & \scriptstyle 0 & \scriptstyle 0 & \scriptstyle 0 & \scriptstyle 0 & \\
 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
+ & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
\hline
 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0
\end{array}
$$

(e) 136_{10}+35_{10}=10001000_2+00100011_2=10101011_2 ⇒ ultimo riporto uguale a 0 ⇒ no overflow

$$
\begin{array}{cccccccccc}
 & {\scriptstyle 0} & {\scriptstyle 0} & {\scriptstyle 0} & {\scriptstyle 0} & {\scriptstyle 0} & {\scriptstyle 0} & {\scriptstyle 0} & {\scriptstyle 0} & \\
 & & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
+ & & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \\
\hline
 & & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1
\end{array}
$$

(f) 63_{10}+44_{10}=00111111_2+00101100_2=01101011_2 $\Rightarrow$ ultimo riporto uguale a 1 $\Rightarrow$ no overflow

$$
\begin{array}{cccccccccc}
 & {\scriptstyle 0} & {\scriptstyle 0} & {\scriptstyle 0} & {\scriptstyle 0} & {\scriptstyle 0} & {\scriptstyle 0} & {\scriptstyle 0} & {\scriptstyle 0} & \\
 & & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\
+ & & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \\
\hline
 & & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1
\end{array}
$$

3 Somma e sottrazione tra numeri binari con segno

Considerando le operazioni tra numeri binari con segno le cose si complicano leggermente, ad esempio per eseguire sottrazioni del tipo $A - B$ conviene riscrivere l'operazione come una addizione nel seguente modo:

$$A + (-B)$$

Dobbiamo solo capire come esprimere $-B$ in binario.

La notazione scelta per la rappresentazione dei numeri con segno è quella basata sul **complemento a due**. Dato un numero positivo n (con bit di segno uguale a 0), per ricavare il corrispettivo negativo $-n$ possiamo utilizzare uno dei due algoritmi di cambio segno:

- Complemento a 1 del numero n e somma 1
- Complemento a 1 di tutti i bit a sinistra della cifra 1 meno significativa

Per calcolare il valore corrispondente in decimale usiamo la seguente formula:

$$-1 \cdot 2^{n-1} + d_{n-2} \cdot 2^{n-2} + \cdots + d_1 \cdot 2^1 + d_0 \cdot 2^0$$

Infine ricordiamo che anche nel caso di operazioni tra numeri con segno esiste la possibilità di generare overflow. La tabella riassuntiva è la seguente:

tipo di operazione

segni degli operandi

	somma	sottrazione
pos. pos.	SI (neg.)	NO
neg. neg.	SI (pos.)	NO
pos. neg.	NO	SI (neg.)
neg. pos.	NO	SI (pos.)

Figure 1: Tabella riassuntiva overflow.

Esercizi

Tradurre in complemento a due su 8 bit i numeri decimali che seguono ed eseguire le operazioni indicate sempre su 8 bit. Verificare se vi è overflow motivando la risposta:

(a) $110_{10} + 120_{10}$

(b) $110_{10} - 120_{10}$

(c) $-35_{10} - 44_{10}$

(d) $-127_{10} - 8_{10}$

Soluzioni

(a) Convertiamo i numeri da decimale a binario $+110_{10} = 01101110_2$ mentre $+120_{10} = 01111000_2$. Osserviamo che entrambi i numeri sono rappresentabili in complemento a due su 8 bit. Proseguiamo eseguendo la somma:

$$\begin{array}{ccccccccc} \scriptstyle 0 & \scriptstyle 1 & \scriptstyle 1 & \scriptstyle 1 & \scriptstyle 1 & \scriptstyle 0 & \scriptstyle 0 & \scriptstyle 0 & \\ & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 \\ + & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ \hline & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \end{array}$$

Il segno dei due addendi è concorde (somma di due positivi), ma otteniamo un numero negativo, di conseguenza vi è overflow. Infatti $120_{10} + 110_{10} =$

230_{10} che non è esprimibile su 8 bit in complemento a due. Ricordiamo che il numero massimo positivo esprimibile in complemento a due su 8 bit corrisponde a $2^7 - 1 = 127_{10}$. In alternativa possiamo applicare la regola dei riporti (ultimi due più significativi), che in questo caso sono discordi, ossia vi è overflow.

(b) Convertiamo i numeri da decimale a binario $+110_{10} = 01101110_2$ mentre $+120_{10} = 01111000_2$. Osserviamo che entrambi i numeri sono rappresentabili in complemento a due su 8 bit. Riscriviamo l'operazione in:

$$110_{10} + (-120_{10})$$

Ricaviamo l'espressione binaria in complemento a due di -120_{10} l'algoritmo di cambio di segno:

$$01111000_2 \rightarrow 10001000_2$$

Eseguiamo la somma:

$$\begin{array}{ccccccccc}
 & {\scriptstyle 0} & {\scriptstyle 0} & {\scriptstyle 0} & {\scriptstyle 0} & {\scriptstyle 1} & {\scriptstyle 0} & {\scriptstyle 0} & {\scriptstyle 0} \\
 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 \\
+ & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
\hline
 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 0
\end{array}$$

Il segno dei due addendi è discorde non possiamo avere overflow. In alternativa possiamo applicare la regola dei riporti (ultimi due più significativi), che in questo caso sono concordi, ossia non vi è overflow. Per verificare la correttezza del risultato convertiamolo in decimale (ci aspettiamo $110_{10} - 120_{10} = -10_{10}$).

$$-2^7+2^6+2^5+2^4+2^2+2^1 = -128+64+32+16+4+2 = -128+118 = -10_{10}$$

(c) Convertiamo i numeri da decimale a binario $+35_{10} = 00100011_2$ mentre $+44_{10} = 00101100_2$. Osserviamo che entrambi i numeri sono rappresentabili in complemento a due su 8 bit. Riscriviamo la sottrazione in:

$$-35_{10} + (-44_{10})$$

Ricaviamo l'espressione binaria in complemento a due di -35_{10} e di -44_{10} eseguendo l'algoritmo di cambio di segno:

$$00100011_2 \rightarrow 11011101_2$$

$$00101100_2 \rightarrow 11010100_2$$

Eseguiamo la somma:

$$
\begin{array}{rcccccccc}
 & {\scriptstyle 1} & {\scriptstyle 1} & {\scriptstyle 0} & {\scriptstyle 1} & {\scriptstyle 1} & {\scriptstyle 1} & {\scriptstyle 0} & {\scriptstyle 0} \\
 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 \\
+ & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\
\hline
 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1
\end{array}
$$

Il segno dei due addendi è concorde (somma di due negativi) ed anche il risultato della somma è negativo, di conseguenza non vi è overflow. In alternativa possiamo applicare la regola dei riporti (ultimi due più significativi), che in questo caso sono concordi, ossia non vi è overflow. Per verificare l'esattezza del risultato convertiamolo in decimale (ci aspettiamo $-35_{10} - 44_{10} = -79_{10}$)

$$-2^7 + 2^5 + 2^4 + 2^0 = -128 + 32 + 16 + 1 = -128 + 49 = -79_{10}$$

(d) Convertiamo i numeri da decimale a binario $+127_{10} = 01111111_2$ mentre $+8_{10} = 00001000_2$. Osserviamo che entrambi i numeri sono rappresentabili in complemento a due su 8 bit. Riscriviamo la sottrazione in:

$$-127_{10} + (-8_{10})$$

Ricaviamo l'espressione binaria in complemento a due di -127_{10} e di -8_{10} eseguendo l'algoritmo di cambio di segno:

$$01111111_2 \rightarrow 10000001_2$$

$$00001000_2 \rightarrow 11111000_2$$

Eseguiamo la somma:

$$
\begin{array}{rcccccccc}
 & {\scriptstyle 1} & {\scriptstyle 0} & {\scriptstyle 0} & {\scriptstyle 0} & {\scriptstyle 0} & {\scriptstyle 0} & {\scriptstyle 0} & {\scriptstyle 0} \\
 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
+ & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\
\hline
 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1
\end{array}
$$

Il segno dei due addendi è concorde (somma di due negativi), ma il risultato della somma è positivo, di conseguenza vi è overflow. Infatti $-127_{10} - 8_{10} = -135_{10}$ che non è esprimibile su 8 bit in complemento a due. Ricordiamo che il numero massimo negativo rappresentabile in complemento a due con 8 bit corrisponde a $-2^7 = -128_{10}$. In alternativa possiamo applicare la regola dei riporti (ultimi due più significativi), che in questo caso sono discordi, ossia vi è overflow.