

# **Sperimentazioni di Fisica I**

## **mod. A – Lezione 2**

### **Numeri Relativi**

*Dipartimento di Fisica e Astronomia “G. Galilei”,  
Università degli Studi di Padova*

# **Rappresentazione dei Numeri**

## **Lezione II: Numeri Relativi**

### **1. Introduzione**

# Complemento alla Radice

Dato un numero  $x$  espresso in base- $R$  con un numero  $n$  di cifre, si definisce **complemento alla radice  $R$** , il numero:

$$C_R(x) = R^n - x$$

Allo stesso modo si definisce il **complemento alla radice  $R$  diminuita di uno**, il numero:

$$C_{R-1}(x) = (R^n - 1) - x = R^n - 1 - x = (R^n - x) - 1$$

$C_R$  e  $C_{R-1}$  sono legati dalla relazione:

$$C_{R-1}(x) = C_R(x) - 1$$

$$C_R(x) = C_{R-1}(x) + 1$$

# Complementi in Base-10

In **base-10**, si definiscono il **complemento a 10** ( $C_{10}$ , complemento alla radice 10) e il **complemento a 9** ( $C_9$ , complemento alla radice 10 diminuita di 1).

Consideriamo il numero  **$873_{10}$**

$$C_{10}(873) = 10^3 - 873 = 1000 - 873 = 127$$

$$C_9(873) = (10^3 - 1) - 873 = 999 - 873 = 126$$

$$C_{10}(873) = C_9(873) + 1 = 126 + 1 = 127$$

Altro esempio, consideriamo il numero  **$75911_{10}$**

$$C_{10}(75911) = 10^5 - 75911 = 100000 - 75911 = 24089$$

$$C_9(75911) = (10^5 - 1) - 75911 = 99999 - 75911 = 24088$$

# Complementi in altre Basi

## 1. Complemento alla radice R diminuita di uno, $C_{R-1}(x)$

$$C_F(369_{16}) = FFF_{16} - 369_{16} = C96_{16}$$

$$C_9(873_{10}) = 999_{10} - 873_{10} = 126_{10}$$

$$C_7(1551_8) = 7777_8 - 1551_8 = 6226_8$$

$$C_1(1101101001_2) = 1111111111_2 - 1101101001_2 = 0010010110_2$$

## 2. Complemento alla radice R, $C_R(x)$

$$C_{16}(369_{16}) = C_F(369_{16}) + 1_{16} = C96_{16} + 1_{16} = C97_{16}$$

$$C_{10}(873_{10}) = C_9(873_{10}) + 1_{10} = 126_{10} + 1_{10} = 127_{10}$$

$$C_8(1551_8) = C_7(1551_8) + 1_8 = 6226_8 + 1_8 = 6227_8$$

$$C_2(1101101001_2) = C_1(1101101001_2) + 1_2 = 0010010111_2$$

# **Rappresentazione dei Numeri**

## **Lezione II: Numeri Relativi**

### **2. La Rappresentazione del Segno**

# Il Segno

Il parametro che contraddistingue due numeri con lo stesso valore assoluto è il **segno**.

Il segno può assumere unicamente **due valori**: “+” e “-”

Per rappresentare l’informazione-segno è necessario e sufficiente **un bit**, una cifra che possa assumere valori 0 ed 1.

Aggiungiamo una cifra  $d_n$ , che rappresenta il segno

$$d = (d_n d_{n-1} d_{n-2} \dots d_2 d_1 d_0)_R$$

dove  $d_i \in \{0, 1, \dots, R-1\}$ ,  $i \in \mathbb{N}$  e

$d_n = 0$ , se il numero è **positivo**

$d_n = 1$ , se il numero è **negativo**

$$d = (1 - 2d_n) \sum_{i=0}^{n-1} d_i R^i$$

# I Numeri Relativi in Base-2

**Interi Positivi.** Si aggiunge **uno zero (0)** come cifra più significativa ( $d_n = 0$ ), mentre il resto delle cifre rimane invariato ( $d_n d_{n-1} \dots d_1 d_0$ ).

$$\begin{aligned} |19_{10}| &= 10011_2 \\ +19_{10} &= \mathbf{0}10011_2 \end{aligned}$$

**Interi Negativi.** Ci sono diverse possibili rappresentazioni:

1. Rappresentazione **Modulo e Segno**,
2. Rappresentazione in **Complemento ad Uno**,
3. Rappresentazione in **Complemento a Due**.
4. Rappresentazione in **Eccesso-q**



# **Rappresentazione dei Numeri**

## **Lezione II: Numeri Relativi**

### **3. Rappresentazione**

#### **Modulo e Segno**

# Modulo e Segno (I)

Un primo approccio al problema è di **separare** nettamente **il modulo** (valore assoluto) dell'intero rappresentato e di riservare il **bit più significativo** per la rappresentazione del **segno**.

Tale bit avrà valore **0** per i numeri **positivi** e **1** per i numeri **negativi**.

$$|19_{10}| = 10011_2$$

$$+19_{10} = 010011_2$$

$$-19_{10} = 110011_2$$

Fissato il **numero di bit disponibili** per la rappresentazione di un numero, resta individuato l'**intervallo di valori rappresentabili**.

# Modulo e Segno (II)

Supponiamo di descrivere un intero con **n bit**, **1 bit** sarà riservato al **segno** e **n-1** al suo **modulo**. L'intervallo rappresentabile sarà

$$[-2^{n-1} + 1, 2^{n-1} - 1]$$

Vediamo un esempio con **n = 8**,  $[-2^7 + 1, 2^7 - 1] = [-127, +127]$

Sistema Binario	Signed (8-bit)	Unsigned (8-bit)
00000000 <sub>2</sub>	+0 <sub>10</sub>	0 <sub>10</sub>
00000001 <sub>2</sub>	+1 <sub>10</sub>	1 <sub>10</sub>
01111111 <sub>2</sub>	+127 <sub>10</sub>	127 <sub>10</sub>
10000000 <sub>2</sub>	-0 <sub>10</sub>	128 <sub>10</sub>
10000001 <sub>2</sub>	-1 <sub>10</sub>	129 <sub>10</sub>
11111111 <sub>2</sub>	-127 <sub>10</sub>	255 <sub>10</sub>

Rappresentazione ridondante per la presenza di **due “0”**: **+0** e **-0**.

Utilizzata inizialmente nei primi computer (IBM 7090).

Oggi usata per il **significante/mantissa** nei numeri a **virgola mobile**.

# **Rappresentazione dei Numeri**

## **Lezione II: Numeri Relativi**

### **4. Rappresentazione Complemento ad Uno**

# Complemento ad Uno

Ricordiamo la definizione di **complemento ad uno** (alla radice R diminuita di uno in base-2) di un numero x rappresentato con n bit:

$$C_{R-1}(x_R) = R^n - 1_R - x_R$$

$$C_1(x_2) = 2_2^n - 1_2 - x_2$$

Per passare da un numero al suo **negativo**, si utilizza l'operazione di **complemento ad uno**.

La stessa trasformazione permette di passare da un numero negativo al corrispondente **positivo**:

$$\begin{aligned} C_1(C_1(x_2)) &= 2_2^n - 1_2 - C_1(x_2)_2 = \\ &= 2_2^n - 1_2 - (2_2^n - 1_2 - x_2) = \\ &= 2_2^n - 1_2 - 2_2^n + 1_2 + x_2 = \\ &= x_2 \end{aligned}$$

# Regola Pratica

**Intero Positivo:** si aggiunge un **bit di segno (0)** come cifra più significativa.

**Intero Negativo:** si considera il **complemento ad uno** dell'intero positivo.

$$|19_{10}| = 10011_2$$

$$+19_{10} = 010011_2$$

$$-19_{10} = C_1(010011_2) = 111111_2 - 010011_2 = 101100_2$$

→ Per trovare la rappresentazione in complemento ad uno, **invertire ad uno ad uno tutti i bit** della parola.

L'**intervallo** di valori rappresentato è lo stesso della rappresentazione Modulo-Segno:  $[-2^{n-1} + 1, +2^{n-1} - 1]$ .

Per esempio, per **n = 8**:  $[-127_{10}, +127_{10}]$ .

# Numeri Rappresentati in C<sub>1</sub>

<b>Sistema Binario</b>	<b>Signed (8-bit)</b>	<b>Unsigned (8-bit)</b>
00000000 <sub>2</sub>	+0 <sub>10</sub>	0 <sub>10</sub>
00000001 <sub>2</sub>	+1 <sub>10</sub>	1 <sub>10</sub>
01111101 <sub>2</sub>	+125 <sub>10</sub>	125 <sub>10</sub>
01111110 <sub>2</sub>	+126 <sub>10</sub>	126 <sub>10</sub>
01111111 <sub>2</sub>	+127 <sub>10</sub>	127 <sub>10</sub>
10000000 <sub>2</sub>	-127 <sub>10</sub>	128 <sub>10</sub>
10000001 <sub>2</sub>	-126 <sub>10</sub>	129 <sub>10</sub>
10000010 <sub>2</sub>	-125 <sub>10</sub>	130 <sub>10</sub>
11111101 <sub>2</sub>	-2 <sub>10</sub>	253 <sub>10</sub>
11111110 <sub>2</sub>	-1 <sub>10</sub>	254 <sub>10</sub>
11111111 <sub>2</sub>	-0 <sub>10</sub>	255 <sub>10</sub>

# Alcuni Inconvenienti

La rappresentazione è **ridondante** per la scrittura dello “0”:

$$00\dots00_2 = +0_{10}$$

$$11\dots11_2 = -0_{10}$$

Il complemento ad uno permetterebbe un **design più semplice dell'hardware** (es. i circuiti di addizione e sottrazione sono più semplici rispetto alla rappresentazione modulo e segno), ma la duplice rappresentazione dello “0” può generare degli inconvenienti nelle operazioni.

Nelle **sottrazioni**, come vedremo, possono inoltre verificarsi degli **errori** se non si considera opportunamente il **riporto**.

CDC 6000 e Univac 1100 utilizzarono il complemento ad uno.

Successivamente **Intel** fu tra i primi ad integrare il **complemento a due** (Intel 8080). La scelta fu quindi seguita anche da AMD ed IBM.



# Addizione

Consideriamo la somma di  $A_1 = +121_{10}$  e  $A_2 = -55_{10}$

$$|A_1| = 121_{10} = 1111001_2$$

$$A_1 = +121_{10} = 01111001_2$$

$$|A_2| = 55_{10} = 110111_2$$

$$+|A_2| = +55_{10} = 00110111_2$$

$$A_2 = -55_{10} = C_1(00110111)_2 = 11001000_2$$

11111

$$01111001_2 + A_1 + A_2 = 01000001_2 = 64_{10} + 1_{10} = 65_{10}$$

$$11001000_2 = \text{ma } A_1 + A_2 = 121_{10} - 55_{10} = 66_{10} !!!$$

occorre **sommare l'overflow** per ottenere

il risultato corretto:  $65_{10} + 1_2 = 66_{10}$

---

$$(1)01000001_2$$

# **Rappresentazione dei Numeri**

## **Lezione II: Numeri Relativi**

### **5. Rappresentazione**

### **Complemento a Due**

# Complemento a Due

I problemi relativi alla **doppia rappresentazione dello “0”** e del **riporto dell’overflow nelle sottrazioni** sono evitati con la rappresentazione dei numeri negativi in **complemento a due**.

$$C_R(x_R) = R_R^n - x_R$$

$$C_2(x_2) = 2_2^n - x_2 = C_1(x_2) + 1$$

Ci sono **due regole pratiche** per il calcolo del complemento a due:

1. Calcolare il **complemento ad 1,  $C_1(x_2)$** , invertendo tutti i bit, e poi **sommare 1** al risultato;

$$C_2(0101100_2) = 1010011_2 + 1_2 = 1010100_2$$

2. Partendo da destra, **copiare tutti i bit fino al primo “1”** e poi **invertire tutti i bit a sinistra del primo “1”**

$$C_2(\mathbf{0101100}_2) = \mathbf{1010100}_2$$

# Numeri Rappresentati in $C_2$

Sistema Binario	Signed (8-bit)	Unsigned (8-bit)
$00000000_2$	$0_{10}$	$0_{10}$
$00000001_2$	$+1_{10}$	$1_{10}$
$01111101_2$	$+125_{10}$	$125_{10}$
$01111110_2$	$+126_{10}$	$126_{10}$
$01111111_2$	$+127_{10}$	$127_{10}$
$10000000_2$	$-128_{10}$	$128_{10}$
$10000001_2$	$-127_{10}$	$129_{10}$
$10000010_2$	$-126_{10}$	$130_{10}$
$11111101_2$	$-3_{10}$	$253_{10}$
$11111110_2$	$-2_{10}$	$254_{10}$
$11111111_2$	$-1_{10}$	$255_{10}$

# Confronto $C_1$ e $C_2$

<b>Sistema Binario</b>	<b><math>C_1</math> (8-bit)</b>	<b><math>C_2</math> (8-bit)</b>
00000000 <sub>2</sub>	+0 <sub>10</sub>	0 <sub>10</sub>
00000001 <sub>2</sub>	+1 <sub>10</sub>	+1 <sub>10</sub>
01111101 <sub>2</sub>	+125 <sub>10</sub>	+125 <sub>10</sub>
01111110 <sub>2</sub>	+126 <sub>10</sub>	+126 <sub>10</sub>
01111111 <sub>2</sub>	+127 <sub>10</sub>	+127 <sub>10</sub>
10000000 <sub>2</sub>	-127 <sub>10</sub>	-128 <sub>10</sub>
10000001 <sub>2</sub>	-126 <sub>10</sub>	-127 <sub>10</sub>
10000010 <sub>2</sub>	-125 <sub>10</sub>	-126 <sub>10</sub>
11111101 <sub>2</sub>	-2 <sub>10</sub>	-3 <sub>10</sub>
11111110 <sub>2</sub>	-1 <sub>10</sub>	-2 <sub>10</sub>
11111111 <sub>2</sub>	-0 <sub>10</sub>	-1 <sub>10</sub>

# Proprietà in $C_2$

È il **metodo più diffuso** nella rappresentazione dei numeri negativi in **informatica**.

Si utilizza un **unico circuito** per **addizione** e **sottrazione**, permettendo tecnologie più semplici e maggior precisione.

Il **valore assoluto** di un **numero negativo** si ottiene **invertendo** il valore dei singoli bit e **sommando 1** al numero binario risultante

(unica eccezione  $10000000_2 = -128_{10}$ )

$$\begin{aligned} -5_{10} &= 11111011_2 \rightarrow C_2(11111011_2) = C_1(11111011_2) + 1_2 = \\ &= 00000100_2 + 1_2 = 00000101_2 = 4_{10} * 1_{10} + 1_{10} = 5_{10} \end{aligned}$$

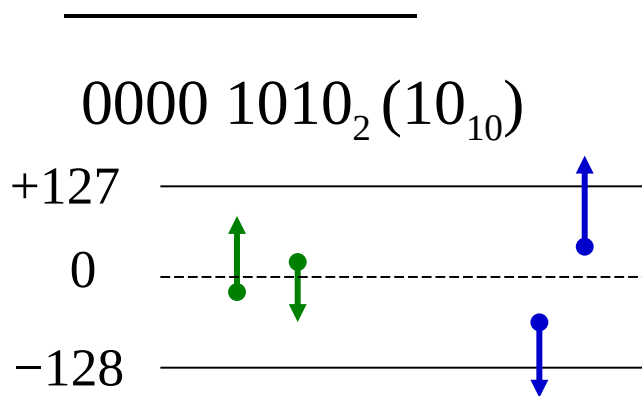
L'**intervallo** dei numeri rappresentati è  $[-2^{n-1}, +2^{n-1} - 1]$ , per esempio per una rappresentazione in **n = 8** bit:  $[-128, 127]$ .

C'è un'**unica rappresentazione** dello **0**.

# Addizione

Sommando due numeri di **segno diverso**, il **segno** del risultato è determinato **automaticamente**.

$$\begin{array}{r} \textcircled{11} \text{ } 111 \text{ } 111 \\ 0000 \text{ } 1111_2 + (+15_{10}) \\ 1111 \text{ } 1011_2 = (-5_{10}) \end{array}$$



Sommando due numeri dello **stesso segno**, occorre determinare se il risultato è **fuori intervallo**.

$$\begin{array}{r} \textcircled{01} \text{ } 1 \\ 0100 \text{ } 1011_2 + (+75_{10}) \\ 0100 \text{ } 0010_2 = (+66_{10}) \end{array}$$

---

1000 1101<sub>2</sub> (-115<sub>10</sub>) ( $\neq 141_{10}$ )

Se gli **ultimi due riporti** sono **diversi**, si è verificata una condizione di **overflow**!

# **Rappresentazione dei Numeri**

## **Lezione II: Numeri Relativi**

### **6. Rappresentazione in Eccesso-q**



# Eccesso-q

Questa notazione, detta anche **“biased”**, si realizza **aggiungendo** un valore fissato **“q”** (detto eccesso, **offset**) ai numeri della rappresentazione.

Dato il numero di bit, **n**, della rappresentazione **l'eccesso è** (generalmente)

$$q = 2^{n-1} - 1$$

Lo **“0”** è rappresentato dal numero **“q”**, mentre una combinazione di **bit tutti nulli** rappresenta il numero **“-q”**.

**Intervallo:  $[-2^{n-1} + 1, 2^{n-1}]$**

Tale notazione è utilizzata per indicare l'**esponente** nei **numeri in virgola mobile**. Per esempio, secondo lo standard IEEE-754, nei numeri in precisione singola (**32-bit**) l'esponente utilizza 8 bit ed è rappresentato in **eccesso-127**. Nei numeri in precisione doppia (**64-bit**), sono riservati 11 bit all'esponente rappresentato in **eccesso-1023**.

# Numeri Rappresentati in Eccesso-127

**Sistema Binario**

**Offset-127 (8-bit)**

**Unsigned (8-bit)**

**00000000<sub>2</sub>**

**-127<sub>10</sub>**

**0<sub>10</sub>**

00000001<sub>2</sub>

-126<sub>10</sub>

1<sub>10</sub>

01111101<sub>2</sub>

-2<sub>10</sub>

125<sub>10</sub>

01111110<sub>2</sub>

-1<sub>10</sub>

126<sub>10</sub>

**01111111<sub>2</sub>**

**0<sub>10</sub>**

**127<sub>10</sub>**

10000000<sub>2</sub>

1<sub>10</sub>

128<sub>10</sub>

10000001<sub>2</sub>

2<sub>10</sub>

129<sub>10</sub>

10000010<sub>2</sub>

3<sub>10</sub>

130<sub>10</sub>

11111110<sub>2</sub>

127<sub>10</sub>

254<sub>10</sub>

**11111111<sub>2</sub>**

**128<sub>10</sub>**

**255<sub>10</sub>**

# Esercizio 1 (I)

Dato il numero decimale  $x = -80_{10}$ , rappresentarlo in base binaria nelle seguenti notazioni, utilizzando (complessivamente) 8 bit:

1. **Modulo e Segno MS in base due**
2. **Complemento ad Uno  $C_1$  in base due**
3. **Complemento a Due  $C_2$  in base due**
4. **Eccesso-q**


**Passo 1.** Conversione del numero  $x$  in base 2.

$$80 = \dots$$

$$= 64 + 16 = 2^6 + 2^4 =$$

$$1 \cdot 2^6 + 0 \cdot 2^5 + 1 \cdot 2^4 + 0 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 0 \cdot 2^0$$

$$80_{10} = 1010000_2$$

$$\begin{array}{l} 80 / 2 = 40 + 0 / 2 \\ 40 / 2 = 20 + 0 / 2 \\ 20 / 2 = 10 + 0 / 2 \\ 10 / 2 = 5 + 0 / 2 \\ 5 / 2 = 2 + 1 / 2 \\ 2 / 2 = 1 + 0 / 2 \\ 1 / 2 = 0 + 1 / 2 \end{array}$$


# Esercizio 1 (II)

**Passo 2.** Quanti bit abbiamo a disposizione? 8

Quanti bit servono per rappresentare il modulo di x? 7

**Passo 3. Modulo e Segno**

Modulo: 1010000

Segno: 1 (negativo)

→ **1-1010000**

**Passo 4. Complemento ad Uno ( $C_1$ )**

Invertiamo tutti i bit del Modulo

→ **1-0101111**

# Esercizio 1 (III)

## Passo 5. Complemento a Due ( $C_2$ )

Sommiamo 1 al Complemento ad 1:  $10101111 + 1$

→ **1-0110000**

## Passo 6. Eccesso $-q$

Quanto vale  $q$ ?

$$q = 2^{n-1} - 1 = 2^7 - 1 = 128 - 1 = 127$$

Per rappresentare  $x$  in eccesso  $-127$ , dobbiamo sommare  $q = 127$  a  $x = -80$  e convertire il numero risultante ( $-80 + 127 = 47$ ) dalla base decimale alla base binaria.

→ **0-0101111**

# Esercizio 2

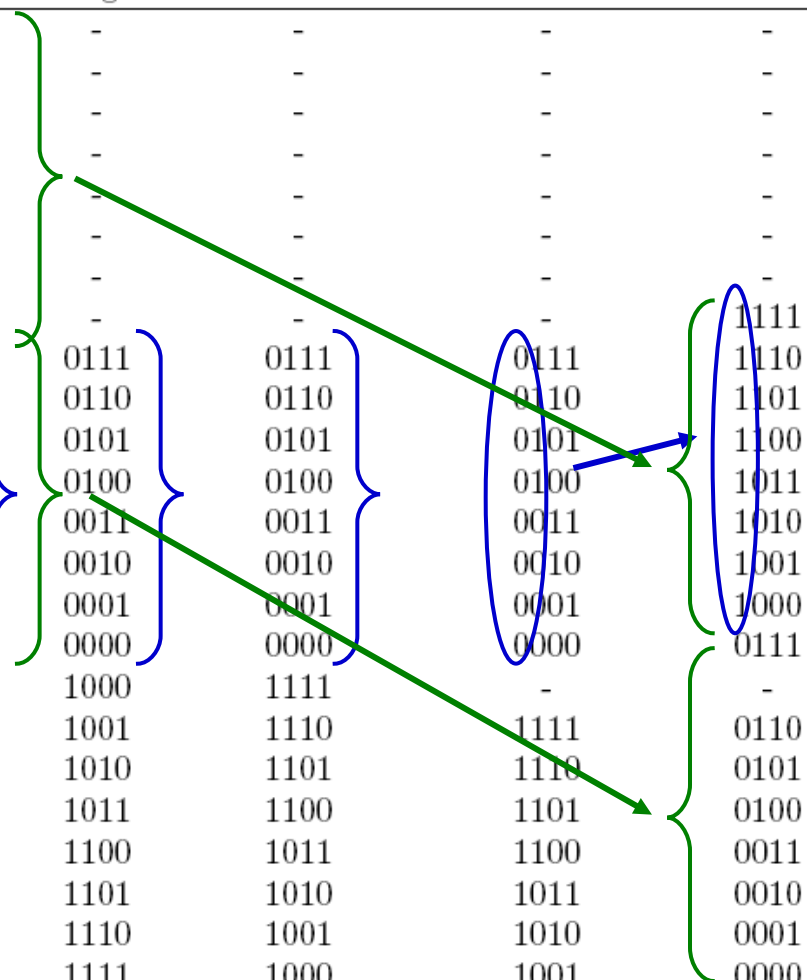
Dato il numero decimale  $x = -153_{10}$ , rappresentarlo in base binaria nelle seguenti notazioni, utilizzando (complessivamente) 9 bit:

1. **Modulo e Segno MS in base due**
2. **Complemento ad Uno  $C_1$  in base due**
3. **Complemento a Due  $C_2$  in base due**
4. **Eccesso-q**

Soluzione: Modulo e Segno:	1-10011001
Complemento ad Uno:	1-01100110
Complemento a Due:	1-01100111
Eccesso-255:	0-01100110

# Confronto tra le Rappresentazioni (I)

$R = 10$	$R = 2$				
	Valore Assoluto	Modulo e Segno	Complemento a Uno	Complemento a Due	Eccesso -7
+15	1111	-	-	-	-
+14	1110	-	-	-	-
+13	1101	-	-	-	-
+12	1100	-	-	-	-
+11	1011	-	-	-	-
+10	1010	-	-	-	-
+9	1001	-	-	-	-
+8	1000	-	-	-	-
+7	0111	0111	0111	0111	1111
+6	0110	0110	0110	0110	1110
+5	0101	0101	0101	0101	1101
+4	0100	0100	0100	0100	1100
+3	0011	0011	0011	0011	1011
+2	0010	0010	0010	0010	1010
+1	0001	0001	0001	0001	1001
+0	0000	0000	0000	0000	1000
-0	-	1000	1111	-	0111
-1	-	1001	1110	1111	-
-2	-	1010	1101	1110	0110
-3	-	1011	1100	1101	0101
-4	-	1100	1011	1100	0100
-5	-	1101	1010	1011	0011
-6	-	1110	1001	1010	0010
-7	-	1111	1000	1001	0001
-8	-	-	-	1000	0000



# Confronto tra le Rappresentazioni (II)

$R = 10$	$R = 2$				
	Valore Assoluto	Modulo e Segno	Complemento a Uno	Complemento a Due	Eccesso -7
+15	1111	-	-	-	-
+14	1110	-	-	-	-
+13	1101	-	-	-	-
+12	1100	-	-	-	-
+11	1011	-	-	-	-
+10	1010	-	-	-	-
+9	1001	-	-	-	-
+8	1000	-	-	-	1111
+7	0111	0111	0111	0111	1110
+6	0110	0110	0110	0110	1101
+5	0101	0101	0101	0101	1100
+4	0100	0100	0100	0100	1011
+3	0011	0011	0011	0011	1010
+2	0010	0010	0010	0010	1001
+1	0001	0001	0001	0001	1000
+0	0000	0000	0000	0000	0111
-0	-	1000	1111	-	-
-1	-	1001	1110	1111	0110
-2	-	1010	1101	1110	0101
-3	-	1011	1100	1101	0100
-4	-	1100	1011	1100	0011
-5	-	1101	1010	1011	0010
-6	-	1110	1001	1010	0001
-7	-	1111	1000	1001	0000
-8	-	-	-	1000	-