# Rappresentazione di numeri interi

**Argomento**:

- Sistema posizionale e cambio di base
- Rappresentazione di interi

**Materiale di studio:**

- Capitolo 9
- Capitolo 10: Sezioni 10.1-10.3 (esclusa sottosezione "Multiplication")

# Rappresentazione delle informazioni

Stringa di 32 bit nella memoria di un elaboratore:

**11100000100000010010000000000000**

Cosa rappresenta?

# Rappresentazione delle informazioni

**11100000100000010010000000000000**

Un oggetto tra $2^{32}$ =4294967296 possibili oggetti:

- istruzione assembly ARM: **`add r2, r1, r0`**
- numero naturale (in base 2): **$4034994176_{10}$**
- numero intero (complemento a 2): **$-259973120_{10}$**
- numero razionale (floating point):
  **$-744354946411796000000_{10}$**
- stringa di 4 caratteri (ASCII): **`‘<SOH> <NUL>`**
- stringa di 4 caratteri (ASCII estesa): **`αü <NUL>`**
- …

# Cosa vedremo

1. I sistemi di numerazione
   - Decimale, binario, esadecimale…
   - Conversioni di base

2. Le informazioni numeriche
   - Numeri naturali (senza segno o “unsigned”)
   - Numeri interi (con segno o “signed”)
   - Numeri reali (fixed-/floating-point)

3. Le informazioni non numeriche
   - Caratteri e stringhe

# Sistema posizionale

# Sistema posizionale

$$(\ldots a_2 a_1 a_0 . a_{-1} a_{-2} \ldots)_b =$$

$$= \sum_{i=-\infty}^{+\infty} a_i \cdot b^i = \ldots + a_2 \cdot b^2 + a_1 \cdot b^1 + a_0 \cdot b^0 + a_{-1} \cdot b^{-1} + a_{-2} \cdot b^{-2} + \ldots$$

- b è la base del sistema di numerazione
- Gli $a_i$ sono le cifre della rappresentazione, con valore in {0, …, b-1}
- Le cifre con peso maggiore sono quelle *più significative*

Il valore di una cifra dipende dalla sua posizione

# Cambio base da b a 10

Usiamo la definizione di sistema posizionale

$$(\ldots\ a_2 a_1 a_0 . a_{-1} a_{-2} \ldots)_b = \sum_{i=-\infty}^{+\infty} a_i \cdot b^i$$

Il valore in base 10 di $(\ldots\ a_2 a_1 a_0 . a_{-1} a_{-2} \ldots)_b$ si ottiene svolgendo le operazioni a destra in base 10.

# Sistema decimale

- b=10
- Possibili cifre: {0;1;2;3;4;5;6;7;8;9}

$$\mathbf{2095.42}_{10} = \mathbf{2} \cdot 10^3 + \mathbf{0} \cdot 10^2 + \mathbf{9} \cdot 10 + \mathbf{5} + \mathbf{4} / 10 + \mathbf{2} / 10^2$$

# Sistema binario

- b=2
- Possibili cifre: {0;1}

$$\mathbf{11011.101}_2 = \mathbf{1} \cdot 2^4 + \mathbf{1} \cdot 2^3 + \mathbf{0} \cdot 2^2 + \mathbf{1} \cdot 2 + \mathbf{1} + \mathbf{1} / 2 + \mathbf{0} / 2^2 + \mathbf{1} / 2^3 = \mathbf{27.625}_{10}$$

# Sistema ottale

- b=8
- Possibili cifre: {0;1;2;3;4;5;6;7}

$$\mathbf{675.1}_8 = \mathbf{6} \cdot 8^2 + \mathbf{7} \cdot 8 + \mathbf{5} + \mathbf{1} / 8 = \mathbf{445.125}_{10}$$

# Sistema esadecimale

- b=16
- Possibili cifre: {0;1;2;3;4;5;6;7;8;9;A;B;C;D;E;F}

$$\mathbf{8B9.1}_{16} = \mathbf{8} \cdot 16^2 + \mathbf{11} \cdot 16 + \mathbf{9} + \mathbf{1} / 16 = 2233.0625_{10}$$

# Cambio base da 10 a b

Consideriamo la definizione di sistema posizionale

$$(\dots\ a_2 a_1 a_0 . a_{-1} a_{-2} \dots)_{10} = \sum_{i=-\infty}^{+\infty} a_i \cdot 10^i =$$

$$= \sum_{i=-\infty}^{+\infty} a'_i \cdot b_i = (\dots a'_2 a'_1 a'_0 . a'_{-1} a'_{-2} \dots)_b$$

La rappresentazione in base b si ottiene estraendo ogni cifra con operazioni di divisione o moltiplicazione per potenze di b.

Calcoliamo parte intera e parte frazionaria separatamente, e poi concateniamo i risultati

# Da base 10 a base b: parte intera

**Parte intera**: si prendono i resti delle divisioni successive per b

Esempio: $2009_{10} = 7D9_{16}$

2009/16 = 125 resto: 9

125/16 = 7 resto: 13=$D_{16}$

7/16 = 0 resto: 7

L'ultima cifra è la più significativa

Il procedimento si arresta

# Da base 10 a base b: parte frazionaria

**Parte frazionaria**: si prendono le parti intere delle moltiplicazioni successive per b

Esempio: $0.6875_{10} = 0.1011_2$

| | | |
|---|---|---|
| $0.6875 \cdot 2 =$ | $1.375$ | parte intera: 1 |
| $0.375 \cdot 2 =$ | $0.75$ | parte intera: 0 |
| $0.75 \cdot 2 =$ | $1.5$ | parte intera: 1 |
| $0.5 \cdot 2 =$ | $1$ | parte intera: 1 |

La prima cifra è la più significativa

Il procedimento si arresta

# Da base 10 a base b: parte frazionaria (2)

**Parte frazionaria:** il procedimento può anche essere infinito!

Esempio: $0.3_{10} = 0.\overline{01001}_2$

| | | |
|---|---|---|
| 0.3·2 = | 0.6 | parte intera: 0 |
| 0.6·2 = | 1.2 | parte intera: 1 |
| 0.2·2 = | 0.4 | parte intera: 0 |
| 0.4·2 = | 0.8 | parte intera: 0 |
| 0.8·2 = | 1.6 | parte intera: 1 |
| 0.6·2 = | 1.2 | parte intera: 1 |
| … | … | … |

# Cambi di base: bin, oct, hex

- **Da ottale/esadecimale a binario**: espansione di ogni cifra in una terna/quaterna di cifre binarie.

- **Da binario a ottale/esadecimale**: raggruppamento in terne/quaterne di cifre e sostituzione di ciascuna terna/quaterna con l'opportuna cifra ottale/esadecimale.

# Cambi di base: bin, oct, hex

Una cifra ottale è rappresentata da 3 cifre binarie:

100 111 101

6 7 5

$100111101_2 = 675_8 = 445_{10}$

Una cifra esadecimale è rappresentata da 4 cifre binarie:

1000 1011 1001

8 B 9

$100010111001_2 = 8B9_{16} = 2233_{10}$

| Decimal (base 10) | Binary (base 2) | Hexadecimal (base 16) |
|---|---|---|
| 0 | 0000 | 0 |
| 1 | 0001 | 1 |
| 2 | 0010 | 2 |
| 3 | 0011 | 3 |
| 4 | 0100 | 4 |
| 5 | 0101 | 5 |
| 6 | 0110 | 6 |
| 7 | 0111 | 7 |
| 8 | 1000 | 8 |
| 9 | 1001 | 9 |
| 10 | 1010 | A |
| 11 | 1011 | B |
| 12 | 1100 | C |
| 13 | 1101 | D |
| 14 | 1110 | E |
| 15 | 1111 | F |
| 16 | 0001 0000 | 10 |
| 17 | 0001 0001 | 11 |
| 18 | 0001 0010 | 12 |
| 31 | 0001 1111 | 1F |
| 100 | 0110 0100 | 64 |
| 255 | 1111 1111 | FF |
| 256 | 0001 0000 0000 | 100 |

# Rappresentazione finita negli elaboratori

# Rappresentazione negli elaboratori

- L'elemento base per la rappresentazione delle informazioni è il BInary digiT o più semplicemente **BIT** (Tukey, 1947).

- Un bit può assumere uno tra due possibili valori.
  - I valori possono essere rappresentati con 0 o 1, Vero o Falso, …
  - Un bit può essere realizzato in modi diversi: carica elettrica, campo magnetico, …

- Qualsiasi informazione in un elaboratore è rappresentata con un numero finito di bit.

# Rappresentazione finita vs posizionale

- In un sistema posizionale binario, un numero viene rappresentato tramite
  - Cifre {0,1},
  - Segno – per numeri negativi
  - Virgola (o punto) per separare parte intera e parte frazionaria

- In un elaboratore, sono disponibili solo bit e non simboli come segno, virgola, cifre.
- Un bit e una cifra binaria sono simili, ma rappresentano concetti diversi.

# Rappresentazione finita

- Numeri naturali, interi, frazionari si rappresentano solo con sequenze di bit.

- Utilizzando un numero di bit finito si può rappresentare solo una quantità finita di numeri.

| **n. di bit** | **quantità** |
|---|---|
| 4 | $2^4$ |
| 10 | $2^{10}$ |
| 20 | $2^{20}$ |
| 30 | $2^{30}$ |
| M | $2^M$ |

# Rappresentazione di numeri naturali

- Si rappresenta un numero intero con la sua rappresentazione nel sistema posizionale a base 2

- Non si possono rappresentare numeri la cui rappresentazione (esclusi gli 0 più significativi) eccede il numero di bit disponibili.
  - k bits -> 0, 1, …, $2^k$-1

# Rappresentazione di numeri naturali

Esempio di rappresentazione con M=4 bit:

| Rappresentazione | | Numero in base 2 | | Numero in base 10 |
|---|---|---|---|---|
| 0000 | ⇔ | $0000_2$ | ⇔ | $0_{10}$ |
| 0001 | ⇔ | $0001_2$ | ⇔ | $1_{10}$ |
| 0010 | ⇔ | $0010_2$ | ⇔ | $2_{10}$ |
| 0011 | ⇔ | $0011_2$ | ⇔ | $3_{10}$ |
| 0100 | ⇔ | $0100_2$ | ⇔ | $4_{10}$ |
| | … | | … | |
| 1110 | ⇔ | $1110_2$ | ⇔ | $14_{10}$ |
| 1111 | ⇔ | $1111_2$ | ⇔ | $15_{10}$ |

I numeri maggiori di 15 non sono rappresentabili.

# Encoding: da numero naturale a stringa di bit

Un numero naturale $n$ è rappresentato dalla stringa di bit $R_{N,k}(n) = (x_{k-1}, \dots x_0)$ data dalle $k$ cifre binarie meno significative della rappresentazione di $n$ in base 2.

- Potrebbe essere necessario aggiungere bit a 0 nella parte più significative.

- Sono rappresentabili solo i numeri naturali:

$$\mathrm{n} \in \{0, \dots, 2^k - 1\}$$

# Decoding: da stringa di bit a numero naturale

La stringa di bit $x = (x_{k-1}, \dots x_0)$ rappresenta il numero naturale:

$$R_{N,k}^{-1}(x) = \sum_{i=0}^{k-1} x_i \cdot 2^i$$

# Rappresentazione di numeri interi

Ci sono vari modi di rappresentare numeri interi su un numero finito di bit

- Ampiezza e segno
- Eccesso P
- Complemento a 1
- Complemento a 2

# Ampiezza e segno

Il primo bit (quello più significativo) viene utilizzato per indicare il segno

$1111_2 \leftrightarrow -7_{10}$
$1110_2 \leftrightarrow -6_{10}$
$1101_2 \leftrightarrow -5_{10}$
…
$1001_2 \leftrightarrow -1_{10}$
$1000_2 \leftrightarrow -0_{10}$
$0000_2 = +0_{10}$

2 rappresentazioni dello 0

Discontinuità tra -0 e 0

I restanti bit sono usati per la rappresentazione del valore assoluto

$0001_2 = +1_{10}$
…
$0110_2 = +6_{10}$
$0111_2 = +7_{10}$

Si riduce il numero di interi positivi rappresentabili

# Encoding: da numero intero a ampiezza e segno

Un numero intero $q$ è rappresentato dalla stringa di bit $R_{AS,k}(q) = (x_{k-1}, \dots x_0)$ con

$$x_{k-1} \begin{cases} 1 \text{ se } q < 0 \\ 0 \text{ altrimenti} \end{cases}$$

$$x_{k-2} \dots x_0 = R_{N,k}(|q|)$$

- Sono rappresentabili solo i numeri interi:

$$\mathrm{q} \in \{-2^{\mathrm{k}-1} + 1, \dots, 0, \dots, 2^{k-1} - 1\}$$

# Decoding: da ampiezza e segno a numero intero

La stringa di bit $x = (x_{k-1}, \dots x_0)$ rappresenta il numero intero:

$$R_{AS,k}^{-1}(x) = (1 - 2 \cdot x_{k-1}) \cdot \sum_{i=0}^{k-2} x_i \cdot 2^i$$

# Eccesso P

La rappresentazione di z coincide con quella del numero naturale z+P

- $P=2^{k-1}$ o $P=2^{k-1}-1$

Esempio con k=4 e P=8

$0000_2 \leftrightarrow -8_{10} = 0_{10}-8_{10}$

$0001_2 \leftrightarrow -7_{10} = 1_{10}-8_{10}$

$0010_2 \leftrightarrow -6_{10} = 2_{10}-8_{10}$

…

$0111_2 \leftrightarrow -1_{10}$

$1000_2 \leftrightarrow 0_{10}$

$1001_2 \leftrightarrow 1_{10}$

…

$1110_2 \leftrightarrow 6_{10}$

$1111_2 \leftrightarrow 7_{10}$

Unica rappresentazione dello 0

Dissimmetria

# Encoding: da numero intero a eccesso P

- Un numero intero $q$ è rappresentato dalla stringa di bit $R_{E,P,k}(q) = (x_{k-1}, \dots x_0)$ con

$$(x_{k-1} \dots\ x_0) = R_{N,k}(q + P)$$

- Sono rappresentabili solo i numeri:

$$\mathrm{q} \in \{-P, \dots, 0, \dots, P-1\}$$

# Decoding: da eccesso P a numero intero

La stringa di bit $x = (x_{k-1}, \dots x_0)$ rappresenta il numero intero:

$$R^{-1}_{E,P,k}(x) = -P + \sum_{i=0}^{k-1} x_i \cdot 2^i$$

# Complemento a 1

- La rappresentazione di un intero positivo coincide con quella del corrispondente numero naturale.
- La rappresentazione di un intero negativo si ottiene con il **complemento bit a bit** del suo valore assoluto.

$1000_2 \leftrightarrow -7_{10}$
$1001_2 \leftrightarrow -6_{10}$
…
$1110_2 \leftrightarrow -1_{10}$
$1111_2 \leftrightarrow -0_{10}$
$0000_2 = +0_{10}$
$0001_2 = +1_{10}$
…
$0110_2 = +6_{10}$
$0111_2 = +7_{10}$

2 rappresentazioni dello 0

# Encoding: da numero intero a complemento a 1

Un numero intero $q$ è rappresentato dalla stringa di bit $R_{C1,k}(q) = (x_{k-1}, \dots x_0)$ con

- Se $q \geq 0$: $(x_{k-1} \dots x_0) = R_{N,k}(q)$
- Se $q < 0$: $(x_{k-1} \dots x_0) = NOT(R_{N,k}(-q))$

- $NOT(x)$: inversione di tutti i bit della stringa

Sono rappresentabili solo i numeri interi:

$$q \in \{-2^{k-1} + 1, \dots, 0, \dots, 2^{k-1} - 1\}$$

# Decoding: da complemento a 1 a numero intero

La stringa di bit $x = (x_{k-1}, \dots x_0)$ rappresenta il numero intero:

$$R_{C1,k}^{-1}(x) = -x_{k-1}(2^{k-1}-1) + \sum_{i=0}^{k-1} x_i \cdot 2^i$$

# Complemento a 2

- La rappresentazione di un intero positivo coincide con quella del corrispondente numero naturale.
- La rappresentazione di un intero negativo si ottiene **sommando 1** al complemento a 1.

$$
\begin{aligned}
1000_2 &\leftrightarrow -8_{10} \\
1001_2 &\leftrightarrow -7_{10} \\
&\dots \\
1111_2 &\leftrightarrow -1_{10} \\
0000_2 &= 0_{10} \\
0001_2 &= +1_{10} \\
&\dots \\
0110_2 &= +6_{10} \\
0111_2 &= +7_{10}
\end{aligned}
$$

# Encoding: da numero intero a complemento a 2

Un numero intero $q$ è rappresentato dalla stringa di bit $R_{C2,k}(q) = (x_{k-1}, \dots x_0)$ con

Se $q \geq 0$: $(x_{k-1} \dots x_0) = R_{N,k}(q)$

Se $q < 0$: $(x_{k-1} \dots x_0) = NOT(R_{N,k}(-q))+1$

- $NOT(x)$: inversione di tutti i bit della stringa

Sono rappresentabili solo i numeri interi:

$$q \in \{-2^{k-1}, \dots, 0, \dots, 2^{k-1} - 1\}$$

# Decoding: da complemento a 2 a numero intero

La stringa di bit $x = (x_{k-1}, \dots x_0)$ rappresenta il numero intero:

$$R_{C2,k}^{-1}(x) = -x_{k-1}2^{k-1} + \sum_{i=0}^{k-1} x_i \cdot 2^i$$

# Utilizzo

- **Segno & ampiezza:** IBM 704, 709 (anni ’50)
- **Complemento 1:** CDC, PDP (i.e., minicomputer), UNIVAC (anni ‘60)
- **Eccesso P:** usata nelle rappresentazioni floating point e double point (Microsoft Binary format, IEEE 754,…)
- **Complemento 2:** x86, Power Architecture, MIPS, SPARC, ARM, Itanium, …

# Complemento a 2: proprietà

È il metodo di rappresentazione più diffuso, perché è l'unico tra quelli visti con tutte le seguenti proprietà.

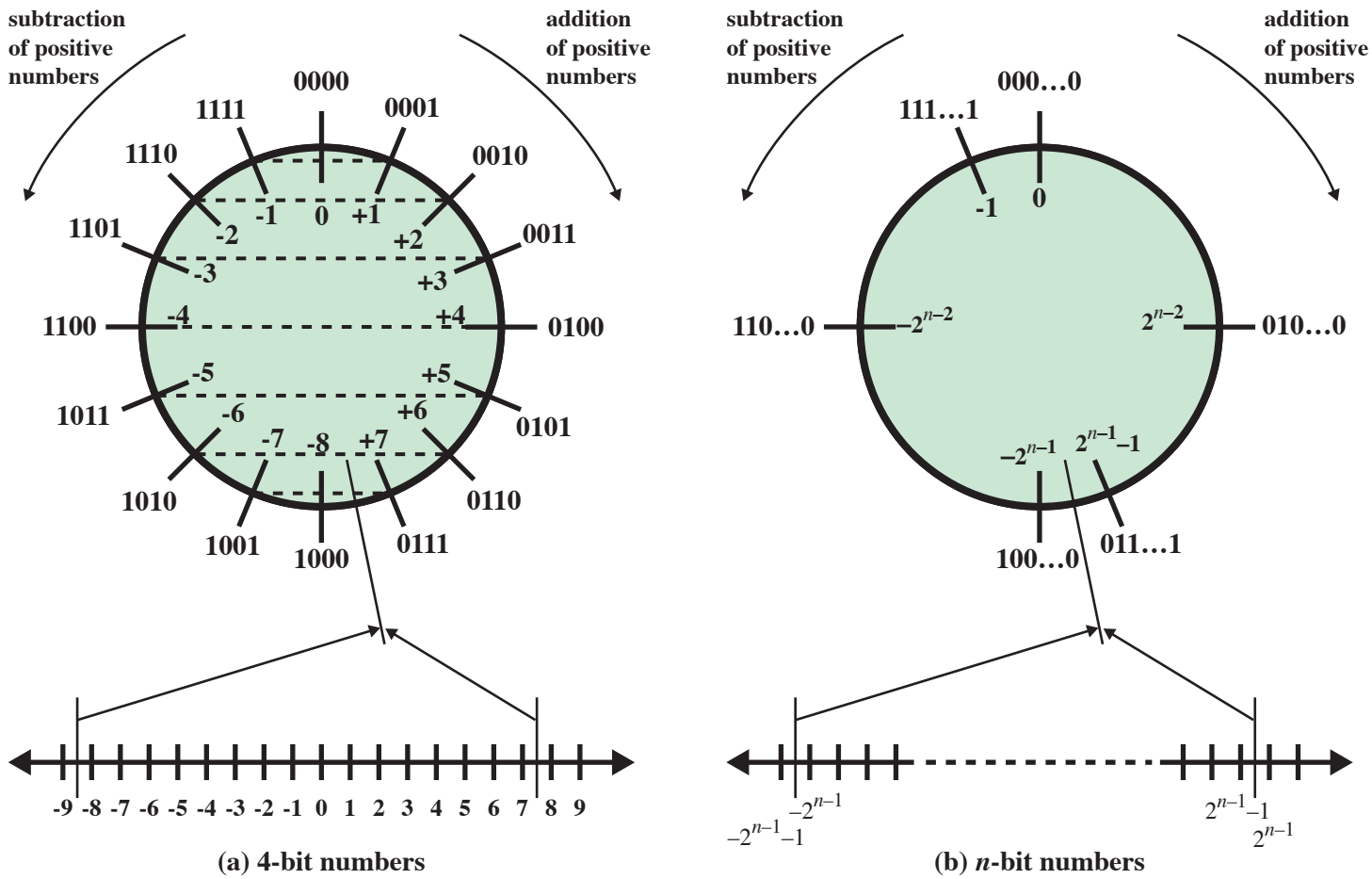
1. Ha una sola rappresentazione dello 0.
2. Il bit più significativo indica il segno
3. Ha una struttura ciclica: aggiungendo 1 al massimo numero rappresentabile si ottiene il minimo numero rappresentabile.

# Complemento a 2: proprietà (2)

4. Consente le operazioni aritmetiche con i numeri negativi usando le **stesse regole** valide per i numeri positivi. Esempi:

| | | | |
|---|---|---|---|
| 0011 | +3 | 0011 | +3 |
| 0100 | +4 | 1100 | -4 |
| 0111 | +7 | 1111 | -1 |

# Complemento a 2: struttura ciclica



# Complemento a 2: overflow

- Nell'eseguire un'operazione aritmetica, bisogna verificare che il risultato sia rappresentabile con il numero di bit a disposizione.
  - Se ciò non è vero (**overflow**), l'esito dell'operazione è privo di significato.
- Come riconoscere l'overflow?
  - Se sommo due numeri positivi e ottengo un numero negativo
  - Se sommo due numeri negativi e ottengo un numero positivo
  - No overflow con un numero negativo e uno positivo

# Overflow: esempi

0101 +5

<u>0010 +2</u>

0111 +7

| Rip. bit segno | Rip. SX bit segno | Esito |
|---|---|---|
| No | No | OK |

1011 -5

<u>1110 -2</u>

1←1001 -7

| Rip. bit segno | Rip. SX bit segno | Esito |
|---|---|---|
| Si | Si | OK |

# Overflow: esempi

| | |
|---|---|
| 0011 | +3 |
| 0110 | +6 |
| 1001 | +9 |

| Rip. bit segno | Rip. SX bit segno | Esito |
|---|---|---|
| Si | No | OVERFLOW |

| | |
|---|---|
| 1011 | -5 |
| 1010 | -6 |
| 1←0101 | -11 |

| Rip. bit segno | Rip. SX bit segno | Esito |
|---|---|---|
| No | Si | OVERFLOW |

L'overflow si verifica quando bit di segno e bit di riporto sono diversi

# Confronto delle rappresentazioni su 4 bit

| Decimal Representation | Sign-Magnitude Representation | Twos Complement Representation | Biased Representation |
|---|---|---|---|
| +8 | — | — | 1111 |
| +7 | 0111 | 0111 | 1110 |
| +6 | 0110 | 0110 | 1101 |
| +5 | 0101 | 0101 | 1100 |
| +4 | 0100 | 0100 | 1011 |
| +3 | 0011 | 0011 | 1010 |
| +2 | 0010 | 0010 | 1001 |
| +1 | 0001 | 0001 | 1000 |
| +0 | 0000 | 0000 | 0111 |
| –0 | 1000 | — | — |
| –1 | 1001 | 1111 | 0110 |
| –2 | 1010 | 1110 | 0101 |
| –3 | 1011 | 1101 | 0100 |
| –4 | 1100 | 1100 | 0011 |
| –5 | 1101 | 1011 | 0010 |
| –6 | 1110 | 1010 | 0001 |
| –7 | 1111 | 1001 | 0000 |
| –8 | — | 1000 | — |

Eccesso $2^3-1 = 7$

# Estensione del numero di bit (ampiezza e segno)

- Per aumentare l'intervallo di numeri è sufficiente aumentare il numero di bit
  - Un bit in più raddoppia l'intervallo

- Con rappresentazione ampiezza e segno:
  - Si sposta il bit di segno nel nuovo bit più significativo
  - Si mettono a 0 i rimanenti bit non impostati

Esempio passando da 4 a 8 bit

1011 → 1000 0011
(valore -3)

0110 → 0000 0110
(valore 6)

# Estensione del numero di bit (complemento a 2)

- La procedure precedente non vale con il complemento a due di numeri negativi

- **Regola**: estensione del segno
  - I nuovi bit vengono impostati come il bit di segno

Esempio passando da 4 a 8 bit

1101 → 11111101
(valore -3)

0110 → 0000 0110
(valore 6)