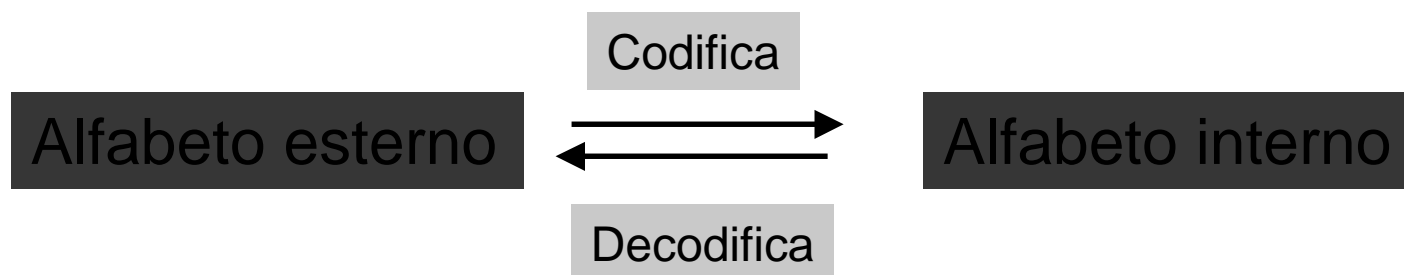


Unità Didattica 2

Rappresentazione dei Dati

Codifica dell'Informazione

- Ad un calcolatore le informazioni sono fornite come sequenze di caratteri alfanumerici (alfabeto esterno [26 lettere maiuscole e minuscole, 10 cifre decimali, segni di interpunzione, matematici e di controllo]);
- Tali caratteri sono rappresentati in memoria come sequenze di un alfabeto interno, secondo una codifica che associa ad ogni carattere dell'alfabeto esterno, una e una sola sequenza di caratteri dell'alfabeto interno:



Alfabeto Interno Binario

- Per rappresentare l'informazione all'interno di un calcolatore (alfabeto interno) si usa l'alfabeto binario poiché le sue componenti possono trovarsi in due soli stati (0/1);
- I motivi per i quali i calcolatori sono stati costruiti con tali componenti sono i seguenti:
 1. Alta tolleranza dei circuiti al rumore: garanzia di un alto grado di tolleranza agli errori;
 2. Circuiti più semplici, tali da poter essere costruiti a basso costo.

Bit, Byte, Word

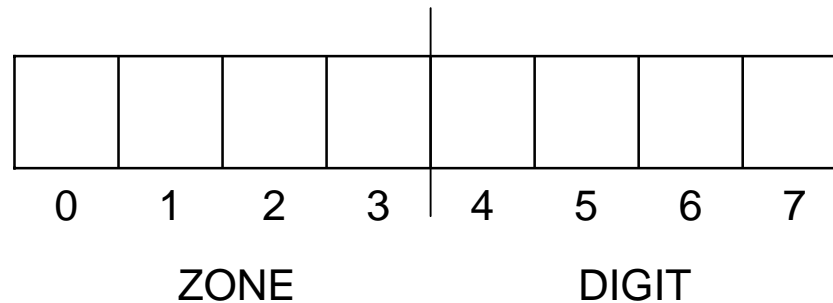
- Bit = Binary digit (cifra binaria (1/0))
- Byte = 8 bit (raggruppamento più comune nelle memorie e nei dispositivi I/O)
- Parola (Word) = raggruppamento più grande del byte si misura in n° di byte

Codifica Dati Alfanumerici

- I codici usati per i dati alfanumerici sono:
 1. EBCDIC (Extended Binary Code Decimal Interchange Code);
 2. ASCII (American Standard Code for Information Interchange).
- In tali codifiche le sequenze numeriche sono interpretate come stringhe (sequenze di caratteri) (es: “31124” non è un numero intero ma, per esempio, un numero di telefono).

Codifica EBCDIC

- Ogni carattere è codificato in un byte:



- Si possono rappresentare $2^8=256$ caratteri > n° caratteri da rappresentare;
- Alcune codifiche non sono utilizzate.

Codifica ASCII

- Ogni carattere è rappresentato su 8 bit;
- L'ASCII originario era di 7 bit, forzato a 8 bit con il bit più significativo = 0;
- Quindi si utilizzano solo metà delle 256 codifiche, ovvero si rappresentano:
 $2^7 = 128$ caratteri:



Codifica Dati Numerici

- Dati numerici in aritmetica decimale:
 - BCD (Binary Coded Decimal);
- Codifica numeri interi (\mathbb{Z}):
 - Modulo e segno;
 - Complemento a 2;
- Codifica numeri reali (\mathbb{R}):
 - Codifica in virgola mobile (floating point).

Codifica BCD

- Codifica per la rappresentazione di quantità numeriche in rappresentazione decimale;
- Ogni cifra decimale (0...9) viene rappresentata mediante la corrispondente codifica binaria;
- Per rappresentare una cifra decimale occorrono 4 bit infatti:

$$2^3 = 8 \text{ cifre} < 10 \text{ cifre} < 2^4 = 16 \text{ cifre}$$

Es: $(82)_{10} \Leftrightarrow \overbrace{1000/0010}^{\text{Codifica BDC}} \Leftrightarrow \overbrace{(1010010)}^{\text{Codifica Binaria}}_2$

8 2

- Le codifiche binaria e BCD sono diverse;
- Es: Codifica BCD Packed (4 bit per ogni cifra + 1 byte di segno):
 - il coprocessore 80387 può rappresentare gli interi con un'aritmetica decimale packed (10 byte = 18 cifre (9 byte) + 1 byte per il segno).

Codifica Interi: Codifica Modulo e Segno

- Si rappresenta un intero mediante la rappresentazione separata del modulo e del segno;
- Qualunque sia il numero dei bit usati, quello più a sinistra rappresenta il *segno* ($0 \Leftrightarrow +$; $1 \Leftrightarrow -$) (tale bit è simbolico, non ha peso);
- I restanti (n-1) bit (che hanno un peso funzione della posizione) rappresentano la codifica binaria del *modulo*;
- Rango di rappresentabilità della codifica modulo e segno:

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

Somma e Sottrazione in codifica Modulo e Segno

- Se la somma di due interi dà un risultato minore/maggiore del rango permesso dai bit di rappresentazione si dice che si è verificato un “underflow”/“overflow” e il risultato non è corretto;
- Esempi di operazioni in codifica modulo e segno:

$$\begin{array}{r} 1 | 10010 \\ 0 | 11110 \\ \hline 0 | 01100 \end{array} + \begin{array}{r} 1 | 010010 \\ 0 | 001110 \\ \hline 1 | 100000 \end{array} =$$

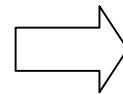
Codifica Interi: Codifica in Complemento a 2

- È la codifica più usata per rappresentare gli interi.
- Si basa sul teorema del complemento:

$$X - Y = X + C_b(Y) \quad \text{su } k \text{ cifre.}$$

- Si distingue il caso di $X \geq 0$ e $X < 0$:

Rappresentazione in
complemento a 2



$$\begin{cases} X \geq 0 & (X)_2 \\ X < 0 & C_2(|X|) \end{cases}$$

- Esempi di rappresentazione in complemento a 2 su 5 bit:

$$X = (12)_{10} \Leftrightarrow (01100)_{c_2} \quad X = (-9)_{10} \Leftrightarrow C_2(|X|) = C_2(01001) = (10111)_{c_2} \quad 12$$

Proprietà della Codifica in Complemento a 2

- Forma polinomiale della codifica in complemento a 2 su n bit:

$$-2^{n-1} \cdot c_{n-1} + 2^{n-2} \cdot c_{n-2} \dots + 2^0 \cdot c_0$$

- Il bit più significativo c_{n-1} ha peso (-2^{n-1}) (c_{n-1} non è un bit simbolico);
- Dalla forma polinomiale in complemento a 2 segue che anche per questa codifica i numeri negativi hanno il bit più significativo $c_{n-1} = 1$;
- Rango di rappresentabilità della codifica in complemento a 2 su n bit:

$$\left[\underbrace{-2^{n-1}}_{c_{n-1}=1, c_{n-2}=0, \dots, c_0=0}, \underbrace{2^{n-1}-1}_{c_{n-1}=0, c_{n-2}=1, \dots, c_0=1} \right]$$

Somma e Sottrazione in codifica Complemento a 2

- Il bit più significativo (che caratterizza il segno) è trattato allo stesso modo degli altri bit;
- Il risultato è espresso nella forma “complemento a 2”;
- Se la somma di due interi dà un risultato minore/maggiore del rango permesso dai bit di rappresentazione, si dice che si è verificato un “underflow”/“overflow” e il risultato non è corretto;
- Regole operative per riconoscere un risultato di underflow/overflow:
 - i) riporto al di fuori del bit di segno e nessun riporto sul bit di segno;
 - ii) nessun riporto al di fuori del bit segno e riporto sul bit di segno.

Esempi di Operazioni in codifica Complemento a 2

- Esempi di somma e sottrazione in complemento a 2:

$$\begin{array}{r}
 110011 + \\
 \underline{011010} = \\
 \begin{array}{r}
 \downarrow \\
 1 \\
 \text{carry} \\
 \text{(si ignora)}
 \end{array}
 \begin{array}{r}
 \underline{001101} \\
 \text{risultato}
 \end{array}
 \end{array}
 \quad \Bigg| \quad
 \begin{array}{r}
 110011 - \\
 \underline{011010} = \\
 \end{array}
 \quad \begin{array}{c}
 \text{Teorema} \\
 \text{del Complemento} \\
 \Rightarrow
 \end{array}
 \quad
 \begin{array}{r}
 110011 + \\
 \underline{100110} = \\
 \begin{array}{r}
 \downarrow \\
 1 \\
 \text{carry} \\
 \text{(si ignora)}
 \end{array}
 \begin{array}{r}
 \underline{011001} \\
 \text{risultato}
 \end{array}
 \end{array}
 \quad \begin{array}{c}
 \text{overflow}
 \end{array}$$

- Il prodotto e la divisione vengono calcolati mediante iterazioni di somme e sottrazioni.

Codifica Floating Point

- Codifica per la rappresentazione dei reali \mathbb{R} ;
- La rappresentazione con posizione fissa della virgola determina un intervallo (rango) di rappresentabilità troppo limitato;
- Per estendere l'intervallo dei valori rappresentabili con uno stesso numero di bit si usa una rappresentazione in virgola mobile (floating point).

Codifica Floating Point

- Corrisponde alla rappresentazione scientifica

$$\pm f \cdot s^{\pm e}$$

f = parte significativa
 s = fattore di scala
 e = esponente

- Rappresentazione scientifica normalizzata in base 10:

$$x = \pm f_0.f_{-1}f_{-2} \dots f_{-p} \cdot 10^{\pm e} \begin{cases} x \neq 0 & f_0 \neq 0 \Rightarrow f_0 = \{1,2,\dots,9\} \\ x = 0 & f_i = 0 \end{cases}$$

- Rappresentazione scientifica normalizzata in base 2:

$$x = \pm f_0.f_{-1}f_{-2} \dots f_{-p} \cdot (10)_2^{\pm e} \begin{cases} x \neq 0 & f_0 \neq 0 \Rightarrow f_0 = 1 \\ x = 0 & f_i = 0 \end{cases}$$

Codifica Floating Point

- La rappresentazione di macchina dei reali utilizza la codifica floating point normalizzata in base 2;
- I bit di rappresentazione sono suddivisi in tre parti:
 - 1 bit di segno (bit simbolico);
 - m bit per la parte frazionaria della parte significativa (mantissa);
 - n bit per l'esponente (si rappresenta la caratteristica).

$$x = \underbrace{\pm}_{\text{segno}} f_0 \cdot \underbrace{f_{-1} f_{-2} \dots f_{-p}}_{\text{mantissa}} \cdot (10)_2^{\pm e}$$

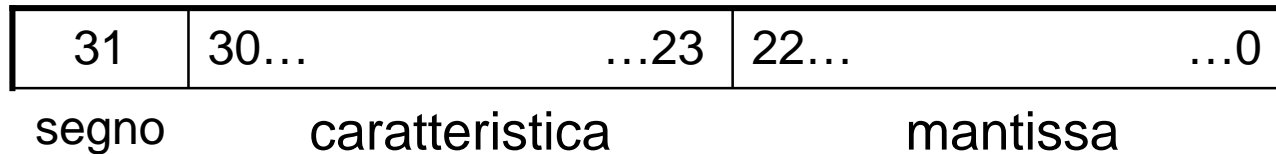
← esponente (si rappr. la "caratteristica")

1 bit	m bit	n bit
segno	mantissa	caratteristica

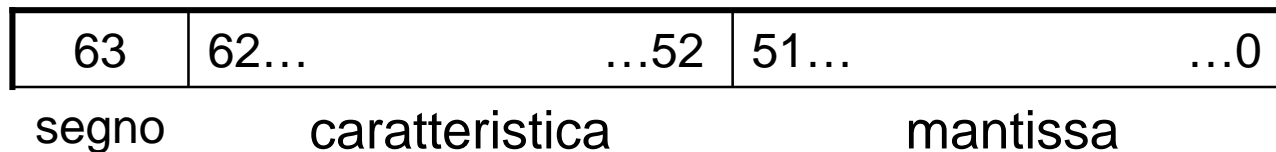
- Il bit f_0 (sempre = 1) e il fattore di scala (sempre = $(10)_2$) non si rappresentano.

Modi di Rappresentazione Floating Point (IEEE)

1. Short Real: 4 byte = 32 bit



2. Long Real: 8 byte = 64 bit



3. Temporary Real: 10 byte = 80 bit



Codifiche di Mantissa e Caratteristica

- Mantissa

- È codificata nella forma modulo e segno;
- Negli short e long real il bit più significativo della mantissa non è rappresentato essendo sempre = 1;
- Non accade nei temporary real perché con essi possono essere rappresentati anche valori non normalizzati;

- Caratteristica

$$c = e + t \quad \text{essendo } t = 2^{k-1} - 1$$

- È codificata nella forma eccesso a t;
- Ovvero su k bit la caratteristica è data dal valore c:

caratteristica	0	1	...	126	127	128	...	255
esponente	*	-126	...	-1	0	+1	...	+128

$$k=8 \text{ bit};$$

$$t=2^7-1=127$$

20

* Valore riservato per la rappresentazione dello 0

Rappresentazione Floating Point del valore 0

- Consideriamo il valore corrispondente alla configurazione:

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
s	Mantissa (8 bit)								Caratteristica (8 bit)							

- Tale valore corrisponde al numero:

$$\underbrace{+1.00000000 \cdot (10)_2^{e=c-t=-1111111}}_{\text{binario}} \Rightarrow \underbrace{1 \cdot 2^{-127}}_{\text{decimale}} = 5.87 \cdot 10^{-39}$$

- Il valore rappresentato è molto piccolo ma non è zero;
- Per convenzione si assume che la configurazione precedente rappresenti lo zero.

Esempi di rappresentazioni Floating Point

$$-(101000.0)_2 \Rightarrow -(1.010000)_2 \cdot (10)_2^{+101}$$

Spostamento a sinistra \Rightarrow esp $>$ 0

$$m = (01000000)_2$$

$$c = e + t = +(101)_2 + (1111111)_2 = (10000100)_2$$

1	01000000	10000100
segno	mantissa (8 bit)	caratteristica (8 bit)

$$+(0.000110)_2 \Rightarrow +(1.100000)_2 \cdot (10)_2^{-100}$$

Spostamento a destra \Rightarrow esp $<$ 0

$$m = (10000000)_2$$

$$c = e + t = -(100)_2 + (1111111)_2 = (1111011)_2$$

0	10000000	01111011
segno	mantissa (8 bit)	caratteristica (8 bit)

Proprietà della codifica Floating Point

- La rappresentazione della caratteristica in eccesso a t evita l'uso del bit di segno per l'esponente;
- Precisione = numero di cifre decimali della parte frazionaria (corrisponde al minor modulo rappresentabile);
- Rango = intervallo di rappresentabilità;

Tipo di rappr. Floating point	<u>Rango</u>	<u>Precisione</u>
Short	$\pm 3.39 \cdot 10^{38}$	$\pm 1.18 \cdot 10^{-38}$
Long	$\pm 1.80 \cdot 10^{308}$	$\pm 2.23 \cdot 10^{-308}$
Temporary	$\pm 1.19 \cdot 10^{4932}$	$\pm 3.36 \cdot 10^{-4932}$

Addizione di numeri floating point

1. Caratteristiche non uguali \Rightarrow per il numero a caratteristica più piccola:
 - a) Spostamento di un posto verso destra della mantissa (equivale a spostare il punto a sinistra e aumentare la caratteristica);
 - b) Incremento della caratteristica di 1 finché le caratteristiche sono uguali;
2. Il risultato ha per mantissa la somma delle mantisse e per caratteristica la caratteristica degli operandi;
3. Se il risultato ha un riporto oltre la cifra più significativa \Rightarrow
 - a) Traslazione di un posto a destra della mantissa;
 - b) Traslazione del riporto nel bit più significativo della mantissa;
 - c) Incremento della caratteristica di 1.

Esempi di addizioni Floating Point

$$\begin{aligned}
 X + Y &= -1.0110 \cdot 10^{10} + 1.1011 \cdot 10^{-1} = \\
 &= -1.0110 \cdot 10^{10} + 0.0011011 \cdot 10^{10} = \\
 &= -(1.0110 \cdot 10^{10} - 0.0011011 \cdot 10^{10}) =
 \end{aligned}$$

$$\begin{array}{r}
 1.0110000 \cdot 10^{10} \quad - \\
 \underline{0.0011011 \cdot 10^{10}} \quad = \\
 1.0010101 \cdot 10^{10}
 \end{array}$$

$$X + Y = -1.0010101 \cdot 10^{10}$$

Il risultato è in forma normalizzata (se non lo fosse si normalizzerebbe mediante lo shift della mantissa e la corrispondente modifica della caratteristica).

Moltiplicazione di numeri Floating Point

1. Moltiplicazione delle due mantisse;
2. Addizione delle due caratteristiche;
3. Traslazione a sinistra dei bit del prodotto delle due mantisse;
4. Troncamento del prodotto delle mantisse al numero di bit usati;
5. Sottrazione dell'eccesso alla somma delle caratteristiche ottenendo la caratteristica del prodotto;

Infatti:

$$c_1 = e_1 + t$$

$$\underline{c_2 = e_2 + t}$$

$$c_1 + c_2 = e_1 + e_2 + 2t$$

- Quindi per ottenere l'eccesso a t occorre sottrarre t dalla somma delle caratteristiche.

Esempi di Moltiplicazione Floating Point

$$X \cdot Y = 1.101 \cdot 10^1 \times 1.01 \cdot 10^{-10} = 10.00001 \cdot 10^{-1}$$

Normalizzando si ha:

$$X \cdot Y = 10.00001 \cdot 10^{-1} = 1.000001 \cdot 10^0$$

Aritmetica degli Elaboratori

- Proprietà:
 1. Rappresentazione binaria dei numeri;
 2. Rango finito;
 3. Alcune operazioni sono espresse in funzione di altre più semplici;
 4. Precisione finita (precisione di macchina):
la rappresentazione in precisione finita determina un errore di troncamento;
 5. Operazioni per mezzo di altre:
 - Sottrazione per mezzo di una complementazione e una addizione
 - Moltiplicazione con successione di somme e shift.

Errore di Troncamento

Errore assoluto = valore vero - valore rappresentato;

Errore relativo = $\frac{\text{valore vero} - \text{valore rappresentato}}{\text{valore vero}}$

- Teorema

- Hp) Rappresentazione binaria di un numero frazionario X alla p -esima cifra dopo il punto;
- Ts) Errore assoluto di troncamento $\leq 2^{-p}$

Dim:

$$\text{Err. assoluto} = \underbrace{\sum_{i=0}^m c_i 2^i + \sum_{i=1}^{+\infty} c_{-i} 2^{-i}}_{\text{valore vero}} - \underbrace{\left(\sum_{i=0}^m c_i 2^i + \sum_{i=1}^{+p} c_{-i} 2^{-i} \right)}_{\text{valore rappresentato}} =$$
$$= \sum_{i=p+1}^{+\infty} c_{-i} 2^{-i} \stackrel{\text{se } c_i=1}{\leq} \sum_{i=p+1}^{+\infty} 2^{-i} = \sum_{i=0}^{+\infty} 2^{-i} - \sum_{i=0}^{+p} 2^{-i} = \frac{1}{1-2^{-1}} - \frac{1-2^{-(p+1)}}{1-2^{-1}} = 2^{-p}$$