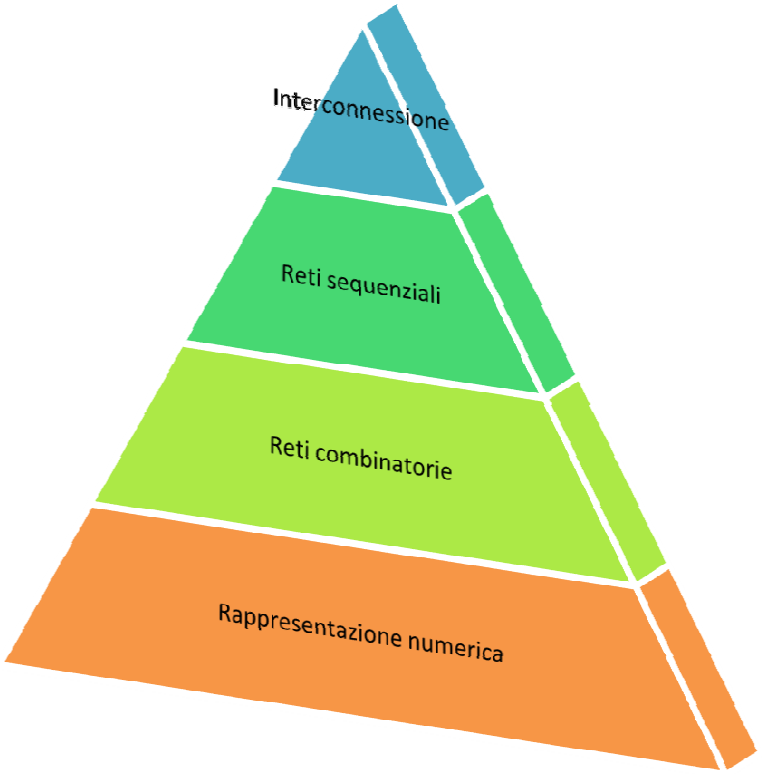
# Architettura degli elaboratori

## Principi

# ARGOMENTI DELLA LEZIONE

- Richiamo della rappresentazione numerica nei calcolatori: numeri interi e numeri reali
- Reti combinatorie
  - Codificatore e Decodificatore
  - Addizionatore e Sottrattore
  - Comparatore aritmetico e logico
- Reti sequenziali
  - Registro di memorizzazione
  - Registro Contatore
  - Registro a scorrimento
- Interconnessione
  - Multiplexer e demultiplexer, mesh, bus



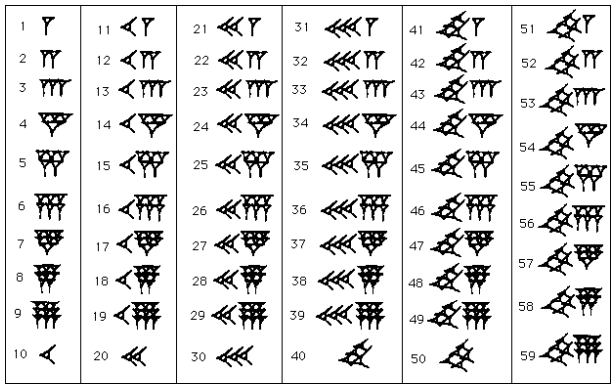
# Architettura degli elaboratori

## *Rappresentazione numerica*

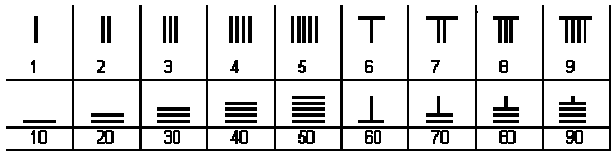
# RAPPRESENTAZIONE NUMERICA

## Numeri e loro rappresentazione

- Un **numero** può essere rappresentato in vari modi con simboli diversi



| | | | |
|---|---|---|---|
| 1 | I | 8 | VIII |
| 2 | II | 9 | IX |
| 3 | III | 10 | X |
| 4 | IV | 50 | L |
| 5 | V | 100 | C |
| 6 | VI | 500 | D |
| 7 | VII | 1000 | M |



# RAPPRESENTAZIONE NUMERICA

## Sistema posizionale

- Introdotto da **Leonardo Fibonacci** dopo essere entrato in contatto con alcuni commercianti arabi (che avevano acquisito le nozioni da matematici indiani)
- Fibonacci, nel suo saggio *Liber abaci* del 1202, spiega il nuovo metodo di rappresentazione che assegnava un valore diverso a una stessa cifra in relazione alla sua posizione. Ne spiega anche la modalità per svolgere calcoli
- In più introdusse, per la prima volta, lo zero (*zephirum*); un nuovo segno grafico che simboleggiava il concetto di vuoto (o di assenza di valore), ovvero l'unico numero a non essere positivo né negativo

$$N = \sum_{i=0}^{m} a_i b^i \quad \text{con} \quad 0 \le a_i \le b-1 \quad \text{e scelta } b > 1$$

(2312) $=2\cdot10^3+ 3\cdot10^2+ 1\cdot10^1+ 2\cdot10^0$

$=2\cdot1000+3\cdot100+1\cdot10+2=2312$

# RAPPRESENTAZIONE NUMERICA

## Sistema posizionale in base due

- Presentato da **Gottfried Wilhelm von Leibniz** nel 1666 nel suo trattato l'*Arte combinatoria* con il quale gettò le basi della logica simbolica, su cui si regge il funzionamento dei moderni calcolatori, e formulò per l'appunto l'idea del calcolo binario, *che riduca in forma più semplice le 'leggi del pensiero'*

$$N = \sum_{i=0}^{m} a_i 2^i \quad \text{con} \quad 0 \le a_i \le 1$$

(1011) $=1\cdot2^3+0\cdot2^2+1\cdot2^1+2\cdot2^0=$

$=8 + 0 + 2 + 1 = 13$

# RAPPRESENTAZIONE NUMERICA

## Il numero e la sua rappresentazione

- In campo matematico si usa la base 10
- In campo informatico si fa riferimento al sistema binario, ottale e esadecimale

| Valore | Base |
|---|---|
| **110582** | **10** |
| 11010111111110110 | 2 |
| 327766 | 8 |
| 1AFF6 | 16 |

# RAPPRESENTAZIONE NUMERICA

## Numero e dimensione della parola

- Lo spazio a disposizione per immagazzinare le informazioni relative a un numero binario è limitato dalla **dimensione della parola** di rappresentazione
- In una parola di dimensione *n* bit, tutte le informazioni relative al numero, segno compreso, devono occupare non più di *n* bit.
- Dopo una operazione aritmetica il numero da rappresentare può eccedere la dimensione della parola provocando un **overflow**

PAROLA DI 8 BIT

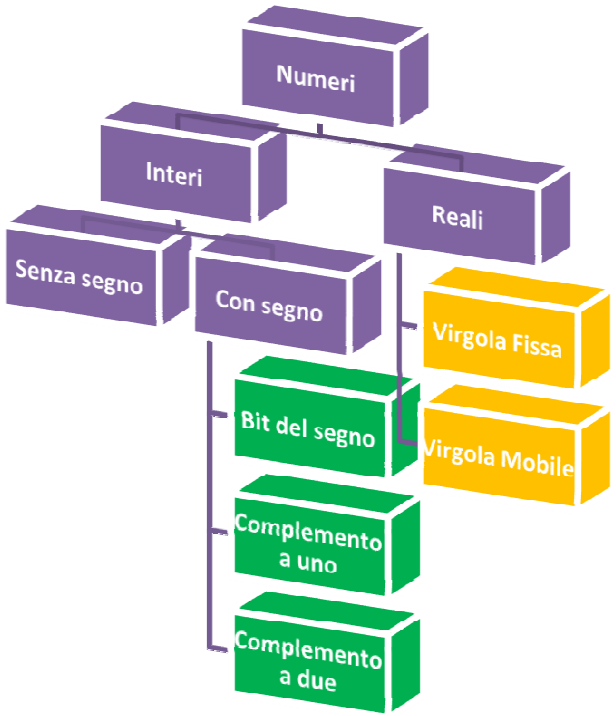
PAROLA DI 16 BIT

PAROLA DI 16 BIT CON OVERFLOW

# RAPPRESENTAZIONE NUMERICA

## Insieme numerici

- Gli elaboratori elettronici supportano delle funzioni logiche (e logico-aritmetiche) operanti sulle stringhe e delle funzioni aritmetiche che gestiscono:
  - **Numeri interi**, per cui si ha una rappresentazione:
    - Senza segno
    - Con segno
      - Bit del segno
      - Complemento a uno
      - Complemento a due
  - **Numeri reali**, per cui si ha una rappresentazione:
    - virgola fissa
    - virgola mobile



# INTERO SENZA SEGNO

## Sistema binario

- Un **intero senza segno** è un valore assunto positivo
- Gli *n* bit della parola sono usati tutti per rappresentare le cifre del numero
- Una parola a *n* bit permette di trattare un qualsiasi numero il cui **intervallo di rappresentazione** (o *range*) è $[0;2^n-1]$

$$N = \sum_{i=0}^{\lceil \log_2 N \rceil} a_i 2^i \qquad 0 \le a_i \le 1$$

| DECIMALE | BINARIO |
|---|---|
| 0 | 000 |
| 1 | 001 |
| 2 | 010 |
| 3 | 011 |
| 4 | 100 |
| 5 | 101 |
| 6 | 110 |
| 7 | 111 |

# INTERI SENZA SEGNO

## Rappresentazioni con base non binaria

- Per rappresentazioni generiche con base maggiore a 10 si introducono, per convenzione, caratteri alfabetici in maiuscolo
  - Sistema **esadecimale**
    0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F
  - Sistema in **base venti**
    0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F,G,H,I,L
- È importante usare simboli per evitare ambiguità

  ES: Nel sistema esadecimale 10 indica il valore 16 (cioè $1 \cdot 16^1 + 0 \cdot 16^0$ ); mentre il valore decimale 10 è espresso con la lettera A

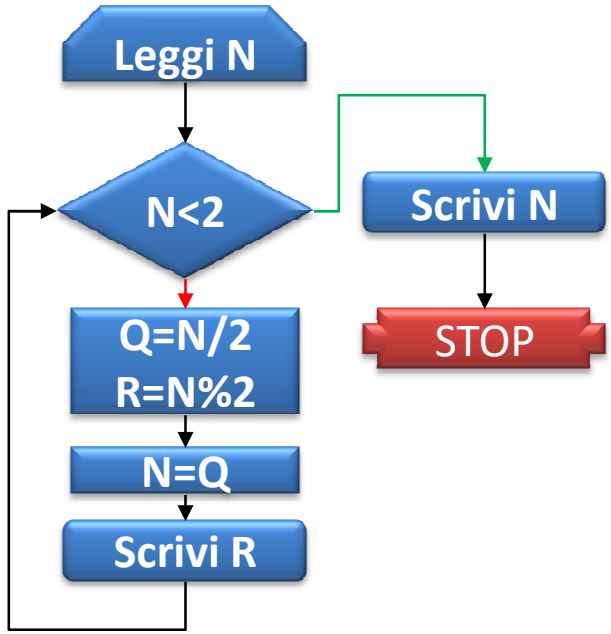
  $$(10)_{16}=16 \text{ e } (A)_{16}=10$$
- Superati i simboli alfabetici si ricorre a simboli speciali (es.: ♣ ♦ ♥ ♠) definibili dall'utente (le associazioni tra simboli e numero devono essere noti)

| VALORE | BASE | RAPPRESENTAZIONE |
|---|---|---|
| **2502** | | |
| | **10** | $(2502)_{10}$ |
| | **16** | $(9C6)_{16}$ |
| | **8** | $(4706)_{8}$ |
| | **2** | $(0000100111000110)_{2}$ |

# INTERI SENZA SEGNO

## Rappresentazione in base due

- Un metodo per convertire un valore da una base decimale ad una binaria è quello delle **divisione successive**




| 82 | Quoziente | Resto |
|---|---|---|
| 82:2 | 41 | 0 LSB |
| 41:2 | 20 | 1 |
| 20:2 | 10 | 0 |
| 10:2 | 5 | 0 |
| 5:2 | 2 | 1 |
| 2:2 | 1 | 0 |
| 1MSB | | |
| | | 1010010 |

$(82)_{10}=(01010010)_2$

# INTERI SENZA SEGNO

## Rappresentazione da base binaria a decimale

$(82)_{10}=(01010010)_2$

$N=0\cdot 2^7+1\cdot 2^6+0\cdot 2^5+1\cdot 2^4+0\cdot 2^3+0\cdot 2^2+1\cdot 2^1+0\cdot 2^0$

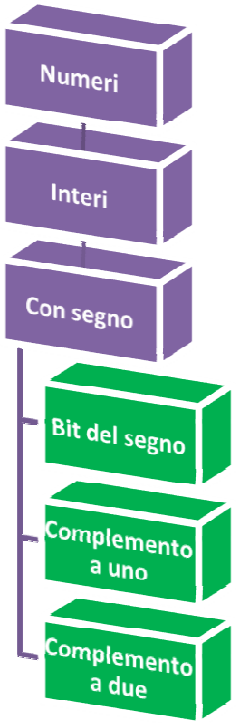
$N=0+64+0+16+0+0+2+0$

$N=82$

# INTERI CON SEGNO

## Modalità di rappresentazione

- Nella rappresentazione dei **numeri interi** c'è la necessità di riservare un bit relativo al segno (**+** oppure **-**) o di scegliere una codifica che consenta di discriminare in maniera chiara e precisa i valori negativi da quelli positivi
- I tre metodi usati sono:
  - bit del segno
  - complemento a uno
  - complemento a due (affermato in quasi tutti i calcolatori elettronici)



# INTERI CON SEGNO

## Bit del segno

- Nel formato **bit del segno**, il bit più significativo dell'intera parola è riservato al segno del numero:
  - **1** per identificare il segno **-**
  - **0** per identificare il segno **+**
- In rappresentazione:
  - si codifica il numero in binario
  - si antepone il bit del segno
- Per una parola di lunghezza *n* l'intervallo di valori rappresentabili va da $[-2^{n-1}+1; 2^{n-1}-1]$

$(+15)_{10} = (\mathbf{0}0001111)_2$

$(-12)_{10} = (\mathbf{1}0001100)_2$

$(+0)_{10} = (\mathbf{0}0000000)_2$

$(-0)_{10} = (\mathbf{1}0000000)_2$

# INTERI CON SEGNO

## Complemento a uno

- Il **complemento a uno** è realizzato complementando (ovvero ¬0=1 e ¬1=0) tutte le cifre che costituiscono la parola
- Il segno di un numero è individuato dal valore del primo bit, infatti tutti i numeri positivi iniziano con il bit 0, mentre quelli negativi hanno il bit più significativo impostato a 1

$(+15)_{10}$ $=(00001111)_2$

$(-15)_{10}$ $=(11110000)_2$

$(+0)_{10}$ $=(00000000)_2$

$(-0)_{10}$ $=(11111111)_2$

# INTERI CON SEGNO

## Complemento a due

- Nel **complemento a due** il bit più significativo ha peso negativo
- I numeri negativi in complemento a due si ottengono complementando l'intero positivo e sommando a questo ultimo il valore 1
- L'intervallo di valori rappresentabile è

  $[-2^{n-1}; 2^{n-1}-1]$

$$N = -a_{n-1}2^{n-1} + \sum_{i=0}^{n-2} a_i 2^i$$

$(+15)_{10}$ $=(00001111)_2$

$(-12)_{10}$ $=(11110100)_2$

$(\mathbf{0})_{10}$ $=(00000000)_2$

$(-12)_{10}$ Valore assoluto: $(00001100)_2$

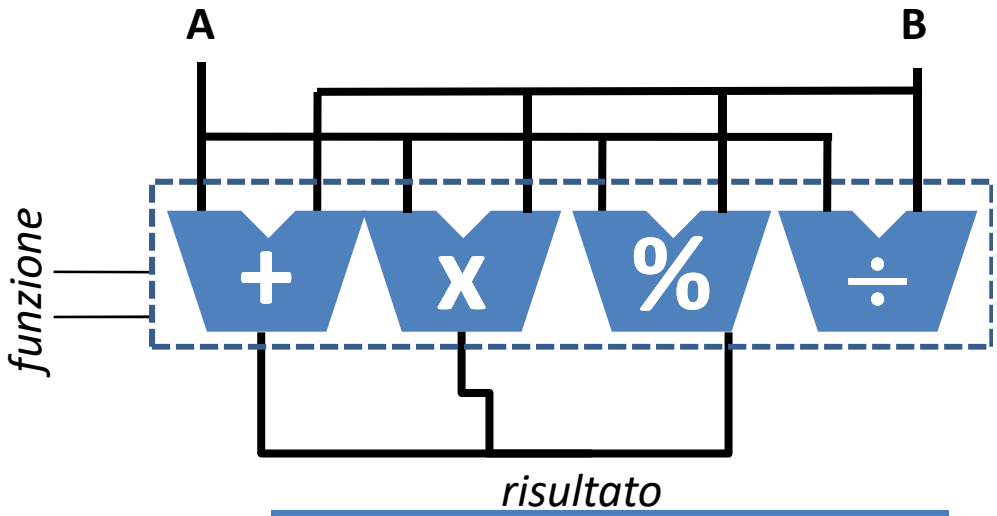
Complementare: $(11110011)_2$

Somma di 1: $(11110100)_2$

# INTERI CON SEGNO

## Complemento a due: operazioni

- Sui numeri si interviene mediante **operazioni elementari** (somma, sottrazione, moltiplicazione, divisione, modulo) o **complesse** (integrazione, derivazione, calcolo di convoluzione,…)




| Comando | Funzione |
|---|---|
| 00 | Somma (A+B) |
| 01 | Moltiplicazione (A*B) |
| 10 | Resto della divisione (A%B) |
| 11 | Quoziente della divisione (A/B) |

# INTERI CON SEGNO

## Complemento a due: addizione

- Nel caso della **addizione tra numeri interi binari** si ha (considerando la somma di cifre ad uguale posizione) il risultato
  - **1** se una delle due cifre è 1 e l'altra è zero e non c'è riporto
  - **0** altrimenti (con un riporto nel caso in cui entrambe le cifre sono 1)
  - Il riporto all'ultima cifra può essere un **trabocco** o **overflow**

| Cifra addendo1 | Cifra addendo2 | Risultato | Riporto |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |

# INTERI CON SEGNO

## Complemento a due: addizione

**Esempio**

| | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | | | Riporto |
| 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 6444 | + Operando 1 |
| 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 7018 | = Operando 2 |
| 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 13462 | Risultato |
| | | | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | | *posizione* |

# INTERI CON SEGNO

## Complemento a due: addizione (trabocco)

- La somma di due numeri binari in complemento a due può comportare un **trabocco** (*carry*) non influente sul risultato finale

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | | | **Riporto** |
| | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 15 | + **Sottraendo** |
| | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | -5 | **Minuendo** |
| 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | =10 | 10 **Risultato** |
| | | | | | | | | | | |
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | | ***Posizione*** |

# INTERI CON SEGNO

## Complemento a due: addizione (overflow)

- In caso di due numeri in complemento a due con segno uguale può presentarsi una **condizione di non rappresentabilità del risultato** (*overflow*) a causa della dimensione prefissata della parola

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | | | Riporto |
| | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | -137 | + Sottraendo |
| | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | -95 | = Minuendo |
| 1 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 42 | (-232) Risultato |
| | | | | | | | | | | |
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | | *posizione* |

# INTERI CON SEGNO

## Osservazione

- Il trabocco (C), la negatività del numero (N), la mancanza di presenza di Overflow (W) contribuiscono a determinare se due operandi sono uguali o maggiori
- La presenza di un Overflow, tipicamente, individua una condizione di errore dopo un calcolo aritmetico

# INTERI CON SEGNO

## Complemento a due: sottrazione

- Nel caso della **sottrazione tra numeri in base binaria** si ha (considerando le cifre ad uguale posizione) il risultato
  - **1** se una delle due cifre è 1 e l'altra è zero; ma se la prima cifra è minore della seconda si deve richiedere un prestito alle cifre successive
  - **0** altrimenti

| Cifra1 | Cifra2 | Risultato | Prestito |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 |

# INTERI CON SEGNO

## Complemento a due: sottrazione

**Esempio 58-44**

| | | | | | |
|---|---|---|---|---|---|
| 111010 - | 111010 - | 111010 - | 110110 - | 110110 - | 101110 - |
| 101100= | 101100= | 101100= | 101100= | 101100= | 101100= |
| ------ | ------ | ------ | ------ | ------ | ------ |
| 0 | 10 | 10 | 110 | 110 | 1110 |

| | |
|---|---|
| 101110 - | 101110 - |
| 101100= | 101100= |
| ------ | ------ |
| 01110 | 001110 |

**14**

| Cifra1 | Cifra2 | Risultato | Prestito |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 |

# INTERI CON SEGNO

## Complemento a due: sottrazione (caso pratico)

**Esempio 58 + (-44)**

| | | |
|---|---|---|
| 111010 - | 0111010 + | 0111010+ |
| 101100= | **1010100**= | 1010100= |
| _____ | _____ | _____ |
| | | ***1***0001110 |

# INTERI CON SEGNO

## Complemento a due: moltiplicazione

- Nel caso della **moltiplicazione tra numeri in base binaria** si procede in maniera analoga a quella per il calcolo decimale
- Si esegue la somma dei prodotti parziali opportunamente slittati

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | | | 1 | 0 | 0 | 0 | · |
| | | | 1 | 0 | 0 | 1 | = |
| | | | 1 | 0 | 0 | 0 | + |
| | | 0 | 0 | 0 | 0 | | + |
| | 0 | 0 | 0 | 0 | | | + |
| 1 | 0 | 0 | 0 | | | | = |
| 1 | 0 | 0 | 1 | 0 | 0 | 0 | |

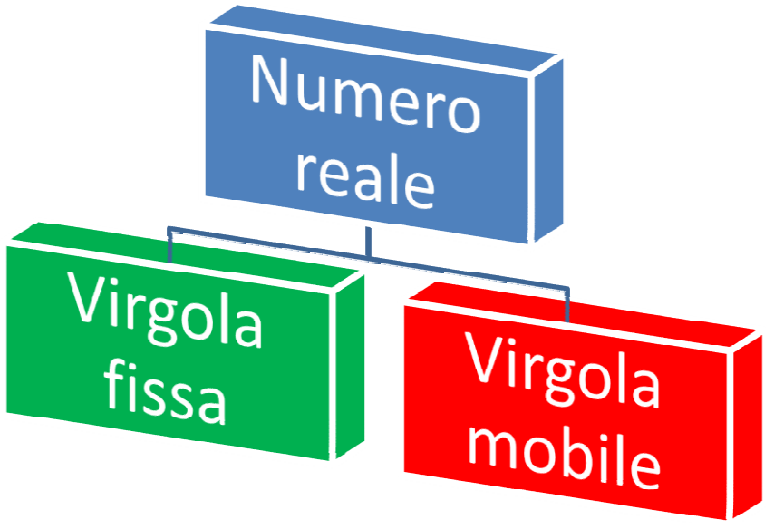
# NUMERI REALI

- Un **numero reale**, in generale, è definibile da una tripla $\Re$=<S;I;F> dove:
  - S è il segno
  - I è la parte intera
  - F la parte frazionaria
- La parte frazionaria è distinta da quella intera con una virgola o (nella notazione scientifica internazionale) con un punto

| | |
|---|---|
| **+3.75** | **<+;3;75>** |
| **-1.75** | **<-;1;75>** |
| **+11.504** | **<+;11;504>** |
| **-146.9** | **<-;146;9>** |

# NUMERI REALI

- Nei calcolatori elettronici i numeri reali possono essere rappresentati in due modi: **virgola fissa** (*fixed point*) o **virgola mobile** (*floating point*)
- Attualmente i calcolatori elettronici utilizzano, per lo più, il sistema in virgola mobile



# NUMERI REALI

## Virgola fissa

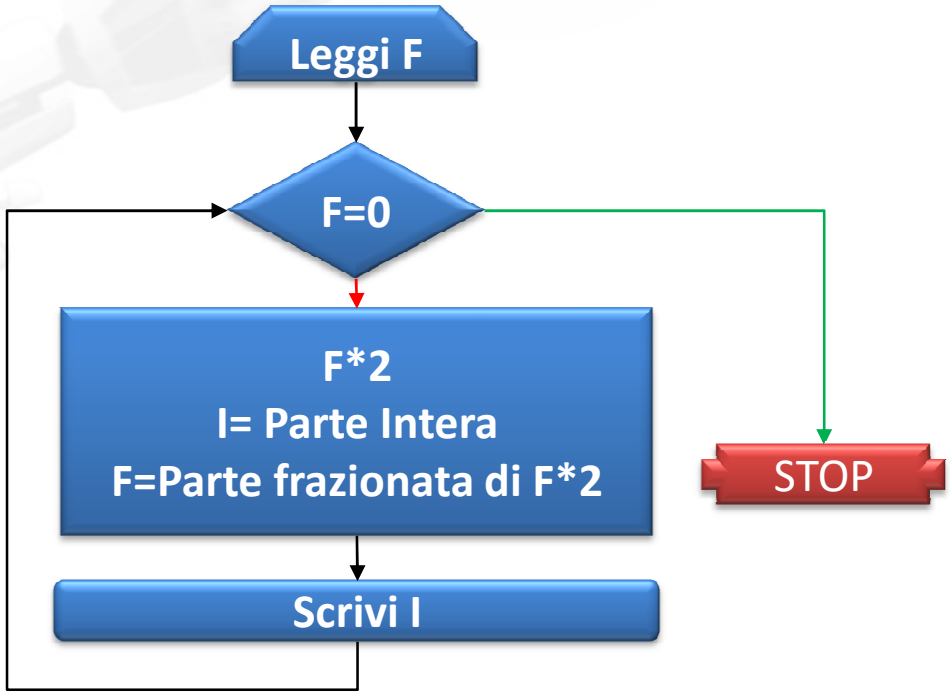
- I numeri in **virgola fissa** offrono una possibile rappresentazione dei valori reali
- Per operare con questo formato è necessario dapprima stabilire la lunghezza della parola e poi determinare la suddivisone in due campi: uno per la **parte intera** (rappresentabile ad esempio con il complemento a due) e un'altra per la **parte frazionaria**

| Parte Intera | Parte frazionaria |
|---|---|
| *k* | *n-k* |

| | |
|---|---|
| **+3,75** | **0011 \| 1100** |
| **-1,75** | **1111 \| 1100** |

# NUMERI REALI

## Virgola fissa: moltiplicazioni successive




0,8515625=
0,8515625·2= 1 + 0,703125
0,703125·2= 1 + 0,40625
0,40625·2= 0 + 0.8125
0.8125·2= 1 + 0.625
0.625·2= 1 + 0.25
0.25·2= 0 + 0.5
0.5·2= 1 + 0
0

**1101101**

# NUMERI REALI

## Virgola fissa

| Parte Intera | | | | | | | | Parte frazionaria | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| *S* | $2^{k-1}$ | | | | $2^{2}$ | $2^{1}$ | $2^{0}$ | $2^{-1}$ | $2^{-2}$ | $2^{-3}$ | $2^{-4}$ | | | | $2^{-n+k}$ |

| Parte Intera | | | | Parte frazionaria | | | |
|---|---|---|---|---|---|---|---|
| *0* | *1* | *0* | *1* | *1* | *1* | *0* | *0* |
| | | | | | | | |

+5,75

| Parte Intera | | | | Parte frazionaria | | | |
|---|---|---|---|---|---|---|---|
| *1* | *1* | *1* | *0* | *1* | *0* | *1* | *0* |

-2,625

# NUMERI REALI

## Virgola mobile

- La rappresentazione in **virgola mobile** è definita dalla tripla:

  < *S; M*; *E* > dove:
  - *S* è il *segno* del numero
  - *M* la sua *mantissa*
  - *E* il suo *esponente*
- Il motivo principale per l'introduzione dei numeri in virgola mobile (o *floating point*) sta nella possibilità di utilizzare un **intervallo di numeri più ampio** per effettuare i calcoli riducendo però il numero di valori disponibili a parità di bit utilizzati per la codifica
- Non tutti i numeri possono essere rappresentati, ma solo un sottoinsieme limitato (aumenta il range, ma diminuisce la precisione)

$$\Re = (-1)^S \cdot M \cdot 10^E$$

| | |
|---|---|
| **+3,75** | **S=0 M=375 E=-2** |
| **-1,75** | **S=1 M=175 E=-2** |
| **+11,504** | **S=0 M=11504 E=-3** |
| **-146,9** | **S=1 M=1469 E=-1** |

# NUMERI REALI

## Virgola mobile: standard IEEE 754

- Nella **aritmetica in virgola mobile in base 2** la prima decisione da prendere è quanti bit devono essere assegnati a esponente e mantissa
- La notazione scientifica, stabilita dall'IEEE754, definisce quattro formati:
  - Mezza precisione (16bit)
  - **Singola precisione (32bit)**
  - **Doppia precisione (64bit)**
  - Quadrupla precisione (128bit)
  - Ottupla precisione (256bit)

$$\Re = (-1)^S \cdot M \cdot 2^E$$

Formati standard **IEEE-754**

Singola precisione **(FLOAT)**

| Segno | Esponente | Mantissa |
|---|---|---|
| 1 bit | 8 bit | 23 bit |

Doppia precisione **(DOUBLE)**

| Segno | Esponente | Mantissa |
|---|---|---|
| 1 bit | 11 bit | 52 bit |

# NUMERI REALI

## Virgola mobile: standard IEEE 754

- La notazione scientifica adotta una **forma standard**
- La forma standard elimina gli zeri iniziali per avere esattamente una cifra diversa da zero a sinistra della virgola decimale (Es: il numero 34.56 ha forma standard: 3.456)
- Pertanto, un numero **in virgola mobile in base due normalizzato** ha una mantissa la cui cifra più a sinistra è diversa da zero
  - Per i numeri floating point normalizzati in base due, si ha una cifra uguale ad 1

| **+00101010,11001000100000000000000** | | | |
|---|---|---|---|
| | **S** | **M** | **E** |
| **NON normalizzata** $+0,01010101100100010 \ldots 0 \cdot 2^{7}$ | $(0)_2$ | $(10101011001000 1)_2$ | 00000111 (+7) |
| **Normalizzata** $+1,0101011001000 10 \ldots 0 \cdot 2^{5}$ | $(0)_2$ | $(10101011001000 1)_2$ | 00000101 (+5) |
| | | 23bit | 8bit |

# NUMERI REALI

## Virgola mobile: standard IEEE 754

***Osservazione.*** Poiché il bit più a sinistra di una mantissa in base due normalizzata è sempre 1, non è necessario codificarlo (così si risparmia un bit); perciò il bit iniziale, in alcune architetture, diventa un **bit nascosto** (*hidden bit*)

| **+0101010,110010001 (42.783203)** | | | |
|---|---|---|---|
| | **S** | **M** | **E** |
| **Normalizzato con bit nascosto**<br>**+1,010101100100010…0·2$^5$** | $(0)_2$ | $(1010101100100010)_2$<br>23bit | 00000101 (5)<br>8bit |

# NUMERI REALI

## Virgola mobile: standard IEEE 754

Lo standard usa anche la **polarizzazione dell'esponente** (o **notazione in eccesso**) cioè all'esponente si somma 127, singola precisione, o 1023, doppia precisione, e si sfruttano delle combinazioni speciali (es.:00000000 e 11111111) per rappresentare determinati valori

L'uso della polarizzazione dell'esponente semplifica l'ordinamento dei valori in virgola mobile attraverso il confronto tra interi

| **+0101010,110010001 (42.783203)** | | | |
|---|---|---|---|
| | **S** | **$E^{POLARIZZATO}$** | **M** |
| **Normalizzato con bit nascosto e polarizzato**<br>**$+1,010101100100010\ldots0 \cdot 2^{5+127}$** | 0<br>1bit | $(10000100)_2$<br>8 bit | $(10101011001000 10\ldots0)_2$<br>23 bit |

# NUMERI REALI

## Virgola mobile: standard IEEE 754

- La notazione scientifica, infatti, sfruttando l'esponente polarizzato definisce dei **casi particolari** utili nel campo matematico

| Singola Precisione | | Doppia precisione | | Significato |
|---|---|---|---|---|
| Esponente | Mantissa | Esponente | Mantissa | |
| 0 | 0 | 0 | 0 | Zero |
| 0 | ≠0 | 0 | ≠0 | ± valore denormalizzato |
| 1-254 | Qualsiasi | 1-2046 | qualsiasi | ± valore floating point IEEE |
| 255 | 0 | 2047 | 0 | ±∞ |
| 255 | ≠0 | 2047 | ≠0 | NaN |

# NUMERI REALI

## Virgola mobile: esempio singola precisione

- Il numero **-5,125** è rappresentabile in singola precisione come segue:
  Trasformazione delle parte intera e frazionata in binario:
  $-101{,}001 \cdot 2^0$
  Normalizzazione: $-01{,}01001 \cdot 2^2$
- Rappresentazione:
  Ë=2+127=129=$(10000001)_2$.
  M=$(10100100000000000000000)_2$

| Esponente polarizzato | Mantissa con bit nascosto |
|---|---|
| **1**10000001 | 01001000000000000000 |
| 9bit | 23bit |

# NUMERI REALI

## Virgola mobile: esempio singola precisione

**Verifica**

- Il segno è 1 quindi il numero è negativo
- La mantissa è 1*(bit nascosto)*+$2^{-2}$+$2^{-5}$ =1.28125.
- Esponente è 10000001=129 quindi 129-127=2
- In sintesi: (-1)· 1.28125·$2^{2}$=- 1.28125·4=-5.125

| Esponente polarizzato | Mantissa con bit nascosto |
|---|---|
| **1**10000001 | 01001000000000000000000 |
| 9bit | 23bit |

# NUMERI REALI

## Virgola mobile: esempio doppia precisione

- Il numero **+10,125** è rappresentabile in doppia precisione come segue:
  Trasformazione delle parte intera e frazionata in binario: $+1010{,}001 \cdot 2^0$
  Normalizzazione: $+01{,}010001 \cdot 2^3$
- Rappresentazione:
  S=segno positivo=$(0)_2$
  Ë=3+1023=1026=$(10000000010)_2$.
  M=$(1010001000000000000000000000000000000000000000000000)_2$

| Esponente polarizzato | Mantissa con bit nascosto |
|---|---|
| **0**10000000010 | 0100010000000000000000000000000000000000000000000000 |
| 12bit | 52bit |

# NUMERI REALI

## Virgola mobile: esercizio proposto

**Esercizio proposto**

Dato il numero reale -**110,90625**

Rappresentarlo secondo lo standard IEEE754 singola precisione normalizzato e polarizzato e effettuare la verifica

# NUMERI REALI

## Virgola mobile: operazioni somma e sottrazione

- L'algoritmo di **addizione** (e sottrazione) in virgola mobile si divide in quattro fasi fondamentali:
  1. Allineamento degli esponenti
  2. addizione (o sottrazione) delle mantisse
  3. normalizzazione
  4. arrotondamento delle mantisse (ed eventuale normalizzazione)

**Esempio.**

Dati i valori 17,12 e 423,50:

Rappresentazione: $1{,}712 \cdot 10^1$ e $4{,}235 \cdot 10^2$

Allineamento esponenti: $1{,}712 \cdot 10^1$ e $42{,}35 \cdot 10^1$

Somma: $44{,}062 \cdot 10^1$

Normalizzazione: $4{,}4062 \cdot 10^2$

# NUMERI REALI

## Virgola mobile: Overflow - Underflow

- Un elaboratore presenta un **overflow/underflow aritmetico** nel caso in cui una operazione aritmetica che utilizza variabili a virgola mobile genera un risultato più grande/piccolo di quelli rappresentabili con la parola a disposizione

$7{,}6252 = 76252 \cdot 10^{-4}$

| 7 | 6 | 2 | 5 | 2 |
|---|---|---|---|---|

$12345 = 12345 \cdot 10^{0}$

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|

| 1 | 2 | 3 | 4 | 5 | 0 | 0 | 0 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | 7 | 6 | 2 | 5 | 2 |

| 1 | 2 | 3 | 5 | 2 | 6 | 2 | 5 | 2 |
|---|---|---|---|---|---|---|---|---|

# NUMERI REALI

## Virgola mobile: operazioni moltiplicazione

- La **moltiplicazione** in virgola mobile si articola in:
  1. somma degli esponenti
  2. prodotto delle mantisse
  3. Normalizzazione (ed eventuale arrotondamento delle mantisse)

**Esempio.**
Dati i valori 980,12 e 50,00:

Rappresentazione: $9{,}8012 \cdot 10^2$ e $5 \cdot 10^1$

Somma esponenti: 2+1

Prodotto mantisse: 49,006

Normalizzazione mantissa:4, $9006 \cdot 10^1$

Rappresentazione: $4{,}9006 \cdot 10^{1+(3)}$

# NUMERI REALI

## Virgola mobile: operazioni

Dati i valori **5,703125** e **10,25**:

$(5,703125)_{10}=$ $(101,101101)_2=$ $1,011011010\cdot 2^2$

$(10,25)_{10}=$ $(1010,010000)_2=$ $1,010010000\cdot 2^3$

# NUMERI REALI

## Virgola mobile: addizione

$(5{,}703125)_{10} = (101{,}101101)_2 = 1{,}011011010 \cdot 2^2$
$(10{,}25)_{10} = (1010{,}010000)_2 = 1{,}010010000 \cdot 2^3$

**Somma**

$1{,}011011010 \cdot 2^2$ *e* $1{,}010010000 \cdot 2^3$

Allineamento esponenti:

$1{,}011011010 \cdot 2^2$ *e* $10{,}10010000 \cdot 2^2$

Somma:

$1{,}011011010 + 10{,}100100000$

$11{,}11111101 \cdot 2^2$

Normalizzazione:

$1{,}111111101 \cdot 2^3$

Rappresentazione (normalizzata e polarizzata):

<0, 10000010, 11111111101000000…0> 15,953125

# NUMERI REALI

## Virgola mobile: moltiplicazione

$(5{,}703125)_{10}=$ $(101{,}101101)_2=$ $1{,}011011010\cdot 2^2$
$(10{,}25)_{10}=$ $(1010{,}010000)_2=$ $1{,}010010000\cdot 2^3$

**Moltiplicazione**

$1{,}01101101\cdot 2^2$ *e* $1{,}01001\cdot 2^3$

Somma esponenti:

2+3=5 cioè 10+11=101

Moltiplicazione mantisse:

1,1101001110101

Risultato :

$1{,}1101001110101\cdot 2^5$

Normalizzazione mantissa:

$1{,}1101001110101\cdot 2^5$

Rappresentazione (normalizzata e polarizzata):

<0, 10000100, 11101001110101000000…0> 58,45703125

# NUMERI REALI

## Virgola mobile: operazioni in binario

**Esercizio proposto**

Dati i numeri reali **0,5** e -**0,4375**

1. Rappresentarli in virgola mobile singola precisione in modalità normalizzata e polarizzata
2. Effettuare la somma
   A. Rappresentare il risultato della somma in modalità normalizzata e polarizzata
3. Effettuare il prodotto
   A. Rappresentare il risultato del prodotto in modalità normalizzata con il bit nascosto e polarizzata

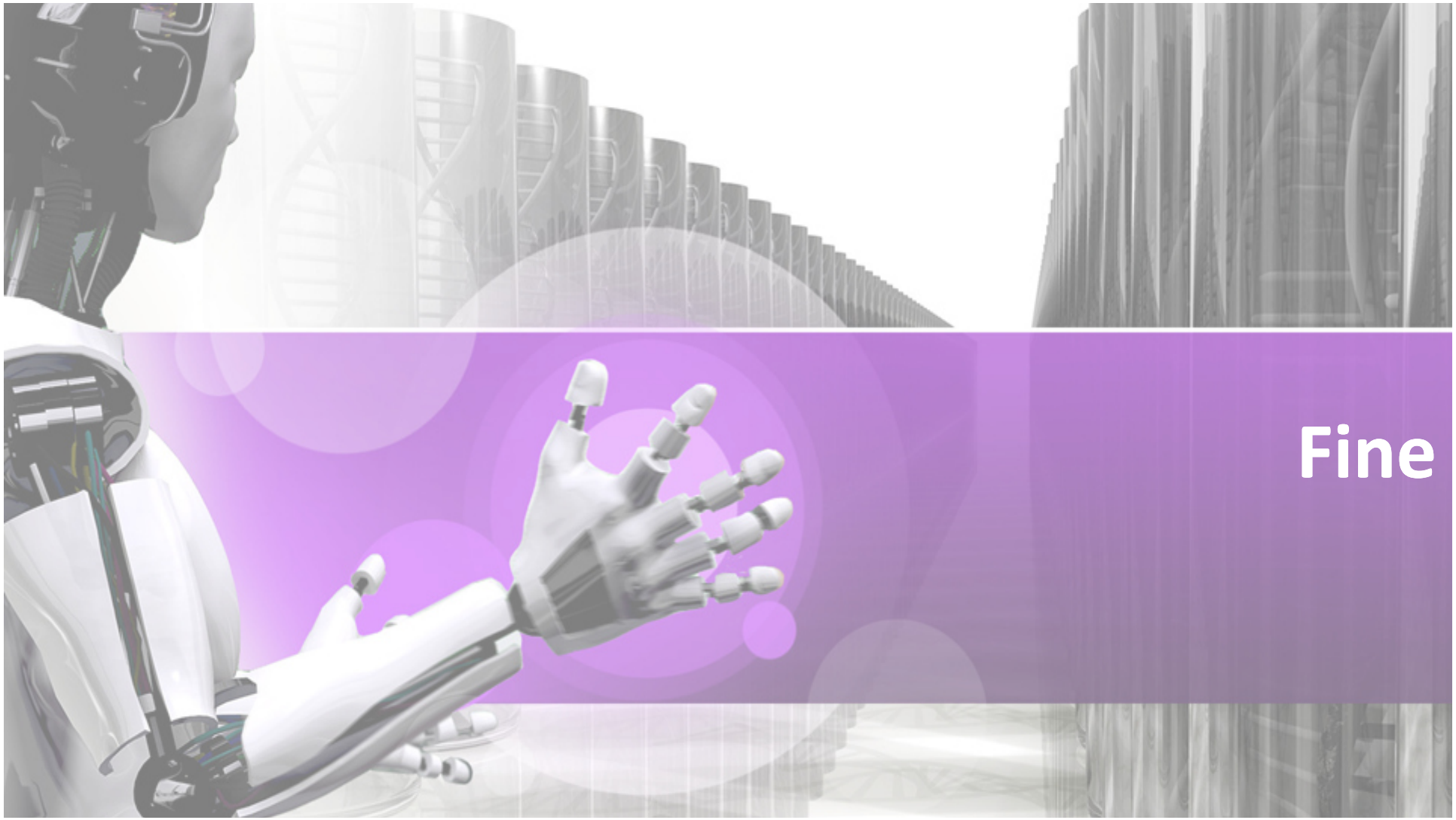
# NUMERI REALI

## Virgola mobile: operazioni in binario

**Esercizio proposto (per casa)**

Dati i numeri reali **12,0625** e -**28,1875**

1. Rappresentarli in virgola mobile singola precisione
2. Effettuare la somma
   A. Rappresentare il risultato della somma in modalità normalizzata e polarizzata
3. Effettuare il prodotto
   A. Rappresentare il risultato del prodotto in modalità normalizzata con il bit nascosto e polarizzata


Fine

# NUMERI REALI

## Virgola mobile: operazioni in binario

**Esercizio proposto**

Dati i numeri reali **0,5** e **-0,4375**

- Rappresentarli in virgola mobile singola precisione in modalità normalizzata e polarizzata

SEGNO=+
PARTE INTERA=0 = $(0)_2$
PARTE FRAZIONATA=0,5=$(0.1)_2$
VALORE=0,1 $\cdot 2^0$
VALORE=$1 \cdot 2^{-1}$

<0,11111111,1>
<0,01111110,00000000000000000000000>

SEGNO=-
PARTE INTERA=0 = $(0)_2$
PARTE FRAZIONATA=0,4375=$(0,0111)_2$
VALORE=0,0111 $\cdot 2^0$
VALORE=$-1,110 \cdot 2^{-2}$

<1,11111110,111000>
<1,01111101,11000000000000000000000>

# NUMERI REALI

## Virgola mobile: operazioni in binario

**Esercizio proposto**

Dati i numeri reali **0,5** e **-0,4375**

- Effettuare la somma
  - Rappresentare il risultato della somma in modalità normalizzata e polarizzata

VALORE1=$+1 \cdot 2^{-1}$
VALORE2=$-1,110 \cdot 2^{-2}$

**ALLINEAMENTO**
VALORE1=$+1 \cdot 2^{-1}$
VALORE2=$-0,1110 \cdot 2^{-1}$

**SOTTRAZIONE**

| | SOTT |
|---|---|
| 1,000- | *0,110-* |
| 0,111= | *0,111=* |
| 0,001 | *0,001* |

**NORMALIZZAZIONE**
$0,001=1,0 \cdot 2^{-3}$ ($\cdot 2^{-1}$)

Cioè:
$+1,0 \cdot 2^{-4}$

<0,01111011,00000000000000000000>

# NUMERI REALI

## Virgola mobile: operazioni in binario

**Esercizio proposto**

Dati i numeri reali **0,5** e -**0,4375**

- Effettuare la somma (variante con il complemento a due

VALORE1=$+1\cdot 2^{-1}$
VALORE2=$-1,11\cdot 2^{-2}$

**ALLINEAMENTO**
VALORE1=$+1\cdot 2^{-1}$
VALORE2=$-0,111\cdot 2^{-1}$

**SOTTRAZIONE COMPL 2**

| | | |
|---|---|---|
| VALORE 1:$1000\cdot 2^{-4}$ | 01000 | 01000 |
| VALORE 2:$0111\cdot 2^{-4}$ | 00111 | 11001 |
| | | ——— |
| | | *1*00001 |

$1 \cdot 2^{-4}$

**NORMALIZZAZIONE**
$1\cdot 2^{-4}$

<0,01111011,00000000000000000000>

# NUMERI REALI

## Virgola mobile: operazioni in binario

**Esercizio proposto**

Dati i numeri reali **0,5** e -**0,4375**

- Effettuare il prodotto
  - Rappresentare il risultato della somma in modalità normalizzata e polarizzata

VALORE=$+1 \cdot 2^{-1}$

VALORE=$-1{,}110 \cdot 2^{-2}$

SOMMA ESPONENTI
-3

MOLTIPLICAZIONEù1,1
1,110x1=1,11

NORMALIZZAZIONE
Non necessaria

Cioè:
$-1{,}11 \cdot 2^{-3}$

<1,01111100,11000000000000000000000>