UNIVERSITÀ DEGLI STUDI DELLA TUSCIA

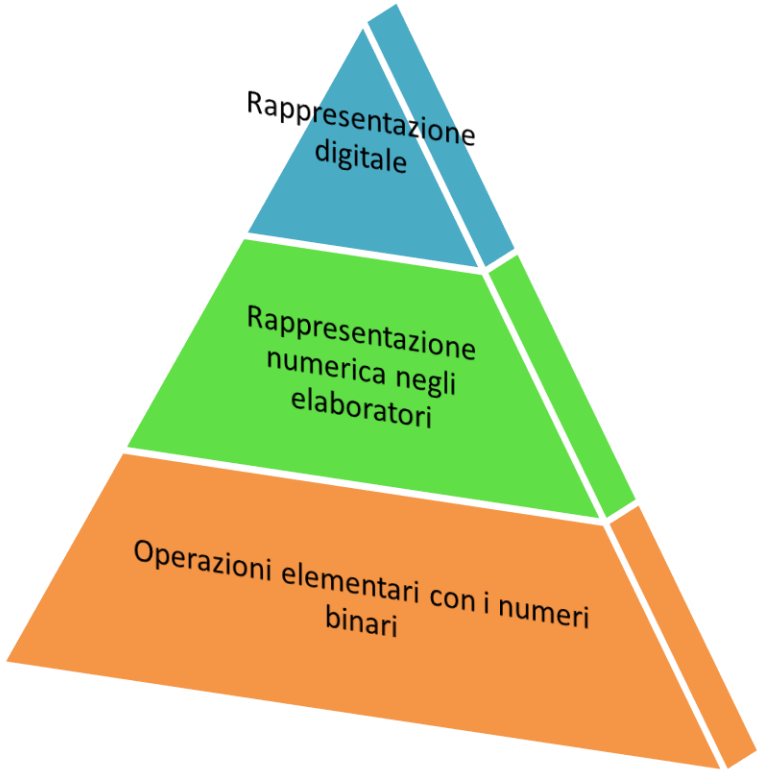
PROGRAMMAZIONE

Fondamenti di informatica

Dott. Franco Liberati
franco.liberati@unitus.it

ARGOMENTI DELLA LEZIONE

- Rappresentazione digitale
- Rappresentazione numerica negli elaboratori
- Operazioni elementari con i numeri binari

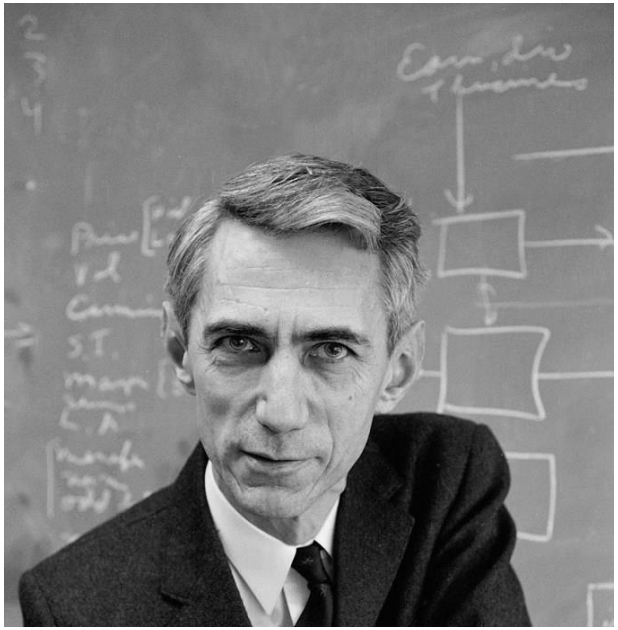


Principi generali della Teoria dell'Informazione

TEORIA DELL'INFORMAZIONE

Generalità

- *A Mathematical Theory of Communication* (1948)
  - Definizione delle componenti fondamentali delle comunicazioni digitali



CLAUDE SHANNON

TEORIA DELL'INFORMAZIONE

Un sistema di comunicazione

- La teoria enunciata da Shannon poneva l'attenzione su **come inviare un messaggio**, rappresentato da una concatenazione di parole, simboli (o segnali), e **ricostruirlo in modo esatto** (o con una buona approssimazione) quando ricevuto da una postazione remota
- Claude Shannon propose uno schema di **sistema di comunicazione** nel quale indicò gli elementi fondamentali costituenti (la sorgente, trasmettitore, canale, ricevitore, destinatario)

SISTEMA DI COMUNICAZIONE DEFINITO DA CLAUDE SHANNON

TEORIA DELL'INFORMAZIONE

Un sistema di comunicazione

- La **sorgente** indica l'insieme delle parole possibili
- Il **trasmettitore** è l'entità che codifica una parola in un segnale
- Il **canale** è il mezzo attraverso il quale si propaga il segnale codificato
- Il **ricevitore** è l'intermediario che decodifica il segnale nella corrispondente parola
- Il **destinatario**, colui al quale si indirizza la collezione di simboli

SISTEMA DI COMUNICAZIONE DEFINITO DA CLAUDE SHANNON

TEORIA DELL'INFORMAZIONE

Alfabeto

- Shannon non prese in considerazione il significato dei termini né del messaggio, ma nel definire la sorgente evidenziò che questa può essere specificata come un **insieme di simboli** possibili
- Lo scienziato statunitense comprese che in un sistema efficiente è importante garantire unicamente che siano i singoli segnali a giungere a destinazione in maniera corretta

Σ:={A, B, C, D, E, F, G, H, I, L, M, N, O, P, Q, R, S,T,U,V, Z}

Σ:={A, B, Γ, Δ, E, Z, H, Θ, I, K, Λ, M, N, Ξ, O, Π, P, Σ, T, Y, Φ, X, Ψ, Ω}

Σ:={A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, W, X, Y, Z}

Σ:={0,1,2,3,4,5,6,7,8,9}

TEORIA DELL'INFORMAZIONE

Alfabeto binario

- Per una comunicazione affidabile Shannon scelse come **alfabeto** quello costituito da due soli simboli 1 e 0, i bit (o 'unità minima di informazione')
- Shannon dimostrò che sfruttando l'alfabeto binario era possibile costituire delle **parole** rappresentanti un messaggio

ESEMPIO

Per specificare, definire, riprodurre o memorizzare dieci parole differenti bisogna - in un sistema di comunicazione con alfabeto binario- associare almeno $\lceil \log_2 10 \rceil = \lceil 3.32 \rceil = 4$ bit

0000, 0001, 0010, 0011, 0100, 0101, 0110, 0111, 1000 e 1001.

TEORIA DELL'INFORMAZIONE

Vantaggi dell'alfabeto binario

- Gli studi di Shannon si concentrarono anche sulla definizione dei criteri (cioè delle formule matematiche) grazie ai quali è possibile ricavare le parole originarie trasmesse lungo un canale affetto da rumore
- La base binaria si dimostrò quella più utile e più robusta nel caso di interferenze
  - Nel caso di errori, in un sistema di comunicazione a cifre binarie il messaggio originale può essere deducibile, cioè rimane 'visibile' anche oltre il disturbo. Inoltre la modifica di una cifra in un sistema non binario induce ad un numero di errori elevato
- Avere un alfabeto binario di riferimento inoltre è comodo perché molti dei componenti negli elettronici (porte logiche, commutatori elettronici, amplificatori) operano proprio con due livelli distinti di corrente elettrica (segnale alto, 1, e segnale basso, 0)

ESEMPIO

Una parola formata da tre simboli binari 000 può essere alterata e ricevuta, al più, come 111; con una distanza 7 dal valore reale (000,001,010,011,100,101,110,111)

Viceversa, nel caso si sfruttasse un alfabeto decimale il valore 000 potrebbe diventare 999, con una distanza mille (una differenza più marcata e ambigua) 000,001,002,003,…,997,998,999

TEORIA DELL'INFORMAZIONE

Alfabeto binario conseguenze

- Oltre a queste nozioni Shannon introdusse il concetto di **ridondanza**, cioè un numero di dati (o caratteri) rappresentanti una parola maggiore di quello richiesto, che sono aggiunti per garantire l'integrità dei messaggi nel caso di eventuali disturbi, e di **codifica economica**, cioè una riscrittura delle parole con un numero minimo di caratteri.

- Dal concetto di ridondanza si derivarono i **codici per il rilevamento e per la correzione degli errori (ECC)**; molto impiegati nelle trasmissioni lungo la rete internet o nell'archiviazione dei dati sui supporti digitali.

- Lo studio inerente ai codici economici, invece, portò alla **compressione dati**, strategia usata per accelerare i tempi di trasmissione dei segnali e per sfruttare al meglio lo spazio dei dispositivi di memorizzazione

ESEMPIO ECC

BIT DI PARITÀ

Si accoda alla parola il bit 1 se il numero di 1 è dispari altrimenti si accoda 0

10101000 1

11101000 1 *ANOMALIA*

ESEMPIO COMPRESSIONE DATI

ALGORITMO RLE

00000000000001111111100000110000

0,13;1,8;0,5;1,2;0,3

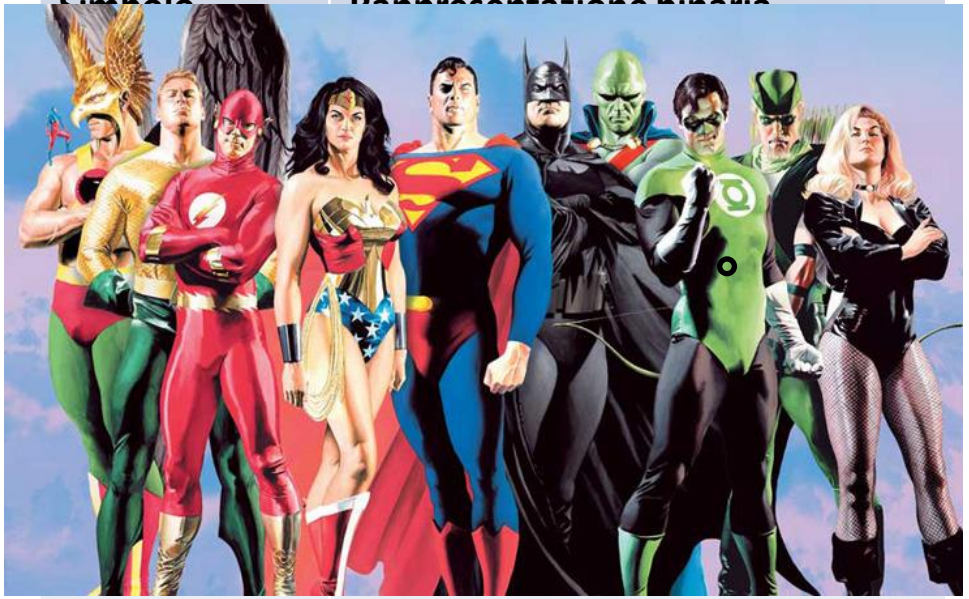
31 caratteri contro 20 (guadagno 20/31 cioè 35.48%)

TEORIA DELL'INFORMAZIONE

Significato avulso dalla parola rappresentante

- Poiché Shannon non focalizzò l'attenzione sull'associazione tra **il significato e la parola rappresentante**, nella teoria dell'informazione questa relazione è di tipo non esclusivo ovvero il significato muta in accordo al dominio di utilizzo: la parola 0 può identificare il colore nero se si considera una immagine o il silenzio se si riproduce un suono o il valore zero se si fa riferimento a numeri

ASCII STANDARD	
Simbolo	Rappresentazione binaria
z	01111010



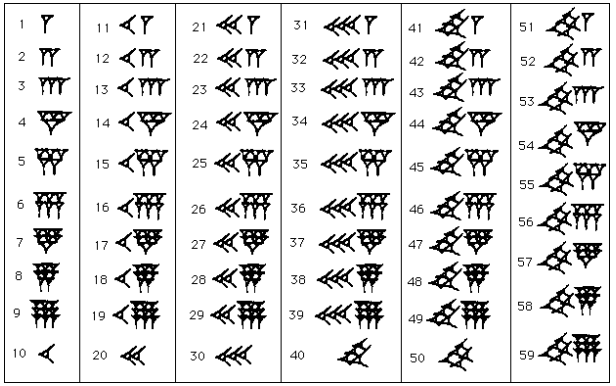
00010000 10110000 01010000

Rappresentazione numerica binaria

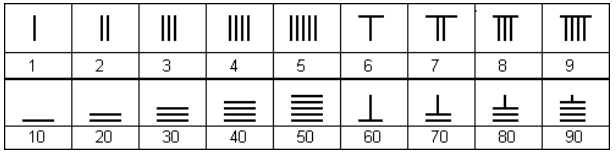
RAPPRESENTAZIONE NUMERICA

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

Numeri e loro rappresentazione

- Nel sistema moderno un **numero** è rappresentato considerando un alfabeto descritto da una **base** (il numero di elementi dell'insieme)
- In campo matematico si usa la base 10
- In campo informatico si fa riferimento al sistema binario (alfabeto {0,1}; base 2); ottale (alfabeto {0,1,2,3,4,5,6,7}; base 8); e esadecimale {alfabeto 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F} base 15)

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

$(2312) \quad = 2 \cdot 10^3 + 3 \cdot 10^2 + 1 \cdot 10^1 + 2 \cdot 10^0$

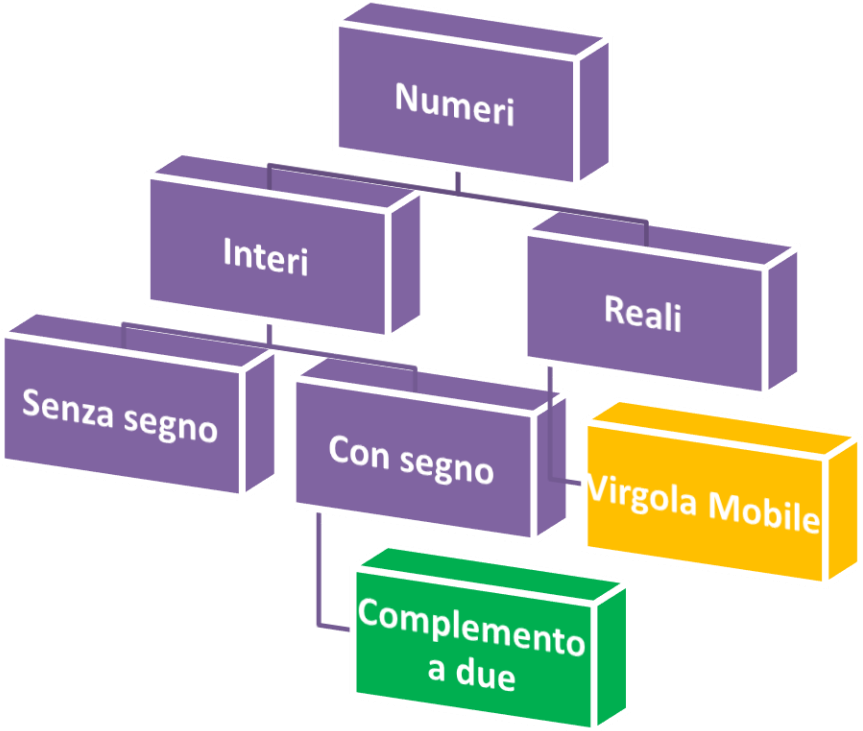
$= 2 \cdot 1000 + 3 \cdot 100 + 1 \cdot 10 + 2 = 2312$

Valore	Base
110582	**10**
110101111111110110	2
327766	8
1AFF6	16

RAPPRESENTAZIONE NUMERICA

Numeri e loro rappresentazione negli elaboratori elettronici

- In campo numerico gli elaboratori elettronici supportano delle funzioni aritmetiche e logiche (nonché logico-aritmetiche) operanti su parole aventi il significato di:
  - **Numeri interi**, per cui si ha una rappresentazione:
    - Senza segno (Numeri Naturali)
    - Con segno (Numeri Interi)
      - Complemento a due
  - **Numeri reali**, per cui si ha una rappresentazione:
    - virgola mobile



RAPPRESENTAZIONE NUMERICA

La dimensione di rappresentazione: 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

INTERO SENZA SEGNO

Base binaria

- 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 $[0;2^n-1]$

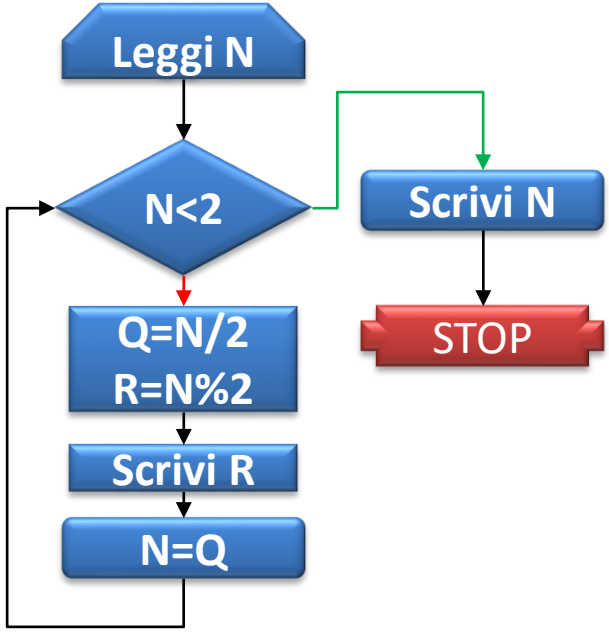
$$N = \sum_{i=0}^{\lceil \log_2 N \rceil} a_i 2^i \qquad 0 \le a_i \le 1$$

DECIMALE	BINARIO (*dimensione parola 3*)
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

INTERI SENZA SEGNO

Rappresentazione in base binaria

- 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 base 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 SENZA SEGNO

Esercizio (in aula)

Parole di 8bit

$(7)_{10}=?_2$ $(00000111)_2$

$(66)_{10}=?_2$ $(01000010)_2$

$(79)_{10}=?_2$ $(01001111)_2$

$(122)_{10}=?_2$ $(01111010)_2$

$(255)_{10}=?_2$ $(11111111)_2$

Parole di 8bit

$(01010100)_2=?_{10}$ $(54)_{10}$

$(00100001)_2=?_{10}$ $(33)_{10}$

$(00000010)_2=?_{10}$ $(2)_{10}$

$(11100011)_2=?_{10}$ $(227)_{10}$

$(11000111)_2=?_{10}$ $(199)_{10}$

INTERI SENZA SEGNO

Rappresentazioni con base non binaria

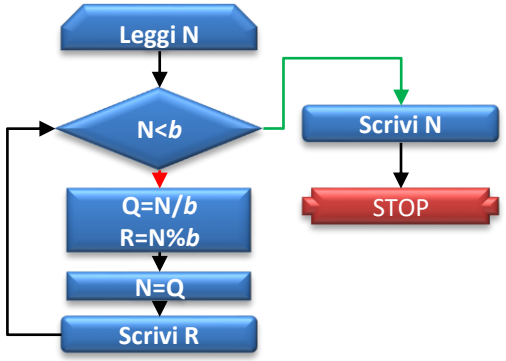
- Per rappresentazioni generiche con base maggiore a 10 si introducono, per convenzione, caratteri alfabetici in maiuscolo
  - Sistema **esadecimale** 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F
  - Sistema in **base venti** 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F,G,H,I,L
- È importante usare simboli per evitare ambiguità

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

  $$(10)_{16}=16 \text{ e } (A)_{16}=10$$

- Superati i simboli alfabetici si ricorre a simboli speciali definibili dall'utente (però 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

Esercizio (in aula)

Parole di 8bit

$(66)_{10}=?_{8}$ $(00000102)_{8}$

$(122)_{10}=?_{16}$ $(0000007A)_{16}$

Parole di 8bit

$(117)_{8}=?_{10}$ $(00000079)_{10}$

$(3B)_{16}=?_{10}$ $(00000059)_{10}$

INTERI SENZA SEGNO

Somma

- L'**addizione tra numeri naturali in base binaria** comporta (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 SENZA SEGNO

Somma: *esempio*

Parole di 8bit

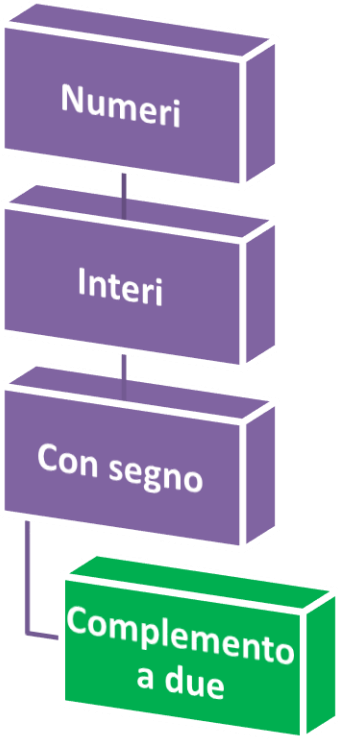
1	1	0	1	0	0	0				Riporto
0	1	1	0	1	1	0	0	108	+	Operando 1
0	0	1	0	1	0	1	0	42	=	Operando 2
1	0	0	1	0	1	1	0	150		Risultato
7	6	5	4	3	2	1	0			*posizione*

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

Rappresentazione in base binaria

- 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
- Il metodo più usato è il **complemento a due**



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 (lo zero diventa uno e viceversa, per ogni bit) 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

Esercizio (in aula)

Parole di 8bit

$(-45)_{10}=?_2$ $(11010011)_2$

$(-72)_{10}=?_2$ $(10111000)_2$

$(-105)_{10}=?_2$ $(10010111)_2$

Parole di 8bit

$(10010010)_2=?_{10}$ $(-110)_{10}$

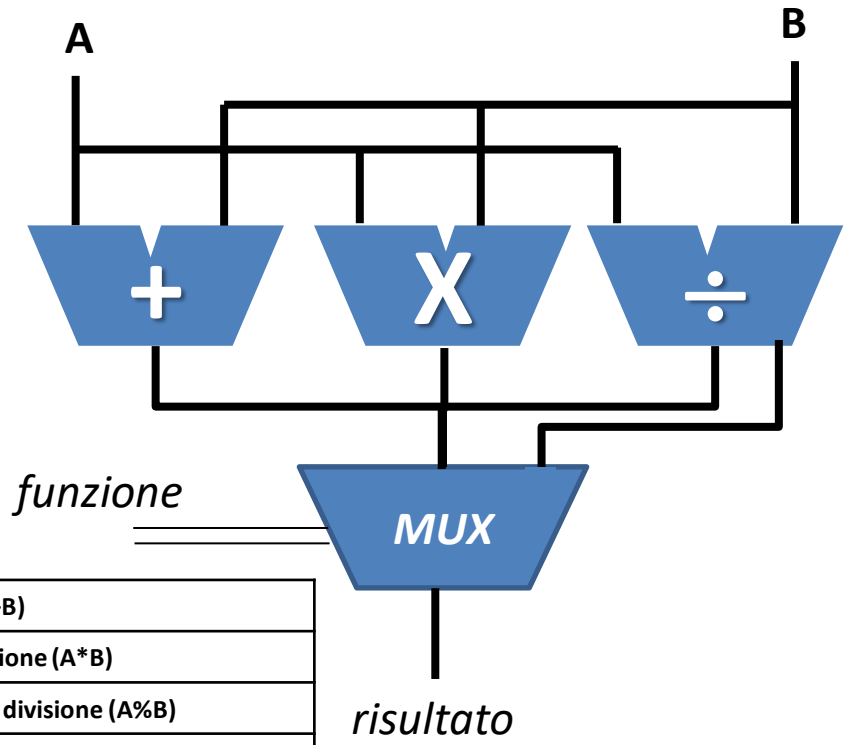
$(00011111)_2=?_{10}$ $(+31)_{10}$

$(10000001)_2=?_{10}$ $(-127)_{10}$

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,…)



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: somma

- La **somma tra numeri in base binaria per interi con segno** si procede come per gli interi senza segno. La somma bit per bit restituisce:
  - **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: somma

Esempio

0	0	1	1	0	1	1	0	1	1	0	1	0	0	0			Riporto
1	0	0	1	1	0	0	1	0	0	1	0	1	1	0	0	-26324	+ Operando 1
0	0	0	1	1	0	1	1	0	1	1	0	1	0	1	0	7018	= Operando 2
1	0	1	1	0	1	0	0	1	0	0	1	0	1	1	0	-19306	Risultato
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		*posizione*

INTERI CON SEGNO

Complemento a due: somma (trabocco)

- La somma di due numeri binari in complemento a due con segno opposto può comportare un **trabocco** (*carry*) che **non influisce 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: somma (overflow)

- Nel 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. **In questo caso oltre al bit in eccesso si deve verificare un cambio di segno**
- L'**overflow influisce negativamente sul risultato**

1	0	0	0	0	0	0	1			Riporto
	1	0	0	0	1	0	0	1	-119	+ Sottraendo
	1	0	1	0	0	0	0	1	-95	= Minuendo
1	0	0	1	0	1	0	1	0	42	(-214) Risultato
	7	6	5	4	3	2	1	0		*posizione*

INTERI CON SEGNO

Complemento a due: sottrazione

- Nel caso della **sottrazione tra numeri in base binaria** si considera la somma tra un numero positivo ed uno negativo
- ES: 25-13 equivale a 25+(-13)

011001 -	011001 +	011001 +
001101=	**110011**=	110011 =
_____	_____	_____
		*1*001100

INTERI CON SEGNO

Esercizio (a casa)

Parole di 8bit

$(77\text{-}45)_{10} = ?_2$

$(50\text{-}60)_{10} = ?_2$

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

- La tabella dei prodotti tra i bit ad uguale posizione prevede un 1 solo se entrambi i bit sono 1

- Il risultato finale è ottenuto dalla somma dei prodotti parziali

Cifra Moltiplicando	Cifra moltiplicatore	Risultato Prodotto
0	0	0
0	1	0
1	0	0
1	1	1

			0	1	0	0	·	
			0	1	1	1	=	
			0	1	0	0	+	
		0	1	0	0		+	
	0	1	0	0			+	
0	0	0	0				=	
0	0	1	1	1	0	0		

NUMERI REALI

Rappresentazione generale

- 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

Rappresentazione canonica

- La **rappresentazione canonica** è definita dalla tripla < *S; M*; *E* > dove:
  - *S* è il *segno* del numero (0, positivo; 1, negativo)
  - *M* la sua *mantissa* cioè il valore assoluto privato della virgola
  - *E* il suo *esponente* ovvero il numero positivo o negativo che indica la posizione della virgola

*(quando si sposta la virgola a destra il valore di E è il numero, **negato**, di spostamenti; quando si sposta la virgola a sinistra il valore di E corrisponde al numero **non negato** di spostamenti)*

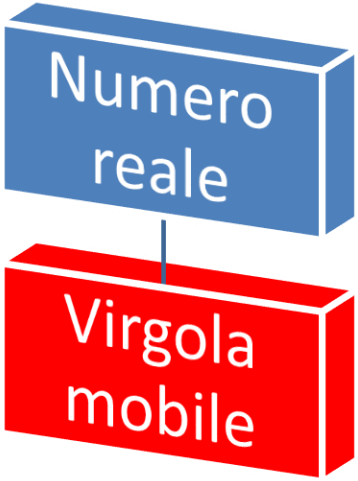
$$\Re = (-1)^S \cdot M \cdot 10^E$$

+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

Rappresentazione negli elaboratori elettronici

- Nei calcolatori elettronici i numeri reali sono rappresentati nel formato **virgola mobile** (*floating point*)
- 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)



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

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

Segno	Esponente	Mantissa
1 bit	8 bit	23 bit

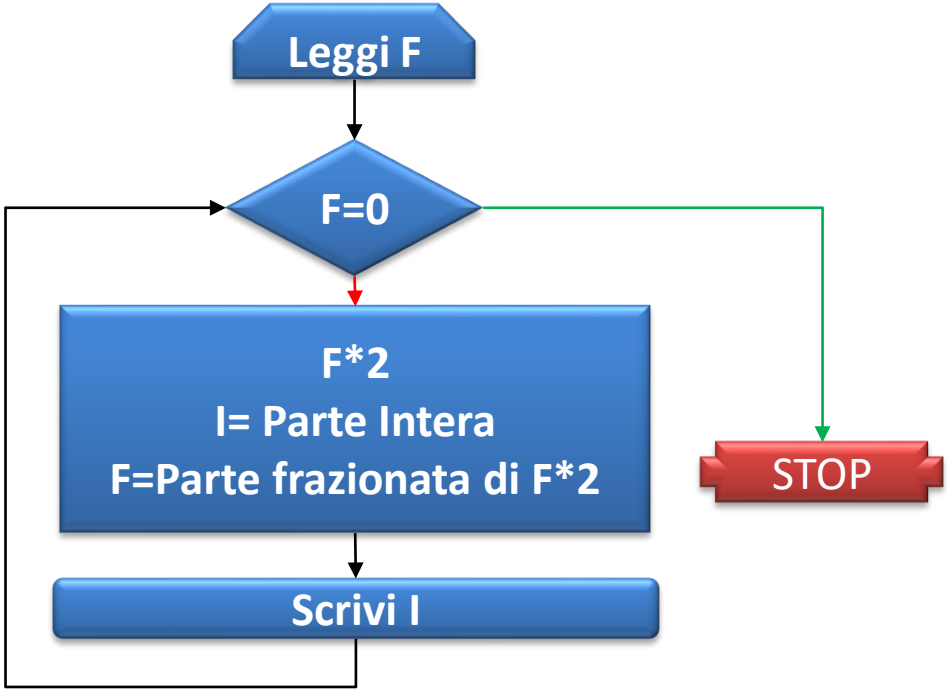
Doppia precisione

Segno	Esponente	Mantissa
1 bit	11 bit	52 bit

NUMERI REALI

Virgola mobile: rappresentazione

- Nella rappresentazione in virgola mobile in base 2 la **parte intera** si codifica escludendo il segno e convertendo il valore decimale in binario
- La parte frazionata si ottiene sfruttando l'algoritmo dei prodotti successivi
- La **parte intera in binario ha pesi positivi**, quella **frazionata ha pesi negativi**



NUMERI REALI

Virgola mobile: calcolo parte frazionata

Considerato il valore reale: **-5.125**

- Il segno è negativo quindi S=1
- Alla parte intera 5 si applica l'algoritmo delle divisione successive e in binario diventa 101
- Alla parte frazionata 0.125 si applica l'algoritmo delle moltiplicazioni successive:

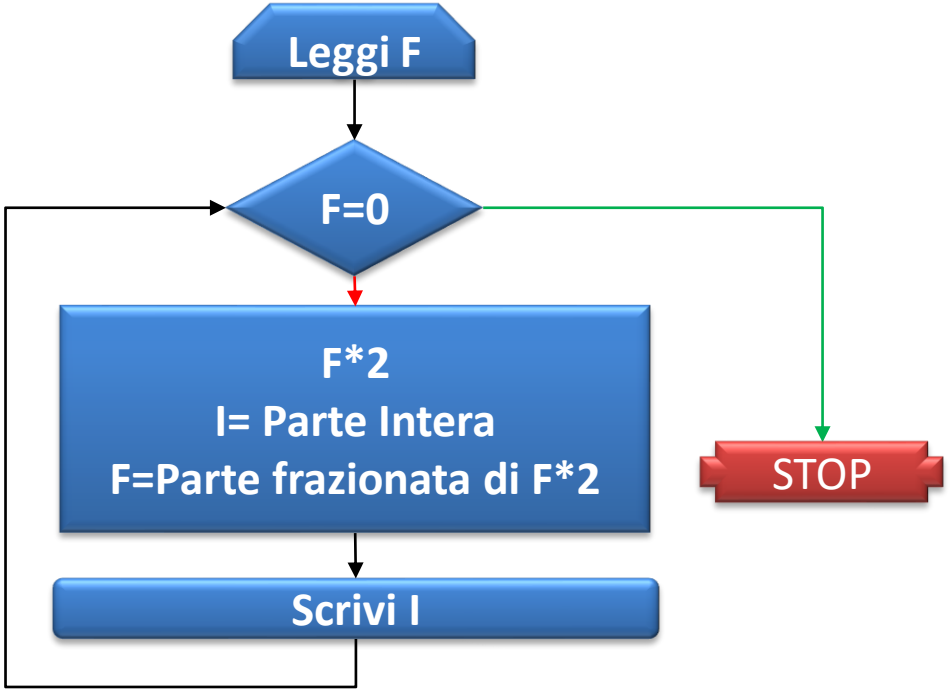
  0.125*2=0.25

  0.25*2=0.5

  0.5*2=1.00
- Il risultato è:

  <1,101.001,0>

  <1,101001,-3>



NUMERI REALI

Virgola mobile: da virgola mobile a reale

- Contro verifica

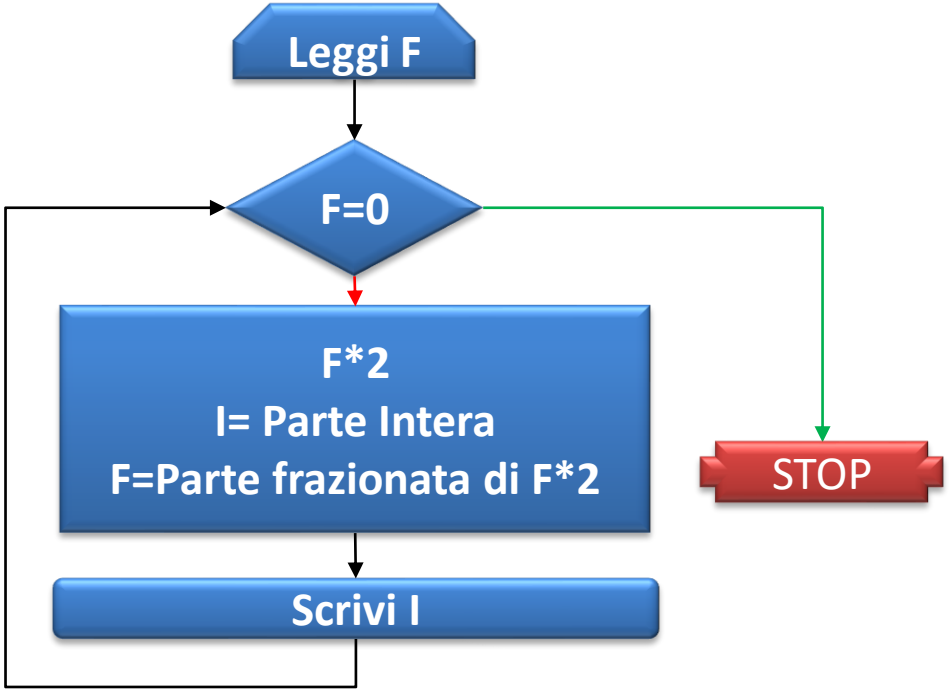
<1,101.001,0>

S=1 indica il segno –
infatti $(-1)^1=-1$

La mantissa 101.001 equivale a
$1*2^2+0*2^1+1*2^0+0*2^{-1}+0*2^{-2}+1*2^{-3}$

L'esponente e $2^0=1$

Cioè $-1*5.125*1=-5.125$



NUMERI REALI

Virgola mobile: da virgola mobile a reale

- Contro verifica

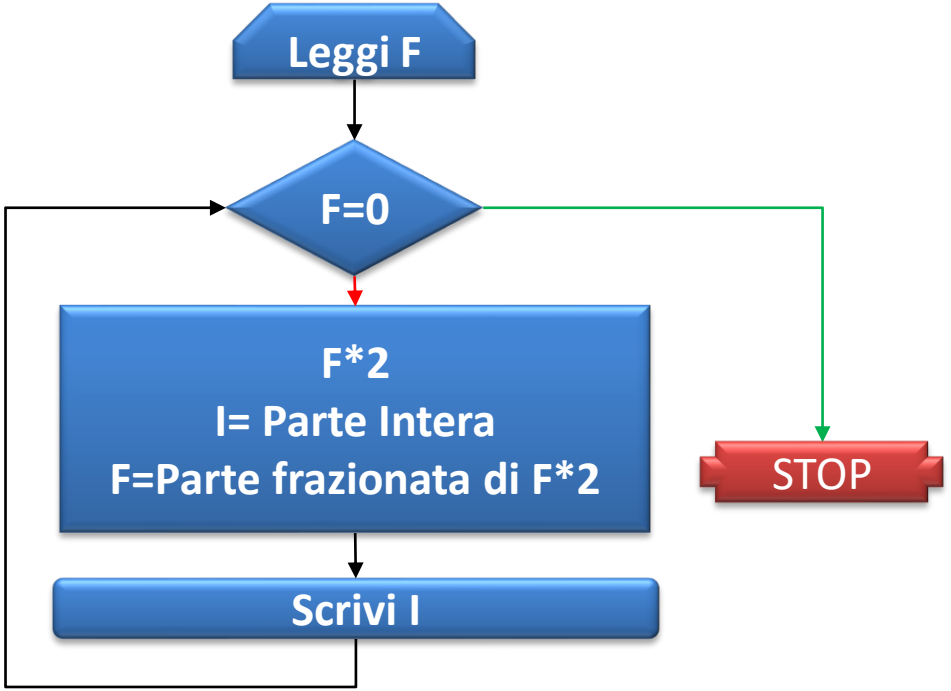
<1,101001,-3>

S=1 indica il segno –
infatti $(-1)^1=-1$

La mantissa 101001 in decimale equivale a 41

L'esponente in decimale è 2^{-3} che equivale a 0,125

Cioè -1*41*0,125=-5.125



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 $+0,01010101100100010...0\cdot2^{7}$	$(0)_2$	$(101010110010001)_2$	00000111 (+7)
Normalizzata $+1,010101100100010...0\cdot2^{5}$	$(0)_2$	$(101010110010001)_2$	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 **+1,010101100100010…0·2^5**	$(0)_2$	$(101010110010001)_2$ 23bit	00000101 (5) 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 $+1,010101100100010\ldots0 \cdot 2^{5+127}$	0 1bit	$(10000100)_2$ 8 bit	$(10101011001000 10\ldots0)_2$ 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: $-1{,}01001 \cdot 2^2$
- Rappresentazione:
  Ë=2+127=129=$(10000001)_2$.
  M=$(10100100000000000000000)_2$

Segno	Esponente polarizzato	Mantissa con bit nascosto
1	10000001	01001000000000000000
	8bit	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

Segno	Esponente polarizzato	Mantissa con bit nascosto
1	10000001	01001000000000000000
	8bit	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=$(1010001000000000000000000000000000000000000000000000)_2$

Segno	Esponente polarizzato	Mantissa con bit nascosto
0	**0**10000000010	0100010000000000000000000000000000000000000000000000
	11bit	52bit

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

NORM: <0,11111111,1>
POL:<0,01111110,00000000000000000000000>

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

NORM:<1,11111110,111000>
POL:<1,01111101,11000000000000000000000>

NUMERI REALI

Virgola mobile: operazione di somma

- L'algoritmo di **addizione** (e sottrazione) in virgola mobile si divide in quattro fasi fondamentali:
  1. Allineamento degli esponenti (gli esponenti assumono lo stesso valore... Si deve modificare la posizione della virgola nella mantissa)
  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: operazioni di somma (esempio)

Dati i valori **5,703125** e **7,25**:

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

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

NUMERI REALI

Virgola mobile: operazione di somma (esempio)

Somma

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

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

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

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, 11111111010000000…0> 15,953125

NUMERI REALI

Virgola mobile: operazioni di 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 di moltiplicazione (es.)

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

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

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: 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


Fine