

Architettura degli Elaboratori 2020-2021

Rappresentazione numeri reali Rappresentazione altre informazioni

Prof. Elisabetta Fersini
elisabetta.fersini@unimib.it

Rappresentazione di numeri reali

- I numeri reali possono essere rappresentati come segue:
 - Virgola fissa
 - Virgola mobile (*floating point*)

Virgola fissa

- Il metodo più semplice per rappresentare numeri reali attraverso codici binari è quello detto in **Virgola Fissa**.
- Avendo a disposizione N bit essi vengono usati in questo modo:
 - 1 bit per il **segno** del numero da rappresentare
 - $I < (N-1)$ bit per rappresentare la **parte intera** del numero
 - $D = N - (I+1)$ bit per rappresentare la **parte decimale** del numero

+/-	I-1	I-2	...	0	-1	-2	...	-D
Segno	Parte Intera				Parte Frazionaria			

Virgola fissa

- Un sistema di numerazione in virgola fissa è quello in cui:
 - La posizione della virgola decimale è implicita
 - La posizione della virgola decimale uguale in tutti i numeri
- Con questo metodo l'**intervallo di numeri interi** rappresentabili è

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

- L'**intervallo** rappresentabile dalla **parte decimale** è

$$[0, 1/2^D].$$

Virgola fissa

- Supponiamo di volere rappresentare numeri reali con 8 bit. Possiamo decidere di dedicare:
 - 1 bit al segno, 3 bit alla parte intera e 4 bit alla parte decimale.
 - Oppure 1 bit al segno, 5 bit alla parte intera e 2 bit alla parte decimale.
- La scelta è dettata dalle necessità pratiche.
 - Se volessimo rappresentare numeri che hanno una parte intera grande (e quindi che hanno un valore I grande) ma una precisione piccola (dedicando meno bit alla parte decimale del numero) sceglieremmo la **seconda opzione**.
 - Se volessimo rappresentare numeri che hanno parte intera piccola ma una grande precisione decimale, sceglieremmo la **prima opzione**.

Virgola fissa

- Consideriamo il numero 5.125 in base 10. Esso viene rappresentato in binario in virgola fissa come segue:
- Si considera la **parte intera** 5 e la si riporta in base 2

$$5_{10} = 101_2$$

- La **parte decimale** viene scomposta per moltiplicazioni successive:

$$0.125 \times 2 = 0.25 \quad \rightarrow \quad 0 \quad \text{e riporto } 0.25$$

$$0.25 \times 2 = 0.5 \quad \rightarrow \quad 0 \quad \text{e riporto } 0.5$$

$$0.5 \times 2 = 1 \quad \rightarrow \quad 1 \quad \rightarrow \text{stop}$$

- Quindi $5.125_{10} = 101.001_2$

Virgola fissa

- Per rappresentare un numero binario in virgola fissa in base decimale, ricorriamo alla definizione vista nelle lezioni relative alle conversioni da base a base.
- Consideriamo il numero 101.01_2 , e rappresentiamolo in base 10

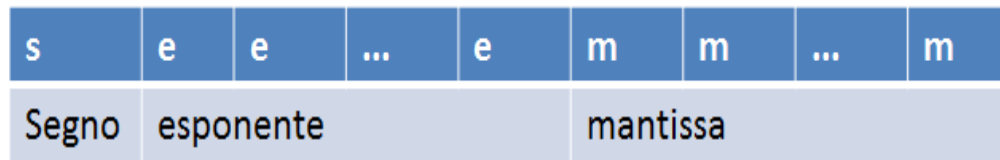
$$101.01_2 = 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 + 0 \cdot 2^{-1} + 1 \cdot 2^{-2} = 4 + 0 + 1 + 0 + 0.25 = 5.25_{10}$$

Virgola fissa

- **Svantaggi:**
 - Rigidità della posizione assegnata alla virgola
 - Sono fissi i bit assegnati per codificare la parte intera e la parte frazionaria
 - Impatta sulla precisione nel codificare i numeri
 - Maggiore è il numero di bit per codificare la parte intera, più bassa sarà la precisione nel codificare i numeri piccoli

Virgola mobile

- Usa un bit per rappresentare il **segno s**
- Usa altri bit per rappresentare la **mantissa m**
- Usa altri per codificare l'**esponente e**



- Dal 1985 adotta lo standard IEEE¹ 754² nella base 2.

¹IEEE, Institute of Electrical and Electronics Engineers

²Adottata da tutti i programmi e le componenti di calcolo dell'elaboratore

- La **posizione** della **virgola** è **variabile** per avere una rappresentazione in notazione scientifica in cui:
 - un'unica cifra a sinistra della virgola
 - una parte frazionaria
 - un esponente al quale si deve elevare la base del numero
- Esempi:

$$546.768_{10} \rightarrow 5.46768_{10} \cdot 10^2$$

$$1011.0110_2 \rightarrow 1,0110110_2 \cdot 2^3$$

Osservazione:

Nell'esempio l'esponente della base 2 è stato rappresentato per semplicità grafica in base 10; in realtà il calcolatore rappresenta in base 2 anche l'esponente

Virgola mobile

- Estende l'intervallo di numeri rappresentati a parità di cifre, rispetto alla notazione in *virgola fissa*
- Numeri reali rappresentati da una coppia di numeri:
 - Mantissa (M)
 - Esponente (E)
 - Segno (S)
 - Un numero X sarà scritto come $X = (-1)^S * M * B^E$

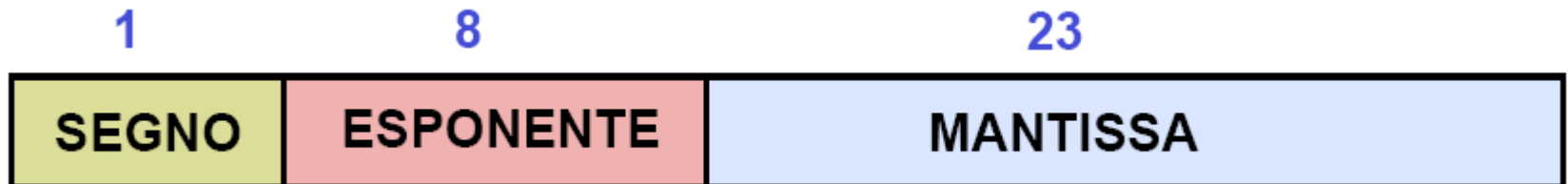
Virgola mobile

- La notazione scientifica per la base 2:

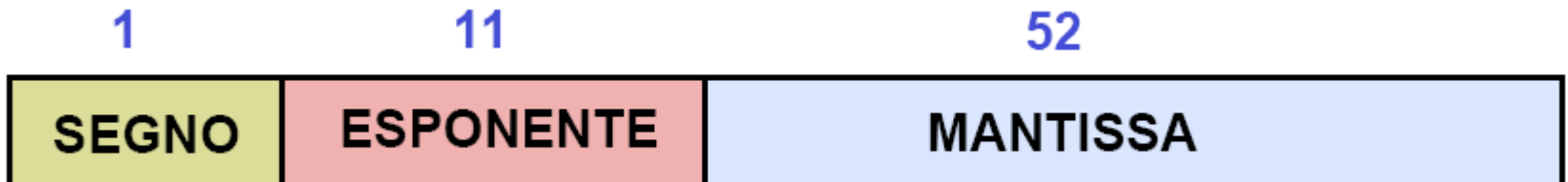
$$1,xx...xx_2 \cdot 2^{yy...yy_2}$$

dove le x rappresentano la parte frazionaria e le y l'esponente a cui elevare la base 2

- Semplice/singola precisione su 32 bit (notazione MS):



- Doppia precisione su 64 bit (notazione MS):



- Come convertire un numero reale da base 10 in base 2, in virgola mobile?
- Esempio: convertire $(7.5)_{10}$ in base 2.
 - si converte la parte intera $(7)_{10} = (111)_2$;
 - si considera la parte frazionaria $(0.5)_{10} = (1.0)_2$;
 - si considera il numero binario ottenuto convertendo parte intera e parte frazionaria

$$(111.1)_2 = 1 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 + 1 \cdot 2^{-1}$$

- Osservazione: l'esponente può assumere valori negativi (quindi i numeri in virgola mobile possono essere rappresentati in MS e CA2)
- Una rappresentazione comune rappresenta:
 - l'esponente in eccesso 127 (per la semplicità dei calcoli)
 - la mantissa nell'intervallo [1, 2)
- Per calcolare il valore di un numero in virgola mobile:

$$(-1)^S \cdot (1 + M') \cdot 2^{(E-127)}$$

dove $M=1+M'$

Standard IEEE 754: normalizzata

- ▶ La codifica da binario a decimale dell'esponente assume un valore intero compreso tra 1 e 254 (estremi inclusi).
- ▶ La mantissa prevede una qualunque sequenza di bit.
- ▶ Numero corrispondente ad un float (32 bit) risulta:

$$N_{10} = (-1)^s \cdot 1.m \cdot 2^{e-127}$$

m é il numero intero ottenuto dalla codifica da binario a decimale della mantissa;

e é il numero intero ottenuto dalla codifica da binario a decimale dell'esponente;

s é il segno;

la costante 127 é detta di polarizzazione.

- Il **range** è dato dal numero di bit dell'esponente
 - Il numero reale float più piccolo in valore assoluto si ha quando la mantissa m è composta da tutti 0 e l'esponente e assume il valore minimo, ossia 1:
$$N_{\min} = 1.(0000000000 \dots 0)_2 \cdot 2^{1-127} = 1.0 \cdot 2^{-126}$$
 - Il numero reale float più grande si ha quando la mantissa m è composta da tutti 1 e l'esponente e assume il valore massimo, ossia 254:
$$N_{\max} = 1.(1111111111 \dots 1)_2 \cdot 2^{254-127} = 1.(1111111111 \dots 1)_2 \cdot 2^{127} \approx 3.4 \times 10^{38}$$
- La **precisione** è data dal numero di bit nella mantissa

Errore assoluto ed errore relativo

- Rappresentando un numero reale n in virgola mobile si commette un errore di approssimazione.
- In realtà viene rappresentato un numero razionale n' con un numero limitato di cifre significative:

ERRORE ASSOLUTO: $e_A = n - n'$

ERRORE RELATIVO: $e_R = e_A / n = (n - n') / n$

- L'ordine di grandezza dell'errore assoluto dipende dal *numero di cifre significative e dall'ordine di grandezza del numero*
- L'ordine di grandezza dell'errore relativo dipende solo dal *numero di cifre significative*

Rappresentazione di altre informazioni

- Caratteri
- Suoni
- Video
-

Rappresentazione di caratteri

- Possiamo associare a ogni **carattere** (quale lettera minuscola, lettera maiuscola, vocale accentata e segno di interpunzione) un numero.
- I caratteri possono essere rappresentati in:
 - **ASCII standard**: 1 carattere è rappresentato con 7 bit per un totale di 128 simboli rappresentabili (quali cifre, lettere maiuscole e lettere minuscole);
 - **ASCII estesa**: 1 carattere è rappresentato con 8 bit rappresentabili fino a 256 simboli (i caratteri in più sono usati per esempio per caratteri accentati);
 - **UNICODE**: 1 carattere è rappresentato con un numero maggiore di bit (tra 8 e 32 bit per carattere).

ASCII Standard

- ASCII standard contiene:
 - 26 + 26 lettere (maiuscole + minuscole)
 - 10 cifre decimali (da 0 a 9)
 - segni di interpunzione
 - caratteri di controllo
- Le cifre sono ordinate per valore
- Le lettere maiuscole sono ordinate alfabeticamente
- Le lettere minuscole sono ordinate alfabeticamente (e sono a distanza fissa dalle maiuscole)

ASCII Standard

- Dal 0 a 31 sono dei caratteri di controllo per periferiche
- Da 32 a 47 vari caratteri
- da 48 a 57 cifre decimali
- Da 58 a 64 vari caratteri
- Da 65 a 90 lettere maiuscole dell'alfabeto
- Da 91 a 96 vari caratteri
- Da 97 a 122 lettere minuscole dell'alfabeto
- Da 123 a 127 vari caratteri

ASCII Standard

bit		000	001	010	011	100	101	110	111
	esad.	0	1	2	3	4	5	6	7
0000	0	NUL	DLE	spz	0	@	P	`	p
0001	1	SOH	DC1	!	1	A	Q	a	q
0010	2	STX	DC2	"	2	B	R	b	r
0011	3	ETX	DC3	#	3	C	S	c	s
0100	4	EOT	DC4	\$	4	D	T	d	t
0101	5	ENQ	NAK	%	5	E	U	e	u
0110	6	ACK	SYN	&	6	F	V	f	v
0111	7	BEL	ETB		7	G	W	g	w
1000	8	BS	CAN	(8	H	X	h	x
1001	9	HT	EM)	9	I	Y	i	y
1010	A	LF	SS	*	:	J	Z	j	z
1011	B	VT	ESC	+	;	k	[k	{
1100	C	FF	FS	,	<	L	\	l	
1101	D	CR	GS	-	=	M]	m	}
1110	E	SOH	RS	.	>	N	^	n	~
1111	F	SI	US	/	?	O		o	DEL

$p(A) = 100\ 0001$

→ 65

$$65 = 1 \times 2^0 + 1 \times 2^6$$

$p(\{) = 111\ 1011$

$$123 = 1 \times 2^0 + 1 \times 2^1 + 1 \times 2^3 + 1 \times 2^4 + 1 \times 2^5 + 1 \times 2^6$$

→ 123

La conversione
da b=2 a b=10 va da dx
verso sx.

ASCII Standard

- Le parole sono sequenze di caratteri.

01101001 i	01101110 n	01100110 f	01101111 o	01110010 r
01101101 m	01100001 a	01110100 t	01101001 i	01100011 c
01100001 a				

bit		000	001	010	011	100	101	110	111
	esad.	0	1	2	3	4	5	6	7
0000	0	NUL	DLE	spz	0	@	P	.	p
0001	1	SOH	DC1	!	1	A	Q	a	q
0010	2	STX	DC2	"	2	B	R	b	r
0011	3	ETX	DC3	#	3	C	S	c	s
0100	4	EOT	DC4	\$	4	D	T	d	t
0101	5	ENQ	NAK	%	5	E	U	e	u
0110	6	ACK	SYN	&	6	F	V	f	v
0111	7	BEL	ETB		7	G	W	g	w
1000	8	BS	CAN	(8	H	X	h	x
1001	9	HT	EM)	9	I	Y	i	y
1010	A	LF	SS	*	:	J	Z	j	z
1011	B	VT	ESC	+	:	k	[k	{
1100	C	FF	FS	,	<	L	\	l	
1101	D	CR	GS	-	=	M]	m	}
1110	E	SOH	RS	.	>	N	^	n	~
1111	F	SI	US	/	?	O		o	DEL

- Con 1 byte ($2^8=256$) è possibile realizzare 256 diverse combinazioni, quantità sovrabbondante per la descrizione dei vari caratteri della tabella standard (un bit di troppo).
- L'ultimo bit alla descrizione del carattere viene definito parity bit, dedicato al controllo di parità (parity check).
 - Il bit di parità è un bit di controllo usato per prevenire errori nella trasmissione o nella memorizzazione dei dati. Tale sistema prevede l'aggiunta di un «bit ridondante», calcolato a seconda che il numero di bit che valgono 1 sia pari o dispari.
 - Se un numero dispari di bit è cambiato durante la trasmissione, allora il bit di parità non sarà corretto e indicherà che è avvenuto un errore durante la trasmissione

ASCII Esteso

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
20		!	"	#	\$	%	&	'	()	*	+	,	-	.	/
30	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
40	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
50	P	Q	R	S	T	U	V	W	X	Y	Z	[\]	^	_
60	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
70	p	q	r	s	t	u	v	w	x	y	z	{		}	~	
80			,	f	„	...	†	‡	^	%	Š	<	Œ			
90		‘	’	“	”	•	—	—	~	™	š	>	œ			ÿ
A0		¡	¢	£	¤	¥	¦	§	¨	©	ª	«	¬	-	®	¯
B0	°	±	²	³	´	µ	¶	·	,	¹	º	»	¼	½	¾	¿
C0	À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í	Î	Ï
D0	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß
E0	à	á	â	ã	ä	å	æ	ç	è	é	ê	ë	ì	í	î	ï
F0	ð	ñ	ò	ó	ô	õ	ö	÷	ø	ù	ú	û	ü	ý	þ	ÿ

- La tabella ASCII estesa varia come già accennato in base alla zona geografica di utilizzo e al software utilizzato. Le principali estensioni previste dall'ISO 8859 sono:
 - [ISO-8859-1\(Latin-1\)](#), utilizzato nella Zona Europea Occidentale
 - [ISO-8859-2 \(Latin-2\)](#), utilizzato zona Europea Orientale (Serbia, Albania, Ungheria, Romania)
 - [ISO-8859-3 \(Latin-3\)](#), utilizzato nell' Europea del Sud (Malta), include l'Esperanto
 - [ISO-8859-4 \(Latin-4\)](#), obsoleto
 - [ISO-8859-5 \(Part 5, Cyrillic\)](#), alfabeto Cirillico
 - [ISO-8859-6 \(Part 6, Arabic\)](#), alfabeto Arabo
 - [ISO-8859-7 \(Part 7, Greek\)](#), alfabeto Greco
 - [ISO-8859-8 \(Part 8, Hebrew\)](#), alfabeto Ebraico

- ASCII è un codice accettato da tutti i computer.
 - Usato dai tempi delle telescriventi durante la prima guerra mondiale.
- Tuttavia non considera i caratteri internazionali di numerose lingue straniere.
- Per ovviare a tale problematica, è stata introdotta un'ulteriore codifica, ossia UNICODE

- E' una evoluzione dello standard ASCII
- Unicode è uno standard per la rappresentazione di testo
- Codifica tutti i caratteri utilizzati nelle principali lingue del mondo
- Indipendente dalla lingua, dal sistema operativo e dal programma utilizzato
- Inizialmente rappresentato come una codifica su 16 bit, ma poi esteso a 24 e 32 bit
 - Disporre di 32 bit significa avere 4 miliardi di caratteri diversi codificabili!
- Unicode è in continua evoluzione e continua ad aggiungere sempre più caratteri

- Un carattere UNICODE è caratterizzato dal suo codice numerico, detto code point, solitamente rappresentato con 8 cifre esadecimali
 - Esempio: «fi» è rappresentato dal codice 0000FB01
 - Esempio: il simbolo “do doppio diesis strumentale” della notazione musicale greca antica (simile a una lambda maiuscola con una gambetta) è 0001D235
- Con UNICODE, è possibile creare e gestire senza troppa pena documenti multilingue:

A, Δ, Ъ, 7, ๗, あ, 叶, 葉

- In particolare, tutti gli standard W3C (incluso HTML) supportano UNICODE
- A marzo 2020 è stato presentato l'ultima versione UNICODE 13.0

Il problema della codifica

- UNICODE può codificare 4.294.967.296 caratteri distinti
- Ogni carattere occupa 32 bit (contro gli 8 delle altre codifiche); i documenti richiedono quindi 4 volte lo spazio
- La quasi totalità dei documenti usa da 60 a 1000 caratteri, per cui basterebbero da 6 a 10 bit.
- Per ovviare a questo problema, e garantire maggiore compatibilità con S.O. e applicazioni che non sono in grado di gestire 32 bit per carattere, UNICODE definisce vari formati di codifica più compatti

- UTF-8 (8-bit UCS/Unicode Transformation Format) è una codifica a lunghezza variabile fra una sequenza di valori a 8 bit e una sequenza di caratteri UNICODE
 - I primi 128 caratteri di UNICODE (0-7F), equivalenti ai caratteri ASCII, sono codificati con il loro codice “naturale”
 - Tutti gli altri caratteri sono codificati con due, tre o quattro valori a 8 bit (byte)

UTF-8

- Nel linguaggio di programmazione Java (e derivati), le stringhe sono codificate con UTF-8; i programmi Java sono quindi in grado di gestire nativamente UNICODE.
- I file system Macintosh, DVD, e alcuni su UNIX usano UTF-8 per i nomi dei file.
- Gli standard relativi al Web e alla e-mail richiedono che un programma compatibile supporti *almeno* UTF-8 come standard di codifica.
- I programmi che trattano testi ASCII sono generalmente UTF-8 compatibili.

Link di interesse

- Il sito principale relativo a UNICODE è <http://www.unicode.org>
 - la pagina <http://unicode.org/charts/> è particolarmente affascinante
- Alcune emoji codificate in UNICODE 13.0
 - <https://youtu.be/cxRDjci6POs>

- [Tabella ASCII con rappresentazione DEC + BIN + HEX + OCT File](#)
- Numeri in virgola mobile: Appendice B (esclusa parte su IEEE754) libro A. S. Tanenbaum, "Structured computer Organization", 5th ed.
- Rappresentazione dei numeri in virgola mobile: Appendice B (solo parte relativa all'introduzione dello standard IEEE754) del libro di A. S. Tanenbaum, "Structured computer Organization", 5th ed.