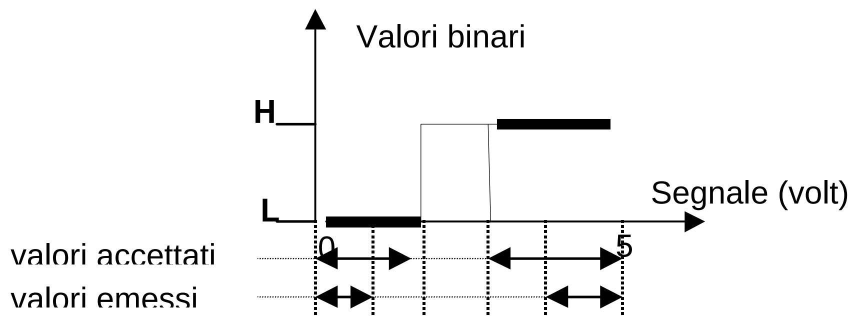
- 2° Modulo –
  - La rappresentazione dell'Informazione
  - Struttura dei programi C
  - Tipi Semplici del C
  - Costanti Variabili, Operatori
  - Espressioni
  - Operatori aritmetici, logici, bitwise

http://www.elet.polimi.it/upload/martucci/index.html

Indice

Segnali digitali binari e rappresentazione delle informazioni

La grandezza fisica che si utilizza (segnale elettrico di tensione) assume solo **due valori discreti** (binaria)



L'elemento tecnologico base per la realizzazione di circuiti digitali è il **transistore** il cui funzionamento può essere modellato (in modo molto semplice) come il funzionamento di un interruttore (*aperto* o *chiuso*), quindi con due stati fisici, cui corrispondono 2 opportune tensioni (in genere 0V e 5V)..

BIT (*binary digit*) = cifra binaria. (unità di informazione elementare)
Un bit può assumere due valori che possono essere associati ai simboli:

L(ow)	**H(igh)**	*aspetto fisico del segnale*
0	**1**	*aspetto aritmetico*
false	**true**	*aspetto logico*

Terminologia e «unita' di misura»

1 cifra	= bit	1 Kilo	$= 2^{10} = 1024$	$\cong 10^3$
8 bit	= byte	1 Mega	$= 2^{20} = 1048576$	$\cong 10^6$
16 bit	= word (parola)	1 Giga	$= 2^{30}$	$\cong 10^9$
32 bit	= double word			
64 bit	= quad word			

Elementi base della tecnologia elettronica

Caratteristiche principali della tecnologia elettronica che utilizza segnali digitali binari:

- gli *elementi base* (realizzati utilizzando uno o più transistori opportunamente collegati) sono di pochi tipi e relativamente semplici, e sono dotati di ingressi e uscite:

  $\Rightarrow$ **porte logiche** (*gate*) che realizzano gli operatori e consentono le elaborazioni

  $\Rightarrow$ **elemento di memoria** (*flip-flop* o *bistabile*) che consente il mantenimento di una singola informazione binaria

Gli elementi complessi si ottengono con una «costruzione» incrementale e ripetitiva degli elementi base, cioè aggregando anche numerosi elementi base con opportune interconnessioni. Le interconnessioni consentono la propagazione dei segnali, e quindi delle informazioni associate, dall'uscita di un elemento all'ingresso di uno o più altri elementi.

La tecnologia e il processo costruttivo dei **circuiti integrati** consentono di realizzare circuiti molto complessi in poco spazio e con un buon rapporto costi/prestazioni

Rappresentazione binaria

Nella rappresentazione binaria l'**alfabeto** (*l'insieme dei simboli utilizzabili*) è costituito dalle **cifre 0** e **1**.

Un'informazione è rappresentabile da una **sequenza** di cifre.

Quante sono le informazioni **distinte** rappresentabili?

- se sono disponibili **N** cifre binarie si possono avere $\mathbf{2^N}$ configurazioni diverse e quindi rappresentare al più $\mathbf{2^N}$ informazioni distinte

- se si devono rappresentare **M** informazioni distinte sono necessarie $\mathbf{N} = \lceil \mathbf{\log_2 M} \rceil$ cifre binarie

Quindi a seconda della cardinalità (M) dell'insieme di valori dell'informazione da rappresentare con una certa variabile, quest'ultima dovrà essere basata su un opportuno numero (N) di bit come indicato dalla relazione precedente.

Codifica dell’informazione

Una codifica è un **insieme di regole** per **costruire** e **interpretare** la sequenza di cifre binarie che rappresenta l’informazione di un dato tipo (caratteri, numeri interi, ecc.)

Codifica di informazioni di un dato tipo

definizione

corrispondenza biunivoca tra ***rappresentazione*** dell’informazione e ***significato*** dell’informazione

codifica

RAPPRESENTAZIONE $\Leftrightarrow$ SIGNIFICATO

Corrispondenza:

è definita in modo arbitrario (è una *convenzione*) ma deve essere nota e sempre rispettata da chi genera e da chi utilizza le informazioni. Vengono in genere definiti degli ***standard***.

Arbitrarietà:

è utile per avere delle *proprietà* particolari sulla rappresentazione dell’informazione.

Le proprietà desiderabili possono dipendere dall’uso che verrà fatto delle informazioni.

Codifica dell’informazione

Le informazioni che consideriamo devono essere rappresentate e elaborate da una macchina (calcolatore elettronico).

Aspetto fondamentale:

> il **numero di elementi «fisici»** (elementi di memoria, collegamenti ....) disponibile per contenere la rappresentazione di ogni informazione è **finito**.

Poichè ogni elemento fisico «contiene» il valore di una cifra binaria, in ogni componente di un calcolatore il **numero di cifre binarie** disponibili per rappresentare l’informazione è **finito**. Quindi il numero di informazioni distinte rappresentabili è finito.

Nasce quindi il concetto di **non rappresentabilità** di informazioni che richiedono un numero di cifre maggiore di quelle disponibili.

Considerazione sulla codifica dell'informazione

Siamo abituati a considerare la disponibilità di un numero «illimitato» (o comunque sufficiente) di elementi per rappresentare le informazioni.

Esempio 1

Numeri decimali e operazioni aritmetiche: siamo abituati ad usare tutte le cifre necessarie senza particolari limiti.

Esempio 2

Informazioni da rappresentare: parole della lingua italiana

- alfabeto: 21 lettere
- lunghezza delle parole non limitata (si può ipotizzare un limite ragionevole Hp: L_{max}=26)
- alcune sequenze di lettere non hanno significato

Vocabolario italiano

- numero di parole esistenti è $<< 21^{26}$
- non sono esaurite tutte le sequenze possibili (configurazioni) di lettere
- l'introduzione di nuove parole non richiede di aumentare la lunghezza e/o di aumentare il numero di simboli

Classi (tipi) di informazioni da rappresentare

1. Informazioni enumerative

Caratteristiche:

- numerabili
- non numeriche
- l'ordine di enumerazione è significativo: può denotare delle proprietà tra le informazioni e consentire delle operazioni tra le informazioni

2. Valori numerici

Caratteristiche:

- devono consentire di rappresentare in modo adeguato gli insiemi della matematica (naturali, interi, razionali, reali)
- sono dei **sottoinsiemi** di alcuni degli insiemi della matematica del punto precedente
- devono essere possibili tutte le operazioni della matematica (e, almeno quelle fondamentali, devono essere facili da realizzare con dei circuiti)

Informazioni enumerative

Esempi:

1. Colori dell'arcobaleno: 7 colori $\Rightarrow$ 3 bit e quindi 8 possibili configurazioni distinte

- scelta della corrispondenza (arbitraria, ma si può preservare la posizione nell'arcobaleno, cioè l'ordine per frequenze della luce crescenti)
- la configurazione libera disponibile viene usata per rappresentare il «non colore» (nero)

significato	***codifica***
nero	000
rosso	001
arancio	010
giallo	011
verde	100
azzurro	101
indaco	110
violetto	111

Questa tabella riporta una possibile (non standard) codifica binaria dei 7 colori.

2. Giorni della settimana: lu ma me gio ve sa do

3. Mesi dell'anno:.......

Informazioni enumerative: caratteri alfanumerici

I caratteri alfanumerici consentono all'uomo di rappresentare tutte le informazioni.

Si tratta quindi di una codifica **molto importante ed utilizzata**.

Si devono rappresentare:

- lettere maiuscole/minuscole A a .. Z z
- spazio
- cifre 0 9
- segni di interpunzione , : ; .
- simboli ! « # % @) < =
- caratteri di controllo per gestire la visualizzazione, la stampa, la trasmissione dei caratteri (*inizio riga, salto di riga, salto pagina* ....)

La rappresentazione dei caratteri alfanumerici fa uso di una codifica **standard** universalmente accettata: **codifica ASCII** (American Standard Code for Information Interchange)

Codifica ASCII: caratteristiche

- 7 bit per rappresentare ogni carattere ⇒ 128 caratteri alfanumerici distinti: le possibili configurazioni vanno da 0000000 a 1111111

- la codifica è stata scelta in modo da rispettare alcune «proprietà » dei caratteri:
  - ⇒ ordinamento delle cifre
  - ⇒ ordinamento delle lettere

- introduce le seguenti ulteriori proprietà :
  - ⇒ le lettere maiuscole precedono tutte le lettere minuscole
  - ⇒ la «distanza» tra una lettera maiuscola e la sua corrispondente minuscola è la stessa per tutte le lettere

ASCII esteso (8 bit) 256 configurazioni:

le prime 128 (da 00000000 a 01111111) sono associate ai caratteri dell’ASCII Standard, le rimanenti 128 (da 10000000 a 11111111) sono associate a lettere accentate..., a caratteri semigrafici ...

Tabella ASCII

bit meno significativi

bit più significativi

	000	001	010	011	100	101	110	111
0000	NUL	DLE	SP 32	0 48	@ 64	P 80	` 96	p 112
0001	SOH	DC1	! 33	1 49	A 65	Q 81	a 97	q 113
0010	STX	DC2	« 34	2 50	B 66	R 82	b 98	r 114
0011	ETX	DC3	# 35	3 51	C 67	S 83	c 99	s 115
0100	EOT	DC4	$ 36	4 52	D 68	T 84	d 100	t 116
0101	ENQ	NAK	% 37	5 53	E 69	U 85	e 101	u 117
0110	ACK	SYN	& 38	6 54	F 70	V 86	f 102	v 118
0111	BEL	ETB	’ 39	7 55	G 71	W 87	g 103	w 119
1000	BS	CAN	(40	8 56	H 72	X 88	h 104	x 120
1001	HT	EM	) 41	9 57	I 73	Y 89	i 105	y 121
1010	LF	SUB	* 42	: 58	J 74	Z 90	j 106	z 122
1011	VT	ESC	+ 43	; 59	K 75	[91	k 107	{ 123
1100	FF	FS	, 44	< 60	L 76	\ 92	l 108	| 124
1101	CR	GS	– 45	= 61	M 77	] 93	m 109	} 125
1110	SO	RS	. 46	> 62	N 78	^ 94	n 110	~ 126
1111	SI	US	/ 47	? 63	O 79	– 95	o 111	DEL

N.B.

I valori numerici in piccolo sono il valore decimale corrispondente al codice ASCII

La codifica ASCII di un carattere nella tabella è ottenuta prendendo i 3 bit corrispondenti alla colonna, seguiti dai 4 bit corrispondenti alla riga (questi gruppi di bit sono anche facilmente associabili a cifre esadecimali - v.seguito).

Codice ASCII

Byte	Cod.	Char	Byte	Cod.	Char	Byte	Cod.	Char	Byte	Cod.	Char
00000000	0	Null	00100000	32	Spc	01000000	64	@	01100000	96	`
00000001	1	Start of heading	00100001	33	!	01000001	65	A	01100001	97	a
00000010	2	Start of text	00100010	34	”	01000010	66	B	01100010	98	b
00000011	3	End of text	00100011	35	#	01000011	67	C	01100011	99	c
00000100	4	End of transmit	00100100	36	$	01000100	68	D	01100100	100	d
00000101	5	Enquiry	00100101	37	%	01000101	69	E	01100101	101	e
00000110	6	Acknowledge	00100110	38	&	01000110	70	F	01100110	102	f
00000111	7	Audible bell	00100111	39	’	01000111	71	G	01100111	103	g
00001000	8	Backspace	00101000	40	(	01001000	72	H	01101000	104	h
00001001	9	Horizontal tab	00101001	41	)	01001001	73	I	01101001	105	i
00001010	10	Line feed	00101010	42	*	01001010	74	J	01101010	106	j
00001011	11	Vertical tab	00101011	43	+	01001011	75	K	01101011	107	k
00001100	12	Form Feed	00101100	44	,	01001100	76	L	01101100	108	l
00001101	13	Carriage return	00101101	45	-	01001101	77	M	01101101	109	m
00001110	14	Shift out	00101110	46	.	01001110	78	N	01101110	110	n
00001111	15	Shift in	00101111	47	/	01001111	79	O	01101111	111	o
00010000	16	Data link escape	00110000	48	0	01010000	80	P	01110000	112	p
00010001	17	Device control 1	00110001	49	1	01010001	81	Q	01110001	113	q
00010010	18	Device control 2	00110010	50	2	01010010	82	R	01110010	114	r
00010011	19	Device control 3	00110011	51	3	01010011	83	S	01110011	115	s
00010100	20	Device control 4	00110100	52	4	01010100	84	T	01110100	116	t
00010101	21	Neg. acknowledge	00110101	53	5	01010101	85	U	01110101	117	u
00010110	22	Synchronous idle	00110110	54	6	01010110	86	V	01110110	118	v
00010111	23	End trans. block	00110111	55	7	01010111	87	W	01110111	119	w
00011000	24	Cancel	00111000	56	8	01011000	88	X	01111000	120	x
00011001	25	End of medium	00111001	57	9	01011001	89	Y	01111001	121	y
00011010	26	Substitution	00111010	58	:	01011010	90	Z	01111010	122	z
00011011	27	Escape	00111011	59	;	01011011	91	[	01111011	123	{
00011100	28	File separator	00111100	60	<	01011100	92	\	01111100	124	|
00011101	29	Group separator	00111101	61	=	01011101	93	]	01111101	125	}
00011110	30	Record Separator	00111110	62	>	01011110	94	^	01111110	126	~
00011111	31	Unit separator	00111111	63	?	01011111	95	_	01111111	127	Del

Codice ASCII esteso

Byte	Cod.	Char	Byte	Cod.	Char	Byte	Cod.	Char	Byte	Cod.	Char
10000000	128	Ç	10100000	160	á	11000000	192	+	11100000	224	Ó
10000001	129	ü	10100001	161	í	11000001	193	-	11100001	225	ß
10000010	130	é	10100010	162	ó	11000010	194	-	11100010	226	Ô
10000011	131	â	10100011	163	ú	11000011	195	+	11100011	227	Ò
10000100	132	ä	10100100	164	ñ	11000100	196	-	11100100	228	õ
10000101	133	à	10100101	165	Ñ	11000101	197	+	11100101	229	Õ
10000110	134	å	10100110	166	ª	11000110	198	ã	11100110	230	µ
10000111	135	ç	10100111	167	º	11000111	199	Ã	11100111	231	þ
10001000	136	ê	10101000	168	¿	11001000	200	+	11101000	232	Þ
10001001	137	ë	10101001	169	®	11001001	201	+	11101001	233	Ú
10001010	138	è	10101010	170	¬	11001010	202	-	11101010	234	Û
10001011	139	ï	10101011	171	½	11001011	203	-	11101011	235	Ù
10001100	140	î	10101100	172	¼	11001100	204	¦	11101100	236	ý
10001101	141	ì	10101101	173	¡	11001101	205	-	11101101	237	Ý
10001110	142	Ä	10101110	174	«	11001110	206	+	11101110	238	¯
10001111	143	Å	10101111	175	»	11001111	207	¤	11101111	239	´
10010000	144	É	10110000	176	–	11010000	208	ð	11110000	240	-
10010001	145	æ	10110001	177	–	11010001	209	Ð	11110001	241	±
10010010	146	Æ	10110010	178	–	11010010	210	Ê	11110010	242	‗
10010011	147	ô	10110011	179	¦	11010011	211	Ë	11110011	243	¾
10010100	148	ö	10110100	180	¦	11010100	212	È	11110100	244	¶
10010101	149	ò	10110101	181	Á	11010101	213	i	11110101	245	§
10010110	150	û	10110110	182	Â	11010110	214	Í	11110110	246	÷
10010111	151	ù	10110111	183	À	11010111	215	Î	11110111	247	¸
10011000	152	ÿ	10111000	184	©	11011000	216	Ï	11111000	248	°
10011001	153	Ö	10111001	185	¦	11011001	217	+	11111001	249	¨
10011010	154	Ü	10111010	186	¦	11011010	218	+	11111010	250	·
10011011	155	ø	10111011	187	+	11011011	219	_	11111011	251	¹
10011100	156	£	10111100	188	+	11011100	220	_	11111100	252	³
10011101	157	Ø	10111101	189	¢	11011101	221	¦	11111101	253	²
10011110	158	×	10111110	190	¥	11011110	222	Ì	11111110	254	_
10011111	159	ƒ	10111111	191	+	11011111	223	_	11111111	255	

ASCII Completo

0		32		64	@	96	`	128	Ç	160	á	192	└	224	Ó
1	☺	33	!	65	A	97	a	129	ü	161	í	193	┴	225	ß
2	☻	34	"	66	B	98	b	130	é	162	ó	194	┬	226	Ô
3	♥	35	#	67	C	99	c	131	â	163	ú	195	├	227	Ò
4	♦	36	$	68	D	100	d	132	ä	164	ñ	196	─	228	õ
5	♣	37	%	69	E	101	e	133	à	165	Ñ	197	┼	229	Õ
6	♠	38	&	70	F	102	f	134	å	166	ª	198	ã	230	µ
7	•	39	'	71	G	103	g	135	ç	167	º	199	Ã	231	þ
8	◘	40	(	72	H	104	h	136	ê	168	¿	200	╚	232	Þ
9	○	41	)	73	I	105	i	137	ë	169	®	201	╔	233	Ú
10	◙	42	*	74	J	106	j	138	è	170	¬	202	╩	234	Û
11	♂	43	+	75	K	107	k	139	ï	171	½	203	╦	235	Ù
12	♀	44	,	76	L	108	l	140	î	172	¼	204	╠	236	ý
13	♪	45	-	77	M	109	m	141	ì	173	¡	205	═	237	Ý
14	♫	46	.	78	N	110	n	142	Ä	174	«	206	╬	238	¯
15	☼	47	/	79	O	111	o	143	Å	175	»	207	¤	239	´
16	►	48	0	80	P	112	p	144	É	176	░	208	ð	240	-
17	◄	49	1	81	Q	113	q	145	æ	177	▒	209	Ð	241	±
18	↕	50	2	82	R	114	r	146	Æ	178	▓	210	Ê	242	‗
19	‼	51	3	83	S	115	s	147	ô	179	│	211	Ë	243	¾
20	¶	52	4	84	T	116	t	148	ö	180	┤	212	È	244	¶
21	§	53	5	85	U	117	u	149	ò	181	Á	213	ı	245	§
22	▬	54	6	86	V	118	v	150	û	182	Â	214	Í	246	÷
23	↨	55	7	87	W	119	w	151	ù	183	À	215	Î	247	¸
24	↑	56	8	88	X	120	x	152	ÿ	184	©	216	Ï	248	°
25	↓	57	9	89	Y	121	y	153	Ö	185	╣	217	┘	249	¨
26	→	58	:	90	Z	122	z	154	Ü	186	║	218	┌	250	·
27	←	59	;	91	[	123	{	155	ø	187	╗	219	█	251	¹
28	∟	60	<	92	\	124	|	156	£	188	╝	220	▄	252	³
29	↔	61	=	93	]	125	}	157	Ø	189	¢	221	¦	253	²
30	▲	62	>	94	^	126	~	158	×	190	¥	222	Ì	254	■
31	▼	63	?	95	_	127	⌂	159	ƒ	191	┐	223	▀	255	

Codice ASCII

ALT + 033 = !	ALT + 034 = "	ALT + 035 = #	ALT + 036 = $
ALT + 037 = %	ALT + 038 = &	ALT + 039 = '	ALT + 040 = (
ALT + 041 =)	ALT + 042 = *	ALT + 043 = +	ALT + 044 = ,
ALT + 045 = -	ALT + 046 = .	ALT + 047 = /	ALT + 048 = 0
ALT + 049 = 1	ALT + 050 = 2	ALT + 051 = 3	ALT + 052 = 4
ALT + 053 = 5	ALT + 054 = 6	ALT + 055 = 7	ALT + 056 = 8
ALT + 057 = 9	ALT + 058 = :	ALT + 059 = ;	ALT + 060 = <
ALT + 061 = =	ALT + 062 = >	ALT + 063 = ?	ALT + 064 = @
ALT + 065 = A	ALT + 066 = B	ALT + 067 = C	ALT + 068 = D
ALT + 069 = E	ALT + 070 = F	ALT + 071 = G	ALT + 072 = H
ALT + 073 = I	ALT + 074 = J	ALT + 075 = K	ALT + 076 = L
ALT + 077 = M	ALT + 078 = N	ALT + 079 = O	ALT + 080 = P
ALT + 081 = Q	ALT + 082 = R	ALT + 083 = S	ALT + 084 = T
ALT + 085 = U	ALT + 086 = V	ALT + 087 = W	ALT + 088 = X
ALT + 089 = Y	ALT + 090 = Z	ALT + 091 = [	ALT + 092 = \
ALT + 093 =]	ALT + 094 = ^	ALT + 095 = _	ALT + 096 = `
ALT + 097 = a	ALT + 098 = b	ALT + 099 = c	ALT + 100 = d
ALT + 101 = e	ALT + 102 = f	ALT + 103 = g	ALT + 104 = h
ALT + 105 = i	ALT + 106 = j	ALT + 107 = k	ALT + 108 = l
ALT + 109 = m	ALT + 110 = n	ALT + 111 = o	ALT + 112 = p
ALT + 113 = q	ALT + 114 = r	ALT + 115 = s	ALT + 116 = t
ALT + 117 = u	ALT + 118 = v	ALT + 119 = w	ALT + 120 = x
ALT + 121 = y	ALT + 122 = z	ALT + 123 = {	ALT + 124 = |
ALT + 125 = }	ALT + 126 = ~	ALT + 128 = Ç	ALT + 129 = ü
ALT + 130 = é	ALT + 131 = â	ALT + 132 = ä	ALT + 133 = à
ALT + 134 = å	ALT + 135 = ç	ALT + 136 = ê	ALT + 137 = ë
ALT + 138 = è	ALT + 139 = ï	ALT + 140 = î	ALT + 141 = ì

ALT + 146 = Æ	ALT + 147 = ô	ALT + 148 = ö	ALT + 149 = ò
ALT + 150 = û	ALT + 151 = ù	ALT + 152 = ÿ	ALT + 153 = Ö
ALT + 154 = Ü	ALT + 155 = ø	ALT + 156 = £	ALT + 157 = Ø
ALT + 158 = ×	ALT + 159 = ƒ	ALT + 160 = á	ALT + 161 = í
ALT + 162 = ó	ALT + 163 = ú	ALT + 164 = ñ	ALT + 165 = Ñ
ALT + 166 = ª	ALT + 167 = º	ALT + 168 = ¿	ALT + 169 = ®
ALT + 170 = ¬	ALT + 171 = ½	ALT + 172 = ¼	ALT + 173 = ¡
ALT + 174 = «	ALT + 175 = »	ALT + 176 = _	ALT + 177 = _
ALT + 178 = _	ALT + 179 = ¦	ALT + 180 = ¦	ALT + 181 = Á
ALT + 182 = Â	ALT + 183 = À	ALT + 184 = ©	ALT + 185 = ¦
ALT + 186 = ¦	ALT + 187 = +	ALT + 188 = +	ALT + 189 = ¢
ALT + 190 = ¥	ALT + 191 = +	ALT + 192 = +	ALT + 193 = -
ALT + 194 = -	ALT + 195 = +	ALT + 196 = -	ALT + 197 = +
ALT + 198 = ã	ALT + 199 = Ã	ALT + 200 = +	ALT + 201 = +
ALT + 202 = -	ALT + 203 = -	ALT + 204 = ¦	ALT + 205 = -
ALT + 206 = +	ALT + 207 = ¤	ALT + 208 = ð	ALT + 209 = Ð
ALT + 210 = Ê	ALT + 211 = Ë	ALT + 212 = È	ALT + 213 = i
ALT + 214 = Í	ALT + 215 = Î	ALT + 216 = Ï	ALT + 217 = +
ALT + 218 = +	ALT + 219 = _	ALT + 220 = _	ALT + 221 = ¦
ALT + 222 = Ì	ALT + 223 = _	ALT + 224 = Ó	ALT + 225 = ß
ALT + 226 = Ô	ALT + 227 = Ò	ALT + 228 = õ	ALT + 229 = Õ
ALT + 230 = µ	ALT + 231 = þ	ALT + 232 = Þ	ALT + 233 = Ú
ALT + 234 = Û	ALT + 235 = Ù	ALT + 236 = ý	ALT + 237 = Ý
ALT + 238 = ¯	ALT + 239 = ´	ALT + 240 = -	ALT + 241 = ±
ALT + 242 = _	ALT + 243 = ¾	ALT + 244 = ¶	ALT + 245 = §
ALT + 246 = ÷	ALT + 247 = ¸	ALT + 248 = °	ALT + 249 = ¨
ALT + 250 = ·	ALT + 251 = ¹	ALT + 252 = ³	ALT + 253 = ²
ALT + 254 = _	ALT + 255 = spazio	ALT + 265 = tab	ALT + 266 = invio
[illegible]	[illegible]	[illegible]	[illegible]

ALT + 291 = #	ALT + 292 = $	ALT + 293 = %	ALT + 294 = &
ALT + 295 = '	ALT + 296 = (	ALT + 297 =)	ALT + 298 = *
ALT + 299 = +	ALT + 300 = ,	ALT + 301 = -	ALT + 302 = .
ALT + 303 = /	ALT + 304 = 0	ALT + 305 = 1	ALT + 306 = 2
ALT + 307 = 3	ALT + 308 = 4	ALT + 309 = 5	ALT + 310 = 6
ALT + 311 = 7	ALT + 312 = 8	ALT + 313 = 9	ALT + 314 = :
ALT + 315 = ;	ALT + 316 = <	ALT + 317 = =	ALT + 318 = >

Rappresentazione di valori numerici

Le classi di valori numerici che si possono rappresentare sono le seguenti:

- Numeri naturali
- Numeri interi (relativi)
- Numeri razionali

Rappresentazione di numeri naturali: codifica pesata (binario naturale)

La codifica si basa sulla notazione posizionale o «pesata» che adottiamo usualmente per i numeri codificati in decimale.

Terminologia:

- Base: B (B=2, B=10, B=3, B=16)
- Valori delle cifre: 0 .. B-1
- Notazione posizionale: B^k (2^k, 10^k)

Nella notazione posizionale, in qualunque base, il valore numerico rappresentato da una cifra dipende

⇒ dal valore della cifra

⇒ dalla posizione della cifra nel numero

Esempio in base 10

$$P_{10} = 121_{10} = \mathbf{1}\cdot 10^2 + \mathbf{2}\cdot 10^1 + \mathbf{1}\cdot 10^0$$

Data una base **B** e un numero di cifre disponibili **N**

⇒ i numeri naturali (interi ≥ 0) rappresentabili sono

$$\mathbf{0 \le P \le B^N - 1}$$

Codifica pesata dei numeri naturali in base 2

$P_2 = b_{N-1}\ 2^{N-1} + b_{N-2}\ 2^{N-2} + ... + b_2\ 2^2 + b_1\ 2^1 + b_0\ 2^0$

bit piu' significativo: b_{N-1}
bit meno significativo: b_0

Ad esempio:

numero di bit disponibili N=3
i valori numerici rappresentabili sono

$0 \leq P \leq 2^3 - 1$ cioè $0 \leq P \leq 7$

decimale	*binario*	*valore*
0	000	$0 \cdot 2^2 + 0 \cdot 2^1 + 0 \cdot 2^0$
1	001	$0 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0$
2	010	$0 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0$
3	011	$0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0$
4	100	$1 \cdot 2^2 + 0 \cdot 2^1 + 0 \cdot 2^0$
5	101	$1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0$
6	110	$1 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0$
7	111	$1 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0$

Codifica pesata dei numeri naturali in base 2

numero di bit disponibili N=5 $\Rightarrow$ i valori numerici P rappresentabili sono $0 \leq P \leq 2^5-1$ cioè $0 \leq P \leq 31$

decimale	*binario*	*valore*
0	00000	$0\cdot 2^4+0\cdot 2^3+0\cdot 2^2+0\cdot 2^1+0\cdot 2^0$
1	00001	$0\cdot 2^4+0\cdot 2^3+0\cdot 2^2+0\cdot 2^1+1\cdot 2^0$
2	00010	$0\cdot 2^4+0\cdot 2^3+0\cdot 2^2+1\cdot 2^1+0\cdot 2^0$
.....	......	
7	00111	$0\cdot 2^4+0\cdot 2^3+1\cdot 2^2+1\cdot 2^1+1\cdot 2^0$
8	01000	$0\cdot 2^4+1\cdot 2^3+0\cdot 2^2+0\cdot 2^1+0\cdot 2^0$
.....	......	
15	01111	$0\cdot 2^4+1\cdot 2^3+1\cdot 2^2+1\cdot 2^1+1\cdot 2^0$
16	10000	$1\cdot 2^4+0\cdot 2^3+0\cdot 2^2+0\cdot 2^1+0\cdot 2^0$
.....	......	
30	11110	$1\cdot 2^4+1\cdot 2^3+1\cdot 2^2+1\cdot 2^1+0\cdot 2^0$
31	11111	$1\cdot 2^4+1\cdot 2^3+1\cdot 2^2+1\cdot 2^1+1\cdot 2^0$

gli zeri davanti «non contano» ma ci devono essere.

Siamo abituati a non scrivere gli zeri non significativi, lasciandoli impliciti.
Invece i segnali fisici possono solo indicare i valori 0 o 1 per tutti i bit usati per rappresentare un valore numerico, e non possono «sparire».
Per una certa informazione si adotta un numero fisso di bit, che deve essere stabilito a priori, e non una quantità che dipende dal valore di volta in volta rappresentato.

N=6 $\Rightarrow$ 0 - 63
N=7 $\Rightarrow$ 0 - 127
N=8 $\Rightarrow$ 0 - 255
N=10 $\Rightarrow$ 0 - 1023 (1 K)
N=16 $\Rightarrow$ 0 - $2^6 \cdot 2^{10}$ = 0 - 65535 $\cong$ 0 - 64.000 (64 K)
N=32 $\Rightarrow$ 0 - $2^2 \cdot 2^{10} \cdot 2^{10} \cdot 2^{10}$ $\cong$ 0 - 4.000.000.000 (4 G)

Codifica esadecimale (HEX)

E’ la codifica *pesata* dei numeri naturali in base B=16, sono quindi disponibili 16 valori delle cifre.

valore	cifra hex	binario
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
10	A	1010
11	B	1011
12	C	1100
13	D	1101
14	E	1110
15	F	1111

La comodità della rappresentazione HEX sta nella facilità di conversione in binario e viceversa:
ogni gruppo di 4 bit corrisponde direttamente ad una cifra HEX, come nella tabella accanto.

ESEMPIO

binario	0110	1001	1101	0100
HEX	6	9	D	4

Generalmente si premette uno «0» iniziale e una «h» finale.

Il generico valore in esadecimale si ottiene dalla relazione (dove N rappresenta il numero di cifre esadecimali):

$$P_{16}=b_{N-1}\,16^{N-1}+b_{N-2}\,16^{N-2}+......+b_2\,16^2+b_1\,16^1+b_0\,16^0$$

Nella documentazione tecnica di calcolatori ed in generale di circuiti logici occorre spesso citare delle configurazioni binarie (codifiche, indirizzi, ecc.).
Sequenze di 16 o 32 bit rappresentati con «1» e «0» sono, per l’uomo, scomode da gestire e difficili da ricordare.
Si ricorre quindi generalmente alla rappresentazione esadecimale (HEX). In particolare la forma HEX è usata nel linguaggio Assembler.

Altre codifiche di valori numerici

Per la rappresentazione dei numeri naturali esistono altre codifiche che non sfruttano la proprietà della notazione posizionale (utile per l'aritmetica) ma che **privilegiano altri aspetti** e proprietà delle sequenze di 0 e 1 che rappresentano il valore numerico.

Codifica Gray
(detta anche codifica a distanza di Hamming = 1)

Questa codifica è caratterizzata dalla seguente proprietà:

dato un valore numerico da rappresentare

⇓

la sua rappresentazione è diversa per il valore di una sola cifra (distanza di Hamming = 1) dalla rappresentazione del valore numerico precedente e di una sola cifra (distanza di Hamming = 1) dalla rappresentazione del valore numerico successivo

Decimale	*Codifica Gray*
0	000
1	001
2	011
3	010
4	110
5	111
6	101
7	100

Esempio di codifica Gray con 3 bit

Utile per rappresentare senza incertezze con segnali fisici dei valori numerici che nel tempo si incrementano e decrementano, evitando di richiedere le commutazioni di più di un bit alla volta, che potrebbero verificarsi non contemporaneamente.
Questa codifica è usata in applicazioni di controllo industriale.

Codifica BCD (Binary Coded Decimal)

Questa codifica rappresenta ciascuna **cifra decimale** con il valore numerico binario corrispondente.

Poichè le cifre decimali vanno da 0 a 9, il valore numerico massimo da rappresentare in binario è 9: sono quindi necessari 4 bit.
Le 6 configurazioni rimanenti non sono utilizzate.

Cifra decimale	*Codifica BCD*
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

Ad esempio: il valore decimale 1429
è codificato come 0001 0100 0010 1001
1 4 2 9

Questa codifica ha la proprietà di rendere facile la conversione con i numeri rappresentati in base dieci.

Conversione della rappresentazione pesata

Conversione =

passaggio dalla rappresentazione pesata di un numero naturale da una base **b** ad una base **B**.

Le regole di conversione si basano:

- sull'uguaglianza dei valori numerici, cioè indipendentemente dalla base di rappresentazione il valore del numero è lo stesso

- sul concetto di notazione pesata

Consideriamo i casi di conversione da binario a decimale e da decimale a binario di numeri naturali

Conversione binario decimale (numeri naturali): è molto semplice

$$P_2 = 11110_2 = \mathbf{1}\cdot 2^4 + \mathbf{1}\cdot 2^3 + \mathbf{1}\cdot 2^2 + \mathbf{1}\cdot 2^1 + \mathbf{0}\cdot 2^0$$

$$= 16 + 8 + 4 + 2 + 0 = 30_{10}$$

Conversione decimale binario di numeri naturali

Algoritmo di conversione:

- si considera l'uguaglianza $\mathbf{P = Q \cdot b + R}$, dove

⇒ **P** è il numero decimale da convertire
⇒ **Q** è il risultato della divisione intera per la base **b** (=2)
⇒ **R** è il resto della divisione intera per la base **b** ($0 \leq R \leq b-1$)

- si eseguono ripetute divisioni per la base b = 2 operando in base B = 10.
  Le divisioni considerano come valore da dividere, di volta in volta, il Q ottenuto al passo precedente. Il procedimento termina quando il valore da dividere è pari a 0.

Il numero di iterazioni necessarie per arrivare al termine del procedimento determina il **numero minimo di bit** necessari per rappresentare quel particolare numero in binario.

Conversione decimale binario di numeri naturali

Esempio: 37 in base 2.

P	37	R_0	**1**
Q_0	18	R_1	**0**
Q_1	9	R_2	**1**
Q_2	4	R_3	**0**
Q_3	2	R_4	**0**
Q_4	1	R_5	**1**
Q_5	0		

$\mathbf{37_{10} = 100101_2}$

Valgono le seguenti uguaglianze:

$P=Q_0b+R_0 = (Q_1b+R_1)b+R_0 = Q_1b^2+R_1b^1+R_0b^0=.....$

Quindi R_0 R_1.... sono le cifre della rappresentazione binaria a partire dal bit meno significativo.

Nell'esempio, il numero minimo di bit è 6.
R_5 rappresenta il bit più significativo.
Continuando il procedimento si otterrebbero degli 0 in posizione più significativa.

Conversione decimale binario della parte frazionaria

Rappresentazione della parte frazionaria:

$$\mathbf{P = 0.p_{-1}p_{-2}p_{-3}}$$

Nella notazione posizionale per la parte frazionaria il valore del numero rappresentato è dato da:

$P_{10} = 0 + d_{-1}\, 10^{-1} + d_{-2}\, 10^{-2} + ...$ (d_{-1} ... cifre decimali)

$P_{2} = 0 + b_{-1}\, 2^{-1} + b_{-2}\, 2^{-2} + b_{-3}\, 2^{-3} + ...$ (b_{-1} ... cifre binarie)

La conversione da binario a decimale è come quella vista per i numeri naturali, ma con le potenze della base negative.

Per la conversione da decimale a binario l’algoritmo di conversione è «duale» rispetto a quello per la parte intera. In questo caso si opera per moltiplicazioni successive (per la base 2) della sola parte frazionaria. Le cifre (bit) ottenute sono quelle dei riporti sulle unità intere.

Conversione della parte frazionaria

Esempi e osservazioni sulla conversione:

Esempio 1

0.5	1.0	b_{-1}	1
0.0			

$\color{red}{0.5}_{10}$ = $\color{red}{0.1}_{2}$ $\Rightarrow$ il valore è **esattamente** rappresentabile in base 2 con un qualsiasi numero ($\geq$ 1) di bit per la parte frazionaria.

Esempio 2

0.25	0.5	b_{-1}	0
0.5	1.0	b_{-2}	1
0.0			

$\color{red}{0.25}_{10}$ = $\color{red}{0.01}_{2}$ $\Rightarrow$ il valore è **esattamente** rappresentabile in base 2 con un qualsiasi numero ($\geq$ 2) di bit per la parte frazionaria.

Conversione della parte frazionaria

Esempi e osservazioni sulla conversione:

Esempio 3

0.6875	**1**.375	b_{-1}	**1**
0.375	**0**.75	b_{-2}	**0**
0.75	**1**.5	b_{-3}	**1**
0.5	**1**.0	b_{-4}	**1**
0.0			

$0.6875_{10} = 0.1011_2 \Rightarrow$ è rappresentabile esattamente con 4 (o più) bit per la parte frazionaria.
Se si hanno a disposizione solo 3 bit, la conversione genera il valore approssimato 0.101_2 (=0.625_{10}).

Esempio 4

0.1	0.2	b_{-1}	0
0.2	0.4	b_{-2}	0
0.4	0.8	b_{-3}	0
0.8	1.6	b_{-4}	1
0.6	1.2	b_{-5}	1
0.2	0.4	b_{-6}	0
0.4	0.8	b_{-7}	0
0.8	1.6	b_{-8}	1
....	....	......	....

$0.1_{10} = 0.0011001..._2 \Rightarrow$ **non** è rappresentabile esattamente ed è periodico (numero illimitato di cifre).

E' molto importante notare che il valore frazionario decimale 0.1 molto frequente nella normale numerazione decimale, non è rappresentabile esattamente in forma binaria. Ciò comporta talvolta risultati approssimati in calcoli eseguiti in base 2, che invece darebbero un risultato esatto in base 10.

Operazioni aritmetiche tra naturali in notazione posizionale

- Le operazioni aritmetiche in base 2 seguono le stesse «regole» di quelle in base 10.

- E’ fondamentale ricordarsi del problema della rappresentabilità dell’informazione con un numero di bit predefinito.

  Ad esempio, nel caso di addizione di due interi rappresentati su N bit, il risultato della somma può richiedere N+1 bit (*overflow* = traboccamento, o superamento).

- I circuiti per eseguire le operazioni aritmetiche in notazione posizionale sono molto semplici. Notevole vantaggio della codifica posizionale.

Esempio: tabella della **somma** su un bit (come nel decimale esiste il **riporto** verso la cifra più significativa)

			b
	+	0	1
a	0	0	1
	1	1	0 (1)

=

a	b	a+b
0	0	0
0	1	1
1	0	1
1	1	0 (1)

Operazioni aritmetiche: esempi

Somma: N = 4 bit $\Longrightarrow$ valori rappresentabili da 0 a 15

	111	$\leftarrow$ *riporti*
7 +	0111 +	
3	0011	
10	1010	

	1 110	$\leftarrow$ *riporti*
7 +	0111 +	
10	1010	
17	(1)0001	non rappresentabile con 4 bit

Sottrazione: N = 4 bit $\Longrightarrow$ valori rappresentabili da 0 a 15

	110	$\leftarrow$ *prestiti*
12 -	1100 -	
6	0110	
6	0110	

	1 000	$\leftarrow$ *prestiti*
5 -	0101 -	
8	1000	
- 3	(1)1101	non rappresentabile

Rappresentazione di valori numerici interi relativi

In aritmetica decimale, per rappresentare i numeri *relativi* siamo abituati ad utilizzare la rappresentazione *in modulo e segno*.

Le operazioni aritmetiche (algebriche) di somma e sottrazione, così come siamo abituati ad eseguirle, lavorano in modulo e segno e implicano una serie di operazioni elementari per ottenere il risultato corretto:

- analisi dei segni degli operandi — ***esempio: 5 + (-7) Þ sottrazione***
- confronto tra i moduli degli operandi — ***7 > 5 Þ 7 è il minuendo***
- somma tra moduli (naturali) *oppure*
- sottrazione tra moduli (naturali) — ***7 - 5 = 2 (modulo del risultato)***
- determinazione del segno del risultato — ***risultato negativo Þ -2***

Rappresentazione dei numeri interi (relativi): Codifica in complemento a 2

All’interno del calcolatore si utilizza una diversa rappresentazione – **in complemento a 2** – dei numeri interi relativi che consente di trattare somme e sottrazioni algebriche in modo indifferenziato e quindi è realizzabile con circuiti aritmetici più semplici.

Rappresentazione in complemento a 2

Date N cifre binarie, sono disponibili 2^N configurazioni distinte: di queste
2^{N-1} vengono utilizzate per rappresentare valori ≥ 0 e
2^{N-1} vengono utilizzate per rappresentare valori < 0.

Ad esempio, se N = 4 sono disponibili 16 configurazioni distinte, con queste posso rappresentare

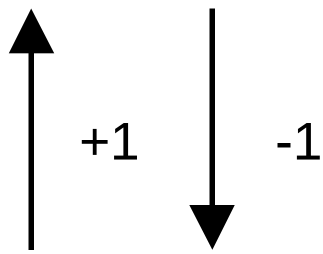
- valori ≥ 0: da 0 a $2^{N-1}-1$ cioe' da 0 a 7
- valori < 0: da -1 a -2^{N-1} cioe' da -1 a -8

N=6 $\Rightarrow$ da -32 a +31
N=7 $\Rightarrow$ da -64 a +63
N=8 $\Rightarrow$ da -128 a +127
N=10 $\Rightarrow$ da -512 a + 511
N=16 $\Rightarrow$ da -32.768 a +32.767
N=32 $\Rightarrow$ da $\cong$ - 2.000.000.000 a $\cong$ + 2.000.000.000

Rappresentazione in complemento a 2

Codifica su 4 bit

+7	0111
+6	0110
+5	0101
+4	0100
+3	0011
+2	0010
+1	0001
0	0000
-1	1111
-2	1110
-3	1101
-4	1100
-5	1011
-6	1010
-7	1001
-8	1000



Nota bene:

il valore del bit piu' significativo e' **indicativo** del segno:

$0 \Rightarrow$ valore ≥ 0

$1 \Rightarrow$ valore < 0

ma **non** è il segno: non basta cambiarlo per cambiare segno al numero

La rappresentazione in cpl 2 su N bit si ottiene:

$P \quad (\geq 0) \quad \Rightarrow \quad P$ in cpl2 $= \quad P_2$ su N bit

$-P \quad (<0) \quad \Rightarrow \quad -P$ in cpl2 $= \quad (2^N - P)_2$ su N bit

Rappresentazione in complemento a 2

Regola di conversione

Si scrive il valore assoluto del numero da rappresentare in notazione posizionale su N cifre

- se il numero da rappresentare è ≥ 0, questa è la rappresentazione in complemento a 2

- se il numero da rappresentare è < 0, si complementano tutti i bit e si somma 1

  In *alternativa*, partendo dal bit meno significativo, si lasciano inalterati i valori dei bit fino al primo 1 incluso e si complementano i rimanenti bit

Esempio su 4 bit.

P = - 6	6_{10} =	0110_2
	complemento i bit	1001
	sommo 1	1010

$$\mathbf{-\,6_{10} = 1010_{cpl2}}$$

Esempi di operazioni aritmetiche tra relativi in complemento a 2

	011
3 +	0011 +
-5	1011
-2	1110

	110
2 +	0010 +
7	0111
9	1001

1001 e’ negativo!
superamento

	1 111
- 1 +	1111 +
1	0001
0	(*1*)0000

Codifica di numeri razionali:
Rappresentazione in virgola mobile (floating point)

Il valore di un numero razionale R è esprimibile con la seguente forma generale per qualunque base **b**

$$R = M \cdot b^{E}$$

R - valore razionale

M - mantissa frazionaria con segno

b - base

E - esponente intero con segno

Quindi, data una base b, un numero rappresentato in virgola mobile può avere la forma

E	M

Dove M è la rappresentazione della mantissa nella base b e con un certo numero di cifre, ed E è la rappresentazione, anche questa nella base b, dell'esponente da dare alla base.

⇒ **Il campo di valori rappresentabili dipende da E**

⇒ **la risoluzione (precisione) dipende da M**

La rappresentazione si dice normalizzata se la mantissa ha un valore compreso tra l'inverso della base e l'unità, cioè

$$b^{-1} \leq M < b^{0}$$

Codifica di numeri razionali:
Rappresentazione in virgola mobile (floating point)

Base 10

Es. 1250.3 = $0.12503 \cdot 10^{4}$ normalizzato

4	**0.12503**

= $0.0012503 \cdot 10^{6}$ non normalizzato

Per sfruttare al meglio le cifre di mantissa si **normalizza**

Base b = 2

Es. 1011010 . 011 = 0 . 101101001100 • b^{0111}

0111	**101101001100**

In questo esempio, puramente indicativo, la mantissa è di 12 bit e l'esponente di 4 bit.

Codifica di numeri razionali: rappresentazione in virgola mobile

Standard IEEE floating point su 32 bit

1 bit di segno della mantissa
8 bit di esponente E
23 bit di mantissa M (pari a 7 cifre decimali)

S	E	M

$\Rightarrow$ Campo di valori rappresentabili
$\sim -10^{38}$ . . $\sim -10^{-38}$ 0 $\sim 10^{-38}$ . . $\sim 10^{38}$

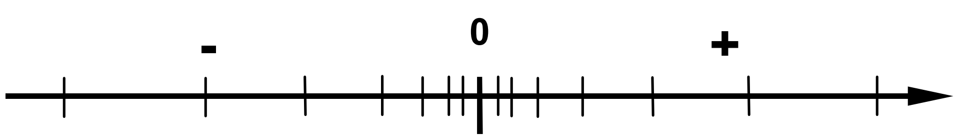
$\Rightarrow$ Risoluzione uno su 8 milioni

NB. I circa 4 miliardi di configurazioni dei 32 bit usati consentono di coprire un campo di valori molto ampio grazie alla distribuzione non uniforme.

Per i numeri piccoli i valori sono «fitti», ma si diradano per i grandi!

Approssimativamente gli intervalli tra **valori contigui** sono:

- per valori di 10000 l’intervallo è di un millesimo
- per valori di 10 milioni l’intervallo è di un’unità
- per valori di 10 miliardi l’intervallo è di mille
- ecc.



Ci sono anche standard IEEE su 64 e su 80 bit che estendono il campo di valori rappresentabili e la precisione.

The IEEE standard for floating point arithmetic

The IEEE (Institute of Electrical and Electronics Engineers) has produced a standard for floating point arithmetic. This standard specifies how single precision (32 bit) and double precision (64 bit) floating point numbers are to be represented, as well as how arithmetic should be carried out on them.

At the PSC, the Ultrix front-ends use the IEEE format. The VMS front-ends and Cray C90 systems have their own vendor-specific formats for data.

Because many of our users may have occasion to transfer unformatted or "binary" data between an IEEE machine and the Cray or the VAX/VMS, it is worth noting the details of this format for comparison with the Cray and VAX representations. The differences in the formats also affect the accuracy of floating point computations.

Summary:

Single Precision

The IEEE single precision floating point standard representation requires a 32 bit word, which may be represented as numbered from 0 to 31, left to right. The first bit is the sign bit, S, the next eight bits are the exponent bits, 'E', and the final 23 bits are the fraction 'F':

```
S EEEEEEEE FFFFFFFFFFFFFFFFFFFFFFF
0 1      8 9                    31
```

The value V represented by the word may be determined as follows:

- If E=255 and F is nonzero, then V=NaN ("Not a number")
- If E=255 and F is zero and S is 1, then V=-Infinity
- If E=255 and F is zero and S is 0, then V=Infinity
- If 0<E<255 then V=(-1)**S * 2 ** (E-127) * (1.F) where "1.F" is intended to represent the binary number created by prefixing F with an implicit leading 1 and a binary point.
- If E=0 and F is nonzero, then V=(-1)**S * 2 ** (-126) * (0.F) These are "unnormalized" values.
- If E=0 and F is zero and S is 1, then V=-0
- If E=0 and F is zero and S is 0, then V=0

In particular,

```
0 00000000 00000000000000000000000 = 0
1 00000000 00000000000000000000000 = -0

0 11111111 00000000000000000000000 = Infinity
1 11111111 00000000000000000000000 = -Infinity

0 11111111 00000100000000000000000 = NaN
1 11111111 00100010001001010101010 = NaN

0 10000000 00000000000000000000000 = +1 * 2**(128-127) * 1.0 = 2
0 10000001 10100000000000000000000 = +1 * 2**(129-127) * 1.101 = 6.5
1 10000001 10100000000000000000000 = -1 * 2**(129-127) * 1.101 = -6.5

0 00000001 00000000000000000000000 = +1 * 2**(1-127) * 1.0 = 2**(-126)
0 00000000 10000000000000000000000 = +1 * 2**(-126) * 0.1 = 2**(-127)
0 00000000 00000000000000000000001 = +1 * 2**(-126) *
                                     0.00000000000000000000001 =
                                     2**(-149)  (Smallest positive value)
```