

UNIVERSITÀ DI PISA
DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

INGEGNERIA INFORMATICA

Rappresentazione dell'informazione

Da Fondamenti di programmazione

A.A 2019-2020 - Primo semestre

File realizzato da Gabriele Frassi, disponibile presso la copisteria *OneCent*

Indice

1	Introduzione	2
1.1	Un accenno alle basi numeriche	2
2	Rappresentazione del testo	4
3	Rappresentazione dei num. naturali \mathbb{N}	6
3.1	Formula di sommatoria	6
3.2	Algoritmo DIV&MOD	7
3.3	Potenze di due, basi particolari	7
3.3.1	Base otto ($\beta = 8$)	8
3.3.2	Base sedici ($\beta = 16$)	8
3.4	Intervallo di rappresentabilità su p bit	10
3.5	Somme di numeri naturali espressi in binario	10
4	Rappresentazione dei num. interi \mathbb{Z}	11
4.1	Rappresentazione in modulo e segno	11
4.1.1	Da intero a rappresentazione	11
4.1.2	Da rappresentazione a intero	12
4.1.3	Tabella di conversione da -7 a $+7$ in $p = 4$	12
4.1.4	Intervallo di rappresentabilità in modulo e segno su p bit	13
4.2	Rappresentazione in complemento a 2	13
4.2.1	Da intero a rappresentazione	13
4.2.2	Da rappresentazione a intero	14
4.2.3	Tabella di conversione da -8 a $+7$ in $p = 4$	14
4.2.4	Intervallo di rappresentabilità in complemento a 2 su p bit	15
4.3	Rappresentazione con bias	15
4.3.1	Da intero a rappresentazione	15
4.3.2	Da rappresentazione a intero	16
4.3.3	Intervallo di rappresentabilità in bias	16
4.4	Somma di numeri interi	16
4.5	Tabella di confronto delle rappresentazioni	17
5	Rappresentazione dei num. reali \mathbb{R}	18
5.1	Virgola fissa (notazione ordinaria)	18
5.1.1	Procedura <i>parte frazionaria-parte intera</i>	18
5.2	Virgola mobile (notazione scientifica)	20
5.2.1	Notazione scientifica	20
5.2.2	Rappresentazione di r	20
5.2.3	Trasformazione $R \rightarrow r$	21
5.2.4	Trasformazione $r \rightarrow R$	22
5.2.5	Numeri con modulo massimo - $\pm\infty$ - intervallo di rappresentabilità	22
5.2.6	Numeri con modulo minimo - ± 0	23
6	Esercizi	24

Capitolo 1

Introduzione

“Al mondo ci sono 10 tipi di persone: quelle che capiscono la numerazione binaria e quelle che non la capiscono.”

- Marco Cococcioni durante le prime lezioni, *battuta molto diffusa*

Le informazioni sono qualcosa di astratto, manipolabile solo se rappresentato. In un calcolatore i vari tipi di informazioni si rappresentano mediante sequenze di bit (cifre binarie).

Il bit (abbreviazione di Binary Digit, ossia cifra binaria) è l'unità di misura elementare dell'informazione ma anche la base del sistema numerico utilizzato dai computer. Può assumere solo due valori: 0 oppure 1.

8 bit corrispondono ad un Byte, altra unità di misura dell'informazione.

È importante, dato un tipo di informazione in C++, conoscere il numero di byte a disposizione. La variabile `char`, per esempio, occupa soltanto 8bit: quindi può rappresentare soltanto una lettera!

Quante informazioni possono essere contenute in una sequenza formata da n bit? L'informazione consiste in tutte le possibili combinazioni di zeri ed uno ottenibili in n caselle. Il numero di combinazioni consiste in:

$$\text{numero combinazioni} = 2^n$$

Esempio: se io pongo $n = 2$ otterrò quattro combinazioni possibili, precisamente le seguenti:

00, 01, 10, 11

Si sottolinea che una sequenza binaria non è univoca: può avere significato diverso in base al contesto. Il numero 01000001 corrisponde all'intero 65, ma anche al carattere *A*!

1.1 Un accenno alle basi numeriche

I numeri sono espressi mediante configurazioni finite di simboli che specificano, da sinistra verso destra, l'eventuale simbolo e le cifre. In alcuni casi può essere presente un punto (nei calcolatori si utilizzano i punti, non le virgole!) per dividere la parte intera dalla parte decimale.

L'uomo, comunemente, utilizza la base dieci (0, 1, 2, 3, 4, 5, 6, 7, 8, 9); nei calcolatori, invece, si utilizza la base due (le sequenze binarie formate da zeri e uno che abbiamo già rammentato).

Nelle sezioni successive rappresenteremo i vari numeri (naturali, interi e reali) analizzando eventuali algoritmi per convertire un numero da una base a un'altra.

*Data una base $\beta \geq \text{due}$, ogni numero naturale N minore di β ($N < \beta$) è associato ad un simbolo elementare detto **cifra***

Esempi di basi con cifre utilizzate	
BASE	CIFRE
DUE	0, 1
CINQUE	0, 1, 2, 3, 4
OTTO	0, 1, 2, 3, 4, 5, 6, 7
SEDICI	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F

Osservazione sui numeri binari I numeri binari che presentano 1 come cifra finale sono dispari, quelli che presentano 0 sono pari!

Esempio 1: $(101)_2 = 5$

Esempio 2: $(101010)_2 = 42$

Consiglio importante L'insegnante ci ha consigliato caldamente di imparare a memoria almeno i primi otto elevamenti a potenza di due. Vi garantisco che fare ciò renderà più veloce la risoluzione di diversi esercizi!

$$\begin{array}{cccc} 2^1 = 2 & 2^2 = 4 & 2^3 = 8 & 2^4 = 16 \\ 2^5 = 32 & 2^6 = 64 & 2^7 = 128 & 2^8 = 256 \end{array}$$

Capitolo 2

Rappresentazione del testo

Ogni singolo carattere parte di un testo può essere codificato mediante un'opportuna sequenza di bit. Ne consegue che la rappresentazione binaria di un intero testo sia costituita dall'unione delle sequenze che codificano i singoli caratteri.

La codifica dei caratteri più nota è la **ASCII**¹ (*American Standard Code for Information Interchange*). Ogni carattere è codificato su 7bit (il primo bit del byte è sempre 0), pertanto il numero di caratteri possibili mediante questa codifica sarà $2^7 = 128$. Tra questi sono presenti caratteri speciali utilizzati nella trasmissione di dati tra dispositivi remoti.

Un calcolatore, ogni volta che noi premiamo un certo tasto, produce in uscita la codifica ASCII del corrispondente carattere. Stessa cosa avviene con le stampanti, che ricevono la codifica ASCII di ciascun carattere da stampare.

Attraverso la codifica ASCII esprimeremo:

1. I numeri da 0 a 9
2. Le lettere maiuscole dell'alfabeto
3. Le lettere minuscole dell'alfabeto
4. Simboli particolari come lo spazio, il punto interrogativo, il punto esclamativo...

Ogni calcolatore può avere un certo tipo di codifica. La ASCII, pur essendo la più nota, risulta limitante per le nostre necessità. Nel tempo sono state create nuove codifiche su un numero di bit più elevato: avendo 16bit, per esempio, potremo codificare $2^{16} = 65536$ caratteri!

Esempio di nuovi simboli: le emoticon sulle chat, anch'esse codificate mediante sequenze binarie. *Svengo pensando al numero di bit necessari per includere questi simboli (semi-cit).*

¹Si legge *ASCHI*

Codifica ASCII – Tabella di corrispondenza

3 cifre più significative

000	001	010	011	100	101	110	111	
NUL	DLE	SP	0	@	P	'	p	0000
SOH	XON	!	1	A	Q	a	q	0001
STX	DC2	"	2	B	R	b	r	0010
ETX	XOFF	#	3	C	S	c	s	0011
EQT	DC4	\$	4	D	T	d	t	0100
ENQ	NAK	%	5	E	U	e	u	0101
ACK	SYN	&	6	F	V	f	v	0110
BEL	ETB	'	7	G	W	g	w	0111
BS	CAN	(8	H	X	h	x	1000
HT	EM)	9	I	Y	i	y	1001
LF	SUB	*	:	J	Z	j	z	1010
VF	ESC	+	;	K	[k	{	1011
FF	FS	,	<	L	\	l		1100
CR	GS	-	=	M]	m	}	1101
SO	RS	.	>	N	^	n	~	1110
SI	US	/	?	O	_	o	DEL	1111

4 cifre meno significative

Codifica ASCII - Esempio: *Caro amico,*

Sequenze	Caratteri	
00110000	0	01000011 ←
00110001	1	01100001
00110010	2	01110010
...	...	01101111
00111001	9	00100000
...	...	01100001
01000001	A	01101110
01000010	B	01101001
01000011	C	01100011
...	...	01101111
01100001	a	00101100 ←
01100010	b	
...	...	

Caro amico,

Capitolo 3

Rappresentazione dei num. naturali

N

L'insieme dei numeri naturali, in senso informatico¹, consiste nel seguente insieme:

$$\mathbb{N} = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, \dots\}$$

Il compilatore, tenendo conto del numero di byte disponibili, non può rappresentare l'insieme completo, ma soltanto un suo sottoinsieme:

$$\overline{\mathbb{N}} = \{0, 1, \dots, max\}$$

Parleremo più avanti di **intervallo di rappresentabilità** individuando gli estremi di questo sottoinsieme.

3.1 Formula di sommatoria

I numeri arabi permettono la rappresentazione di numeri maggiori o uguali alla base con una sequenza di cifre secondo la rappresentazione posizionale.

Dato un numero N espresso in base β con $N \geq \beta$ espresso nel seguente modo:

$$a_{p-1}a_{p-2} \dots a_1a_0$$

Posso convertire questo numero formato da p cifre in base *dieci* mediante la **formula della sommatoria**. a_{p-1} è detta cifra più significativa, a_0 cifra meno significativa.

$$N = \sum_{i=0}^{p-1} a_i \beta^i = a_{p-1} \beta^{p-1} + a_{p-2} \beta^{p-2} + \dots + a_2 \beta^2 + a_1 \beta + a_0$$

Esempio 1 123 (La convenzione stabilisce che se non indico la base il num. è in base 10)

$$123 = 1 * 10^2 + 2 * 10^1 + 3 * 10^0 = 100 + 20 + 3$$

Esempio 2 $(102)_6$

$$(102)_6 = 1 * 6^2 + 0 * 6^1 + 2 * 6^0 = 36 + 0 + 2 = 38$$

Importantissimo il modo in cui viene letto un numero: non leggo *centodieci in base sei*, ma **uno zero due in base sei**.

¹Non dite al prof di **Analisi I** che zero fa parte dell'insieme dei numeri naturali.

Esempio 3 $(101)_2$

$$(101)_2 = 1 * 2^2 + 0 * 2^1 + 1 * 2^0 = 4 + 0 + 1 = 5$$

3.2 Algoritmo DIV&MOD

Input: $\bar{N} \ni N$

Dato un numero N in base dieci e una base scelta β , attraverso l'Algoritmo **DIV&MOD** posso convertire il numero N in base β .

Con *DIV* indichiamo il quoziente di una divisione tra interi, con *MOD* il resto di questa divisione; p consiste nel numero di cifre necessarie per rappresentare N in base β .

Se $N = 0$

$$a_0 = 0$$

FINE

ALTRIMENTI

$$q_0 = N$$

$$q_1 = q_0 \text{ DIV } \beta \quad a_0 = q_0 \text{ MOD } \beta$$

$$q_2 = q_1 \text{ DIV } \beta \quad a_1 = q_1 \text{ MOD } \beta$$

\dots

$$q_{p-1} = q_{p-2} \text{ DIV } \beta \quad a_{p-2} = q_{p-2} \text{ MOD } \beta$$

$$q_p = q_{p-1} \text{ DIV } \beta \quad a_{p-1} = q_{p-1} \text{ MOD } \beta$$

Svolgo la divisione ciclicamente fino a quando q_p non sarà uguale a 0. A quel punto, dopo aver calcolato a_{p-1} , arresto il tutto.

Il numero ottenuto avrà come cifre quelle dei resti, posti insieme dall'ultimo al primo ($a_{p-1}, a_{p-2}, \dots, a_1, a_0$).

Posso dire che:

$$N = a_{(p-1)} * \beta^{(p-1)} + a_{(p-2)} * \beta^{(p-2)} + \dots + a_1 * \beta^1 + a_0 * \beta^0$$

Esempio $N = 38, \beta = 6$

$$q_0 = 38$$

$$q_1 = 38 \text{ DIV } 6 = 6 \quad a_0 = 38 \text{ MOD } 6 = 2$$

$$q_2 = 6 \text{ DIV } 6 = 1 \quad a_1 = 6 \text{ MOD } 6 = 0$$

$$q_3 = 1 \text{ DIV } 6 = 0 \quad a_2 = 1 \text{ MOD } 6 = 1$$

$$a_2 a_1 a_0 = 102$$

3.3 Potenze di due, basi particolari

I numeri che sono potenze della base due possono essere individuati con agilità.

$$2^p = (1 \text{ (} 0 \dots 0 \text{)}_{p \text{ volte}})_2$$

$$\text{Esempio 1: } 2^2 = 4 = (100)_2$$

$$\text{Esempio 2: } 2^8 = 256 = (100000000)_2$$

3.3.1 Base otto ($\beta = 8$)

Le cifre della base otto sono $\{0, 1, 2, 3, 4, 5, 6, 7\}$.

Il fatto che otto sia potenza di due ($2^3 = 8$) ci permette di avere un metodo aggiuntivo per convertire un numero naturale da **base 2** a **base 8**!

Prendiamo il numero $(100101)_2$ e dividiamolo in terne, partendo da destra. Ottengo:

$$n_1 = 100$$

$$n_2 = 101$$

Osservo che il primo, convertito in base dieci, è **4**; il secondo, invece, è **5**. Posso quindi affermare che $(100101)_2 \Rightarrow (45)_8$, unendo i due numeri!

Attenzione! *Se l'ultima terna (quella più a sinistra) ha meno di tre cifre aggiungete degli zeri. Prendiamo il numero $(10101)_2$:*

$$n_1 = 010 \text{ [Ho aggiunto lo zero]}$$

$$n_2 = 101$$

*Il primo, convertito in base dieci, è **2**; il secondo, invece, è **5**. Quindi $(10101)_2 \Rightarrow (25)_8$.*

Convertiamo da base otto a base due il numero $(532)_8$

$$(532)_8 = (101 \ 011 \ 010)_2$$

Ho sostituito le cifre del numero in base otto con il loro corrispondente in base due! $(532)_8$

Tabella di conversione	
$(N)_2$ $p = 3$	$(N)_{10}$
000	0
001	1
010	2
011	3
100	4
101	5
110	6
111	7

3.3.2 Base sedici ($\beta = 16$)

Le cifre della base sedici sono $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F\}$.

La base sedici risulta particolare in un aspetto: il fatto di avere sedici simboli. Avendo superato le dieci cifre si utilizzano anche le prime sei lettere maiuscole dell'alfabeto. Osservo che $2^4 = 16$.

Prendiamo $(2A)_{16}$

$$(2A)_{16} = 2 * 16^1 + A * 16^0 = 32 + 10 * 1 = (42)_{10}$$

Dividiamo il numero in quaterne da sinistra verso destra.

$$(2A)_{16} \Rightarrow (0010 \ 1010)_2 \Rightarrow (42)_{10}$$

Ottengo anche

$$(2A)_{16} \Rightarrow (0010 \ 1010)_2 \Rightarrow (000 \ 101 \ 010)_2 \Rightarrow (52)_8$$

Posso convertire un numero da **base sedici** a **base due** sostituendo le cifre con il loro corrispondente in base binaria!

Convertiamo da base due a base sedici $(01011011)_2$

Divido in quaterne:

$$n_1 = 0101 = (5)_{16}$$

$$n_2 = 1011 = (B)_{16}$$

Quindi:

$$(01011011)_2 = (5B)_{16}$$

Tabella di conversione	
$(N)_2$ $p = 4$	N
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	<i>A</i>
1011	<i>B</i>
1100	<i>C</i>
1101	<i>D</i>
1110	<i>E</i>
1111	<i>F</i>

3.4 Intervallo di rappresentabilità su p bit

Dato p , numero di cifre utilizzabili per costruire un numero, individuo che con p cifre potrò costruire i numeri naturali compresi nel seguente intervallo:

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

La cosa è importantissima nel C++ Quando creiamo una variabile *unsigned int*, per esempio, il sistema alloca 4bytes, cioè **32bit**.

Ciò significa che in una variabile *unsigned int* l'intervallo di rappresentabilità sarà il seguente...

$$[0; 2^{32} - 1] \Rightarrow [0; 4294967295]$$

Potrò porre *unsigned int* $N = \text{numero}$ solo se:

$$0 \leq \text{numero} \leq 4294967295$$

Se non rispetto queste condizioni abbiamo un **overflow**: siamo usciti dall'intervallo di rappresentabilità.

3.5 Somme di numeri naturali espressi in binario

La somma tra due numeri naturali può essere svolta in colonna (metodo migliore per evitare errori, almeno per i principianti) utilizzando le stesse regole che abbiamo in base dieci.

Unica differenza: il riporto viene generato superando 1!

Rappresentazione dei numeri naturali (11/11)

Somma fra due numeri naturali espressi in binario.
Per sommare 10011 e 101101, basta metterli in colonna allineandoli a destra (come si fa con i numeri in base 10) e poi tenere presenti le seguenti regole:

0 +	0 +	1 +	1 +	1 +
0 =	1 =	0 =	1 =	1 =
0	1	1	10	11

Esempio: calcolare la somma di 101100 e 111010

11 ← riporti generati

```
101100 +
111010 =
1100110
```

Attenzione all'overflow! Il calcolatore lavora sempre con un numero finito di bit. Se svolgendo certe operazioni (come la somma) ottengo un numero esterno all'intervallo di rappresentabilità avrò un *overflow*.

Capitolo 4

Rappresentazione dei num. interi \mathbb{Z}

L'insieme dei numeri interi è il seguente:

$$\mathbb{Z} = \{\dots, -3, -2, -1, 0, 1, 2, 3, \dots\}$$

Il compilatore, tenendo conto del numero di byte disponibili, non può rappresentare l'insieme completo, ma soltanto un suo sottoinsieme:

$$\overline{\mathbb{Z}} = \{min, \dots, -1, 0, 1, \dots, max\}$$

I numeri interi possono essere rappresentati in tre modi, che spiegheremo nelle prossime sezioni.

4.1 Rappresentazione in modulo e segno

Alla base di questa rappresentazione abbiamo la divisione del numero binario in due parti:

- la prima (su *1bit*), che consiste nella cifra più significativa, indica il segno del numero intero (se abbiamo 0 il segno è positivo, altrimenti è negativo);
- la seconda (su $p - 1bit$), che consiste nelle cifre rimanenti, indica il valore assoluto dell'intero.

Osservazione La presenza della prima parte che indica il segno comporta avere due rappresentazioni dello zero: **zero positivo** e **zero negativo**.

4.1.1 Da intero a rappresentazione

Dato un numero rappresentato su p bit, la rappresentazione dell'intero a è la seguente:

$$A = a_{p-1}, \dots, a_0 = (segno_a; ABS_a) = (a_{p-1}; a_{p-2}, \dots, a_0)$$

Esempio 1 $a = +5$ ($p = 4$)

Il valore assoluto (ABS_a) è $5 = (101)_2$

Il segno dell'intero è positivo, quindi $segno_a = 0$.

A = (**segno_a**; **ABS_a**) = (**0**, **101**) = **0101**

Esempio 2 $a = -5$ ($p = 4$)

Il valore assoluto (ABS_a) è $5 = (101)_2$

Il segno dell'intero è negativo, quindi $segno_a = 1$.

A = (**segno_a**; **ABS_a**) = (**1**, **101**) = **1101**

4.1.2 Da rappresentazione a intero

Data una rappresentazione A in modulo e segno, l'intero a consisterà nel seguente:

$$a = (a_{p-1} == 0) ? +ABS_a : -ABS_a$$

Ho un numero in base binaria, so che è un intero e voglio convertirlo in base decimale:

- controllo la cifra più significativa a_{p-1} , se è uguale a 1 l'intero ha segno negativo, altrimenti ha segno positivo;
- controllo le cifre rimanenti, che consistono in ABS_a , e le converto in base decimale;
- esprimo come numero ABS_a , ponendo eventualmente il segno negativo

Esempio 1 $A = 01000$ ($p = 5$)

La cifra più significativa è 0, quindi ho segno positivo

Le cifre rimanenti, ABS_a , sono $(1000)_2 = 8$ **a = +8**

Esempio 2 $A = 11001$ ($p = 5$)

La cifra più significativa è 1, quindi ho segno negativo

Le cifre rimanenti, ABS_a , sono $(1001)_2 = 9$ **a = -9**

4.1.3 Tabella di conversione da -7 a $+7$ in $p = 4$

N	A	a
0	0000	+0
1	0001	+1
2	0010	+2
3	0011	+3
4	0100	+4
5	0101	+5
6	0110	+6
7	0111	+7
8	1000	-0
9	1001	-1
10	1010	-2
11	1011	-3
12	1100	-4
13	1101	-5
14	1110	-6
15	1111	-7

Osserviamo la tabella:

- sono presenti lo **zero positivo** e lo **zero negativo** annunciati precedentemente;
- il range di rappresentabilità cambia. Dato $p = 4$:
 - nei numeri naturali ho $[0, 15]$;
 - nei numeri interi, con questa rappresentazione, ho $[-7, +7]$.

4.1.4 Intervallo di rappresentabilità in modulo e segno su p bit

Dato p , numero di cifre utilizzabili per costruire un numero, individuo che con p cifre potrò costruire i numeri interi compresi nel seguente intervallo:

$$[-(2^{p-1} - 1); +(2^{p-1} - 1)]$$

Esempio 1 $p = 4$ $[-(2^3 - 1); +(2^3 - 1)]$ $[-7; +7]$

Esempio 2 $p = 8$ $[-(2^7 - 1); +(2^7 - 1)]$ $[-127; +127]$

Attenzione all'overflow Prima di ottenere la rappresentazione di un intero ($a \Rightarrow A$) è necessario verificare che il valore assoluto sia effettivamente rappresentabile in $p - 1$ bit. Ricordiamo che la cifra più significativa del numero binario è utilizzata per esprimere il segno.

Esempio in overflow $a = -9$ con $p = 4$ bit. $segno_a = 1$ $ABS_a = 9 = (1001)_2$
Il numero ottenuto è 11001. Tenendo conto che di 4 bit posso usarne solo 3 per ABS_a (ne sto usando 4), il numero non può essere rappresentato in $p = 4$!

4.2 Rappresentazione in complemento a 2

Anche in questa rappresentazione la cifra più significativa dei numeri binari indica il segno dell'intero! La parte rimanente, tuttavia, non può essere associata al valore assoluto dell'intero come prima.

4.2.1 Da intero a rappresentazione

Dato un intero a , la sua rappresentazione *in complemento a 2* su p bit è la seguente:

$$A = a_{p-1} \dots a_0 = (a \geq 0) ? ABS_a : (2^p - ABS_a)$$

Sia ABS_a che $2^p - ABS_a$ sono rappresentati in *base due* come numeri naturali su p bit.
In breve:

- Ho un intero a che voglio rappresentare su p bit;
- verifico se l'intero a è maggiore, uguale, o minore di zero
 - Se l'intero è maggiore o uguale a zero il numero consiste nel valore assoluto dell'intero, quindi nella sua conversione in base binaria.
 - Se l'intero è minore di zero svolgerò il calcolo presente nella notazione dove ABS_a è convertito in base binaria $(2^p - ABS_a)$.

Esempio 1 $a = +7$ ($p = 4$)
L'intero a è maggiore di zero.

$$ABS_a = 7 = (0111)_2$$
$$A = ABS_a = (0111)_2$$

Esempio 2 $a = -8$ ($p = 4$)
L'intero a è minore di zero.

$$ABS_a = 8$$
$$A = 2^p - ABS_a = 2^4 - 8 = 8 = (1000)_2$$

4.2.2 Da rappresentazione a intero

Data una rappresentazione A in complemento a due, l'intero a consisterà nel seguente:

$$a = (a_{p-1} == 0) ? +A : -(2^p - A)$$

Sia A che $2^p - A$ sono visti come numeri naturali su p bit.

In breve:

- ho una rappresentazione A su p bit, voglio convertirla nell'intero corrispondente;
- verifico se l'intero è positivo o negativo controllando il valore della cifra più significativa:
 - se il valore è zero ho un intero positivo ed a è uguale ad A convertito in base decimale;
 - se il valore è uno ho un intero negativo, svolgo il calcolo presente nella notazione dove A è convertito in base decimale $[-(2^p - A)]$.

Esempio 1 $(0111)_2$ ($p = 4$)

La cifra più significativa è 0, quindi ho un intero positivo.

Converto A in base decimale, ottengo $\mathbf{a} = \mathbf{A} = 7$

Esempio 2 $(1111)_2$ ($p = 4$)

La cifra più significativa è 1, quindi ho un intero negativo.

Converto A in base decimale, ottengo $\mathbf{A} = 15$

Svolgo il calcolo e ottengo: $\mathbf{a} = -(2^p - \mathbf{A}) = -(2^4 - 15) = -1$

4.2.3 Tabella di conversione da -8 a $+7$ in $p = 4$

\mathbb{N}	\mathbf{A}	\mathbf{a}
0	0000	+0
1	0001	+1
2	0010	+2
3	0011	+3
4	0100	+4
5	0101	+5
6	0110	+6
7	0111	+7
8	1000	-8
9	1001	-7
10	1010	-6
11	1011	-5
12	1100	-4
13	1101	-3
14	1110	-2
15	1111	-1

Osserviamo la tabella:

- non ho, in questa rappresentazione, la presenza dello **zero positivo** e dello **zero negativo** (ne ho uno solo);
- il range di rappresentabilità cambia ancora. Dato $p = 4$:
 - nei numeri naturali ho $[0, 15]$;

- nei numeri interi, con rappresentazione in modulo e segno, ho $[-7, +7]$.
- nei numeri interi, con questa rappresentazione, ho $[-8, +7]$.

Dettaglio interessante: dato $a = -1$, con qualunque p , la rappresentazione A consisterà in un numero binario formato esclusivamente da 1.

$$\begin{aligned} a = -1 \quad (p = 3) &\Rightarrow A = 111 \\ a = -1 \quad (p = 6) &\Rightarrow A = 111111 \\ a = -1 \quad (p = 7) &\Rightarrow A = 1111111 \end{aligned}$$

4.2.4 Intervallo di rappresentabilità in complemento a 2 su p bit

Dato p , numero di cifre utilizzabili per costruire un numero, individuo che con p cifre potrò costruire i numeri interi compresi nel seguente intervallo:

$$[-2^{p-1}; +(2^{p-1} - 1)]$$

Attenzione! Prima di calcolare la rappresentazione mediante l'algoritmo è necessario verificare che l'intero a sia rappresentabile su p bit

Esempio $a = -9$ ($p = 4$ bit)

L'intero è negativo, pertanto svolgo il calcolo $2^4 - 9 = 7 = (0111)_2$

Attenzione, il dato è sbagliato (vedere per esempio la prima cifra uguale a zero) perchè per rappresentare -9 avrei bisogno di almeno 5 bit.

Riprovo con $p = 5$: $A = 2^5 - 9 = 23 = (10111)_2$

4.3 Rappresentazione con bias

La rappresentazione **con bias** sarà estremamente utile nella rappresentazione dei numeri reali in virgola mobile. Osservando la tabella di conversione possiamo comprendere perchè essa è definita **rappresentazione con polarizzazione!**

Alla base di essa abbiamo **bias** = $2^{p-1} - 1$.

Anche qua, come nel complemento a due, non abbiamo la doppia rappresentazione dello zero.

4.3.1 Da intero a rappresentazione

Dato un intero a , la sua rappresentazione *in bias* su p bit è la seguente:

$$A = a_{p-1} \dots a_0 = a + \text{bias} = a + (2^{p-1} - 1)$$

Dove $a + (2^{p-1} - 1)$ è supposto non negativo e quindi numero naturale su p bit.

Esempio 1 $a = 8$ ($p = 4$)

$$A = 8 + (2^{4-1} - 1) = 15$$

15, convertito in base binaria, consiste in $A = (1111)_2$

Esempio 2 $a = -7$ ($p = 4$)

$$A = -7 + (2^{4-1} - 1) = 0$$

Ovviamente avrò $A = (0000)_2$

4.3.2 Da rappresentazione a intero

Data una rappresentazione A in *bias*, l'intero a consisterà nel seguente:

$$a = A - \text{bias} = A - (2^{p-1} - 1)$$

Dove A è visto come un numero naturale su p bit.

Esempio 1 $A = 0111$ ($p = 4$)
 $a = 7 - (2^{4-1} - 1) = 0$

Esempio 2 $A = 0101$ ($p = 4$)
 $a = 5 - 2^{4-1} - 1 = -2$

4.3.3 Intervallo di rappresentabilità in bias

Dato p , numero di cifre utilizzabili per costruire un numero, individuo che con p cifre potrò costruire i numeri interi compresi nel seguente intervallo:

$$[-\text{bias}; \text{bias} + 1] = [-2^{(p-1)} + 1; 2^{(p-1)}]$$

Esempio 1 $p = 4$ $[-7; +8]$

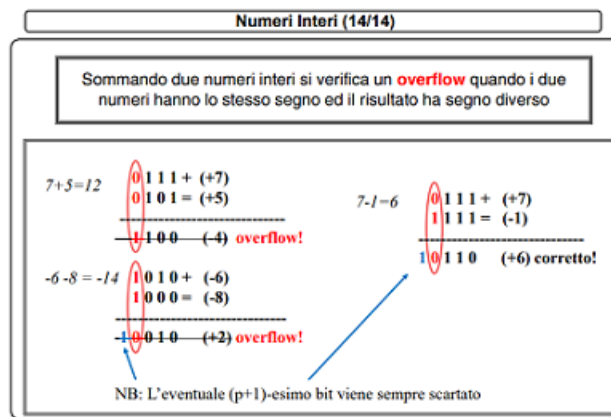
Esempio 2 $p = 7$ $[-63; +64]$

Come sempre attenzione all'overflow Prima di individuare la rappresentazione dell'intero è necessario verificare che l'intero sia effettivamente rappresentabile su p bit.

Prendiamo $a = -9$ con $p = 4$ bit:

ottengo $A = -9 + (2^{4-1} - 1) = -9 + 7 = -2$ che non è numero naturale!

4.4 Somma di numeri interi



57

Osservazioni sulla somma nel complemento a due Il complemento a due presenta un vantaggio importante: le rappresentazioni possono essere sommate come numeri naturali, ottenendo in base binaria l'effettivo risultato della somma. Ciò ci permette di utilizzare direttamente la circuiteria per la somma e la sottrazione dei numeri naturali, evitandone una aggiuntiva per gli interi!

4.5 Tabella di confronto delle rappresentazioni

\mathbb{N}	$\{0,1\}^5$	\mathbf{a}^{ms}	\mathbf{a}^{c2}	\mathbf{a}^{bias}
0	00000	+0	0	-15
1	00001	+1	+1	-14
2	00010	+2	+2	-13
3	00011	+3	+3	-12
4	00100	+4	+4	-11
5	00101	+5	+5	-10
6	00110	+6	+6	-9
7	00111	+7	+7	-8
8	01000	+8	+8	-7
9	01001	+9	+9	-6
10	01010	+10	+10	-5
11	01011	+11	+11	-4
12	01100	+12	+12	-3
13	01101	+13	+13	-2
14	01110	+14	+14	-1
15	01111	+15	+15	0
16	10000	-0	-16	+1
17	10001	-1	-15	+2
18	10010	-2	-14	+3
19	10011	-3	-13	+4
20	10100	-4	-12	+5
21	10101	-5	-11	+6
22	10110	-6	-10	+7
23	10111	-7	-9	+8
24	11000	-8	-8	+9
25	11001	-9	-7	+10
26	11010	-10	-6	+11
27	11011	-11	-5	+12
28	11100	-12	-4	+13
29	11101	-13	-3	+14
30	11110	-14	-2	+15
31	11111	-15	-1	+16

Capitolo 5

Rappresentazione dei num. reali \mathbb{R}

5.1 Virgola fissa (notazione ordinaria)

Nella rappresentazione mediante *virgola fissa* di un numero reale r (che occupa p bit) si distingue una parte intera $I(r)$ di $(p-f)$ bit e una frazionaria $F(r)$ di (f) bit. Individuo che la rappresentazione del numero reale in questione sarà:

$$R = a_{(p-f-1)} \dots a_0 a_{(-1)} \dots a_{(-f)}$$

Il procedimento inverso ci porterà alla seguente sommatoria (dove β è la base in cui convertiremo la rappresentazione R espressa in base binaria):

$$r \cong \sum_{i=-f}^{(p-f-1)} a_i \beta^i = a_{(p-f-1)} \beta^{(p-f-1)} + \dots + a_{(-1)} \beta^{(-1)} + \dots + a_{(-f)} \beta^{(-f)}$$

Nelle sequenze binarie non si rappresenta la virgola. La parte intera è facilmente individuabile mediante le tecniche che già conosciamo, la parte frazionaria richiede la cosiddetta ***procedura parte frazionaria-parte intera***.

5.1.1 Procedura *parte frazionaria-parte intera*

Lo scopo di questo algoritmo è individuare la parte frazionaria di un numero. Teniamo conto che in alcuni casi la procedura può continuare all'infinito. Possiamo fermarci in due possibili circostanze:

- non appena raggiungiamo la precisione desiderata (o troviamo tutte le cifre decimali, $f_x = 0$)
- quando abbiamo occupato tutte i p bit a disposizione. In un compilatore non posso rappresentare un numero r avente infinite cifre!

Ricordiamoci che maggiore è la precisione, maggiore è il numero di cifre individuate. Fermandomi effettuerò un troncamento a una *certa cifra* dopo la virgola, ottenendo un numero r' diverso da quello iniziale r . Proprio per questo si parla di ***errore di troncamento***!

Alla *slide 63* è presente un esempio di troncatura: l'errore di troncamento viene calcolato individuando la differenza tra r e r' (ovviamente maggiore è la precisione, minore è l'errore di troncamento).

Procedimento - Troviamo parte frazionaria del numero r

- Inizio ponendo $f_0 = F(r)$ (parte frazionaria del numero r).
- Se la parte frazionaria è già uguale a zero posso fermarmi subito (significa che non ho cifre decimali).
- In caso contrario svolgo le seguenti operazioni:
$$\begin{aligned}f_{(-1)} &= F(f_0 * 2) & a_{-1} &= I(f_0 * 2) \\f_{(-2)} &= F(f_{(-1)} * 2) & a_{-1} &= I(f_{(-1)} * 2) \\&\dots \text{e così via!}\end{aligned}$$
 - Moltiplico la precedente parte frazionaria per due.
 - Del risultato trovo sia la parte decimale (che userò nello step successivo) che la parte intera.**Quest'ultima mi fornirà una cifra della parte frazionaria in base binaria!**
- Continuo fermandomi nei casi citati all'inizio della sezione.

Esempio di conversione di un numero reale r

Numero di bit a disposizione: $p = 16$

Numero di bit a disposizione per la parte intera: $f = 5$

Numero da rappresentare: $r = +331,6875$

Troviamo la parte intera $I(r)$

Posso trovare questa parte come un qualunque numero naturale.

Mediante l'Algoritmo *DIV&MOD* individuo le cifre binarie. Trovo **331** = **(101001011)₂**

Troviamo la parte frazionaria $F(r)$

$$f_0 = F(r) = 0,6875$$

$$\begin{aligned}f_{(-1)} &= F(f_0 * 2) = F(0,6875 * 2 = 1.375) = 0.375 \\a_{(-1)} &= I(f_0 * 2) = I(1.375) = \mathbf{1}\end{aligned}$$

$$\begin{aligned}f_{(-2)} &= F(f_{(-1)} * 2) = F(0.375 * 2 = 0.75) = 0.75 \\a_{(-2)} &= I(f_{(-1)} * 2) = I(0.75) = \mathbf{0}\end{aligned}$$

$$\begin{aligned}f_{(-3)} &= F(f_{(-2)} * 2) = F(0.75 * 2 = 1.5) = 0.5 \\a_{(-3)} &= I(f_{(-2)} * 2) = I(1.5) = \mathbf{1}\end{aligned}$$

$$\begin{aligned}f_{(-4)} &= F(f_{(-3)} * 2) = F(0.5 * 2 = 1.0) = 0 \\a_{(-4)} &= I(f_{(-3)} * 2) = I(1.0) = \mathbf{0}\end{aligned}$$

Risultato:

- Ricordo che rappresento il numero su 16 *bit* e che 5 *bit* sono riservati alla parte frazionaria.
- Della parte frazionaria ho trovato soltanto quattro cifre (poichè alla quarta cifra ho parte intera nulla e sono costretto a fermarmi), quindi aggiungo uno zero alla fine ($f = 5$).
- Aggiungo due zeri all'inizio del numero poichè con le cifre trovate occupo 14/16 *bit*.
- **Quindi:** $R = (0010100101110110)_2$

5.2 Virgola mobile (notazione scientifica)

La notazione di cui abbiamo parlato fino ad ora risulta buona se applicata a semplici calcoli: spostandoci in ambito scientifico ci imbattiamo nella necessità di avere una precisione elevata, quindi numeri aventi molte cifre. Risulta necessario, conseguentemente, separare l'intervallo di rappresentabilità dalla precisione, cioè dal numero di cifre!

5.2.1 Notazione scientifica

Un numero binario in virgola mobile consiste nel seguente:

$$r = \pm m * \beta^e$$

Dove abbiamo i seguenti elementi:

- **m**: la *mantissa*. Parte espressa in virgola fissa, la sua parte intera occupa un solo bit. Il numero di cifre della mantissa determina la precisione del numero.
- **e**: l'*esponente*. Stabilisce la vera posizione del punto nel numero (per questo si parla di virgola mobile). L'intervallo di rappresentabilità è fissato dal suo numero di cifre.

Tra tutte le forme possibili mediante la notazione scientifica ne scelgo una da utilizzare in ogni caso che definirò **forma normalizzata**: in questa è riservato un solo bit alla parte intera della mantissa.

Esempio con differenza tra virgola fissa e virgola mobile:

Numero reale binario espresso in virgola fissa: +1110.01

Esempi in cui esprimo lo stesso numero in virgola mobile¹:

$$+1.11001 * 2^{+11}$$

$$+111001 * 2^{-10}$$

5.2.2 Rappresentazione di r

La rappresentazione R consiste in una tripla costituita da tre numeri naturali, che sono:

$$R = \left\{ \underbrace{s}_{1 \text{ bit}}, \underbrace{E}_{K \text{ bit}}, \underbrace{F}_{G \text{ bit}} \right\}$$

- **s**: bit singolo per la codifica del segno del numero (0 = positivo, 1 = negativo)
- **E**: parte dedicata alla codifica dell'esponente
- **F**: parte dedicata alla codifica della parte frazionaria della mantissa

Individuiamo una serie di rappresentazioni possibili, secondo lo standard internazionale, in cui variano i bit distribuiti:

- *half-precision*: 16 bit, di cui 5 per l'esponente della mantissa e 10 per la parte frazionaria della stessa ($K = 5$ e $G = 10$).
- *single-precision*: 32 bit, di cui 8 per l'esponente della mantissa e 23 per la parte frazionaria della stessa ($K = 8$ e $G = 23$)
- *double-precision*: 64 bit, di cui 11 per l'esponente della mantissa e 52 per la parte frazionaria della stessa ($K = 11$ e $G = 52$)

¹Esponenti espressi in base binaria

- *quadruple-precision*²: 128 bit, di cui 15 per l'esponente della mantissa e 112 per la parte frazionaria della stessa ($K = 15$ e $G = 112$)

Successivamente parleremo dell'intervallo di rappresentabilità.

5.2.3 Trasformazione $\mathbf{R} \rightarrow \mathbf{r}$

Prendiamo come riferimento la seguente formula, presente nelle slides. Le lettere presenti fanno riferimento alla tripla (cioè alla rappresentazione):

$$r = (s == 0) ? [(1 + f) 2^e] : [-(1 + f) 2^e]$$

- Per prima cosa individuo il segno del numero reale. Prendo s : se è 0 ho segno positivo, altrimenti segno negativo
- E è una rappresentazione bias: con i metodi che già conosciamo trovo l'esponente e

$$e = E - bias \quad (bias = 2^{K-1} - 1)$$

- Trovo la mantissa m , partendo dalla sequenza binaria F :
 - Trovo la parte frazionaria della mantissa: $f = \frac{F}{2^G} = F 2^{-G}$ (un numero $0, \dots$)
 - Sommo 1 alla parte frazionaria: $m = 1 + f = 1 + F 2^{-G}$.

Osservazione Lo zero non può essere rappresentato!

- La parte intera della mantissa avrà sempre valore 1 ($\rightarrow m \neq 0$)
- La potenza 2^e non mi restituirà mai un valore nullo ($2^e \neq 0$)

Quindi $r = m * 2^e \neq 0$

Esempio 1 $R = \{1, 10011, 1110100101\}$

- $s = 1$, quindi il numero r avrà segno negativo
- Prendo la rappresentazione con bias E e trovo l'esponente

$$e = E - bias = 19 - (2^{5-1} - 1) = 19 - (16 - 1) = 19 - 15 = 4$$

- Trovo la mantissa m
 - $f = F 2^{-10} = 1110100101 2^{-10} = 0.1110100101$
 - Sommo 1: $m = 1 + f = 1 + 0.1110100101 = 1.1110100101$

- Quindi:

$$r = -(1.1110100101 2^4) = -(11110.100101)$$

- Trovo il numero mediante la sommatoria:

$$\begin{aligned} r &= -(1 * 2^4 + 1 * 2^3 + 1 * 2^2 + 1 * 2^1 + 0 + 1 * 2^{-1} + 0 + 0 + 1 * 2^{-4} + 0 + 1 * 2^{-6}) \\ &= -\left(16 + 8 + 4 + 2 + \frac{1}{2} + \frac{1}{16} + \frac{1}{64}\right) \\ &= -\left(30 + \frac{1}{2} + \frac{1}{16} + \frac{1}{64}\right) \\ &= -(30 + 0.5 + 0.0625 + 0.015625) = -30.578125 \end{aligned}$$

²aggiunto da uno standard più recente

Esempio 2 $R = \{0, 01110, 0000000000\}$

- $s = 0$, quindi il numero r avrà segno positivo
- Prendo la rappresentazione con bias E e trovo l'esponente

$$e = E - bias = 14 - (2^{5-1} - 1) = 14 - (16 - 1) = -1$$

- Trovo la mantissa m
 - $f = F 2^{-10} = 0$, non ho bisogno di far calcoli
 - Sommo 1: $m = 1 + f = 1$
- Quindi:

$$r = +1.0 2^{-1} = +\frac{1}{2} = +0.5$$

Vista la semplicità non è necessario utilizzare la sommatoria.

5.2.4 Trasformazione $r \rightarrow R$

Per spiegare questa trasformazione prendiamo direttamente un numero: $r = 2.3$ in half precision.

- Il numero ha segno positivo, quindi $s = 0$
- Individuo la rappresentazione in virgola fissa, che è 10.01001
- Porto il numero in forma normalizzata

$$1.001001 2^{+1}$$

- Ho bisogno di dieci cifre nella parte frazionaria. La parte sottolineata è periodica, quindi:

$$1.0010011001 2^{+1}$$

- Dalla parte frazionaria della mantissa ottengo F

$$0.0010011001 2^{10} = 0010011001$$

- Ottengo la rappresentazione con bias dell'esponente, quindi E

$$E = e + bias = 1 + (2^{5-1} - 1) = 1 + (2^4 - 1) = 16 \rightarrow E = 10000$$

- Risultato:

$$R = \{0, 10000, 0010011001\}$$

5.2.5 Numeri con modulo massimo - $\pm\infty$ - intervallo di rappresentabilità

Ho le seguenti rappresentazioni, che consistono nei numeri aventi modulo massimo:

- $R = \{0, 11111, 1111111111\}$
- $R = \{1, 11111, 1111111111\}$

Individuiamo la trasformazione $R \rightarrow r$:

- $e = E - bias = 31 - (2^4 - 1) = 31 - 15 = 16$
- $f = F 2^{-10} = 0.1111111111$
- $|r| = 1.111111111 2^{16} < 2^{(2^{K-1}-1)+2} = 2^{bias+2}$

L'intervallo di rappresentabilità $[-\infty, +\infty]$ è approssimabile dal seguente

$$[\approx -2^{bias+2}, \approx 2^{bias+2}]$$

5.2.6 Numeri con modulo minimo - ± 0

Ho le seguenti rappresentazioni, che consistono nei numeri aventi modulo minimo:

- $R = \{0, 00000, 0000000000\}$
- $R = \{1, 00000, 0000000000\}$

Individuiamo la trasformazione $R \rightarrow r$:

- $e = E - bias = 0 - (2^4 - 1) = 0 - 15 = -15$
- $f = 0.0000000000$, senza stare a fare calcoli
- $|r| = 1.0000000000 \cdot 2^{-15} = 2^{-bias}$

Nella rappresentazione half-precision individuo che:

- $+0 = 2^{-15} \approx 0.31 \cdot 10^{-4}$
- $-0 = -2^{-15} \approx -0.31 \cdot 10^{-4}$

Capitolo 6

Esercizi

I seguenti esercizi sono stati forniti dal Cococcioni verso metà semestre (in fondo trovate anche un esercizio fatto dal Perazzo durante un esercitazione)

Esercizio 1

Sia **A** il naturale che rappresenta l'intero **a** = -31 in complemento a due su 6 bit. Trovare **A**, esprimendolo in binario. Successivamente trovarne anche la rappresentazione in base 16.

A (espresso in binario) ☐ ☐ ☐ ☐ ☐ ☐

A (espresso in base 16) _____

Nello spazio sottostante riportare i passaggi più significativi della soluzione

La rappresentazione cercata è data dal naturale $2^6 - ABS(-31) = 64 - 31 = 33$, ossia $2^5 + 2^0$: 100001.
Il naturale 33 in base sedici ha rappresentazione 21, come si può facilmente scoprire raggruppando le cifre di quattro in quattro da destra ed aggiungendo 2 zeri a sinistra:
 $\overbrace{0010}^2\text{-}\overbrace{0001}^1$
Si sarebbe giunti alla stessa soluzione, ovviamente, applicando l'algoritmo di *mod&div* al naturale 33 da dividere per 16:
$$\begin{array}{r|l} 33 & 16 \\ 2 & 1 \\ \hline 0 & \end{array}$$

(ovviamente i due resti 1 e 2 vanno presi in ordine inverso).

Soluzione Esercizio 1

Esercizio 2

Data la rappresentazione $A = 01101$, dire a quale intero a essa corrisponde nell'ipotesi di aver utilizzato la rappresentazione con *bias* su 5 bit.

Inoltre, data la rappresentazione $R = \{0, 01101, 1000000000\}$, trovare il numero reale r corrispondente nell'ipotesi di rappresentazione in virgola mobile e precisione dimezzata (*half precision*).

$a = \underline{\hspace{2cm}} \qquad r = \underline{\hspace{2cm}}$

Nello spazio sottostante riportare i passaggi più significativi della soluzione

Per trovare a da A basta togliere ad A il bias. Il bias vale 2^{K-1} . Per $K=5$, bias = $01111 = 15$.
 Pertanto $01101-01111 = 13-15 = -2 \Rightarrow$ Pertanto l'intero a cercato vale -2 .
 Data $R=\{0, 01101, 1000000000\}$, si ha che:
 $s = 0$ (il numero reale r cercato è dunque positivo)
 $E = 01101$ ed
 $F = 1000000000$
 La mantissa m sarà pertanto il reale 1.1000000000 (per via dell'1 implicito).
 L'esponente $e = -2$, in virtù della rappresentazione con bias (vedere sopra).
 Pertanto $r = m \cdot 2^e = 1.1 \cdot 2^{-2} = 0.011 \cdot 2^0 = 2^{-2} + 2^{-3} = 0.25 + 0.125 = 0.375$

Soluzione Esercizio 2

Esercizio 3

Trovare la rappresentazione di -64 in complemento a due su 7 bit, se rappresentabile. Trovarne inoltre la rappresentazione in modulo e segno, sempre su 7 bit (e sempre al patto che sia rappresentabile).

<input type="checkbox"/> -64 è rappresentabile in <i>c2</i> su 7 bit (<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>)	<input type="checkbox"/> -64 è rappresentabile in <i>ms</i> su 7 bit (<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>)
<input type="checkbox"/> -64 non è rappresentabile in <i>c2</i> su 7 bit	<input type="checkbox"/> -64 non è rappresentabile in <i>ms</i> su 7 bit

Nello spazio sottostante riportare i passaggi più significativi della soluzione

L'intervallo di rappresentabilità in complemento a 2 su 7 bit è $[-2^{(7-1)}, +2^{(7-1)}-1]$, ossia $[-64, +63]$.
Pertanto -64 è rappresentabile. La sua rappresentazione A è data da $2^7 - 64 = 128 - 64 = 64 \Rightarrow$ **1000000**.
Invece l'intervallo di rappresentabilità in modulo e segno su 7 bit è $[-2^{(7-1)}-1, +2^{(7-1)}-1]$, ossia $[-63, +63]$ e di conseguenza -64 non è rappresentabile in modulo e segno su 7 bit.

Soluzione Esercizio 3

Esercizio 4

Dara la rappresentazione in virgola mobile $R=\{s,F,E\}$, dove $s=1$, $F=01000$ (mantissa su 5 bit) ed $E=010$ (esponente su 3 bit), a quale numero reale r corrisponde?

$r =$ _____

(spazio riservato allo svolgimento dell'esercizio)

La mantissa del numero è 1.01000 per via dell'uno implicito.
Il numero r in base due è 1.01000·due⁻¹, ossia 0.101000, che in base dieci vale $2^{-1}+2^{-3} = 0.5 + 0.125 = > r = -0.625$

Soluzione Esercizio 4

Esercizio 5

Trovare la rappresentazione binaria **A** in complemento a 2 dell'intero **a** = -31 su $p = 6$ bit.

Trovare inoltre la rappresentazione in base 8 del naturale **A** appena trovato.

A (in base 2) _____ A (in base 8) _____

(spazio riservato allo svolgimento dell'esercizio)

Trattandosi di un numero negativo, il naturale A vale $2^p - |-31| = 64 - 31 = 33$ (10), ossia $A = 10001$ (2).
Per esprimere A in base 8 basta raggruppare le cifre a tre a tre partendo da sinistra e decodificare
ciascun gruppo mediante conversione da binario a decimale: $A = 41$ (8)

Soluzione Esercizio 5

Esercizio 6

Dato il numero naturale A la cui codifica in base 5 è 104, trovarne la codifica in base 2.

Successivamente, assumendo che la rappresentazione binaria trovata sia quella dell'esponente E di un numero reale in *half precision* ($K=5$), dire quanto vale l'esponente intero e corrispondente.

A (in base 2) _____ e _____

(spazio riservato allo svolgimento dell'esercizio)

Il numero $104_{(5)}$ corrisponde al decimale $1 \cdot 5^2 + 4 \cdot 5^0 = 29_{(10)}$.
 $29_{(10)}$ equivale a $16+8+4+1$, ossia $11101_{(2)}$.
Considerando 11101 come il valore di E , l'intero e corrispondente è il naturale che si ottiene sottraendo ad E il bias $B = K \cdot 5^{-1} - 1 = 15$, ossia $29 - 15 = 14$. Dunque l'esponente e è positivo e vale 14.

Soluzione Esercizio 6

Esercizio 7

Rappresentare il numero intero decimale $a = -31$ in base 3 su 5 cifre, mediante tecnica del modulo e segno (si utilizzi la cifra 1 per codificare il segno).

A (ms base 3)

(spazio riservato allo svolgimento dell'esercizio)

Per prima cosa occorre trovare la rappresentazione del modulo di a su 4 cifre.
Notando che $31 = 3^3 + 3^1 + 3^0$, segue che la rappresentazione su quattro cifre è 1011.
Premettendo il segno (rappresento con la cifra ad 1, in analogia a quanto avviene in base 2), si ottiene la rappresentazione cercata: 11011.

Soluzione Esercizio 7

Esercizio 8

Sia data la seguente rappresentazione R_1 di un numero reale in virgola mobile su 8 bit (tre per l'esponente, quattro per la mantissa, uno per il segno): $R_1 = \{S, E, F\}$, con $S = 0$, $E = 010$ ed $F = 1010$.

- A quale numero reale r_1 corrisponde la rappresentazione R_1 ?
- Quanto vale in base dieci il massimo numero r_2 rappresentabile su 8 bit, ossia quello associato al “ $+\infty$ ” ?

r_1 _____	r_2 _____
-------------	-------------

Nello spazio sottostante riportare i passaggi più significativi della soluzione

La mantissa $m_1 = 1.1010$, per via dell'uno implicito.
 L'esponente e_1 vale $E_1 - bias = 010 - 011 = -1$
 $S = 0$ (ossia il numero è positivo).
 Pertanto $r_1 = 1.1010 \cdot \text{due}^{-1} = 0.11010 = 2^{-1} + 2^{-2} + 2^{-4} = 0.5 + 0.25 + 0.0625 = +0.8125$
 Il massimo reale positivo è quello associato al segno positivo ($S_2 = 0$), al massimo esponente $E_2 = 111$ ed alla massima mantissa $F_2 = 1111$.
 La mantissa m_2 associata vale: 1.1111 , mentre l'esponente e_2 associato vale: $111 - 011 = 7 - 3 = 4$.
 Pertanto $r_2 = 1.1111 \cdot \text{due}^{+4} = 11111 = 2^4 + 2^3 + 2^2 + 2^1 + 2^0 = 16$

Soluzione Esercizio 8

Esercizio 9

- Sia dato il numero 105 in base 6. Trovarne la rappresentazione in base 10 e quella in base 2 .

Base 10 _____

Base 2 _____

Nello spazio sottostante riportare i passaggi più significativi della soluzione

Pertanto la rappresentazione in base due cercata consta di 6 bit ($a_5 \dots a_0$) : **101001**.

$$\begin{array}{ll}
 q_6 = 1 \text{ div } 2 = 0 & a_5 = 1 \text{ mod } 2 = 1 \\
 q_5 = 2 \text{ div } 2 = 1 & a_4 = 2 \text{ mod } 2 = 0 \\
 q_4 = 5 \text{ div } 2 = 2 & a_3 = 5 \text{ mod } 2 = 1 \\
 q_3 = 10 \text{ div } 2 = 5 & a_2 = 10 \text{ mod } 2 = 0 \\
 q_2 = 20 \text{ div } 2 = 10 & a_1 = 20 \text{ mod } 2 = 0 \\
 q_1 = 41 \text{ div } 2 = 20 & a_0 = 41 \text{ mod } 2 = 1 \\
 q_0 = 41 &
 \end{array}$$

La rappresentazione di $(41)_{10}$ in base 2 si può ottenere con la procedura Div&Mod:

$$1 \cdot 6^2 + 5 \cdot 6^0 = 36 + 5 = \mathbf{(41)_{10}}.$$

105 in base 6 corrisponde a 41 in base 10. Infatti, utilizzando la formula della sommatoria, si ottiene:

Soluzione Esercizio 9

Esercizio 10

Dato il numero reale $r = 0.3125$, trovarne la rappresentazione binaria R in virgola fissa, usando 6 bit per la parte decimale.

$$R = 0.\square\square\square\square\square\square$$

Nello spazio sottostante riportare i passaggi più significativi della soluzione

$f_0 = 0.3125$

$f_1 = F(f_0 \cdot 2 = 0.3125 \cdot 2 = 0.625) = 0.625$	$a_1 = I(0.625) = 0$
$f_2 = F(f_1 \cdot 2 = 0.625 \cdot 2 = 1.25) = 0.25$	$a_2 = I(1.25) = 1$
$f_3 = F(f_2 \cdot 2 = 0.25 \cdot 2 = 0.5) = 0.5$	$a_3 = I(0.5) = 0$
$f_4 = F(f_3 \cdot 2 = 0.5 \cdot 2 = 1.0) = 0$	$a_4 = I(1.0) = 1$

Avevo ottenuto f_4 pari a zero, l'algoritmo si ferma. Questo significa che il numero reale 0.3125 ha rappresentazione binaria esatta, pari a $(a_1 a_2 a_3 a_4) = 0101$.

Poiché era richiesta la rappresentazione su 6 bit per la parte frazionaria, si aggiungono due ulteriori zeri a destra: $R = 0.010100$.

Si pone $f_0 = 0.3125$ e poi si applica l'algoritmo "Parte frazionaria – Parte intera":

Soluzione Esercizio 10

Esercizio 11

Data la rappresentazione $A = (11110001)_2$ di un intero in complemento a 2, trovarne il valore in base 10.

a = _____

Nello spazio sottostante riportare i passaggi più significativi della soluzione

Ricapitolando **a** = **-15**.

$$2^8 - 111110001 = 256 - (2^7 + 2^6 + 2^5 + 2^4 + 2^0) = 256 - (128 + 64 + 32 + 16 + 1) = 256 - 241 = (15)_{10}, \text{ ossia } \textit{quindici}.$$

dalla formula due⁸-A, dove A viene trattato come se fosse un naturale:

Dopo aver notato che A inizia per 1 e che pertanto **a** non può che essere negativo, il suo valore assoluto è dato

Soluzione Esercizio 11

Esercizio 12

Data la rappresentazione in virgola mobile $\mathbf{R} = \{s, F, E\}$, dove $s=1$, $F=01000$ (mantissa su 5 bit) ed $E=010$ (esponente su 3 bit), a quale numero reale r corrisponde?

$r =$ _____

Nello spazio sottostante riportare i passaggi più significativi della soluzione

La mantissa m di r è: 1.01000, per via dell'uno implicito. L'esponente è dato da E meno il bias. In questo caso E vale 010 ed il bias 011, dunque la differenza $r=1.01000\cdot 2^{-1}$, ossia 0.101000 (basta infatti spostare il separatore decimale a sinistra di uno).

Pertanto $r = 2^{-1} + 2^{-3} = 0.5 + 0.125 = -0.625$.

Soluzione Esercizio 12

Esercizio 13

Dato il numero reale r , la cui rappresentazione in virgola mobile $\mathbf{R} = \{s, F, E\}$ sia: $s=1$, $F=01000$ (mantissa su 5 bit) ed $E=010$ (esponente su 3 bit), trovare la rappresentazione del numero reale $r'=2 \cdot r$.

$$\mathbf{R}' = \{s' = \square, F' = \square\square\square\square\square, E' = \square\square\square\}$$

Nello spazio sottostante riportare i passaggi più significativi della soluzione

Controlla: decodificando \mathbf{R} si scopre che $r = 0.625$, mentre decodificando \mathbf{R}' si scopre che r' è 1.25, laddove 1.25 è effettivamente il doppio di 0.625.

$$\mathbf{R}' = \{s' = 1, F' = 01000, E' = 011\}$$

r' avrà la stessa rappresentazione di r , ma con l'esponente incrementato di uno. Pertanto $E' = 011$. Complessivamente la rappresentazione cercata è:

Soluzione Esercizio 13

Esercizio 14

Sia data la seguente rappresentazione R_1 di un numero reale in virgola mobile su 8 bit (due per l'esponente, cinque per la mantissa, uno per il segno): $R_1 = \{S, E, F\}$, con $S = 0$, $E = 10$ ed $F = 00110$.

- A quale numero reale r_1 corrisponde la rappresentazione R_1 ?
- Quanto vale in base dieci il minimo numero r_2 rappresentabile su 8 bit, ossia quello associato al “ $-\infty$ ” ?

r_1 _____	r_2 _____
-------------	-------------

Nello spazio sottostante riportare i passaggi più significativi della soluzione

La mantissa $m_1 = 1.00110$, per via dell'uno implicito.
 L'esponente e_1 vale $E_1 - bias = 10 - 01 = +1$
 $S=1$ (ossia il numero è negativo).
 Pertanto $r_1 = -1.00110 \cdot due^{+1} = -10.0110 = -(2^{+1} + 2^{-2} + 2^{-3}) = -(2 + 0.25 + 0.125) = -2.375$
 Il massimo reale negativo è quello associato al segno positivo ($S_2=1$), al massimo esponente $E_2=1$ ed alla massima mantissa $F_2=1111$.
 La mantissa m_2 associata vale: 1.11111 , mentre l'esponente e_2 associato vale: $11-01 = +2$.
 Pertanto $r_2 = -1.11111 \cdot due^{+2} = -111.111 = 2^2 + 2^1 + 2^0 + 2^{-1} + 2^{-2} + 2^{-3} = -7.875$.

Soluzione Esercizio 14

Esercizio 1

Sia dato il numero reale $r = -13.8125$. Calcolare la sua rappresentazione in virgola mobile su 16 bit. Seguire lo standard IEEE-754 half-precision, ovvero: 1 bit per il segno, 5 bit per l'esponente, 10 bit per la parte frazionaria del significando.

Soluzione esercizio 1

Siccome r è negativo, il bit del segno sarà $s=1$. Calcolo adesso la rappresentazione del valore assoluto di r in base 2.

$|r| = 13.8125 = 13 + 0.8125$. La parte intera (13) e quella frazionaria (0.8125) possono essere convertite in base 2 separatamente, rispettivamente con l'algoritmo "dimezza-e-resto" (DIV-MOD) e l'algoritmo "raddoppia-e-parte intera".

Conversione della parte intera in base 2:

$$13/2 = 6, \text{ resto} = 1$$

$$6/2 = 3, \text{ resto} = 0$$

$$3/2 = 1, \text{ resto} = 1$$

$$1/2 = 0, \text{ resto} = 1 \rightarrow \text{FINE}$$

$$13 = (1101)_2$$

Conversione della parte frazionaria in base 2:

$$0.8125 * 2 = 1.625, \text{ parte intera} = 1$$

$$0.625 * 2 = 1.25, \text{ parte intera} = 1$$

$$0.25 * 2 = 0.5, \text{ parte intera} = 0$$

$$0.5 * 2 = 1, \text{ parte intera} = 1 \rightarrow \text{FINE}$$

$$0.8125 = (0.1101)_2$$

$$\text{Quindi: } 13.8125 = (1101.1101)_2$$

Sposto la virgola di tre posizioni verso sinistra per ottenere il significando nella forma $(1.<F>)_2$.

$$(1101.1101)_2 = (1.1011101)_2 * 2^3$$

La parte frazionaria sarà quindi $F = 1011101000$. Adesso devo rappresentare l'esponente, che vale 3, nella rappresentazione intera con bias E. Quindi devo imporre che $E - \text{bias} = 3$. Il bias vale $2^{k-1}-1$ dove k sono i bit su cui è rappresentato l'esponente. Nel nostro caso $k=5$, quindi $\text{bias}=15$. Se $E-15 = 3$ allora $E = 3+15 = 18$.

Conversione di E in base 2:

$$18/2 = 9, \text{ resto} = 0$$

$$9/2 = 4, \text{ resto} = 1$$

$$4/2 = 2, \text{ resto} = 0$$

$$2/2 = 1, \text{ resto} = 0$$

$$1/2 = 0, \text{ resto} = 1 \rightarrow \text{FINE}$$

$$18 = (10010)_2$$

La rappresentazione di r sarà quindi $R = (1\ 10010\ 1011101000)_2$.