Rappresentazione dell'informazione all'interno di un calcolatore

Gabriele Frassi, 1T Ingegneria informatica

A.A 2019-2020

Indice

Capitolo 1

Introduzione

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:

$$numero\ combinazioni = 2^n$$

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 $\beta \geq$ 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!

$$2^1 = 2 \quad 2^2 = 4 \quad 2^3 = 8 \quad 2^4 = 16$$
$$2^5 = 32 \quad 2^6 = 64 \quad 2^7 = 128 \quad 2^8 = 256$$

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).*

[1] Si legge *ASCHI*

2.1 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

2.2 Codifica ASCII - Esempio: *Caro amico,*

Sequenze	Caratteri
00110000	0
00110001	1
00110010	2
...	...
00111001	9
...	...
01000001	A
01000010	B
01000011	C
...	...
01100001	a
01100010	b
...	...

01000011 01100001 01110010 01101111 00100000 01100001 01101110 01101001 01100011 01101111 00101100

Caro amico,

Capitolo 3

Rappresentazione dei num. naturali $\mathbb{N}$

L'insieme dei numeri naturali, in senso informatico[1], 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:

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

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)

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

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

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 *centodue in base sei*, ma ***uno zero due in base sei***.

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

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
$q_0 = N$
$q_1 = q_0\ DIV\ \beta \qquad a_0 = q_0\ MOD\ \beta$
$q_2 = q_1\ DIV\ \beta \qquad a_1 = q_1\ MOD\ \beta$
$\dots$
$q_{p-1} = q_{p-2}\ DIV\ \beta \quad a_{p-2} = q_{p-2}\ MOD\ \beta$
$q_p = q_{p-1}\ DIV\ \beta \quad a_{p-1} = q_{p-1}\ MOD\ \beta$

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 $N = 38,\ \beta = 5$

$q_0 = 38$
$q_1 = 38\ DIV\ 6 = 6 \qquad a_0 = 38\ MOD\ 6 = 2$
$q_2 = 6\ DIV\ 6 = 1 \qquad a_1 = 6\ MOD\ 6 = 0$
$q_3 = 1\ DIV\ 6 = 0 \qquad a_1 = 1\ MOD\ 6 = 1$
$a_2 a_1 a_0 = 102$

3.3 Potenze di due, basi particolari

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

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

Esempio 1: $2^2 = 4 = (100)_2$ **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 $(\mathbf{100101})_\mathbf{2}$ 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 $(\mathbf{100101})_\mathbf{2} => (\mathbf{45})_\mathbf{8}$, unendo i due numeri!

Attenzione! *Se l'ultima terna (quella più a sinistra) ha meno di tre cifre aggiungete degli zeri. Prendiamo il numero* $(\mathbf{10101})_\mathbf{2}$*:*

$n_1 = \mathbf{0}10$ *[Ho aggiunto lo zero]*
$n_2 = 101$

Il primo, convertito in base dieci, è **2***; il secondo, invece, è* **5***. Quindi* $(\mathbf{100101})_\mathbf{2} => (\mathbf{25})_\mathbf{8}$.

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	
$(\mathbf{N})_\mathbf{2}$ **p = 3**	$(\mathbf{N})_\mathbf{10}$
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.

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

Ottengo anche

$$(2A)_{16} => (0010\ \ 1010)_2 => (\mathbf{0}00\ 101\ \ 010)_2 => (52)_8$$

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	
$(\mathbf{N})_\mathbf{2}$ $\mathbf{p = 4}$	**N**
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	A
1011	B
1100	C
1101	D
1110	E
1111	F

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...

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

Potrò porre *unsigned int* $N = numero$ solo se:

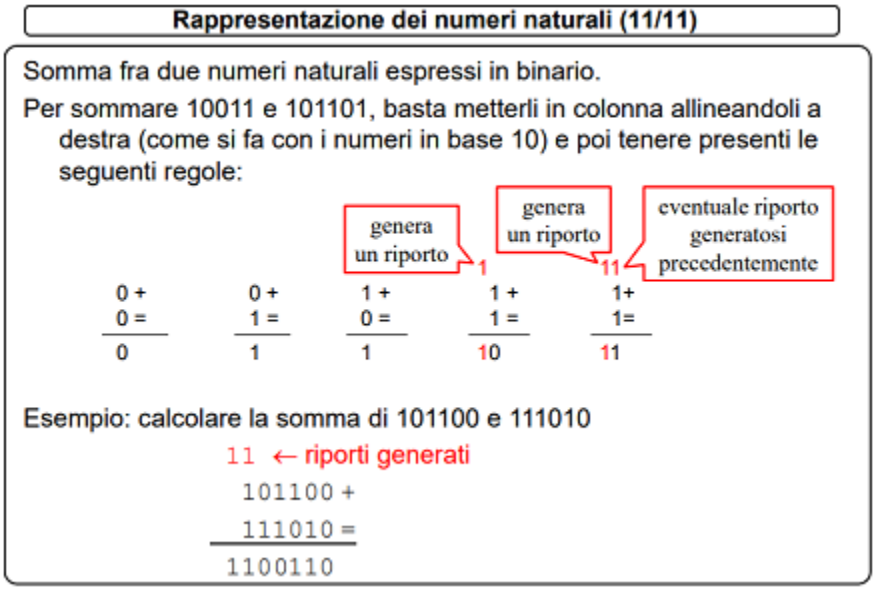
$$0 \leq numero \leq 4294967295$$

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!



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 $a = +5 \quad (p = 4)$
Il valore assoluto (ABS_a) è $5 = (101)_2$
Il segno dell'intero è positivo, quindi $segno\_a = 0$.
$\mathbf{A} = (\mathbf{segno\_a}; \mathbf{ABS\_a}) = (\mathbf{0}, \mathbf{101}) = \mathbf{0101}$

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

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

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

$\mathbf{A} = (\mathbf{segno\_a}; \mathbf{ABS\_a}) = (\mathbf{1}, \mathbf{101}) = \mathbf{1101}$

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 $A = 01000 \quad (p = 5)$

La cifra più significativa è 0, quindi ho segno positivo

Le cifre rimanenti, ABS_a, sono $(1000)_2 = 8$ $\qquad \mathbf{a} = +\mathbf{8}$

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

La cifra più significativa è 1, quindi ho segno negativo

Le cifre rimanenti, ABS_a, sono $(1001)_2 = 9$ $\qquad \mathbf{a} = -\mathbf{9}$

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

$\mathbb{N}$	**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 $p = 4 \quad [-(2^3 - 1); +(2^3 - 1)] \quad [-7; +7]$

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

Attenzione all'overflow Prima di ottenere la rappresentazione di un intero ($a => A$) 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*. $segno\_a = 1 \quad ABS\_a = 9 = (1001)_2$
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 $a = +7 \quad (p = 4)$
L'intero a è maggiore di zero.

$ABS\_a = 7 = (0111)_2$
$A = ABS\_a = (0111)_2$

Esempio 2 $a = -8 \quad (p = 4)$
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 $(0111)_2 \quad (p = 4)$
La cifra più significativa è 0, quindi ho un intero positivo.
Converto A in base decimale, ottengo $\mathbf{a = A = 7}$

Esempio 2 $(1111)_2 \quad (p = 4)$
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: $\mathbf{a = -(2^p - A) = -(2^4 - 15) = -1}$

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

$\mathbb{N}$	**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	−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.

$a = -1 \ \ (p = 3) => A = 111$
$a = -1 \ \ (p = 6) => A = 111111$
$a = -1 \ \ (p = 7) => A = 1111111$

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 necesario verificare che l'intero a sia rappresentabile su p *bit*

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

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** $= \mathbf{2^{p-1} - 1}$.

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:

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

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

Esempio 1 $a = 8 \quad (p = 4)$
$A = 8 + (2^{4-1} - 1) = 15$
15, convertito in base binaria, consiste in $A = (1111)_2$

Esempio 2 $a = -7 \quad (p = 4)$
$A = -7 + (2^{4-1} - 1) = 0$ Ovviamente avrò $A = (0000)_2$

4.3.2 Da rappresentazione a intero

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

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

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

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

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

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:

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

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

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

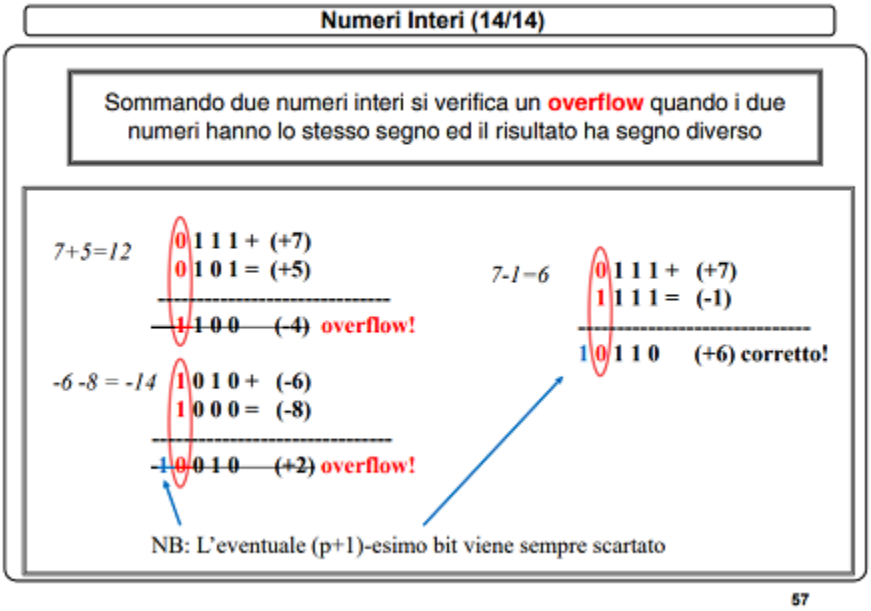
Come sempre attenzione all'overflow Prima di individuare la rappresentazione dell'intero è necessario verificare che l'intero sia effettivamente rappresentabile su p *bit*.

Prendiamo $\mathbf{a = -9}$ con $\mathbf{p = 4}$ **bit**:
ottengo $A = -9 + (2^{4-1} - 1) = -9 + 7 = -2$ che non è numero naturale!

4.4 Tabella di confronto delle rappresentazioni

$\mathbb{N}$	$\{0,1\}^5$	$\mathbf{a^{ms}}$	$\mathbf{a^{c2}}$	$\mathbf{a^{bias}}$
0	10000	+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

4.5 Somma di numeri interi



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 circuteria per la somma e la sottrazione dei numeri naturali, evitandone una aggiuntiva per gli interi!

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:
  $f_{(-1)} = F(f_0 * 2) \quad a_{-1} = I(f_0 * 2)$
  $f_{(-2)} = F(f_{(-1)} * 2) \quad a_{-1} = I(f_{-(1)} * 2)$
  $\ldots$ e così via!
  - 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 $\mathbf{331} = (\mathbf{101001011})_\mathbf{2}$

Troviamo la parte frazionaria $F(r)$

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

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

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

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

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

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'inzio 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: $+110.01$

Esempi in cui esprimo lo stesso numero in virgola mobile[1]:
$+1.11001 * 2^{+11}$
$+111001 * 2^{-10}$

5.2.2 Rappresentazione di r

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

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

- **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 varia il numero di bit a disposizione:

[1] Esponenti espressi in base binaria

- *half-precision*: 16 bit, di cui 5 per l'esponente della mantissa e 23 per la parte frazionaria della stessa ($K = 5$ e $G = 10$).
- *single-precision*: 32 bit, di cui 10 per l'esponente della mantissa e 23 per la parte frazionaria della stessa ($K = 10$ 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 $G = 52$)
- *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 R → 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 $\boxed{r = m * 2^e \neq 0}$

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

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

[2] aggiunto da uno standard più recente

- 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:

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

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 → 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 $10.0\underline{1001}$
- Porto il numero in forma normalizzata
$$1.00\underline{1001}\ 2^{+1}$$

- 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 $R \rightarrow r$:

- $e = E - bias = 31 - (2^4 - 1) = 31 - 15 = 16$
- $f = F\ 2^{-10} = 0.1111111111$
- $|r| = 1.1111111111\ 2^{16} < 2^{\left(2^{K-1}-1\right)+2} = 2^{bias+2}$

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
- $|r| = 1.0000000000\ 2^{-15} = 2^{-\mathbf{bias}}$

Nella rappresentazione half-precision individuo che:

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