3 Rappresentazione dell'Informazione

3.1 Scopo della rappresentazione

trovare un modo opportuno per rappresentare all'interno di un sistema di calcolo le informazioni (dati e programmi) in modo efficiente rispetto a:

- Realtà fisica del sistema
- Loro manipolazione

L'efficacia della rappresentazione non bisogna valutarla in base alla facilità di conversione dell'informazione dal modo in cui è rappresentanta al Plain Text comprensibile dall'uomo poichè è irrilevante il tempo di conversione, l'aspetto importante è la facilità di elaborazione.

3.2 Linguaggi

3.2.1 Confronto linguaggio naturale e linguaggio macchina

- Linguaggio naturale (es: italiano): composto da base decimale per numeri e alfabeto per parole.
- Linguaggio macchina:
  - Alfabeto: insieme dei simboli utilizzati ditinguibili tra loro. // S=...
  - Codice: sequenze e simboli di elementi dell'afabeto che hanno un significato particolare e, insieme, compongono l'insieme delle regole del linguaggio (regole definiscono associazione biunivoca tra parole di codice e relative azioni).
  - Cardinalità dell'alfabeto: numero di elementi totali dell'alfabeto, dato dal valore assoluto di S. // —S—
  - Numero di elementi considerati nel messaggio da rappresentare. // n
- Insieme delle configurazioni k:

$$\mathrm{k} = \left\lceil \log_{|S|} n \right\rceil \ ^{1} \tag{1}$$

- Numero di configurazionei diverse di lunghezza k:

$$\mathrm{n} = |S|^{k} \tag{2}$$

3.2.2 Linguaggio del Calcolatore

Nel caso del calcolatore, noi andiamo a considerare il così detto linguaggio macchina, overo quel linguaggio il cui alfabeto è composto da 0 e 1:

- Condensatore scarico/carico.
- linea di tensione alta/bassa.

Si riduce al Codice Binario, ovvero un linguaggio con alfabeto 0, 1 che ha il bit come cifra di codifica (Binary Digit).

Tabella di conversione bit byte kilobyte megabyte

	bit	byte	Kilobyte	Megabyte	Gigabyte
bit	1				
byte	8	1			
Kilobyte	8,192	1,024	1		
Megabyte	8,388,608	1,048,576	1,024	1	
Gigabyte	8,589,934,592	1,073,741,824	1,048,576	1,024	1
Terabyte	8,796,093,022,208	1,099,511,627,776	1,073,741,824	1,048,576	1,024
Petabyte	9,007,199,254,740,990	1,125,899,906,842,620	1,099,511,627,776	1,073,741,824	1,048,576
Exabyte	9,223,372,036,854,780,000	1,152,921,504,606,850,000	1,125,899,906,842,620	1,099,511,627,776	1,073,741,824
Zettabyte	9,444,732,965,739,290,000,000	1,180,591,620,717,410,000,000	1,152,921,504,606,850,000	1,125,899,906,842,620	1,099,511,627,776

Figure 6: Scala conversione Byte

[1] Il simbolo che assomiglia ad una parentesi quadra indica l'arrontondamento per eccesso all'intero più vicino (es: 2.3 =¿ 3)

3.2.3 Esempio linguaggio carte da gioco

- l'insieme degli elementi da rappresentare: i semi delle carte da gioco (4)
- l'insieme delle configurazioni ammissibili
- le regole del gioco delle carte (detto codice) definiscono l'associazione biunivoca =¿ scelta codice binario
  1. S = 0,1
  2. —S— = 2
  3. dimensione delle configurazioni:

$$\left\lceil \log_{|S|} n \right\rceil = \lceil \log_2 4 \rceil = 2 \tag{3}$$

- l'insieme delle configurazioni ammissibil 00, 01, 10, 11 viene associato ai 4 semi delle carte da gioco picche, fiori, cuori, quandri mediante il codice (associazione biunivoca).

3.2.4 Scelta della codifica

Gli aspetti principali da considerare sono i seguenti:

- Insieme degli elementi da rappresentare
- Adozione di una codifica che semplifichi le operazioni che si svolgono più di frequente
- Adozione di una codifica che "conservi" ove utile le proprietà dell'insieme degli elementi da rappresentare (es: elementi adiacenti abbiano codifiche adiacenti)

3.2.5 Tipologie informazioni da rappresentare

Le tipologie di infromazioni che possiamo gestire e analizzare sono le seguebti illustrate nell'immagine. Nello specifico, nel corso di Fondamenti di Informatica andremo a trattare solamente le i dati e infromazioni delle tipologie presenti all'interno dell'insieme rosso.

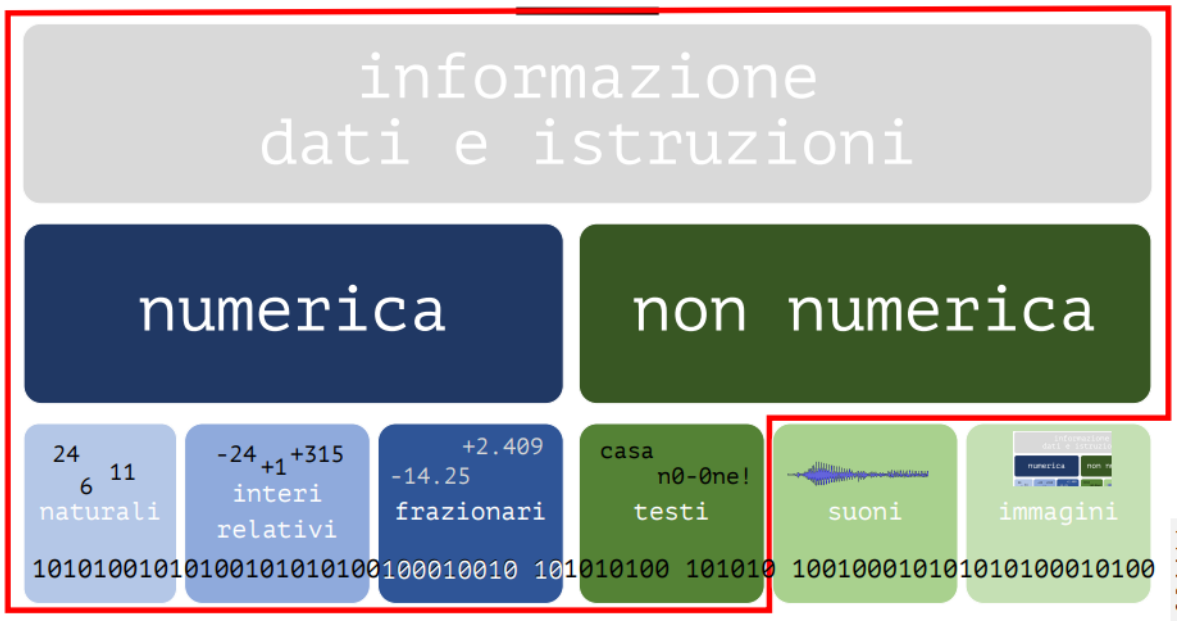

Figure 7: Tipologie di informazioni da rappresentare

- Tipologie trattate: numerico (naturale, intero relativo, frazionario) - non numerico (testi)
- Tipologie non trattate: non numerico (suono, immagine)

3.3 Sistema numerico binario

3.3.1 Definizione

13 Settembre 2022

Definition 3.1. *Il sistema binario, o sistema numerico binario, è un sistema di numerazione posizionale in base 2. A differenza del sistema decimale (in base 10) le uniche cifre che compongono i numeri sono 0 ed 1, e per tale motivo essi vengono detti numeri binari.*

Il sistema binario ha cardinalità 2 poichè l'alfabeto degli elementi dell'insieme è $A = \{0, 1\}$ e ha una notazione posizionale (conta la posizione delle cifre) e pesata (b_i) (a seconda della posizione le cifre hanno un peso diverso).

$$v(N) = \alpha_{n-1}b^{n-1} + \alpha_{n-2}b^{n-2} + \ldots + \alpha_0 b^0 = \sum_{i=0}^{n-1} \alpha_i b^i \tag{4}$$

Nella rappresentazione posizionale, la cifra più a sinistra. Nella rappresentazione posizionale, la cifra più a sinistra prende il nome di cifra più significativa in quanto ha peso massimo, quella più a destra prende il nome di cifra più significativa in quanto ha peso massimo, quella più a destra ($\alpha 0$) di cifra meno significativa, in inglese rispettivamente Most Significant Digit (MSD) e Least Significant Digit (LSD).

3.3.2 Numeri naturali

$$v(N) = \sum_{i=0}^{n} c_i b^i \tag{5}$$

Di seguito riporto la tabella con i numeri decimali e i corrispondenti binari:

Base 10	Base 2
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

Possiamo notare che esiste una corrispondenza biunivoca (definita da un codice) fra numero in base 10 e numero in base 2. Per rappresentare un numero n, in realtà si definiscono tutti i valori $0 \leq n \leq numero$ (formula 1).

Gli zeri a sinistra non sono significativi e un numero binario è pari se il bit meno significativo è 0, mentre se è 1 è dispari.

3.3.3 Metodo di conversione base 2

Per convertire da decimale a binario e viceversa sfruttiamo il fatto che esse siano entrambi delle scale posizionali e pesate, ora mostro i due casi:

- Base 10 $\implies$ Base 2: utilizzando la formula 5 possiamo trovare il valore, il procedimento (metodo) ha validità generale e posssono essere applicate a qualsiasi conversione $A \implies B$, ad esempio nel caso $356_2 = (6 \cdot 2^0 + 5 \cdot 2^1 + 3 \cdot 2^2)$
- Base 2 $\implies$ Base 10: il numero decimale viene scomposto dividendo ripetutamente per 2 come nella seguente tabella, il risulate della conversione bisogna leggerlo dal bit più basso a quello più alto ($MSD \implies LSD$):

:2	
17	1
8	0
4	0
2	0
2	0
1	1

:2	
35	1
17	1
8	0
4	0
2	0
2	0
1	1

La conversione del numero 17_{10} in base 2 è 10001_2.

3.3.4 Numeri relativi

Nella notazione MS (modulo e segno) possiamo rappresentare sia i numeri interi positivi che negativi grazie all'introduzione di un ulteriore bit che indica il segno (0 positivo e 1 negativo) del numero rappresentato, ad esempio la conversione del numero -35_{10MS} corrisponde a:

:2	
35	1
17	1
8	0
4	0
2	0
2	0
1	1

La conversione del numero 34_{10} in base 2 è "100011_2", che in notazione MS è "$0\,100011_2$", noi stavamo convertendo -35_{10MS} quindi dobbiamo farlo diventare negativo: "$1\,100011_{2MS}$".

È importante segnare con l'apice la notazione con cui stiamo esprimendo il numero perchè, ad esempio il numero 1011 potrebbe essere interpretato come 1011_{10}, 1011_2 (11_{10}) e come 1011_{MS} (-2_{MS}). Per quanto riguaqrda il calcolatore, esso interpreta tutto in nella notazione 2C2 perchè ha un'aritmetica estremamente efficiente ed ottimizzata che vedremo in seguito.

3.3.5 Notazione complemento alla base

Il complemento a due, o complemento alla base, è il metodo più diffuso per la rappresentazione dei numeri con segno in informatica, questo perchè l'aritmetica è estemamente ottimizzata (si evita il problema dell'analisi del segno dei numeri nelle somme). In questa notazione se si somma $x_{2C2} + (-x_{2C2}) = 0000$. Ad esempio avendo a disposizione 4 bit, in notazione complemento a 2 si ottiene:

Base 10	Base 2	2C2
+0	0000	+0
+1	0001	+1
+2	0010	+2
+3	0011	+3
+4	0100	+4
+5	0101	+5
+6	0110	+6
+7	0111	+7
-0	1000	-0
-1	1001	-7
-2	1010	-6
-3	1011	-5
-4	1100	-4
-5	1101	-3
-6	1110	-2
-7	1111	-1

Si nota che c'è una rindondanza perchè compare sia -0 che +0, che viene sistemata assegnando al valore in 2C2 di -0 il valore di -8 (per quanto riguarda l'aritmetica interna il valore può essere ± 8 perchè è circolare, ma per convenzione del MSD=1 si considera -8).
Analogamente esiste il complemento alla base 3, è importante sottolineare che, essendo la base dipari, non vale più la regola dello $MSD(0) \implies positivo$ e $MSD(1) = pari$; un altro aspetto particolare delle basi dispari è che, secondo la formula 1 delle configurazioni, esse non hanno l'inefficienza del doppio zero (± 0) che nelle basi pari viene gestito due volte, sia come negativo che positivo.

3.3.6 Somma e differenza in complemento a 2

Per essere sommati due numeri devono avere la stessa configurazione di bit, quindi nel caso uno fosse espresso con meno bit bisogna estenderlo senza farlo cambiare di valore, per fare ciò si ripete il bit più significativo (primo da sinistra).

I valori in complemento a 2 sono calcolati trovando il complementare che sommato ad essi dia 0000, così facendo l'aritmetica è ottimale e rapida in modo da rendere veloce l'elaborazione. Di conseguenza in questa notazione non esiste il concetto di segno e modulo, perchè essi sono già inclusi nel numero; possiamo affermare che se il bit più significativo è 1 il numero sarà negativo, viceversa se il MSD è 0 il numero sarà negativo.

3.3.7 Operazioni aritmetiche

L'aspetto più imposrtante da considerare in un sistema numerico su cui basare un calcolatore è l'ottimizzazione dell'aritmetica in tale base. In base binaria le operazioni svolte possono essere ricondotte e poche operazioni base che non necessitano di molte condizioni e ragionamenti. Di seguito ci somo degli esempi di operazioni in base 2:

- Somma (+): i numeri devono essere delle stesse dimensioni e vengono sommati bit a bit tenendo il riporto nel caso un bit risulti essere maggiore di 1:
- Differenza (-): la differenza è di fatto la somma del numero cambiato di segno (opposto del numero), quindi basta cambiare di segno il secondo termine per ritornare nel caso precedente di somma:

$$\begin{array}{ll} 100111 & + \\ 011100 & = \\ \hline 000011 & \end{array}$$

Chiaramente questo metodo diventa estremamente efficiente quando è molto facile ricavare l'opposto di un numero, ovvero quel numero $x \in \mathbf{Z} \,|\, x + (-x) = 0$, per questo motivo i calcolatori operano con la notaione complemento a 2.

3.3.8 Overflow

Definition 3.2. *Con overflow si intende il superamento della capacità massima di memoria o di operatività di un computer, che è causa di errore.*

L'overflow di per sè, all'interno dell'ambiente di un calcolatore, non è un problema e l'artimetica funziona corretamente come si può vedere nell'immagine ref(fig:overflow) perchè viene trascurato il bit in eccesso. Il problema

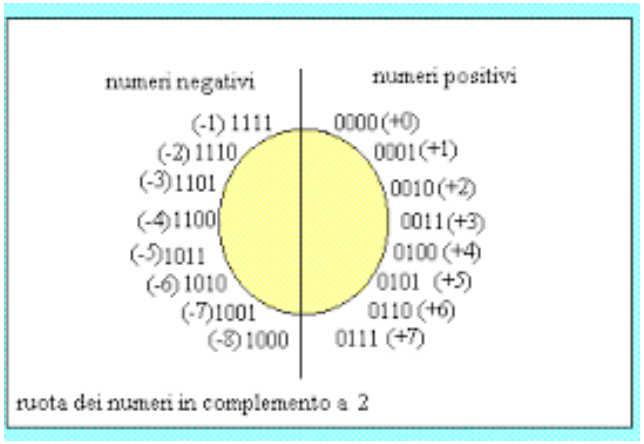

Figure 8: Aritmetica nel calcolatore

nasce quando il calcolatore deve trasmettere delle informazioni all'esterno dove è importante il valore effettivo del dato, quindi il fatto che il bit più significativo vengo troncato influisce sul numero che viene trasmesso nel mondo esterno al calcolatore (nella conversione da 2C2 a base 10). Per stabilire se si è verificato overlow si devono considerare due casistiche, ovvero quando i due elementi sommati sono:

- Concordi: se il risultato è discorde con i due numeri sommati (es: $+2+4=-3$) si è verificato overlow, ovvero quando il risultato non è plausibile .
- Discordi: non si può verificare overflow.

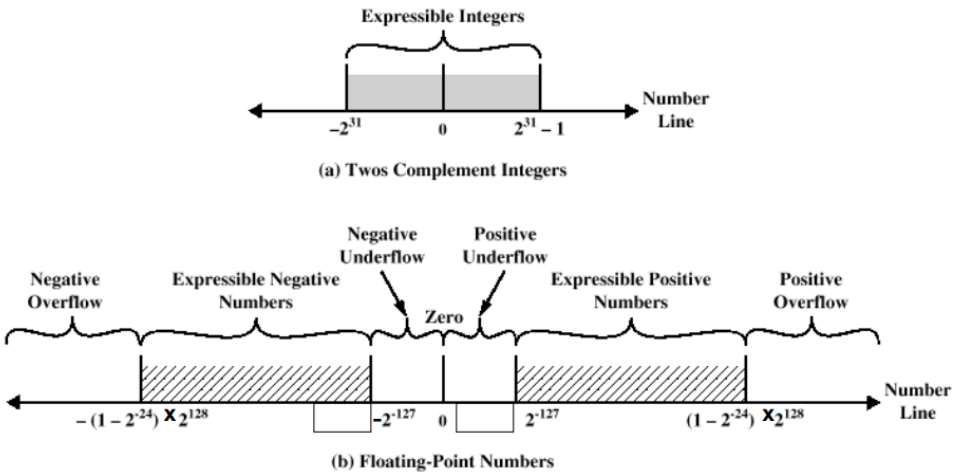

Figure 9: Linea dei numeri ed overflow

3.3.9 Metodo di conversione 2C2

- Notazione $10_{MS} \implies$ 2C2:
  - Positivo ($MSD = 0$):
    1. Applicare il procedimento classico di conversione di un numero in base 2
    2. Scomporre dividendo per 2
    3. Leggere il numero ottenuto nella scomposizione dal basso verso l'alto ($MSD \implies LSD$)
  - Negativo ($MSD = 1$):
    1. Trasfromare il numero negativo x_{10MS} in $-x_{10MS}$, che consiste nel trovare l'opposto (cambio segno)
    2. Applicare il procedimento classico di conversione di un numero in base 2
    3. Scomporre dividendo per 2
    4. Leggere il numero ottenuto nella scomposizione dal basso verso l'alto ($MSD \implies LSD$)
    5. Ricordarsi di cambiere segno nel risultato finale perchè noi abbiamo lavorato con $-x_{10MS}$ e noi dovevamo convertire x_{10MS}
- Notazione 2C2 $\implies 10_{MS}$:
  - Positivo ($MSD = 0$):
    1. Moltiplicare per 2 elevato alla posizione di ogni cifra secondo la formula 5
    2. Sommare i risultati parziali e si ottiene il numero in notazione 10_{MS}
  - Negativo ($MSD = 1$):
    1. Trasfromare il numero negativo x_{2c2} in $-x_{2c2}$, che consiste nel trovare il complementare
    2. Moltiplicare per 2 elevato alla posizione di ogni cifra secondo la formula 5
    3. Sommare i risultati parziali e si ottiene il numero in notazione 10_{MS}
    4. Ricordarsi di cambiere segno nel risultato finale perchè noi abbiamo lavorato con $-x_{2c2}$ e noi dovevamo convertire x_{2c2}

3.3.10 Rappresentazione valori numerici razionali

Considerato un numero razionale $\frac{m}{n} \quad , \quad n \in \mathbf{N} - \{o\}$, vale la stessa regola dei numeri interi in base 2, ad esempio:

$$101.01_2 = 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 + 0 \cdot 2^{-1} + 1 \cdot 2^{-2} = 4 + 0 + 1 + 0 + \frac{1}{4} = 5,25 \tag{6}$$

Nella scrittura del decimale si deve definire quanti bit vanno a rappresentare la parte intera e quanti quella decimale. Esisteono due tipologie di notazioni per rappresentare i numeri razionali:

- Notazione con virgola fissa: in questa notazione vengono stabiliti e fissati il numero di bit detinati alla rappreswnazione della aprte intera e a quella decimale, ad esempio in 101101_2 possiamo decidere arbitrariamente che $\underbrace{10}_{\text{intero}}\ \underbrace{1101}_{\text{decimale}}$, che si sicrive 10.1101.

  Per la formula delle configurazioni 1 sappiamo che se, ad esempio, destiniamo 2 bit abbaimo 4 configurazioni disponibili quindi fra un numero intero n e il suo successivo $n+1$ ci sono $4-1$ numeri rappresentabili.

  Porprio per questo, a differenza degli interi che o il numero è corretto o si è verificato overflow, i numeri razionali possono o essere rappresentati correttamente o il numero viene approssimato (troncato) alla configurazione vicina (precisione vincolata al numero di bit, ovvero alle configurazioni).

  Nella notazione a virgola fissa l'errore assoluto rimane costante, mentre è variabile l'errore relativo come si può vedere nella formula $\varepsilon_R = \frac{\varepsilon_A}{valore}$.

- Notazione con virgola mobile: in questa notazione viene dedicato il numero minimo di bit alla parte intera del numero da rappresentare e il resto dei bit discponibili viene riservata alla parte decimale.

  Così facendo la firgola viene spostata, da qui il nome virgola mobile (floating point), dove è ottimale e si sfruttano tutti i bit disponibili per avere una precisione della parte decimale massima. La virgola mobile la ritroveremo in programmazione quando utilizzeremo le variabili di tipo $float$.

  Nella notazione a virgola mobile l'errore relativo rimane costante, mentre è variabile l'errore assoluto.

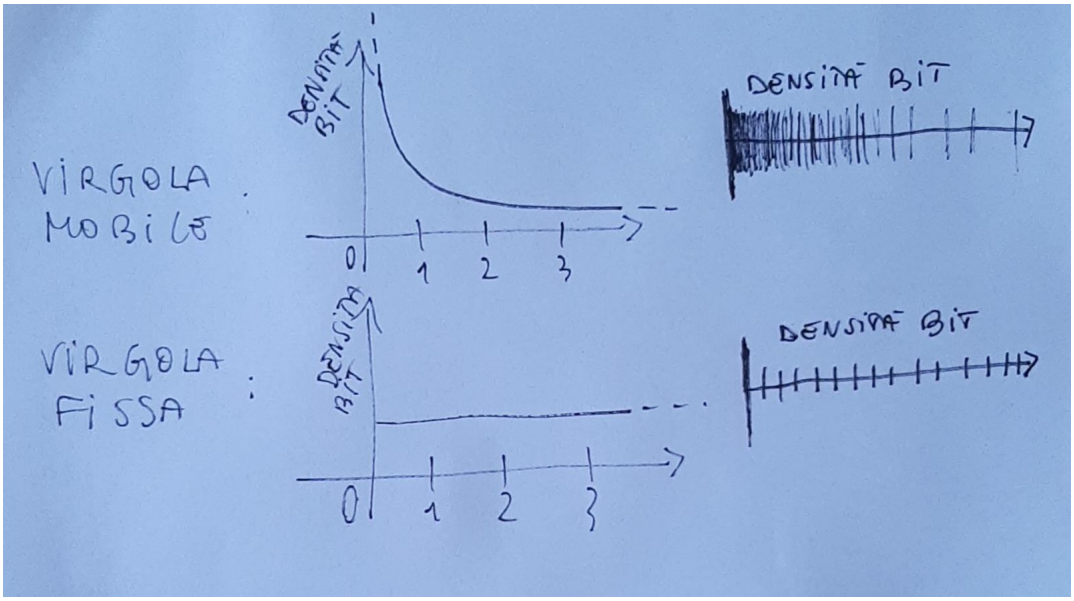

Figure 10: Distribuzione bit in virgola fissa (sopra) e mobile (sotto)

Come per i numeri interi, anche per i numeri razionali valgono le stesse regole di converisone che si basano sulla formula 5, un esempio di esercizio potrebbe essere trasformare il numero $13.75_{\{10\}}$ in base 2 comolemento a 2:

:2	
13	1
6	0
3	1
1	1

x2	
0.75	/
0.75*2=1.50	1
0.50*2=1.00	1
0.00	0

Nella tabella a sinistra si calcola il corrispondente della parte intera dividendo ripetutamente la base (2) e considerando il risultato dal basso verso l'alto ($MSD \implies LSD$), nell'esempio il risultato della parte intera è 1101.

Nella tabella a destra si calcola il corrispondente della parte decimale:

1. si moltiplica per la base, in questo caso 2
2. il risultato ottenuto se è maggiore o uguale a 1 corrisponde a 1, viceversa si scrive zero
3. si considera solo la parte decimale e si ripete il procedimento dal punto 1

4. una volta arrivati a 0.00, il risultato va considerato leggendolo dall'alto verso il basso ($MSD \implies LSD$), nell'esempio il risultato della parte decimale è 110.

Sia nel caso di virgola fissa che mobile si può utilizzare la notazione Modulo Segno (MS), dove 0 identifica un numero positovo e 1 negativo.

3.3.11 Notazione scientifica

I numeri binari possono essere scitti anche in notazione scientifica, questa notazione è utilizzata perchè è utile per riuscire a scrivere dei numeri che richiedono l'impiego di molti bit in maniera più ottimizzata. Ad esempio:

$$13.75_{10} = 1.375 \cdot 10^1 = 1101.110_2 = 1.101110 \cdot 2^3 \tag{7}$$

Il numero in notazione scientifica può essere scritto come:

$$1.M \cdot base^{exp} \tag{8}$$

In questo caso stiamo considerando la base 2, la parte decimale è rapresentata dalla mantissa e infine l'1 è implicito e non necessita di essere salvato sprecando bit. L'esponente viene ritoccato per renderlo sempre positivo, gli viene sommato il numero di configurazioni negative diposnibili meno uno in base ai bit a disposizione in modo da evitare che si presenti un numero negativo ad esponente, ad esempio se ci sono 8 bit, allora la somma sarà di $2^8 - 1 = 255$.

Segno	Esponente	Mantissa

Lo Standard della notazione scientifica nei calcolatori è IEEE754 e definisce come vengono storate le informazioni numeriche binarie in notazione scientifca. Esistono due tipologie di notazioni in base alla precisione necessaria:

- Singola precisione:

+/−	Exp	Mantissa
1	8	23

32 bit totali

- Doppia precisione:

+/−	Exp	Mantissa
1	11	52

64 bit totali

Lo standard IEEE 754 della notazione scientifica attribuisce valori convenzionali a particolari configurazioni di esponente e mantissa:

Normalizzato	±	0<esp<Max	Qualsiasi stringa di bit
Denormalizzato	±	0	Qualsiasi stringa di bit diversa da zero
Zero	±	0	0
Infinito	±	111…1	0
NaN	±	111…1	Qualsiasi stringa di bit diversa da zero

Figure 11: Valori convenzionali notazione scientifica

3.3.12 Sistema numerico esadecimale

Nell'informatica è importante il sistema numerico in base 16 (esadecimale) perchè semplifica estremamente la conversione avendo la base come potenza della base 2 ($16 = 2^4$) le conversioni vengono svolte di 4bit in 4bit e la lettura per l'utente è facilitata (comunicazione tra utenti di informazioni del compilatore).

Ad esempio, per convertire il numero 0101010111011010_2 in base 16 basta:

$$\underbrace{0101}_{5}\underbrace{0010}_{3}\underbrace{1101}_{D}\underbrace{1010}_{A} \tag{9}$$

Ottennedo facilmente e rapidamente il numero convertito in base 16, viceversa si può risalire al numero in base 2 partendo dal numero in base 16.

Decimal	Binary	Hexadecimal
0	0	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	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

Figure 12: Tabella di conversione sitema decimale, binario ed esadecimale

3.4 Rappresentazione informazioni non-numeriche

L'obiettivo è definire una codifica per ogni possibile carattere rappresentabile, quindi è necessario introdurre, oltre alla codifica per i numeri, una codifica per i caratteri testuali.

3.4.1 Alfabeto caratteri

Inizialmente i caratteri da rappresentare erano:

- Alfabeto base a b ... z A B ... Z
- Caratteri numerici 0 1 ... 9
- Simboli interpunzione . , : ...
- Caratteri "speciali" a-capo nuova-riga spazio

In totale questi caratteri sono 120, di conseguenza per la formula delle configurazionei 1 i bit necessari sono:

$$\left\lceil \log_{|S|} n \right\rceil = \lceil \log_2 120 \rceil = 7 \tag{10}$$

3.4.2 Codice ASCII

Per rappresentare l'informazione non numerica uno dei codici più comunemente utilizzati è il codice ASCII (American Standard Code for Information Interchange), basato sul sistema binario e che consta di parole di codice di 7 bit. Il codice ASCII è non ridondante, perchè i simboli codificati sono in numero pari alle configurazioni ottenibili con 7 cifre binarie.

1. 0-32: caratteri non stampabili come simboli di controllo e funzioni varie
2. 48-57: cifre numeriche per base 10
3. 65-90: caratteri alfabeto maiuscoli
4. 97-123: caratteri alfabeto minuscoli

Poichè i caratteri sono però ben più numerosi di 127 anche nel nostro alfabeto, in realtà si utilizza il codice ASCII esteso, che utilizza 8 bit, avendo quindi a disposizione 256 diverse configurazioni. Questa parte della codifica però presenta varie versioni a carattere nazionale, quindi ci sono diverse codifiche per l'intervallo 128-255.