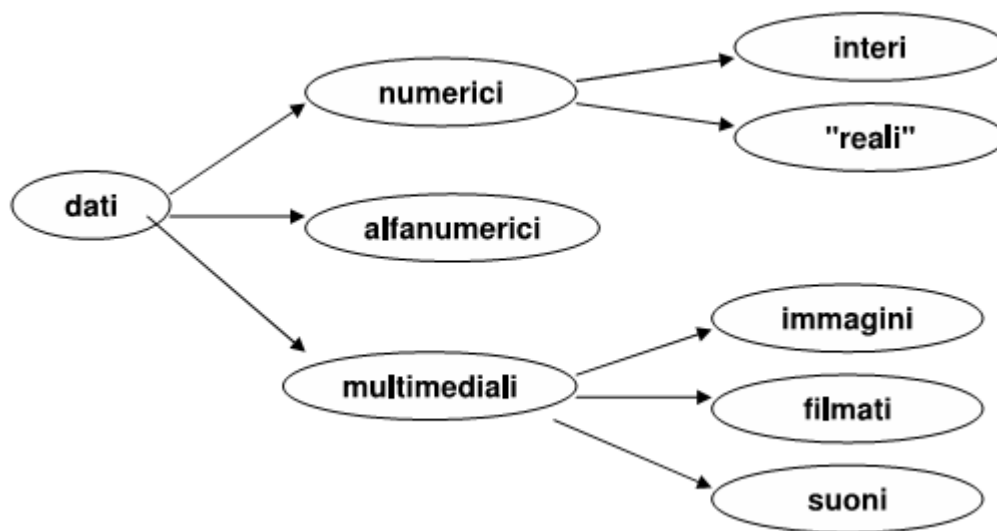


Rappresentazione delle informazioni all'interno di un elaboratore

I calcolatori digitali sono sistemi in grado di elaborare e archiviare nelle loro memorie esclusivamente grandezze binarie. La causa di ciò è da ricercarsi nell'hardware, per cui è molto facile dal punto di vista tecnologico costruire oggetti (componenti elettronici, magnetici o ottici) che possono assumere soltanto due stati distinti, a cui associamo le costanti binarie 0 e 1.

Nella nostra attività, noi siamo invece abituati a utilizzare simboli diversi da sequenze di bit: per comunicare con un elaboratore è quindi necessario stabilire un insieme di regole (un codice) in grado di rappresentare informazioni a noi abituali, quali le lettere dell'alfabeto oppure i numeri, impiegando soltanto variabili binarie.

Lo schema seguente presenta le principali informazioni che devono essere rappresentate mediante codici binari.



Nella comunicazione uomo-macchina possiamo riconoscere:

- un alfabeto esterno, contenente informazioni elementari (numeri o caratteri alfanumerici);
- l'alfabeto della macchina (o interno) che è quello binario;
- un codice, inteso come l'insieme delle regole che pongono in corrispondenza biunivoca gli elementi dell'alfabeto della macchina con le informazioni esterne.

In genere, si parla di codifica quando passiamo dalle informazioni comprensibili all'uomo all'alfabeto binario della macchina, mentre con decodifica si intende la traduzione inversa dal linguaggio comprensibile al calcolatore a quello dell'uomo. In un sistema di elaborazione, il compito della codifica è svolto dalle unità di ingresso, mentre quello della decodifica dalle unità di output.

Bit, byte, word

L'unità di misura "naturale" dell'informazione per l'elaboratore è il **bit**, ovvero una cifra o una variabile binaria.

Poiché con un solo bit è possibile rappresentare solo due elementi (associandoli ai due stati 0 e 1) dell'alfabeto esterno, per rappresentare un numero maggiore di informazioni si devono necessariamente impiegare codici binari costruiti con sequenze (ordinate) di bit. Il termine ordinato indica che, se si scambiano i valori (0 e 1) nella sequenza, si ottiene nella decodifica una informazione diversa.

Una sequenza di n bit, con n numero intero, può assumere 2^n configurazioni diverse e quindi rappresentare altrettante informazioni. Essendo il singolo bit un'unità troppo piccola per rappresentare l'informazione, si introducono i concetti illustrati nella tabella seguente:

semibyte o nibble	sequenza ordinata di 4 bit.
byte o ottetto (octet)	sequenza ordinata di 8 bit.
parola (word)	sequenza ordinata di un numero intero di byte, il cui valore dipende dal tipo di CPU e di memoria centrale del calcolatore. In genere, con il termine word si intendono 2, 4 oppure 8.

Nelle applicazioni, l'unità di misura bit è rappresentata con la lettera **b**, mentre il byte con **B**. Con un semibyte, ad esempio, si possono rappresentare 16 (2^4 bit) simboli elementari diversi dell'alfabeto esterno, mentre con 1 byte si arriva fino a 256 (2^8 bit) informazioni diverse comprensibili per l'uomo.

Attualmente, i sistemi di elaborazione devono gestire enormi quantità di informazioni codificate in binario. La memoria utilizzata per codificare una pagina di testo è di qualche migliaio di byte, quella usata per una immagine può raggiungere il milione di byte, mentre un lungo filmato può richiedere miliardi di byte per essere memorizzato.

Si avverte la necessità, come nel sistema metrico decimale, di utilizzare dei simboli per rappresentare i multipli delle grandezze elementari; nella terminologia informatica sono stati quindi adottati gli stessi simboli del sistema decimale, ma visto che la misurazione della memoria ha come sua base principale il 2, il loro significato è leggermente diverso.

La seguente tabella riassume i simboli e i valori dei multipli più usati:

Multiplo	Sigla	Valore
Kilo	k	$2^{10} = 1024$
Mega	M	$2^{20} = 1024^2 = 1024 \text{ k}$
Giga	G	$2^{30} = 1024^3 = 1024 \text{ M}$
Tera	T	$2^{40} = 1024^4 = 1024 \text{ G}$

Rappresentazione interna dei numeri

Nella codifica dell'informazione numerica si distinguono tre casi:

- numeri interi senza segno,
- numeri relativi,
- numeri reali.

All'interno dell'elaboratore, i numeri sono codificati mediante un numero limitato di bit, in modo da essere memorizzati nelle celle della memoria centrale, che contengono, ad esempio, 8, 16, 32, 64 o 128 bit. Con un numero limitato di bit è possibile codificare solo un sottoinsieme dei numeri interi o reali; per tale motivo, i numeri rappresentabili all'interno di un calcolatore prendono il nome di **numeri macchina**.

Numeri interi senza segno

Per rappresentare numeri interi senza segno, il calcolatore usa il sistema binario puro. Con n bit si può rappresentare il sottoinsieme dei numeri interi che va **da 0 a $2^n - 1$** ; con un byte, ad esempio, si possono rappresentare 256 numeri interi che vanno da 0 a 255.

Numeri interi relativi

Per rappresentare i numeri con il loro segno (interi positivi e negativi) è necessario codificare due tipi di informazioni: il segno ed il valore assoluto del numero (modulo).

Le principali tecniche di rappresentazione impiegate sono:

- modulo e segno,
- complemento a due,
- eccesso-k.

Rappresentazione in modulo e segno

Dati n bit, si riserva un bit al segno e gli altri $n - 1$ sono destinati al numero; utilizzando n bit e riservandone uno al segno, l'applicazione della formula precedente porterà **da $-(2^{n-1} - 1)$ a $2^{n-1} - 1$** ; i possibili valori nel caso di 16 bit saranno quindi compresi tra -32.767 e +32.767, mentre con 4 bit saranno quindi compresi tra -7 e +7:

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

Anche se semplice, possiede però un grosso difetto: esistono due zeri.

Esempio. Interpretare la sequenza di bit 1001 0001 1100 0101 come numero naturale e numero relativo in modulo e segno.

a) Interpretando la sequenza come numero naturale:

$$1001\ 0001\ 1100\ 0101 = 91C5_{16} = 37317$$

b) Interpretando la sequenza come numero relativo in modulo e segno:

$$1\ 001\ 0001\ 1100\ 0101 = -11C5_{16} = -4549$$

Rappresentazione in complemento a due

Tale rappresentazione segue le regole:

- se il numero è positivo, viene codificato in modulo e segno (bit di segno 0);
- se il numero è negativo, è codificato mediante il suo complemento a due.

Con questa convenzione i numeri negativi acquisiscono automaticamente il bit di segno uguale a 1.

Con una parola di n bit si può rappresentare il sottoinsieme dei numeri interi nell'intervallo che va da -2^{n-1} fino a $+(2^{n-1}-1)$, dove lo zero è codificato in modo unico (tutti i bit uguali a 0); ad esempio i numeri rappresentabili con 8 bit vanno da -128 a $+127$, mentre i numeri rappresentabili con 4 bit vanno da -8 a $+7$:

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

Esempio. Si rappresentino i seguenti numeri in complemento a 2 avendo 8 bit a disposizione:

$$\begin{array}{ll} -34 \rightarrow 1101\ 1110 & +72 \rightarrow 0100\ 1000 \\ -25 \rightarrow 1110\ 0111 & +46 \rightarrow 0010\ 1110 \\ -23 \rightarrow 1110\ 1001 & +66 \rightarrow 0100\ 0010 \\ -120 \rightarrow 1000\ 1000 & -10 \rightarrow 1111\ 0110 \end{array}$$

Codici in eccesso

La rappresentazione in eccesso- k si basa sulla regola: per codificare in eccesso k un numero intero, si somma al numero il valore k e si converte il valore ottenuto in binario su n bit.

Viceversa, data una sequenza di bit in eccesso- k , per decodificarla si deve prima convertire il numero in decimale ed in seguito sottrargli il valore k .

Con questa tecnica è possibile rappresentare, con n bit, tutti i numeri interi con segno da $-k$ fino a $2^{n-1} - k$.

In generale, il valore di k non è casuale, ma è legato al numero di bit usati nel

codice dalla relazione: $k=2^{n-1}$ oppure $k=2^{n-1} - 1$; questa posizione consente infatti di rappresentare con n bit un numero circa uguale di numeri positivi e negativi. Ad esempio, se il numero di bit è 4, con il codice in eccesso-8 ($k=2^{4-1}$) possiamo rappresentare i numeri interi relativi da -8 (-2^{4-1}) a +7 ($2^4 - 1 - 8$):

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

Il codice eccesso-k, rispetto agli altri, usa una convenzione per il bit di segno opposta: 0 per i numeri negativi e 1 per quelli positivi.

Numeri reali

Consideriamo l'insieme R dei numeri reali; ogni elemento di R, in generale, può essere espresso come somma di un intero con un numero frazionario.

In pratica, un numero reale è individuato univocamente da:

- una parte intera I,
- una parte frazionaria F.

Rappresentazione in virgola fissa

Nella rappresentazione in virgola fissa consiste un numero prefissato di cifre viene dedicato alla parte intera ed a quella frazionaria (rappresentazione in virgola fissa).

In un numero rappresentato in virgola fissa con n bit, viene utilizzato un bit per il segno (MSB), k bit per rappresentare la parte intera e m bit per rappresentare la parte decimale (ovviamente sarà $n = k + m + 1$).

Rappresentazione in virgola mobile

Un numero reale r, in base B, può essere sempre scritto in forma esponenziale o scientifica $r = m \cdot B^e$ dove m è la **mantissa** del numero reale mentre e è l'esponente (o **caratteristica**),

Il numero reale -45,012 scritto nella forma precedente diventa $-0,45012 \cdot 10^{+2}$ dove la mantissa vale 0,45012 mentre l'esponente è pari a +2.

Nella forma esponenziale, mentre la mantissa contiene tutte le cifre "significative" di un numero, l'esponente ci dice quanto un numero è "grande o piccolo", ovvero l'ordine di grandezza. Ricordando che il prodotto per la base equivale a spostare la virgola di una posizione a destra mentre il quoziente di una a sinistra, l'esponente indica "la posizione" della virgola nel numero reale.

La forma esponenziale non è unica; ad esempio, il numero 15,47 può essere rappresentato, in forma esponenziale, nei seguenti modi equivalenti:

$$15,47 ==> 0,1547 \cdot 10^2 \quad 15,47 ==> 154,7 \cdot 10^1$$

Per rendere unica la forma precedente per un numero reale, in base B, si deve

scegliere il valore dell'esponente (spostare la virgola) in modo che il valore assoluto (numero senza segno) della mantissa verifichi la relazione seguente:

$$\frac{1}{B} \leq |m| < 1$$

L'unica notazione per cui vale la condizione precedente si definisce forma esponenziale normalizzata oppure notazione in virgola mobile (**floating point**).

Il termine "virgola mobile" ricorda che nella forma esponenziale dobbiamo "spostare" la virgola finché non otteniamo la forma normalizzata. Il termine "point" (punto) deriva dal fatto che nella notazione anglosassone dei numeri, così come in quella impiegata nei calcolatori, la virgola è sostituita dal punto radice.

Per scrivere un numero reale in base 10 in forma normalizzata si deve spostare la virgola a destra (esponente decresce) o a sinistra (esponente cresce) finché la mantissa (in valore assoluto) non è minore di 1 e contemporaneamente maggiore o uguale a 0,1 (1/10)

Ad esempio, il numero reale 1.074.234,0 scritto in virgola mobile è uguale a 0,1074234 10⁷. Per scrivere il numero binario 1010,11 (in decimale 10,75) in virgola mobile si deve spostare la virgola a sinistra di quattro posizioni (prodotto per 2) ottenendo 0,101011 2⁴.

Schematizzando, fissata la base, per rappresentare un numero reale in virgola mobile sono necessarie tre informazioni:

- il segno del numero;
- il valore assoluto della mantissa, che, considerando soltanto le cifre dopo la virgola, è sempre un numero intero positivo;
- l'esponente, che è un numero intero dotato di segno.

La rappresentazione in virgola mobile interna in un calcolatore usa le seguenti regole:

1. si converte il numero reale in binario o in esadecimale;
2. si scrive il numero ottenuto nella forma esponenziale normalizzata;
3. si rappresenta il numero in binario riservando:
 - o **1** bit per il segno (0 se positivo, 1 se negativo);
 - o **M** bit per il valore assoluto della mantissa;
 - o **E** bit per l'esponente (intero relativo) rappresentato, in genere, con un codice in eccesso.

Segno	Esponente	Mantissa
1 bit	E bit	M bit

A seconda dell'hardware e dell'azienda costruttrice del calcolatore, sono state introdotte diverse codifiche interne, che si differenziano per la base utilizzata (2 oppure 16) e per il numero di bit riservati per la mantissa e per l'esponente.

Le due notazioni in virgola mobile più diffuse, proposte da uno dei principali organismi che si interessano di standard, l'IEEE (Institute of Electrical and Electronics Engineers), e che utilizzano la base 2, sono quelle in singola e doppia precisione che si differenziano per il numero di bit riservati per rappresentare la mantissa (precisione del numero) e l'esponente (ordine di grandezza).

Singola Precisione (Single o Float Precision)

Segno	Esponente	Mantissa
1 bit	8 bit	23 bit

32 bit – 4 byte

L'esponente è codificato con eccesso-127 ($2^{8-1} - 1$).

Doppia Precisione (Double Precision)

Segno	Esponente	Mantissa
1 bit	11 bit	52 bit

64 bit – 8 byte

L'esponente è codificato con eccesso-1023 ($2^{11-1} - 1$).

Nelle applicazioni, a fianco delle notazioni precedenti, è frequente trovare altre rappresentazioni che differiscono per il numero di bit riservati all'esponente e/o alla mantissa.

Il numero zero, non possedendo una rappresentazione normalizzata, è frequentemente codificato dal numero binario che ha sia la mantissa sia l'esponente uguali a zero.

Esempio. Si converta il numero $N=7,5$ nella rappresentazione binaria in single precision. Si procede nel seguente modo:

1. si converte la parte intera $7 = 111$
2. si considera la parte frazionaria $0,5 = 1$
3. si considera il numero binario ottenuto convertendo la parte intera e la parte frazionaria: $111,1$
4. Si normalizza il numero binario ottenuto al passo precedente: $0,1111 \times 2^3$
5. la mantissa vale 11110000000000000000000
6. l'esponente vale $3+127=130$, la cui codifica su 8 bit è 10000010
7. il segno vale 0
8. il numero convertito è quindi **01000001011100000000000000000000**

Esercizio. Si converta il numero $N=3758,125$ nella rappresentazione binaria IEEE 754 in virgola mobile singola.

Esercizio. Si converta il numero $N=-23552,25$ nella rappresentazione binaria IEEE

754 in virgola mobile singola.

Il problema dell'Overflow

L'insieme dei numeri in floating point rappresentabili è limitato ed è possibile che un numero sia troppo grande o troppo piccolo per essere rappresentato.

Quando il risultato di un calcolo è troppo grande per essere rappresentato in un sistema di numeri in floating point, diciamo che è avvenuto un **overflow**.

Se il risultato di un'operazione aritmetica genera un valore minore in modulo del più piccolo numero positivo rappresentabile, si verifica un **underflow**.

L'underflow dà come risultato dell'operazione il valore zero.