Gli standard servono a far sì che dispositivi di produttori diversi possano interagire. Lo Standard IEEE 754 prevede due rappresentazioni possibili per i numeri razionali. Una rappresentazione su 32 bit e una su 64 bit. Il primo bit (quello più a sinistra) è di segno e usa il metodo di modulo e segno (se è 0 il numero è positivo, se è 1 è negativo). La formula che viene usata per decodificare il valore è quindi v = (-1)^s \*2^exp \* M. 1 bit è dedicato al segno, 8 bit sono dedicati all’esponente, 23 bit alla mantissa. Il valore 1 si può ottenere mettendo 0 nel bit di segno, 1 nella mantissa, 0 nell’esponente. Ci sono però modi alternativi di rappresentare questo valore (ad esempio con una mantissa di valore 0.5 e un esponente di 1, poiché 0.5 \* 2^1 = 1). Per evitare di avere una rappresentazione multipla degli stessi valori (non canonica ?) viene introdotto il concetto di normalizzazione. Esso consiste nel mettere un vincolo ulteriore: il valore M deve sempre (quando possibile) essere >= 1 (ciò permette di rendere unica la rappresentazione di tutti i valori) e si posiziona la virgola subito dopo il primo bit della mantissa (quindi è 1.00..00), che quindi non può essere > 2. Poiché il primo bit (appena prima della virgola) è già scontato che sia 1, non viene neanche scritto nella rappresentazione normalizzata (e quindi il primo bit è già 2^-1 anziché 2^0). La formula vera e propria per la decodifica è quindi: v = (-1)^s \* 2^e \* (M), dove M è in realtà = 1 + M.

Molti valori sono normalizzabili(per esempio 1), ci sono però valori non normalizzabili (come lo 0). Inserendo un opportuno codice all’interno degli 8 bit dell’esponente però si passa alla forma non normalizzata. L’esponente può infatti assumere valori sia positivi che negativi (ha quindi rappresentazione con segno). La rappresentazione dell’esponente è però unica: è infatti in eccesso alla 2^(k)-1. Normalmente prevederebbe, secondo una rappresentazione a eccesso normale, una rappresentazione a eccesso 128: essa è invece a eccesso 127 (in cui si prende il valore che si vuole codificare e lo si rappresenta in binario dopo averci sommato 127). Gli esponenti possibili vanno quindi da -126 a 127. Il -127 è un valore “mancante” che è comunque rappresentabile (con tutti 0) e viene usato per dire che la rappresentazione non è più normalizzata (quindi il valore delle unità non viene più assunto essere 1, ma diventa 0). Ciò permette di scrivere valori compresi tra 0 e 1 particolarmente piccoli. A questo punto infatti la rappresentazione (non normalizzata) viene considerata in virgola fissa con 2^-127 come costante moltiplicativa (la formula diventa quindi V: (-1)^s \* 2^-127 \* (M) dove qui M = M, inizia con 0. E arriva fino a 2^-23 + lo 0).

Nella rappresentazione normalizzata il numero più grande possibile è circa 2^128 (in teoria 2^127 \* 1.999999 ???). Lo 0 è quindi particolare nella rappresentazione floating point IEEE 754 poiché è rappresentabile scrivendo 32 0. Ciò rende particolarmente comodo il verificare se un valore è 0.

Un altro valore rappresentabile che manca dal range dell’esponenti è il 128 (tutti 1). Ciò permette la rappresentazione di valori particolari che normalmente non sarebbero rappresentabili come numeri: ossia +infinito, -infinito e NaN. Ciò permette di codificare operazioni non consentite su un sistema di calcolo (come la divisione per 0). Gli infiniti vengono codificati con il bit di segno (che indica il + o il -), il 128 all’esponente e tutti 0 alla mantissa.

NaN è una sigla che sta per Not a Number e viene tipicamente generato con la divisione 0/0. È quindi un altro tipo di errore. Viene codificato con 128 all’esponente e con un valore di mantissa diverso da 0 (il bit di segno perde importanza). Ci sono quindi tantissime configurazioni possibili per NaN.

Poiché lo Standard IEEE 754 vede il segno rappresentato con modulo e segno la rappresentazione non è esattamente canonica (poiché -0 è comunque rappresentabile, con 1 al bit di segno e a tutti gli altri 0).

La versione a 64 bit dello Standard IEEE 754 è del tutto simile nella rappresentazione. L’unica differenza è che 1 bit è dedicato al segno, 11 bit all’esponente e 52 bit alla mantissa. Il modo di codifica è del tutto analogo; cambia l’eccesso che, avendo 11 bit di esponente, diventa a 1023. Ciò permette all’esponente di assumere valori tra 1023 e -1022. I valori “particolari” (tutti 0 e tutti 1) sono quindi -1023 e 1024.

La rappresentazione a 64 bit viene chiamata double, mentre quella a 32 bit viene chiamata float.

Le codifiche standardizzate (quella trattata in particolare) fan sì che venga ridotto leggermente il numero di possibili numeri rappresentabili (per esempio la rappresentazione dei double a 64 bit standardizzata NON permette la rappresentazione di 2^64 numeri diversi).

Per capire quanto sia efficiente l’utilizzo di una certa codifica bisogna introdurre il concetto di ridondanza. Se si prende in considerazione una rappresentazione di un codice a lunghezza fissa (per esempio, a 32 bit), il numero di combinazioni possibili è dato dalla lunghezza scelta (nell’esempio proposto è 2^32). Se si prendono ad esempio 7 bit di rappresentazione (tabella ASCII) ci sarebbero 2^7 combinazioni possibili. Ma quando si va a elencare tutti i tipi di dato che Voglio rappresentare non è detto che io necessiti di occupare tutte le combinazioni (tipicamente è difficile trovare una situazione in cui il numero di combinazioni necessarie sia uguale a quelle possibili, un esempio di ciò è la tabella ASCII, che contiene 90 caratteri pur potendo arrivare fino a 128. Nota: essendo 90 > 64 non era possibile usare 6 bit). Si introduce quindi Il numero di ridondanza, definito come il rapporto tra il numero di combinazioni possibili e il numero di combinazioni usate (è quindi sempre maggiore o uguale a 1). Più grande è il numero di ridondanza e meno efficiente è il nostro codice (per quanto riguarda l’utilizzo delle risorse a disposizione).

In genere, si tende ad utilizzare dei codici con coefficiente di ridondanza compreso tra 1 e 2 (strettamente: 1 <= R < 2): esso è un codice minimale.

La rappresentazione di NaN è una rappresentazione ridondante (poiché tutte le codifiche della mantissa diverse da tutti 0 lo possono rappresentare). Nei float le rappresentazioni di Not a Number sono (2^23 -1)\*2 = 2^24 – 2 (circa 2 milioni di combinazioni diverse tutte col significato di NaN). Il coefficiente di ridondanza per la parte di codice che rappresenta +infinito, - infinito e NaN è quindi (2^24) / 3. Questa inefficienza corrisponde però a una facilità di uso del dato rappresentato (poiché basta guardare l’esponente per capire se è un numero o no e se è un numero se è normalizzato o no).

Una delle cose che si possono osservare, introducendo questo tipo di analisi, è che un codice non può mai soddisfare tutti i requisiti possibili immaginabili.

Nota: la rappresentazione della mantissa è detta esplicita.