Rappresentazione dell’informazione

RAPPRESENTAZIONE DELL'INFORMAZIONE

- Per poter rappresentare le informazioni è necessario **codificare** le informazioni per poterne garantire l'affidabilità.
- I **simboli** per la codifica possono essere rappresentati da:
  - una tensione elettrica {Vhigh (alta), Vlow (bassa)};
  - un differente stato di polarizzazione magnetica (positiva, negativa);
  - luce e buio.
- I valori ad essi associati sono 0 e 1 e vengono detti bit (binary digit); tutti i dati (numeri, parole, in generale oggetti) sono rappresentati da sequenze di **bit**.
- Per cui ogni informazione viene rappresentata nel calcolatore in forma binaria.

Sistemi di numerazione

Si chiama **sistema di numerazione** l'insieme di un numero finito di **simboli** e delle **regole** che assegnano uno ed un solo significato ad ogni scrittura formata coi simboli stessi

- I **simboli** di un sistema di numerazione prendono il nome di **cifre**

- Es. sistemi di numerazione basati su un principio puramente additivo:

  MLXII

- **LIMITI:** richiede sempre nuovi simboli e la rappresentazione dei numeri di grandi dimensioni è complessa

Sistemi di numerazione posizionali

- I moderni sistemi di numerazione sono **posizionali** cioè tutti i simboli (o cifre) vengono ordinati in modo che ognuno abbia un valore di un'unità più alto di quello della cifra precedente e quando più cifre vengono combinate insieme il valore del numero rappresentato dipende anche dalle loro posizioni relative.

- Il numero delle cifre di cui si avvale un sistema di numerazione prende il nome di **base**
- In un numero espresso con un sistema di numerazione posizonale
  - la cifra all'estrema destra ha il valore minore (cifra meno significativa)
  - la cifra all'estrema sinistra il valore maggiore (cifra più significativa)
- La scelta della base in un sistema di numerazione è determinata da ragioni di comodo.
- Nel caso di sistemi di numerazione con base dieci parleremo di numero decimale

Sistemi si numerazione posizionali

SISTEMA	BASE	CIFRE
decimale	10	0 1 2 3 4 5 6 7 8 9
binario	2	CIFRE
ottale	8	0 1 2 3 4 5 6 7
esadecimale	16	0 1 2 3 4 5 6 7 8 9 A B C D E F

Rappresentazione polinomiale

3847

è equivalente a

$$3 \times 10^3 + 8 \times 10^2 + 4 \times 10^1 + 7 \times 10^0$$

che è un polinomio ordinato delle potenze del 10 (base).

Tale rappresentazione viene detta polinomiale

Spostando una cifra di una posizione verso sinistra si moltiplica il suo valore per 10

Il significato di questa scrittura è

3 migliaia + 8 centinaia + 4 decine + 7 unità

Rappresentazione polinomiale

La rappresentazione polinomiale è valida anche per la parte frazionaria di un numero

In questo caso ogni cifra alla destra della virgola viene moltiplicata per una potenza negativa del 10 a partire dalla cifra più vicina alla virgola

Esempio

$$307{,}251 =$$

$$3 \times 10^{2} + 0 \times 10^{1} + 7 \times 10^{0} + 2 \times 10^{-1} + 5 \times 10^{-2} + 1 \times 10^{-3}$$

cioè

$$300 + 0 + 7 + 0{,}2 + 0{,}05 + 0{,}001$$

Sistema di numerazione binario

La base 2 è quella teoricamente più piccola per un sistema di numerazione

Le cifre utilizzate sono 0 e 1 e si indicano con il nome di bit (abbreviazione di binary digit)

IL valore posizionale di ogni cifra è legato alle potenze di 2

1 1 0 0 1 1 1

103

Sistema di numerazione binario

Per la rappresentazione polinomiale di un numero frazionario vengono utilizzate le potenze negative

1 1 0 , 0 1 1 1

= 4 + 2 + 0,25 + 0,125 + 0.0625 = 6,4375

Conversione da decimale a binario

- Numeri interi $\Rightarrow$ metodo delle divisioni successive
- Numeri frazioniari $\Rightarrow$
  - si usa il metodo delle divisioni successive per la parte a sinistra della virgola (parte intera)
  - si usa il metodo delle moltiplicazioni successive per la parte a destra della virgola (parte frazionaria)

Metodo delle divisioni successive

- Si divide il numero N da convertire per la nuova base 2; sia Q il quoziente ed R il resto.
- Si converte il R nella corrispondente cifra della nuova base
- Si aggiunge la cifra cosi' ottenuta a sinistra delle cifre ottenute in precedenza.
- Se Q=0, fine; Altrimenti poni N = Q e torna al passo 1.
- Esempio: Conversione del numero 23 in base 2

23:2 = 11 con resto 1
11:2 = 5 con resto 1
5:2 = 2 con resto 1
2:2 = 1 con resto 0
1:2 = 0 con resto 1

$23_{10} = 10111_2$

Metodo delle moltiplicazioni successive

- Con tale metodo si prende come cifra binaria la parte intera e si moltiplica per 2 la parte decimale fino a quando la parte frazionaria è diventata nulla o quando si sia trovato un numero sufficiente di cifre binarie.

- Conversione del numero 0.65625 in base 2

0.65625*2 = 1.31250 parte intera 1
0.31250*2 = 0.62500 parte intera 0
0.62500*2 = 1.250 parte intera 1
0.250*2 = 0.500 parte intera 0
0.5*2 = 1.00 parte intera 1

$0.65625_{10} = 0{,}10101_2$

Metodo delle moltiplicazioni successive (2)

0.6*2 = 1.2 parte intera 1
0.2*2 = 0.4 parte intera 0
0.4*2 = 0.8 parte intera 0
0.8*2 = 1.6 parte intera 1
0.6*2 = 1.2 parte intera 1

$0.6_{10} = 0.1001_2$

0.6352*2 = 1.2704 parte intera 1
0.2704*2 = 0.5408 parte intera 0
0.5408*2 = 1.0816 parte intera 1
0.0816*2 = 0,1632 parte intera 0
0.1632*2 = 0.3264 parte intera 0
0.3264*2 = 0,6528 parte intera 0
0.6528*2 = 1.3056 parte intera 1

$0.6_{10} = 0.1010001_2$

Esempio

14.23

Conversione parte intera
14:2=7 resto 0
7:2 = 3 resto 1
3:2= 1 resto 1
1:2 = 0 resto 1

Conversione parte frazionaria
0.23*2=1,46 parte intera 1
0.46*2 = 0.92 parte intera 0
0.92*2= 1.84 parte intera 1
0.84*2 = 1.68 parte intera 1
0.68*2= 1.36 parte intera 1
0.36*2 = 0.72 parte intera 0

$14.23_{10} = 1110.101110_2$

Operazioni aritmetiche

Regole per la somma

0 + 0 = 0

0 +1 = 1

1 + 0 = 1

1 + 1 = 0 con il riporto di 1

```
  11001 +
 100100 =
 111101
```

```
  11001 +
  10010 =
 101011
```

Regole per la sottrazione

0 - 0 = 0

1 - 1 = 0

1 - 0 = 1

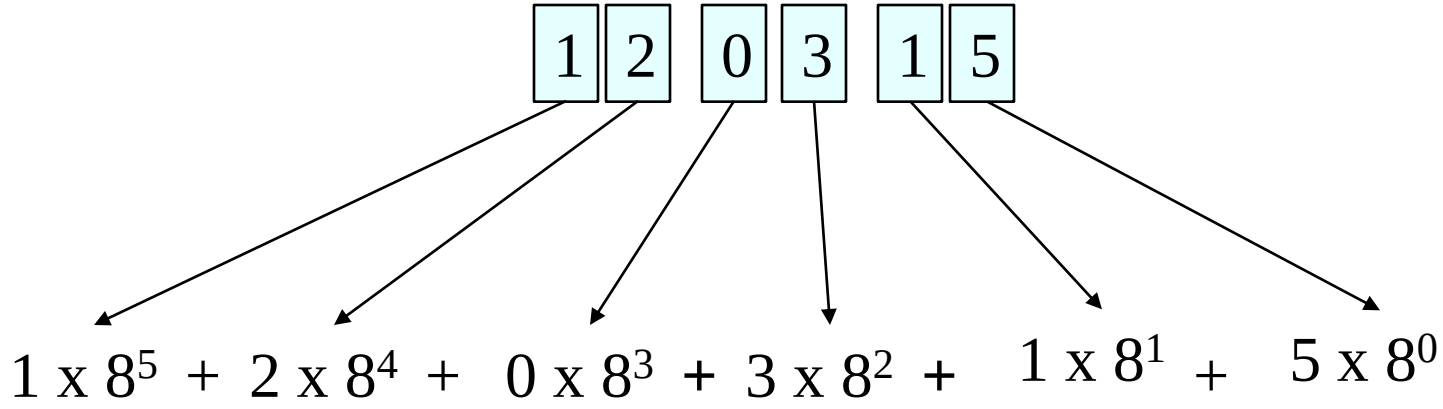
0 - 1 = 1 prestito di 1

```
 111101 -
 100100 =
  11001
```

```
 101011 -
  10010 =
  11001
```

Sistema di numerazione ottale

Le cifre utilizzate sono 0 1 2 3 4 5 6 e 7 Il valore posizionale di ogni cifra è legato alle potenze di 8



= 1 * 32768 + 2 * 4096 +0 * 512+ 3 * 64 + 1 * 8 + 5 * 1=

= 32768 + 8192 +192 + 8 + 5 = 41165

Conversione da decimale a ottale

- Numeri interi $\Rightarrow$ metodo delle divisioni successive
- Numeri frazioniari $\Rightarrow$
  - si usa il metodo delle divisioni successive per la parte a sinistra della virgola (parte intera)
  - si usa il metodo delle moltiplicazioni successive per la parte a destra della virgola (parte frazionaria)

Metodo delle divisioni successive

- Si divide il numero N da convertire per la nuova base 8; sia Q il quoziente ed R il resto.
- Si converte il R nella corrispondente cifra della nuova base
- Si aggiunge la cifra cosi' ottenuta a sinistra delle cifre ottenute in precedenza.
- Se Q=0, fine; Altrimenti poni N = Q e torna al passo 1.
- Esempio: Conversione del numero 123 in base 8

123:8 = 15 con resto 5
15:8 = 1 con resto 7
1:8 = 0 con resto 1

$123_{10} = 175_{8}$

Metodo delle moltiplicazioni successive

- Con tale metodo si prende come cifra binaria la parte intera e si moltiplica per 8 la parte decimale fino a quando la parte frazionaria è diventata nulla o quando si sia trovato un numero sufficiente di cifre binarie.

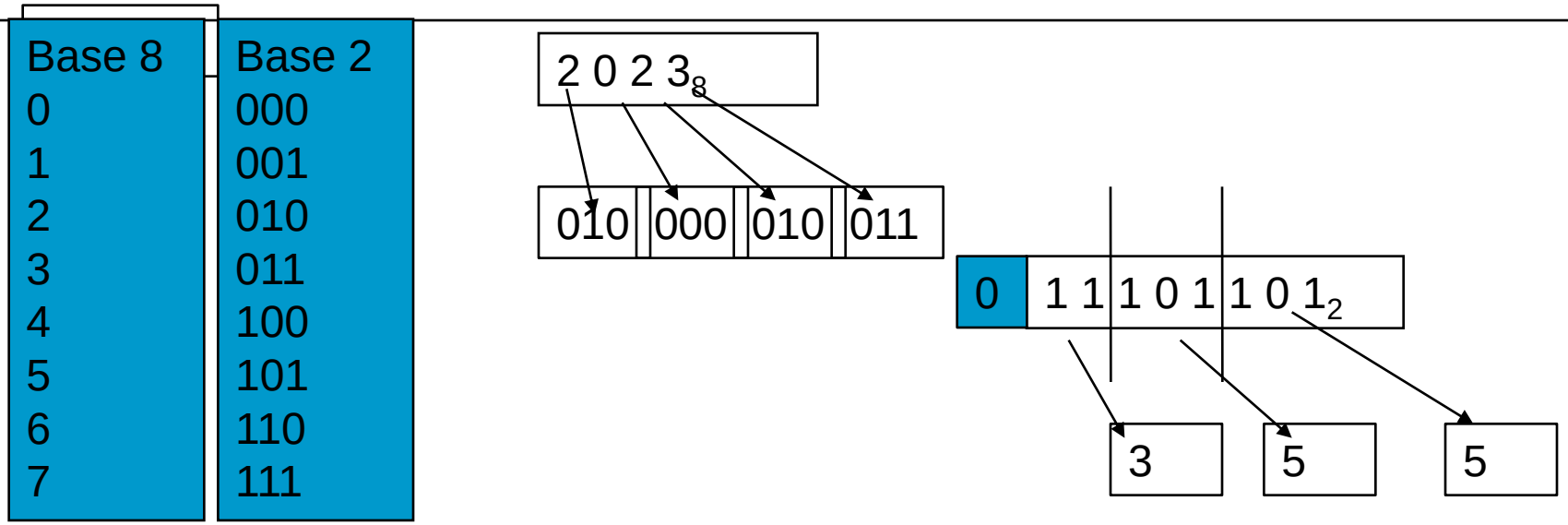
- Conversione del numero 0.65625 in base 8

0.65625*2 = 5.25 parte intera 5
0.25*8 = 2.0 parte intera 2

$0.65625_{10} = 0{,}52_{8}$

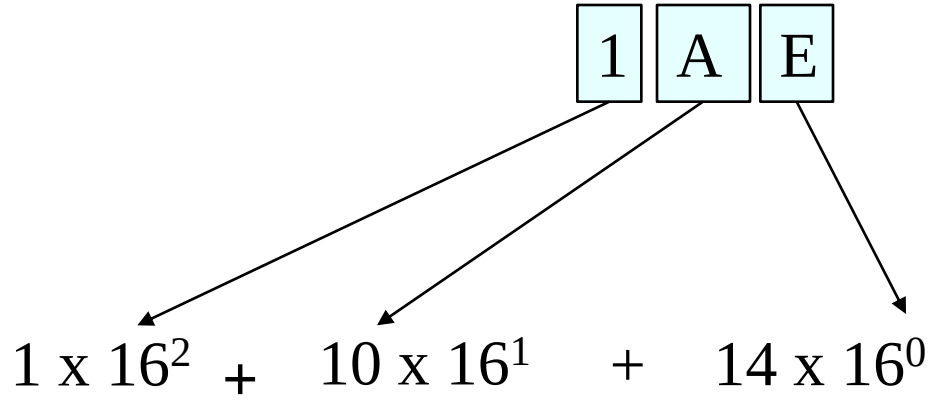
Conversione da binario ⇔ ottale

- Ogni elemento del numero in base 8 viene sostituito dal corrispondente valore in base 2 espresso con tre cifre.



Sistema di numerazione esadecimale

Le cifre utilizzate sono 0 1 2 3 4 5 6 7 8 9 A B C D E F Il valore posizionale di ogni cifra è legato alle potenze di 16



= 1 * 256 + 10 * 16 +14 * 1=

= 256 + 160 +14 = 430

Conversione da decimale a esadecimale

- Numeri interi $\Rightarrow$ metodo delle divisioni successive
- Numeri frazioniari $\Rightarrow$
  - si usa il metodo delle divisioni successive per la parte a sinistra della virgola (parte intera)
  - si usa il metodo delle moltiplicazioni successive per la parte a destra della virgola (parte frazionaria)

Metodo delle divisioni successive

- Si divide il numero N da convertire per la nuova base 16; sia Q il quoziente ed R il resto.
- Si converte il R nella corrispondente cifra della nuova base
- Si aggiunge la cifra cosi' ottenuta a sinistra delle cifre ottenute in precedenza.
- Se Q=0, fine; Altrimenti poni N = Q e torna al passo 1.
- Esempio: Conversione del numero 2453 in base 16

2453:16 = 153 con resto 5
153:16 = 9 con resto 9
9:16 = 0 con resto 9

$2453_{10} = 995_{16}$

Metodo delle moltiplicazioni successive

- Con tale metodo si prende come cifra binaria la parte intera e si moltiplica per 8 la parte decimale fino a quando la parte frazionaria è diventata nulla o quando si sia trovato un numero sufficiente di cifre binarie.

- Conversione del numero 0.65625 in base 16

0.65625*2 = 10.5 parte intera 10
0.5*16 = 8.0 parte intera 8

$0.65625_{10} = 0,A8_{16}$

Conversione da binario ⇔ esadecimale

- Ogni elemento del numero in base 16 viene sostituito dal corrispondente valore in base 2 espresso con quattro cifre.

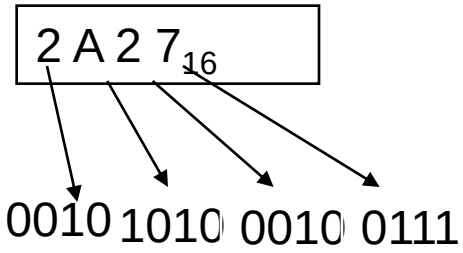
Base 16							
0	1	2	3	4	5	6	7
Base 2							
0000	0001	0010	0011	0100	0101	0110	0111

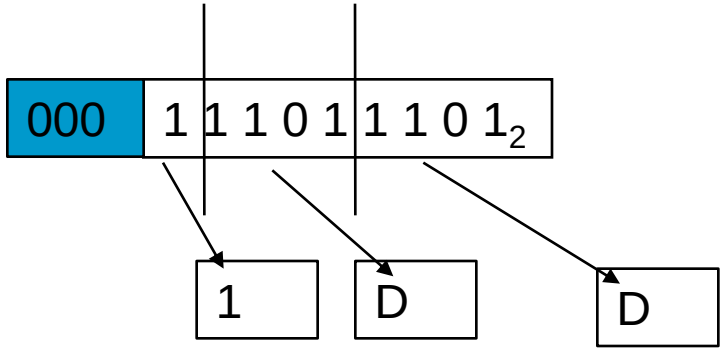
Base 16							
8	9	A	B	C	D	E	F
Base 2							
1000	1001	1010	1011	1100	1101	1110	1111

Conversione da binario ⇔ esadecimale

Base 16							
0	1	2	3	4	5	6	7
Base 2							
0000	0001	0010	0011	0100	0101	0110	0111

Base 16							
8	9	A	B	C	D	E	F
Base 2							
1000	1001	1010	1011	1100	1101	1110	1111





Numeri reali

- I numeri interi relativi possono essere rappresentati in due modi:
  - ***modulo e segno***
  - ***complemento a due.***

Modulo e segno

- Vengono rappresentati in modo analogo a quanto fatto per i numeri interi positivi, riservando un bit (tra gli N disponibili) per rappresentare il segno.
  - Generalmente viene scelto il bit più significativo per rappresentare il segno.
    - Se tale bit vale 1 allora il segno rappresentato vale –
    - se il bit è 0 allora il segno vale +.
- Date N cifre potro' rappresentare i numeri nell'intervallo

  – $(2^{N-1}-1)$ $+(2^{N-1}-1)$

Complemento a due

- Dato un numero binario di N bit, il complemento a 2 di tale numero si ottiene tramite il seguente algoritmo:
  - si procede dal bit meno significativo verso quello più significativo
  - se si incontrano tutti bit 0, essi vengono lasciati inalterati
  - se si incontra il primo bit 1, anche esso viene lasciato inalterato
  - tutti i bit successivi al primo bit 1, vengono invertiti (0 diviene 1, e viceversa)

Esempio

- **Esempio1**: si determini il complemento a 2 del numero 10100.
  - Tutti i bit 0 a partire dal bit meno significativo sono lasciati inalterati e così anche il primo bit 1.
  - Tutti gli altri bit vengono invertiti, ottenendo: 01100.
- **Esempio**: si determini il complemento a 2 del numero 01101001.
  - In questo caso non esistono bit 0 a partire dal bit meno significativo. Sono il primo bit 1 viene lasciato inalterato. Gli altri vengono invertiti, ottenendo: 10010111.

Rappresentazione in complemento a due di un numero relativo

- La rappresentazione in complemento a 2 di un numero intero relativo, si effettua nella seguente maniera:
  - i numeri interi positivi sono rappresentati in modulo e segno (con il bit di segno = 0)
  - i numeri interi negativi sono rappresentati realizzando il complemento a 2 del numero intero in valore assoluto

Esempi

- **Esempio:** si voglia convertire il numero 105 con 8 bit
  - Essendo il numero positivo, la sua rappresentazione in complemento a 2 coincide con quella in modulo e segno, pari a 01101001
- **Esempio:** si voglia convertire il numero -105 con 8 bit
  - Essendo il numero negativo, la sua rappresentazione in complemento a due si ottiene determinando il complemento a 2 del numero binario corrispondente al valore assoluto (105), ossia di 01101001. Il complemento a 2 è 10010111, che è la rappresentazione di -105.

Perchè la Rappresentazione in Complemento a 2 è più Conveniente ?

- La rappresentazione in complemento a 2 viene utilizzata maggiormente rispetto quella modulo e segno per differenti motivi.
  - Il più rilevante è relativo ai vantaggi ottenibili nell'esecuzione di operazioni elementari come la somma e la sottrazione. Queste due operazioni sono quelle che vengono più frequentemente realizzate in un computer, e, dunque, un risparmio nel tempo necessario alla loro esecuzione comporta un indiscusso aumento delle prestazioni di un computer.

Addizioni e Sottrazioni tra numeri binari in Rappresentazione Modulo e Segno.

- Dati due numeri binari in rappresentazione modulo e segno, l'operazione di addizione può avvenire solo se i due segni sono gli stessi, ossia il bit più significativo è per entrambi i numeri uguale a 0 o a 1. In tal caso la somma tra i due numeri con stesso segno viene realizzata con le regole dell'addizione applicate a tutti i bit tranne il bit di segno. Il numero binario risultante sarà ottenuto aggiungendo il bit di segno ai bit ottenuti dalla somma.
- Dati due numeri binari in rappresentazione modulo e segno, l'operazione di sottrazione può avvenire solo se i due segni sono diversi. In tal caso la sottrazione tra i due numeri viene realizzata con le regole della sottrazione applicate a tutti i bit tranne il bit di segno, sottraendo il numero più piccolo in valore assoluto al numero più grande in valore assoluto. Il numero binario risultante sarà ottenuto aggiungendo ai bit ottenuti dalla sottrazione il bit di segno del numero in valore assoluto più grande.

Addizioni e Sottrazioni tra numeri binari in Rappresentazione in Complemento a Due.

- Dati due numeri binari in complemento a due, sia l'operazione di addizione che quella di sottrazione avviene semplicemente applicando le regole dell'addizione a tutti i bit compreso il bit di segno.

Esempi

- **Esempio:** Siano dati i numeri a 4 bit 0010 (+2) e 1010 (-6). Si voglia determinare la sottrazione.

  A tal fine basta applicare semplicemente le regole dell'addizione ai bit:

  0010+
  1010=

  1100 (-4)

  – Il numero binario risultante è già il risultato con il segno giusto.

- **Esempio:** Siano dati i numeri in complemento a 2 a 6 bit 001100 (+12) e 100000 (-32). Si voglia determinare la sottrazione.

  A tal fine basta applicare semplicemente le regole dell'addizione ai bit:

  001100+
  100000=

  101100 (-20)

  – Il numero binario risultante è già il risultato son il segno giusto.

Riconoscimento Automatico dell'Overflow del Segno e dello Zero

- In molti casi è necessario che il calcolatore capisca subito se una operazione di somma in complemento a due abbia fornito come risultato un numero zero, positivo o negativo. E' chiaro che il calcolatore potrebbe ottenere queste informazioni effettuando un'operazione di confronto tra il risultato ottenuto e lo zero. Ma è ovvio, altresì, che ciò comporterebbe una grossa perdita di tempo.
- Inoltre ci sono casi in cui il calcolatore deve capire se il risultato che è stato ottenuto sia valido o meno. Un risultato non valido si ottiene quando si sommano due numeri tali da produrre un numero più grande del massimo numero codificabile con il numero di bit disponibili. In tal caso il risultato che è stato ottenuto deve essere immediatamente riconosciuto come errato dal calcolatore e dunque scartato.

Overflow

- Si supponga di lavorare a 4 bit (il massimo numero positivo codificabile in complemento a due è +7, mentre il massimo numero negativo codificabile in complemento a due è –8)
- Si consideri la seguente operazione di somma in complemento a due:

  si sommino i numeri 1001 (-7) e 1111 (-1).
- E’ chiaro che la somma è -8, ossia il limite inferiore codificabile con 4 bit, tramite la codifica in complemento a due. Eseguendo la somma, il calcolatore ottiene:

$$\begin{array}{r} 1001+ \\ 1111= \\ \hline 1\ \ 1000 \end{array}$$

- dove l'1 a sinistra, ottenuto come resto, viene perso per superamento della capacità dei registri.
- Il numero 1000 in complemento a 2 significa –8, che rappresenta il risultato corretto.

Flag

- Al fine di poter comprendere immediatamente se il risultato di un'operazione di somma in complemento a due è errata, e, nel caso in cui il risultato sia corretto, se il risultato sia positivo, negativo o uguale a zero, il calcolatore utilizza tre particolari flag:
  - Overflow Flag (OF),
  - Sign Flag (SF)
  - Zero Flag (ZF).
- Il valore assunto da tali flag alla fine dell'operazione di somma viene stabilito con le seguenti regole:
  - Siano X e Y i due numeri in complemento a due da sommare. Sia S il risultato ottenuto, alla fine dell'operazione di somma i valori dei flag sono:
    - OF=1, ossia il risultato S non è valido, se i bit più significativi di X e Y sono uguali e il bit più significativo di S è diverso da essi.
    - SF=1, ossia il risultato che deve essere valido è negativo, se il bit più significativo di S è uguale a 1
    - ZF=1, ossia il risultato che deve essere valido è zero, se S = 0.

Numeri Reali

- I numeri reali possono essere rappresentati con due modalità:
  - virgola fissa
  - virgola mobile.

Virgola fissa

- La rappresentazione in virgola fissa consiste nel rappresentare un numero reale con segno tramite N bit, supponendo fissa la posizione della virgola.
- In un numero rappresentato in virgola fissa a N bits, viene utilizzato
  - un bit per il segno
  - I bits per rappresentare la parte intera
  - D bits per rappresentare la parte decimale
- (ovviamente sarà N = I + D +1).

Virgola fissa

1 1110111 1111011

N = 16

I = 8

D = 7

$$N_{10} = (-1)^s \cdot \left(\sum_{i=0}^{I-1} a_i \cdot 2^i + \sum_{d=-1}^{-D} b_d \cdot 2^d \right)$$

Rappresentazione in Virgola Mobile.

- In un numero rappresentato in virgola mobile vengono stabiliti un certo numero di bit assegnati per codificare il segno (s), la mantissa (m) ed un certo numero di bit per codificare l'esponente (e).
- A differenza degli interi, non esistono convenzioni adottate universalmente.

Esempio

- A titolo di esempio si consideri la seguente rappresentazione. Il bit più significativo è il bit del segno (s), mentre l'esponente è stato rappresentato con 7 bits (e) e la mantissa con i restanti 24 bits (m).

  s (31 bit)e(30...24 bit)m(23...0 bit)

- Il numero occupa in tutto 32 bit.
- La mantissa è rappresentata in modulo (bits 0-23) e segno (bit 31).
- L'esponente è rappresentato in 7 bits in eccesso 64 cioè, il vero esponente si ottiene dalla sua rappresentazione sottraendo 64. Ad esempio:
  - 0000000 corrisponde ad un esponente -64 (perchè 0-64=-64)
  - 1111111 corrisponde ad un esponente +63 (perchè 127-64=63)

virgola fissa $\rightarrow$ Numero reale

- Un numero in Virgola Mobile è rappresentato in decimale secondo la formula:

$$N_{10} = (-1)^{s} \cdot m \cdot 10^{e}$$

Numero reale → virgola fissa

- Si procede nel seguente modo:
  - si normalizza il numero in modo che la parte intera sia 0. La normalizzazione si ottiene moltiplicando o dividendo il numero per la minima potenza di 10, in modo che la parte intera ottenuta dalla divisione sia 0. La potenza di 10 così trovata rappresenta l'esponente della codifica binaria (e). In base al fatto che si è operata una moltiplicazione o una divisione, l'esponente può essere positivo o negativo.
  - si converte il numero ottenuto (quello con la parte intera pari a 0) utilizzando la rappresentazione a Virgola Fissa in cui solo 1 bit è utilizzato per codificare la parte intera mentre tutti gli altri sono utilizzati per codificare la parte razionale.
  - Si codifica in binario l'esponente trovato al passo 1, ricordando che l'esponente è in eccesso 64.
  - Si mettono assieme segno, esponente e mantissa trovati ai passi precedenti.

Esempio

- Si converta il numero N=18.75 nella rappresentazione binaria in virgola mobile. Si procede nel seguente modo:
  - si normalizza il numero in modo che la parte intera si 0. Nel nostro caso occorre spostare la virgola di 2 posizioni verso sinistra, ossia dividere per 100 (102). Quindi si ottiene:

$$N=0.1875 \quad e=+2$$

- si converte il numero N ottenuto al passo 1 utilizzando la rappresentazione a Virgola Fissa in cui solo 1 bit è utilizzato per codificare la parte intera mentre tutti gli altri sono utilizzati per codificare la parte razionale.
- Nel nostro caso: il numero binario in virgola fissa è 0011.
- Si codifica in binario l'esponente. Ricordando che l'esponente è in eccesso 64 deve essere: e - 64 = 2 , cioè e=66. La sua rappresentazione in binario su 7 bits è 1000010
- In definitiva si ha: 18.75 = 01000010000011000000.....000

Lo standard 754 dell'IEEE

- Nell'anno 1985 l'IEEE (The Institute of Electrical and Electronics Engineering) ha definito uno standard per la codifica dei numeri reali in virgola mobile, che verrà illustrato nel seguito, facendo riferimento ad una codifica a 32 bit.
- Vengono assegnati:
  - 1 bit per il segno (il bit più significativo), s
  - 8 bit per l'esponente, e
  - 23 bit per la mantissa, m
- La rappresentazione IEEE 754 prevede due diverse forme: quella normalizzata e quella denormalizzata.

Lo standard 754 dell'IEEE:forma normalizzata

la mantissa è sempre del tipo 1,xxxxx ossia $1 \leq m < 2$.

La forma normalizzata si ha quando l'esponente è compreso tra 1 e 254 (estremi inclusi). La codifica dei numeri reali nella forma normalizzata è:

$$N_{10} = (-1)^s \cdot m \cdot 2^{e-127}$$

dove $1 \leq m < 2$, $1 \leq e \leq 254$.

Il numero reale più piccolo in valore assoluto che si può ottenere è:

$$N_{\min} = 1.0 \cdot 2^{-126}$$

mentre il numero più grande è:

$$N_{\max} = 1.11111\ldots11 \cdot 2^{128}$$

Lo standard 754 dell'IEEE:forma denormalizzata

la mantissa è sempre del tipo 0,xxxxx ossia m<1.

La forma denormalizzata si ha quando l'esponente è uguale a 0.

La codifica dei numeri reali nella forma normalizzata è:

$$N_{10} = (-1)^s \cdot m \cdot 2^{-126}$$

dove m<1.

Il numero reale più piccolo in valore assoluto che si può ottenere è:

$$N_{\min} = 0.00000\ldots01 \cdot 2^{-126}$$

mentre il numero più grande è:

$$N_{\max} = 0.11111\ldots11 \cdot 2^{128}$$

Codici Alfanumerici

- I simboli che vengono usati per rappresentare testi sono, come noto, i caratteri alfanumerici, cioè l'insieme costituito
  - lettere dell'alfabeto
  - dieci cifre decimali.
  - simboli come lo spazio
  - i segni di interpunzione
  - i simboli per indicare il passaggio alla riga o alla pagina successiva, ecc.
- Questo insieme di caratteri alfanumerici può essere facilmente rappresentato attribuendo in maniera univoca a ciascuno dei suoi elementi un numero intero (codice).
- Esistono due diverse rappresentazioni:
  - codifica ASCII e relative estensioni
  - codifica Unicode.

Codifica ASCII

- La codifica ASCII (che si pronuncia ASKI), prende il nome da **American Standard Code for Information Interchange**.
- Utilizza 7 bit per un totale di 128 simboli rappresentabili.
- Da notare che i caratteri dell'alfabeto e le cifre numeriche successive hanno codice anch'esso successivo (ad esempio a ha codice 97, b codice 98, c codice 99, il numero 0 ha codice 48, il numero 1 codice 49, etc.)
- Tra le più utilizzate codifiche ASCII (entro i primi 128 simboli) vi sono:
  - ~ (tilde) codice 126
  - { codice 123
  - } codice 125
  - | codice 124
- I caratteri di controllo (non riproducibili) hanno i codici più bassi.
- Il blank (Spazio) è il primo dei caratteri riproducibili.
- Le maiuscole/minuscole sono ordinate (codice Progressivo).

Codifiche derivate dalla codifica ASCII

- Esistono numerose estensioni della codifica ASCII. Tali estensioni derivano dalla necessità di codificare simboli legati a particolari lingue, e dal fatto che operando su 8 bit, la codifica ASCII consente l'utilizzo dell'ottavo bit, lasciando gli altri 7 inalterati.
- Tutte le estensioni della codifica ASCII non modificano tale codifica ma aggiungono semplicemente altri 128 simboli.
- Tra le estensioni più diffuse vi è la ISO Latin 1.

Codifica Unicode

- La codifica Unicode supera i limiti della codifica ASCII e relativi derivati, in quanto estende il numero di simboli codificabili.
- Utilizza 32 bit
- Attualmente sono più di 95.000 simboli codificati con lo standard Unicode.
- Essi sono divisi in:
  - Script Moderni: Latino, Greco, Giapponese, Cinese, Koreano, etc.
  - Script Antichi: Sumero, Egiziano, etc.
  - Segni Speciali
- La codifica Unicode è in ogni caso compatibile con la codifica ASCII