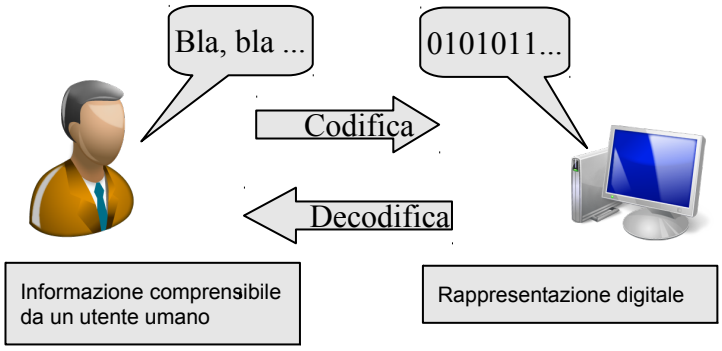
Codifica dell'Informazione

La rappresentazione delle Informazioni

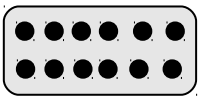
La funzione svolta da un programma di un computer è l'acquisizione delle informazioni dal mondo esterno, la loro trasformazione (elaborazione) e l'emissione delle informazioni verso il mondo esterno. Durante l'elaborazione le informazioni acquisite sono temporaneamente memorizzate nella memoria centrale. Tra i possibili casi di emissione esiste anche la memorizzazione permanente in un file nella memoria di massa. Si ricorda inoltre che nel modello di Von Neumann un programma è costituito da una sequenza di istruzioni codificate nella memoria centrale del computer.

In questo modulo si analizza in che modo le informazioni vengono rappresentate in memoria, siano esse dati del problema o codici delle istruzioni del programma. Questa rappresentazione deve essere adatta alla tecnologia usata per la realizzazione delle memoria e dei sistemi di elaborazione e si parla quindi di "rappresentazione digitale" o anche di "codifica binaria".



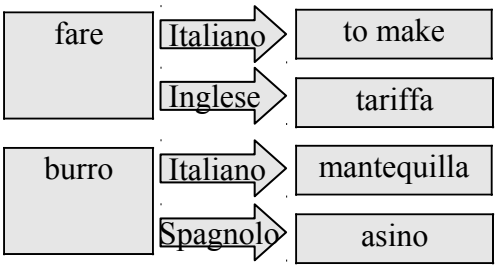
Si deve mantenere separato il concetto di "informazione" dal concetto di "rappresentazione dell'informazione". L'informazione è un evento che toglie incertezza su un qualche fatto mentre la rappresentazione è il modo in cui l'informazione viene comunicata. Supponiamo di dover comunicare una informazione numerica, ad esempio il numero 12 che qui viene espresso nella sua

rappresentazione "decimale posizionale". Oltre a questo modo di rappresentarlo esistono anche molti altri modi:



1100_2

La stessa informazione può avere rappresentazioni differenti.
Il numero è una informazione legata in questo caso al concetto di quantità che è indipendente dalla sua rappresentazione e può essere "codificata" in molti modi diversi; chi riceve l'informazione codificata, per comprenderne il significato deve conoscere le regole di codifica e decodificare l'informazione.
Esempio:



La stessa rappresentazione può corrispondere a diverse informazioni in diverse codifiche.

Sistemi di codifica

Un sistema di codifica:

- Usa un insieme di simboli di base detto "alfabeto"
- I simboli dell'alfabeto possono essere combinati ottenendo differenti configurazioni distinguibili una dall'altra dette "codici" o "stati".
- Associa ogni configurazione ad una particolare entità di informazione in modo da poterla rappresentare.

Codifica unaria
Consente di rappresentare quantità numeriche.
L'alfabeto è composto da un unico simbolo: ●
Le quantità si formano ripetendo più volte il simbolo:
● = 1
● ● = 2
● ● ● = 3

Codifica decimale posizionale

- Alfabeto
  - Cifre "0", "1", "2", …, "9"
  - separatore decimale (",")
  - separatore delle migliaia (".")
  - Segni positivo ("+") e negativo ("-")
- Regole di composizione (sintassi=struttura)
  - Definiscono le combinazioni ben formate
- 12.318,43 (valida in IT non valida in UK)
- 12,318.43 (valida in UK non valida in IT)
- Regole di codifica (semantica=significato)
  - Associano ad ogni configurazione un'entità di informazione
  - **$12.318,43 = 1\times10^4+2\times10^3+3\times10^2+1\times10^1+1\times10^0+4\times10^{-1}+3\times10^{-2}$**

Esistono anche codifiche numeriche non posizionali come ad esempio la numerazione romana (codifica addittiva). In questo caso il significato dei non cambia con la posizione ma la quantità si ottiene sommando (o sottraendo) tutti i simboli che sono presenti nel numero.
Alfabeto:

- I=1, V=5, X=10, L=50, C=100, M=1000
- III=3 (1+1+1), IV=4 (5-1), IX=9 (10-1), XIV=14 (10+5-1)

Codifica binaria

All'interno di un sistema di elaborazione ogni informazione è codificata in forma binaria o digitale, utilizzando cioè un alfabeto di due soli simboli: 0 e 1

Le ragioni della scelta della codifica binaria sono prevalentemente di carattere tecnologico; i due simboli possono corrispondere a:

- due possibili stati di polarizzazione di una sostanza magnetizzabile
- Passaggio/non passaggio di corrente attraverso un conduttore
- Passaggio/non passaggio di luce attraverso una fibra ottica

Indipendentemente dall'effettivo valore della grandezza fisica che rappresenta l'informazione si divide in due intervalli l'insieme dei valori che la grandezza può assumere: ogni intervallo corrisponde ad un simbolo
Il vantaggio di usare solo due simboli è di ridurre la probabilità di errore; tanti più simboli si devono distinguere e tanto meno la rilevazione sarà affidabile perché gli intervalli della grandezza fisica saranno meno ampi.

BIT
L' entità minima di informazione che si può individuare in un sistema di elaborazione viene detta BIT e può assumete due valori:
Il termine BIT deriva dalla contrazione delle due parole: BInary digiT (cifra binaria)
Con una cifra binaria posso rappresentare soltanto due informazioni, ognuna delle quali viene fatta corrispondere convenzionalmente
al simbolo 1 o 0 (ma anche on/off, si/no ecc ecc).

Per poter rappresentare un numero maggiore di informazioni si usano sequenze di bit.
Ad esempio utilizzando 2 bit si possono rappresentare quattro informazioni diverse:
00 01 10 11

Il processo che fa corrispondere a una informazione una configurazione di bit prende il nome di **codifica binaria** dell'informazione
L'associazione informazione/configurazione di bit è convenzionale e può seguire diverse regole: l'importante è che tutti quelli che devono condividere l'informazione usino la stessa convenzione.

Con 1 bit si codificano 2 informazioni: $N=2^1$
Con 2 bit si codificano 4 informazioni: $N=2^2$
Con 3 bit si codificano 8 informazioni: $N=2^3$
…
Con K bit si codificano N informazioni: $N=2^k$
La sequenza di K bit si può trovare in N stati diversi.

Esercizio 1:
Supponiamo di volere codificare i giorni della settimana.
Significato delle informazioni: Lunedì, Martedì, Mercoledì, Giovedì, Venerdì, Sabato, Domenica.
La quantità di informazioni da codificare è 7.
Si deve scegliere una sequenza di bit che possa assumere almeno 7 stati diversi.
La scelta cade su una sequenza di 3 bit che consente di codificare 8 stati quindi:
Lunedì=000, Martedì=001, Mercoledì=010, Giovedì=011, Venerdì=100, Sabato=101, Domenica=110 (111 non usato)

Esercizio 2:
Supponiamo di volere codificare una griglia di valutazione di un compito; i possibili voti sono: insufficiente, sufficiente, discreto, buono, ottimo.
La quantità di informazioni da codificare è 5.
Anche in questo caso la scelta su una sequenza di 3 bit perchè una sequenza di due bit codifica solo 4 stati quindi:
insufficiente=000, sufficiente=001, discreto=010, buono=011, ottimo=100 (101,110,111 non usati)

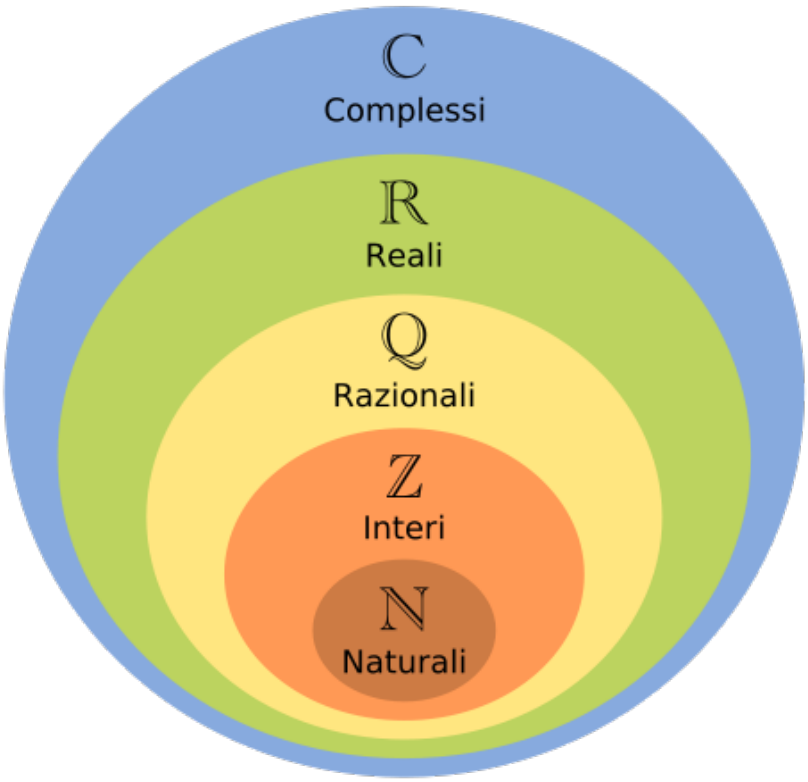
A questo punto ci si può chiedere: ma la codifica 010 rappresenta "Martedì" oppure "discreto"?
La risposta a questa domanda è già stata definita nella pagina precedente: **l'associazione informazione/configurazione di bit è convenzionale, si può interpretare un codice solo se si conoscono a priori le regole di codifica.**

BYTE
Una sequenza di 8 bit viene denominato byte. Il byte assume una particolare importanza in informatica perché rappresenta l'unità di misura degli elementi di memoria.
Un byte consente di codificare 2^8= 256 informazioni diverse cioè un byte di memoria si può trovare in 256 stati diversi
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 1
0 0 0 0 0 0 1 0
...
1 1 1 1 1 1 1 1

Per misurare la quantità di informazione si usano i multipli del byte:
- Kilo KB 2^{10} = 1024 byte (circa mille byte)
- Mega MB 2^{20} = 1KB*1024 (circa un milione di byte)
- Giga GB 2^{30} = 1MB*1024 (circa 1 miliardo di byte)
- Tera TB 2^{40} = 1GB*1024 (circa mille miliardi di byte)

Tipi di informazione e codifica binaria
Qualsiasi tipo di informazione può essere codificato in binario. Tra i più comuni tipi di informazione che necessitano di una codifica troviamo:
- numeri
- caratteri tipografici
- immagini
- suoni
- video

Rappresentazione degli insiemi numerici



Codifica binaria dei numeri naturali

I numeri naturali rappresentano un insieme numerico formato dallo 0 (assioma) e da una quantità infinita di numeri ottenuti ciascuno dal precedente aggiungendo 1 (successore).

Quindi definito lo 0 si ottiene l'insieme (N) formato dai numeri:

0,1, 2, 3, 4,5,6,7,8,9,10,11,12, … ∞

I numeri naturali sono rappresentati con il sistema decimale posizionale privo della virgola decimale e delle cifre a destra della virgola che appartengono agli insiemi dei razionali (Q) o reali (R).

Mancano anche i numeri negativi che appartengono ad un distinto insieme definito "insieme dei numeri interi" (Z).

E' possibile dare una rappresentazione binaria dei numeri naturali mantenendo la codifica posizionale che hanno nel sistema decimale passando ad una codifica "binaria posizionale".

Nella numerazione decimale posizionale le cifre che formano un numero assumono un significato diverso in relazione alla posizione che occupano nella sequenza.

Ad esempio se scrivo il numero naturale 129 intendo dire che è formato da 9 unità, 2 decine ed un centinaio dove le cifre 1, 2 e 9 sono i coefficienti da moltiplicare rispettivamente per 100, 10 ed 1 e da sommare tra loro per ottenere la quantità desiderata.
Esprimendo questo concetto con una formula si ottiene:

$$\mathbf{129=1\times10^2+2\times10^1+9x10^0}$$

Generalizzando il concetto si ottiene la formula per un generico numero naturale a k cifre:

$$\mathbf{N=a_{k-1}\times10^{k-1}+a_{k-2}\times10^{k-2}+...+a_1\times10^1+a_0x10^0=}\sum_{i=0}^{k-1} a_i\times10^i$$

I coefficienti $\mathbf{a_i}$ sono le cifre che si trovano nelle varie posizioni e possono assumere valori da 0 a 9. Ogni cifra è moltiplicata per un "peso" $\mathbf{10^i}$ che dipende dalla posizione della cifra. I pesi sono ottenuti dalla "base di numerazione" che rappresenta la quantità di cifre disponibili nel sistema di numerazione.

$$\mathbf{N=a_{k-1}\times B^{k-1}+a_{k-2}\times B^{k-2}+...+a_1\times B^1+a_0xB^0=}\sum_{i=0}^{k-1} a_i\times B^i$$

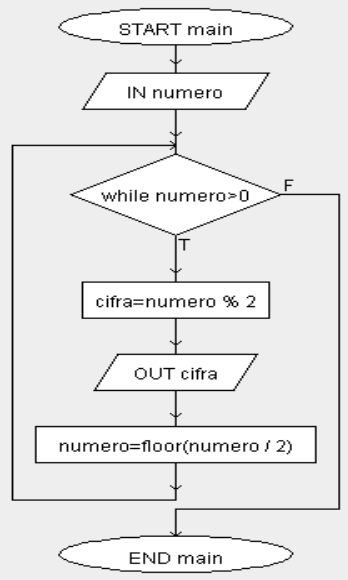
Questa formula vale anche per la numerazione binaria dove i simboli dell'alfabeto sono {0,1} e la base B=2:

$$\mathbf{N=a_{k-1}\times2^{k-1}+a_{k-2}\times2^{k-2}+...+a_1\times2^1+a_0x2^0=}\sum_{i=0}^{k-1} a_i\times2^i$$

Ad esempio se prendiamo in considerazione la sequenza di simboli binari **1101** e la interpretiamo come un numero naturale espresso in numerazione binaria posizionale possiamo scrivere:

$\mathbf{1101_2=1\times2^3+1\times2^2+0\times2^1+1\times2^0=1\times8+1\times4+0\times2+1\times1=8+4+1=13_{10}}$
Si tratta di due rappresentazione diverse della stessa informazione numerica. Le due rappresentazione condividono la regola di codifica che è la numerazione posizionale ma usano basi diversi.
Poiché la regola di codifica è la stessa è possibile passare da una rappresentazione all'altra effettuando una conversione di base.

Conversione da base 10 a base 2
Per convertire un numero decimale alla sua rappresentazione binaria si deve dividere ripetutamente il numero per 2. Ad ogni divisione si ottiene un quoziente ed un resto. Il resto di una divisione per 2 può essere 0 oppure 1. La cifra che si ottiene come resto rappresenta la cifra binaria del numero nella posizione corrispondente al passo di ripetizione quindi se siamo al primo passo rappresenta la cifra più a destra (posizione 0). Al passo successivo si divide il quoziente della divisione precedente nuovamente per 2 ottenendo la cifra di posizione successiva ed un nuovo quoziente. Si procede in questo modo determinando tutte le cifre binarie di posizione crescente fino a quando il quoziente si azzera.
Questa descrizione può essere rappresentata con un algoritmo iterativo:




```
PROG main
  IN numero
  WHILE numero>0
    ASSIGN cifra=numero%2
    OUT cifra
    ASSIGN
    numero=floor(numero/2)
  END WHILE //numero>0
END PROG //main
```

Esempio di divisioni successive:
Conversione in binario del numero naturale 210

Quoziente	Divisore	Resto	
210	2	0	Cifra meno significativa
105	2	1	
52	2	0	
26	2	0	
13	2	1	
6	2	0	
3	2	1	
1	2	1	Cifra più significativa
0			

La rappresentazione binaria del numero naturale decimale 210 è quindi: 11010010

In generale per convertire da base dieci ad una qualsiasi altra base B si usa questo algoritmo sostituendo il divisore con la nuova base ed ottenendo resti che vanno da 0 a B-1.

Conversione da base 2 a base 10

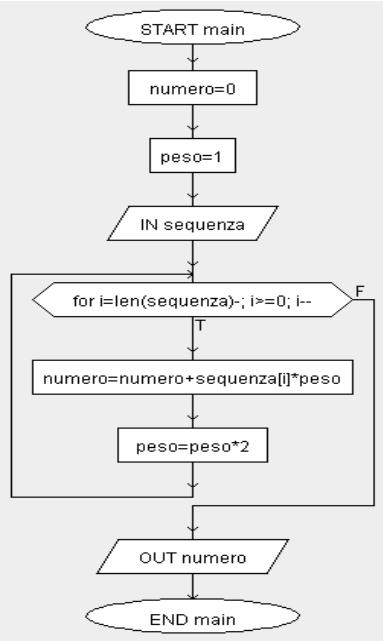
Per convertire un numero dalla base 2 alla base 10 si applica la definizione di numerazione posizionale già definita in precedenza:

$$N=a_{k-1}\times 2^{k-1}+a_{k-2}\times 2^{k-2}+\ldots+a_1\times 2^1+a_0 x 2^0=\sum_{i=0}^{k-1} a_i\times 2^i$$

Esempio:

$11010010 = 1x2^7+1x2^6+0x2^5+1x2^4+0x2^3+0x2^2+1x2^1+0x2^0=$
$=1x128+1x64+0x32+1x16+0x8+0x4+1x2+0x1=$
$=128+64+16+2=210$

Dal punto di vista algoritmico si può descrivere nel seguente modo:




```
PROG main
  ASSIGN numero=0
  ASSIGN peso=1
  IN sequenza
  FOR
  i=len(sequenza);i>=0;i--
    ASSIGN
    numero=numero+
           sequenza[i]*peso
    ASSIGN peso=peso*2
  END FOR
  OUT numero
END PROG
```

Finitezza della codifica: la precisione

L'insieme numerico N della matematica è illimitato; si può procedere da 0 individuando un successore senza limitazioni e per questo motivo viene definito il concetto di infinito (∞) che non è un numero ma la segnalazione che non esiste un limite superiore al conteggio.

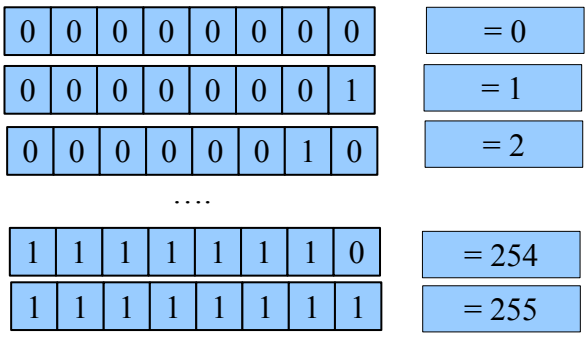
Nella rappresentazione delle informazioni numeriche in un sistema di elaborazione (codifica binaria) le informazioni devono essere invece rappresentate con un **numero finito e predefinito** di cifre binarie.

La quantità di cifre binarie utilizzate per rappresentare una informazione viene chiamata "**precisione**".

La definizione della precisione pone anche un limite alla quantità di informazioni rappresentabili e quindi l'insieme delle codifiche costituisce un sotto-insieme dell'insieme numerico di partenza.

Esempio:

una rappresentazione binaria dei naturali ad 8 bit consente di rappresentare solo $2^8 = 256$ codifiche; in questa codifica esisteranno quindi solamente i numeri naturali che vanno da 0 a 255.



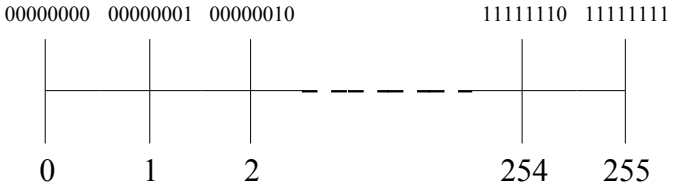
A differenza dalle rappresentazioni manuali dei numeri dove si omettono gli zeri a sinistra (cifre non significative) nella rappresentazioni automatiche non si possono mai omettere gli zeri a sinistra: un numero in una determinata precisione è formato sempre

da una quantità prefissata di cifre binarie indipendentemente dalla sua grandezza.

La codifica adottata è quella posizionale binaria quindi si ottengono le corrispondenti rappresentazioni decimali applicando la regola:

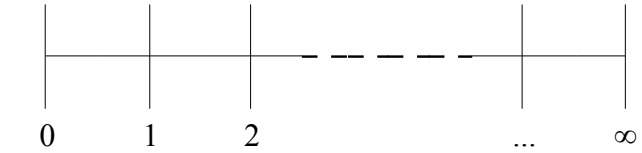
$$\mathbf{N=a_7\times 2^7+a_6\times 2^6+...+a_1\times 2^1+a_0 x 2^0=}\ \sum_{i=0}^{7} a_i \times 2^i$$

Non è possibile rappresentare con questa precisione il numero 256 perché richiederebbe 9 bit (100000000).
Si può descrivere l'insieme con una rappresentazione grafica lineare rappresentando gli elementi su una retta:



Cambiando il numero di bit della precisione cambia il numero di segmenti presenti nella retta ma il loro numero rimane finito.
Ad esempio con 32 bit si ottengono 4.294.967.296 elementi di codifica (circa quattro miliardi).

L'insieme N della matematica invece si rappresenterebbe in questo modo:



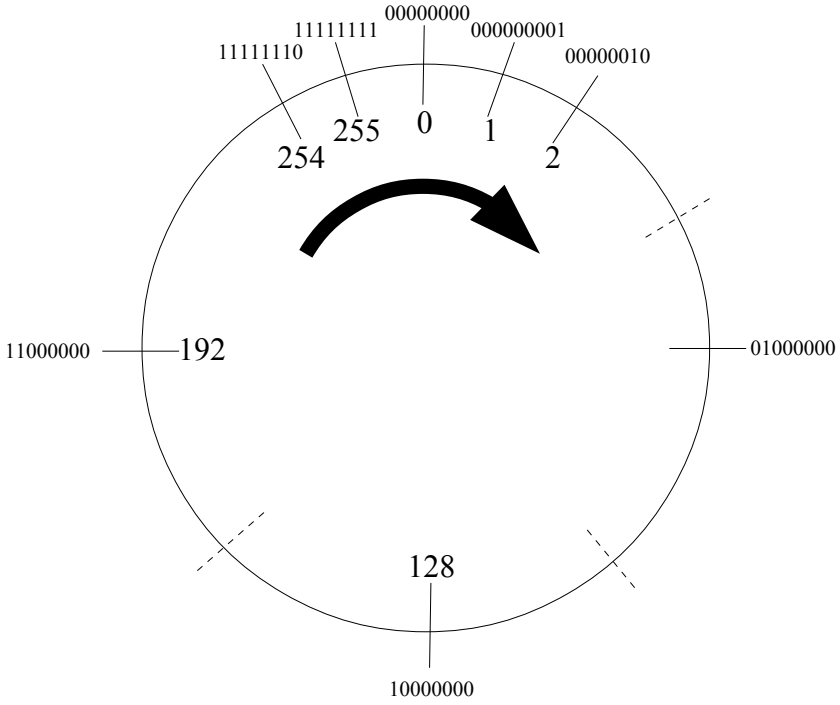
Non è possibile realizzare una codifica binaria di questo insieme che è formato da un numero infinito di elementi.

Overflow
La finitezza della rappresentazione porta come conseguenza alla possibilità di ottenere, facendo elaborazioni tra elementi dell'insieme, elementi che non fanno parte dell'insieme.
Se ad esempio, in una codifica ad 8 bit, abbiamo il numero 255 (11111111) ed aggiungiamo 1 a questo numero si ottiene:

```
riporti:  11111111
           11111111+
           00000001=
          ----------
          100000000
```

Poiché, a parte la differenza di base, la codifica è posizionale come nel caso della rappresentazione decimale la somma segue le stesse regole. Nell'esempio è descritta l'operazione di somma in colonna con riporti. Si parte dalla cifra meno significativa (colonna 0) e si sommano le cifre dei due addendi:
1+1 = 10
la somma tra le due cifre binarie ha determinato un riporto sulla cifra di peso superiore che viene rappresentato in grigio nello schema.
La somma della seconda colonna, tenendo conto del riporto è:
1+1+0 = 10
Ancora una volta si verifica un riporto e questa situazione si ripete fino all'ultima colonna (colonna 7) che provoca un riporto sulla colonna 8. La colonna 8 però non esiste in una rappresentazione ad 8 bit quindi il nono bit viene scartato ed il risultato della operazione è 00000000 che in decimale non rappresenta 256 ma 0.

Si è verificato un "**overflow**" (**traboccamento**); cioè l'operazione tra due elementi dell'insieme avrebbe portato ad un valore che non appartiene all'insieme: usando una codifica posizionale il traboccamento si presenta come un passaggio a valori che si trovano all'altro estremo della rappresentazione.
Si può capire meglio la situazione se invece che una rappresentazione grafica lineare dell'insieme si una una rappresentazione grafica circolare:



Immaginiamo di prendere i due estremi della linea e di unirli insieme incurvando la linea fino a formare un cerchio.
L'operazione di somma 255+1 corrisponde ad un movimento verso destra lungo la circonferenza che dalla posizione 255 porta alla posizione 0.