Corso di Calcolatori Elettronici

Rappresentazione dei numeri interi in un calcolatore

Prof. Roberto Canonico

Università degli Studi di Napoli Federico II
Dipartimento di Ingegneria Elettrica e
delle Tecnologie dell'Informazione

Sistemi di Numerazione

- Un **sistema di numerazione** può essere visto come **un insieme di simboli (cifre) e regole** che **assegnano ad ogni sequenza di cifre uno ed un solo valore numerico**
- I sistemi di numerazioni vengono di solito classificati in sistemi **posizionali** e **non posizionali**
  - Nei **sistemi posizionali** ogni cifra della sequenza ha un'importanza variabile a seconda della relativa posizione
    - Es.: Nel sistema decimale "15" è diverso da "51".
  - Nei **sistemi non posizionali** ogni cifra esprime una quantità non dipendente dalla posizione
    - Es.: Nel sistema romano il simbolo "L" esprime la quantità 50, indipendentemente dalla posizione.

Numerazione posizionale pesata

- In un **sistema di numerazione posizionale pesata**, ogni numero n si rappresenta con una sequenza di caratteri-cifra c_i, dove:

$$n = c_i \times b^i + c_{i-1} \times b^{i-1} + c_{i-2} \times b^{i-2} + \ldots + c_2 \times b^2 + c_1 \times b^1 + c_0 \times b^0 + c_{-1} \times b^{-1} + c_{-2} \times b^{-2} + \ldots$$

- Per i numeri interi, possiamo **omettere le potenze negative** della base, dal momento che i loro coefficienti sono tutti nulli; es.:

$$1234 = 1\text{x}10^3 + 2\text{x}10^2 + 3\text{x}10^1 + 4\text{x}10^0$$

- Un sistema di numerazione posizionale è definito dalla **base** (o radice) **utilizzata per la rappresentazione**
- Data una base b, ogni **coefficiente** c_i può assumere come valore uno tra b simboli (*cifre*)

Base	Denominazione	Valori delle cifre
10	Decimale	0 1 2 3 4 5 6 7 8 9
2	Binaria	0 1
8	Ottale	0 1 2 3 4 5 6 7
16	Esadecimale	0 1 2 3 4 5 6 7 8 9 A B C D E F

Proprietà delle rappresentazioni

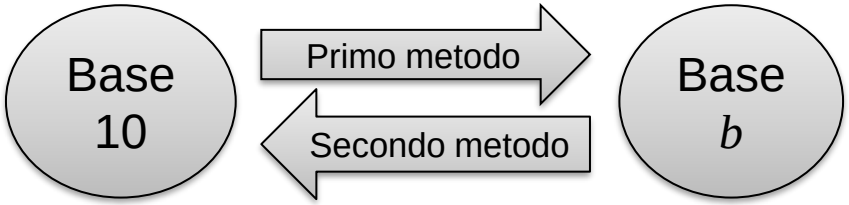
- Nel passaggio da una base all'altra alcune proprietà dei numeri possono cambiare, ad esempio una divisione può essere periodica in base dieci ma non in base due, o viceversa.
- Quando si usano più basi, per evitare ambiguità si indica in pedice la base con la quale è rappresentato il numero:
  - $(27)_{10} = (11011)_2$
- Un numero rappresentato in binario (ma in generale con una base minore di quella di partenza) ha una stringa più lunga
  - Per avere rappresentazioni più compatte rispetto a quella binaria ci si avvale del sistema a base 8 (ottale) e di quello a base 16 (esadecimale).

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

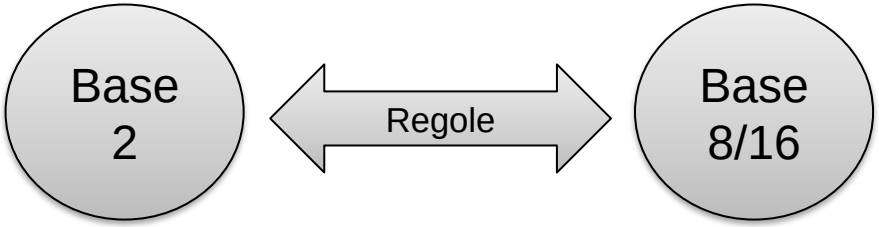
Esadecimale	Binario
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
A	1010
B	1011
C	1100
D	1101
E	1110
F	1111

Conversione tra sistemi di numerazione

- Per cambiare la base di rappresentazione di un numero, esistono:
  1. Un algoritmo basato sull'applicazione ripetuta di divisioni e moltiplicazioni da compiere nel sistema di origine.
  2. Un metodo basato sul calcolo della somma di potenze come nella definizione, esprimendo i numeri nella base di destinazione.
- Usiamo il primo per convertire dal sistema decimale, mentre usiamo il secondo per convertire in decimale.



- Poiché 8 e 16 sono potenze di 2, esistono poi regole semplici per convertire dal sistema binario a quello ottale/esadecimale, e viceversa.

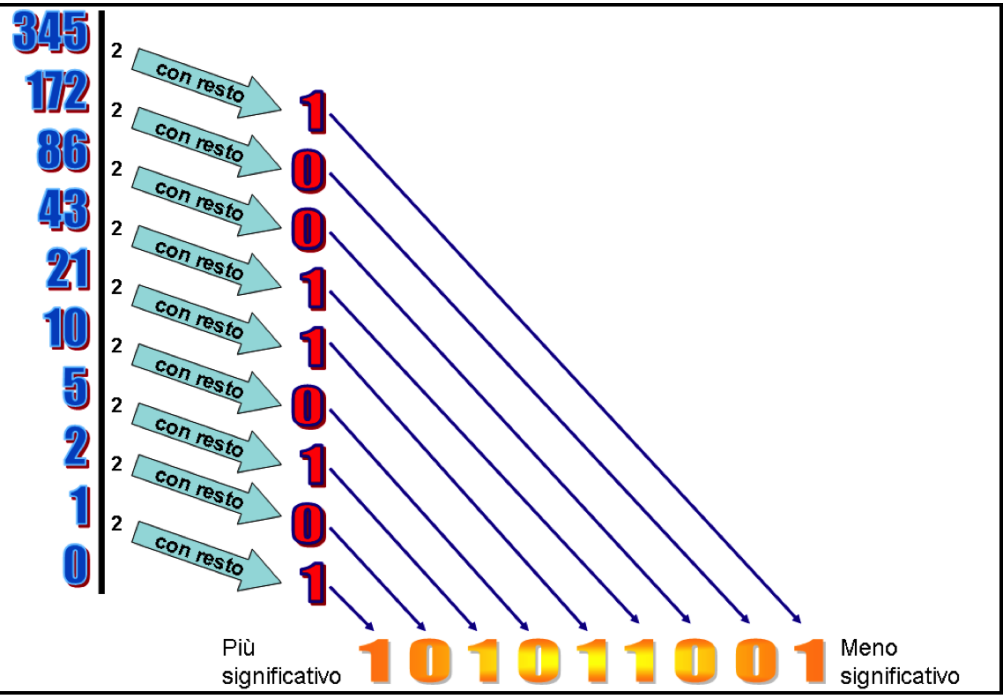


Conversione decimale-binario: parte intera

ALGORITMO

1) dividere la parte intera del numero ***d*** per la base ***b***
2) scrivere il resto della divisione
3) se il quoziente è maggiore di zero, usare tale risultato al posto del numero ***d*** di partenza e continuare dal punto 1)
4) se il quoziente è zero, scrivere tutte le cifre ottenute come resto in sequenza inversa

Si noti che l'algoritmo consente di convertire un numero intero in base dieci in una qualunque base ***b***. Nel caso di ***b*** =2 si ottiene la conversione in binario del numero assegnato.

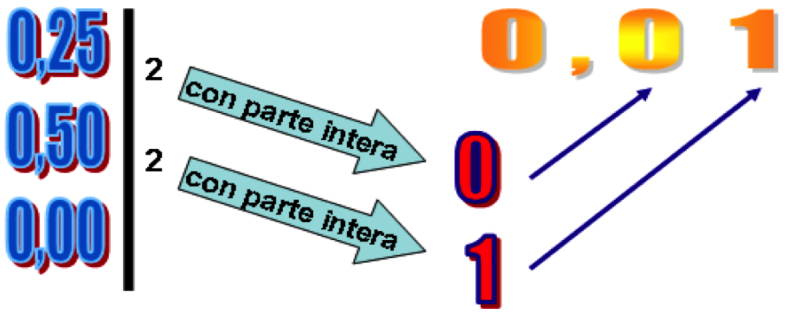


Conversione decimale-binario: parte frazionaria

ALGORITMO

1) moltiplicare la parte frazionaria del numero ***d*** per la base ***b***
2) scrivere la parte intera del prodotto
3) se la nuova parte frazionaria del prodotto è diversa da zero o non si ripete periodicamente, oppure si non sono state determinate le cifre binarie prefissate, usare tale risultato al posto del numero ***d*** di partenza e continuare dal punto 1)
4) se la nuova parte frazionaria verifica una delle tre condizioni di terminazione, scrivere tutte le cifre ottenute come parte intera nell'ordine in cui sono state calcolate

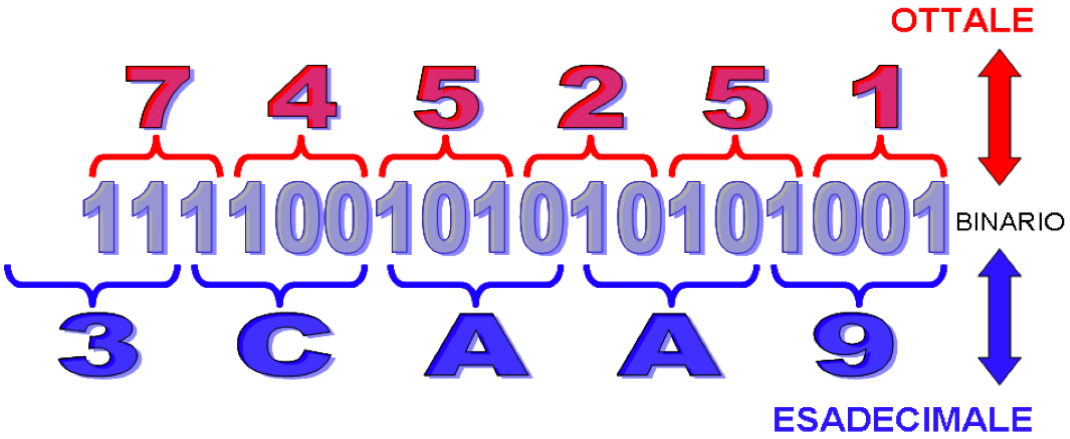
Si noti che l'algoritmo consente di convertire un numero frazionario in base dieci in una qualunque base ***b***. Nel caso di ***b*** =2 si ottiene la conversione in binario del numero assegnato.



- Condizioni di terminazione:
  - la parte frazionaria si annulla;
  - la parte frazionaria si ripete con periodicità;
  - si raggiunge un numero di cifre sufficienti per l'approssimazione desiderata del numero.
- Solo la prima condizione garantisce una conversione senza approssimazione.

Conversione da base 2 a base 8 o 16

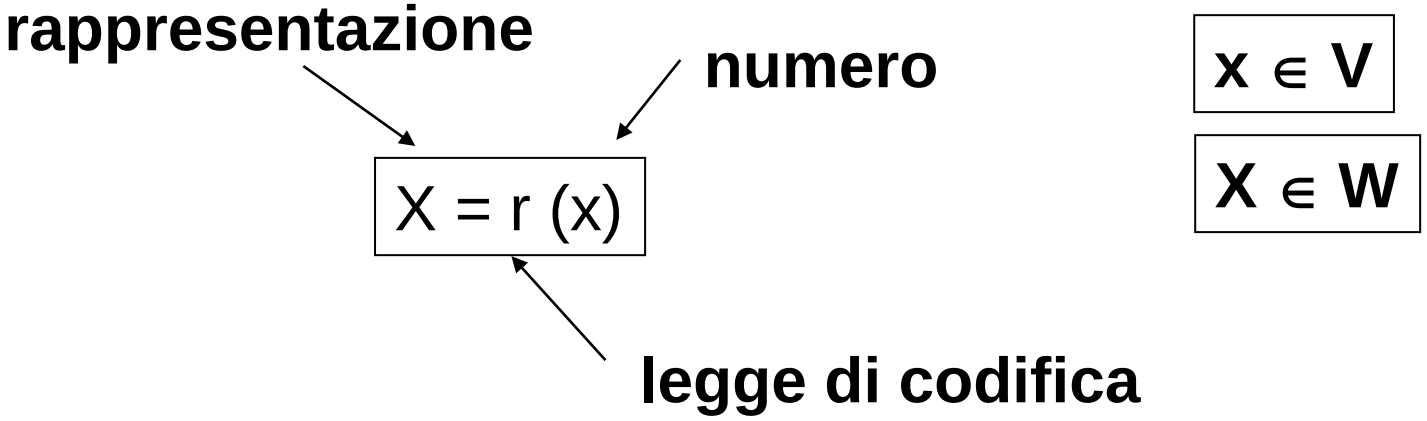
- Poiché 8 e 16 sono potenze di 2, per convertire da binario a ottale o esadecimale è possibile semplicemente applicare una codifica tenendo conto che:
  - 3 cifre binarie (bit) codificano 1 cifra ottale e viceversa.
  - 4 cifre binarie (bit) codificano 1 cifra esadecimale e viceversa.
- Per convertire da binario suddividiamo la stringa binaria in porzioni di 3 o 4 bit e convertiamo ogni gruppo nella corrispondente cifra dell'altra base
- Usiamo il procedimento inverso per la decodifica



Un byte (8 bit) si può rappresentare con 2 cifre esadecimali

Rappresentazione di numeri in un calcolatore

- Così come per qualsiasi altro tipo di dato, anche i numeri, per essere immagazzinati nella memoria di un calcolatore, devono essere codificati, cioè tradotti in sequenze di simboli
- Nei calcolatori si usano strategie di codifica binaria (k=2)
- L'alfabeto sorgente è costituito dall'insieme dei numeri che si vogliono rappresentare



Rappresentazione

- Bisogna tener conto dei seguenti fattori:
  - L'insieme V dei *numeri da rappresentare*
  - L'insieme W dei *numeri rappresentanti*
  - Tra i due insiemi si stabilisce una corrispondenza che trasforma un elemento x di V in uno X di W
  - Si dice allora che **X è la rappresentazione di x**
  - La decomposizione in cifre del numero X
  - La codifica in bit delle cifre

Strategie di codifica in macchina

- **Codifica binaria a lunghezza fissa**
- Il numero di bit varia a seconda della cardinalità dell'insieme dei numeri che si desidera rappresentare
  - Nella pratica, resta comunque pari ad un multiplo di 8 bit (tipicamente 8, 16, 32, 64 bit)
- L'associazione di un numero alla parola codice viene
  - Realizzata differentemente a seconda della tipologia di numeri che si desidera rappresentare
    - naturali, relativi, razionali, ecc …
  - Influenzata da aspetti che mirano a preservare la facile manipolazione delle rappresentazioni da parte del calcolatore
    - operazioni aritmetiche, confronti logici, ecc …
- **Le operazioni aritmetiche vengono eseguite sulle rappresentazioni binarie dei numeri**

Somme e Sottrazioni in aritmetica binaria

- Si effettuano secondo le regole del sistema decimale, ossia sommando (sottraendo) le cifre di pari peso
- Come nelle usuali operazioni su numeri decimali, si può avere un riporto sul bit di peso immediatamente superiore (**carry**), o un prestito dal bit di peso immediatamente superiore (**borrow**)
- Le somme (differenze) bit a bit sono definite come segue:

0+0=0	0-0=0
0+1=1	1-0=1
1+0=1	1-1=0
1+1=0 (carry=1)	0-1=1 (borrow=1)

- Ulteriore caso elementare:

1 + 1 + 1 = 1 (carry=1)

Moltiplicazione in aritmetica binaria

- La moltiplicazione bit a bit può essere definita come segue:

  0x0=0
  0x1=0
  1x0=0
  1x1=1

Rappresentazione di insiemi numerici infiniti

- Sia la dimensione che il numero dei registri in un calcolatore sono finiti
- La cardinalità degli insiemi numerici che occorre rappresentare è, invece, infinita
  - N = insieme dei numeri Naturali
  - Z = insieme dei numeri Relativi
  - Q = insieme dei numeri Razionali
  - R = insieme dei numeri Reali
- È inevitabile dunque che di un insieme di cardinalità infinita solo un sotto-insieme finito di elementi possa essere rappresentato

Overflow

- Gli operatori aritmetici, pur essendo talvolta chiusi rispetto all'intero insieme numerico su cui sono definiti, non lo sono rispetto ad un suo sottoinsieme di cardinalità finita
- Quando accade che, per effetto di operazioni, si tenta di rappresentare un numero non contenuto nel sottoinsieme si parla di *overflow*
- *Es.* sottoinsieme dei numeri naturali compresi tra 0 e 127 (rappresentabili con 7 bit):
  - La somma 100 +100 genera un overflow, essendo il numero 200 non rappresentabile nel sottoinsieme

Rappresentazione dei numeri naturali

- Rappresentare di un sottoinsieme dei numeri naturali attraverso stringhe di bit di lunghezza costante n
  - Il numero degli elementi rappresentabili è pari a 2^n
  - Tipicamente, volendo rappresentare sempre anche lo zero, si rappresentano i numeri compresi tra 0 e $2^n – 1$
- L'associazione tra ogni numero e la propria rappresentazione avviene, nei casi pratici, nella maniera più intuitiva
  - Ad ogni numero si associa la stringa di bit che lo rappresenta nel sistema di numerazione binario posizionale
- L'overflow avviene quando si tenta di rappresentare un numero esterno all'intervallo $[0, 2^n –1]$

Esempio

Rappresentazione dei numeri naturali su 4 bit

n=4

V = [0,15] ∩ N

Codifica: X=x

x	X_2
15	1111
14	1110
13	1101
12	1100
11	1011
10	1010
9	1001
8	1000
7	0111
6	0110
5	0101
4	0100
3	0011
2	0010
1	0001
0	0000

Operazioni sui numeri naturali

- Per realizzare le operazioni, il calcolatore può lavorare direttamente sulle rappresentazioni

- La correttezza dei calcoli è garantita dalle leggi dell'aritmetica binaria posizionale (analoghe a quelle della classica aritmetica decimale)

- L'overflow può essere facilmente rilevato attraverso la valutazione del riporto (o del prestito) sull'ultima cifra
  - In tale aritmetica, overflow = riporto uscente

Esempi

```
   6+          0110+
   8=          1000=
-----        -------
  14   ---->   1110
```

```
  11 -         1011-
   5 =         0101=
------       -------
   6   ---->   0110
```

```
   14+         1110 +
    3=         0011 =
-----        --------
  17   ---->   10001

 overflow
```

```
    9-          1001-
    7=          0111=
------       --------
   2   ---->    0010
```

```
      0101 ×
      0011 =
----------
      0101
     0101=
    0000==
   0000===
----------
   0001111
```

Rappresentazione dei numeri relativi

- Esistono diverse tecniche
- Segno e modulo
  - Corrispondente a quella comunemente utilizzata per i calcoli “a mano”
  - Poco utilizzata in macchina per le difficoltà di implementazione degli algoritmi, basati sul confronto dei valori assoluti degli operandi e gestione separata del segno
- Complementi
  - Complementi alla base
  - Complementi diminuiti
- Per eccessi

Rappresentazione in segno e modulo

- un singolo bit di X codifica il segno
  - Es. il più significativo, 0 se positivo, 1 se negativo
- i restanti n-1 bit di X rappresentano il modulo (numero naturale)
- La legge di codifica X=r(x) è: $X = |x| + 2^{n-1} * \text{sign}(x)$
  - $\text{sign}(x) = 0$ per $x \geq 0$, 1 per $x < 0$
- Si possono rappresentare i numeri relativi compresi nell'intervallo $[-(2^{n-1} - 1), 2^{n-1} - 1]$
- I numeri relativi rappresentati sono 2^n-1
- Lo zero ha 2 rappresentazioni 0positivo e 0negativo

Esempio

Rappresentazione in segno e modulo su 4 bit

n=4

$V = [-7,7] \cap Z$

Codifica:

$X=|x|+8 * sign(x)$

x	X_2	X_{10}
7	0111	7
6	0110	6
5	0101	5
4	0100	4
3	0011	3
2	0010	2
1	0001	1
0	0000;1000	0;8
-1	1001	9
-2	1010	10
-3	1011	11
-4	1100	12
-5	1101	13
-6	1110	14
-7	1111	15

Operazioni in segno e modulo

•Diversamente dalla rappresentazione dei numeri naturali, questa volta non è possibile lavorare direttamente sulle rappresentazioni dei numeri per realizzare le operazioni aritmetiche

• È necessario lavorare separatamente sul segno e sul modulo

•Quando, ad esempio, si sommano due numeri di segno discorde, bisogna determinare quello con modulo maggiore e sottrarre ad esso il modulo dell'altro. Il segno del risultato sarà quello dell'addendo maggiore in modulo.

•Tale caratteristica, insieme con il problema della doppia rappresentazione dello zero, rende i calcoli particolarmente laboriosi e, per questo motivo, non è molto utilizzata nella pratica.

Rappresentazione in complementi alla base

- Una seconda tecnica per la rappresentazione dei numeri relativi consiste nell'associare a ciascun numero il suo **resto modulo $M=2^n$**, definito come:

$$|x|_M = x-[x/M]*M$$

- Questo tipo di codifica, su n bit, è equivalente ad associare:
  - il numero stesso (cioè X=x), ai numeri positivi compresi tra 0 e $2^{n-1} – 1$;
  - il numero $X = 2^n – |x|$, ai numeri negativi compresi tra -2^{n-1} e – 1;
- I numeri rappresentati sono quelli compresi nell'intervallo

$$[–2^{n-1}; 2^{n-1} – 1]$$

Funzione intero

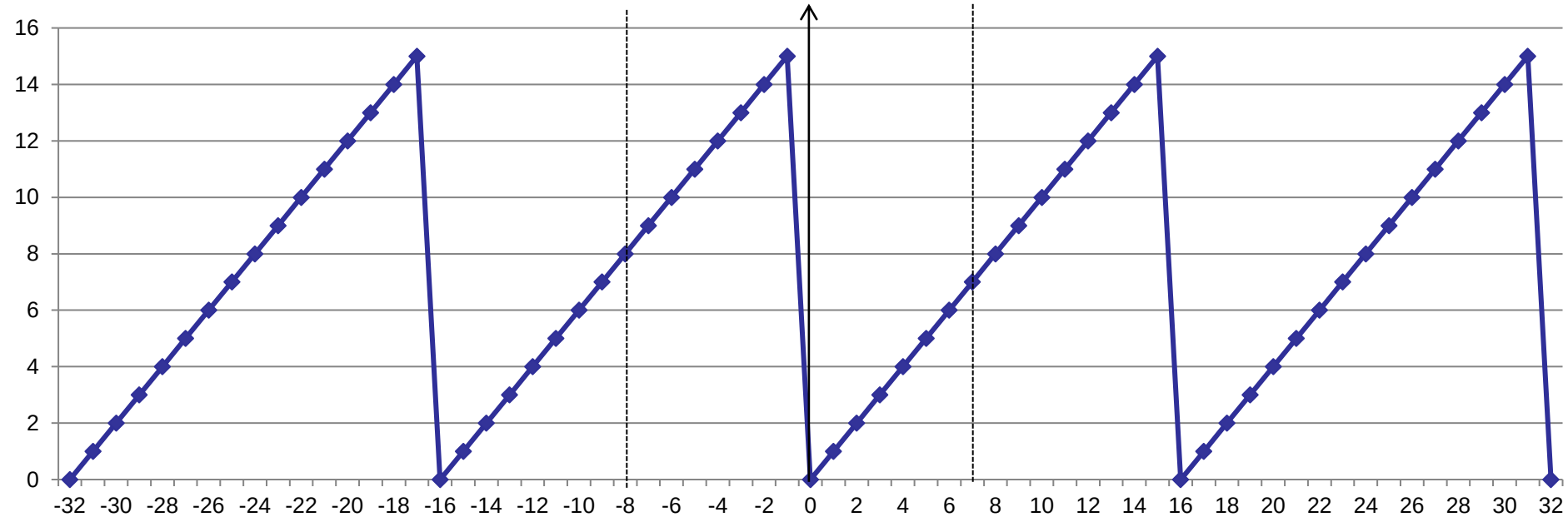
- Detto r un numero reale, si definisce intero di r il massimo intero y ≤ r

$$y = [\, r \,]$$

– confronto tra funzione intero [ ] e ceiling ⌈ ⌉

r	7.9	7	-7	-7.9
[r]	7	7	-7	-8
⌈r⌉	8	7	-7	-7

Resto modulo M (M=16)



Esempio

Rappresentazione in complementi alla base su 4 bit

n=4

$V = [-8,7] \cap Z$

Codifica:

Per $0 \le x \le 7$: $X = x$

Per $-8 \le x \le -1$: $X = 2^n - |x|$

x	X_2	X_{10}
7	0111	7
6	0110	6
5	0101	5
4	0100	4
3	0011	3
2	0010	2
1	0001	1
0	0000	0
-1	1111	15
-2	1110	14
-3	1101	13
-4	1100	12
-5	1011	11
-6	1010	10
-7	1001	9
-8	1000	8

Complementi alla base: proprietà

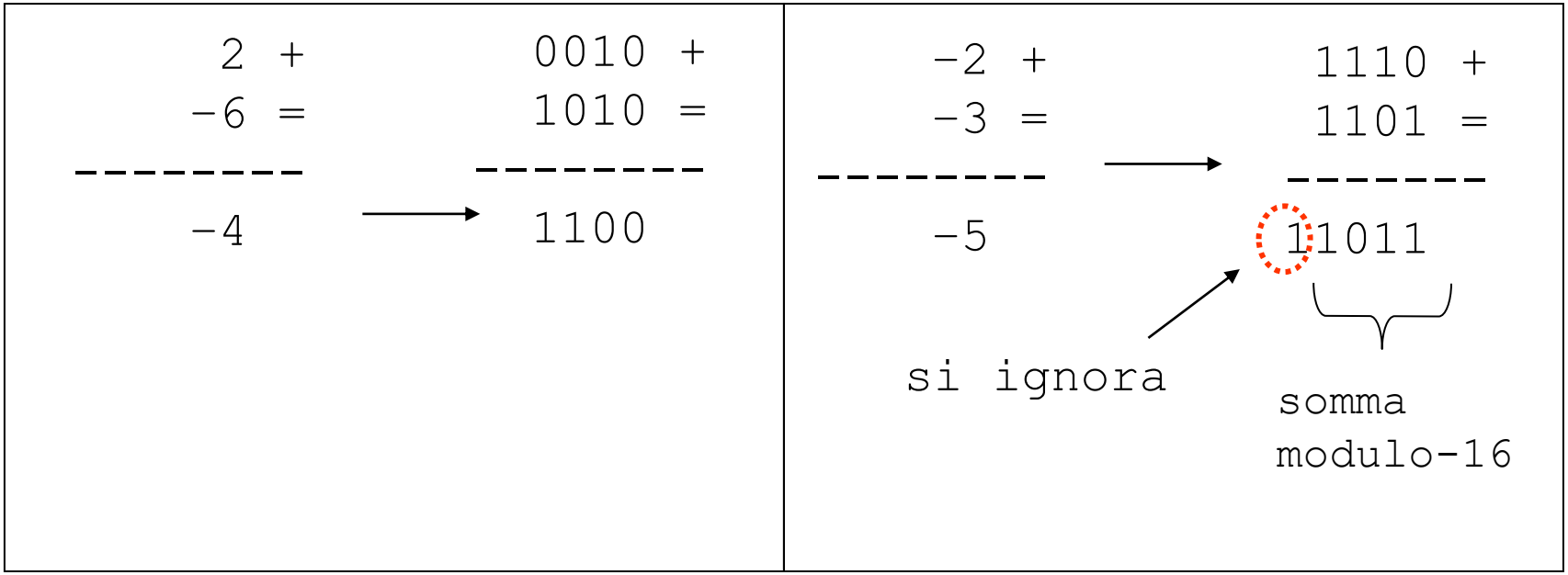
- Questa rappresentazione ha il fondamentale vantaggio di permettere, nell'ambito di operazioni aritmetiche, di lavorare direttamente sulle rappresentazioni.
- La regola sulla quale questa affermazione si basa è la seguente:

  la rappresentazione della somma (algebrica) di x ed y si ottiene come somma (modulo-M) delle rappresentazioni di x e y; analoghe sono le proprietà della differenza e del prodotto.

$$\left|x+y\right|_M = \left|\left|x\right|_M + \left|y\right|_M\right|_M$$

- Questo tipo di codifica conserva, inoltre, la proprietà delle rappresentazioni di avere il primo bit 1 se (e solo se) il corrispondente numero è negativo (bit di segno)

Esempi di addizioni in complementi alla base



È possibile effettuare la somma direttamente tra le rappresentazioni modulo-M: il risultato ottenuto in questo modo, è proprio la rappresentazione (modulo-M) del risultato corretto

Complementi alla base: la complementazione

- In complementi alla base, a partire dalla rappresentazione di un numero, è anche particolarmente semplice ottenere la rappresentazione del suo opposto
- È infatti sufficiente:
  - *complementare tutti i bit a partire da sinistra tranne l'uno più a destra ed eventuali zero successivi*
- *oppure, equivalentemente:*
  - *complementare tutti i bit e poi sommare uno al risultato*
- In virtù di ciò, è possibile realizzare le sottrazioni attraverso la composizione di una complementazione (nel senso sopra detto) ed un'addizione
- Nell'aritmetica in complementi alla base, di conseguenza, l'addizionatore e il complementatore rappresentano i componenti fondamentali per la realizzazione di tutte le operazioni

Esempi di complementazione su 4 bit

- La rappresentazione di 6_{10} su 4 bit è 0110_2.
- Complementando tutti i bit tranne l'uno più a destra e gli zero successivi si ottiene: 1010_2.
- 1010_2 è la rappresentazione di –6 in complementi alla base.

- La rappresentazione di 5_{10} su 4 bit è 0101_2.
- Complementando tutti i bit tranne l'uno più a destra e gli zero successivi si ottiene: 1011_2.
- 1011_2 è la rappresentazione di –5 in complementi alla base.

- La rappresentazione di 1_{10} su 4 bit è 0001_2.
- Complementando tutti i bit tranne l'uno più a destra e gli zero successivi si ottiene: 1111_2.
- 1111_2 è la rappresentazione di –1 in complementi alla base.

Complementi alla base: esempio di moltiplicazione

```
   2 *                0010 ×
 - 3 =                1101 =
--------          -----------
 - 6                 0010
                    0000=
                   0010==
                  0010===
                -------------
                  00111010
```

si ignora prodotto

Estensione del segno

- Problema:
  - Sia dato un intero N, rappresentato in complemento mediante n bit
  - Rappresentare N usando n+q bit (q>0)
- Soluzione:
  - Fare q copie di MSB

- Dimostrazione (banale per N positivo)
  - Sia N<0 (N=1bb…b , dove b è una cifra binaria)
  - Per induzione: Sia N_q la stringa con estensione di q bit
    - q=1: Poiché $-2^{n-1}=-2^n+2^{n-1}$, allora $V(N)=V(N_1)$.
    - q>1: estendere di un bit la stringa ottenuta da N con estensione di q-1 bit → $V(N_q)=V(N_{q-1})$
- Esempio
  - -6 $\rightarrow (1010)_2$ con 4 bit diventa $(\mathbf{1111}1010)_2$ su 8 bit
- Nota: questa operazione viene eseguita quando si fa in C un typecast da tipo short int ad int

Complementi diminuiti

- La rappresentazione in complementi diminuiti costituisce un'ulteriore alternativa per la codifica dei numeri relativi
- Concettualmente è analoga alla rappresentazione in complementi alla base
- La differenza rispetto ad essa è che la legge di codifica dei numeri negativi è leggermente differente:
  - $X=2^n - |x|;$ (complementi alla base)
  - $X=2^n - 1 - |x|;$ (complementi diminuiti)
- I numeri rappresentabili, se si utilizzano n bit, sono quelli compresi nell'intervallo $[-(2^{n-1} - 1), 2^{n-1} - 1]$.
- I numeri rappresentabili sono $2^n - 1$
- lo zero ha una doppia rappresentazione

Esempio

Rappresentazione in complementi diminuiti su 4 bit

n=4

$V = [-7,7] \cap Z$

Codifica:

Per $0 \le x \le 7$: $X = x$

per $-7 \le x \le -1$: $X = 2^n - 1 - |x|$

x	X_2	X_{10}
7	0111	7
6	0110	6
5	0101	5
4	0100	4
3	0011	3
2	0010	2
1	0001	1
0	0000;1111	0;15
-1	1110	14
-2	1101	13
-3	1100	12
-4	1011	11
-5	1010	10
-6	1001	9
-7	1000	8

Complementi diminuiti: perché?

- Maggiore semplicità con cui è possibile calcolare la rappresentazione dell'opposto di un numero, a partire dalla rappresentazione del numero stesso: basta semplicemente complementare tutti i bit della rappresentazione indistintamente
- Esempi:
  - la rappresentazione in complementi diminuiti su 4 bit di 4 è 0100;
    - complementando tutti i bit si ottiene 1011;
    - 1011 è la rappresentazione in complementi diminuiti su 4 bit di –4
  - la rappresentazione in complementi diminuiti su 4 bit di – 6 è 1001;
    - complementando tutti i bit si ottiene 0110;
    - 0110 è la rappresentazione in complementi diminuiti su 4 bit di 6

Aritmetica in complementi diminuiti

- Componenti:
  - Ancora l'addizionatore modulo-2^n (e non 2^n-1)
    - L'addizionatore modulo-2^n è più semplice da realizzare
  - Un complementatore
- Il risultato però deve essere opportunamente "corretto" per renderlo compatibile con l'aritmetica in modulo 2^n-1
- In particolare deve essere aggiunta un'unità al risultato nei seguenti casi:
  - se entrambi gli addendi sono negativi
  - se un addendo è positivo, l'altro negativo e la somma è positiva
- Nei casi suddetti l'aritmetica degli interi positivi (quella sulle rappresentazioni) dà overflow
  - Il riporto delle cifre più significative (carry) quindi può essere interpretato come la necessità di effettuare la correzione

Esempi di somme in complementi diminuiti

```
- 2 +          1101 +
- 3 =          1100 =
-----   ⟶     -------
- 5            11001 +
                   1 =
               -------
                1010
```

Somma di due numeri negativi.
Si è generato overflow tra le rappresentazioni.
Necessita correzione.

```
  5 +          0101 +
- 2 =          1101 =
-----   ⟶     -------
  3            10010 +
                   1 =
               --------
                0011
```

Somma di un numero positivo e un numero negativo.
Il risultato è positivo.
Si è generato overflow tra le rappresentazioni.
Necessita correzione.

```
  3 +          0011 +
- 4 =          1011 =
-----   ⟶     ------
- 1            1110
```

Somma di un numero positivo e un numero negativo.
Il risultato è negativo.
Non si è generato overflow tra le rappresentazioni.
Non necessita alcuna correzione.

Rappresentazione eccesso-k

- La rappresentazione in eccesso-k costituisce un metodo diverso da quello dei resti in modulo per ricondurre i numeri negativi a positivi
- In particolare, tutti i numeri sono traslati “verso l’alto” di k, che viene scelto maggiore o uguale al numero più piccolo da rappresentare

$$X = x + k$$

Rappresentazione eccesso-k: proprietà

- Analogamente al caso dei complementi diminuiti, la somma va corretta aggiungendo o sottraendo la costante k, e quindi in maniera sufficientemente semplice
- Moltiplicazioni e divisioni risultano invece più complesse
- Il vantaggio di tale codifica è che viene conservata la proprietà della disuguaglianza sulle rappresentazioni:

$$x_1 > x_2 \Leftrightarrow X_1 > X_2$$

- Questa rappresentazione, perciò, è utilizzata soltanto laddove siano richieste fondamentalmente somme algebriche e confronti logici fra gli operandi
- Tipicamente si utilizza per rappresentare gli esponenti nella rappresentazione in virgola mobile (prossima lezione)

Esempio

Rappresentazione in eccesso-8 su 4 bit

n=4

$V = [-8,7] \cap Z$

Codifica:

$X = x + k$

x	X_2	X_{10}
7	1111	15
6	1110	14
5	1101	13
4	1100	12
3	1011	11
2	1010	10
1	1001	9
0	1000	8
-1	0111	7
-2	0110	6
-3	0101	5
-4	0100	4
-5	0011	3
-6	0010	2
-7	0001	1
-8	0000	0