Corso di Calcolatori Elettronici I
A.A. 2012-2013

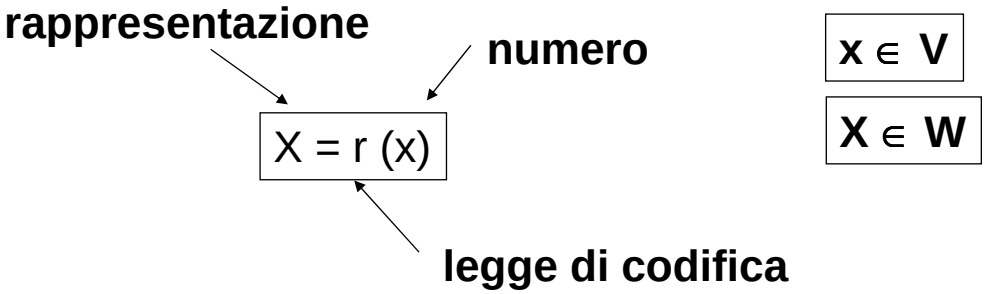
Rappresentazione dei numeri interi in un calcolatore

Prof. Roberto Canonico

Università degli Studi di Napoli Federico II
Dipartimento di Ingegneria Elettrica e
delle Tecnologie dell'Informazione
Corso di Laurea in Ingegneria Informatica (allievi A-DE)
Corso di Laurea in Ingegneria dell'Automazione

Rappresentazione dei numeri

- 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+ 8= --- 14 →	0110+ 1000= ---- 1110	11 – 5 = --- 6 →
		1011– 0101= ---- 0110
		0101 × 0011 = ---- 0101 0101= 0000== 0000=== ---- 0001111
14+ 3= --- 17 →	1110 + 0011 = ---- 10001 overflow	9– 7= --- 2 →
		1001– 0111= ---- 0010

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: $X = |x| + 2^{n-1} * \text{sign}(x)$
  - 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] ∩ 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

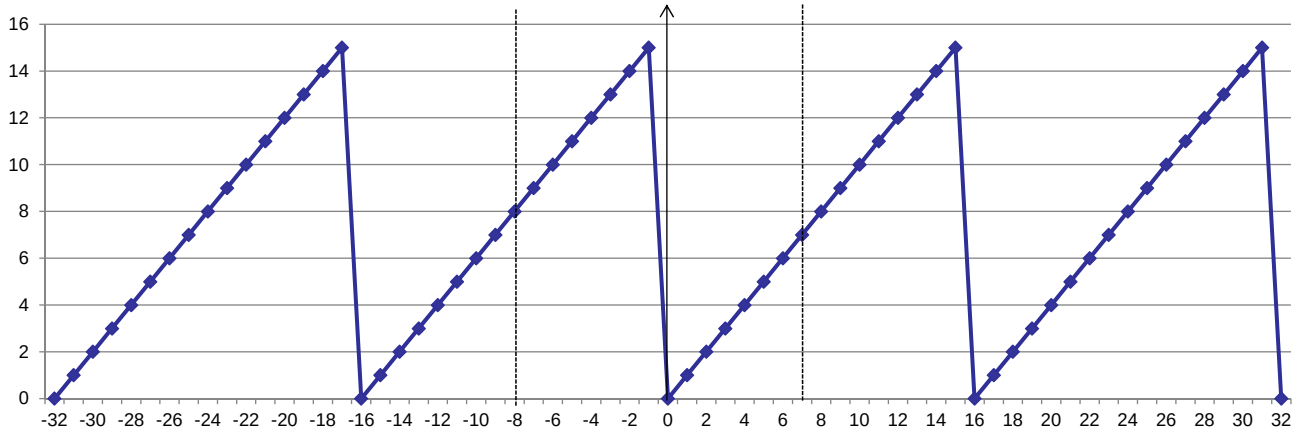
- Detto r un numero reale, si definisce intero di r il massimo intero y ≤ r

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

– 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.

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

- 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

```
   2 +        0010 +
  -6 =        1010 =
--------    --------
  -4    →     1100
```

```
  -2 +         1110 +
  -3 =     →   1101 =
--------      -------
  -5          11011

si ignora (il primo 1)     somma
                           modulo-16 (1011)
```

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*
- Questa ulteriore caratteristica consente di 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
  - -2 = $(110)_2$ con 3 bit diventa $(111110)_2$ su 6 bit
- Nota: questa operazione viene eseguita quando si fa in C un typecast da tipo short int ad int