RAPPRESENTAZIONE DEI DATI

Prof. Francesco Viglietti

Sistemi Classe 3 ASI a.s. 2012-2013

Rappresentazione numeri in un computer

- Usa la notazione binaria
- Ogni numero viene rappresentato con un numero finito di cifre binarie (*bit*)
- Numeri di tipo diverso hanno rappresentazioni diverse
  - es. Naturali, Interi, Razionali, Reali, Complessi
- Alcuni termini utili:
  - *byte* : una sequenza di 8 bit
  - *word* (parola) : 2 o 4 byte (dipende dalla macchina) unità minima che può essere fisicamente letta o scritta nella memoria
- Tipicamente gli interi positivi si rappresentano usando 2 o 4 byte
- Notazione

Numeri Relativi

Interi positivi e negativi, (Relativi) ci sono diverse convenzioni di rappresentazione:

- *modulo e segno* in cui il primo bit viene riservato al segno (1 negativo, 0 positivo) e gli altri 31 al modulo
- *Complemento a due*
- *Complemento a uno* (la trascuriamo)
- rimane comunque il problema dell'overflow

- Modulo e segno (es. con 3 bit) [0 segno+; 1 segno-]
  - codifica semplice
  - operazioni aritmetiche complesse

es.: +2 ↔ 010 e -2 ↔ 110

001 +	1 +
110 =	-2 =
111	-3

Occorre differenziare tra i bit del numero e quelli di segno. Bisogna codificare in modo diverso le operazioni aritmetiche.

Notazione Complemento a 2

- Complemento a due (es con 4 bit)

  es: +5 = 0101 -5 ??

Partendo da +5=0101 si invertono gli 1 con gli 0: 1010
→Si aggiunge 1: 1010+1=1011= -5

$-1x2^3+0x2^2+1x2^1+1x2^0 = -8+0+2+1=-5$

Il primo bit non rappresenta solo il segno!

Non occorre più pertanto differenziare i bit.

-5 con quattro bit (il bit di segno è 1)

Conversione: $5_{10} = 0101_2$

Inversione: 0101 → 1010

Somma di 1: 1010 + 1 = 1011

Verifica: + 5 → 0101

– 5 → 1011

= 0 = (1)0000

Conversione da complemento a 2 in decimale con segno

- se prima cifra 0 → numero positivo → conversione solita (es. 0100 → +4)
- se prima cifra 1 → numero negativo → inversione dei bit (tranne il primo) → conversione da binario a decimale → aggiungere 1
- Esempio: 1101
  - tolgo il bit di segno → 101
  - Inversione → 010
  - Conversione in decimale → $010_2 = 2_{10}$
  - Somma → 2 + 1 = 3
  - Segno → -3

Numeri Razionali

- numero finito di cifre periodiche dopo la virgola (ad esempio 3.12 oppure 3.453
- rappresentazione solitamente su 4/8 byte
- rappresentazione in *virgola fissa* : riservo X bit per la parte frazionaria
- es : con 3 bit per la parte intera e 2 per quella frazionaria 011.11, 101.01

- Come si converte in base 10 una rappresentazione in virgola fissa
  - es : $101.01 = 1 * 2^2 + 0 * 2^1 + 1 * 2^0 + 0 * 2^{-1} + 1 * 2^{-2}$

    $= 4 + 1 + 0.25 = 5.25$

    dove $2^{-1} = 1/2 = 0.5$, $2^{-2} = 1/2^2 = 0.25$) (in generale $2^{-n} = 1/2^n$)

Numeri Razionali: virgola fissa

- Problemi della rappresentazione in virgola fissa:
  - overflow
  - underflow quando si scende al di sotto del minimo numero rappresentabile
    - es. vediamo in base 10, con 2 cifre riservate alla parte frazionaria 0.01 / 2 = 0.005 non rappresentabile usando solo due cifre
  - spreco di bit per memorizzare molti '0' quando lavoro con numeri molto piccoli o molto grandi
    - es. vediamo in base 10, con 5 cifre per la parte intera e 2 cifre riservate alla parte frazionaria

    10000.00 oppure 00000.02
  - i bit vengono usati più efficientemente con la notazione *esponenziale* o *floating point* (*virgola mobile*)

Numeri razionali: virgola mobile

- Rappresentazione in virgola mobile
  - se lavoro con numeri molto piccoli uso tutti i bit disponibili per rappresentare le cifre dopo la virgola e se lavoro con numeri molto grandi le uso tutte per rappresentare le cifre in posizioni elevate. Questo permette di rappresentare numeri piccoli con intervalli minori fra loro rispetto ai numeri grandi, e riduce gli errori nel calcolo a parità di bit utilizzati
  - ogni numero N è rappresentato da una coppia (*mantissa* M, *esponente* E) con il seguente significato $N = M * 2^E$

Esempi:

- 1. in base 10, con 3 cifre per la mantissa e 2 cifre per l'esponente riesco a rappresentare $349000000000 = 3{,}49 * 10^{11}$
  con la coppia (3.49,11) perché M = 3.49 ed E = 11
- 2. in base 10, con 3 cifre per la mantissa e 2 per l'esponente riesco a rappresentare $0.000000002 = 2.0 * 10^{-9}$
  con la coppia (2.0,-9) perché M = 2.0 ed E = -9
- sia 0.000000002 che 349000000000 non sono rappresentabili in virgola fissa usando solo 5 cifre decimali

Standard IEEE

- *Precisione singola* su 32 bit
  - 1 bit di segno
  - 8 di esponente (da -126 a +127)
  - 23 di mantissa
  - Si possono rappresentare valori fino a 2 elevato a (-150)
- *Precisione doppia* su 64 bit
  - 1 bit di segno
  - 11 di esponente (da -1022 a +1023)
  - 52 di mantissa
  - Si possono rappresentare valori fino a 2 elevato a (-1075)

Ulteriore sistema di codifica dei numeri

BCD (Binary-Coded Decimal)

- Si codificano in binario (4 bit) le singole cifre decimali.
- es.: 254

$$\overbrace{0010}^{2}\ \overbrace{0101}^{5}\ .\ \overbrace{0100}^{4}$$

- nessun errore di conversione
- precisione dei calcoli decimali
- spreco di cifre
- usato nelle calcolatrici tascabili

Rappresentazione di un insieme finito di oggetti

- Vogliamo rappresentare i giorni della settimana :
  {Lu, Ma, Me, Gio, Ve, Sa, Do}
  usando sequenze 0 e 1
- Questo significa costruire un ‘codice’, cioè una tabella di corrispondenza che ad ogni giorno associa una opportuna sequenza
- In principio possiamo scegliere in modo del tutto arbitrario….

Rappresentazione di un insieme finito di oggetti (2)

- Per rappresentare 7 oggetti diversi servono almeno 3 bit (minima potenza di due che supera 7 è 8= 2^3) quindi :

000 Lunedì	110 Domenica
001 Martedì	111 *non ammesso*
010 Mercoledì	
011 Giovedì	
100 Venerdì	
101 Sabato	

Rappresentazione di caratteri e stringhe

- Tipologia di caratteri:
  - alfabeto e interpunzioni: A, B, .., Z, a, b, .., z, ;, :, “, ..
  - cifre e simboli matematici: 0, 1, ..., 9, +, -, >, ..
  - caratteri speciali: £, $, %, ...
  - caratteri di controllo: CR, DEL, ...
- Le stringhe sono sequenze di caratteri terminate in modo particolare.
- I caratteri sono un insieme finito di oggetti e seguono la strategia vista per i giorni della settimana

Rappresentazione di caratteri e stringhe (2)

- **ASCII** (*American Standard Code for Information Interchange*): Codice a 7 bit (standard)
- **ASCII** esteso a 8 bit (non standard)
- es.: A 01000001
  (00101000
- **UNICODE**: su 16 bit (65536 diverse configurazioni): più recente, permette di rappresentare anche alfabeti diversi e simboli per la scrittura di lingua orientali.

Rappresentazione di caratteri e stringhe (3)

- **ASCII a 7 bit**

I 7 bit sono suddivisi logicamente in 7 campi rispettivamente di 3 e 4 bit.

I primi tre bit rappresentano categorie di caratteri, mentre gli ultimi quattro servono a rispettare l' ordinamento dei caratteri all' interno di ogni categoria.

Rappresentazione di caratteri e stringhe (4)

1°bit	2°bit	3°bit	Caratteri rappresentati
0	1	0	simboli di punteggiatura, simboli speciali e di operazione
0	1	1	simbolo di =
0	1	1	numerali
1	0	0	maiuscole (A - O)
1	0	1	maiuscole (P - Z)
1	1	0	minuscole (a - o)
1	1	1	minuscole (p - z)