Appunti di informatica

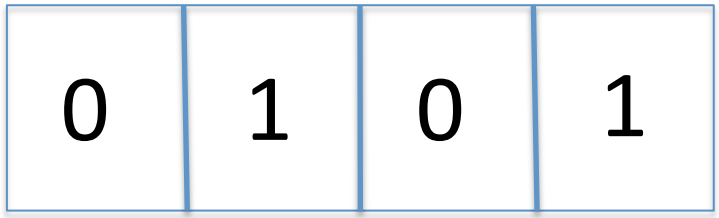
Lezione 4

anno accademico 2016-2017

Mario Verdicchio

Numeri binari in memoria

- In un calcolatore, i numeri binari sono tipicamente memorizzati in sequenze di caselle (note anche come parole) di lunghezza fissa dipendente dalla struttura del calcolatore stesso.
- Ad esempio, una parola di 4 bit può contenere il numero 0101_2



Dimensioni delle memorie

- 8 bit = 1 Byte (1B)
- 2^{10} Byte = 1024 Byte = 1 KiloByte (1KB)
- 2^{20} Byte = 1048576 Byte =
  1 MegaByte (1MB)
- 2^{30} Byte = 1073741824 Byte =
  1 GigaByte (1GB)
- 2^{40} Byte = 1099511627776 Byte =
  1 TeraByte (1TB)

Dimensioni delle memorie (2)

- Oramai è sempre più diffusa la seguente approssimazione:
- 1000 (10^3) Byte (o bit) = 1 KiloByte o KiloBit (1KB o 1Kb)
- 1000000 (10^6) Byte (o bit) = 1 MegaByte o MegaBit (1MB o 1Mb)
- 1 miliardo (10^9) Byte (o bit) = 1 GigaByte o GigaBit (1GB o 1Gb)
- 1000 miliardi (10^{12}) Byte (o bit) = 1 TeraByte o TeraBit (1TB o 1Tb)

Combinazioni possibili di numeri

- Una parola di 4 bit può contenere $2^4 = 16$ numeri binari diversi: da 0000 a 1111
- In generale, una parola di n bit può contenere 2^n numeri binari diversi

Interpretazioni possibili dei numeri

- Se non ci preoccupiamo del segno dei numeri, e li consideriamo sempre positivi, la sequenza che va da 0000_2 a 1111_2 corrisponde ai numeri da 0_{10} a 15_{10}
- In generale, data una parola da n bit e interpretando i numeri binari come numeri senza segno, solo positivi, i numeri esprimibili con tale parola vanno da 0 a 2^n-1

Numeri con segno

- Se vogliamo introdurre anche i numeri negativi, una possibililità è di usare il primo bit a sinistra per esprimere il segno del numero: 0 sta per +, 1 sta per -
- Con questa convenzione, chiamata “modulo e segno”, $1010_2 = -2_{10}$, e $0111_2 = 7_{10}$
- In generale, con una parola di n bit si possono esprimere i numeri compresi tra $-(2^{n-1} - 1)$ e $2^{n-1} - 1$

Complemento a due

- La rappresenazione “modulo e segno” ha un inconveniente: ci sono due rappresentazioni per 0_{10}: ad esempio, se si hanno parole da 4 bit, sia 0000_2 sia 1000_2 corrispondono a 0_{10}
- Con la rappresentazione “in complemento a due” si ovvia a questo problema: 0_{10} si rappresenta solo con 0000_2, $+1_{10}$ come al solito con 0001_2 mentre per ottenere la rappresentazione binaria di -1_{10}, si procede come segue

Da numero positivo a negativo (1)

- Data la rappresentazione binaria di $+1_{10}$: 0001_2
- La rappresentazione di si ottiene così:
  - si invertono tutti i bit
  - si somma 1
- Quindi, la rappresentazione in complemento a due di -1_{10} è: 1111_2

Da numero positivo a negativo (2)

- Analogamente per $+2_{10}$:
  0010_2
- La rappresentazione in complemento a 2 di per -2_{10} è:
  1110_2
- E così via fino a utilizzare tutte le configurazioni di bit possibili della parola
- Con una parola da n bit, in complemento a due si possono rappresentare i numeri compresi tra tra -2^{n-1} e $2^{n-1} - 1$ (da notare che c'è un numero in più grazie al fatto che 0_{10} ha un'unica rappresentazione)

Metodo alternativo

- L'inversione di tutti i bit e la somma di 1 costituiscono un metodo per scoprire, dato un numero positivo $+n_2$, la codifica binaria del suo opposto: $-n_2$ (o anche viceversa: dato $-n_2$, scoprire $+n_2$)
- In alternativa, si può usare la seguente regola:

  partendo da destra, lascia tutto intatto fino al primo '1' incluso, poi inverti tutto il resto

Somma di numeri binari

- Sui numeri binari si effettuano le classiche operazioni aritmetiche esattamente come per i numeri in base 10
- In particolare, la somma si esegue bit per bit, con le seguenti regole:

  0 + 0 = 0

  1 + 0 = 0 + 1 = 1

  1 + 1 = 0 con carry (o riporto) di 1 a sinistra

Overflow

- Si ha un overflow quando si sommano 2 numeri contenuti in parole da n bit e il risultato non riesce ad essere rappresentato in n bit
- Ad esempio:

```
 0111 +
 0110 =
------
 1101
```

Questa somma non è valida perché sommando due numeri positivi su 4 bit si ottiene un numero negativo, il che è assurdo. Il risultato corretto sarebbe 01101 ma per poterlo rappresentare servono 5 bit invece che 4. Una tecnica per controllare se c'è overflow è di vedere i riporti alla posizione più a sinistra e della posizione a sinistra di essa (vedi le stelle): se i riporti sono diversi c'è overflow. In questo caso c'è riporto alla posizione del quarto bit da destra ma non alla sua sinistra, e infatti c'è overflow.

Sottrazione

- Con il complemento a due, la sottrazione non è altro che sommare con un numero negativo
- 6 – 4 equivale a calcolare 6 + (-4):

$6_{10} = 0110_2$

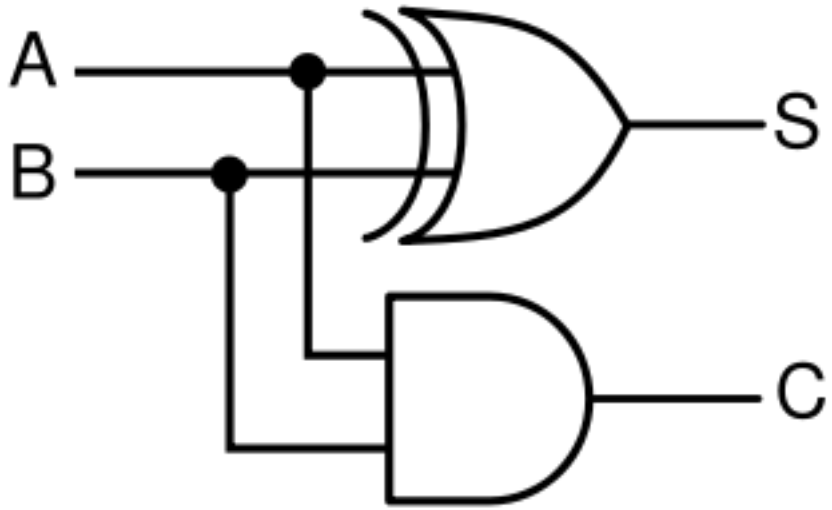
$4_{10} = 0100_2$

$-4_{10} = 1100_2$

$2_{10} = 0010_2$

Circuiti sommatori (1)

- Half-adder (A+B = Somma con Carry)



Circuiti sommatori (2)

- Full-adder (A+B+Carry in ingresso = Somma con Carry in uscita)