

Università di Bologna
Corso di Laurea in Scienze di Internet

Architettura di Internet

Luciano Bononi

E-mail: bononi@cs.unibo.it

Pagina web: <http://www.cs.unibo.it/~bononi/>

Orari di lezione:

Mercoledì ore 15.30-18.30, Giovedì ore 15.30-18.30

Ricevimento:

Martedì ore 9.30-11.00, Mercoledì ore 14.00-15.00

Alcune lezioni saranno svolte dal Dott. Gabriele D'Angelo

© 2004 Luciano Bononi

1

Testi consigliati...

...come complemento alle slide di lezione, al materiale su Web.

Per la parte di architettura del calcolatore (* circa 20% del corso)

**William Stallings, *Computer Organization and Architecture, 5/6-th edition*
Prentice Hall, 2003, ISBN: 0-13-035119-9**

**Andrew Tanenbaum, *Structured Computer Organization, Fourth Edition*
Prentice Hall (1999) ISBN: 0-13-095990-1**

Per la parte di reti di calcolatori e Internet (* circa 65% del corso)

**James F. Kurose, Keith W. Ross, *Internet e reti di Calcolatori*,
McGraw-Hill, (circa 43 euro)
ISBN: 88 386 6011-5 prima edizione (blu),
88 386 6109-X seconda edizione (gialla)**



...altri testi e fonti su WEB saranno citati in seguito.

© 2004 Luciano Bononi

2

Informazioni generali

▪ Cosa cambia rispetto all'anno scorso?

- programma del corso: varia del 25%
 - 20% Architettura del calcolatore
 - Rappresentazione dell'informazione binaria
 - Architettura classica del calcolatore (von Neumann)
 - Funzionamento della CPU, memoria, dispositivi di I/O
 - Trasferimento dati interno all'architettura: Bus
 - Trasferimento dati in rete: dispositivi di rete
 - 20% Sistema operativo e architettura software del calcolatore
 - Funzioni del S.O.
 - Linguaggi di alto e basso livello, applicazioni e loro esecuzione
 - 60% Reti locali, reti di reti (internetworking), Internet
 - Struttura e componenti fisiche
 - Struttura e componenti logiche (protocolli)
 - Servizi
 - Evoluzione? Internet2, Wireless Internet...

Informazioni generali

▪ Come cambia l'esame?

- esame tipo WebTest mid-term (singolo appello, intorno a metà Aprile 2004)
 - facoltativo, con iscrizione obbligatoria via web
 - circa 15 domande a risposta multipla + 3-4 domande aperte (max 30 minuti)
 - valutazione da 0 a + 3 punti, valevole come "orale virtuale"***
- Esame finale tipo WebTest (normali appelli)
 - con iscrizione obbligatoria via web (*regole per iscrizione e partecipazione)
 - circa 45 quiz a risposta multipla, 20 minuti, valutazione immediata in trentesimi
 - Si può eseguire una sola volta per ogni appello -> max 2 volte per sessione
 - Valutazione inferiore a 18/30 -> esame non superato, si ripete WebTest
 - Valutazione tra 18/30 e 27/30 -> orale obbligatorio, oppure si ripete WebTest
 - Valutazione superiore a 27/30 -> orale facolt. (o mid-term**), oppure si ripete Webtest
 - Se si ripete il webtest si cancella la valutazione precedente
 - ...quindi non si può venire a reclamare un voto migliore precedente (!)
- Le domande del webtest attuale saranno completamente revisionate

Carrellata del programma

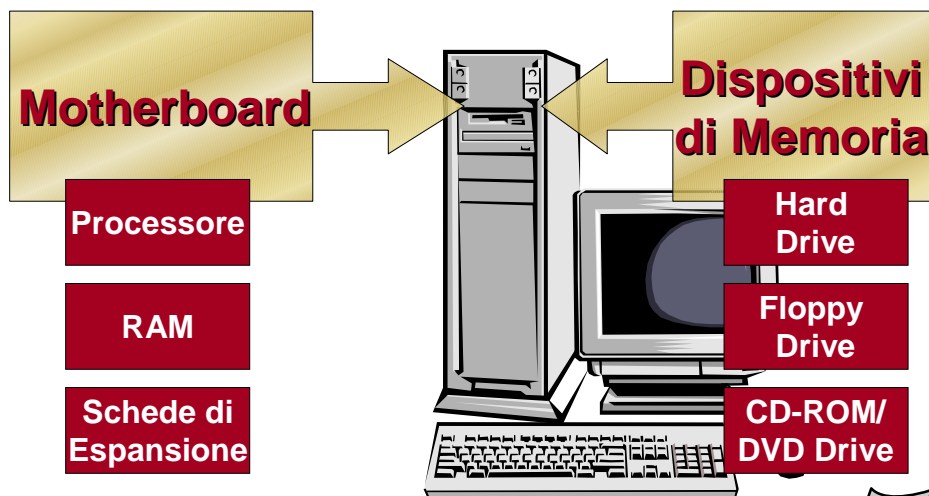
▪ Cosa faremo da grandi?... ☺ Saprete orientarvi riguardo ai temi...

- L'Elaboratore Elettronico, le sue componenti Hardware e la sua architettura
- il Software, la rappresentazione dei dati e dei programmi e la loro esecuzione
- Il Sistema Operativo: caratteristiche, problematiche e servizi offerti
- Reti locali, reti di reti (internetworking) e Internet
- Architettura e componenti HW e SW delle Reti di Calcolatori
 - Componenti di rete
 - Lo Stack dei protocolli (Iso/Osi)
- Componenti, Architettura e Servizi Fondamentali di Internet
 - Problematiche: indirizzamento, routing, prestazioni, accesso, sicurezza...
 - i protocolli di rete locale e Internet: es. Ethernet, ATM, IP, TCP, UDP,
 - I protocolli dei servizi di rete: es. ICMP, HTTP, SMTP, IMAP, POP3, FTP, Telnet...
 - i servizi: World Wide Web, e-mail, News, Chat, file-sharing...
 - protocolli, formati dei dati e argomenti connessi: HTML, Unicode, MIME, ...
 - i paradigmi architetturali: Client/Server, Peer to peer (P2P), agent-based....
 - Sicurezza delle reti di calcolatori: problemi, soluzioni, protocolli
- Cenni su Internet2, Servizi differenziati e Qualità del Servizio (QoS)
- Cenni sulle reti Wireless e Wireless Internet

© 2004 Luciano Bononi

5

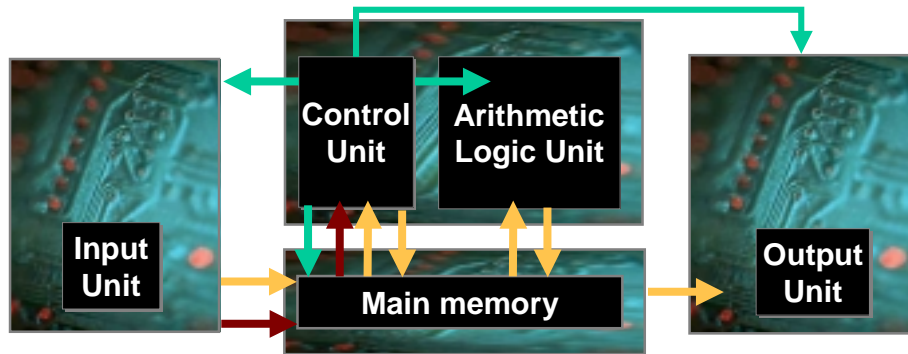
PC System Unit



© 2004 Luciano Bononi

6

Cosa succede all'interno della motherboard?



→ Control flow → Instruction flow → Data flow

© 2004 Luciano Bononi

7

Componenti del microprocessore

Unità di Controllo

- Legge e Interpreta le istruzioni del programma
- Dirige le operazioni dei componenti interni
- Controlla il flusso di dati/istruzioni da/verso RAM

Decoder

Program Register

Instruction Register

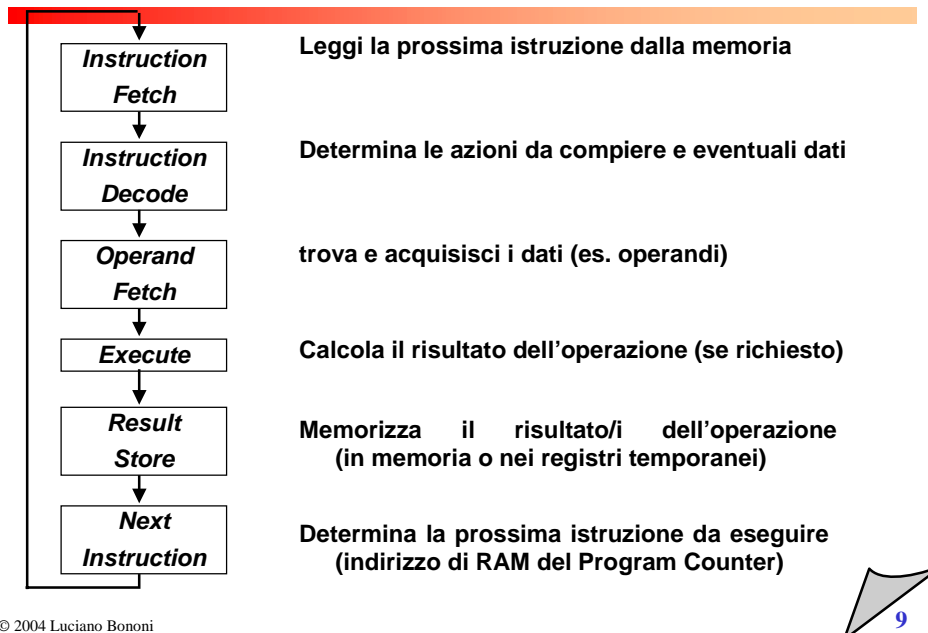
Accumulator

Unità Aritmetico/logica (ALU)

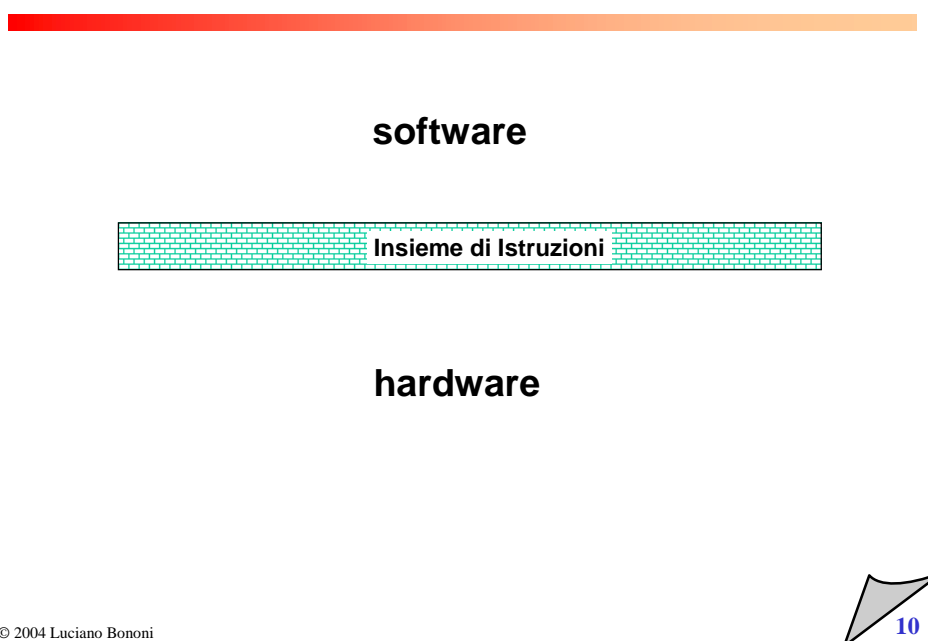
© 2004 Luciano Bononi

8

Ciclo di esecuzione dell'istruzione



Insieme di istruzioni: interfaccia critica



Il Sistema Operativo e le applicazioni

- Servizi del sistema operativo
- Applicazioni utente

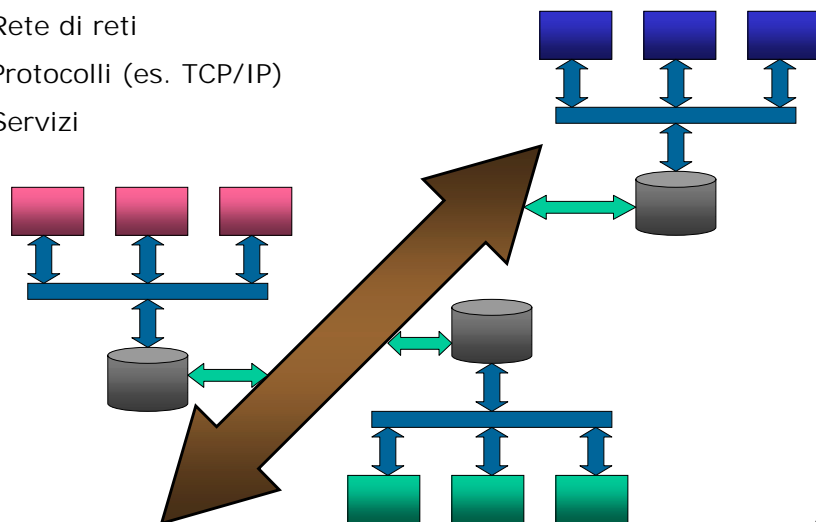
◦
◦

Comunicazione in INTERNET

Rete di reti

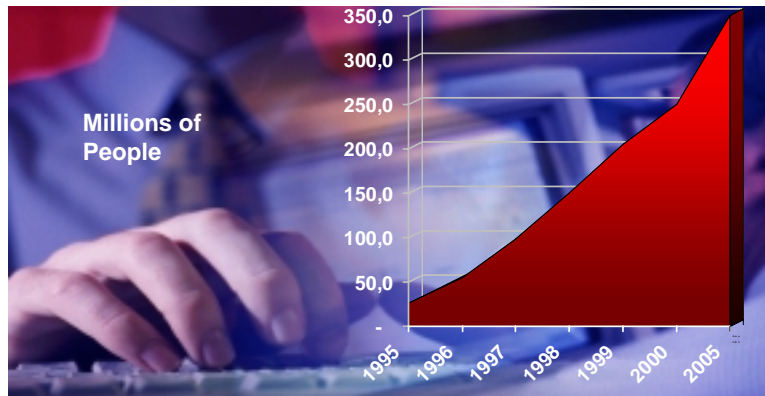
Protocolli (es. TCP/IP)

Servizi



Servizi di Internet

Architetture: C/S, P2P...
World Wide Web (WWW)
e-mail, News, Chat, File Sharing

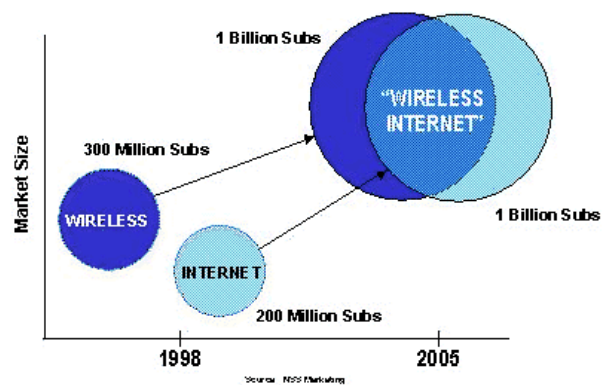


© 2004 Luciano Bononi

13

Wireless Internet

- **Evoluzione della rete Internet e dei servizi Internet verso il mondo wireless**



© 2004 Luciano Bononi

14

Introduzione all'Architettura del Calcolatore

Luciano Bononi

bononi@cs.unibo.it

<http://www.cs.unibo.it/~bononi/>

Figure credits: some of the figures have been taken and modified from existing figures found on the web

© 2004 Luciano Bononi

15

Introduzione all'Architettura del Calcolatore

Luciano Bononi

bononi@cs.unibo.it

<http://www.cs.unibo.it/~bononi/>

Figure credits: some of the figures have been taken and modified from existing figures found on the web

© 2004 Luciano Bononi

16

Contenuti di questo modulo

- **Gli argomenti affrontati includono :**
 - Architettura del calcolatore e tecnologia
 - Aritmetica del calcolatore e rappresentazione dati
 - Insiemi di istruzioni del calcolatore
 - es. di progettazione di un processore
 - in emulazione (simulazione) circuitale
 - a partire da un set minimale di istruzioni
 - es. di realizzazione di un semplice programma
 - in esecuzione sull'architettura del processore+memoria

Cos'è l'architettura di un calcolatore?

- **L'architettura di un calcolatore è la progettazione di un calcolatore a livello delle interfacce HW e SW**
- **Architettura del Calcolatore = insieme di istruzioni + organizzazione della macchina**

Architettura del calcolatore

insieme di istruzioni

Interfaccia del PC

Interfaccia SW e HW

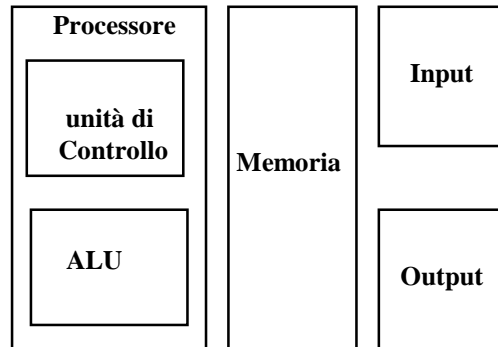
organizzazione macchina

Componenti HW

interfaccia logica

Visione strutturale di alto livello

- Queste sono le principali unità che costituiscono il calcolatore



© 2004 Luciano Bononi

19

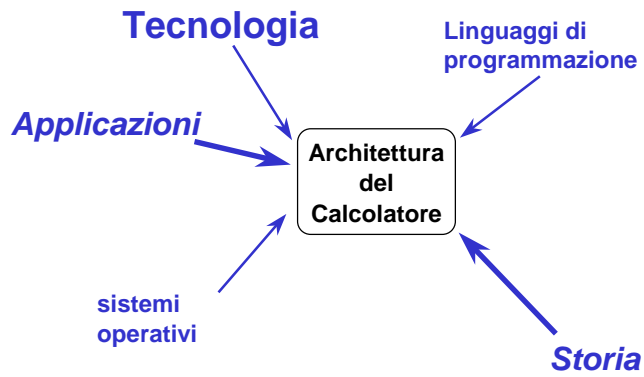
Componenti del calcolatore

- **Queste sono le principali funzioni svolte dalle componenti del calcolatore**
 - ALU – esegue operazioni aritmetico-logiche
 - es. Somme, moltiplicazioni, shift
 - memoria – mantiene dati e istruzioni (programmi)
 - es. memoria cache, memoria principale, dischi
 - input – acquisisce dati verso il computer
 - es. tastiera, mouse, scheda di rete
 - output – restituisce dati dal computer
 - es. schermo, scheda audio, scheda di rete
 - controllo – gestisce attività delle componenti
 - es. controllore Bus, Gestore unità di memoria (MMU)

© 2004 Luciano Bononi

20

Evoluzione dell'architettura del calcolatore



© 2004 Luciano Bononi

21

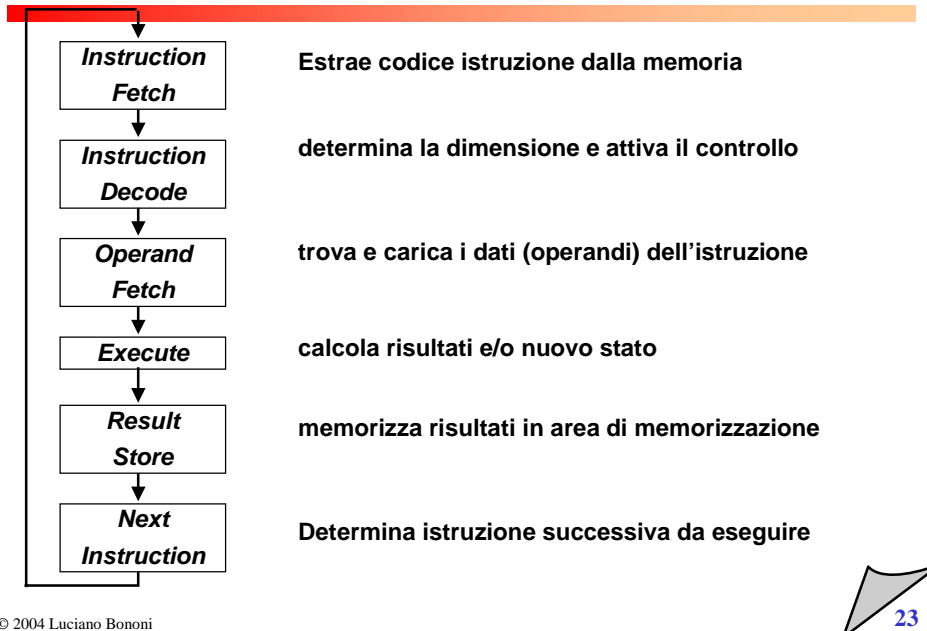
Insieme di istruzioni: interfaccia critica



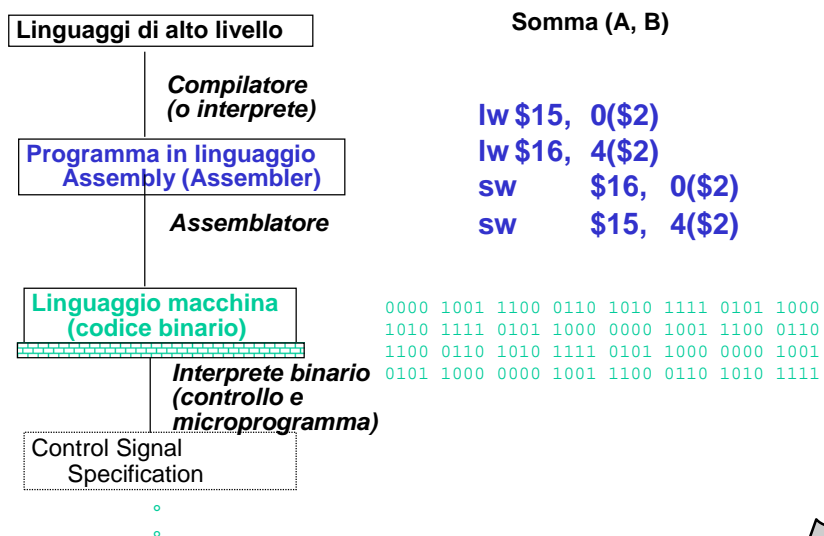
© 2004 Luciano Bononi

22

Ciclo di esecuzione istruzioni di una CPU

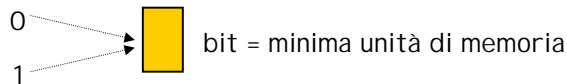


Livelli di rappresentazione delle istruzioni



La rappresentazione dei dati sul calcolatore

- **il calcolatore dispone di soli due simboli per la memorizzazione, rappresentazione e trasmissione di dati**
 - zero e uno (0,1) = valori assumibili da un binary digit (bit)
 - ogni valore deve essere rappresentato usando questi due simboli (notazione binaria)
 - esistono infinite interpretazioni (codifiche) che possono associare sequenze generiche di simboli 0 e 1 a valori numerici
 - noi vedremo alcune delle più significative



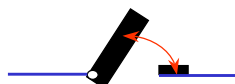
© 2004 Luciano Bononi

25

Il bit: valore e logica positiva (lp) e negativa (ln)



levetta:
alta = 1 (lp) 0(ln)
bassa = 0 (lp) 1(ln)



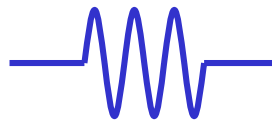
interruttore:
aperto = 0 (lp) 1 (ln)
chiuso = 1 (lp) 0 (ln)



lampadina:
spenta = 0 (lp) 1 (ln)
accesa = 1 (lp) 0 (ln)



tensione elettrica:
bassa = 0 (lp) 1 (ln)
alta = 1 (lp) 0 (ln)



corrente elettrica 50 Hz:
assente = 0 (lp) 1 (ln)
presente = 1 (lp) 0 (ln)

© 2004 Luciano Bononi

26

La rappresentazione dei dati sul calcolatore

▪ Il calcolatore deve poter memorizzare e elaborare dati e programmi

- simboli, valori numerici naturali e interi (positivi e negativi)
 - possono essere rappresentati con un numero finito di bit se appartenenti a un range opportunamente limitato
 - anche i caratteri possono essere mappati su valori interi
- valori numerici reali, razionali e irrazionali
 - possono essere rappresentati attraverso opportune codifiche binarie, se appartenenti a range opportunamente limitati e tollerando errori di approssimazione
- programmi espressi attraverso
 - istruzioni (codici) e operandi (dati)
 - es. (Somma 18 3) equivale a (+ 18 3)
 - un simbolo (cioè un valore) può rappresentare un'istruzione del programma memorizzato ed eseguito

© 2004 Luciano Bononi

27

La rappresentazione dei dati sul calcolatore

▪ i bit possono essere considerati in sequenza (in memoria)

- sequenza di 8 bit = 1 Byte



0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 1
0 0 0 0 0 0 1 0
0 0 0 0 0 0 1 1

⋮

1 1 1 1 1 1 1 0
1 1 1 1 1 1 1 1

2ⁿ possibili sequenze diverse da n bit

n = numero di bit considerati

© 2004 Luciano Bononi

28

La rappresentazione dei dati sul calcolatore

▪ Dato un byte come possiamo associarvi i simboli o valori?

- sequenza di 8 bit = 1 Byte (primo esempio poco utile)



| | | | | | | | | |
|---|---|---|---|---|---|---|---|------------|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | = valore 1 |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | = valore 2 |
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | = valore 3 |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | = valore 4 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | = valore 5 |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | = valore 6 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | = valore 7 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | = valore 8 |

Solo valori da 1 a 8 ?
non sfruttato tutte le
possibili combinazioni!!!



La rappresentazione dei dati sul calcolatore

▪ Dato un byte come possiamo associarvi i simboli o valori?

- sequenza di 8 bit = 1 Byte



| | | | | | | | | |
|---|---|---|---|---|---|---|---|--------------|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | = valore 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | = valore 1 |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | = valore 2 |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | = valore 3 |
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | = valore 4 |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | = valore 254 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | = valore 255 |

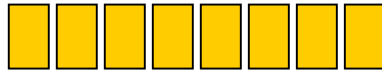
Idea: sfruttare tutte le possibili
combinazioni diverse di 8 bit
256 valori [0..255]



La rappresentazione dei dati sul calcolatore

▪ Dato un byte come possiamo associarvi i simboli o valori?

- sequenza di 8 bit = 1 Byte



Idea: sfruttare tutte le possibili combinazioni diverse di 8 bit
256 simboli

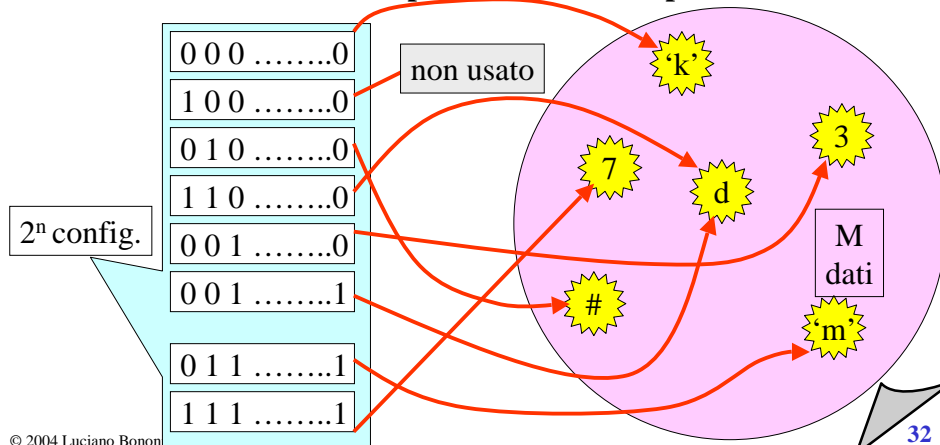
| | | | | | | | | |
|---|---|---|---|---|---|---|---|--------------|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | = simbolo A |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | = simbolo B |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | = simbolo C |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | = simbolo E |
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | = simbolo F |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | = simbolo % |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | = simbolo \$ |

Codici binari: convenzioni per rappresentare info.

Funzioni dall'insieme delle 2^n configurazioni di n bit ad un insieme di M informazioni o dati

(valori, simboli, istruzioni, ecc.).

Condizione necessaria per la codifica completa: $2^n \geq M$



Codici binari: quanti sono?

La **scelta** di un codice è condivisa da **sorgente** e **destinazione** ed ha due gradi di libertà:

- il **numero di bit n** (qualsiasi, a patto che sia $2^n \geq M$)



- l'**associazione** tra configurazioni e informazioni; a parità di n e di M le associazioni possibili sono

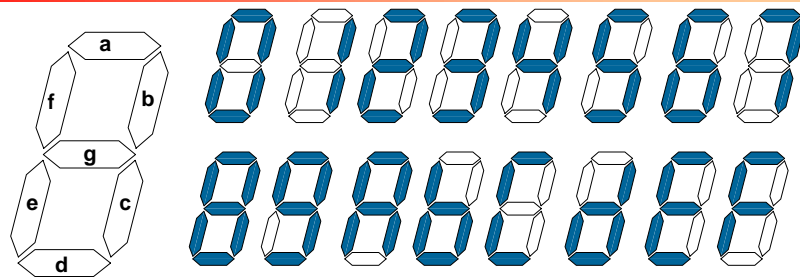
$$C = 2^n! / (2^n - M)!$$

| | |
|---------------|----------------------------|
| n = 1, M = 2 | C = 2 (logica pos. e neg.) |
| n = 2, M = 4 | C = 24 |
| n = 3, M = 8 | C = 64.320 |
| n = 4, M = 10 | C = 29.000.000.000 |

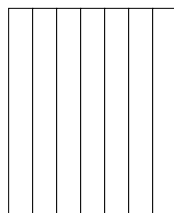
© 2004 Luciano Bononi

33

Es. codici speciali: codice 7 segmenti

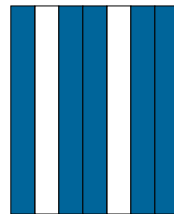


a b c d e f g



A=?
b=?
C=?
.....

a b c d e f g



... = ???

© 2004 Luciano Bononi

34

Sistemi di numerazione

Posizionali

• il valore di un simbolo dipende dalla posizione che esso occupa all'interno della configurazione, seguendo una legge nota. I vari sistemi di numerazione posizionale differiscono per la scelta della base B. La base B indica il numero di simboli usati.

- **decimale B=10, binario B=2, ottale B=8, esadecimale B=16**

Non posizionali

Il valore di un simbolo non dipende dalla posizione che esso occupa all'interno della configurazione. (es: Numeri Romani)

Interpretazione (funzione valore)

• Nei sistemi posizionali, i simboli di una configurazione possono essere interpretati come i coefficienti del seguente polinomio [1] (detto funzione Valore)

$$V = \sum_{i=-m}^{n-1} d_i \cdot B^i$$

B = base

d_i = i-esima cifra $\in [0..B-1]$

n = numero di cifre parte intera

m = numero di cifre parte frazionaria

La virgola e' posta tra le cifre di posizione 0 e -1.

Interpretazione (funzione valore)

Esempio: sistema decimale

Il numero **245.6** decimale può essere rappresentato come segue:

| | | | | |
|---------------|--------------------------------|--------|--------|-----------|
| B = 10 | base | | | |
| n=3 | numero cifre parte intera | | | |
| m=1 | numero cifre parte frazionaria | | | |
| d = {2,4,5,6} | insieme delle cifre | | | |
| cifra | 2 | 4 | 5 | 6 |
| posizione | 2 | 1 | 0 | -1 |
| peso | 10^2 | 10^1 | 10^0 | 10^{-1} |

$$V = \sum_{i=-m}^{n-1} d_i \cdot B^i = 2 \cdot 10^2 + 4 \cdot 10^1 + 5 \cdot 10^0 + 6 \cdot 10^{-1} = 245.6$$

© 2004 Luciano Bononi

37

Esempio di interpretazione valore binario naturale

Esempio: Quale è il valore decimale corrispondente al numero binario **1101.010₂** ?

| | | | | | | | | |
|--------------------|----------------|----------------|----------------|----------------|---|-----------------|-----------------|-----------------|
| cifra ₂ | 1 | 1 | 0 | 1 | . | 0 | 1 | 0... |
| peso | 2 ³ | 2 ² | 2 ¹ | 2 ⁰ | . | 2 ⁻¹ | 2 ⁻² | 2 ⁻³ |
| valore | 1•8 | 1•4 | 0•2 | 1•1 | . | 0•1/2 | 1•1/4 | 0•1/8 |

$$1101.010_2 = 1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 + 0 \cdot 2^{-1} + 1 \cdot 2^{-2} = 13.25_{10}$$

© 2004 Luciano Bononi

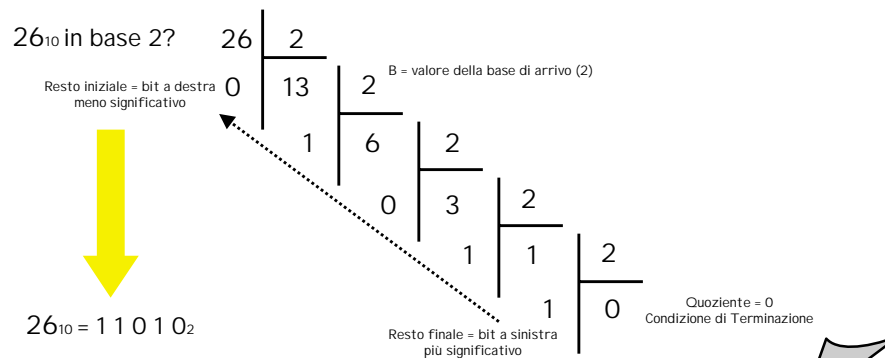
38

Metodo della divisione: Base 10 -> Base B

Es. base 2:

Per valori interi positivi:

Per ottenere il valore in base B, di un numero intero codificato nel sistema decimale, si procede utilizzando un metodo iterativo di successive divisioni per la base: al termine del procedimento i resti delle divisioni, dall'ultimo al primo, rappresentano il valore iniziale in base B.



© 2004 Luciano Bononi

39

Metodo della divisione: Base 10 -> binario naturale

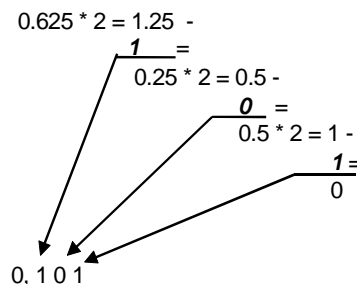
Es. voglio ottenere la parte decimale in base B=2

Per valori con cifre decimali: si separa la parte intera da quella frazionaria, La parte intera si calcola come nel caso precedente La parte frazionaria si ottiene come segue:

1. Si moltiplica la parte frazionaria per 2 il valore della base (2)
2. Se la parte intera del numero ottenuto è maggiore di 1, si sottrae il valore della parte intera (1), altrimenti si sottrae 0 e si prosegue
3. Si ripete dal passo 1 fino a che il numero ottenuto dopo la sottrazione è zero, oppure quando sono esaurite le cifre a disposizione

La sequenza dei valori sottratti ad ogni passo è la rappresentazione decimale ottenuta

Esempio: 0.625₁₀ = 0.101₂



© 2004 Luciano Bononi

40

Somma di valori in binario naturale

Assumiamo di avere solo $n=3$ bit a disposizione per rappresentare i valori e il risultato, quindi posso rappresentare $B^n = 8$ valori diversi interi positivi (da 0 a 7).

$$5_{10} + 1_{10} = 6_{10}$$

$$\begin{array}{r} \color{red}{1} \\ 101 + \\ 001 = \\ \hline 110 \end{array}$$

$$2_{10} + 3_{10} = 5_{10}$$

$$\begin{array}{r} \color{red}{1} \\ 010 + \\ 011 = \\ \hline 101 \end{array}$$

$$5_{10} + 3_{10} = 8_{10}$$

$$\begin{array}{r} \color{red}{111} \\ 101 + \\ 011 = \\ \hline 1000 \end{array}$$



Overflow!!

Per rappresentare il risultato della somma di $5_{10} + 3_{10}$ sono necessari 4 bit !

© 2004 Luciano Bononi

41

Altre basi: ottale e esadecimale

Quando per la rappresentazione di un numero si utilizzano molte cifre binarie può convenire usare altri sistemi di numerazione.

I sistemi **ottale** ed **esadecimale** sono utilizzati principalmente per rappresentare in modo più compatto i numeri binari.

L'algoritmo della divisione per passare da base 10 a ottale o esadecimale vale anche in questo caso. Provare.

I simboli del sistema **Ottale** sono 8 (da 0 a 7):

{ 0, 1, 2, 3, 4, 5, 6, 7, 10, 11, 12, 13, 14, 15, 16, 17, 20, ... }

I simboli del sistema **Esadecimale** sono 16 (da 0 a F):

{ 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F, 10, 11, 12, 13, 14, ... }

© 2004 Luciano Bononi

42

Cambiamenti di base (metodo veloce)

Esiste un metodo veloce quando base di partenza BP e di arrivo BA sono esprimibili come $BA=BP^k$.

BP:Binario -> BA:Ottale , $K=3, (8=2^3)$

Per passare dalla codifica Binaria a quella Ottale, si raggruppano le cifre binarie a gruppi di 3 (**a partire da destra, allineandosi alla virgola**) e le si sostituiscono con una cifra del sistema ottale.

Esempio : $111001010_2 = 712_8$

Ottale -> Binario

Per passare dalla codifica Ottale a quella Binaria, si sostituisce ad ogni cifra ottale la corrispondente codifica binaria (composta da 3 cifre).

Esempio : $302_8 = 011000010_2$

© 2004 Luciano Bononi

43

Cambiamenti di base (metodo veloce)

BP: Binario -> BA:esadecimale , $K=4, (16=2^4)$

Per passare dal codice Binario a quello Esadecimale, si raggruppano le cifre a gruppi di 4 (a partire da destra) e le si sostituiscono con una cifra del sistema esadecimale.

Esempio : $100100011111_2 = 91F_{16}$

Esadecimale -> Binario

Per passare dal codice Esadecimale a quello Binario, si sostituisce ad ogni cifra esadecimale la corrispondente configurazione binaria (composta da 4 cifre).

Esempio : $A7F_{16} = 10100111111_2$

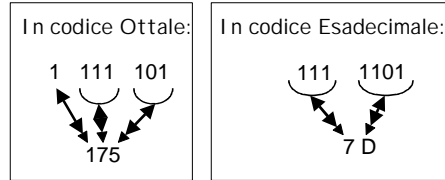
© 2004 Luciano Bononi

44

Esempi

Esempio 1

Codifica del numero $125_{10} = 1111101_2$



Esempio 2

| Decimale | Binario | Ottale | Esadecimale |
|----------|----------|--------|-------------|
| 5 | 101 | 5 | 5 |
| 12 | 1100 | 14 | C |
| 78 | 1001110 | 116 | 4E |
| 149 | 10010101 | 225 | 95 |

© 2004 Luciano Bononi

45

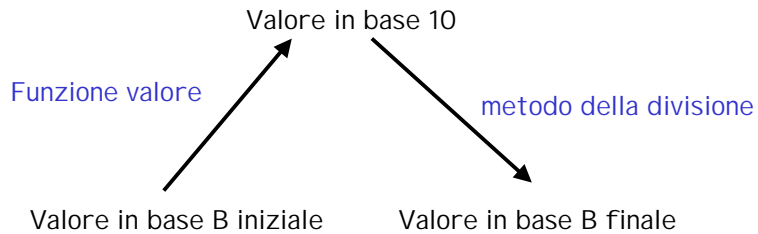
Esempi

| Decimale | Binario | Ottale | Esadecimale |
|----------|-------------|--------|-------------|
| | dcba | | |
| 0 | 0000 | 00 | 0 |
| 1 | 0001 | 01 | 1 |
| 2 | 0010 | 02 | 2 |
| 3 | 0011 | 03 | 3 |
| 4 | 0100 | 04 | 4 |
| 5 | 0101 | 05 | 5 |
| 6 | 0110 | 06 | 6 |
| 7 | 0111 | 07 | 7 |
| 8 | 1000 | 10 | 8 |
| 9 | 1001 | 11 | 9 |
| 10 | 1010 | 12 | A |
| 11 | 1011 | 13 | B |
| 12 | 1100 | 14 | C |
| 13 | 1101 | 15 | D |
| 14 | 1110 | 16 | E |
| 15 | 1111 | 17 | F |

© 2004 Luciano Bononi

46

Riassunto: passaggi di base generici



© 2004 Luciano Bononi

47

Codifica dei caratteri ASCII (7 bit -> 128 caratteri)

Il codice ASCII è non ridondante, perchè i simboli che vengono codificati sono in numero pari alle configurazioni ottenibili con 7 cifre binarie.

| | 000 | 001 | 010 | 011 | 100 | 101 | 110 | 111 | MSB |
|------|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| 0000 | NUL | DLE | | 0 | @ | P | ° | p | |
| 0001 | SOH | DC1 | ! | 1 | A | Q | a | q | |
| 0010 | STX | DC2 | “ | 2 | B | R | b | r | |
| 0011 | ETX | DC3 | # | 3 | C | S | c | s | |
| 0100 | EOT | DC4 | \$ | 4 | D | T | d | t | |
| 0101 | ENQ | NAK | % | 5 | E | U | e | u | |
| 0110 | ACK | SYN | & | 6 | F | V | f | v | |
| 0111 | BEL | ETB | ‘ | 7 | G | W | g | w | |
| 1000 | BS | CAN | (| 8 | H | X | h | x | |
| 1001 | HT | EM |) | 9 | I | Y | i | y | |
| 1010 | LF | SUB | * | : | J | Z | j | z | |
| 1011 | VT | ESC | + | ; | K | [| k | { | |
| 1100 | FF | FS | , | < | L | \ | l | | |
| 1101 | CR | GS | - | = | M |] | m | } | |
| 1110 | SO | RS | . | > | N | ^ | n | ~ | |
| 1111 | SI | US | / | ? | O | _ | o | DEL | |

LSB

© 2004 Luciano Bononi

48

Rappresentazione binaria di numeri interi

Nella rappresentazione binaria di numeri dotati di segno viene tipicamente usato un bit per discriminare tra valori positivi e valori negativi. Dati n bit per la rappresentazione il bit usato per il segno è quello più significativo (More Significant Bit (MSB), in posizione $n-1$).

Rappresentazione *Modulo e Segno (M-S)*

In questa rappresentazione al valore assoluto del numero viene *prefisso* un bit per indicarne il segno.

Il **valore 0** di questo bit codifica il segno *più* e il **valore 1** il segno *meno*.

Esempio $n=8$ (rappresento i valori su $n=8$ bit):

$$+57_{10} = 00111001_2 \quad -57_{10} = 10111001_2$$

↓ ↓
segno modulo

© 2004 Luciano Bononi

49

Rappresentazione modulo e segno

La rappresentazione *M-S* è vantaggiosa per la sua semplicità ma richiede circuiti complessi per l'esecuzione di somme algebriche.



Prima di eseguire una somma algebrica tra due operandi A e B è necessario determinare quale dei due è maggiore in valore assoluto.

- Se A è maggiore di B si esegue la differenza $A-B$ e si assegna al risultato il segno di A .
- Se A è minore di B si esegue la differenza $B-A$ e si assegna al risultato il segno di B .

N.B.

Usando n bit (es. 8) per la codifica, il range di valori rappresentabili risulta: $[-2^{n-1}-1..+2^{n-1}-1]$.
Un bit (MSB) è usato per il segno.

Es. con 8 bit sono rappresentabili valori nell'intervallo $[-127 ..+127]$.

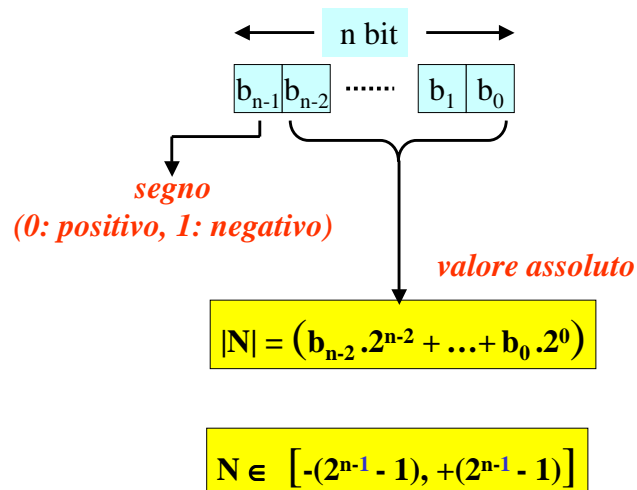
Vi sono due configurazioni per lo zero (00000000_2 e 10000000_2) e ciò rappresenta un problema quando l'architettura deve verificare la condizione di uguaglianza a zero.

Poiché le operazioni vanno eseguite sulla sola parte di valore assoluto è semplice la determinazione dell'**overflow**

© 2004 Luciano Bononi

50

Numeri relativi: rappres. Modulo e Segno (M-S)



© 2004 Luciano Bononi

51

esempio di operazioni (M-S)

$(23 + 16)_{10}$

Stesso segno, si esegue la somma escludendo il bit più significativo. Entrambi gli operandi hanno lo stesso segno (bit 7 = 0), quindi il segno viene mantenuto (bit 7 = 0)

$$\begin{array}{r}
 23_{10} + \quad \overset{1}{0010111_2} + \\
 16_{10} = \quad 0010000_2 = \\
 \hline
 39_{10} \quad 0100111_2
 \end{array}$$

Risultato : $+39_{10}$ 00100111_2

$(22 - 17)_{10}$

Il primo operando ha modulo maggiore del secondo ($|22| > |17|$) => si esegue la differenza tra 22 e 17. In questo caso il segno del risultato (5) è positivo e il bit 7 deve essere posto a 0.

$$\begin{array}{r}
 22_{10} - \quad \overset{1}{0010110_2} - \\
 17_{10} = \quad 0010001_2 = \\
 \hline
 5_{10} \quad 0000101_2
 \end{array}$$

Risultato : $+5_{10}$ 00000101_2

© 2004 Luciano Bononi

52

esempio di operazioni (M-S)

$(8 - 16)_{10}$

Il secondo operando ha modulo maggiore ($|16| > |8|$) del primo. Si esegue la differenza tra 16 e 8. Il segno e' quello dell'operando di valore assoluto maggiore. 16 è negativo e il bit 7 deve essere posto a 1.

$$\begin{array}{r} 16_{10} \quad 0010000_2 - \\ 8_{10} \quad 0001000_2 = \\ \hline 8_{10} \quad 0001000_2 \end{array}$$

Risultato -8_{10} 10001000_2

$(-112 - 39)_{10}$

Entrambi gli operandi di segno negativo => si sommano i valori assoluti.

$$\begin{array}{r} 112_{10} \quad 1110000_2 + \\ 39_{10} \quad 0100111_2 = \\ \hline 151_{10} \quad 10010111_2 \end{array}$$

Overflow. Sono necessari 8 bit per rappresentare 151 !

Usando la rappresentazione M-S sono disponibili per il modulo solo 7 bit => (Overflow).

Rappresentazione Complemento a uno (1's C)

Dati n bit per la codifica del modulo e del segno:

la rappresentazione di un numero negativo in complemento a 1 (1's C) si ottiene complementando tutti i bit della rappresentazione in binario naturale del corrispondente positivo, e viceversa

$$V = \sum_{i=0}^{n-2} d_i \cdot 2^i \quad \text{se } d_{n-1}=0 \quad \Downarrow \quad V = \sum_{i=0}^{n-2} (-d_i) \cdot 2^i \quad \text{se } d_{n-1}=1$$

- Con tale rappresentazione possono essere codificati i valori compresi nell'intervallo $[-2^{n-1}+1, 2^{n-1}-1]$.
- **Esistono due rappresentazioni per lo zero: es. per n=4, 0000, 1111**
- I numeri positivi restano inalterati, come nel binario naturale
- I numeri negativi sono calcolabili partendo dal corrispondente valore positivo, invertendo tutti i bit.

Rappresentazione Complemento a due (2's C)

Dati n bit per la codifica del modulo e del segno:

la rappresentazione in complemento a 2 (2's C) di un numero si ottiene sommando (sottraendo nel caso di numeri negativi) a 2^n il numero codificato in valore assoluto ed eliminando l'eventuale bit di riporto in posizione n ;



- Con tale rappresentazione possono essere codificati i valori compresi nell'intervallo $[-2^{n-1}, (2^{n-1}-1)]$.
- **Esiste solo una rappresentazione dello zero**
- I numeri positivi restano inalterati, come nel binario naturale
- I numeri negativi sono calcolabili partendo dal corrispondente valore positivo, invertendo tutti i bit (complemento a 1, 1's) e sommando 1

Rappresentazione Complemento a due (2's C)

Funzione Valore di un numero codificato in complemento a 2.

In una configurazione binaria di n bit codificata in complemento a 2, il bit più significativo (MSB in posizione $n-1$) assume un peso negativo pari a -2^{n-1} .

$$V_{10} = -2^{n-1} \cdot d_{n-1} + \sum_{i=0}^{n-2} d_i \cdot 2^i \quad d_i \in [0,1]$$

I numeri positivi ($d_{n-1}=0$) codificati in complemento a 2 rimangono del tutto analoghi al binario naturale.

Esempio: $n=4$

1011_2 in complemento a 2 equivale a:

$$1011_2 = -1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 = -8 + 0 + 2 + 1 = -5_{10}$$

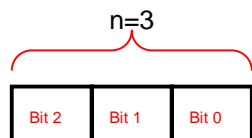
Mentre 0111_2 in complemento a 2 equivale a :

$$0111_2 = 0 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 = 0 + 4 + 2 + 1 = +7_{10}$$

Rappresentazione Complemento a due (2's C)

Siano dati **3 bit** ($n=3$) per la rappresentazione di numeri con segno

Con tale configurazione di bit potranno essere codificati i numeri da -4 a $+3$, cioè $[-2^2, 2^2-1]$



La rappresentazione in complemento a 2 (2's C) si ottiene come segue:

si somma (o si sottrae) a $2^3 = 8_{10} = 1000_2$ la rappresentazione in valore assoluto del numero che si vuole rappresentare in 2's C,

Es $+1$: Per i numeri positivi si somma a 1000_2 il modulo del numero ($|1_{10}|=001_2$).

Quindi $+1_{10} = 1000_2 + 001_2 = x001_2$ in complemento a 2

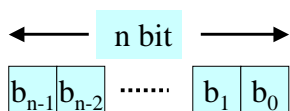
Es -1 : Per i numeri negativi si sottrae da 1000_2 il modulo del numero $|1_{10}|=001_2$.

Quindi $-1_{10} = 1000_2 - 001_2 = x111_2$ in complemento a 2

© 2004 Luciano Bononi

57

Numeri relativi: rappres. 2'sC



Esempi ($n=4$)

$$+1 = 0001$$

$$-1 = 1110$$

$$= 1111$$

$$+7 = 0111$$

$$-7 = 1000$$

$$= 1001$$

N.B. - anche nella rappresentazione in complemento a 2 il bit più significativo indica il segno (0:positivo, 1:negativo).

$$N = -2^{n-1} \cdot b_{n-1} + b_{n-2} \cdot 2^{n-2} + \dots + b_0 \cdot 2^0$$

$$N \in [-2^{n-1}, +(2^{n-1} - 1)]$$

© 2004 Luciano Bononi

58

Rappresentazione Complemento a due (2's C)

N.B.

Come ci si attende, applicando due volte la regola del complemento a 2 si ottiene il numero originale.

Esempio

$n = 4$, voglio rappresentare -5 a partire da $5 = 0101_2$

-5_{10} in complemento a 2 risulta 1011_2

Applicando nuovamente il complemento a 2 si ottiene il valore assoluto del numero

$$-5 \rightarrow 1011_2 \xrightarrow{1's} 0100_2 \xrightarrow{+1} 0101_2 = +5_{10}$$

Vantaggi del Complemento a due

- Vi è una sola rappresentazione per lo zero (00...000), quindi i confronti di uguaglianza a zero sono veloci
- non vi è differenza nell'eseguire somme o sottrazioni, quindi posso usare lo stesso circuito per le somme e sottrazioni, a condizione di assumere i valori espressi in 2'sC. Infatti: $(A - B) = A + (-B)$
- Inoltre, per la sottrazione non è necessario individuare il maggiore, in valore assoluto, tra i due operandi come nel caso della rappresentazione S-VA.

Esercizio di verifica

Utilizzando una rappresentazione 2'sC a 4 bit, e facendo le somme, calcolare

| | | | | | |
|----|----|---|---|------------|-------|
| 3 | +1 | = | ? | [Soluzione | 0100] |
| 3 | -1 | = | ? | [Soluzione | 0010] |
| -1 | -2 | = | ? | [Soluzione | 1101] |
| 3 | -7 | = | ? | [Soluzione | 1100] |

Gestione dell'overflow

M-S, 1'sC

Nel caso della rappresentazione con modulo e segno e 1'sC, la presenza di eventuali situazioni **overflow** puo' essere rilevata analizzando il bit di **carry-out** relativo al bit più significativo del modulo (da notare che in 1'sC occorre anche complementare i bit dei valori negativi prima della somma o sottrazione).

2's C

Nel caso di somme algebriche con numeri rappresentati in complemento a 2 la rilevazione della condizione di **overflow** si ottiene controllando se il bit di **carry-in** e il bit di **carry_out** relativi al bit più significativo (il bit n-1) della codifica sono diversi. Questa operazione puo' essere eseguita utilizzando l'operatore logico **or-esclusivo**.

Esempi: Con **N=3** possono essere rappresentati i numeri tra [+3, -4] in 2's C

| | | | | |
|----------------------|-----------|------------------------|-------------|-------------|
| $3+3 = 6$ (overflow) | $2+1 = 3$ | $-3-3 = -6$ (overflow) | $-3-1 = -4$ | $-3+2 = -1$ |
| 0 1 1 | 0 0 0 | 1 0 1 | 1 1 1 | 0 0 0 |
| 0 1 1+ | 0 1 0+ | 1 0 1+ | 1 0 1+ | 1 0 1+ |
| 0 1 1= | 0 0 1= | 1 0 1= | 1 1 1= | 0 1 0= |
| ----- | ----- | ----- | ----- | ----- |
| 0 1 1 0 | 0 0 1 1 | 1 0 1 0 | 1 1 0 0 | 0 1 1 1 |

© 2004 Luciano Bononi

61

Gestione dell'overflow

2's C

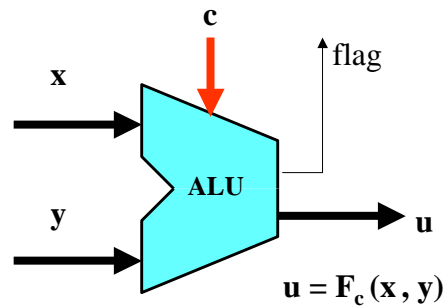
Nel caso di somme algebriche con numeri rappresentati in complemento a 2 la rilevazione della condizione di **overflow** si ottiene solo se i due valori da sommare sono concordi (stesso segno) e il risultato è discorde (segno diverso), oppure se i due valori da sottrarre sono discordi e il risultato è discorde dal primo operando (infatti basta notare che dati A-B discordi/concordi ciò equivale a A+B concordi/discordi).

| | | | | |
|----------------------------------|-----|-----|---------------------|---------------------|
| Somma A e B concordi | A>0 | B>0 | A+B > 0 OK | A+B < 0 Overflow |
| | A<0 | B<0 | A+B > 0 Overflow | A+B < 0 OK |
| Sottrazione A e B discordi | A>0 | B<0 | A-B > 0 OK | A-B < 0 Overflow |
| | A<0 | B>0 | A-B > 0 Overflow | A-B < 0 OK |

© 2004 Luciano Bononi

62

ALU: Aritmetical Logical Unit



ALU - Rete combinatoria in grado di eseguire diverse operazioni di tipo aritmetico o logico. L'operazione di volta in volta eseguita dipende dal valore attribuito ai bit di programmazione (codice operazione C)