Architettura degli elaboratori
Lezione 1

Prof.ssa Valentina Ciriani
Università degli Studi di Milano
www.di.unimi.it/ciriani

1

Contenuto

1. Sistemi numerici [MKM 1.3][PH 2.4]
2. Operazioni aritmetiche [MKM 1.4]
3. Rappresentazione degli interi [PH 2.4] [MKM 3.10, 3.11]:
   1. Modulo e segno
   2. Complemento a 2

-MKM = M. Morris Mano, C.R. Kime, T. Martin, Reti logiche, Pearson
-PH = D.A. Patterson, J.L. Hennessy, Struttura e Progetto dei Calcolatori, Zanichelli

2

2

Cosa capisce un computer?

- I computer non capiscono:
  - il linguaggio umano
  - i linguaggi di programmazione (java, C)
- Capiscono solo il linguaggio dei bit

bit	0 o 1
Byte (B)	8 bit
Parola	4 Byte
kiloByte	1024 Byte
megaByte	10^6 Byte
gigaByte	10^9 Byte
teraByte	10^{12} Byte

3

3

Numeri romani

Problematiche

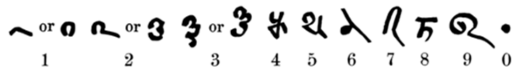
1. Difficoltà nel rappresentare numeri lunghi
2. Non esisteva il concetto di 0
3. Le operazioni (addizione, sottrazione) sono molto difficili

Simbolo	I	V	X	L	C	D	M
Valore	1	5	10	50	100	500	1000

4

4

Numeri indiani (o arabi)



Manoscritto di Bakhshali, prima rappresentazione conosciuta dello 0

1. E' il sistema numerico decimale
2. Utilizza la notazione posizionale
3. I numeri sono in base 10

Esempio:

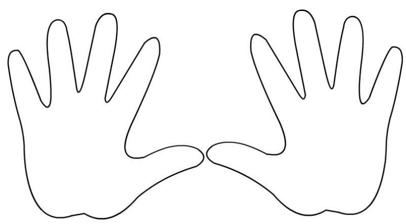
$4903,69 = 4*10^3 + 9*10^2 + 0*10^1 + 3*10^0 + 6*10^{-1} + 9*10^{-2}$

5

5

Sistemi numerici e basi

Perché noi usiamo la base 10?



6

6

Cosa succederebbe....

… in un mondo con persone che hanno solo due dita?

7

7

Sistema binario

Si userebbe un sistema con base 2

Numeri base 10	Numeri base 2
5	101
100	1100100
500	111110100
1024	10000000000

1000110

Bit più significativo MSD (most significant digit)

Bit meno significativo LSD (less significant digit)

$1000110 = (1*2^6 + 0*2^5 + 0*2^4 + 0*2^3 + 1*2^2 + 1*2^1 + 0*2^0)_{10} =$
$= (70)_{10}$

8

8

Sistema a base r

Un generico sistema a base r

- prevede r simboli distinti (0,1,2,…, r-1)
- moltiplicati per potenze intere di r:

$$x_{n-1}r^{n-1} + x_{n-2}r^{n-2} + \ldots + x_1 r^1 + x_0 r^0$$
$$+ x_{-1}r^{-1} + x_{-2}r^{-2} + \ldots + x_{-m+1}r^{-m+1} + x_{-m}r^{-m}$$

- **notazione posizionale**:

$$(x_{n-1}x_{n-2}\ldots x_1 x_0, x_{-1}x_{-2}\ldots x_{-m+1}x_{-m})_r$$

oppure $x_{n-1}x_{n-2}\ldots x_1 x_0, x_{-1}x_{-2}\ldots x_{-m+1}x_{-m}$ se r è ovvio

- in questo caso il MSD è x_{n-1} e il LSD è x_{-m}

9

9

Da base r a base 10

Calcolo in base 10 del valore del numero in base r:

$(x_{n-1}\ldots x_1 x_0, x_{-1}\ldots x_{-m})_r$ =
$(x_{n-1}r^{n-1} + \ldots + x_1 r^1 + x_0 r^0 + x_{-1}r^{-1} + \ldots + x_{-m}r^{-m})_{10}$

Esempio base 7:

$(216,3)_7 = (2 \cdot 7^2 + 1 \cdot 7^1 + 6 \cdot 7^0 + 3 \cdot 7^{-1})_{10}$ =
$= (98 + \underline{7 + 6} + 0,428571\ldots)_{10}$ =
$= (111,428571)_{10}$

10

10

Da base r a base 10

Esempio base r = 2:

$(1011,1)_2 = (1*2^3 + 0*2^2 + 1*2^1 + 1*2^0 + 1*2^{-1})_{10} =$
$= (8+2+1+0,5)_{10} = (11,5)_{10}$

11

11

Da base 10 a base r (interi)

- intero $(n)_{10}$ da convertire a $(m)_r$

Algoritmo: (input n e r)

```
i=1;
while (n!=0){
        m_i = n % r ; // modulo (resto divisione)
        n = n / r ;   // divisione intera
        i++;}
```

- se k è il numero iterazioni del while allora $(m)_r$ è il numero $(m_k m_{k-1} \dots m_2 m_1)_r$

12

12

Esempio conversione (intera)

$r = 2 \quad n= (37)_{10}$

37%2 = 1 37/2 = 18
18%2 = 0 18/2 = 9
9%2 = 1 9/2 = 4
4%2 = 0 4/2 = 2
2%2 = 0 2/2 = 1
1%2 = 1 1/2 = 0

$m = (100101)_2 \quad k=6$

13

13

Da base 10 a base r (decimali)

- decimale $(n)_{10}$ da convertire a $(m)_r$
- p è il numero massimo di cifre decimali per m (precisione di m)

Algoritmo: (input: n, p e r)

```
i=1;
while (n!=0 && i<=p){  //massimo p iterazioni
        m_i = n ×_I r ;     // prodotto (parte intera)
        n   = n ×_D r ;     // prodotto (decimali)
        i++;}
```

- se k è il numero iterazioni del while allora

$(m)_r$ è il numero $(0,m_1m_2\ldots m_{k-1}m_k)_r$

- si noti che $k \leq p$

14

14

Esempio conversione (decimale)

$r = 2$ $n = (0{,}87)_{10}$ $p=7$ (precisione a 7 cifre)

$0{,}87 \times 2 = 1{,}74$ $m_1 = 1$ (parte intera)

$0{,}74 \times 2 = 1{,}48$ $m_2 = 1$

$0{,}48 \times 2 = 0{,}96$ $m_3 = 0$

$0{,}96 \times 2 = 1{,}92$ $m_4 = 1$

$0{,}92 \times 2 = 1{,}84$ $m_5 = 1$

$0{,}84 \times 2 = 1{,}68$ $m_6 = 1$

$0{,}68 \times 2 = 1{,}36$ $m_7 = 1$

$m = (0{,}1101111)_2$

15

15

Esempio conversione (completo)

- Per convertire un numero qualsiasi, basta convertire la parte decimale e quella intera e poi unire i due risultati

Esempio:

- $n = (37{,}87)_{10}$ con $p=7$ la parte decimale
- da convertire in base 2 ($r=2$)
- soluzione: $m = (100101{,}1101111)_2$

16

16

Numeri binari (base 2)

- Solo due simboli 0 e 1
- I **bit** sono le cifre di un numero binario

n	2^n	*n*	2^n	*n*	2^n
0	1	8	256	16	65,536
1	2	9	512	17	131,072
2	4	10	1,024	18	262,144
3	8	11	2,048	19	524,288
4	16	12	4,096	20	1,048,576
5	32	13	8,192	21	2,097,152
6	64	14	16,384	22	4,194,304
7	128	15	32,768	23	8,388,608

17

17

Numeri ottali ed esadecimali

- Utili per rappresentare in modo più compatto numeri binari ($8=2^3$ e $16=2^4$)
- Per gli esadecimali dopo il 9 si usano le lettere A B C D E F

Decimal (base 10)	Binary (base 2)	Octal (base 8)	Hexadecimal (base 16)
00	0000	00	0
01	0001	01	1
02	0010	02	2
03	0011	03	3
04	0100	04	4
05	0101	05	5
06	0110	06	6
07	0111	07	7
08	1000	10	8
09	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

18

18

Conversioni semplici

- Nel caso di conversioni da binario a ottale o esadecimale e viceversa:
  - la conversione è più semplice
  - perché $8 = 2^3$ e $16=2^4$ e quindi sono potenze di 2
  - algoritmi semplificati

19

19

Da binario ad ottale

1. Si divide il numero binario in gruppi di **tre** bit a partire dalla virgola
2. Ogni cifra ottale viene codificata dai tre bit

Esempio: $(1100111000,10111)_2$

1. $(001\ 100\ 111\ 000,\ 101\ 110)_2$
2. $(\ 1\quad 4\quad 7\quad 0,\quad 5\quad 6\ )_8$

$(1470,56)_8$ oppure 01470,56

20

20

Da binario ad esadecimale

1. Si divide il numero binario in gruppi di 4 bit (parole) a partire dalla virgola
2. Ogni cifra esadecimale viene codificata dai 4 bit

Esempio: $(1100111000,10111)_2$

1. $(0011\ 0011\ 1000,\ 1011\ 1000)_2$
2. $(\ 3 \quad 3 \quad 8 \ , \quad B \quad 8\ )_{16}$

$(338,B8)_{16}$ oppure 0x338,B8

21

21

Da ottale/esadec. a binario

Procedure al contrario

Esempio:

1. $(237,31)_8 = (010\ 011\ 111,011\ 001)_2 =$
   $= (10011111,011001)_2$
2. $(3AF,B8)_{16} = (0011\ 1010\ 1111,1011\ 1000)_2 =$
   $= (1110101111,10111)_2$

22

22

Addizione di binari

Come la somma in base 10 (utilizzando il resto):

0+0 = 0
1+0 = 1
0+1 = 1
1+1 = 0 con resto di 1 (ovvero 1+1=10)
1+1+1= 1 con resto di 1 (ovvero 1+1+1=11)

```
111          resto
  1110+      addendo
  1011=      addendo
 11001       somma
```

23

23

Intervalli numerici (senza segno)

- Nei computer digitali, l'intervallo numerico dei valori rappresentabili dipende dal numero di bit disponibili (ovvero N)
- In caso di **interi senza segno** (unsigned):
  $[0, 2^N-1]$
- Esempio numeri interi a 32 bit senza segno: il decimale 246 è rappresentato come
  $(0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 1111\ 0110)_2$
- L'intervallo in questo caso è da 0 a $2^{32}-1$

24

24

Overflow (trabocco)

- L'addizione genera overflow quando la somma non è nell'intervallo di rappresentazione e quindi non è rappresentabile
- Esempio N = 4 (4 bit per la rappresentazione)

```
           101
            1010+
            1011=
overflow   10101
```

(la somma **non** si rappresenta con 4 bit)

25

25

Differenza di binari

Come la differenza in base 10 (utilizzando i prestiti):

0-0=0

1-0=1

0-1=1 con il prestito di 1 (ovvero 10-1=1)

1-1=0

1-1-1=1 con il prestito di 1 (ovvero 11-1-1=1)

0-1-1=0 con il prestito di 1 (ovvero 10-1-1=0)

26

26

Differenza di binari

Come la differenza in base 10 (utilizzando i prestiti)

1. Se il sottraendo è minore del minuendo ho:

```
  11        prestiti
 1110-      minuendo
 1011=      sottraendo
 ----
 0011       differenza
```

27

27

Differenza di binari

2. Se il sottraendo è maggiore del minuendo si scambiano e si ha una differenza negativa:

Esempio: 1011 - 1110

```
  11        prestiti
 1110-      sottraendo
 1011=      minuendo
 ----
-0011       differenza
```

28

28

Differenza di binari

Esempio

```
 1111        prestiti
 10000-      minuendo
  1011=      sottraendo
 00101       differenza
```

Da sinistra verso destra:

1. 10-1 = 1 con prestito di 1
2. 10-1-1= 0 con prestito di 1
3. 10-0-1= 1 con prestito di 1
4. 10-1-1= 0 con prestito di 1
5. 1-1 = 0

29

29

Prodotto di binari

Come il prodotto in base 10 (utilizzando i prodotti parziali che sono poi sommati)

```
   1110 ×   moltiplicando
    110 =   moltiplicatore
   0000     prodotti parziali
  1110
 1110
1010100    prodotto
```

30

30

Riflessione finale!

Ci sono solo 10 tipi di persone nel mondo:

- quelli che conoscono i binari
- e quelli che non li conoscono

31

31

Rappresentazione degli interi

32

32

Rappresentazione con N bit

- Se ho N bit posso rappresentare 2^N valori
- Es: N bit per rappresentare 2^N interi positivi: $[0, 2^N-1]$
- Se ho M valori da rappresentare quanti bit mi servono?
  - $\lceil \log_2 M \rceil$ (parte intera superiore del logaritmo)
- Se ho M valori da rappresentare quante cifre in base r mi servono?
  - $\lceil \log_r M \rceil$
- Per esempio:
  - per rappresentare 14 valori distinti in binario ho bisogno di $\lceil \log_2 14 \rceil = \lceil 3{,}807\ldots \rceil = 4$ bit
  - infatti 3 bit non i basterebbero perché con 3 bit posso rappresentare al massimo $2^3 = 8$ valori distinti

33

33

Modulo e segno

- Rappresentazione **modulo e segno**: si aggiunge un bit con il segno
- Pro: soluzione più semplice
- Contro:
  - dove mettere il bit di segno?
  - Quando si somma bisogna tenere conto del segno
  - Lo zero si rappresenta in due modi -0, +0
- Utilizzata nei primi computer e poi abbandonata
- **Intervallo** di rappresentazione con N bit: $[-2^{N-1}+1, 2^{N-1}-1]$
- Es: N=8 intervallo $[-2^7+1, 2^7-1]=[-127, 127]$

34

34

Complemento a 2

- Rappresentazione a **complemento a 2 (N=32):**

$0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000_2 = 0_{10}$
$0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0001_2 = 1_{10}$
$0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0010_2 = 2_{10}$
. . . . . .
$0111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1101_2 = 2.147.483.645_{10}$
$0111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1110_2 = 2.147.483.646_{10}$
$0111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111_2 = 2.147.483.647_{10} = (2^{31} - 1)_{10}$
$1000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000_2 = -2.147.483.648_{10} = (-2^{31})_{10}$
$1000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0001_2 = -2.147.483.647_{10}$
$1000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0010_2 = -2.147.483.646_{10}$
. . . . . .
$1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1101_2 = -3_{10}$
$1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1110_2 = -2_{10}$
$1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111_2 = -1_{10}$

35

35

Complemento a 2 (C2)

- Intervallo di rappresentazione: $[-2^{N-1}, +2^{N-1}-1]$
- Es: N=8 intervallo $[-2^7, 2^7-1]=[-128, 127]$
- Con N=32 i numeri da 0 a $2.147.483.647_{10}$ ($= 2^{31} - 1$) sono rappresentati come degli interi positivi
- $1000...000_2$ rappresenta il numero negativo di valore assoluto maggiore $-2.147.483.648_{10}$ ($= -2^{31}$)
- $1111...1111_2$ rappresenta -1
- $-2.147.483.648_{10}$ non ha un corrispondente numero positivo
- I numeri negativi iniziano con 1 (bit di segno)

36

36

C2: da decimale a binario

- Sia x intero decimale in $[-2^{N-1}, +2^{N-1}-1]$
- la rappresentazione in complemento a 2 di x è:

1. se $x \geq 0$ allora è come la rappresentazione di x senza segno
2. se $x<0$ è la rappresentazione, come intero senza segno, di 2^N+x
(attenzione: x è negativo)

37

37

C2: da decimale a binario esempio

- Es: N=8 intervallo [-128, 127]
  - 3_{10} si rappresenta come $0000\ 0011_2$
  - -3_{10} si rappresenta come $2^8 - 3 = 256 - 3 = 253 = 1111\ 1101_2$

38

38

C2: cambio di segno

- ricopio le cifre da destra a sinistra fino al primo 1 (compreso) e poi invertendo le rimanenti (da 0 a 1 e da 1 a 0)
- Esempio:

$(0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 1100)_2 = 12_{10}$
$(1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 0100)_2 = -12_{10}$

39

39

C2: da decimale a binario

Algoritmo semplificato:

- Sia x intero decimale in $[-2^{N-1}, +2^{N-1}-1]$
- la rappresentazione in complemento a 2 di x è:

1. se $x \geq 0$ allora è come la rappresentazione di x senza segno
2. se $x<0$ parto dalla rappresentazione di -x (-x è positivo) e poi ricopio le cifre da destra a sinistra fino al primo 1 (compreso) e poi invertendo le rimanenti (da 0 a 1 e da 1 a 0)

40

40

C2: da decimale a binario

- Esempio:

Caso positivo 29_{10}

$(0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0001\ 1101)_2$

Caso negativo -29_{10}

$(0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0001\ 1101)_2 = 29_{10}$

$(1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1110\ 0011)_2 = -29_{10}$

41

41

Da fare a casa

1. studiare, gli argomenti svolti, sulle slide e **sui libri** (per i capitoli, vedere slide 2)
2. provare a svolgere a casa gli esercizi proposti
3. preparare eventuali domande sugli argomenti non compresi (sia teoria che esercizi)

Nella prossima lezione vedremo insieme le soluzioni degli esercizi e le risposte ai dubbi

42

42

Esercizi da fare a casa 1

1. Convertire da base r a base 10:
   1. r=2 $11010,101_2$
   2. r=8 $4156,27_8$
   3. r=16 $A8F,B_{16}$
2. Convertire da base 10:
   1. a base 2 il numero 4526,76 con precisione p=5
   2. a base 2 il numero 4526,75 con precisione p=5
   3. a base 8 il numero 23,2 con precisione p=3
   4. a base 16 il numero 270,12 con precisione p=3
3. Convertire da base 2 a base 8 e a base 16 e viceversa (da 8 a 2 e da 16 a 2) i binari senza segno:
   1. 1010111001
   2. 11101110000

43

43

Esercizi da fare a casa 2

4. Eseguire la somma, la sottrazione e la moltiplicazione tra le seguenti coppie di numeri binari senza segno:
   1. 11010_2 e 101_2
   2. 11011_2 e 10101_2
   3. 101001_2 e 11111_2
5. Quale è l'intervallo di rappresentazione di binari senza segno con 10 cifre?
6. Si consideri N=8. Rappresentare i numeri 85_{10} e -85_{10} in binario:
   1. modulo e segno
   2. complemento a 2
7. Quanti bit mi servono per rappresentare le 21 lettere dell'alfabeto italiano?

44

44