ARCHITETTURA DEGLI ELABORATORI

A.A. 2020-2021

Università di Napoli Federico II
Corso di Laurea in Informatica

Docenti
Proff. Luigi Sauro gruppo 1 (A-G)
Silvia Rossi gruppo 2 (H-Z)

Quale è il maggiore?

A) 1 0 1 0 1 0 1 0

B) 1 0 0 1 0 1 0 0

C) 1 0 1 0 1 0 1 1

Hexadecimal to Binary Conversion

- Hexadecimal to binary conversion:
  - Convert $4AF_{16}$ (also written 0x4AF) to binary

- Hexadecimal to decimal conversion:
  - Convert 0x4AF to decimal

Hexadecimal to Binary Conversion

- Hexadecimal to binary conversion:
  - Convert $4AF_{16}$ (also written 0x4AF) to binary
  - $0100\ 1010\ 1111_2$

- Hexadecimal to decimal conversion:
  - Convert $4AF_{16}$ to decimal
  - $16^2 \times 4 + 16^1 \times 10 + 16^0 \times 15 = 1199_{10}$

Powers of Two

- $2^0 = 1$
- $2^1 = 2$
- $2^2 = 4$
- $2^3 = 8$
- $2^4 = 16$
- $2^5 = 32$
- $2^6 = 64$
- $2^7 = 128$
- $2^8 = 256$
- $2^9 = 512$
- $2^{10} = 1024$
- $2^{11} = 2048$
- $2^{12} = 4096$
- $2^{13} = 8192$
- $2^{14} = 16384$
- $2^{15} = 32768$

Handy to memorize up to 2^9

Large Powers of Two

- 2^{10} = 1 kilo ≈ 1000 (1024)
- 2^{20} = 1 mega ≈ 1 million (1,048,576)
- 2^{30} = 1 giga ≈ 1 billion (1,073,741,824)

Estimating Powers of Two

- What is the value of 2^{24}?

- How many values can a 32-bit variable represent?

Estimating Powers of Two

- What is the value of 2^{24}?

  $2^4 \times 2^{20} \approx 16$ million

- How many values can a 32-bit variable represent?

  $2^2 \times 2^{30} \approx 4$ billion

Esercizi

- Convertire da base 2 a base 10:
  - 0110011
  - 10101100
  - 1100110011

- Convertire in base 10 i seguenti numeri:
  - 102210_3
  - 431204_5
  - 5036_7
  - $198A1_{12}$

Esercizi

- Convertire da base 10 alla base indicata i seguenti numeri:
  - 7562 base 8
  - 1938 base 16
  - 205 base 16
  - 175   base 2
- In un registro a 32 bit è memorizzato il valore 0xF3A7C2A4. Esprimere il contenuto del registro in base 2
- Convertire da base 2 a base 16:
  - 10010
  - 11010101
  - 10010011
- Scrivere in babilonese il numero 4000

Numeri con parole di lunghezza fissa

- I registri dei moderni calcolatori sono tipicamente parole di 32 o 64 bit
- Con una parola di lunghezza fissa sono rappresentabili un numero finito di naturali.

2^{m-1}	**2^{m-2}**							**2^1**	**2^0**

  – Nel caso di parole a 64 bit sono rappresentabili i numeri da 0 a 2^{64}-1

- Chiaramente, poiché ogni numero deve essere rappresentato dallo stesso numero di cifre, occorre ricorrere necessariamente a zeri non significativi

Numeri con parole di lunghezza fissa

- Supponiamo di avere una macchina che opera con parole di 16 bit, il numero 9 sarà quindi rappresentato come: 0000 0000 0000 1001
- Operazioni come l'addizione o la moltiplicazione possono produrre numeri troppo grandi per essere rappresentati. In questo caso parleremo di trabocco o ***overflow***
- Supponiamo di nuovo di avere una parola di 16 bit e supponiamo di voler elevare 1024 al quadrato. Questo produrrà un trabocco, infatti:

$$1024^2 = 1024 \cdot 1024 = 2^{10} \cdot 2^{10} = 2^{20} > 2^{16} - 1$$

Somma binaria

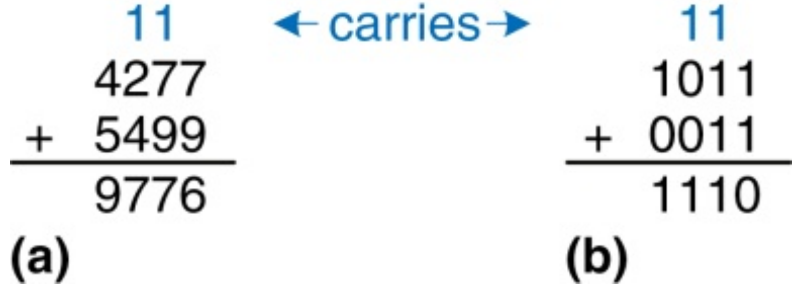
Nel sistema di numerazione in base 2 esistono due soli simboli: 0 e 1 e quindi quando si effettua l'operazione 1 + 1, non si ha un unico simbolo per rappresentare il risultato, ma il risultato è 0 con il riporto di 1, cioè 10 (da leggere uno, zero e non dieci).

Le regole per effettuare l'operazione di somma di due cifre binarie sono riassunte di seguito:

- 0 + 0 = 0
- 0 + 1 = 1
- 1 + 0 = 1
- 1 + 1 = 0 con riporto di 1

Somma in binario

L’operazione di somma in binario è algoritmicamente analoga a quella decimale con la differenza che il riporto si ha quando si eccede 1 (invece che 9)



Somma binaria

La somma dei due numeri interi 10001 e 11011 è pari a:

10001 +

11011 =

101100

Binary Addition Examples

- Add the following 4-bit binary numbers

$$\begin{array}{r} 1001 \\ +\ 0101 \\ \hline \end{array}$$

- Add the following 4-bit binary numbers

$$\begin{array}{r} 1011 \\ +\ 0110 \\ \hline \end{array}$$

Binary Addition Examples

- Add the following 4-bit binary numbers

$$
\begin{array}{r}
{\color{blue}1} \\
1001 \\
+\ 0101 \\
\hline
{\color{blue}1110}
\end{array}
$$

- Add the following 4-bit binary numbers

$$
\begin{array}{r}
{\color{blue}111} \\
1011 \\
+\ 0110 \\
\hline
{\color{red}1}{\color{blue}0001}
\end{array}
$$

Binary Addition Examples

- Add the following 4-bit binary numbers

$$
\begin{array}{r}
{\color{blue}1} \\
1001 \\
+\ \ 0101 \\
\hline
{\color{blue}1110}
\end{array}
$$

- Add the following 4-bit binary numbers

$$
\begin{array}{r}
{\color{blue}111} \\
1011 \\
+\ \ 0110 \\
\hline
{\color{red}1}{\color{blue}0001}
\end{array}
$$

Overflow!

Moltiplicazione binaria

Le regole per la moltiplicazione sono:

- 0 * 0 = 0
- 0 * 1 = 0
- 1 * 0 = 0
- 1 * 1 = 1

Moltiplicazione in binario

Anche la moltiplicazione in binario è analoga a quella decimale

```
    0101 x
    0011 =
----------
    0101
   0101=
  0000==
 0000===
----------
 0001111
```

Rappresentazione di interi

- Occupiamoci ora della rappresentazione di numeri interi …, -2,-1,0,1,2,… mediante parole di lunghezza fissata
- Chiaramente dobbiamo codificarne non solo il valore assoluto ma anche il segno
- Le principali rappresentazioni di numeri interi sono:
  - Rappresentazione con segno
  - Complemento all’intervallo (complemento a due)

Rappresentazione con segno

- Si supponga che le parole siano di lunghezza m e la base sia b
- In tale rappresentazione la cifra più significativa di una parola rappresenta il segno. Per convenzione 0 rappresenta il segno più, 1 rappresenta il segno meno
- Le rimanenti m-1 cifre sono usate per rappresentare il valore assoluto di un intero
- Ad esempio si considerino parole binarie di lunghezza 4
  - 0100 → +100 → 4
  - 1011 → -011 →-3
  - Il più grande numero rappresentabile è 0111 (ovvero 7)
  - Il più piccolo numero rappresentabile è 1111 (ovvero -7)
  - Lo zero ha due possibili rappresentazioni: 0000 e 1000

Rappresentazione con segno

Poiché m-1 cifre sono utilizzate per rappresentare il valore assoluto, il range di numeri codificabili è $-(b^{m-1}-1),\ldots,0,\ldots,b^{m-1}-1$

Svantaggio: nelle operazioni di addizione o sottrazione occorre controllare il segno e i valori assoluti dei due operandi per determinare il segno del risultato.

- 0010 + 0001
  - sono entrambi positivi quindi il segno sarà positivo → 0011
- 0101 + 1010
  - il primo è positivo mentre il secondo è negativo, il valore assoluto del primo è maggiore del valore assoluto del secondo quindi il segno sarà positivo → 0011
- 1100+0011
  - il primo è negativo mentre il secondo è positivo, il valore assoluto del primo è maggiore di quello del secondo, quindi il segno sarà negativo → 1001
- 1011-1010
  - Entrambi sono negativi ma il valore assoluto del secondo è minore di quello del primo. Quindi occorre sottrarre 010 da 011 cambiando il bit del segno → 1001

Complemento a 2

- La rappresentazione è analoga a quella unsigned con la differenza che il bit più significativo corrisponde a -2^{m-1} invece che 2^{m-1}

-2^{m-1}	2^{m-2}							2^1	2^0

- Lo zero ha una unica rappresentazione 0…0
- Il range di rappresentazione è $[-2^{m-1}, 2^{m-1} -1]$
- Essendo -2^{m-1} in valore assoluto il «peso» più grande, se un numero inizia con 1 allora è negativo altrimenti è positivo

Complemento a 2

- Massimo valore rappresentabile: $2^{m-1} -1 \rightarrow$ 01…1
- Minimo valore rappresentabile: $-2^{m-1} \rightarrow$ 10…0
- Il numero -1 è scritto come: 11…1

"Taking the Two's Complement"

- "Taking the Two's complement" **flips the sign** of a two's complement number
- **Method:**
    1. Invert the bits
    2. Add 1
- **Example:** Flip the sign of $3_{10} = 0011_2$

"Taking the Two's Complement"

- "Taking the Two's complement" **flips the sign** of a two's complement number
- **Method:**
  1. Invert the bits
  2. Add 1
- **Example:** Flip the sign of $3_{10} = 0011_2$
  1. **1100**
  2. **+ 1**

  1101 = -3_{10}

Two's Complement Examples

- Take the two's complement of $6_{10} = 0110_2$

- What is the decimal value of the two's complement number 1001_2?

Two's Complement Examples

- Take the two's complement of $6_{10} = 0110_2$
  1. 1001
  2. + 1

  $1010_2 = -6_{10}$

- What is the decimal value of the two's complement number 1001_2?
  1. 0110
  2. + 1

  $0111_2 = 7_{10}$, so $1001_2 = -7_{10}$

Complemento a 2

- Per complementare un numero occorre invertire ogni cifra e sommare 1
  - Rappresentare il numero -2 e -7 con 4 bit
  - 2=0010 → -2= 1101 +0001= 1110
  - 7=0111 → -7= 1000 +0001= 1001

  - Attenzione! questo non vale per -2^{m-1} il cui complemento non è rappresentabile con m bit (i numeri positivi arrivano a 2^{m-1} -1):
  - -2^3 =1000 → 0111 +0001= 1000

- La somma avviene come per i numeri unsigned

  -2+1 = 1110 + 0001= 1111 = -1

  -7+7 = 1001 + 0111= 0000 = 0 (con riporto 1)

Two's Complement Addition

- Add 6 + (-6) using two's complement numbers

$$
\begin{array}{r}
0110 \\
+\ \ 1010 \\
\hline
\end{array}
$$

- Add -2 + 3 using two's complement numbers

$$
\begin{array}{r}
1110 \\
+\ \ 0011 \\
\hline
\end{array}
$$

Two's Complement Addition

- Add 6 + (-6) using two's complement numbers

$$
\begin{array}{r}
\color{blue}{111} \\
0110 \\
+\quad 1010 \\
\hline
\color{gray}{1}0000
\end{array}
$$

- Add -2 + 3 using two's complement numbers

$$
\begin{array}{r}
\color{blue}{111} \\
1110 \\
+\quad 0011 \\
\hline
\color{gray}{1}0001
\end{array}
$$

Overflow nel complemento a 2

- Sommare un numero negativo e uno positivo non genera overflow
- L'esempio -7+7 mostra come l'overflow *non avviene come per gli unsigned* quando ho riporto finale di 1

- L'overflow avviene quando entrambi gli operandi sono negativi (primo bit=1) o entrambi positivi (primo bit=0) è il risultato ha segno opposto

  4+5=0100+0101 =1001 =-7

- Per estendere un numero in una rappresentazione con più bit basta riprodurre a sinistra il bit più significativo

  5=0101 → 00000101

  -4=1100 → 11111100

Sign-Extension

- Sign bit copied to msb's
- Number value is same
- **Example 1:**
  - 4-bit representation of 3 = **0**011
  - 8-bit sign-extended value: **0000**0011
- **Example 2:**
  - 4-bit representation of -5 = **1**011
  - 8-bit sign-extended value: **1111**1011

Number System Comparison

Number System	Range
Unsigned	$[0, 2^{N}-1]$
Sign/Magnitude	$[-(2^{N-1}-1), 2^{N-1}-1]$
Two's Complement	$[-2^{N-1}, 2^{N-1}-1]$

For example, 4-bit representation:

	-8	-7	-6	-5	-4	-3	-2	-1	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Unsigned									0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
Two's Complement	1000	1001	1010	1011	1100	1101	1110	1111	0000	0001	0010	0011	0100	0101	0110	0111								
Sign/Magnitude		1111	1110	1101	1100	1011	1010	1001	0000 1000	0001	0010	0011	0100	0101	0110	0111								

Esercizi

- Fornire la rappresentazione binaria in complemento a 2 ad 8 bit del numero -15
- Fornire la rappresentazione binaria in complemento a 2 ad 8 bit del numero -109
- Eseguire la somma dei numeri 5 e -32 espressa in completamento a 2 ad 8 bit
- Convertire in esadecimale il numero -53248 espresso in completamento a 2 ad 16 bit