Binary Numbers

Digital Design and Computer Architecture
Mohammad Sadrosadati
Frank K. Gürkaynak

http://safari.ethz.ch/ddca

In This Lecture

- How to express numbers using only 1s and 0s
- Using hexadecimal numbers to express binary numbers
- Different systems to express negative numbers
- Adding and subtracting with binary numbers

Number Systems

- **Decimal Numbers**

1000's column
100's column
10's column
1's column

$5374_{10} =$

- **Binary Numbers**

8's column
4's column
2's column
1's column

$1101_2 =$

Number Systems

- **Decimal Numbers**

1000's column
100's column
10's column
1's column

$$5374_{10} = 5 \times 10^3 + 3 \times 10^2 + 7 \times 10^1 + 4 \times 10^0$$

five thousands, three hundreds, seven tens, four ones

- **Binary Numbers**

8's column
4's column
2's column
1's column

$$1101_2 = 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = 13_{10}$$

one eight, one four, no two, one one

Powers of two

2^0	=	2^8	=
2^1	=	2^9	=
2^2	=	2^{10}	=
2^3	=	2^{11}	=
2^4	=	2^{12}	=
2^5	=	2^{13}	=
2^6	=	2^{14}	=
2^7	=	2^{15}	=

Powers of two

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

Handy to memorize up to 2^{15}

Binary to Decimal Conversion

- **Convert 10011_2 to decimal**

Binary to Decimal Conversion

- **Convert 10011_2 to decimal**

$$2^4 \times 1 + 2^3 \times 0 + 2^2 \times 0 + 2^1 \times 1 + 2^0 \times 1 =$$

Binary to Decimal Conversion

- Convert 10011_2 to decimal

$$2^4 \times 1 + 2^3 \times 0 + 2^2 \times 0 + 2^1 \times 1 + 2^0 \times 1 =$$

$$16 \times 1 + 8 \times 0 + 4 \times 0 + 2 \times 1 + 1 \times 1 =$$

$$16 + 0 + 0 + 2 + 1 = 19_{10}$$

Decimal to Binary Conversion

- **Convert 47_{10} to binary**

Decimal to Binary Conversion

- **Convert 47_{10} to binary**
  - Start with $2^6 = 64$ is $64 \leq 47$? no do nothing
  - Now $2^5 = 32$

Decimal to Binary Conversion

- **Convert 47_{10} to binary**
  - Start with $2^6 = 64$ is $64 \le 47$? no do nothing
  - Now $2^5 = 32$ is $32 \le 47$? yes subtract $47 – 32 = 15$
  - Now $2^4 = 16$ is $16 \le 15$? no do nothing
  - Now $2^3 = 8$ is $8 \le 15$? yes subtract $15 – 8 = 7$
  - Now $2^2 = 4$ is $4 \le 7$? yes subtract $7-4 = 3$
  - Now $2^1 = 2$ is $2 \le 3$? yes subtract $3-2 = 1$
  - Now $2^0 = 1$ is $1 \le 1$? yes we are done

Decimal to binary conversion

- **Convert 47_{10} to binary**
  - Start with $2^6 = 64$ is $64 \le 47$? no **0** do nothing
  - Now $2^5 = 32$ is $32 \le 47$? yes **1** subtract $47 - 32 = 15$
  - Now $2^4 = 16$ is $16 \le 15$? no **0** do nothing
  - Now $2^3 = 8$ is $8 \le 15$? yes **1** subtract $15 - 8 = 7$
  - Now $2^2 = 4$ is $4 \le 7$? yes **1** subtract $7-4 = 3$
  - Now $2^1 = 2$ is $2 \le 3$? yes **1** subtract $3-2 = 1$
  - Now $2^0 = 1$ is $1 \le 1$? yes **1** we are done

- **Result is 0101111_2**

Binary Values and Range

- ***N*-digit decimal number**
  - How many values? **10^N**
  - Range? **$[0, 10^N - 1]$**
  - Example: 3-digit decimal number
    - $10^3 = 1000$ possible values
    - Range: [0, 999]

- ***N*-bit binary number**
  - How many values? **2^N**
  - Range: **$[0, 2^N - 1]$**
  - Example: 3-digit binary number
    - $2^3 = 8$ possible values
    - Range: $[0, 7] = [000_2$ to $111_2]$

Hexadecimal (Base-16) Numbers

Decimal	Hexadecimal	Binary
0		
1		
2		
3		
4		
5		
6		
7		
8		
9		
10		
11		
12		
13		
14		
15		

Hexadecimal (Base-16) Numbers

Decimal	Hexadecimal	Binary
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
10	A	1010
11	B	1011
12	C	1100
13	D	1101
14	E	1110
15	F	1111

Hexadecimal Numbers

- **Binary numbers can be pretty long.**
- **A neat trick is to use base 16**
- **How many binary digits represent a hexadecimal digit?**

  4 (since $2^4 = 16$)

- **Example 32 bit number:**

  `0101 1101 0111 0001 1001 1111 1010 0110`

Hexadecimal Numbers

- **Binary numbers can be pretty long.**
- **A neat trick is to use base 16**
- **How many binary digits represent a hexadecimal digit?**

  4 (since $2^4 = 16$)

- **Example 32 bit number:**

```
0101 1101 0111 0001 1001 1111 1010 0110
5    D    7    1    9    F    A    6
```

Hexadecimal Numbers

- **Binary numbers can be pretty long.**
- **A neat trick is to use base 16**
- **How many binary digits represent a hexadecimal digit?**

  4 (since $2^4 = 16$)

- **Example 32 bit number:**

```
0101 1101 0111 0001 1001 1111 1010 0110
5    D    7    1    9    F    A    6
```

- **The other way is just as simple**

```
C    E    2    8    3    5    4    B
```

Hexadecimal Numbers

- **Binary numbers can be pretty long.**
- **A neat trick is to use base 16**
- **How many binary digits represent a hexadecimal digit?**

  4 (since $2^4 = 16$)

- **Example 32 bit number:**

```
0101 1101 0111 0001 1001 1111 1010 0110
5    D    7    1    9    F    A    6
```

- **The other way is just as simple**

```
C    E    2    8    3    5    4    B
1100 1110 0010 1000 0011 0101 0100 1011
```

Hexadecimal to Decimal Conversion

- **Convert $4AF_{16}$ (or 0x4AF) to decimal**

Hexadecimal to decimal conversion

- Convert $4AF_{16}$ (or 0x4AF) to decimal

$$16^2 \times 4 + 16^1 \times A + 16^0 \times F =$$

$$256 \times 4 + 16 \times 10 + 1 \times 15 =$$

$$1024 + 160 + 15 = 1199_{10}$$

Bits, Bytes, Nibbles...

10010110

most significant bit

least significant bit

byte

10010110

nibble

CEBF9AD7

most significant byte

least significant byte

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)

Powers of Two (SI Compatible)

- 2^{10} = 1 kibi ≈ 1000 (1024)
- 2^{20} = 1 mebi ≈ 1 million (1,048,576)
- 2^{30} = 1 gibi ≈ 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

Addition

- **Decimal**

```
   11  ← carries
   3734
 + 5168
 ------
   8902
```

- **Binary**

```
   11  ← carries
   1011
 + 0011
 ------
   1110
```

Add the Following Numbers

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

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

Add the Following Numbers

$$\begin{array}{r} {\color{blue}1} \\ 1001 \\ +\ \ 0101 \\ \hline {\color{blue}1110} \end{array}
\qquad\qquad
\begin{array}{r} {\color{blue}111} \\ 1011 \\ +\ \ 0110 \\ \hline {\color{blue}10001} \end{array}$$

OVERFLOW !

Overflow

- **Digital systems operate on a fixed number of bits**
- **Addition overflows when the result is too big to fit in the available number of bits**
- **See previous example of 11 + 6**

Overflow (Is It a Problem?)

- **Possible faults**
- **Security issues**

(Photograph courtesy ESA/CNES/ ARIANESPACE-Service Optique CS6.)

The $7 billion Ariane 5 rocket, launched on June 4, 1996, veered off course 40 seconds after launch, broke up, and exploded. The failure was caused when the computer controlling the rocket overflowed its 16-bit range and crashed.

The code had been extensively tested on the Ariane 4 rocket. However, the Ariane 5 had a faster engine that produced larger values for the control computer, leading to the overflow.

Binary Values and Range

- ***N*-digit decimal number**
  - How many values? **10^N**
  - Range? **$[0, 10^N - 1]$**
  - Example: 3-digit decimal number
    - 10^3 = 1000 possible values
    - Range: [0, 999]

- ***N*-bit binary number**
  - How many values? **2^N**
  - Range: **$[0, 2^N - 1]$**
  - Example: 3-digit binary number
    - 2^3 = 8 possible values
    - Range: [0, 7] = $[000_2$ to $111_2]$

Signed Binary Numbers

- Sign/Magnitude Numbers
- One’s Complement Numbers
- Two’s Complement Numbers

Sign/Magnitude Numbers

- **1 sign bit, N-1 magnitude bits**
- **Sign bit is the most significant (left-most) bit**
  - Positive number: sign bit = 0
  - Negative number: sign bit = 1

$$A : \{a_{N-1}, a_{N-2}, \cdots a_2, a_1, a_0\}$$

$$A = (-1)^{a_{n-1}} \sum_{i=0}^{n-2} a_i 2^i$$

- **Example, 4-bit sign/mag representations of ± 6:**

  +6 =

  - 6 =

- **Range of an N-bit sign/magnitude number:**

Sign/Magnitude Numbers

- **1 sign bit, N-1 magnitude bits**

- **Sign bit is the most significant (left-most) bit**
  - Positive number: sign bit = 0
  - Negative number: sign bit = 1

$$A: \{a_{N-1}, a_{N-2}, \cdots a_2, a_1, a_0\}$$

$$A = (-1)^{a_{n-1}} \sum_{i=0}^{n-2} a_i 2^i$$

- **Example, 4-bit sign/mag representations of ± 6:**

  +6 = 0110

  - 6 = 1110

- **Range of an N-bit sign/magnitude number:**

  $[-(2^{N-1}-1), 2^{N-1}-1]$

Problems of Sign/Magnitude Numbers

- **Addition doesn't work, for example -6 + 6:**

```
  1110
+ 0110
------
 10100
```
wrong!

- **Two representations of 0 (± 0):**

```
1000
0000
```

- **Introduces complexity in the processor design**
  (Was still used by some early IBM computers)

One’s Complement

- A negative number is formed by **reversing the bits of the positive number** (MSB still indicates the sign of the integer):

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0		One’s Complement	Unsigned
0	0	0	0	0	0	0	0	=	+0	0
0	0	0	0	0	0	0	1	=	1	1
0	0	0	0	0	0	1	0	=	2	2
…	…	…	…	…	…	…	…		…	…
0	1	1	1	1	1	1	1	=	127	127

One's Complement

- A negative number is formed by **reversing the bits of the positive number** (MSB still indicates the sign of the integer):

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0		One's Complement	Unsigned
0	0	0	0	0	0	0	0	=	+0	0
0	0	0	0	0	0	0	1	=	1	1
0	0	0	0	0	0	1	0	=	2	2
…	…	…	…	…	…	…	…		…	…
0	1	1	1	1	1	1	1	=	127	127
1	0	0	0	0	0	0	0	=	**-127**	128
1	0	0	0	0	0	0	1	=	**-126**	129
…	…	…	…	…	…	…	…		…	…
1	1	1	1	1	1	0	1	=	**-2**	253
1	1	1	1	1	1	1	0	=	**-1**	254
1	1	1	1	1	1	1	1	=	**-0**	255

One’s Complement

- **The range of n-bit one’s complement numbers is:**

  $[-2^{n-1}-1, 2^{n-1}-1]$ 8 bits: [-127,127]

- **Addition:**

  Addition of signed numbers in one's complement is performed using binary addition with end-around carry. If there is a carry out of the most significant bit of the sum, this bit must be added to the least significant bit of the sum:

- **Example: 17 + (-8) in 8-bit one’s complement**

```
     0001 0001     (17)
 +   1111 0111     (-8)
 ------------------------
   1 0000 1000
 +           1
 ------------------------
     0000 1001 =    (9)
```

Two’s Complement Numbers

- **Don’t have same problems as sign/magnitude numbers:**
  - Addition works
  - Single representation for 0

- **Has advantages over one’s complement:**
  - Has a single zero representation
  - Eliminates the end-around carry operation required in one's complement addition

Two’s Complement Numbers

- **A negative number is formed by reversing the bits of the positive number (MSB still indicates the sign of the integer) and adding 1:**

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0		Two’s Complement	Unsigned
0	0	0	0	0	0	0	0	=	0	0
0	0	0	0	0	0	0	1	=	1	1
0	0	0	0	0	0	1	0	=	2	2
…	…	…	…	…	…	…	…		…	…
0	1	1	1	1	1	1	1	=	127	127

Two's Complement Numbers

- A negative number is formed by **reversing the bits** of the positive number (MSB still indicates the sign of the integer) **and adding 1**:

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0		Two's Complement	Unsigned
0	0	0	0	0	0	0	0	=	0	0
0	0	0	0	0	0	0	1	=	1	1
0	0	0	0	0	0	1	0	=	2	2
…	…	…	…	…	…	…	…		…	…
0	1	1	1	1	1	1	1	=	127	127
1	0	0	0	0	0	0	0	=	**-128**	128
1	0	0	0	0	0	0	1	=	**-127**	129
…	…	…	…	…	…	…	…		…	…
1	1	1	1	1	1	0	1	=	**-3**	253
1	1	1	1	1	1	1	0	=	**-2**	254
1	1	1	1	1	1	1	1	=	**-1**	255

Two's Complement Numbers

- **Same as unsigned binary, but the most significant bit (msb) has value of -2^{N-1}**

$$I = \sum_{i=0}^{i=n-2} b_i 2^i - b_{n-1} 2^{n-1}$$

  - Most positive 4-bit number:
  - Most negative 4-bit number:

- **The most significant bit still indicates the sign** (1 = negative, 0 = positive)

- **Range of an N-bit two's comp number:**

Two's Complement Numbers

- **Same as unsigned binary, but the most significant bit (msb) has value of -2^{N-1}**

$$I = \sum_{i=0}^{i=n-2} b_i 2^i - b_{n-1} 2^{n-1}$$

  - Most positive 4-bit number: `0111`
  - Most negative 4-bit number: `1000`

- **The most significant bit still indicates the sign** (1 = negative, 0 = positive)

- **Range of an N-bit two's comp number:**

[-2^{N-1}, $2^{N-1}-1$]   8 bits: [-128,127]

"Taking the Two' s Complement"

- **How to flip the sign of a two's complement number:**
  - Invert the bits
  - Add one

- **Example: Flip the sign of** 3_{10} = 0011_2

"Taking the Two's Complement"

- **How to flip the sign of a two's complement number:**
  - Invert the bits
  - Add one

- **Example: Flip the sign of** 3_{10} = 0011_2
  - Invert the bits 1100_2

"Taking the Two's Complement"

- **How to flip the sign of a two's complement number:**
  - Invert the bits
  - Add one

- **Example: Flip the sign of** 3_{10} = 0011_2
  - Invert the bits 1100_2
  - Add one 1101_2

"Taking the Two's Complement"

- **How to flip the sign of a two's complement number:**
  - Invert the bits
  - Add one

- **Example: Flip the sign of 3_{10} = 0011_2**
  - Invert the bits 1100_2
  - Add one 1101_2

- **Example: Flip the sign of -8_{10} = 11000_2**

"Taking the Two' s Complement"

- **How to flip the sign of a two's complement number:**
  - Invert the bits
  - Add one

- **Example: Flip the sign of** 3_{10} = 0011_2
  - Invert the bits 1100_2
  - Add one 1101_2

- **Example: Flip the sign of** -8_{10} = 11000_2
  - Invert the bits 00111_2
  - Add one 01000_2

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**

```
 111
   0110
+  1010
-------
  10000
```

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

```
 111
   1110
+  0011
-------
  10001
```

- **Correct results if overflow bit is ignored**

Increasing Bit Width

- **A value can be extended from N bits to M bits (where M > N) by using:**
  - Sign-extension
  - Zero-extension

Sign-Extension

- **Sign bit is copied into most significant bits**
- **Number value remains the same**
- **Give correct result for two's complement numbers**
- **Example 1:**
  - 4-bit representation of 3 = `0011`
  - 8-bit sign-extended value: `00000011`
- **Example 2:**
  - 4-bit representation of -5 = `1011`
  - 8-bit sign-extended value: `11111011`

Zero-Extension

- **Zeros are copied into most significant bits**
- **Value will change for negative numbers**
- **Example 1:**
  - 4-bit value = $0011_2 = 3_{10}$
  - 8-bit zero-extended value: $00000011_2 = 3_{10}$
- **Example 2:**
  - 4-bit value = $1011_2 = -5_{10}$
  - 8-bit zero-extended value: $00001011_2 = 11_{10}$

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								

Lessons Learned

- **How to express decimal numbers using only 1s and 0s**
- **How to simplify writing binary numbers in hexadecimal**
- **Adding binary numbers**
- **Methods to express negative numbers**
  - Sign Magnitude
  - One's complement
  - **Two's complement (the one commonly used)**