DATA STORAGE

COMS W1001
Introduction to Information Science

Boyi Xie

Today's Topics

- Bits and Logic Gates
- Bits, Bytes and the Binary System
- Binary, Decimal, Octal, Hexadecimal
- Representing Numbers
- Representing Text

Bits and Logic Gates

- Bit – binary digits: on/off, true/false, 0/1
- Boolean operation – in honor of mathematician George Boole
  - AND

	0		0		1		1
AND	0	AND	1	AND	0	AND	1
	0		0		0		1

  - OR

	0		0		1		1
OR	0	OR	1	OR	0	OR	1
	0		1		1		1

  - NOT
    - NOT 0 -> 1
    - NOT 1 -> 0
  - XOR (exclusive or)

	0		0		1		1
XOR	0	XOR	1	XOR	0	XOR	1
	0		1		1		0

Bits and Logic Gates

- Gate – A device that produces the output of a Boolean operation when given operation's input value
  - Electronic circuits in modern computer
  - Represent digits 0 and 1 using voltage levels

AND

Inputs	Output
0 0	0
0 1	0
1 0	0
1 1	1

OR

Inputs	Output
0 0	0
0 1	1
1 0	1
1 1	1

XOR

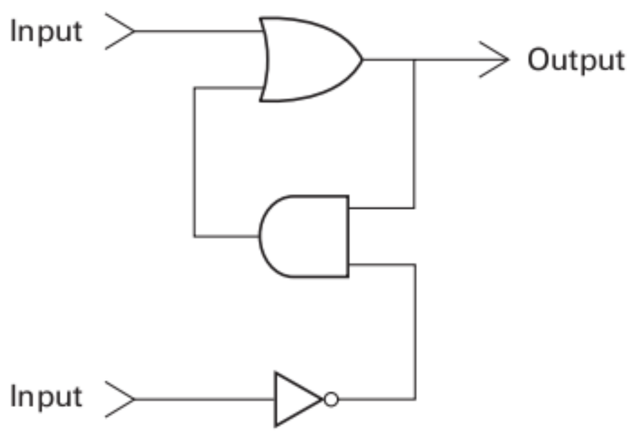
Inputs	Output
0 0	0
0 1	1
1 0	1
1 1	0

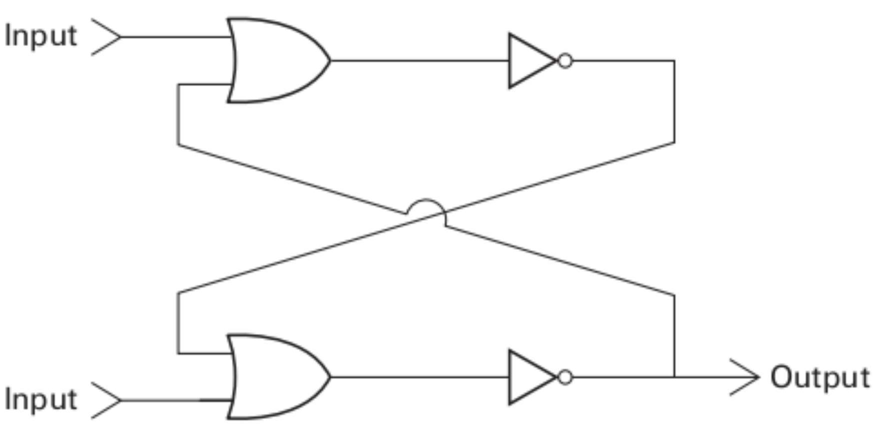
NOT

Inputs	Output
0	1
1	0

Bits and Logic Gates

- Flip-flops
  - A circuit that produces an output value of 0 or 1, which remains constant until a temporary pulse from another circuit causes it to shift to the other value
  - Consider the following two constructions of flip-flops
    - As long as both inputs remain 0, the output will not change
    - Temporarily giving a signal 1 on the upper input will force the output to be 1
    - Temporarily giving a signal 1 on the lower input will force the output to be 0





Bits and Logic Gates

- VLSI – Very large-scale integration
  - A technology that millions electronic components (e.g. flip-flops) are used inside a computer (on a wafer, or called chip) as a means of recording information that is encoded as patterns of 0s and 1s

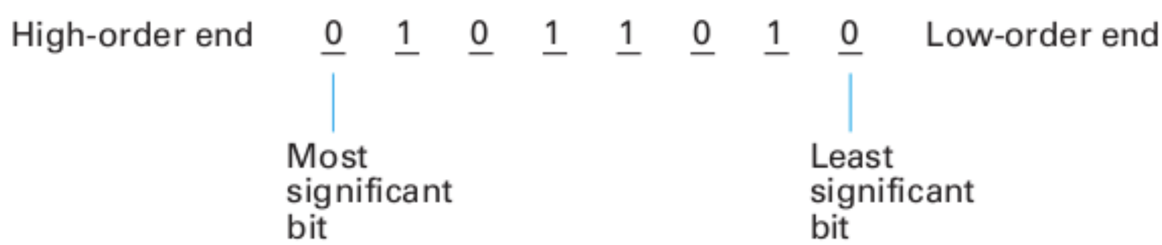
Bits, Bytes and the Binary System

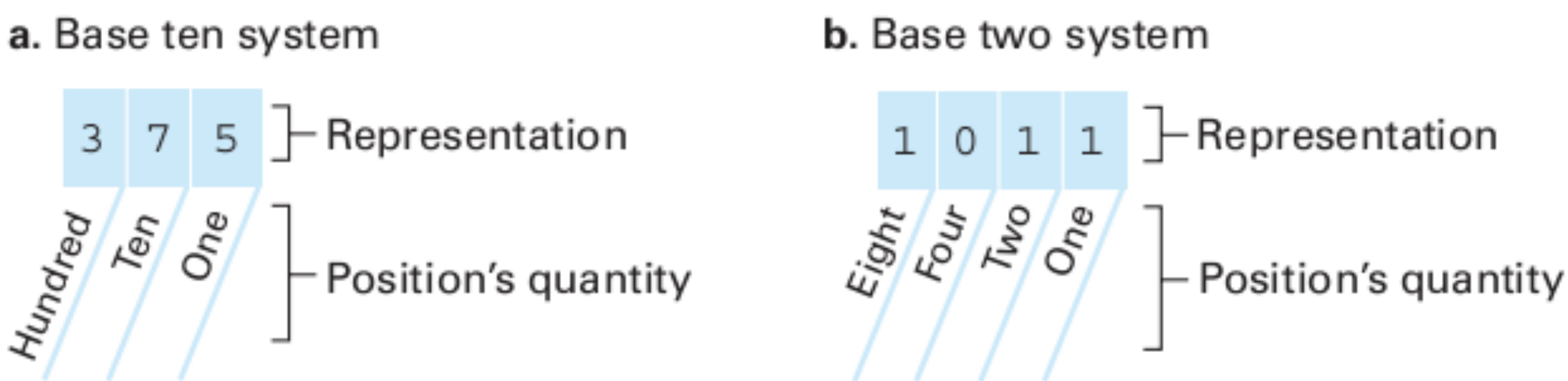
- Byte – 8 bits
- Kilobyte (KB) – 1024 bytes (2^{10} bytes)
- Megabyte (MB) – 1024 KB – 1,048,576 bytes (2^{20} bytes)
- Gigabyte (GB) – 1024 MB – 1,073,741,824 bytes (2^{30} bytes)
- Terabyte (TB) – 1024 GB – 1,099,511,627,776 bytes (2^{40} bytes)
- Petabype (PB)
- …

The Binary System

- A means of representing numeric value (and other information) using only 0 and 1

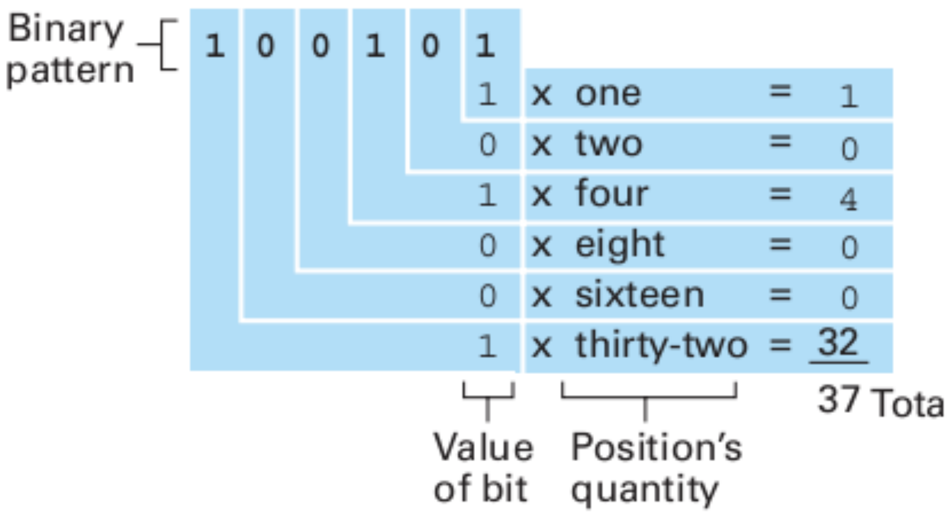
- Binary Notation





The Binary System

- Conversion from binary to decimal



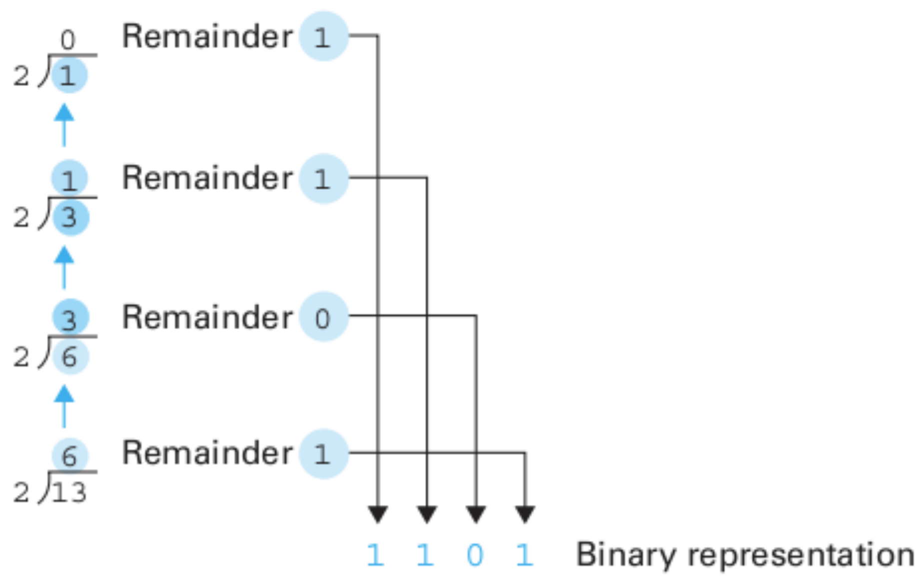
The Binary System

- Conversion from decimal to binary

Step 1. Divide the value by two and record the remainder.

Step 2. As long as the quotient obtained is not zero, continue to divide the newest quotient by two and record the remainder.

Step 3. Now that a quotient of zero has been obtained, the binary representation of the original value consists of the remainders listed from right to left in the order they were recorded.



The Binary System

- Binary Addition

```
  111010
+  11011
--------
```

```
     1
  111010
+  11011
--------
      01
```

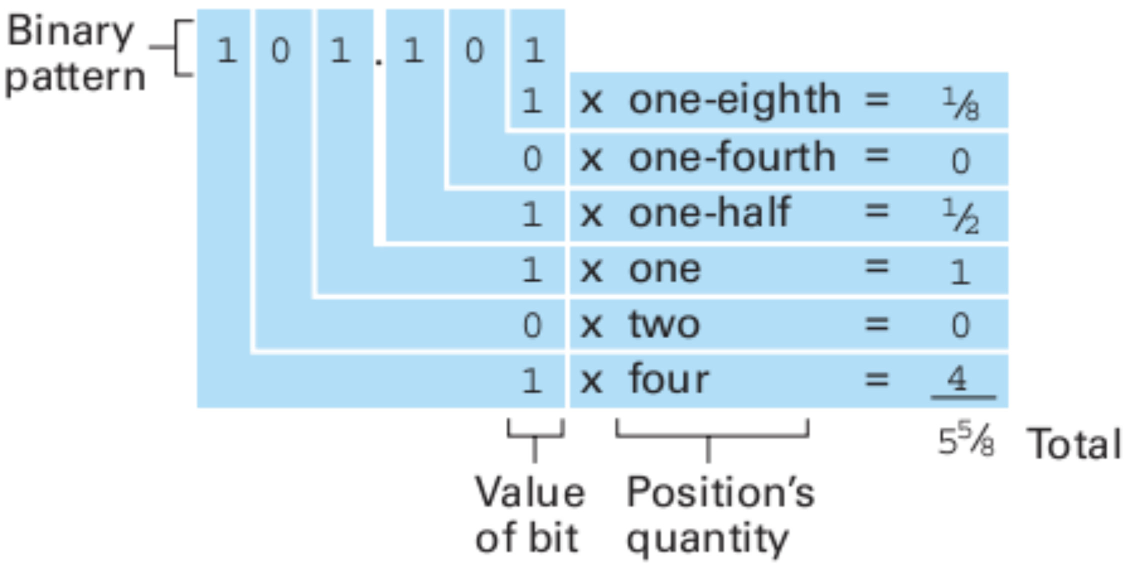
```
   1
  111010
+  11011
--------
    0101
```

```
 1
  111010
+  11011
--------
  010101
```

```
  111010
+  11011
--------
 1010101
```

The Binary System

- Fractions in Binary



- Addition

```
   10.011
+ 100.110
 --------
  111.001
```

Binary, Decimal, Octal, Hexadecimal

- Conversion
  - Binary to decimal
  - Decimal to binary
  - Binary to octal
  - Binary to Hexadecimal

  - To convert decimal to octal
    - First convert decimal to binary
    - Then make 3-bit a group
  - To convert decimal to hexadecimal
    - First convert decimal to binary
    - Then make 4-bit a group

Representing Integers

- Two's Complement Notation

a. Using patterns of length three

Bit pattern	Value represented
011	3
010	2
001	1
000	0
111	−1
110	−2
101	−3
100	−4

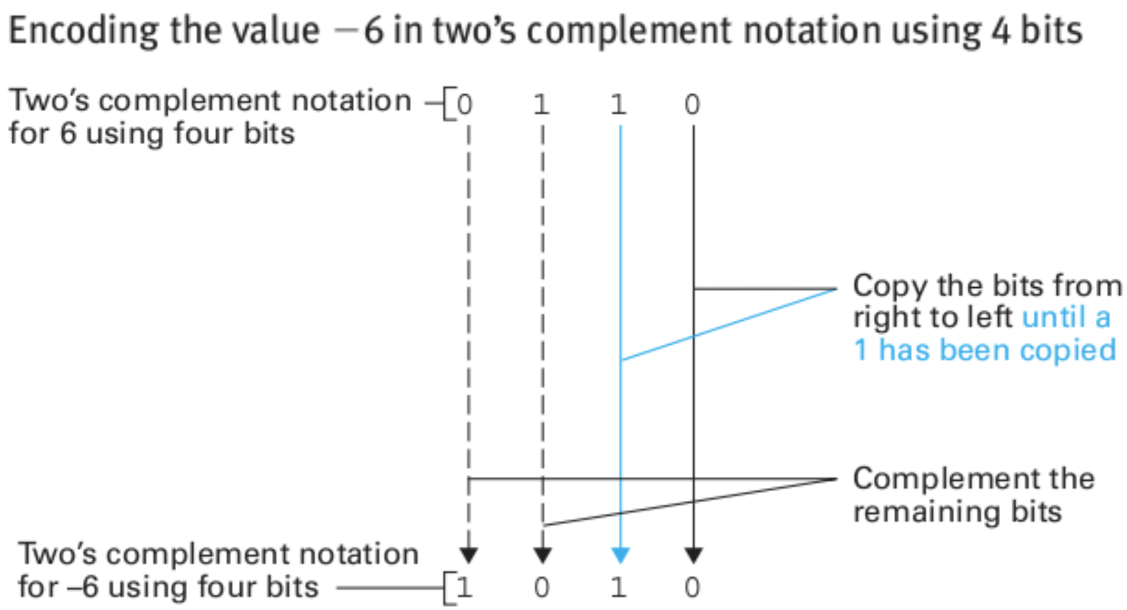
b. Using patterns of length four

Bit pattern	Value represented
0111	7
0110	6
0101	5
0100	4
0011	3
0010	2
0001	1
0000	0
1111	−1
1110	−2
1101	−3
1100	−4
1011	−5
1010	−6
1001	−7
1000	−8

Representing Integers

- Sign bit – leftmost bit
- Complement – 0->1 or 1->0

- Example of negative integer:



Representing Integers

- Addition

Problem in base ten	Problem in two's complement	Answer in base ten
3 + 2	0011 + 0010 = 0101	5
−3 + −2	1101 + 1110 = 1011	−5
7 + −5	0111 + 1011 = 0010	2

- Overflow
  - positive + positive = negative
  - negative + negative = positive

Representing Integers

- Excess Notation
  - 3 bit pattern – excess 4 notation
  - 4 bit pattern – excess 8 notation
  - 5 bit pattern – excess 16 notation
  - …

An excess notation system using bit patterns of length three

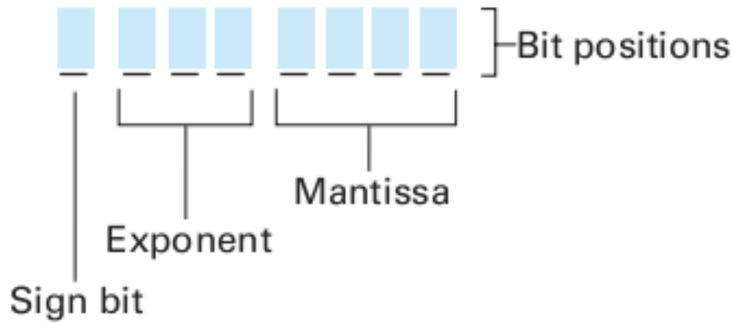
Bit pattern	Value represented
111	3
110	2
101	1
100	0
011	−1
010	−2
001	−3
000	−4

An excess eight conversion table

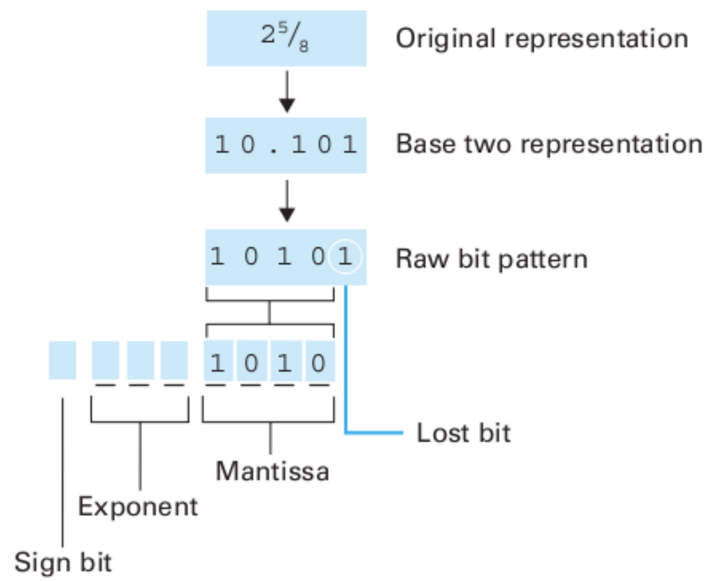
Bit pattern	Value represented
1111	7
1110	6
1101	5
1100	4
1011	3
1010	2
1001	1
1000	0
0111	−1
0110	−2
0101	−3
0100	−4
0011	−5
0010	−6
0001	−7
0000	−8

Representing Fractions

- Floating-Point Notation
  - Decide sign bit
  - Write down the normalized form
  - Filled the exponent and mantissa section



- Truncation Errors



Representing Text

- ASCII – American Standard Code for Information Interchange
  - Use 8 bit per symbol
- Unicode
  - Use 16 bits per symbol

Symbol	ASCII	Hex	Symbol	ASCII	Hex	Symbol	ASCII	Hex
line feed	00001010	0A	>	00111110	3E	^	01011110	5E
carriage return	00001011	0B	?	00111111	3F	_	01011111	5F
space	00100000	20	@	01000000	40	`	01100000	60
!	00100001	21	A	01000001	41	a	01100001	61
"	00100010	22	B	01000010	42	b	01100010	62
#	00100011	23	C	01000011	43	c	01100011	63
$	00100100	24	D	01000100	44	d	01100100	64
%	00100101	25	E	01000101	45	e	01100101	65
&	00100110	26	F	01000110	46	f	01100110	66
'	00100111	27	G	01000111	47	g	01100111	67
(	00101000	28	H	01001000	48	h	01101000	68
)	00101001	29	I	01001001	49	i	01101001	69
*	00101010	2A	J	01001010	4A	j	01101010	6A
+	00101011	2B	K	01001011	4B	k	01101011	6B
,	00101100	2C	L	01001100	4C	l	01101100	6C
-	00101101	2D	M	01001101	4D	m	01101101	6D
.	00101110	2E	N	01001110	4E	n	01101110	6E
/	00111111	2F	O	01001111	4F	o	01101111	6F
0	00110000	30	P	01010000	50	p	01110000	70
1	00110001	31	Q	01010001	51	q	01110001	71
2	00110010	32	R	01010010	52	r	01110010	72
3	00110011	33	S	01010011	53	s	01110011	73
4	00110100	34	T	01010100	54	t	01110100	74
5	00110101	35	U	01010101	55	u	01110101	75
6	00110110	36	V	01010110	56	v	01110110	76
7	00110111	37	W	01010111	57	w	01110111	77
8	00111000	38	X	01011000	58	x	01111000	78
9	00111001	39	Y	01011001	59	y	01111001	79
:	00111010	3A	Z	01011010	5A	z	01111010	7A
;	00111011	3B	[	01011011	5B	{	01111011	7B
<	00111100	3C	\	01011100	5C	|	01111100	7C
=	00111101	3D	]	01011101	5D	}	01111101	7D

References & Photo Credits

- Brookshear, J. Glenn (2011-04-13). Computer Science: An Overview (11th Edition). Prentice Hall. Kindle Edition.