从程序的角度了解计算机系统（1）

code/intro/hello.c

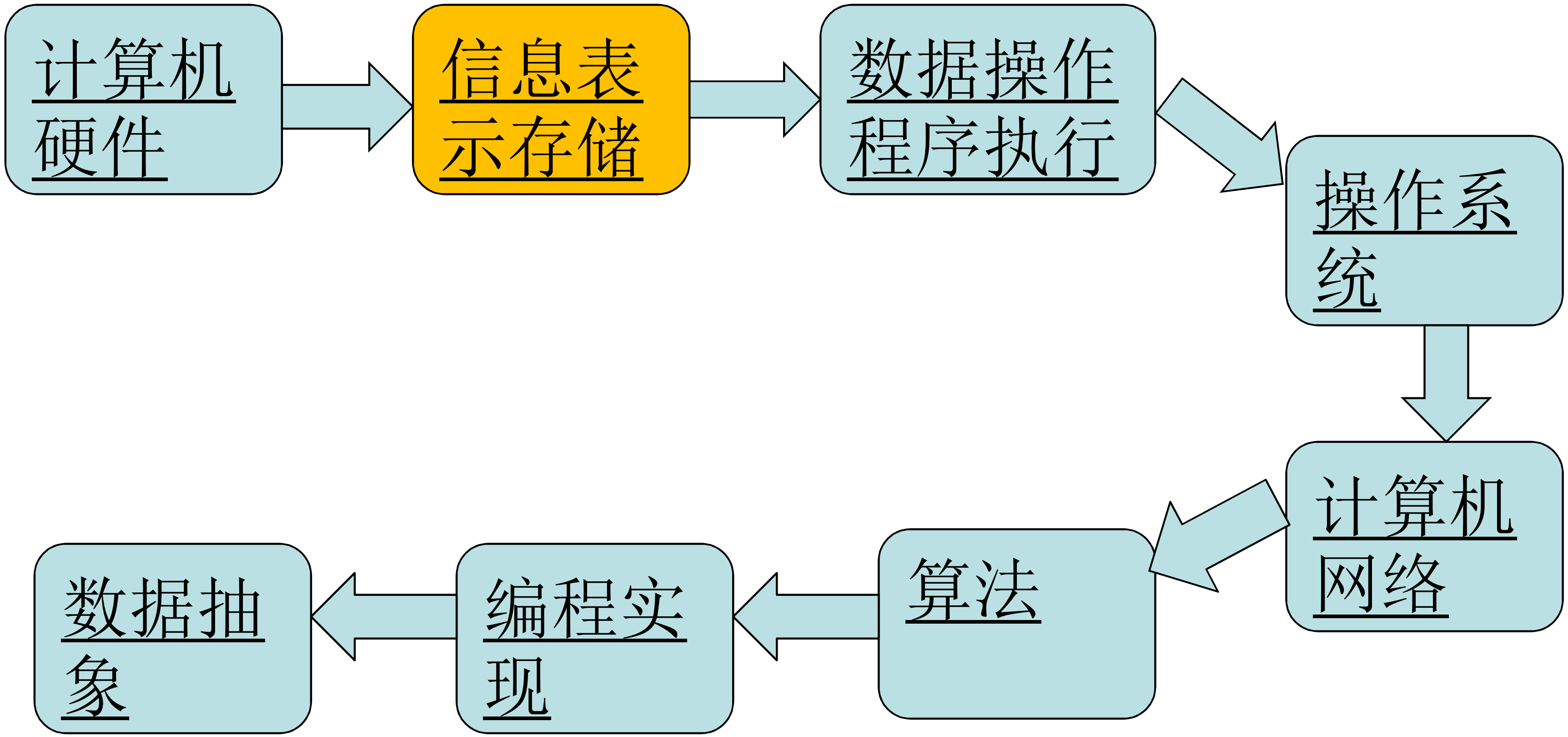
```c
#include <stdio.h>

int main()
{
    printf("hello, world\n");
    return 0;
}
```

code/intro/hello.c

程序生命周期：创建→运行→输出

专业术语、关键概念、组成部分、工作原理

从程序的角度了解计算机系统（2）



Chapter 1

Data Storage

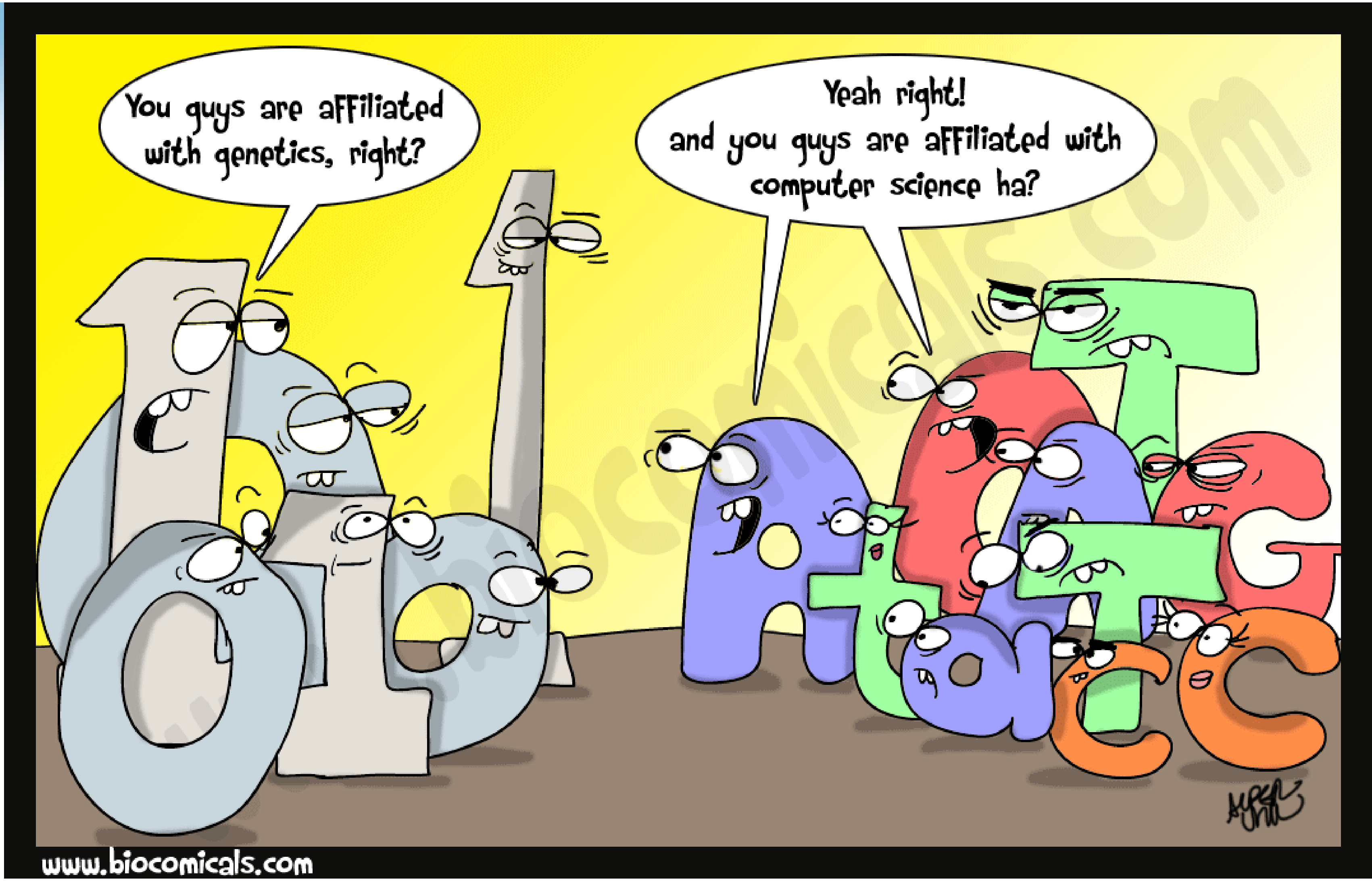
- **Information / data?**

Text, number, sound, image, video

- **Data representation?**

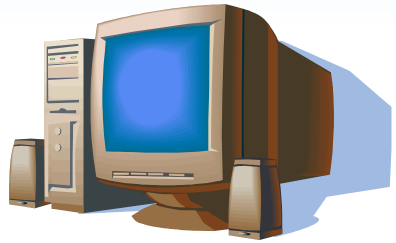
We see?

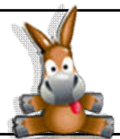
Computers see?



1 / 0 bit位 （Binary：二进制数）

Two Different Worlds



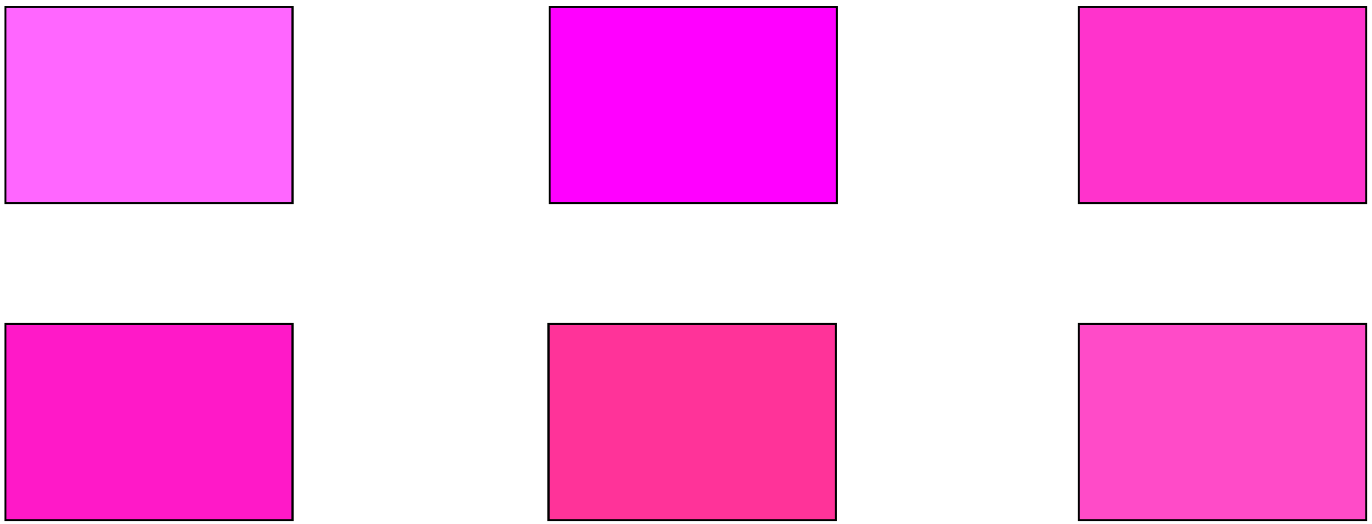
What we see/hear		Inside computers
Text	a,b,c	01100001,01100010,01100011
Number	1,2,3	00000001,00000010,00000011
Sound		010011000101010001101000…
Image		100010010101000000100111...
Video		001100000010011010110001…

Discrete/digital and binary

数字化的二进制的

Problem with Colors

- Which one is pink ?



Why such difficulty?

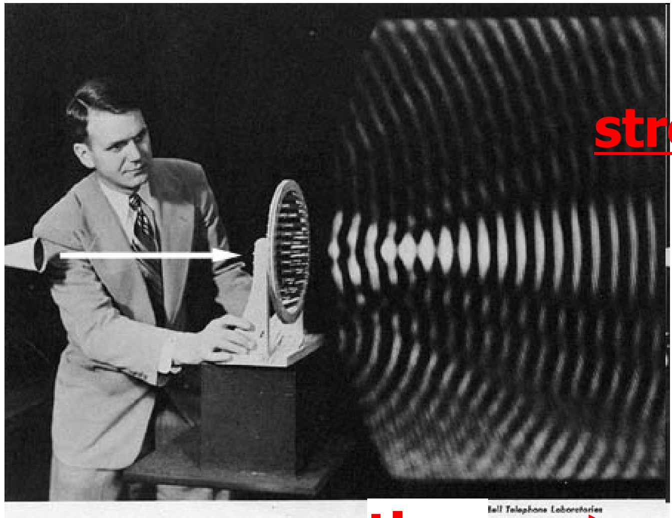
Continuous versus Discrete
连续与离散

- Which are “continuous”?
  - Color
  - Light
  - Cars
  - Sound
  - Height and weight
  - Electric current and voltage
  - English letters

<u>Many natural phenomena are continuous</u>

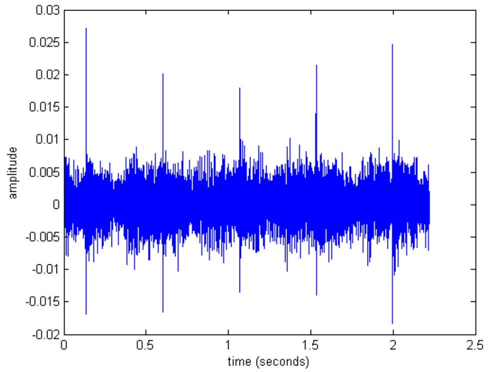
Represent Continuous Things

- Analog signal(模拟信号): simulation of a continuous time varying quantity
  – Voltage or current is an "analog" of the sound



strength

time



voltage

time

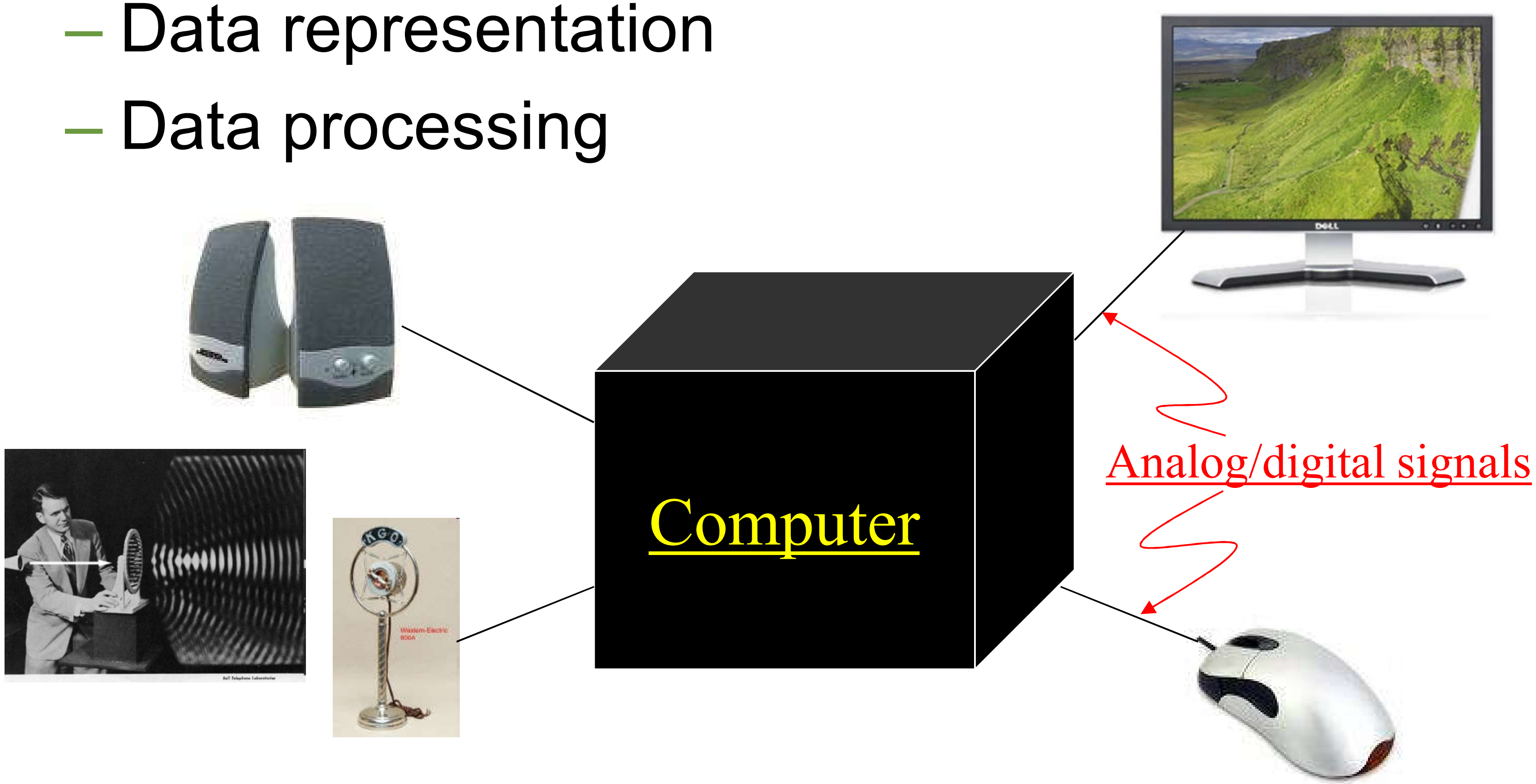
Alternative: Digital/Discrete数字的/离散的

- How many colors in a rainbow?

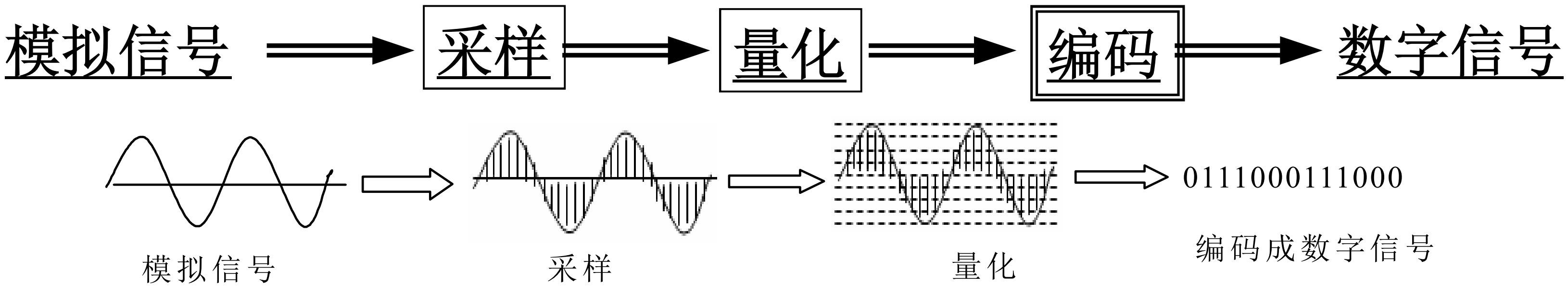
Seven only? → discretization

Computers Work with Signals

- What are inside the "black box"?
  - Data representation
  - Data processing



模拟信号的数字化



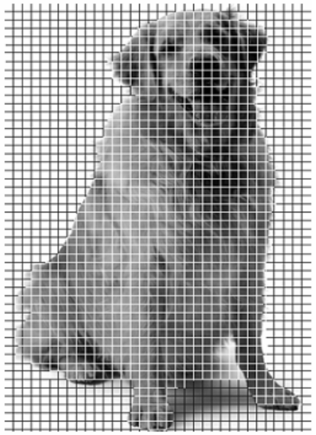
采样 每隔一定时间间隔对模拟波形上取一个幅度值。

量化 将每个采样点得到的幅度值以数字存储。

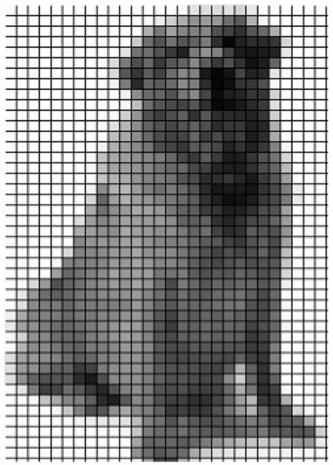
编码 将采样和量化后的数字数据以一定的格式记录下来

图像的数字化

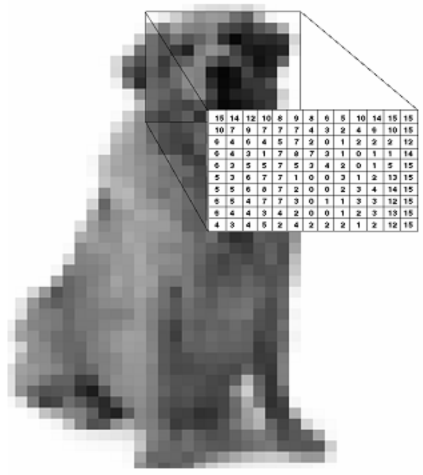
图像



采样



量化



数字图像

Binary System （二进制系统）

- Computers use 0 and 1 to represent and store all kinds of data
- Why binary?
  - We need to find physical objects/phenomena to store, transmit, and process data. Binary is the most straightforward representation.

有无 上下 黑白 真伪 胜负

北南 负正 错对 阳阴 关开 1 0

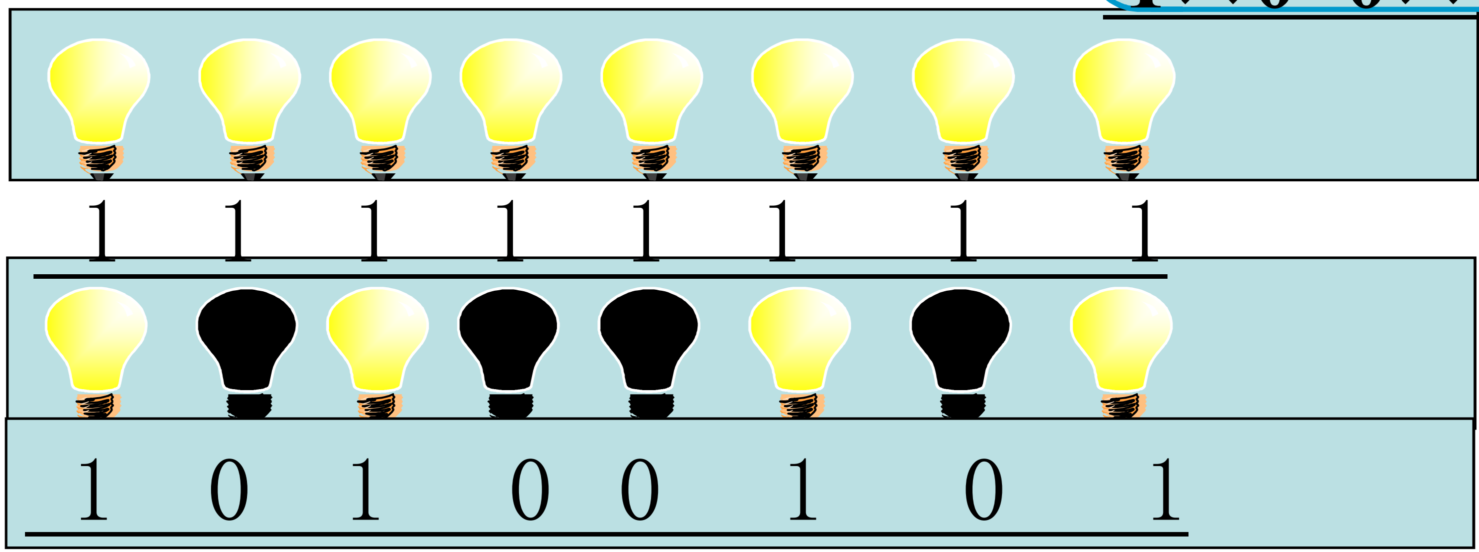
Binary representation

- What kind of computation is needed?
- Why is binary representation used by computer?
  - Easy to implement
  - Simple computation rules

$1 \times 1 = 1$

$0 \times 0 = 0$

$1 \times 0 = 0 \times 1 = 0$



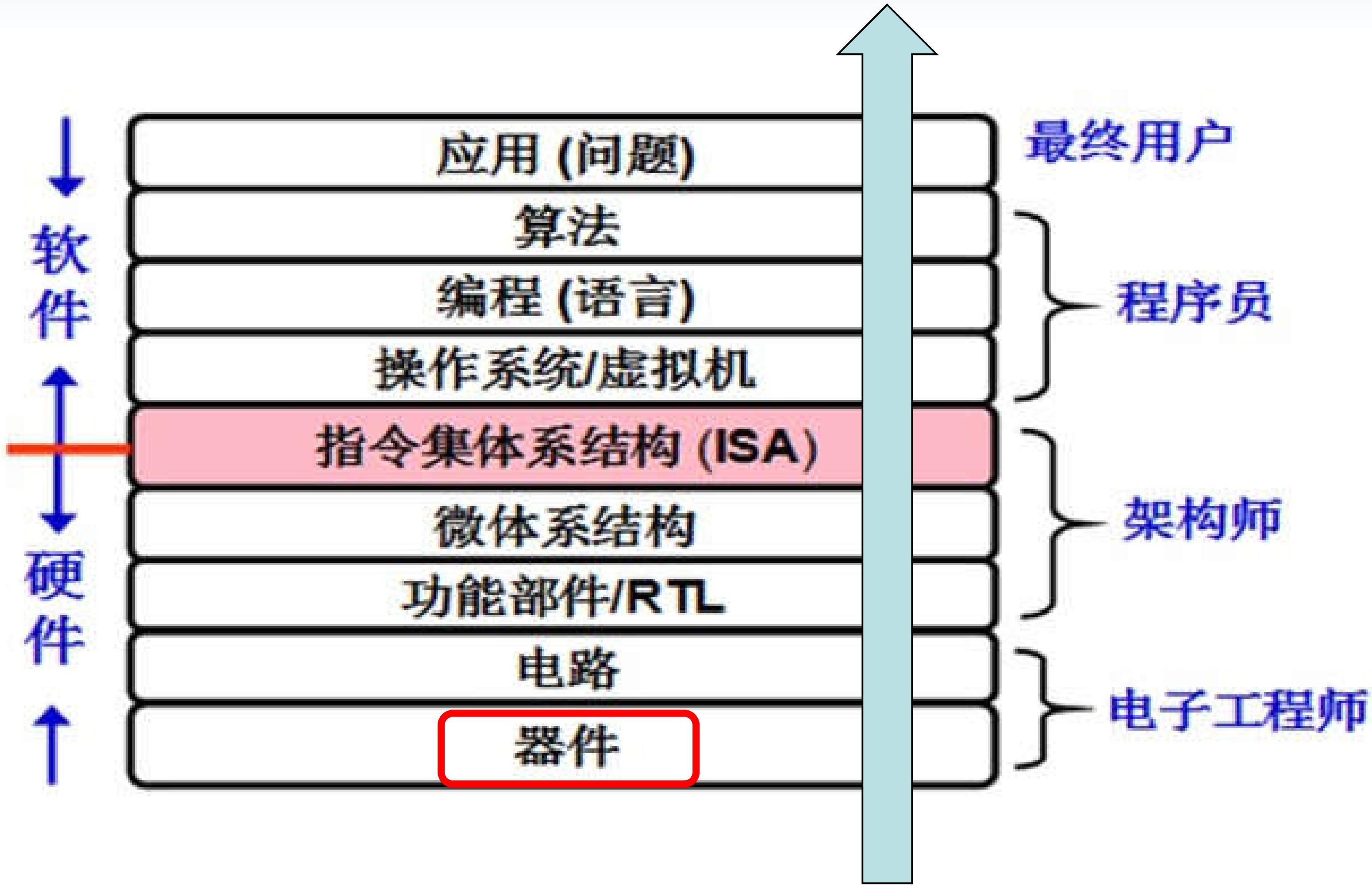
Bits and Bit Patterns

- **Bit（位）:** Binary Digit (0 or 1)
- Bit Patterns are used to represent information.
  - Numbers
  - Text characters
  - Images
  - Sound
  - And others

How is the bit information generated by the computer?

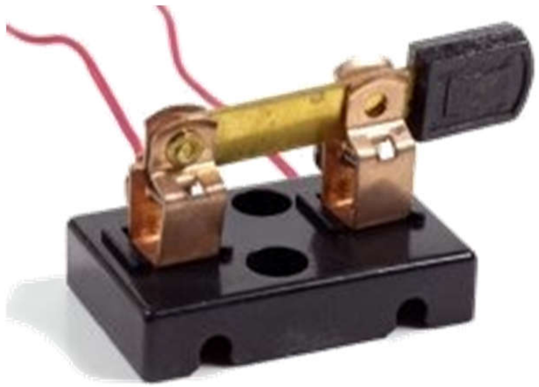
1.1 Bits and their storage pp.20

Basic operations for binary data and the physical devices to implement them


应用 (问题)
算法
编程 (语言)
操作系统/虚拟机
指令集体系结构 (ISA)
微体系结构
功能部件/RTL
电路
器件
软件
硬件
最终用户
程序员
架构师
电子工程师

Electric Switch （电气开关）

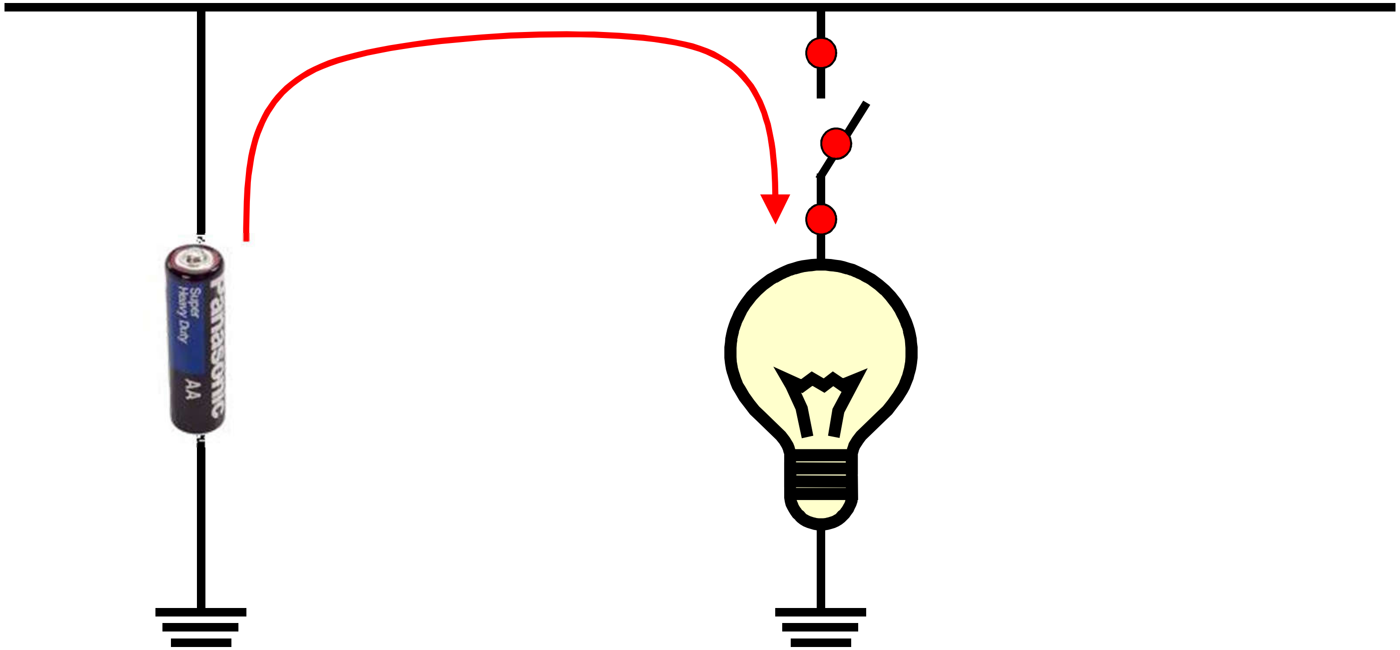
- What are the inputs and outputs?



Switch in a Circuit

- How many "states"?

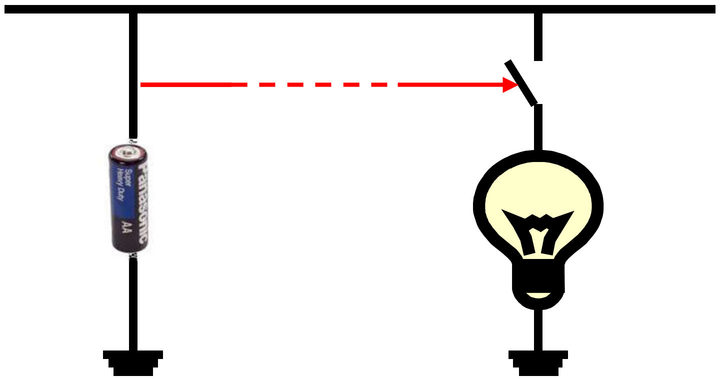
ON, OFF



How to Turn on a Switch?

- By hand

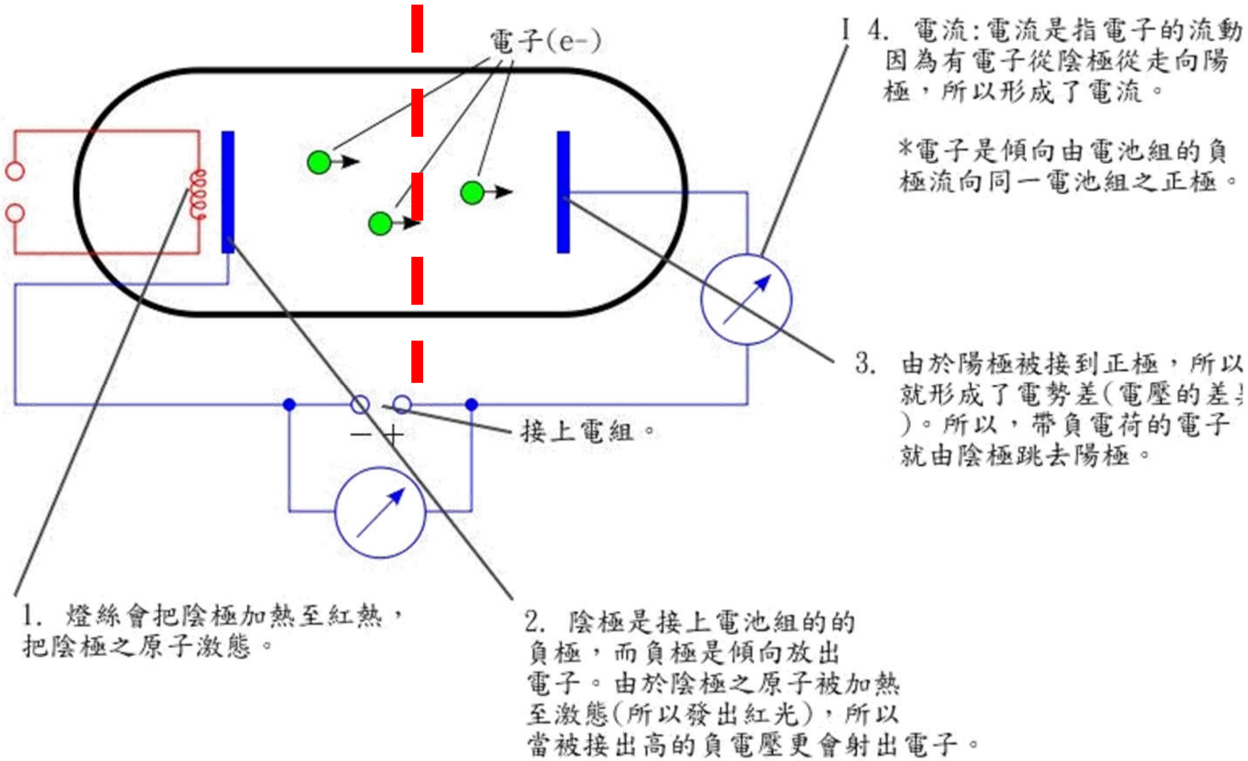
- By electricity?
  - Why do we want to do that?



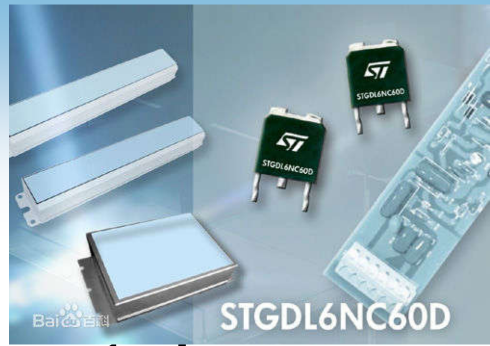
Let’s first study how to operate on ON/OFF

Electronic Switch

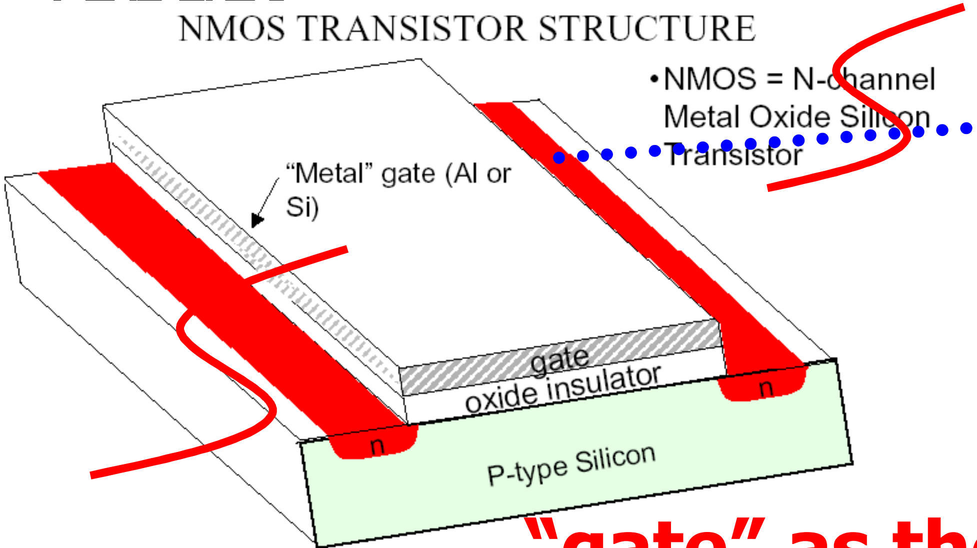
- The earliest one is the *vacuum tube*真空管

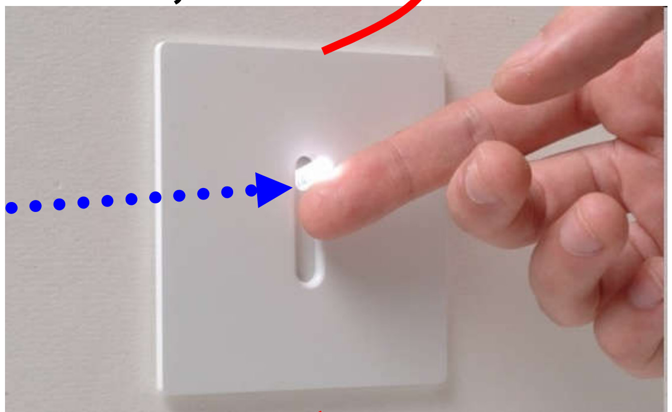


Transistor晶体管



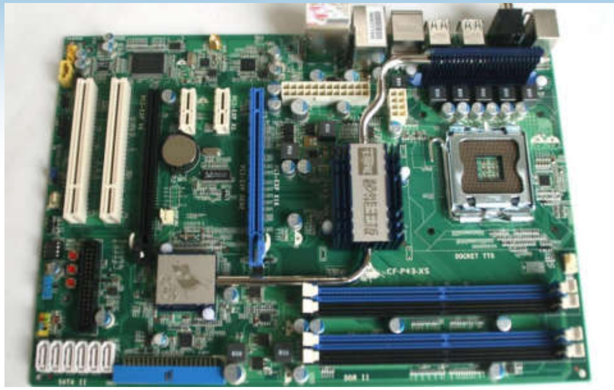
- The problems of vacuum tubes are slow, large, expensive, easy to break
- Transistor can be faster, smaller, and more robust

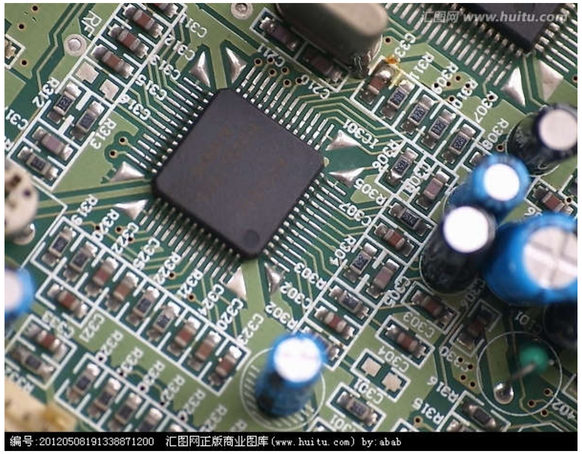


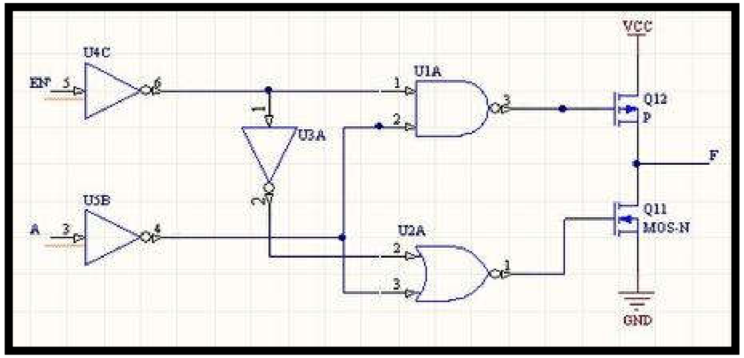


"gate" as the switch


LCD Monitor
Hard Disk Drive
DVD/CD-ROM
Floppy Drive
Motherboard
Keyboard
Mouse



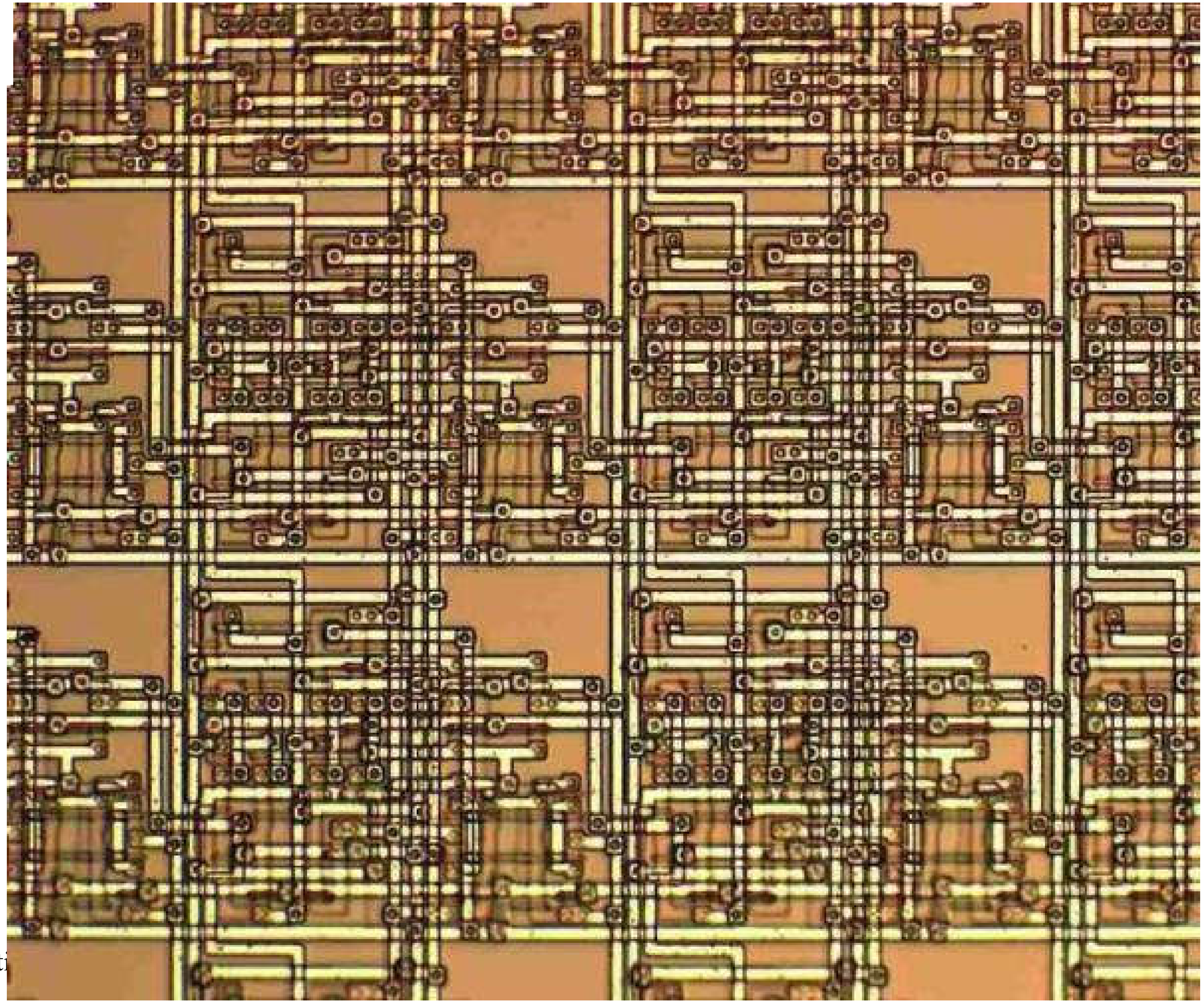


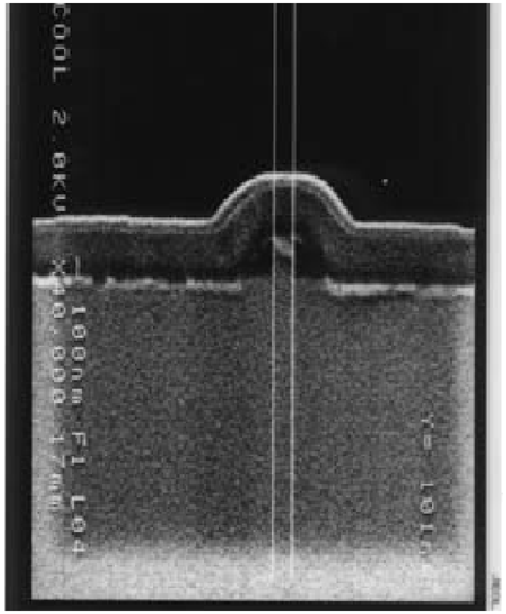

VCC
U4C
EN
U1A
Q12
P
U3A
F
U5B
A
Q11
MOS-N
U2A
GND

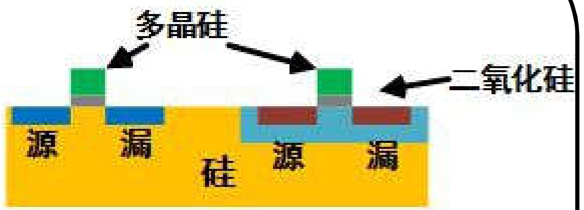
门电路

晶体管 0/1

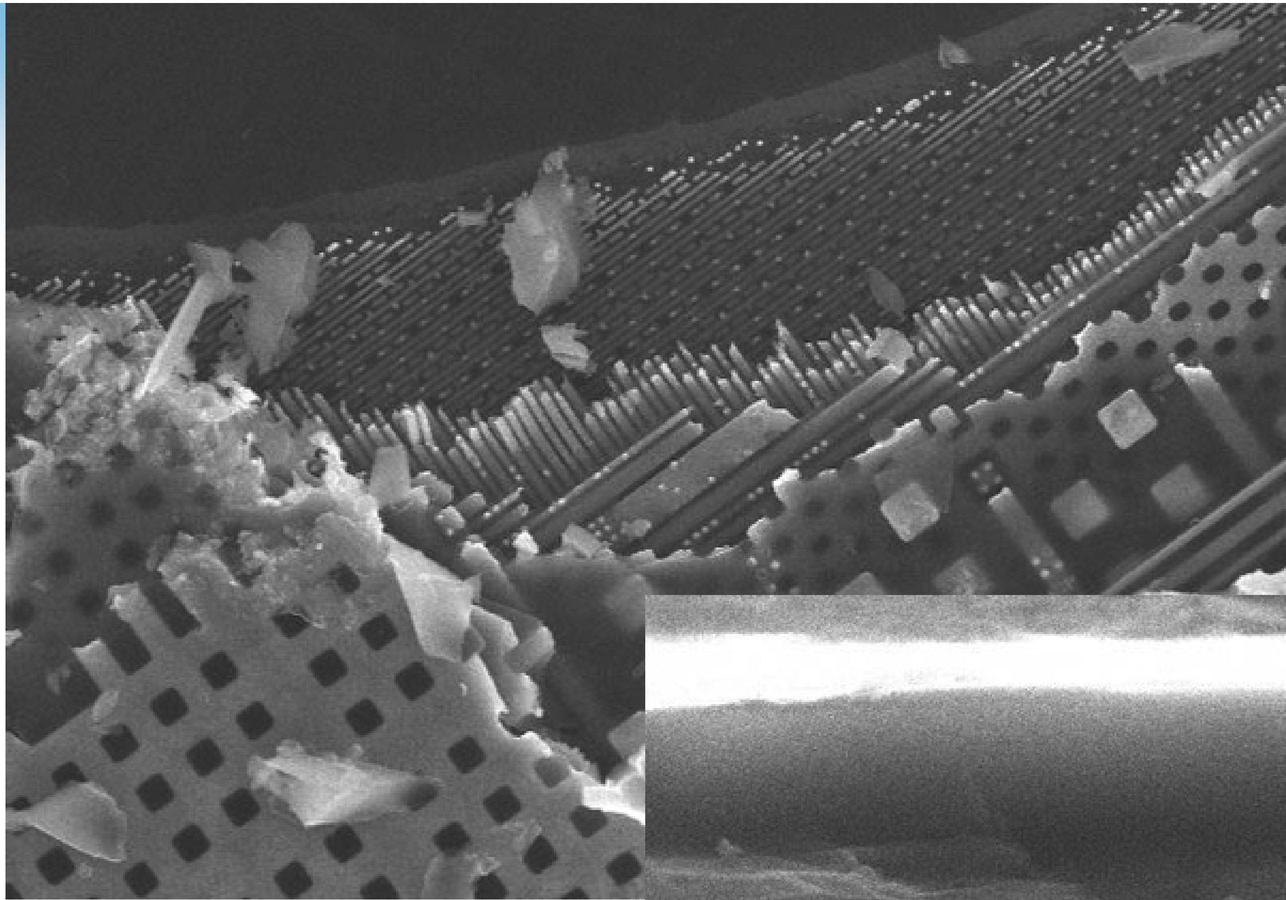
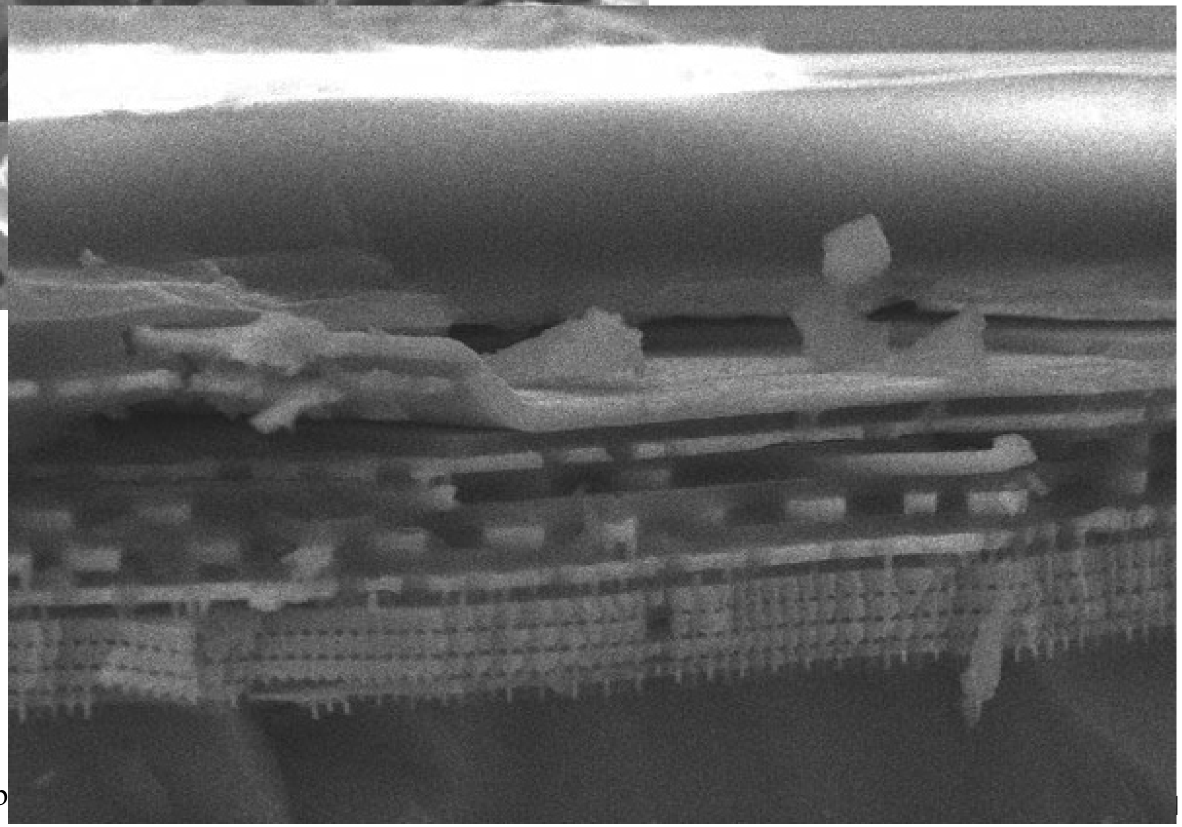

intel
Lynnfield
intel
P55







晶体管 0/1

Binary and Logic逻辑 (Sec. 1.1)

- Logic: concerns about true or false
- Logic operation:
  - If the room is dark and someone is in the room, turn on the light.

Room is dark: Yes (1) / No (0)

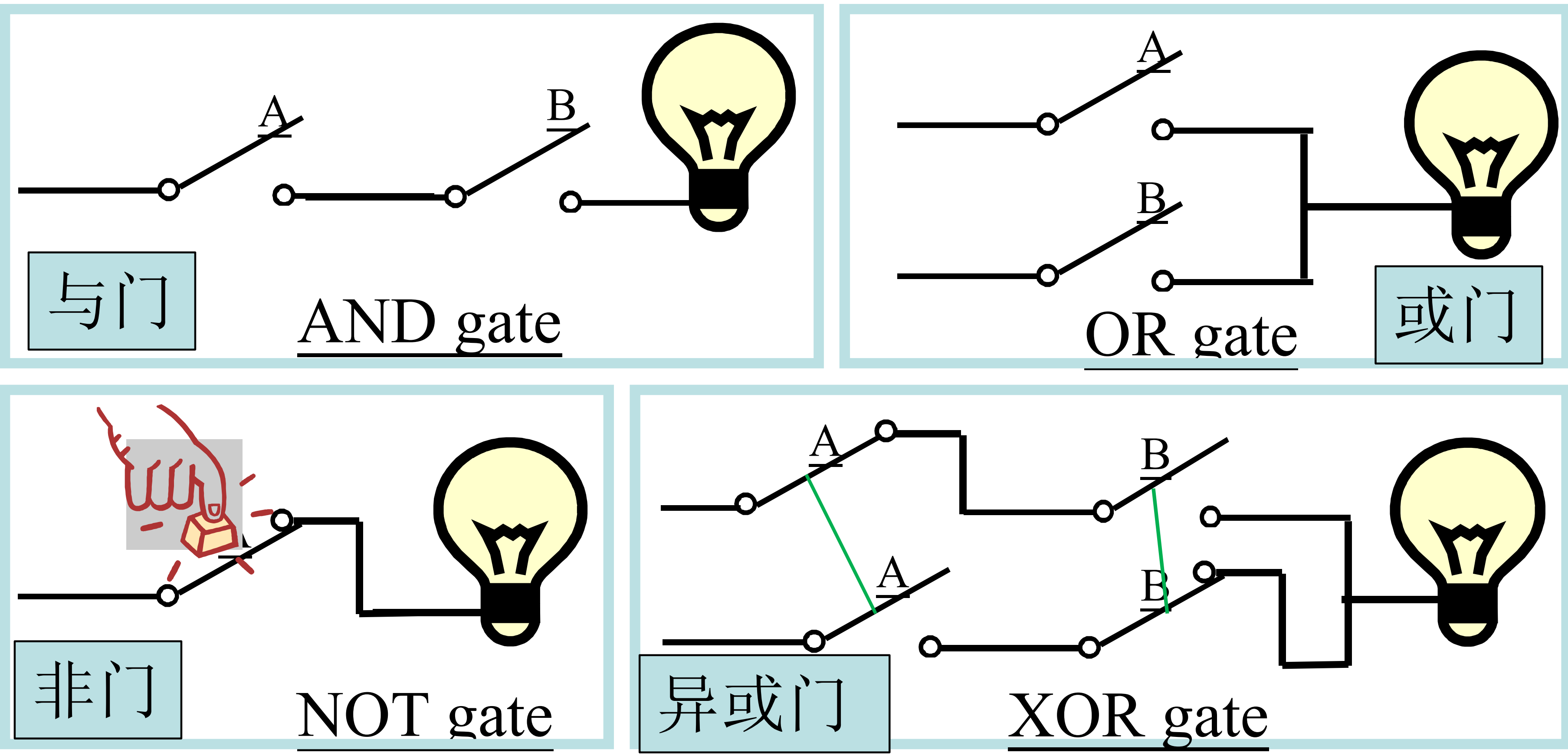
Someone in the room: Yes (1) / No (0)

Light is on: Yes (1) / No (0)

- True/false can be represented by 0/1

  Binary number system in computer ←→ logic

Implement Gate with Switch
用开关实现门（器件）



- Can we flip the switches without hands?

The AND Function

- We can use the AND function to represent the statement

Room is dark A	Someone in the room B	Light is on A .AND. B
0	0	0
0	1	0
1	0	0
1	1	1

$\underbrace{\text{Room is dark (A), Someone in the room (B)}}_{\text{Input}}$ $\underbrace{\text{Light is on (A .AND. B)}}_{\text{Output}}$

Boolean Operations （布尔运算）

- **Boolean Operation:** An operation that manipulates one or more true/false values
- Specific operations
  - AND
  - OR
  - XOR (exclusive or)
  - NOT

Figure 1.1 **The Boolean operations AND, OR, and XOR (exclusive or)**

The AND operation

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

The OR operation

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

The XOR operation

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

Gates门（器件）

- **Gate:** A device that computes a Boolean operation
  - Often implemented as (small) electronic circuits
  - Provide the building blocks from which computers are constructed
  - VLSI (Very Large Scale Integration)

Figure 1.2 **A pictorial representation of AND, OR, XOR, and NOT gates as well as their input and output values**

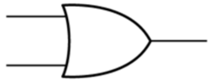
AND

与门

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

OR

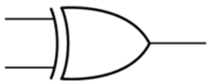
或门



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

XOR

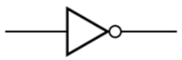
异或门



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

NOT

非门



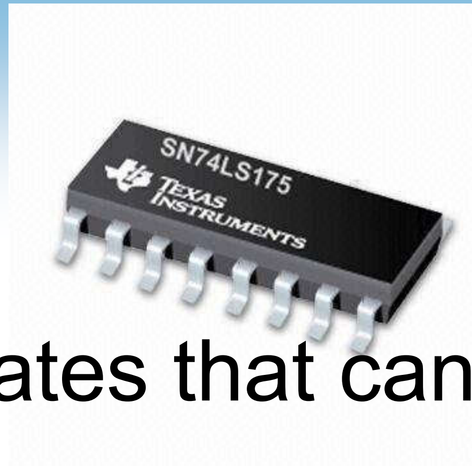
Inputs	Output
0	1
1	0

BIG Idea

- Computers store and process binary
- Logic true and false can be used to represent binary 1 and 0
- Logic operations can be implemented by logic gates
  - and in turn by ON/OFF switches
- Computers can be implemented using logic gates → for storing and processing

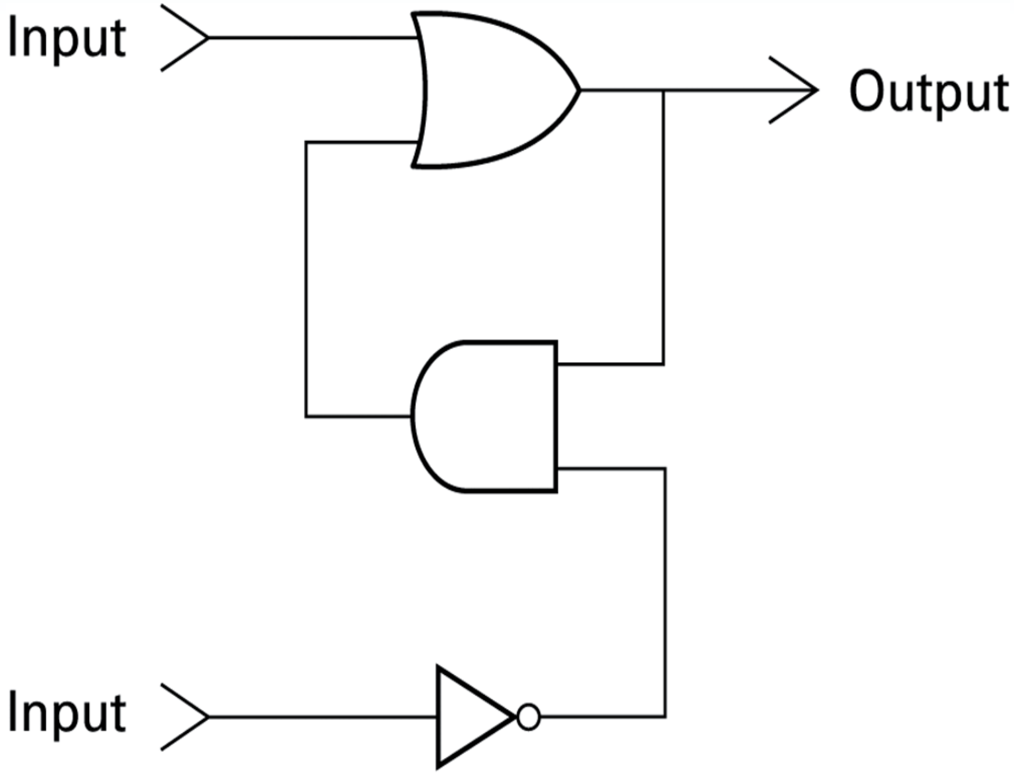
How is the bit information stored by the computer?

Flip-flops （触发器）



- **Flip-flop:** A circuit built from gates that can store one bit.
  - One input line is used to set its stored value to 1
  - One input line is used to set its stored value to 0
  - While both input lines are 0, the most recently stored value is preserved

Figure 1.3 **A simple flip-flop circuit**



Input1（上）	Input2（下）	Output	
0	0	保持不变（0或1）	
1	0	1	即使Input1再回到0，输出仍将一直为1
0	1	0	即使Input2再回到0，输出仍将一直为0

Figure 1.4 **Setting the output of a flip-flop to 1**

a. 1 is placed on the upper input.

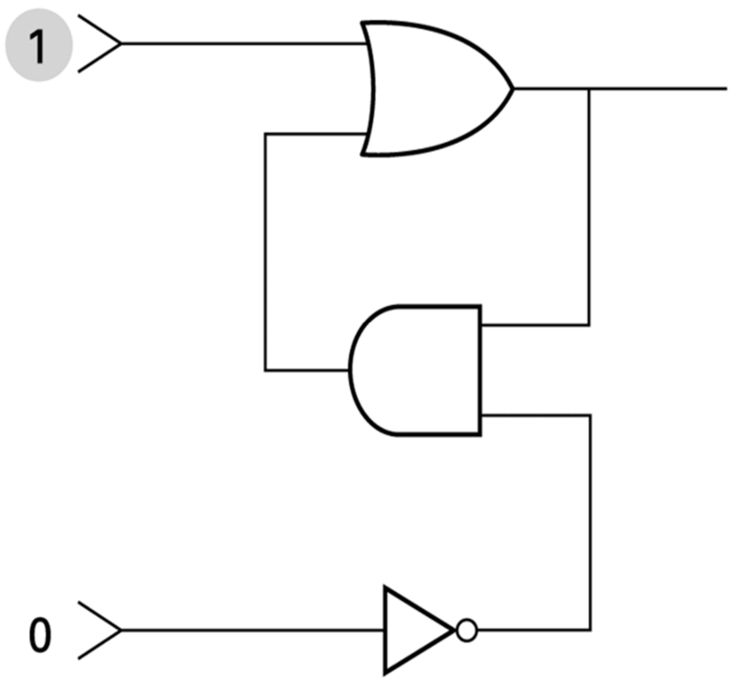


Figure 1.4 **Setting the output of a flip-flop to 1 (continued)**

b. This causes the output of the OR gate to be 1 and, in turn, the output of the AND gate to be 1.

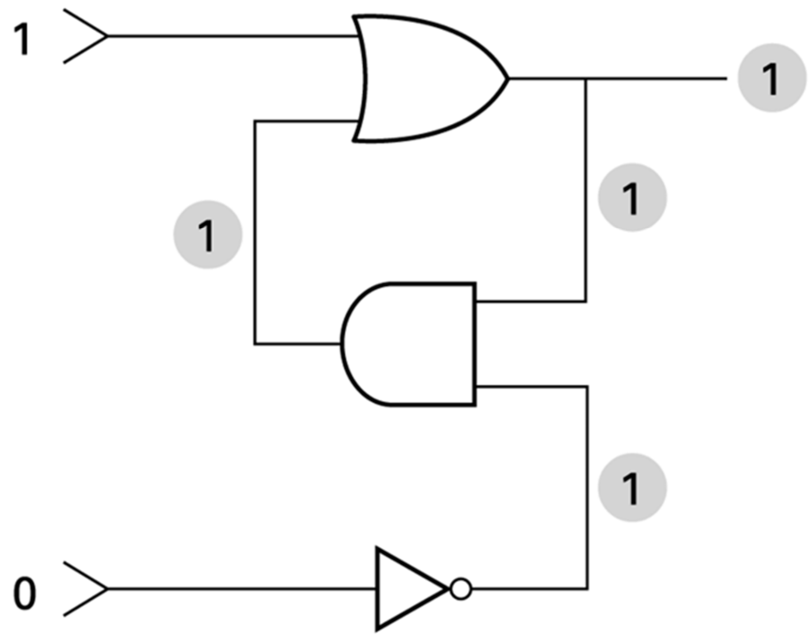
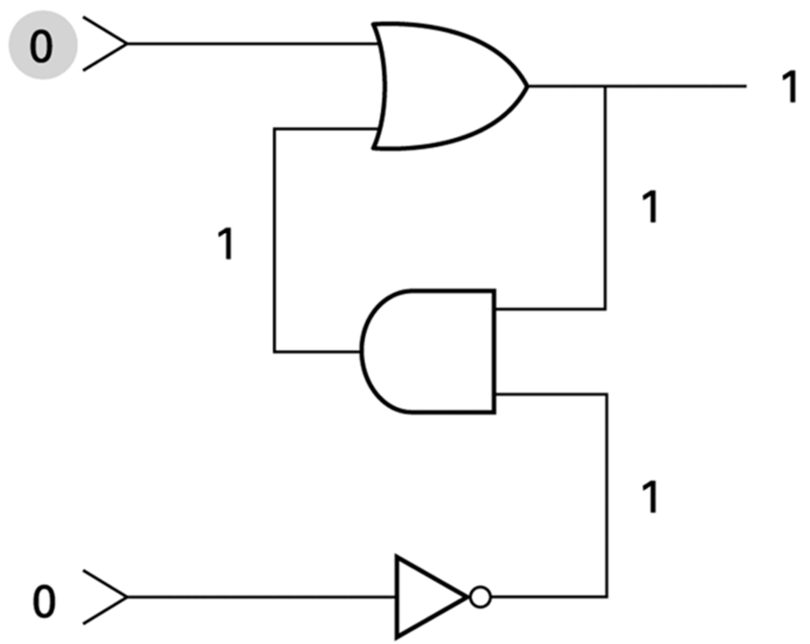


Figure 1.4 **Setting the output of a flip-flop to 1 (continued)**

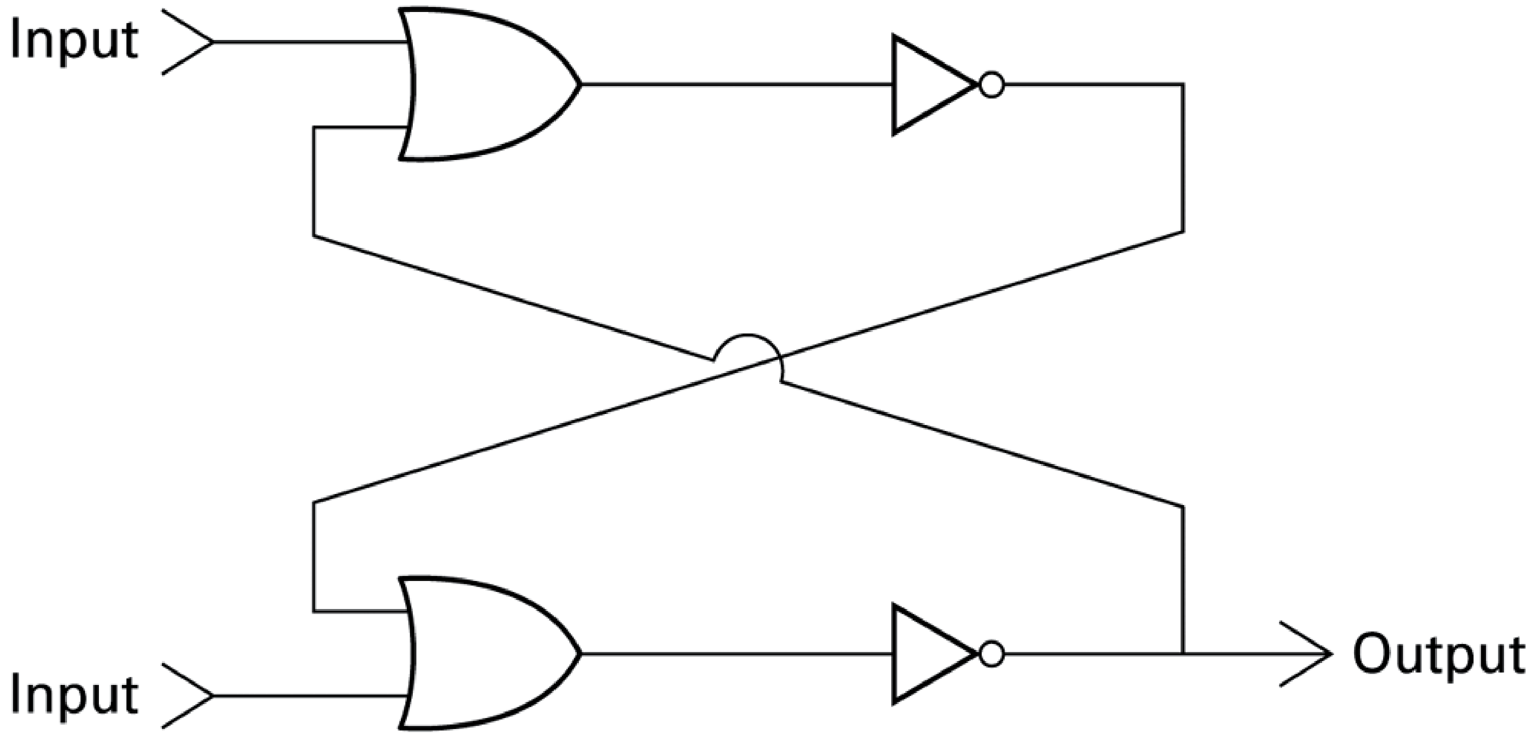
c. The 1 from the AND gate keeps the OR gate from changing after the upper input returns to 0.



Flip-flop

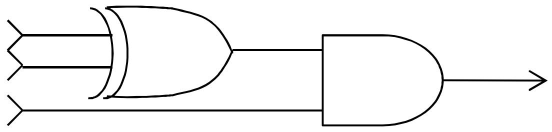
- Shows how a device can be constructed from gates
- Example of abstraction
- Store a bit

Figure 1.5 **Another way of constructing a flip-flop**

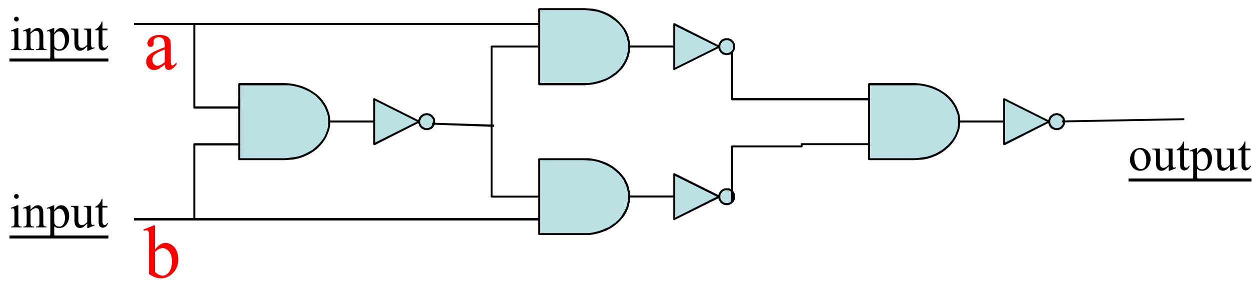


Exercises

- What input bit patterns will cause the following circuit to output 1? And output 0?



- What Boolean operation does the circuit compute?



$$\overline{\left(\overline{(a\wedge\overline{(a\wedge b)})}\wedge\overline{(b\wedge\overline{(a\wedge b)})}\right)}$$

$$=(a\wedge\overline{(a\wedge b)})\vee(b\wedge\overline{(a\wedge b)})$$

$$=(a\vee b)\wedge\overline{(a\wedge b)}$$

$$=(a\vee b)\wedge(\bar{a}\vee\bar{b})$$

<u>Main Memory Cells</u>

- **Cell:** A unit of main memory (typically 8 bits which is one **byte**)
  - **Most significant bit:** the bit at the left (high-order) end of the conceptual row of bits in a memory cell
  - **Least significant bit:** the bit at the right (low-order) end of the conceptual row of bits in a memory cell

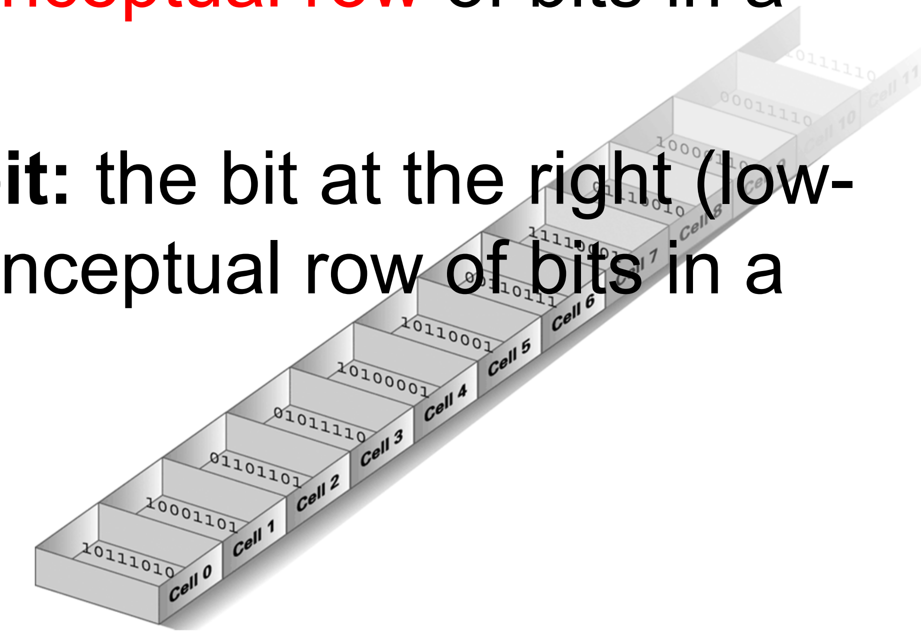
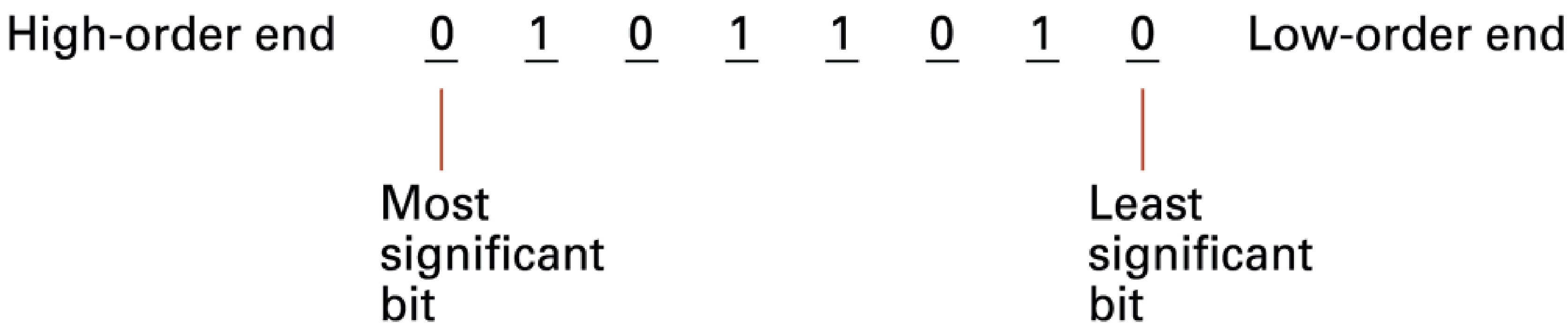


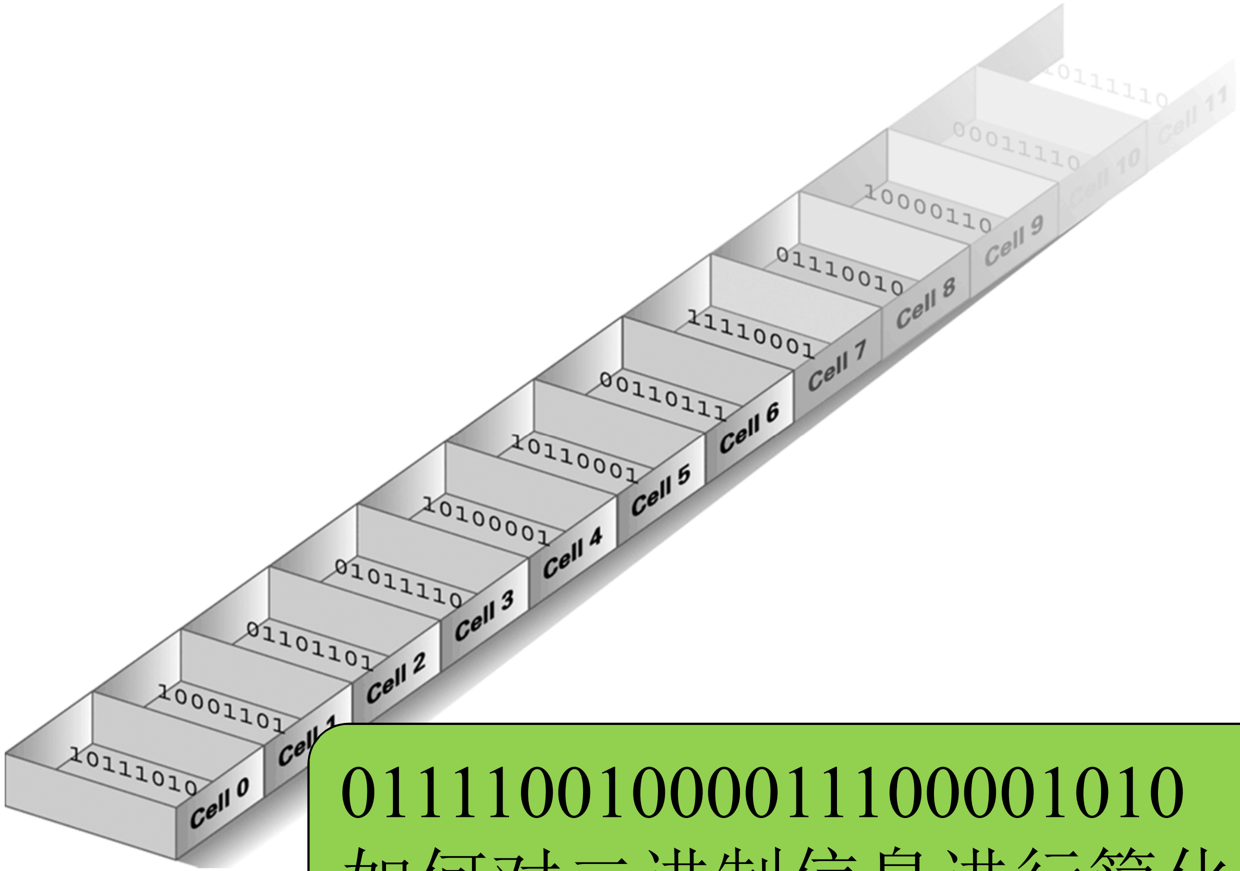
Figure 1.7 **The organization of a byte-size memory cell**



Main Memory Addresses（主存储器地址）

- **Address:** A "name" that uniquely identifies one cell in the computer's main memory
  - The names are actually numbers.
  - These numbers are assigned consecutively starting at zero.
  - Numbering the cells in this manner associates an order with the memory cells.

Figure 1.8 **Memory cells arranged by address**



011110010000011100001010
如何对二进制信息进行简化表示？

Hexadecimal Notation pp.24
十六进制表示

Decimal	Binary	Octal	Hexadecimal
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

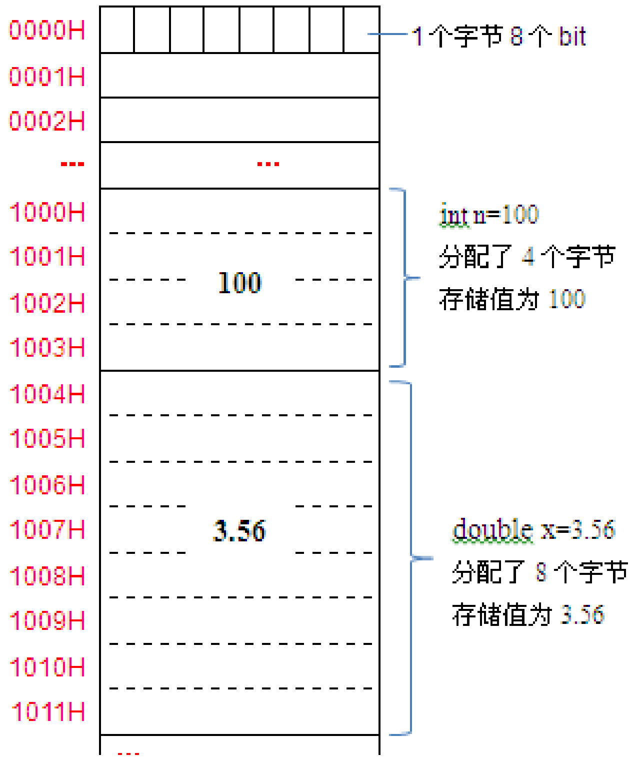
10001000B
88H

内存地址和数据存放

内存：以字节Byte为单位，每个字节有唯一的地址，
就可方便地存取数据。

数据存放：不同的数据类型占据的字节数不同。

```
int    n=100;      //占4个字节
double  x=3.56;   //占8个字节
```



Memory Terminology

- **Random Access Memory (RAM):** Memory in which individual cells can be easily accessed in any order
- **Dynamic Memory (DRAM):** RAM composed of volatile memory (非永久性存储器)

Measuring Memory Capacity

- **Kilobyte:** 2^{10} bytes = 1024 bytes
  - Example: 3 KB = 3 times1024 bytes
  - Sometimes “kibi” rather than “kilo”
- **Megabyte:** 2^{20} bytes = 1,048,576 bytes
  - Example: 3 MB = 3 times 1,048,576 bytes
  - Sometimes “megi” rather than “mega”
- **Gigabyte:** 2^{30} bytes = 1,073,741,824 bytes
  - Example: 3 GB = 3 times 1,073,741,824 bytes
  - Sometimes “gigi” rather than “giga”

Mass Storage pp.29

- On-line versus off-line
- Typically larger than main memory
- Typically less volatile than main memory
- Typically slower than main memory

Mass Storage Systems

- Magnetic Systems
  - Disk
  - Tape
- Optical Systems
  - CD
  - DVD
- Flash Drives

Figure 1.9 **A magnetic disk storage system**

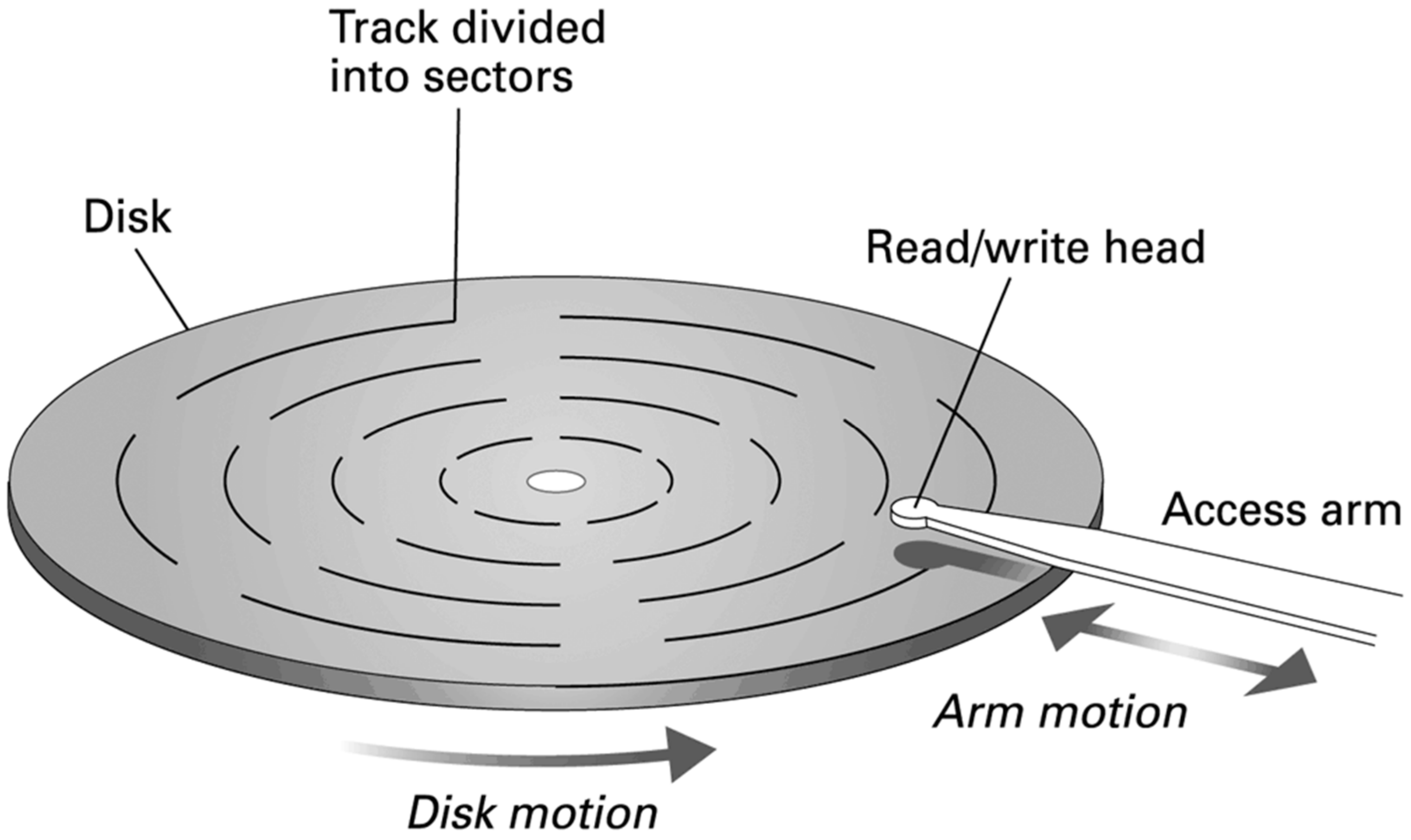


Figure 1.10 **Magnetic tape storage**

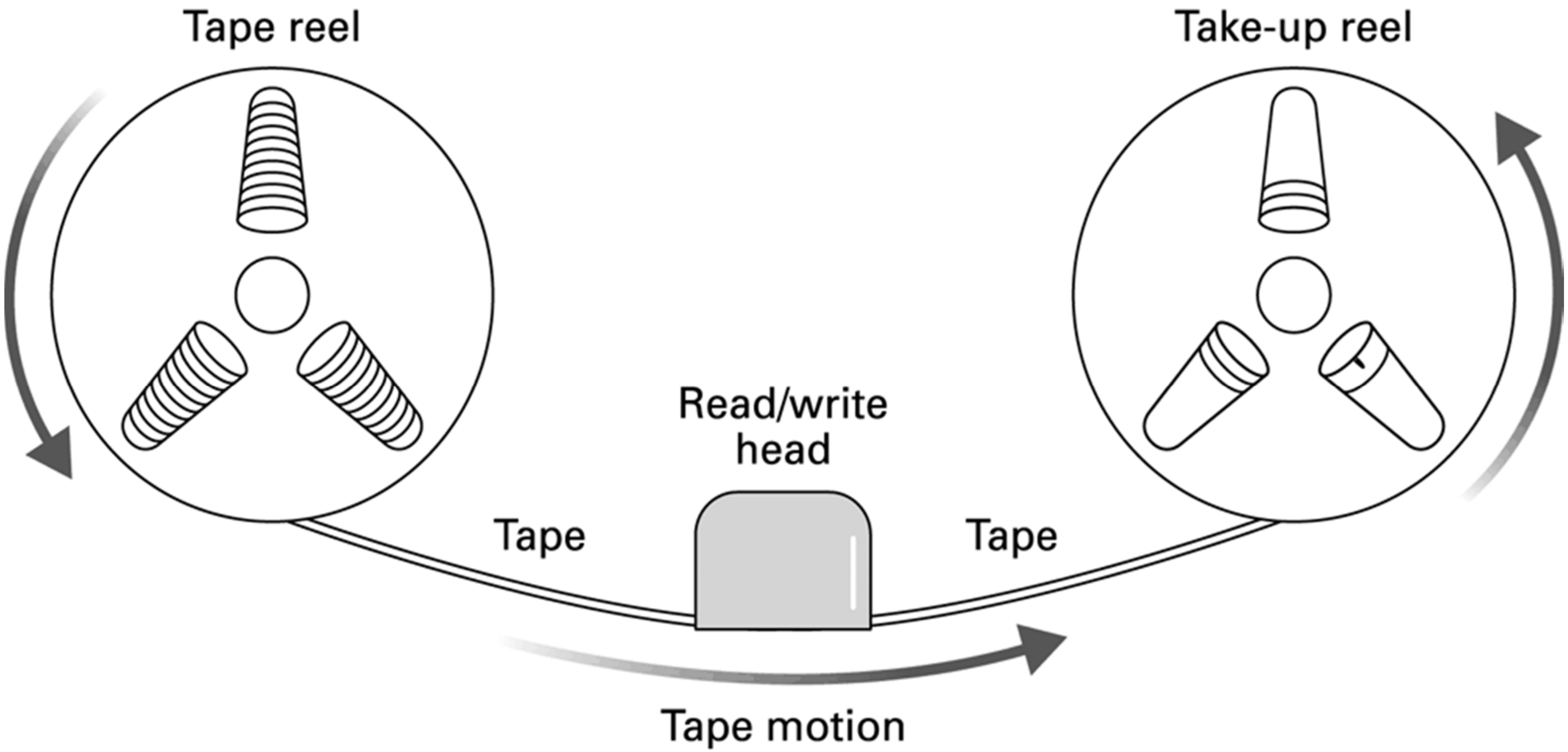
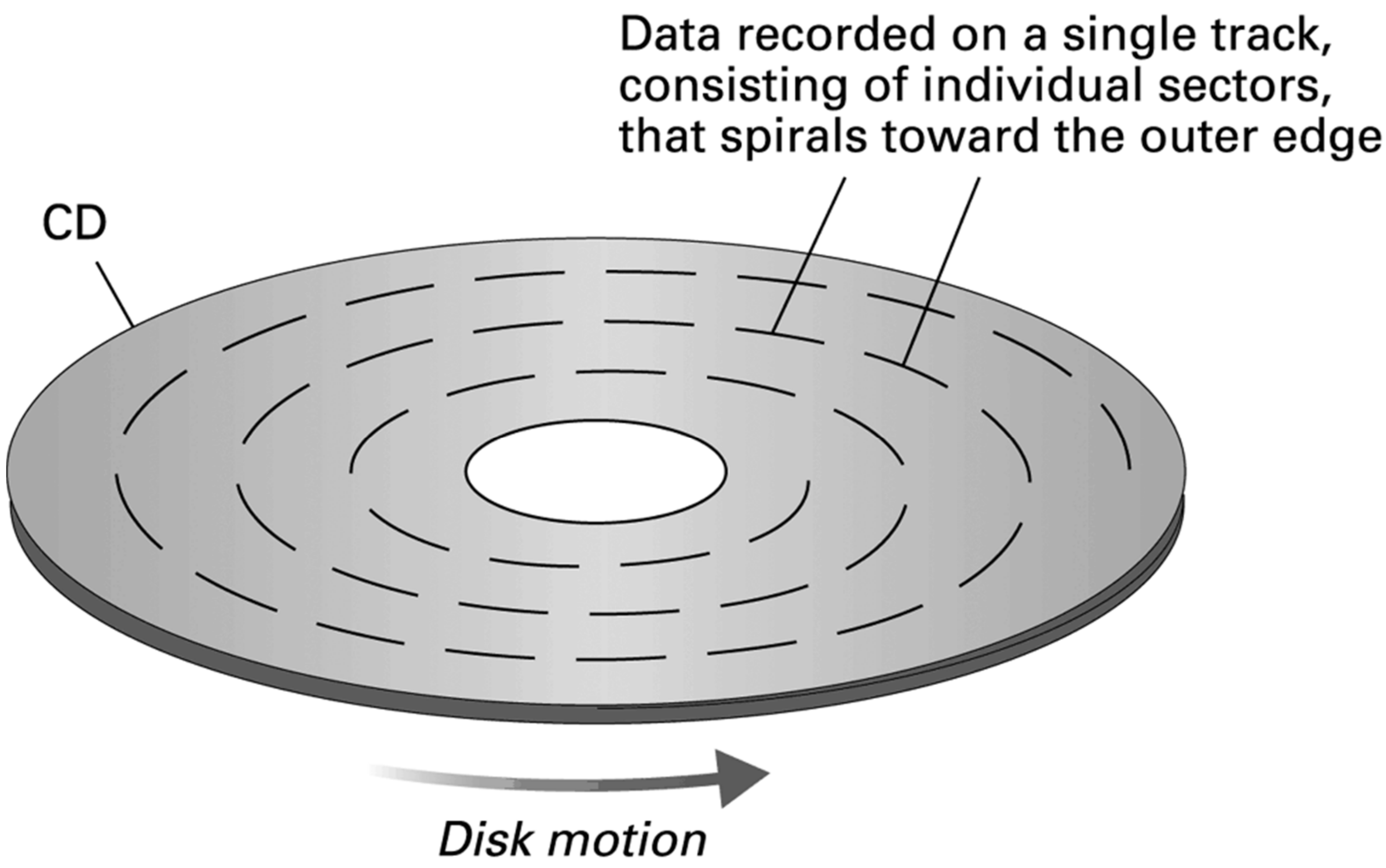


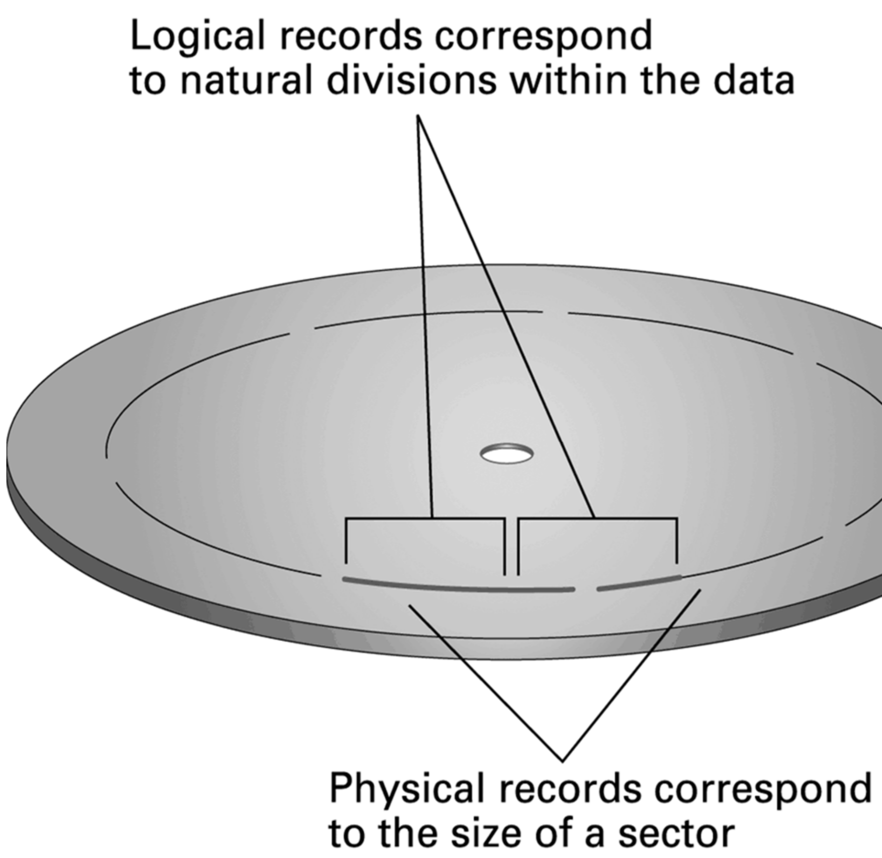
Figure 1.11 CD storage



Files

- **File:** A unit of data stored in mass storage system
  - **Fields** and **keyfields**
- Physical record versus Logical record
- **Buffer**（缓冲区）**:** A memory area used for the temporary storage of data (usually as a step in transferring the data)

Figure 1.12 **Logical records versus physical records on a disk**



Representing Information as Bit Patterns pp.35

Representing Text

- **Each character (letter, punctuation, etc.) is assigned a unique bit pattern.**
  - ASCII: Uses patterns of 7-bits to represent most symbols used in written English text
  - Unicode: Uses patterns of 16-bits to represent the major symbols used in languages world side

西文字符：ACSII码

(American Standard Code for Information Interchange)

用7位二进制编码，最高位0

问题：为什么用7位？

0~127共可表示128个字符

- ‘A’～ ‘Z’ 26
- ‘a’～’ z’ 26
- ‘0’～’ 9’ 10
- 其他键盘字符、控制键

$\Big\}$ <=128

0～32、127为非图形字符，其余94个图形字符

Printable ASCII Codes

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
2	space	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL

- Examples:
  - ASCII code for space character = 20 (hex) = 32 (decimal)
  - ASCII code for 'L' = 4C (hex) = 76 (decimal)
  - ASCII code for 'a' = 61 (hex) = 97 (decimal)

Figure 1.13 **The message "Hello." in ASCII**

01001000	01100101	01101100	01101100	01101111	00101110
H	**e**	**l**	**l**	**o**	**.**

Exercise

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
2	space	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL

- 01000011 01001000 01001001 01001110 01000001