

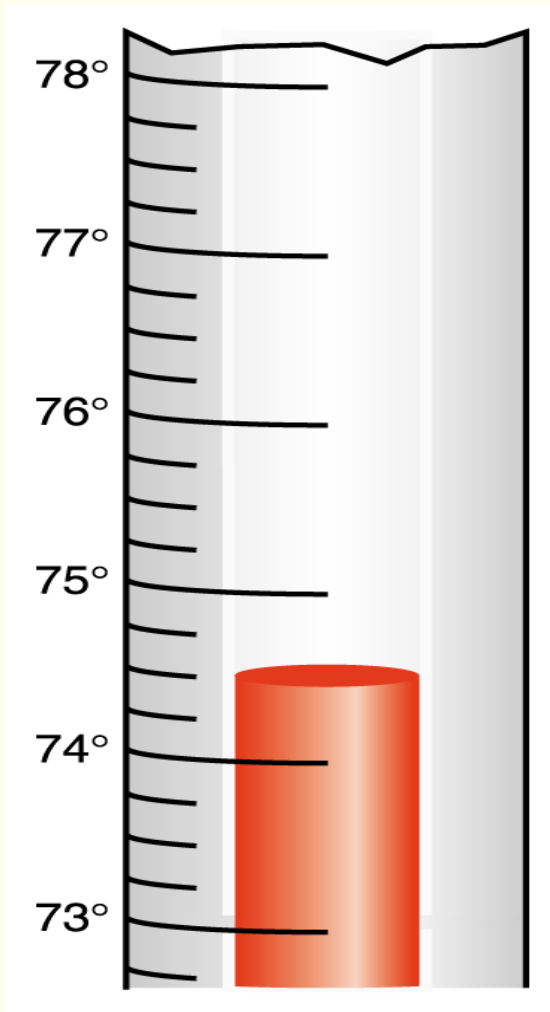
# ***Chapter 1***

## **Digital Systems and Binary Numbers**

# Analog and Digital Information (1/6)

- Information (資訊) can be represented in one of two ways: **analog** or **digital** (類比或數位).
  - **Analog data** A continuous (連續性) representation, analogous to the actual information it represents.
  - **Digital data** A discrete (離散) representation, breaking the information up into separate elements.
- A mercury thermometer is an analog device. The mercury rises in a continuous flow in the tube in direct proportion to the temperature.

# Analog and Digital Information (2/6)



溫度計

← Analog device(類比裝置)

74.568.. degree Fahrenheit

A mercury thermometer continually rises in direct proportion to the temperature

耳溫槍

Digitize(數字化): The act of breaking information down into discrete pieces

Digital device: (數位裝置)

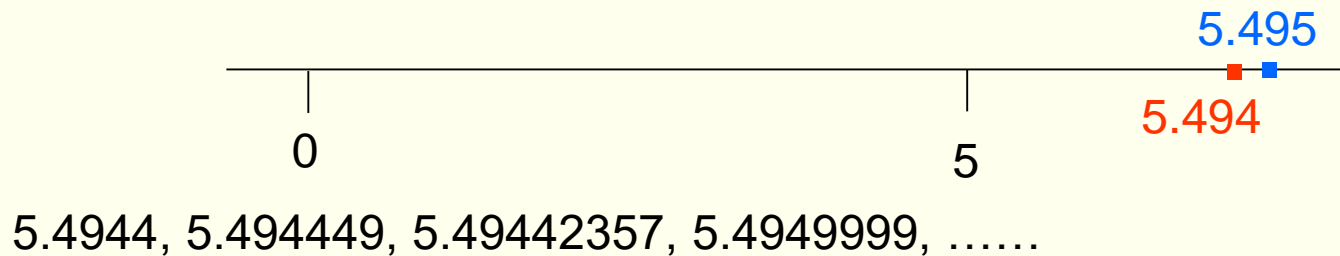
If the resolution is 0.01 degree, the answer is 74.56 or 74.57

If the resolution is 0.1 degree, the answer is 74.5 or 74.6

# Analog and Digital Information (3/6)

- Natural world is continuous and infinite

There are infinite real numbers between 5.494 and 5.495



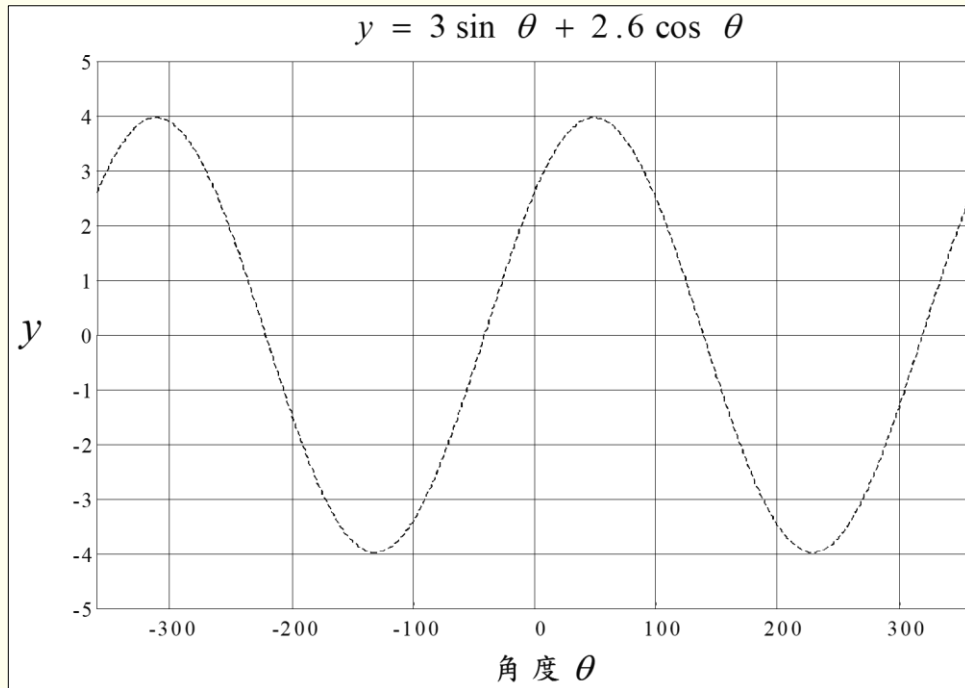
- Hardware (digital system) is finite (有限的).

Computer memory and other hardware devices have only so much room to store and manipulate a certain amount of data. The goal, is to represent enough of the world to satisfy our computational needs and our senses of sight and sound.

# Analog and Digital Information (4/6)

**Analog**

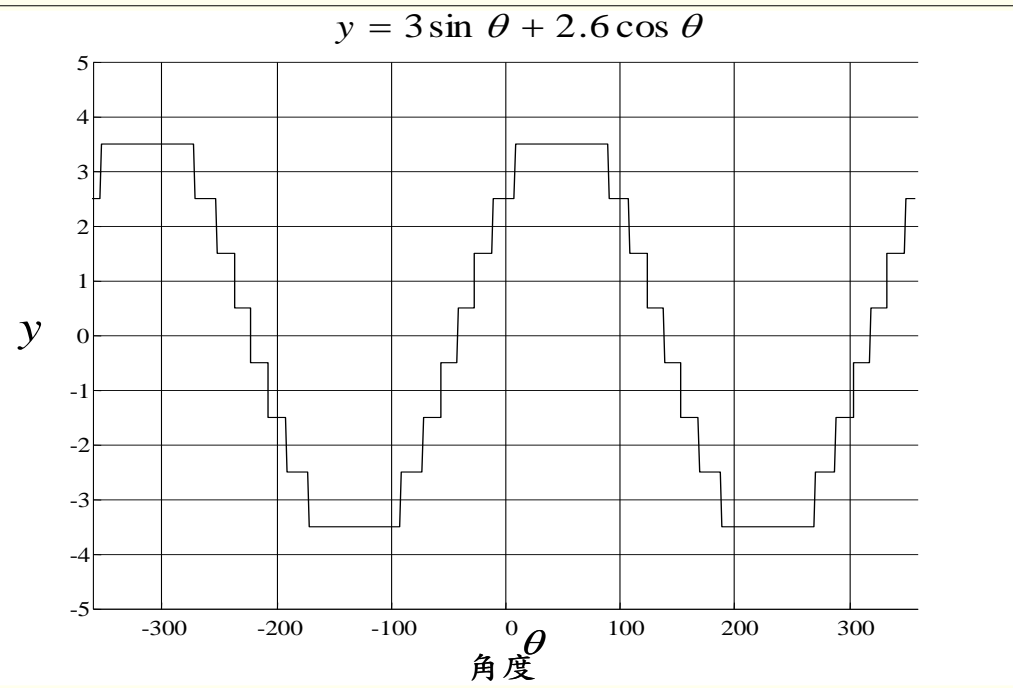
$$y = 3 \sin \theta + 2.6 \cos \theta$$



**Digital**

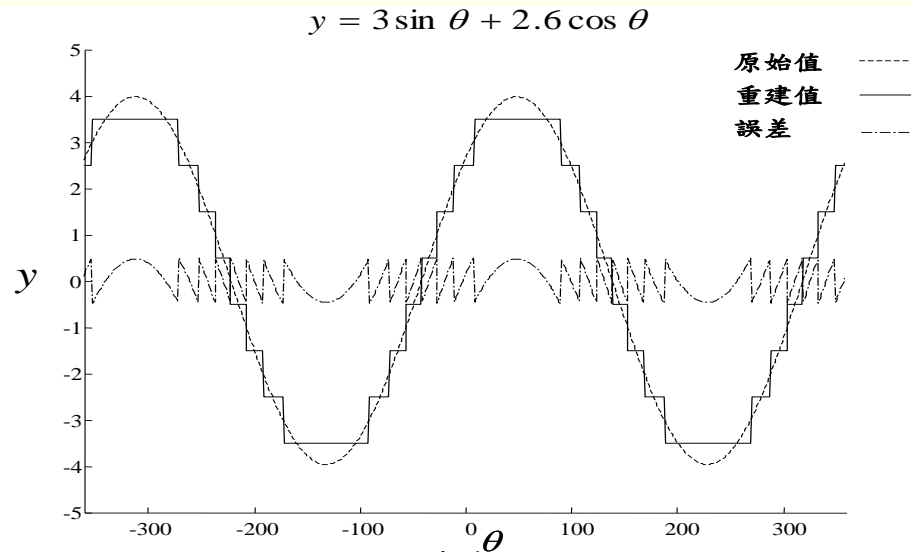
$-0.5 < \text{error} < 0.5$

$$y = 3 \sin \theta + 2.6 \cos \theta$$



**Digitize  $y$  with 8 discrete steps.**  
**-3.5, -2.5, -1.5, -0.5, 3.5, 2.5, 1.5, 0.5**

# Analog and Digital Information (5/6)



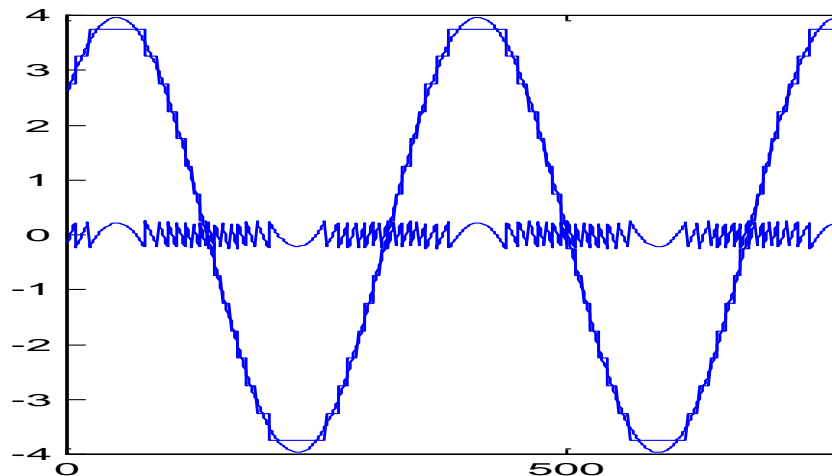
Digitize  $y$  with 8 discrete steps.

-3.5, -2.5, -1.5, -0.5, 3.5, 2.5, 1.5, 0.5

$-0.5 < \text{error} < 0.5$

3.99 or 3.01  $\rightarrow$  3.5

2.99  $\rightarrow$  2.5



Digitize  $y$  with 16 discrete steps.

-3.75, -3.25, -2.75, -2.25, -1.75, -1.25,

-0.75, -0.25, 0.25, 0.75, 1.25, 1.75

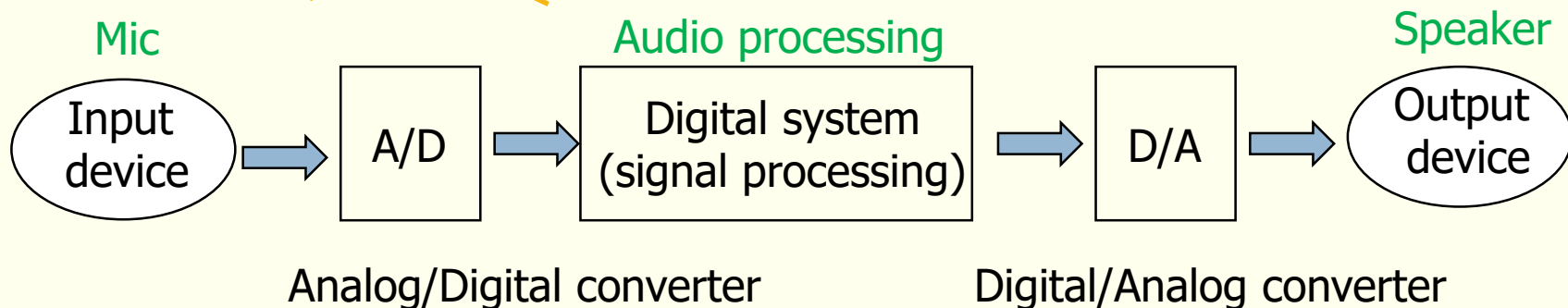
2.25, 2.75, 3.25, 3.75

$-0.25 < \text{error} < 0.25$

3.99 or 3.5  $\rightarrow$  3.75

# Analog and Digital Information (6/6)

- Most sensors are analog devices.
- Computers, cannot work well with analog information. So we **digitize** information by breaking it into pieces and representing those pieces separately.
- Therefore an analog to digital converter is needed.
- Signal processing can be performed on digital information only (analog information).



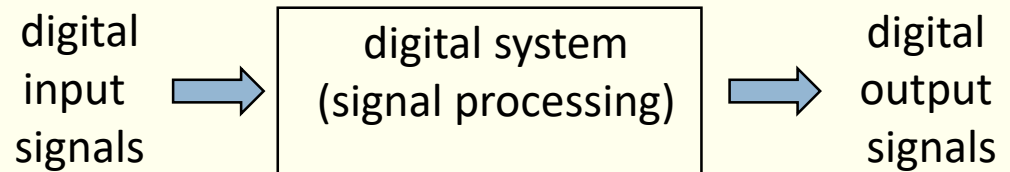
**Traditional TV input signals ??? CRT or LCD ???**

# Digital Systems

- Present technology period -- **Digital Age**

- Digital systems

- Digital phone
- Digital television
- Digital camera
- Electronic calculators, PDA's



- Digital computers

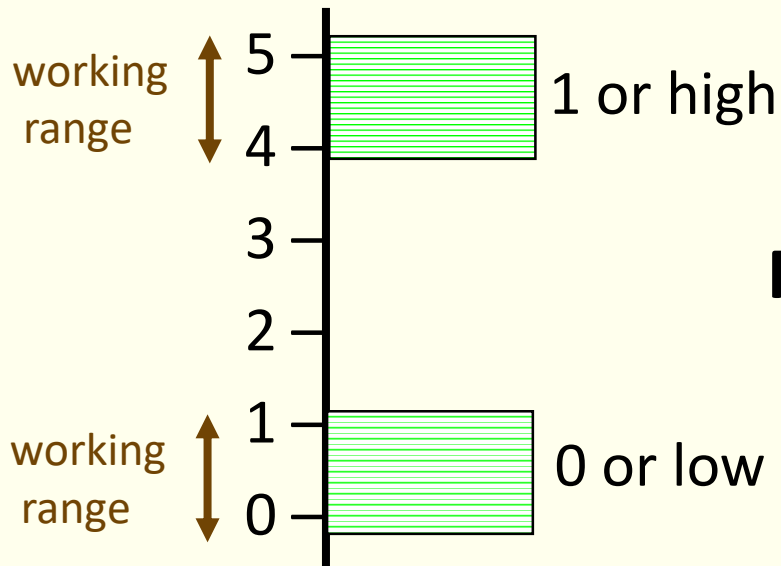
- Follow a sequence of instructions (--program, bitstream)
- General-purpose for information processing



# Signal

- Discrete elements of information are represented by physical quantity.
- Electrical signals such as voltages and currents are the most common.
- Most digital systems use two discrete values.  
(binary). It is easy to realized with the current or voltage.
  - digits 0 and 1
  - False (F) and True (T)
  - Low (L) and High (H)
  - On and Off

## Example: Voltage



**Fault tolerance is allowable.**

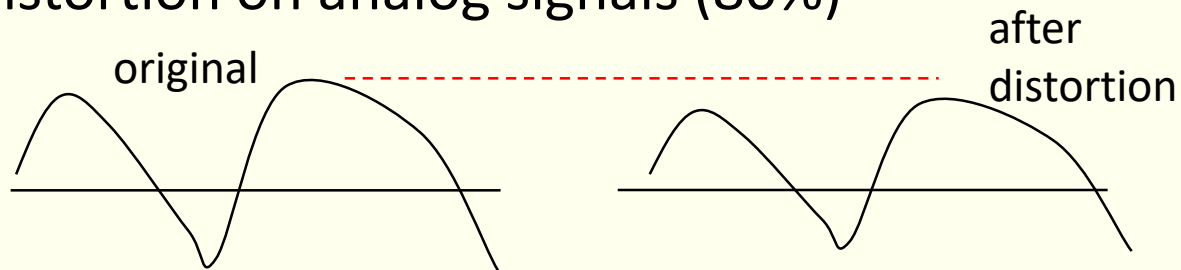
**Why binary ? Almost all digital systems use binary.**

**Distinguishing 2 levels is easier and less distortion than 10 levels for real-world electrical devices**

# Why Digital Signal ? (1/2)

Distortion happens in transmission.

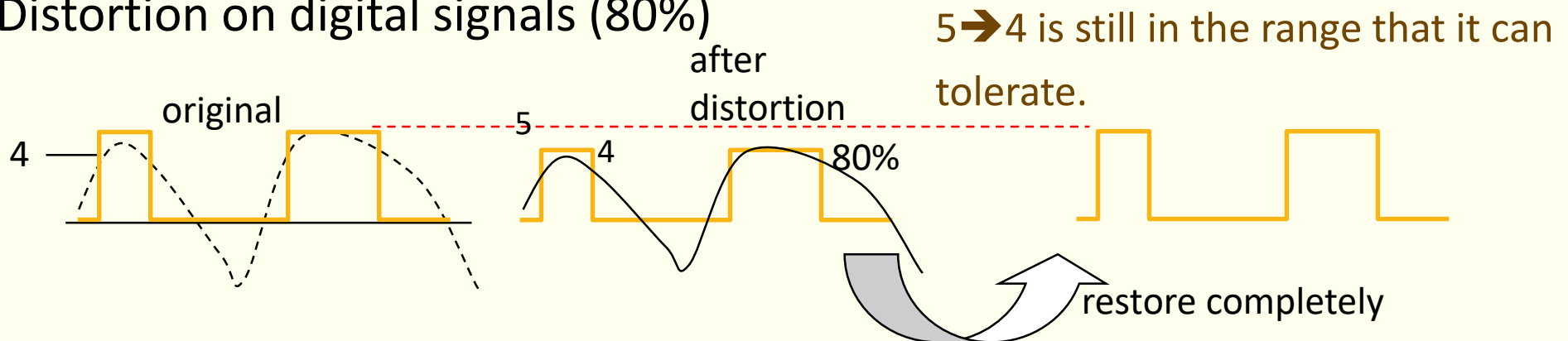
Distortion on analog signals (80%)



It is the reason  
that digital signal  
obtains less distortion  
than analog signal.

It is impossible to restore the original  
analog signal after distortion.

Distortion on digital signals (80%)



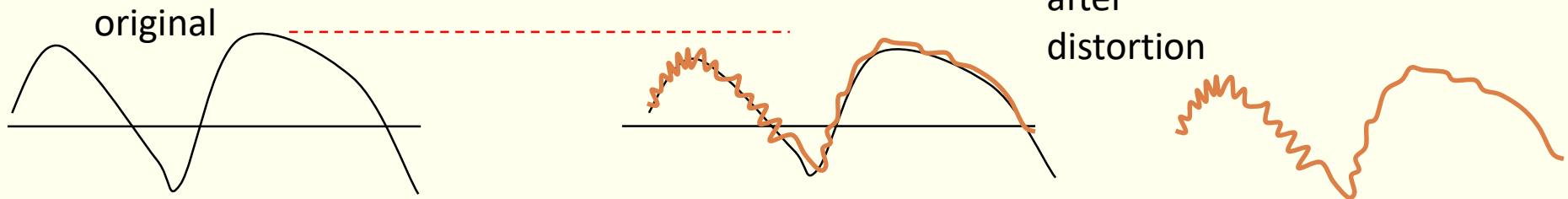
5 → 4 is still in the range that it can  
tolerate.

It is possible to restore the original digital signal after distortion.

# Why Digital Signal ? (2/2)

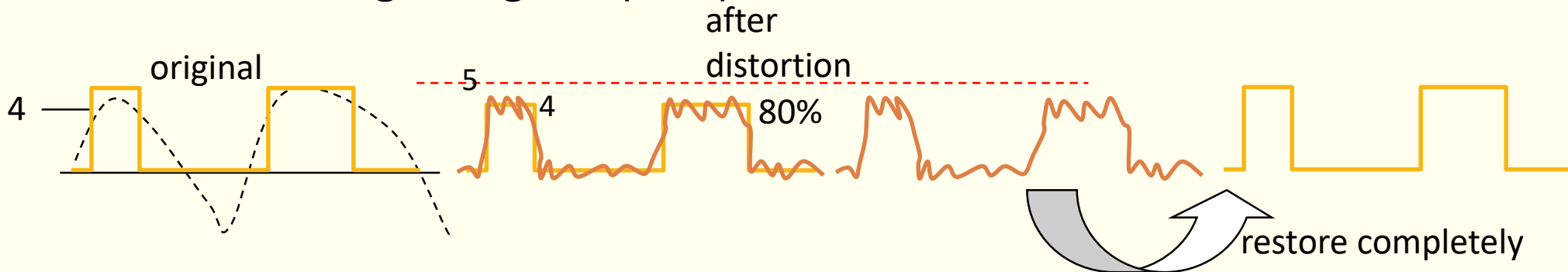
**Distortion happens in transmission.**

**Distortion on analog signals (80%)**



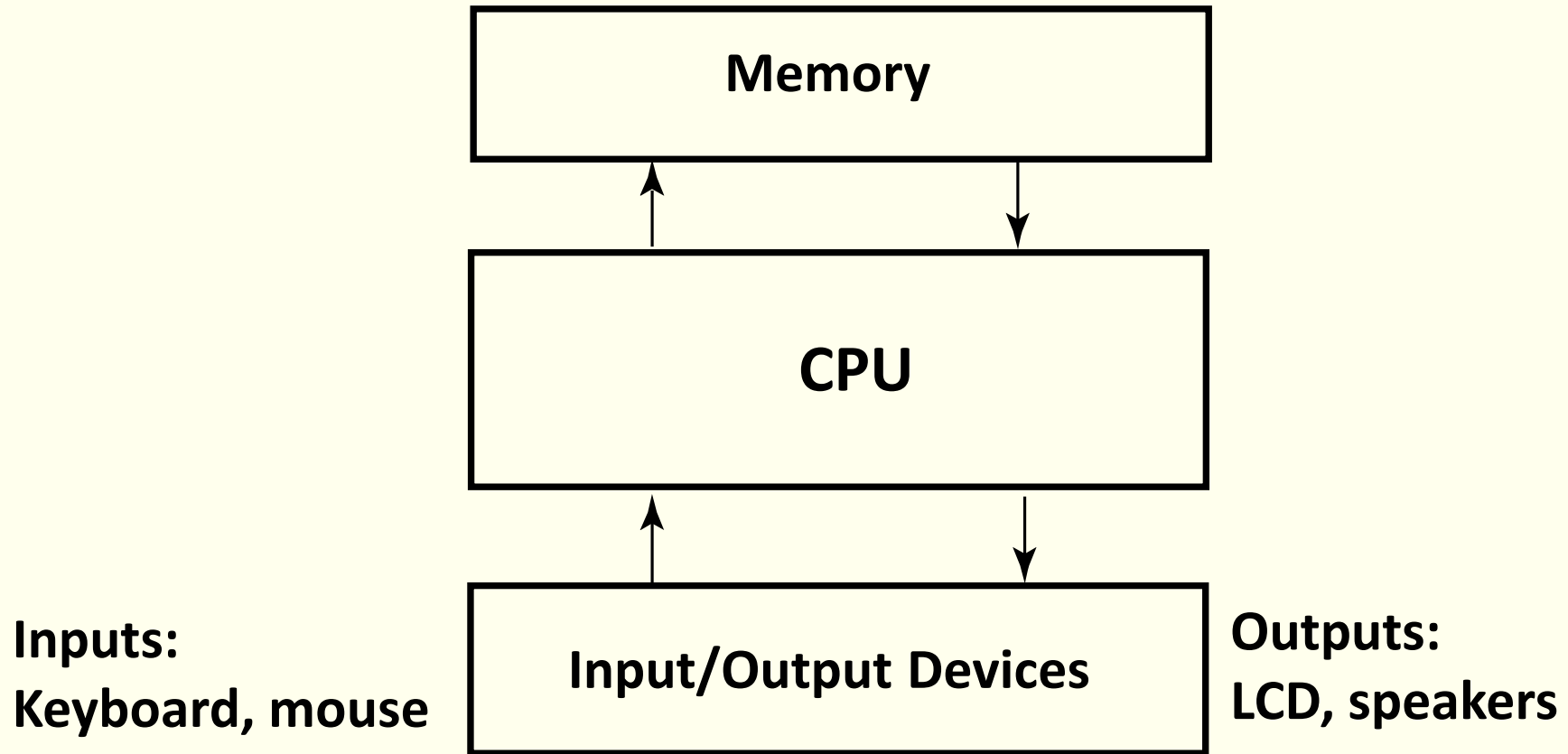
**It is impossible to restore the original analog signal after distortion.**

**Distortion on digital signals (80%)**



**It is possible to restore the original digital signal after distortion.**

# A Digital Computer



Since a digital system is usually programmable, so it is very suitable for commercial products. ➔ **Semiconductor Technology**  
**Transistors ➔ Gate ➔ Digital circuit ➔ Printed circuit board (PCB)**

# Positional Notation

**7392 in base 10 *positional notation* is:**

$$\begin{aligned} &7 \times 10^3 = 7 \times 1000 = 7000 \\ &+ 3 \times 10^2 = 3 \times 100 = 300 \\ &+ 9 \times 10^1 = 9 \times 10 = 90 \\ &+ 2 \times 10^0 = 2 \times 1 = 2 \quad = 7392 \text{ in base 10} \end{aligned}$$

This number is in  
base 10

The power indicates  
the position of  
the number

## Positional Notation in Base 10

- A number with *radix 10* (*base 10*) is represented by a string of digits:

$$a_{n-1}a_{n-2} \dots a_1a_0 \cdot a_{-1}a_{-2} \dots a_{-m+1}a_{-m}$$

where  $a_i \in \{0, 1, 2, \dots, 9\}$  and “.” is the *radix point*.

$$\begin{aligned} & a_{n-1} \times 10^{n-1} + a_{n-2} \times 10^{n-2} + \dots + a_1 \times 10^1 + a_0 \times 10^0 \\ & + a_{-1} \times 10^{-1} + a_{-2} \times 10^{-2} + \dots + a_{-m} \times 10^{-m} \end{aligned}$$

$$(378.25)_{10} = 3 \times 10^2 + 7 \times 10^1 + 8 \times 10^0 + 2 \times 10^{-1} + 5 \times 10^{-2}$$

# Positional Notation in Base 10

$$642.15 = 6 * 10^2 + 4 * 10^1 + 2 * 10^0 \\ + 1 * 10^{-1} + 5 * 10^{-2}$$

$$10^{-1} = 1/10 = 0.1 \quad 10^{-2} = 1/100 = 0.01$$

$$3899.287 = 3 * 10^3 + 8 * 10^2 + 9 * 10^1 + 9 * 10^0 \\ + 2 * 10^{-1} + 8 * 10^{-2} + 7 * 10^{-3}$$



## Positional Notation in Base 8

- A number with *radix 8 (base 8)* is represented by a string of digits:

$$a_{n-1}a_{n-2} \dots a_1a_0 \cdot a_{-1}a_{-2} \dots a_{-m+1}a_{-m}$$

where  $a_i \in \{0, 1, \dots, 7\}$  and “.” is the *radix point*.

$$a_{n-1} \times 8^{n-1} + a_{n-2} \times 8^{n-2} + \dots + a_1 \times 8^1 + a_0 \times 8^0 \\ + a_{-1} \times 8^{-1} + a_{-2} \times 8^{-2} + \dots + a_{-m} \times 8^{-m}$$

$$(157.4)_8 = 1 \times 8^2 + 5 \times 8^1 + 7 \times 8^0 + 4 \times 8^{-1} = (111.5)_{10}$$

## Positional Notation in Base 2

- A number with *radix 2 (base 2)* is represented by a string of digits:  $a_i \in \{0, 1\}$

$$a_{n-1}a_{n-2} \dots a_1a_0 \cdot a_{-1}a_{-2} \dots a_{-m+1}a_{-m}$$

$$a_{n-1} \times 2^{n-1} + a_{n-2} \times 2^{n-2} + \dots + a_1 \times 2^1 + a_0 \times 2^0 \\ + a_{-1} \times 2^{-1} + a_{-2} \times 2^{-2} + \dots + a_{-m} \times 2^{-m}$$

$$(10110.101)_2 = 1 \times 2^4 + 1 \times 2^2 + 1 \times 2^1 + 1 \times 2^{-1} + 1 \times 2^{-3} \\ = (16 + 4 + 2 + 0.5 + 0.125) = (22.625)_{10}$$

↓  
1 bit

## Positional Notation in Base $r$

- A number with *radix  $r$*  (*base  $r$* ) is represented by a string of digits:

$$a_{n-1}a_{n-2} \cdots a_1a_0 \cdot a_{-1}a_{-2} \cdots a_{-m+1}a_{-m}$$

where  $a_i \in \{0, 1, \dots, r-1\}$  and “.” is the *radix point*.

$$a_{n-1} \times r^{n-1} + a_{n-2} \times r^{n-2} + \dots + a_1 \times r^1 + a_0 \times r^0 \\ + a_{-1} \times r^{-1} + a_{-2} \times r^{-2} + \dots + a_{-m} \times r^{-m}$$

$$(324.12)_r = 3 \times r^2 + 2 \times r^1 + 4 \times r^0 + 1 \times r^{-1} + 2 \times r^{-2} \text{ where } r > 4$$

# Positional Notation in Base 13

*What if 642 has the base of 13?*

$$\begin{aligned} 6 \times 13^2 &= 6 \times 169 = 1014 \\ + 4 \times 13^1 &= 4 \times 13 = 52 \\ + 2 \times 13^0 &= 2 \times 1 = 2 \\ &= 1068 \text{ in base 10} \end{aligned}$$

642 in base 13 is equivalent to 1068 in base 10 →

How to distinguish them?

$$(642)_{13} = 1068 = (1068)_{10}$$

$$(642)_{10} = 642$$

# Number Systems

Base 10: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 (10 digits)

Carry happens at 10      Decimal

Base 8: 0, 1, 2, 3, 4, 5, 6, 7 (8 digits)

Carry happens at 8      Octal

Base 16: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D,

E, F (16 digits)      Hexadecimal

Carry happens at 16

## **Bases Higher than 10**

*How are digits in bases higher than 10 represented?*

With distinct symbols for 10 and above.

Base 16 has 16 digits:

**0,1,2,3,4,5,6,7,8,9,A,B,C,D,E, and F**

For a number to exist in a given number system, the number system must include those digits. For example, the number 284 only exists in base 9 and higher.

**Why octal and hexadecimal numbers are used?**

**To make the length of a binary number shorter.**

## Special Powers of 2

A “0” or “1” is called a “bit.”

A 8-bit number (“00101100”) is called a “byte.”

In real measure world,  $K=10^3$ ,  $M=10^6$ , and  $G=10^9$ .

In computer work,

- $2^{10}$  (1024) is Kilo, denoted "K"
- $2^{20}$  (1,048,576) is Mega, denoted "M"
- $2^{30}$  (1,073, 741,824) is Giga, denoted "G"
- $2^{40}$  is Tera, denoted “T”

**4 M Bytes =  $4 \times 2^{20} = 4 \times 1048576$  bytes =  $4 \times 1048576 \times 8$  Bits**

# Arithmetic in Binary

Remember that there are only 2 digits in binary,  
0 and 1

$$\begin{array}{r} 1011111 \\ 1010111 \\ + 1001011 \\ \hline 10100010 \end{array}$$


Carry Values



# Subtracting Binary Numbers

*Remember borrowing? Apply that concept here:*

$$\begin{array}{r} \text{1 2} \\ \text{2 0 2} \\ 1\ 0\ 1\ 0\ 1\ 1\ 1 \\ -\ 1\ 1\ 1\ 0\ 1\ 1 \\ \hline 0\ 0\ 1\ 1\ 1\ 0\ 0 \end{array}$$

$$1011 \quad (11)$$

$$\times 101 \quad (5)$$

$$\hline 1011$$

$$0000$$

$$1011$$

$$\hline 110111 \quad (55)$$

# Converting Octal to Decimal

*What is the decimal equivalent of the octal number 642?*

$$\begin{aligned} 6 \times 8^2 &= 6 \times 64 = 384 \\ + 4 \times 8^1 &= 4 \times 8 = 32 \\ + 2 \times 8^0 &= 2 \times 1 = 2 \\ &= 418 \text{ in base 10} \end{aligned}$$

$$(642)_8 = 6 \times 8^2 + 4 \times 8^1 + 2 \times 8^0 = (418)_{10} = 418$$

$$\begin{array}{ccc} 8^2 & 8^0 & 8^{-2} \\ (257.16)_8 & & \\ 8^1 & 8^{-1} & \end{array}$$

# Converting Hexadecimal to Decimal

*What is the decimal equivalent of the hexadecimal number DEF?*

$$\begin{aligned} D \times 16^2 &= 13 \times 256 = 3328 \\ + E \times 16^1 &= 14 \times 16 = 224 \\ + F \times 16^0 &= 15 \times 1 = 15 \\ &= 3567 \text{ in base 10} \end{aligned}$$

Remember, the digits in base 16 are  
0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F

$$(DEF)_{16} = 13 \times 16^2 + 14 \times 16^1 + 15 \times 16^0 = (3567)_{10} = 3567$$

$$\begin{array}{ccc} 16^2 & 16^0 & 16^{-2} \\ (A & 5 & 7 . 1 & 9)_{16} \\ 16^1 & 16^{-1} & \end{array}$$

# Converting Binary to Decimal

*What is the decimal equivalent of the binary number 1101110?*

$$\begin{aligned}1 \times 2^6 &= 1 \times 64 = 64 \\+ 1 \times 2^5 &= 1 \times 32 = 32 \\+ 0 \times 2^4 &= 0 \times 16 = 0 \\+ 1 \times 2^3 &= 1 \times 8 = 8 \\+ 1 \times 2^2 &= 1 \times 4 = 4 \\+ 1 \times 2^1 &= 1 \times 2 = 2 \\+ 0 \times 2^0 &= 0 \times 1 = 0 \\&= 110 \text{ in base 10}\end{aligned}$$

$$(1101110)_2 = 110$$

$$\begin{array}{cccc}2^2 & 2^0 & 2^{-2} & \\(1101.101)_2 & & & \\2^3 & 2^1 & 2^{-1} & 2^{-3}\end{array}$$

# Power of 2 Number System

<b>Binary</b>	<b>Octal</b>	<b>Decimal</b>	<b>Hexadecimal</b>
<b>000</b>	<b>0</b>	<b>0</b>	<b>0</b>
<b>001</b>	<b>1</b>	<b>1</b>	<b>1</b>
<b>010</b>	<b>2</b>	<b>2</b>	<b>2</b>
<b>011</b>	<b>3</b>	<b>3</b>	<b>3</b>
<b>100</b>	<b>4</b>	<b>4</b>	<b>4</b>
<b>101</b>	<b>5</b>	<b>5</b>	<b>5</b>
<b>110</b>	<b>6</b>	<b>6</b>	<b>6</b>
<b>111</b>	<b>7</b>	<b>7</b>	<b>7</b>
<b>1000</b>	<b>10</b>	<b>8</b>	<b>8</b>
<b>1001</b>	<b>11</b>	<b>9</b>	<b>9</b>
<b>1010</b>	<b>12</b>	<b>10</b>	<b>A</b>
<b>1011</b>	<b>13</b>	<b>11</b>	<b>B</b>
<b>1100</b>	<b>14</b>	<b>12</b>	<b>C</b>

# Converting Binary to Octal

- Groups of Three (from right)
- Convert each group

10101011

<u>10</u>	<u>101</u>	<u>011</u>
2	5	3

10101011 is 253 in base 8

i.e.,  $(10101011)_2 = (253)_8$

How about converting octal to binary ?

# Converting Binary to Hexadecimal

- Groups of Four (from right)
- Convert each group

10101011

1010 1011  
A B

10101011 is AB in base 16

i.e.,  $(10101011)_2 = (AB)_{16}$

How about converting hexadecimal to binary ?

# Converting Decimal to Other Bases

**Algorithm** for converting base 10 to other bases

While the quotient is ***not*** zero

Divide the decimal number by the new base

Make the remainder the next digit to the left in the answer

Replace the original dividend with the quotient



# Converting Decimal to Hexadecimal

Try a Conversion

*The base 10 number 3567 is what number in base 16?*

$$\begin{array}{r} 222 \\ 16 \overline{) 3567} \\ \underline{32} \\ 36 \\ \underline{32} \\ 47 \\ \underline{32} \\ 15 \end{array} \quad \begin{array}{r} 13 \\ 16 \overline{) 222} \\ \underline{16} \\ 62 \\ \underline{48} \\ 14 \end{array} \quad \begin{array}{r} 0 \\ 16 \overline{) 13} \\ \underline{0} \\ 13 \end{array}$$

D E F

# Converting Decimal to Hexadecimal

$$\begin{array}{rcl} 16 & \overline{) 3567} & \\ 16 & \overline{) 222} \dots 15 \text{ (the least significant digit)} & \mathbf{F} \\ 16 & \overline{) 13} \dots 14 & \mathbf{E} \\ & 0 \dots 13 \text{ (the most significant digit)} & \mathbf{D} \end{array}$$

$$3567 = (\mathbf{DEF})_{16}$$

$$\begin{array}{rcl} 8 & \overline{) 3567} & \\ 8 & \overline{) 445} \dots 7 \text{ (the least significant digit)} & \mathbf{7} \\ 8 & \overline{) 55} \dots 5 & \mathbf{5} \\ 8 & \overline{) 6} \dots 7 & \mathbf{7} \\ & 0 \dots 6 \text{ (the most significant digit)} & \mathbf{6} \end{array}$$

$$3567 = (\mathbf{6757})_8$$

# Converting Decimal to Binary

**MSB**  
↓  
 $10 = (1010)_2$   
↑  
**LSB**

2	>	10		
2	>	5 ... 0	(the least significant bit)	0 <b>LSB</b>
2	>	2 ... 1		1
2	>	1 ... 0		0
		0 ... 1	(the most significant bit)	1 <b>MSB</b>

$$10.125 = (1010.001)_2$$

$$0.125 \times 2 = \underline{0}.25$$

$$0.25 \times 2 = \underline{0}.5$$

$$0.5 \times 2 = \underline{1}.0 \quad \text{end here since all digits after point are zero}$$

# Conclusion for Number Transformation

$$10.125 = (1010.001)_2$$

$$(1010.001)_2 =$$

$$(1417.651)_8 =$$

Converting B/O/H to Decimal number

1. Divide the whole part of the decimal number by the new base
2. Multiply the fractional part of decimal number by the new base

$$10.125 = (1010.001)_2$$

$$153.513 = (231.406517..)_8$$

$$\begin{array}{r} 8 \overline{) 153} \\ 8 \overline{) 19 \dots 1} \\ 8 \overline{) 2 \dots 3} \\ \quad 0 \dots 2 \end{array}$$

$$0.513 * 8 = \underline{4}.104$$

$$0.104 * 8 = \underline{0}.832$$

$$0.832 * 8 = \underline{6}.656$$

$$0.656 * 8 = \underline{5}.248$$

$$0.248 * 8 = \underline{1}.984$$

$$0.984 * 8 = \underline{7}.872$$

Converting decimal to B/O/H number

# Binary and Computers

**Byte** 8 bits

The number of bits in a word determines the word length of the computer, but it is usually a multiple of 8

- 32-bit machines
- 64-bit machines etc.

# Binary Representations

- One bit can be either 0 or 1. Therefore, one bit can represent only two things.
- To represent more than two things, we need multiple bits. Two bits can represent four things because there are four combinations of 0 and 1 that can be made from two bits: 00, 01, 10, 11.
- If we want to represent more than four things, we need more than two bits. Three bits can represent eight things because there are eight combinations of 0 and 1 that can be made from three bits.

# Binary Representations

1 Bit	2 Bits	3 Bits	4 Bits	5 Bits
0	00	000	0000	00000
1	01	001	0001	00001
	10	010	0010	00010
	11	011	0011	00011
		100	0100	00100
		101	0101	00101
		110	0110	00110
		111	0111	00111
			1000	01000
			1001	01001
			1010	01010
			1011	01011
			1100	01100
			1101	01101
			1110	01110
			1111	01111
				10000
				10001
				10010
				10011
				10100
				10101
				10110
				10111
				11000
				11001
				11010
				11011
				11100
				11101
				11110
				11111

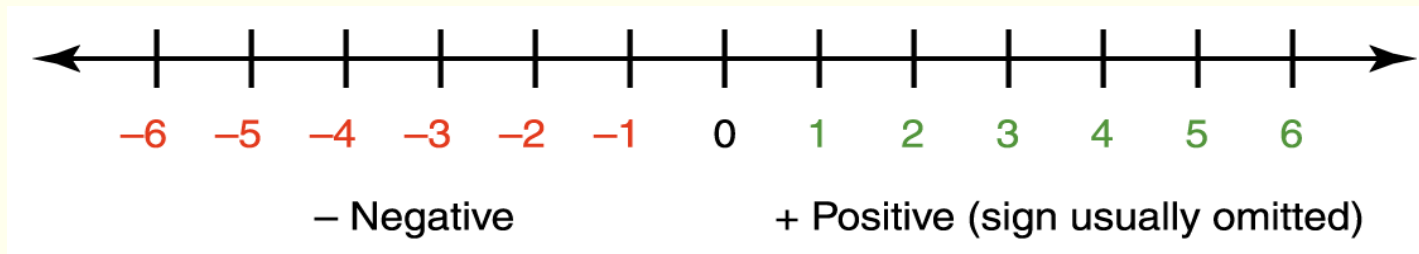
In general,  $n$  bits can represent  $2^n$  things because there are  $2^n$  combinations of 0 and 1 that can be made from  $n$  bits.

Note that every time we increase the number of bits by 1, we double the number of things we can represent.

## Representing Negative Values (1/2)

- We have used the **signed-magnitude representation** of numbers since grade school. The sign represents the ordering, and the digits represent the magnitude of the number (one bit for the sign and others for the magnitude).

+4, 6, -5, -3





## Representing Negative Values (2/2)

- There is a problem with the signed-magnitude representation.

Two representations of zero. Plus zero and minus zero.

Two representations of zero (+0 and -0) within a computer can cause unnecessary complexity.

For signed-magnitude representation:

$n$  bits can be used to represent  $-(2^{n-1}-1) \sim +(2^{n-1}-1)$

3 bits can be used to represent  $-(2^2-1) \sim +(2^2-1)$

signed-magnitude	
000	→ +0
001	→ +1
010	→ +2
011	→ +3
100	→ -0
101	→ -1
110	→ -2
111	→ -3

# Complements (1/2)

The  $r$ 's complement      **Base- $r$  system**      → **radix complement**

the  $r$ 's complement of  $N$  is defined as  $r^n - N$

$n$ : the digit width of  $N$

The  $(r-1)$ 's complement      → **diminished radix complement**

the  $(r-1)$ 's complement of  $N$  is defined as  $(r^n - 1) - N$

---

The 10's complement      **Base-10 system**

the 10's complement of  $N$  is defined as  $10^n - N$

The 9's complement       $n$ : the digit width of  $N$

the 9's complement of  $N$  is defined as  $(10^n - 1) - N$

The 10's complement of 546700 is  $1000000 - 546700 = 453300$  ( $n=6$ )

The 9's complement of 546700 is  $999999 - 546700 = 453299$

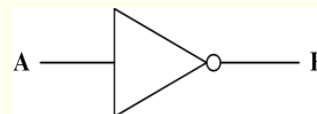
## Complements (2/2)

Base-2 system (Binary system),

The 2's complement of 0101101 is 1010011 ( $2^7 - 0101101 = 1010011$ )  $y'+1$   
The 1's complement of 0101101 is 1010010 ( $(2^7 - 1) - 0101101 = 1010010$ )  $y'$

1's complement can be implemented with the digital circuit easily.

0  $\rightarrow$  1  
1  $\rightarrow$  0



1011000

↓ ↓ ↓

0100111

1's complement  $\rightarrow$  implemented easily with NOT gates

2's complement  $\rightarrow$  1's complement + 1

# Signed Binary Numbers

## 1. Signed-magnitude

- Sign bit + unsigned numbers

'0' positive 0111 = +7    '1' negative 1111 = -7

the most significant  
bit (MSB)

the least  
significant bit (LSB)

## 2. Signed-complement

- negative number ← taking complement
- Signed-1's complement

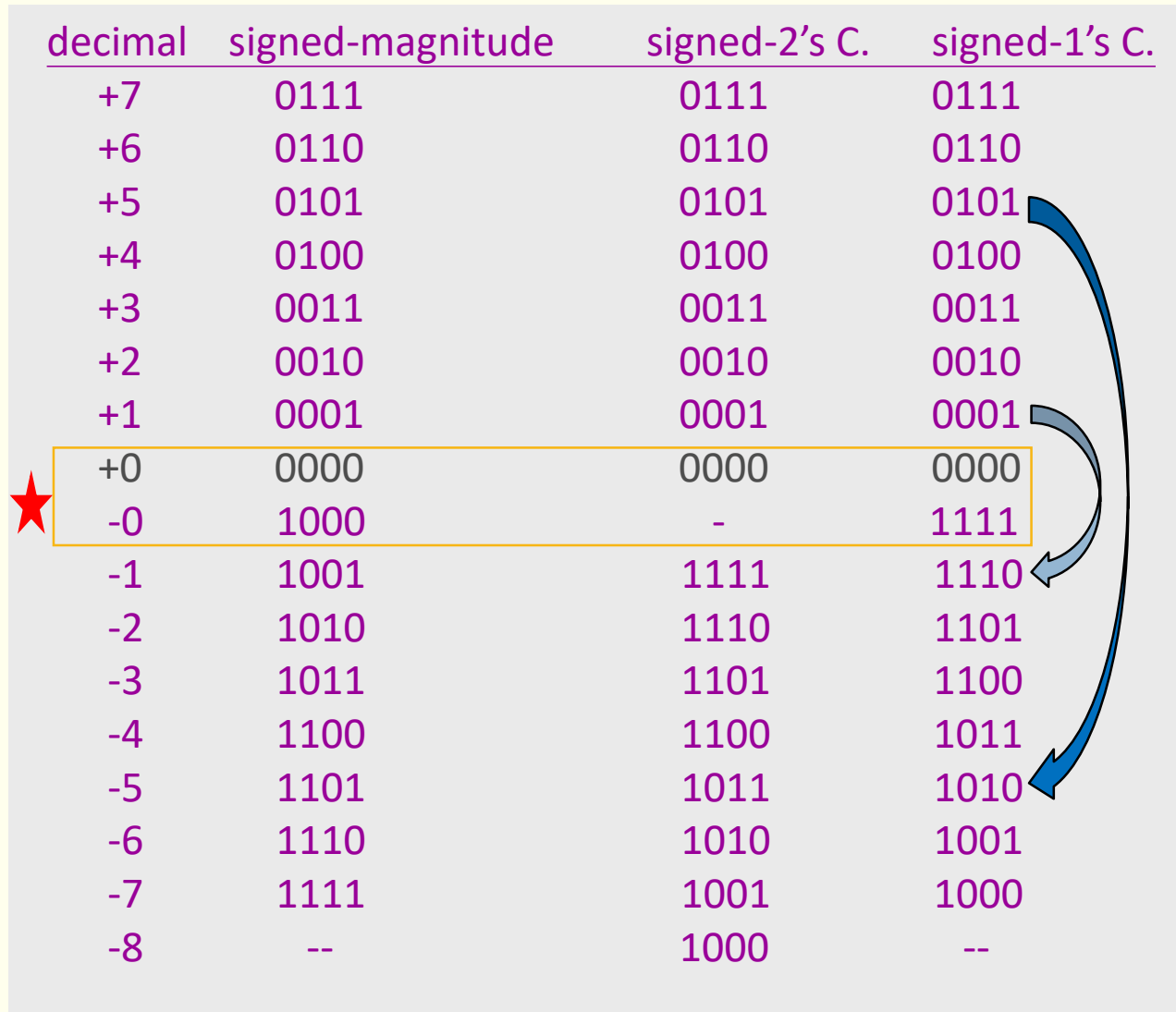
positive 0111 = +7    negative 1000 = -7

- Signed-2's complement

positive 0111 = +7    negative 1001 = -7

# Signed Binary Numbers

decimal	signed-magnitude	signed-2's C.	signed-1's C.
+7	0111	0111	0111
+6	0110	0110	0110
+5	0101	0101	0101
+4	0100	0100	0100
+3	0011	0011	0011
+2	0010	0010	0010
+1	0001	0001	0001
★ +0	0000	0000	0000
-0	1000	-	1111
-1	1001	1111	1110
-2	1010	1110	1101
-3	1011	1101	1100
-4	1100	1100	1011
-5	1101	1011	1010
-6	1110	1010	1001
-7	1111	1001	1000
-8	--	1000	--



Signed-2's complement  
is the only one which  
has the single  
representation of zero.

# Con of Signed-Magnitude Representation

## signed-magnitude representation (binary)

0	0	0	+0
0	0	1	+1
0	1	0	+2
0	1	1	+3
1	0	0	-0
1	0	1	-1
1	1	0	-2
1	1	1	-3

↑  
sign  
↑  
magnitude

$$2 + (-1) = ?$$

$$\begin{array}{r} 010 \\ + 101 \\ \hline 111 \end{array} \begin{array}{l} (-3) \\ \text{error} \end{array}$$

$$3 - 1 = ?$$

$$\begin{array}{r} 011 \\ - 001 \\ \hline 010 \end{array} \begin{array}{l} (2) \\ \text{correct} \end{array}$$

$$1 - 2 = ?$$

$$\begin{array}{r} 001 \\ - 010 \\ \hline 111 \end{array} \begin{array}{l} (-3) \\ \text{error} \end{array}$$

There are a lot problems for a digital circuit to execute + and – with signed magnitude representation (more complex, need 1 adder and 1 subtractor)

# Pro of One's Complement Representation

The 1's complement of 001 (+1) is 110 → -1

The 1's complement of 010 (+2) is 101 → -2

The 1's complement of 011 (+3) is 100 → -3

0 0 0	+0				
0 0 1	+1				
0 1 0	+2				
0 1 1	+3				
1 1 0	-1				
1 0 1	-2				
1 0 0	-3				
1 1 1	-0				

		2-1= 2+(-1)=?	3-1=3+(-1)=?	1-2=1+(-2)=?
		0 1 0	0 1 1	0 0 1
		+ 1 1 0	+ 1 1 0	+1 0 1
		<u>1 0 0 0</u>	<u>1 0 0 1</u>	<u>1 1 0</u>
		+1(=1)	+1(=2)	(-1)
		correct	correct	correct
		(discard carry and add 1)		

若有進位代表真正答案為正，捨棄“目前結果”之進位並在LSB加上1得到答案  
 若無進位代表真正答案為負，取“目前結果”之1補數加上負號得到答案

**Less complex (less cost, only 1 adder is needed)**

# Pro of Two's Complement Representation

The 2's complement of 001 (+1) is 111 → -1

The 2's complement of 010 (+2) is 110 → -2

The 2's complement of 011 (+3) is 101 → -3

$$2-1 = 2+(-1)=? \quad 3-1=3+(-1)=? \quad 1-2=1+(-2)=?$$

0 0 0 +0

0 0 1 +1

0 1 0 +2

0 1 1 +3

1 1 1 -1

1 1 0 -2

1 0 1 -3

1 0 0 -4

$$\begin{array}{r} 010 \\ +111 \\ \hline \end{array}$$

1 0 0 1 (1)  
correct

(discard carry)

$$\begin{array}{r} 011 \\ +111 \\ \hline \end{array}$$

1 0 1 0 (2)  
correct

$$\begin{array}{r} 001 \\ +110 \\ \hline \end{array}$$

1 1 1 (-1)  
correct

若有進位代表真正答案為正，捨棄“目前結果”之進位得到答案  
若無進位代表真正答案為負，取“目前結果”之2補數加上負號得到答案

**Less cost → only one 0 (without -0) + 1 adder is needed (without subtractor)**



# Subtraction

## 2補數之減法:

- (a)先將減數化為2補數
- (b)執行加法運算
- (c)所得結果若有進位，則去掉進位，其結果之值為正；  
若無進位，結果為負，須將計算結果再取2補數，  
才是真正之輸出負值。

## 1補數之減法:

- (a)先將減數化為1補數
- (b)執行加法運算
- (c)所得結果若有進位，其結果為正，將最小位元加1 (也可視為將進位加至最小位元)；若無進位，其結果為負，須將計算結果再取1補數，才是真正之輸出負值。

# Subtraction of Two's Complement Representation

$$X=0010100 \quad Y=0000011$$

$$X-Y=?$$

$$\begin{array}{r} X= \quad 0 \ 0 \ 1 \ 0 \ 1 \ 0 \ 0 \\ \text{2's complement of } Y= + \quad 1 \ 1 \ 1 \ 1 \ 1 \ 0 \ 1 \\ \hline \quad 1 \ 0 \ 0 \ 1 \ 0 \ 0 \ 0 \ 1 \end{array}$$

discard

**Answer: 0 0 1 0 0 0 1**

$$Y-X=?$$

$$\begin{array}{r} Y= \quad 0 \ 0 \ 0 \ 0 \ 0 \ 1 \ 1 \\ \text{2's complement of } X= + \quad 1 \ 1 \ 0 \ 1 \ 1 \ 0 \ 0 \\ \hline \quad 1 \ 1 \ 0 \ 1 \ 1 \ 1 \ 1 \end{array}$$

**Answer: - 0 0 1 0 0 0 1**  
2's complement

# Signed Binary Numbers

decimal	signed-magnitude	signed-2's C.	signed-1's C.
+7	0111	0111	0111
+6	0110	0110	0110
+5	0101	0101	0101
+4	0100	0100	0100
+3	0011	0011	0011
+2	0010	0010	0010
+1	0001	0001	0001
+0	0000	0000	0000
-0	1000	-	1111
-1	1001	1111	1110
-2	1010	1110	1101
-3	1011	1101	1100
-4	1100	1100	1011
-5	1101	1011	1010
-6	1110	1010	1001
-7	1111	1001	1000
-8	--	1000	--



Only 2's complement is OK.

Addition and subtraction is implemented with only one adder, so 2's complement is the most popular in digital circuits.

# Binary Codes

- **Binary Code**
  - Each discrete element of information is assigned a unique bit combination
- **n-bit**
  - $2^n$  distinct combination
  - 8 elements → a 3-bit code (min)
  - 16 → a 4-bit code (min)
- **Example**
  - Binary code for decimal digits (decimal code) requires at least 4 bits per digit.

# Decimal Code

## Binary codes for decimal digits (10)

- BCD (8421)  $2^3 2^2 2^1 2^0$
- 2421
- Excess-3
- 8, 4, -2, -1

**Table 1-5**  
*Four Different Binary Codes for the Decimal Digits*

<b>Decimal digit</b>	<b>BCD 8421</b>	<b>2421</b>	<b>Excess-3</b>	<b>8 4-2-1</b>
0	0000	0000	0011	0 0 0 0
1	0001	0001	0100	0 1 1 1
2	0010	0010	0101	0 1 1 0
3	0011	0011	0110	0 1 0 1
4	0100	0100	0111	0 1 0 0
5	0101	1011	1000	1 0 1 1
6	0110	1100	1001	1 0 1 0
7	0111	1101	1010	1 0 0 1
8	1000	1110	1011	1 0 0 0
9	1001	1111	1100	1 1 1 1
Unused bit combinations	1010	0101	0000	0 0 0 1
	1011	0110	0001	0 0 1 0
	1100	0111	0010	0 0 1 1
	1101	1000	1101	1 1 0 0
	1110	1001	1110	1 1 0 1
	1111	1010	1111	1 1 1 0

10      0001 0000      0001 0000      0100 0011      0111 0000

11      0001 0001      0001 0001      0100 0100      0111 0111

# Binary Coded Decimal (BCD)

- The BCD code is the 8,4,2,1 code.
- BCD is the simplest, most intuitive binary code for decimal digits and uses the same powers of 2 as a binary number, but only encodes the first ten values from 0 to 9.
- $(185)_{10} = \overset{7}{(0)}\overset{5}{1}\overset{4}{1}\overset{3}{1}\overset{0}{001})_2 = 128 + 32 + 16 + 8 + 1$
- $(185)_{10} = (\underline{0001} \ \underline{1000} \ \underline{0101})_{\text{BCD}}$
- Shortcoming
  - needs more bits than its equivalent binary number
- advantage
  - Most people use the decimal system

# Addition of BCD

If

$$1. \text{ sum} \geq \underline{1010} = (10)_{10}$$

$$2. \text{ sum} \geq \underline{10000} = (16)_{10}$$

then add  $(0110)_2 = 6$  to the binary sum

$\Rightarrow$  a carry + correct digit

Ex1.

$$\begin{array}{r}
 \begin{array}{cc} 4 & 0100 & 1 \\ + 8 & +1000 & \\ \hline 12 & 1100 & \geq 1010 \\ + & 0110 & +6 \\ \hline & \underline{1\ 0010} & \\ & \underline{1\ 2} & \end{array}
 \end{array}$$

Ex2.

$$\begin{array}{r}
 \begin{array}{cc} 8 & 1000 \\ + 9 & +1001 \\ \hline 17 & \textcolor{violet}{1\ 0001} \\ + & 0110 \\ \hline & \underline{\textcolor{violet}{1\ 0111}} & \\ & \underline{1\ 7} & \end{array}
 \end{array}$$



# Weighted Codes

- BCD (8421)  $(a_3a_2a_1a_0)_{\text{BCD}} = 8a_3 + 4a_2 + 2a_1 + 1a_0$
- 2421  $(a_3a_2a_1a_0)_{2421} = 2a_3 + 4a_2 + 2a_1 + 1a_0$
- 8,4,-2,-1  $(a_3a_2a_1a_0)_{84-2-1} = 8a_3 + 4a_2 - 2a_1 - 1a_0$
- Example
  - $6 = 0*8 + 1*4 + 1*2 + 0 = (0110)_{\text{BCD}}$
  - $6 = 1*2 + 1*4 + 0*2 + 0 = (1100)_{2421}$ 
    - not only one choice, e.g.,  $6 = (0110)_{2421}$  (unused)

# Self-Complementing

## Self-Complementing

- 9's complement of a decimal number is directly obtained by using 1's complement of the code

<b><i>Decimal Code</i></b>	<b><math>(3)_{10}</math></b>		<b><math>(6)_{10}</math></b>
<b><i>Excess-3</i></b>	<b>0110</b>	<b>↔</b>	<b>1001</b>
<b><i>2421</i></b>	<b>0011</b>	<b>↔</b>	<b>1100</b>
<b><i>8,4,-2,-1</i></b>	<b>0101</b>	<b>↔</b>	<b>1010</b>
<b>BCD</b>	<b>0011</b>	<b>↔</b>	<del><b>0110</b></del>

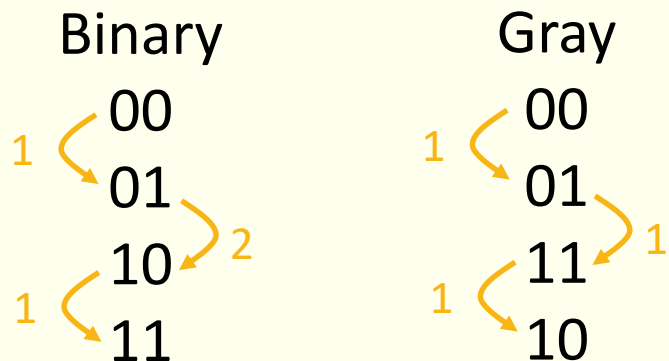
(BCD is not a self-complementing code, but it is easier to understand)

# Gray Code

- Advantage
  - only one bit in the code group changes in going from one number to the next

<i>Decimal</i>	<i>Binary</i>	<i>Gray</i>
7	0111	0100
8	1000	<u>1</u> 100

Some systems need this kind of feature.



Less bit transition (0→1 or 1→0)  
→ power consumption saving

Table 1-6  
Gray Code

Gray code	Decimal equivalent
0000	0
0001	1
0011	2
0010	3
0110	4
0111	5
0101	6
0100	7
1100	8
1101	9
1111	10
1110	11
1010	12
1011	13
1001	14
1000	15

Three factors for digital systems: cost, speed and power  
area performance

## Non-numeric Binary Codes

- A binary code used to distinguish the seven colors of the rainbow

Color	Binary Number
Red	000
Orange	001
Yellow	010
Green	011
Blue	100
Indigo	101
Violet	110

# ASCII Character Code

- American Standard Code for Information Interchange
- A popular code used to represent information sent as character-based data.
- It uses 7-bits (128 combinations) to represent:
  - 94 Graphic printing characters.
  - 34 Non-printing characters
- Some non-printing characters are used for text format (e.g. BS = Backspace, CR = carriage return)
- Other non-printing characters are used for record marking and flow control (e.g. STX and ETX start and end text areas).

**Table 1-7***American Standard Code for Information Interchange (ASCII)*

$b_4b_3b_2b_1$	$b_7b_6b_5$							
	000	001	010	011	100	101	110	111
0000	NUL	DLE	SP	0	@	P	˘	p
0001	SOH	DC1	!	1	A	Q	a	q
0010	STX	DC2	“	2	B	R	b	r
0011	ETX	DC3	#	3	C	S	c	s
0100	EOT	DC4	\$	4	D	T	d	t
0101	ENQ	NAK	%	5	E	U	e	u
0110	ACK	SYN	&	6	F	V	f	v
0111	BEL	ETB	*	7	G	W	g	w
1000	BS	CAN	(	8	H	X	h	x
1001	HT	EM	)	9	I	Y	i	y
1010	LF	SUB	*	:	J	Z	j	z
1011	VT	ESC	+	;	K	[	k	{
1100	FF	FS	,	<	L	\	l	
1101	CR	GS	—	=	M	]	m	}
1110	SO	RS	.	>	N	^	n	~
1111	SI	US	/	?	O	—	o	DEL

### ***Control characters***

NUL	Null	DLE	Data-link escape
SOH	Start of heading	DC1	Device control 1
STX	Start of text	DC2	Device control 2
ETX	End of text	DC3	Device control 3
EOT	End of transmission	DC4	Device control 4
ENQ	Enquiry	NAK	Negative acknowledge
ACK	Acknowledge	SYN	Synchronous idle
BEL	Bell	ETB	End-of-transmission block
BS	Backspace	CAN	Cancel
HT	Horizontal tab	EM	End of medium
LF	Line feed	SUB	Substitute
VT	Vertical tab	ESC	Escape
FF	Form feed	FS	File separator
CR	Carriage return	GS	Group separator
SO	Shift out	RS	Record separator
SI	Shift in	US	Unit separator
SP	Space	DEL	Delete

# Error-Detection/Correction Codes

- Redundancy (e.g. extra information), in the form of extra bits, can be incorporated into binary code words to **detect and correct** errors.
- A simple form of redundancy is parity, an extra bit appended onto the code word to make the number of 1's odd or even. Parity can **detect** all single-bit errors and some multiple-bit errors.
- Even parity : the number of 1's in the code word is even.
- Odd parity : the number of 1's in the code word is odd.



# Parity Code Example (error detection)

Even Parity Message - Parity	Odd Parity Message - Parity
000 - 0	000 - 1
001 - 1	001 - 0
010 - 1	010 - 0
011 - 0	011 - 1
100 - 1	100 - 0
101 - 0	101 - 1
110 - 0	110 - 1
111 - 1	111 - 0

- 1-bit parity + 7-bit ASCII
  - '0' 1000001 (parity bit = 0 for even parity)
- When an error occurs ... re-transmission

# UNICODE

- **UNICODE extends ASCII to 65,536 universal character codes**
  - For encoding characters in world languages
  - Available in many modern applications
  - 2 byte (16-bit) code words

Code (Hex)	Character	Source
0041	A	English (Latin)
042F	Я	Russian (Cyrillic)
0E09	๑	Thai
13EA	Ꮝ	Cherokee
211E	℞	Letterlike Symbols
21CC	⇒	Arrows
282F	⠠	Braille
345F	倝	Chinese/Japanese/ Korean (Common)

# Binary Storage & Registers

- Binary cell
  - 2 stable states, 1-bit information (0 or 1)
- Register
  - a group of binary cells,  $n$ -bit ,  $2^n$  states
  - PC and AC in X86 CPU
  - to hold the data to be processed or output
  - The content may be interpreted differently for different type of data (number, character).

01000001 → 65 or “A” or ....

(depend on its current situation)

# Register Transfer

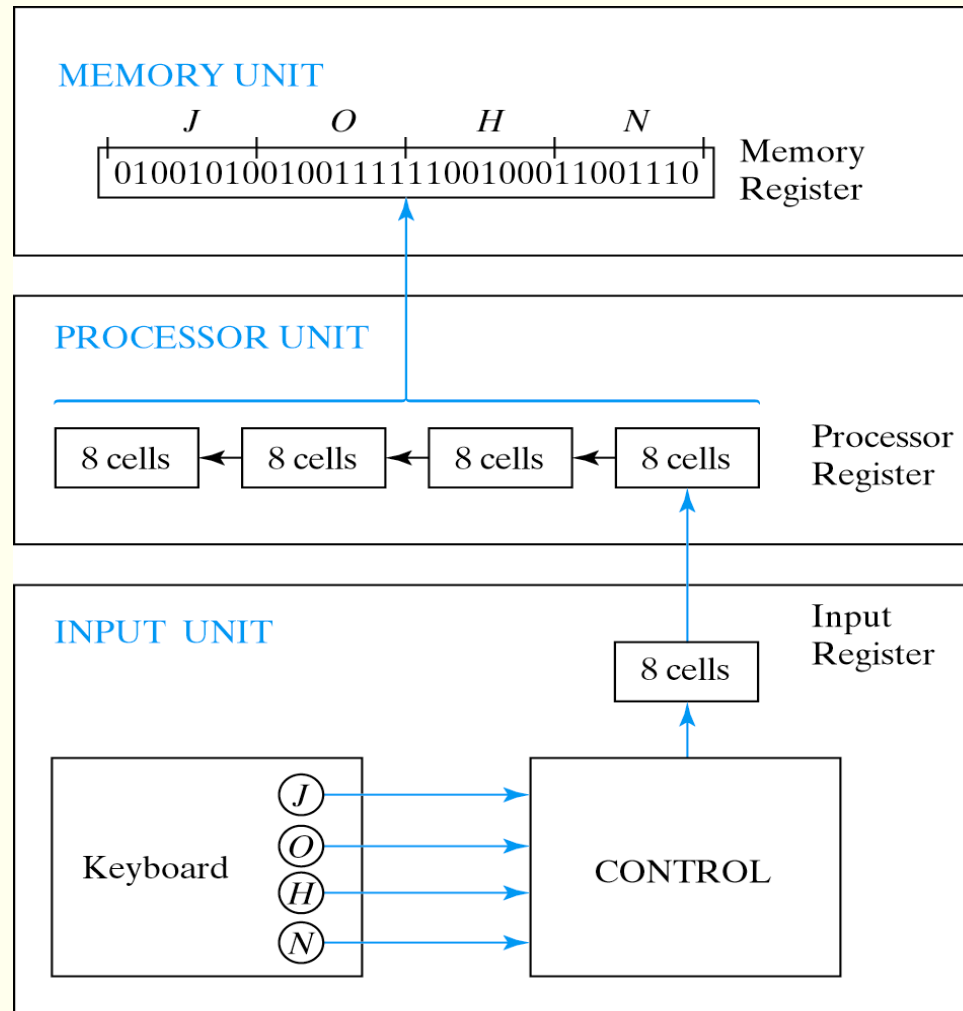


Fig. 1-1 Transfer of information with registers

# Binary Information Processing

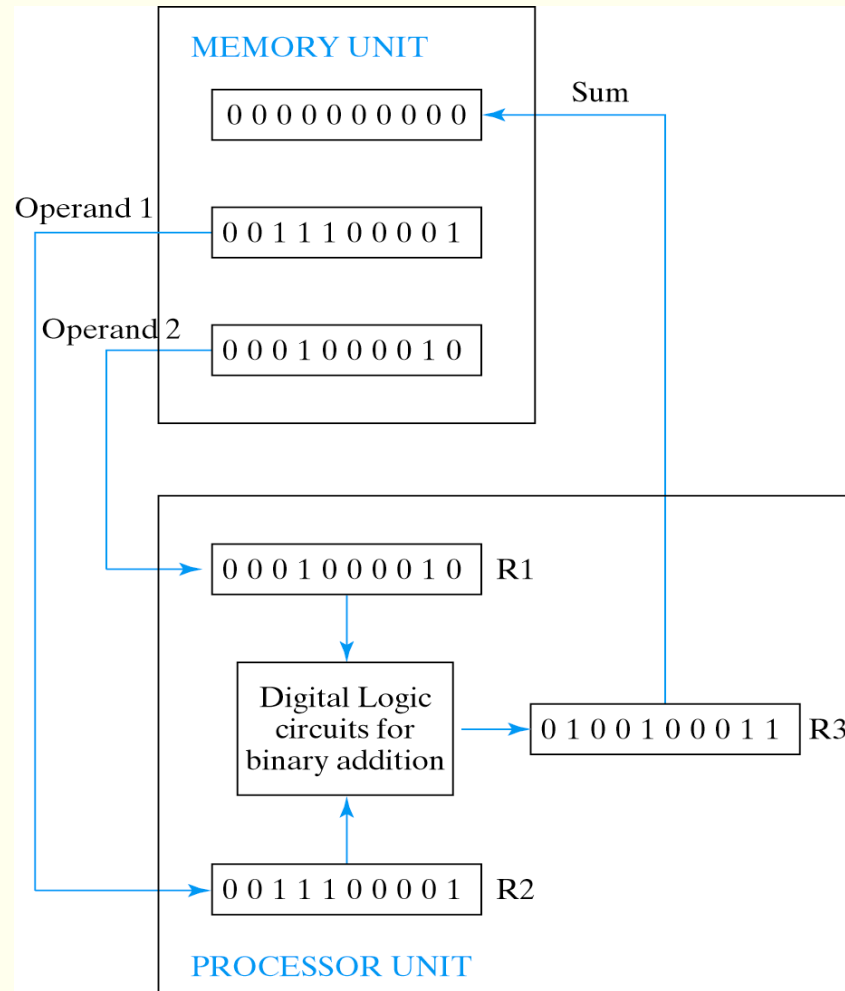


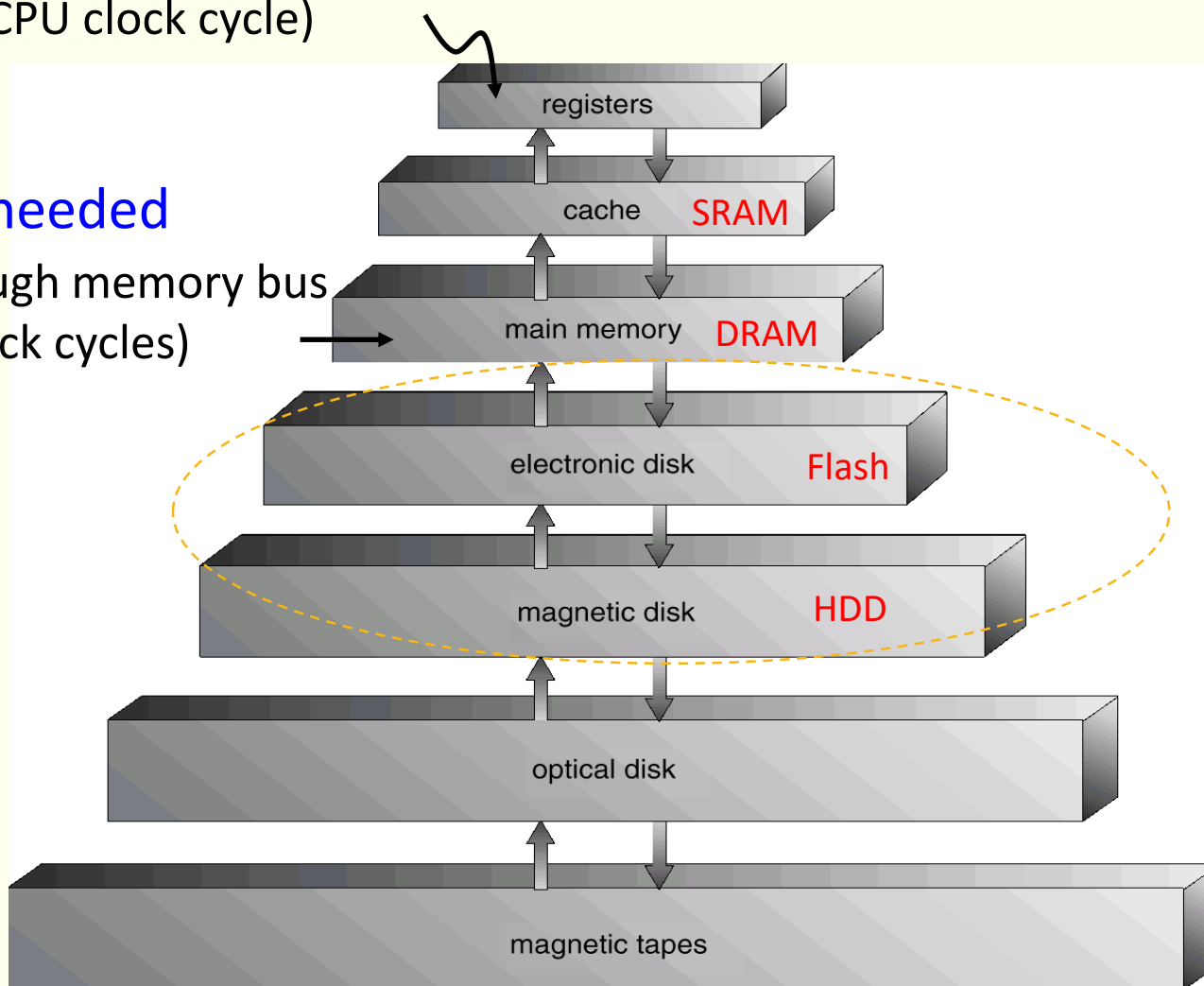
Fig. 1-2 Example of binary information processing

# Storage-Device Hierarchy

accessed directly by CPU  
(one CPU clock cycle)

A cache is needed

accessed through memory bus  
(many CPU clock cycles)



cost

cheap

speed

fast

# Binary Signals

**Logic 1**    3~4V (typical)

**Logic 0**    0~1V (typical)

Binary Logic

Deals with variables  
that take on 2 discrete  
values {0,1}

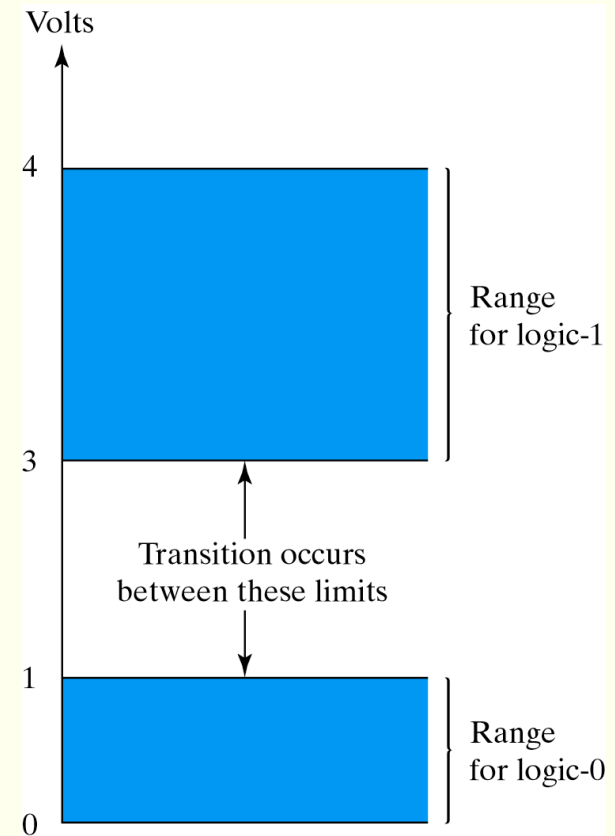


Fig. 1-3 Example of binary signals

# Binary Logic

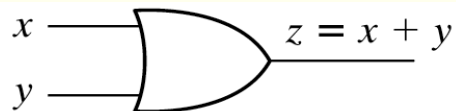
- Binary logic deals with binary variables, each having 2 values {0,1}

Logical operations: AND, OR, NOT, ...

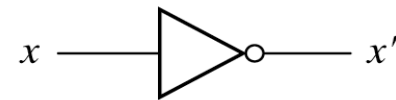
- Logic Gate
  - Electronic circuits that operate on one more input signals to produce an output signal



(a) Two-input AND gate



(b) Two-input OR gate



(c) NOT gate or inverter

AND

$x$	$y$	$x \cdot y$
0	0	0
0	1	0
1	0	0
1	1	1

OR

$x$	$y$	$x + y$
0	0	0
0	1	1
1	0	1
1	1	1

NOT

$x$	$x'$
0	1
1	0



# Time Diagram

- Time Diagram

Illustrate the response of each gate to the input

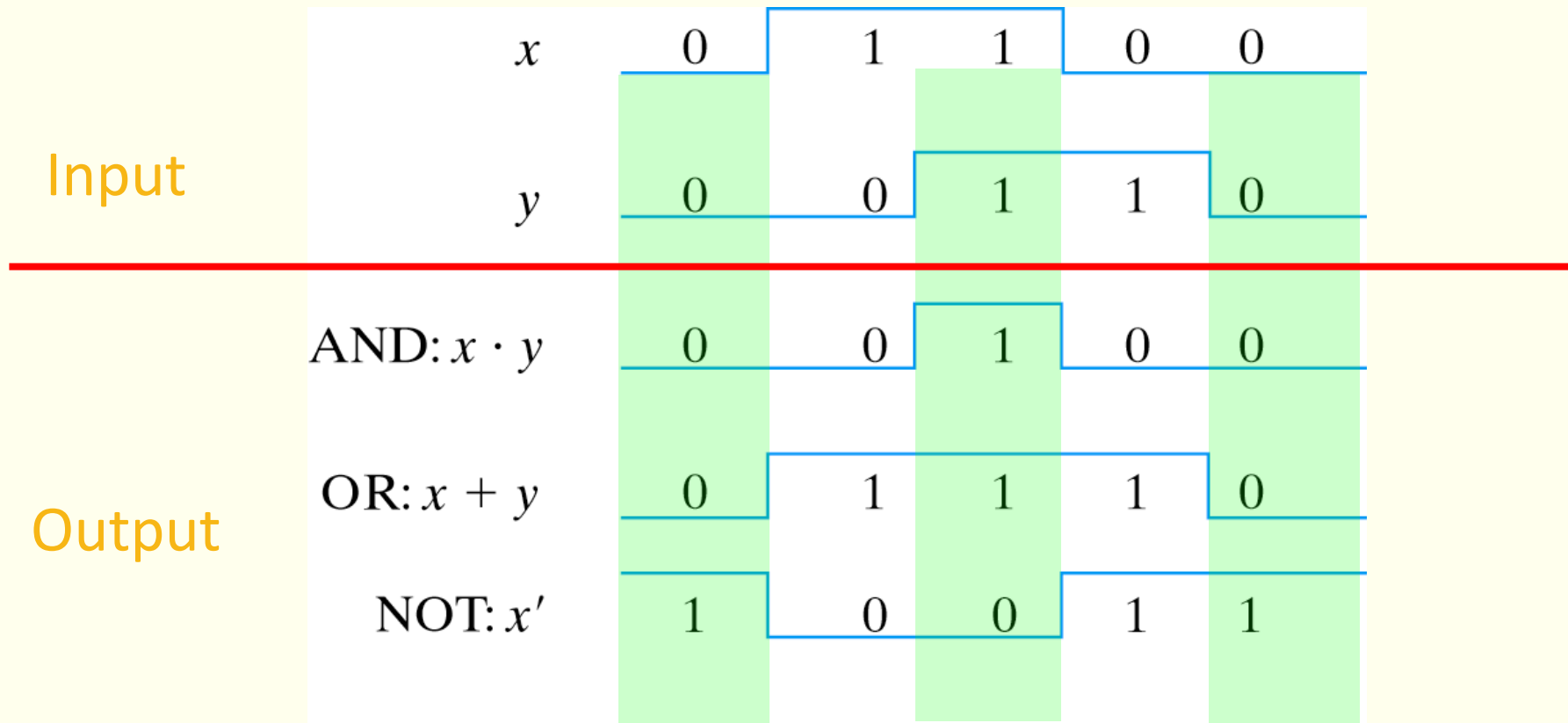


Fig. 1-5 Input-output signals for gates