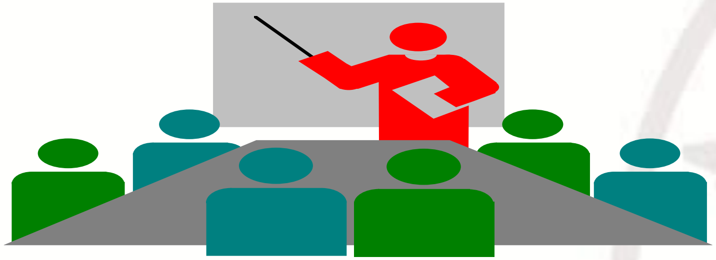


逻辑与计算机设计基础

LOGIC and Computer Design Fundamentals

CHAPTER 1

Digital Systems and Information

施青松
Asso. Prof. Shi Qingsong
College of Computer Science and Technology, Zhejiang University
zjsqs@zju.edu.cn

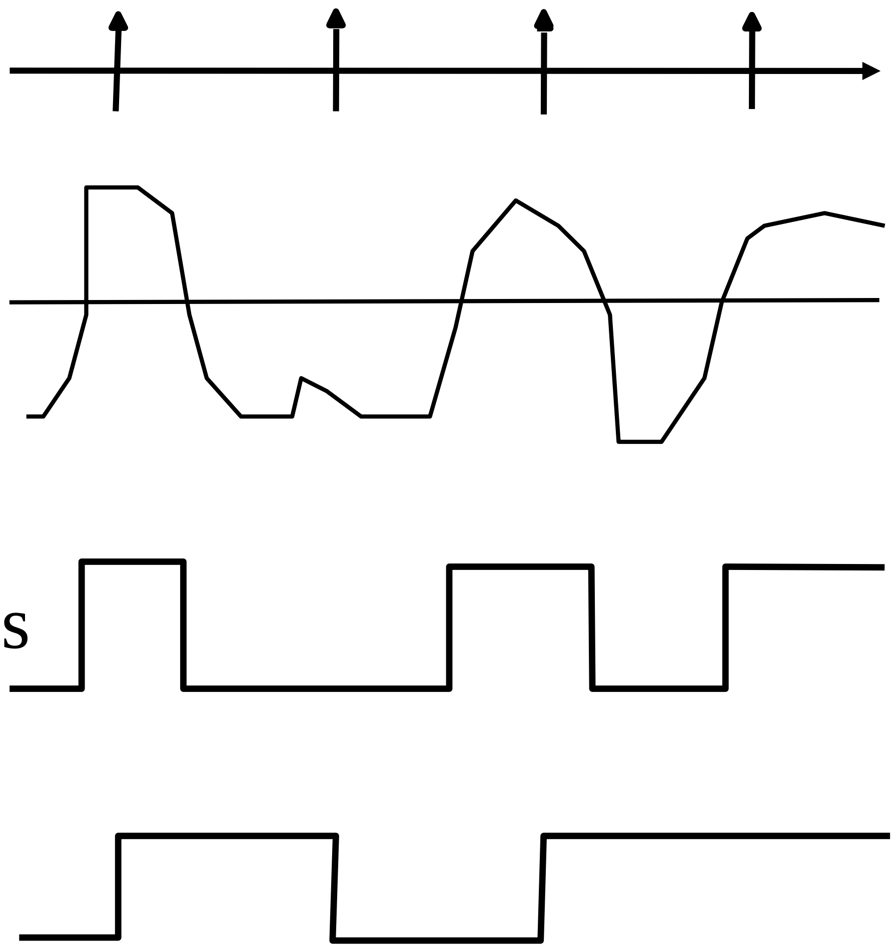
Course Outline

- Signal
- Digital Systems
- Digital Computer
- Organization Of Computer
- Number Systems & Codes

Signal

- **Information variables**
  - represented by physical quantities.
- **For digital systems**
  - the variables take on discrete values
  - Two level, or **binary values** are the most prevalent values in digital systems
- **Represented abstractly by:**
  - digits： 0 、1
  - symbols：
    - False (F) 、 True (T)
    - Low (L) 、 High (H)
    - On 、 Off
- **Binary values are represented by values or ranges of values of physical quantities**

Time Sequence Signal



Analog

Continuous:
value & time

Digital

Asynchronous

Discrete: value
Continuous: time

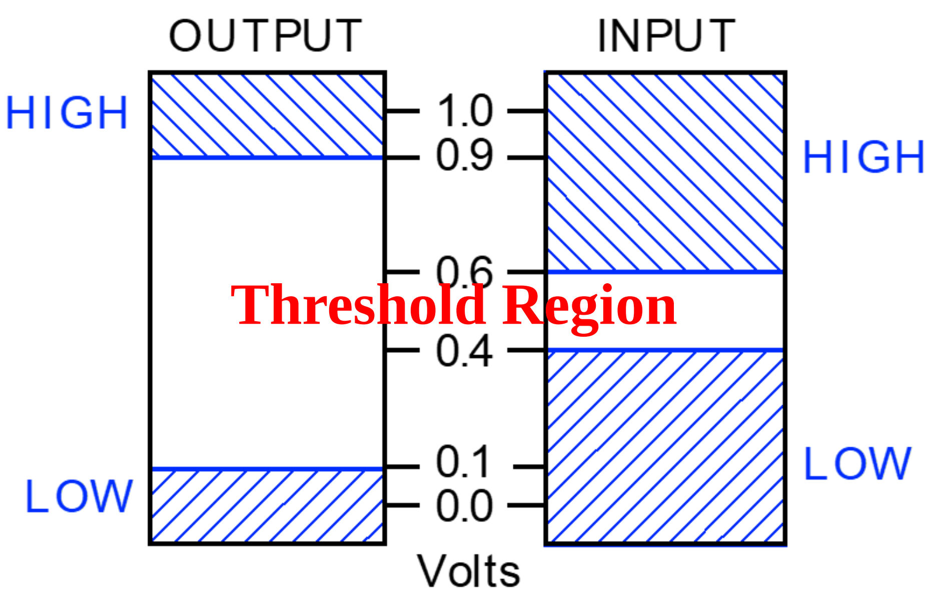
Synchronous

Discrete:
value & time

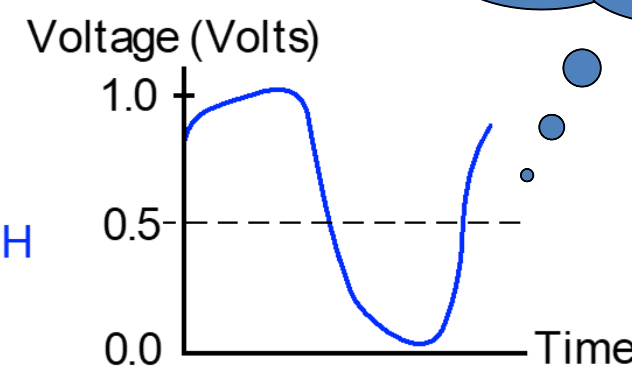
Physical Quantity Example : Voltage

- The most commonly used two-valued information
  - is an electrical signal - voltage or current
- typically two discrete values represented
  - by the voltage range of values

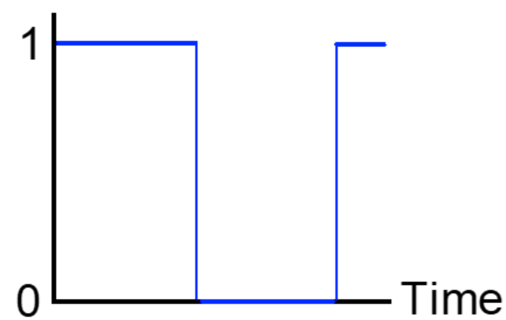
Why not use the decimal?



(a) Example voltage ranges



(b) Time-dependent Voltage



(c) Binary model of time-dependent voltage

Binary Values: Other Physical Quantities

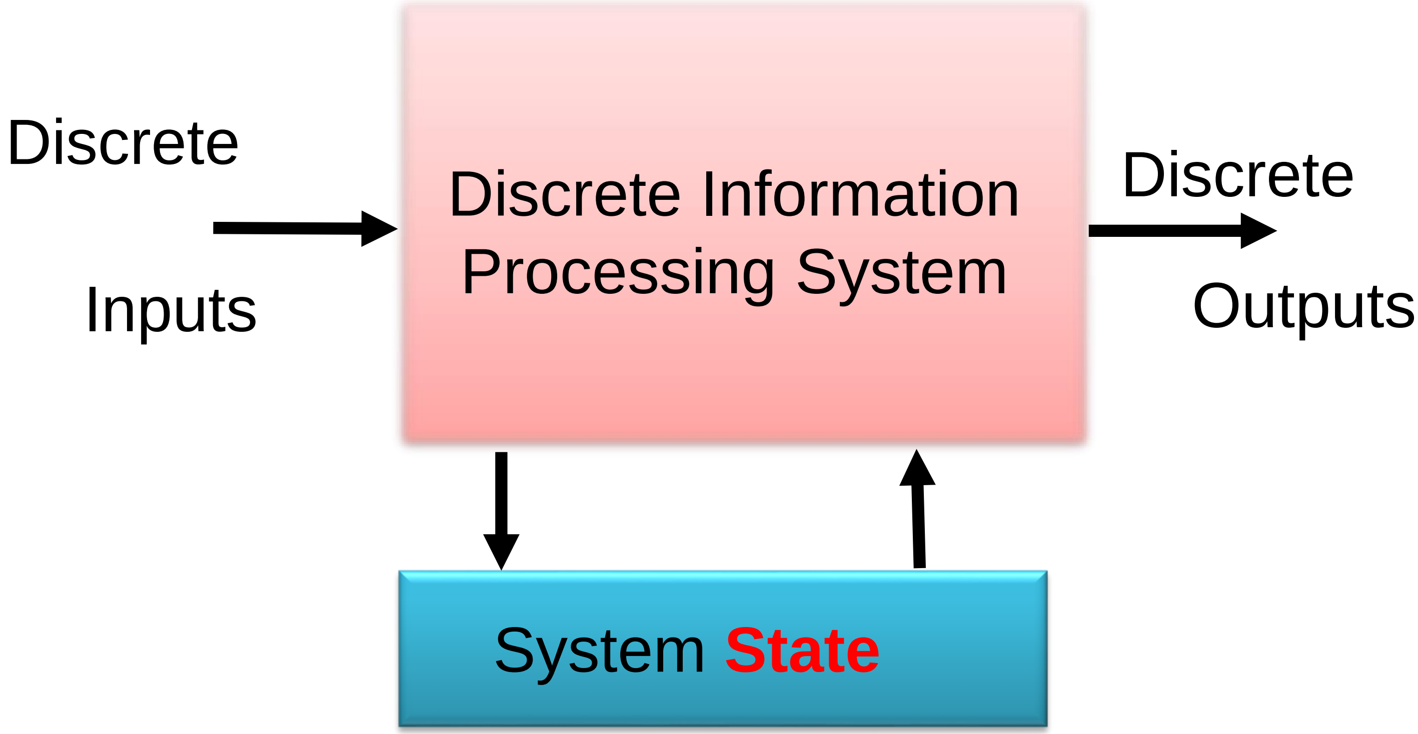
- What are other physical quantities represent 0 and 1?
  - CPU — Voltage
  - Disk — Magnetic Field Direction
  - CD — Surface Pits/Light
  - Dynamic RAM — Electrical Charge

Course Outline

- Signal
- Digital Systems
- Digital Computer
- Organization Of Computer
- Number Systems & Codes

Digital System

- Takes a set of discrete information **inputs**
- and discrete internal information (**system state**)
- and generates a set of discrete information **outputs**.



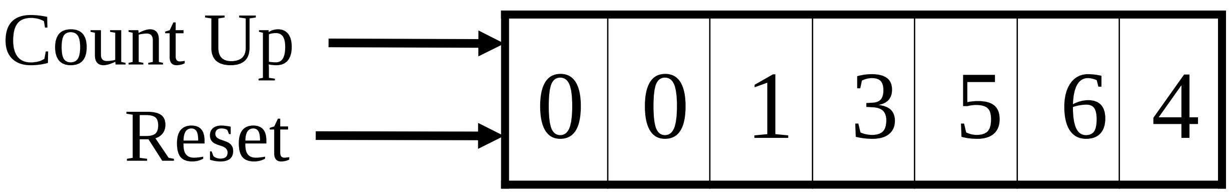
Types of Digital Systems

- **Combinational Logic System**
  - No state present
  - Output function： f_{Output} = Function(Input)
    General multivariate, multi-output function

- **Sequential System**
  - State present
  - State updated at discrete times
    - Synchronous Sequential System
  - State updated at any time
    - Asynchronous Sequential System
  - State Function： f_{State} = Function (State, Input)
  - Output Function： f_{Output} = Function (State)
    Or f_{Output} =Function (State, Input)

Digital System Example:

A Digital Counter (e. g., odometer):



Inputs: Count Up, Reset
Outputs: Visual Display
State: "Value" of stored digits

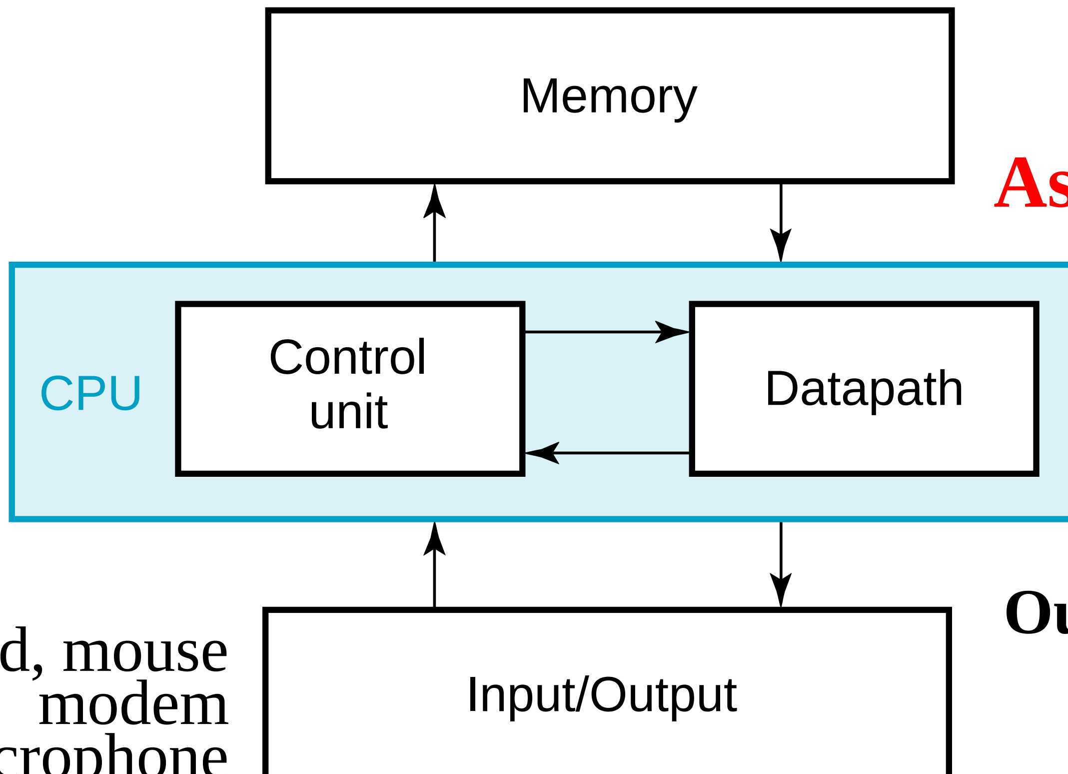
Synchronous or Asynchronous?

Course Outline

- Signal
- Digital Systems
- Digital Computer
- Organization Of Computer
- Number Systems & Codes

Digital Computer Example

A common system used on
the discrete elements in the information processing



Synchronous or Asynchronous?

Inputs:
Keyboard, mouse
modem
microphone

Outputs:
CRT, LCD
modem
speakers

Digital Computer

1. Features: commonality, flexibility, versatility

- A common System for processing the discrete elements of the information

2. Information representation within Computer

- used the binary numerical system： 0 和 1。
- A binary signal is represents one bit （bit）。
- Multi-digit bit used to represent data & Instructions can be executed in the computer
- Analog done automatically converted into a digital-value used on analog-to-digital conversion apparatus

Course Outline

- Signal
- Digital Systems
- Digital Computer
- Organization Of Computer
- Number Systems & Codes

Memory

- Can be stored Program and Data from input & output, and intermediate results
  - Main Memory
  - The external memory (as part of the peripheral)
  - Cache

Datapath(BUS)

- The channel between the processor, memory, and input / output device (connection)
  - Processor bus （within CPU）
  - I/O BUS:Different data transfer rates of the two buses,
    - different bus data communication through the completion of the bus interface hardware

Computer Architecture--2

- **Control unit**
  - Monitoring the exchange of information between the different parts
- **CPU(Central processor Unit)**
  - Composed by the data path and control unit. The modern processor comprises 4 functional modules：CPU， FPU， MMU & Internal cache
    - FPU(Floating-point unit): specific to the implementation of floating-point operations 。
    - MMU (Memory Management Unit): see the CPU storage device the size of multi-size larger than the actual physical RAM.

Computer Architecture--2

- **Input/Output device** （I/O）
  - Device for information processing systems interact with each other

And Beyond – Embedded Systems

- **Specific computer systems**
  - Computers as integral parts of other products
- **Examples of embedded computers**
  - Microcomputers
  - Microcontrollers
  - Digital signal processors

Examples of Embedded

- **Examples of Embedded Systems Applications**
  - Cell phones/Smart phone
  - Automobiles
  - Video games
  - Copiers
  - Dishwashers
  - Flat Panel TVs
  - Global Positioning Systems
  - AI Accelerate

Course Outline

- Signal
- Digital Systems
- Digital Computer
- Organization Of Computer
- **Number Systems & Codes**

Number Systems

- **The rule of the number system that constraints on the number Value**
  - The most commonly encountered daily in the **decimal** counting system, in Digital system widely used in the computer **binary, octal and hexadecimal**.

NUMBER SYSTEMS: Positional Arry

- **Positional Number System**
  - Three important factors: **radix、cardinalit、Weight**

Cardinality—Represents the number of digital collection (basic symbols) within Counting system

radix—Size of the collection(base)

e.g.: assume radix: R
R basic symbols， 0， 1， 2……,R
Every R Carry in 1

Weight or **Bit weights**： Determine the **digit position**

(Weight value of Digit at some position)

Each Weight is a **Power** of R corresponding to the digit's position

E.g.： 2**3**56 in Decimal， 3' **Weight** is 10^2

for **8**421 Coder first bit **Weight** is 8

NUMBER SYSTEMS: Representation

- **Representation Method for R Arry**
  - N Bit digits from left to right
  - Size: m + n

represented by a string of digits $0 \le A \le R$

$$(N)_R = (A_{n-1}A_{n-2}A_{n-3}\ldots A_1A_0 \bullet A_{-1}A_{-2}A\ldots A_{-m+1}A_m)_R$$

MSD LSD

Represents the power series $(N)_R = (\sum_{i=-m}^{n-1} A_i R^i)_R$

Instance represents a decimal number

radix： $R=10$

basic symbols： 0,1,2,3…9

Weight： $W_i = 10^i$

Representation：

$$(N)_{10} = \left(\sum_{i=-m}^{n-1} A_i 10^i\right)_{10} =$$

$$A_{n-1} \bullet 10^{n-1} + A_{n-2} \bullet 10^{n-2} + \ldots + A_1 \bullet 10^1 + A_0 \bullet 10^0 + A_{-1} \bullet 10^{-1} + \ldots + A_{-m} \bullet 10^{-m}$$

e.g. $(123.45)_{10} = 1\cdot10^2 + 2\cdot10^1 + 3\cdot10^0 + 4\cdot10^{-1} + 5\cdot10^{-2}$

Instance represents a binary number

- Representation

radix： R=2

basic symbols： 0， 1

Weight： $W_i = 2^i$

Representation：

$$(N)_2 = (\sum_{i=-m}^{n-1} A_i 2^i)_2 =$$

$$A_{n-1} \cdot 2^{n-1} + A_{n-2} \cdot 2^{n-2} + \ldots + A_1 \cdot 2^1 + A_0 \cdot 2^0 + A_{-1} \cdot 2^{-1} + \ldots\ldots + A_{-m} \cdot 2^{-m}$$

e.g.: $(1011.101)_2 = 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 + 1 \cdot 2^{-1} + 0 \cdot 2^{-2} + 1 \cdot 2^{-3}$

2. Rules of operation

ADD: $0 + 0 = 0$ $\quad$ $0 + 1 = 1$

$1 + 0 = 1$ $\quad$ $1 + 1 = 10$

MUL: $0 \times 0 = 0$ $\quad$ $0 \times 1 = 0$

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

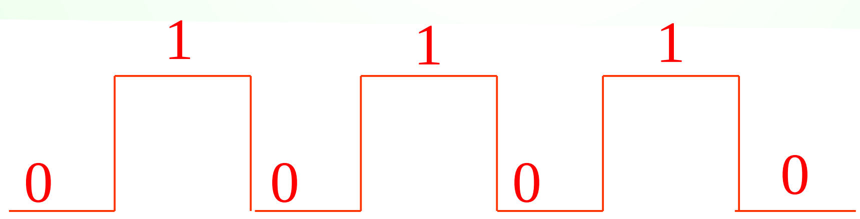
3. Physical representation

Convenience: with transistors or magnetic

Trouble: written 、 memory –

Octal or hexadecimal abbreviations

The high / low voltage can be used to represent the binary number 1,0.



4. May use Boolean algebra

Common values

n	2^n	n	2^n	n	2^n
0	1	8	256	16	65,536
1	2	9	512	17	131,072
2	4	10	1,024	18	262,144
3	8	11	2,048	19	524,288
4	16	12	4,096	20	1,048,576
5	32	13	8,192	21	2,097,152
6	64	14	16,384	22	4,194,304
7	128	15	32,768	23	8,388,608

Instance represents a octal number

radix：R=8

basic symbols：0、1、3…、7

Weight: Wi＝8^i

Representation：

$$(N)_8=\left(\sum_{i=-m}^{n-1} A_i 8^i\right)_8 =$$

$$A_{n-1}\bullet 8^{n-1}+A_{n-2}\bullet 8^{n-2}+\ldots+A_1\bullet 8^1+A_0\bullet 8^0+A_{-1}\bullet 8^{-1}+\ldots\ldots+A_{-m}\bullet 8^{-m}$$

e.g.: $(567.125)_8$

Instance represents a Hexadecimal number

radix： **R=16**

basic symbols： **0、1、3…、9, A、B …、F**

Weight: **$W_i = 16^i$**

Representation：

$$(N)_{16} = \left(\sum_{i=-m}^{n-1} A_i 16^i\right)_{16} =$$

$$A_{n-1} \bullet 16^{n-1} + A_{n-2} \bullet 16^{n-2} + \ldots + A_1 \bullet 16^1 + A_0 \bullet 16^0 + A_{-1} \bullet 16^{-1} + \ldots + A_{-m} \bullet 16^{-m}$$

e.g. $(5AF.9B)_{16}$

If remember then benefit

Decimal (base 10)	Binary (base 2)	Octal (base 8)	Hexadecimal (base 16)
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

ARITHMETIC OPERATIONS

1. Adds and subtracts

example 1： $1100+10001=11101$

$$\begin{array}{r} 01100 \\ +\ \underline{10001} \\ 11101 \end{array}$$

example 2： $10011-11110=-01011$

$$\begin{array}{r} 11110 \\ -\ \underline{10011} \\ -\ 01011 \end{array}$$

（ Set minus sign for difference if Small minus big number） example 3：

$$(59F)_{16}+(E46)_{16}=(13E5)_{16}$$

$$\begin{array}{r} 59F \\ +\quad E46 \\ \hline 13E5 \end{array}$$

2. **Multiplication**

example 1：

$$(762)_8\times(45)_8=(43772)_8$$

$$\begin{array}{r} 762 \\ \times\quad 45 \\ \hline 4672 \\ 3710 \\ \hline 43772 \end{array}$$

ARITHMETIC：BCD OPERATIONS

3. BCD adder

example：

$$448+489=(0100\ 0100\ 1000)_{BCD}+(0100\ 1000\ 1001)_{BCD}$$

$$
\begin{array}{r}
0100\ 0100\ 1000 \\
+\quad 0100\ 1000\ 1001 \\
\hline
1000\ 1101\ 0001 \\
+\quad 0110\ 0110 \\
\hline
1001\ 0011\ 0111
\end{array}
$$

illustrate： When each the sum is greater than 9 or Carry, need to adjustments with using plus 6

Number Systems

Converting from 2、8 、16:

1. The right expansion method
2. Fractional shift method
3. Digital replacement method

Converting between binary and octal

Digital replacement

原理：由于 $2^3 = 8$ 故三位二进制能表示为一位八进制数

方法：以小数点为中心 —— 整数右对齐

—— 小数左对齐

例： $(67.731)_8 =（110\ 111\ .111\ 011\ 001)_2$

$(312.64)_8=(011\ 001\ 010\ .\ 110\ 1)_2$

$(11\ 111\ 101\ .\ 010\ 011\ 11)_2=(375.236)_8$

$(10\ 110.11)_2=(26.6)_8$

Converting between binary and Hexadecimal

Digital replacement

原理：由于 $2^4 = 16$　故**四位**二进制能表示为一位十六进制数

方法：以小数点为中心——整数右对齐

——小数左对齐

例：　$(3AB4.1)_{16} = (0111\ 1010\ 1011\ 100\ .0001)_2$

$(21A.5)_{16} = (0010\ 0001\ 1010\ .\ 0101)_2$

$(1001101.01101)_2 = (\ 0100\ \ 1101.\ 0110\ 1000\ )_2 = (4D.68)_{16}$

$(111\ 1101\ .\ 0100\ 1111)_2 = (7D.4F)_{16}$

$(110\ 0101.101)_2 = (65.A)_{16}$

Converting between binary and decimal

Converting Binary to Decimal

Right expansion

原理：权展开表达式

方法：权相加 ---- 权展开十进制相加

例：$(110\ 0101.101)_2 = 1*2^6+1*2^5+0*2^4+0*2^3+$

$+1*2^2+0*2^1+1*2^0+1*2^{-1}+0*2^{-2}+1*2^{-3}$

$=(813.625)_{10}$

Converting between binary and decimal

Converting Decimal to Binary

原理： **The integer**—权展开式除2，余数构成最低位

Fractional —权展开式乘2，整数构成最高位

方法：**整数** —— 除2取余

小数 —— 乘2取整

例：**(725.678)**＝(10 1101 0101. 1010 1101 1001) $_2$

＝(**2D5.AD9**) $_{16}$

1. The integer portion: divided by 2, to get the remainder

e.g.: Converting $(725)_{10}$ **to Binary**

$$(725)_{10} = (k_{n-1}\ \ k_{n-2} \cdots k_1\ \ k_o)_2$$

$$= k_{n-1} \times 2^{n-1} + k_{n-2} \times 2^{n-2} + \cdots k_1 \times 2^1 + k_o \times 2^0$$

$$= 2(k_{n-1} \times 2^{n-2} + k_{n-2} \times 2^{n-3} + \cdots k_1) + k_o$$

1

$$(362 + \tfrac{1}{2})_{10} = k_{n-1} \times 2^{n-2} + k_{n-2} \times 2^{n-3} + \cdots k_1 + \frac{k_o}{2}$$

0

$$(181 + \frac{0}{2})_{10} = k_{n-1} \times 2^{n-3} + k_{n-2} \times 2^{n-4} + \cdots k_2 + \frac{k_1}{2}$$

$$\vdots$$

$$(725)_{10} = (10\ \ 1101\ \ 0101)_2$$

- ***short division:*** **The integer portion:**

divided by 2, to get the remainder

$$(725)_{10} = (10\ 1101\ 0101)_2$$

```
2 | 7 2 5
2 | 3 6 2 ............1
2 | 1 8 1 ............0
  2 | 9 0 ............1
  2 | 4 5 ............0
  2 | 2 2 ............1
  2 | 1 1 ............0
    2 | 5 ............1
    2 | 2 ............1
    2 | 1 ............0
    2 | 0 ............1
```

2. **Fractional part: multiplied by 2, to take the integral number**

e.g. : Converting $(0.678)_{10}$ **to Binary**

$$(0.678)_{10} = \frac{k_{-1}}{2} + \frac{1}{2}(k_{-2} \times 2^{-1} + \cdots + k_{-m} \times 2^{-m+1})$$

1

$$(1 + 0.356)_{10} = k_{-1} + (k_{-2} \times 2^{-1} + \cdots + k_{-m} \times 2^{-m+1})$$

0

$$(0 + 0.712)_{10} = k_{-2} + (k_{-3} \times 2^{-1} + \cdots + k_{-m} \times 2^{-m+2})$$

$$(1 + 0.424)_{10} = a_{-3} + (k_{-4} \times 2^{-1} + \cdots + k_{-m} \times 2^{-m+3})$$

1

$$\vdots$$

$$(0.678)_{10} = (0.1010\ \ 1101\ \ 1001)_2$$

Note: **can not be accurately converted**

Fractional part: multiplied by 2，

to take the integral number

$2 \times 0.678 \ldots\ldots\ldots\ldots = 1.356$

$2 \times 0.356 \ldots\ldots\ldots\ldots = 0.712$

$2 \times 0.712 \ldots\ldots\ldots\ldots = 1.424$

$2 \times 0.424 \ldots\ldots\ldots\ldots = 0.848$

$2 \times 0.848 \ldots\ldots\ldots\ldots = 1.696$

$2 \times 0.696 \ldots\ldots\ldots\ldots = 1.392$

$2 \times 0.392 \ldots\ldots\ldots\ldots = 0.784$

$2 \times 0.784 \ldots\ldots\ldots\ldots = 1.568$

$2 \times 0.568 \ldots\ldots\ldots\ldots = 1.136$

$2 \times 0.136 \ldots\ldots\ldots\ldots = 0.272$

$2 \times 0.272 \ldots\ldots\ldots\ldots = 0.544$

$2 \times 0.544 \ldots\ldots\ldots\ldots = 1.088$

3.3位二进制数为1位十进制数

$(0.678)_{10} = (0.1010\ 1101\ 1001)_2$

Course Outline

- Signal
- Digital Systems
- Digital Computer
- Organization Of Computer
- Number Systems & Codes

Coding with Binary

Numeric

Binary Code Decimal

Binary Numbers and Binary Coding

- **Flexibility of representation**
  - Within constraints below, can assign any **binary combination** (called a code word) to any data
  - as long as data is uniquely encoded
- **Information Types**
  - **Numeric**
    - Must represent range of data needed
    - Very desirable to represent data such that simple, straightforward computation for common arithmetic operations permitted
    - Tight relation to binary numbers
  - **Non-numeric**
    - Greater flexibility since arithmetic operations not applied.
    - Not tied to binary numbers

Non-numeric Binary Codes

- Given n binary digits (called **bits**), a **binary code** is a **mapping from** a set of **represented elements to** a subset of the 2^n binary numbers.
- Example: A binary code for the seven colors of the rainbow
- Code 100 is not used

Color	Binary Number
Red	000
Orange	001
Yellow	010
Green	011
Blue	101
Indigo	110
Violet	111

Number of Bits Required

- Given M elements to be represented by a binary code, the **minimum number of bits**, *n*, needed, satisfies the following relationships:

  $$2^n \geq M > 2^{(n-1)}$$

  $$n = \lceil \log_2 M \rceil \text{ where } \lceil x \rceil,$$

  called the *ceiling function*:

  is the integer greater than or equal to *x*

- **Example: How many bits are required to represent decimal digits with a binary code?**

  $$n = \log_2 10 = ?$$

Number of Elements Represented

- Given n digits in radix r, there are r^n distinct elements that can be represented
- But, you can represent m elements, $m < r^n$
- **Examples:**
  - **You can represent 4 elements in radix r = 2 with n = 2 digits: (00, 01, 10, 11).**
  - **You can represent 4 elements in radix r = 2 with n = 4 digits: (0001, 0010, 0100, 1000).**
  - **This second code is called a "one hot" code.**

DECIMAL CODES - Binary Codes for Decimal Digits

There are over 8,000 ways that you can chose 10 elements from the 16 binary numbers of 4 bits. **A few are useful:**

Decimal	8,4,2,1	Excess3	8,4,-2,-1	Gray
0	0000	0011	0000	0000
1	0001	0100	0111	0100
2	0010	0101	0110	0101
3	0011	0110	0101	0111
4	0100	0111	0100	0110
5	0101	1000	1011	0010
6	0110	1001	1010	0011
7	0111	1010	1001	0001
8	1000	1011	1000	1001
9	1001	1100	1111	1000

Binary Coded Decimal (BCD)

- The BCD code is the 8,4,2,1 code.
  - 8, 4, 2, and 1 are weights
- BCD is a ***weighted* code**
  - This code 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

Example: 1001 (9) = 1000 (8) + 0001 (1)

- How many "invalid" code words are there?
- What are the "invalid" code words?

Excess 3 Code and 8, 4, –2, –1 Code

Decimal	Excess 3	8, 4, –2, –1
0	0011	0000
1	0100	0111
2	0101	0110
3	0110	0101
4	0111	0100
5	1000	1011
6	1001	1010
7	1010	1001
8	1011	1000
9	1100	1111

What interesting property is common to these two codes?

Warning: Conversion or Coding?

- Do NOT mix up *conversion* of a decimal number to a binary number with *coding* a decimal number with a BINARY CODE.
- $13_{10} = 1101_2$
  - This is *conversion*)
- $13 \Leftrightarrow 0001\ 0011$
  - This is *coding*

编码的“0”不能省略!!!

BCD Arithmetic

Given a BCD code, we use binary arithmetic to add the digits:

8	**1000**	**Eight**
+5	**+0101**	**Plus 5**
13	**1101**	**is 13 (> 9)**

Note that the result is MORE THAN 9, so must be represented by two digits!

To correct the digit, subtract 10 by adding 6 modulo 16.

8	**1000**	Eight
+5	**+0101**	Plus 5
13	**1101**	is 13 (> 9)
	+0110	so add 6
carry = 1	**0011**	leaving 3 + cy
0001	**0011**	Final answer (two digits)

If the digit sum is > 9, add one to the next significant digit

BCD Addition Example

- Add 2905_{BCD} to 1897_{BCD} showing carries and digit corrections.

				0
	0001	1000	1001	0111
+	<u>0010</u>	<u>1001</u>	<u>0000</u>	<u>0101</u>
	____	____	____	____

Parity Bit Error-Detection 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
- A code word has *even* parity
  - if the number of ***1***'s in the code word is *even*
- A code word has *odd* parity
  - if the number of ***1***'s in the code word is *odd*

4-Bit Parity Code Example

- Fill in the even and odd parity bits:

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

- The codeword "1111" has *even parity* and the codeword "1110" has *odd parity*. Both can be used to represent *3-bit data*

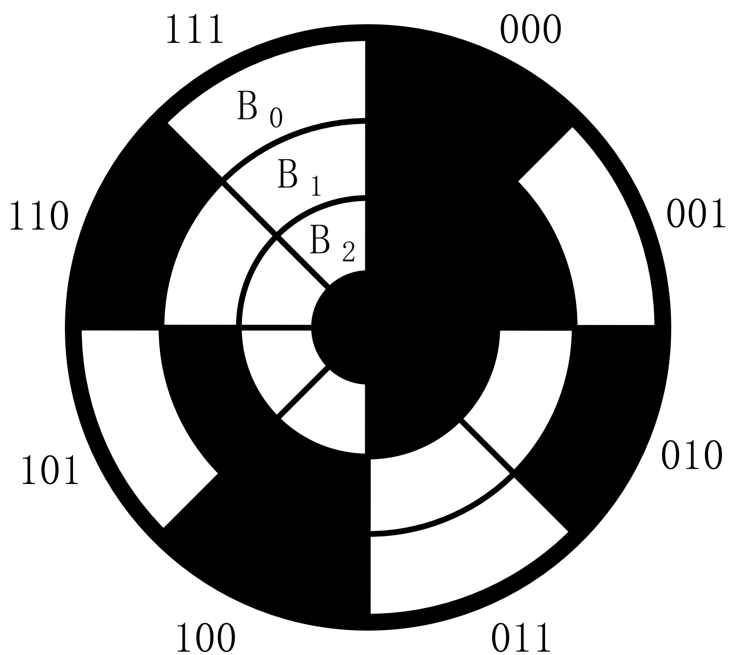
GRAY CODE – Decimal

Decimal	8,4,2,1	Gray
0	0000	0000
1	0001	0100
2	0010	0101
3	0011	0111
4	0100	0110
5	0101	0010
6	0110	0011
7	0111	0001
8	1000	1001
9	1001	1000

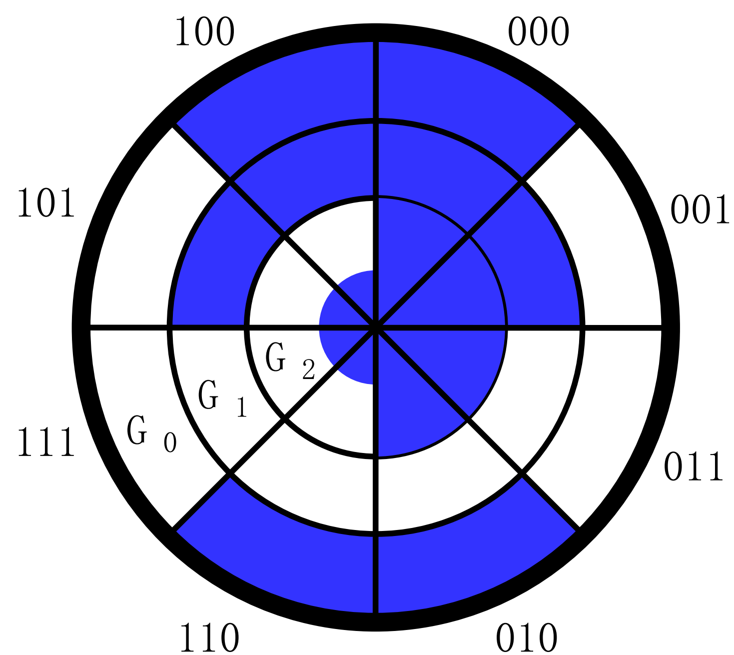
- What special property does the Gray code have in relation to adjacent decimal digits?

Optical Shaft Encoder

- Does this special Gray code property have any value?
- An Example: Optical Shaft Encoder



(a) Binary Code for Positions 0 through 7



(b) Gray Code for Positions 0 through 7

Shaft Encoder (Continued)

- How does the shaft encoder work?

- For the binary code, what codes may be produced if the shaft position lies between codes for 3 and 4 (011 and 100)?

- Is this a problem?

Shaft Encoder (Continued)

- For the Gray code, what codes may be produced if the shaft position lies between codes for 3 and 4 (010 and 110)?

- Is this a problem?

- Does the Gray code function correctly for these borderline shaft positions for all cases encountered in octal counting?

Several decimal coding table

Decimal	8421	5421	2421	Excess 3	GRAY1
0	0000	0000	0000	0011	0000
1	0001	0001	0001	0100	0001
2	0010	0010	0010	0101	0011
3	0011	0011	0011	0110	0010
4	0100	0100	0100	0111	0110
5	0101	0101	0101	1000	0111
6	0110	0110	0110	1001	0101
7	0111	0111	0111	1010	0100
8	1000	1011	1110	1011	1100
9	1001	1100	1111	1100	1000

Coding with Binary

Non-numeric

Character Code

ALPHANUMERIC CODES - ASCII Character Codes

- American Standard Code for Information Interchange
  - This code is a popular code used to represent information sent as character-based data. It uses 7-bits 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

7 BIT ASCII CODE TABLE

b6b5b4 / B3b2b1b0	000	001	010	011	100	101	110	111
0000	NUL	DLE	SP	0	@	P	,	p
0001	SOM	DC	!	1	A	Q	a	q
0010	STX	DC	“	2	B	R	b	r
0011	ETX	DC	#	3	C	S	c	s
0100	EOT	DC	$	4	D	T	d	t
0101	ENQ	NAA	%	5	E	U	e	u
0110	ACA	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	+	;	A	[	k	
1100	FF	FS	,	<	L	\	l	|
1101	CR	GS	—	=	M	]	m	
1110	SO	RS	.	>	N		n	~
1111	SI	US	/	?	O	←	o	DEL

ASCII Properties

ASCII has some interesting properties:

Digits 0 to 9 span Hexadecimal values 30_{16} to 39_{16}

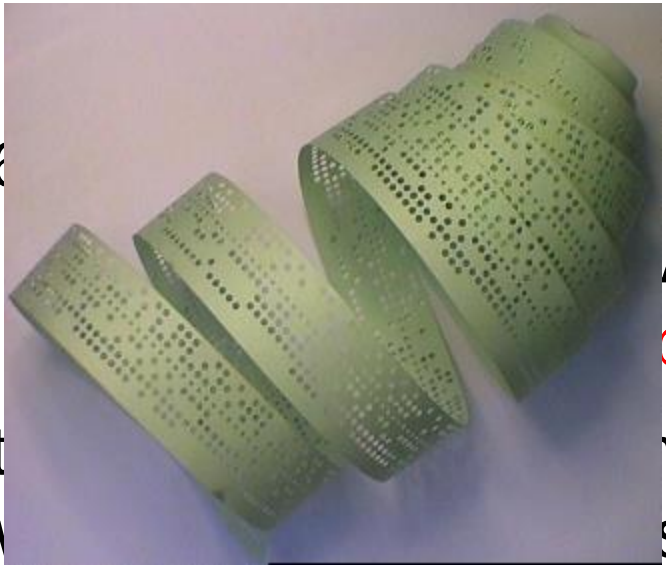
Upper case A- Z span

Lower case a-z span 6

Lower to upper case versa)

ping bit 6

Delete (DEL) is all bit n when punched paper tape ssages



Punching all holes in a row erased a mistake!

UNICODE

- UNICODE extends ASCII to 65,536 universal characters codes
  - For encoding characters in world languages
  - Available in many modern applications
  - 2 byte (16-bit) code words
  - See Reading Supplement – Unicode on the Companion Website **http://www.prenhall.com/mano**

Assignment

Ch1

1-3,1-9,1-12,1-13,1-16,1-18,1-19,1-28

Too simple, do not have much time


Thank you!