微机原理及接口技术

Hardware Principles and Interfacing of Modern Computer

Lecture 1: Number Systems

陈启军，张 伟

Email: zhang_wi@mail.tongji.edu.cn

Dept. Of Automation, TongJi University

Content

- 数的表示
  - 数的十进制表示，二进制表示和十六进制表示
  - 进位计数制的一般表示
  - 非十进制数到十进制数的转换
  - 十进制到非十进制数的转换（整数和小数）
  - 二进制与十六进制间的快速转换
- 数在计算机内部的表示
  - 机器数与真值的概念
  - 无符号数在计算机内的表示及有关运算
  - 有符号数在计算机中的表示：原码，反码和补码
  - 有符号二进制数和十进制数之间的转换
  - 补码的有关运算
- 十进制数的表示——BCD码
- 浮点数的表示与计算
- 非数值数据的表示：ASCII码
- 小结

Number Systems

- 数的表示
  - 从数字到数：数的进位计数制表示
    - 10进制表示、16进制表示、2进制表示、12进制表示等等
  - 数的不同进制表示之间的相互转化
- 数在计算机内部的表示
  - 真值和机器数
  - 二进制数的计算机表示：原码、反码和补码
  - 基于机器数的运算（算术运算和逻辑运算）
  - 其它表示：BCD码

Scale Number of Radix X

- 基本概念
  - 数字和数 (Digit & Number)
  - 基(radix or number base)
- 基本原理
  - 如何从数字组成数(From digits to number)
    - 数的进位计数制表示
    - 10进制表示，16进制表示，2进制表示，…

Scale Number of Radix 10

- 数的十进制表示

特点：以10为底，逢10进一；

共有0-9十个数字符号。

表示：

$$D = D_{n-1} \times 10^{n-1} + D_{n-2} \times 10^{n-2} + \cdots + D_0 \times 10^0$$

$$+ D_{-1} \times 10^{-1} + \cdots + D_{-m} \times 10^{-m}$$

$$= \sum_{i=-m}^{n-1} D_i \times 10^i$$

Scale Number of Radix 2

- 数的二进制表示

特点：以2为底，逢2进位；

只有0和1两个符号。

表示：

$$(B)_2 = B_{n-1} \times 2^{n-1} + B_{n-2} \times 2^{n-2} + \cdots + B_0 \times 2^0$$
$$+ B_{-1} \times 2^{-1} + \cdots + B_{-m} \times 2^{-m}$$
$$= \sum_{i=-m}^{n-1} B_i \times 2^i$$

Scale Number of Radix 16

- 数的十六进制表示

特点：以16为底，逢16进位；

有0--9及A--F共16个数字符号。

表示：

$$(H)_{16} = H_{n-1} \times 16^{n-1} + H_{n-2} \times 16^{n-2} + \cdots + H_0 \times 16^0 + H_{-1} \times 16^{-1} + \cdots + H_{-m} \times 16^{-m} = \sum_{i=-m}^{n-1} H_i \times 16^i$$

Scale Number of Radix X

进位计数制的一般表示

一般地，对任意一个K进制数S都可表示为

$$(S)_k = S_{n-1} \times K^{n-1} + S_{n-2} \times K^{n-2} + \cdots + S_0 \times K^0 + S_{-1} \times K^{-1} + \cdots + S_{-m} \times K^{-m} = \sum_{i=-m}^{n-1} S_i \times K^i$$

其中：

S_i -- S的第i位数码，可以是K个符号中任何一个；

n,m -- 含义同前；

K -- 基数(radix或number base)；

K^i -- K进制数的权(weight)

Scale Number Examples

- 进位计数制表示实例
- **Example 1: 6进制表示下的 25.2**

Power	6^1		6^0		6^-1	
Weight	6		1		.167	
Number	2		5.		2	
Numeric Value	12	+	5.	+	.333	= 17.333

- **Example 2: 2进制表示下的110.101**

Power	2^2	2^1	2^0	2^-1	2^-2	2^-3
Weight	4	2	1	0.5	0.25	0.125
Number	1	1	0.	1	0	1
Numeric Value	4 +	2 +	0 +	0.5 +	0 +	0.125 = 6.625

Origination of Binary Number

- 关于二进制的起源
- 起源1：技术条件
  - 数字电路本质且自然的表达
- 起源2：哲学味道
  - 一生二，二生三，三生万物。“二”蕴含了系统的所有表达
  - 莱布尼兹对中国阴阳学说的热衷和二进制的提出
- 起源3：
  - 程序员用电脑编程时发现只有两个大拇指最空还可用来作扳指头计算之用，于是决定采用二进制。

Conversion Between Different Scale Numbers

- 非十进制数到十进制数的转换
  - 按相应进位计数制的权表达式展开，再按十进制求和。(整数和小数均遵循该原则)

例： $10110010B = (?)_{10}$

$13FAH = (?)_{10}$

Conversion Between Different Scale Numbers

- 十进制到非十进制数的转换
  - 十进制 → 二进制的转换：

    整数部分：除2取余；

    小数部分：乘2取整。
  - 十进制 → 十六进制的转换：

    整数部分：除16取余；

    小数部分：乘16取整。

以小数点为起点求得整数和小数的各个位。

Conversion Between Different Scale Numbers

- 二进制与十六进制间的快速转换

  用4位二进制数表示1位十六进制数

例： ***10110001001.110 = (?)H***

0101 1000 1001.1100

5 8 9 . C

Numbers in Computer

- 基本概念：机器数及其真值
  - 在计算机中，一个数连同它的符号都用二进制的编码形式来表示，这种数称为机器数。(Machine Number)
  - 一个机器数对应的十进制数值称为这个机器数的真值(True Value)
- 基本概念：

  二进制数在计算机内的表示：原码、反码和补码
- 思考
  - 数值、数的二进制表示（二进制数）、真值、机器数、原码、反码和补码之间的联系与区别

Binary Number Operations

- 无符号数
- 有符号数

- 算术运算
- 逻辑运算

注意它们之间的区别

Unsigned Number Operations

- 无符号数在计算机内的表示
  - 思考：与前面讲的二进制数表示有何区别?
- 无符号数的算术运算(Arithmetic Operations)
  - 加法运算
  - 减法运算
  - 乘法运算
  - 除法运算

Unsigned Number Operations

- 计算规则
  - 加法：1+1=0（有进位），…
  - 减法：0-1=1（有借位），…
  - 乘法：…，乘以2相当于左移一位
  - 除法：…，除以2则相当于右移1位。

例：00101110×0000010 = ?

00101110/00000010 = ?

Unsigned Number Operations

- **Example:**

  00001011×0100=00101100B

  00001011÷0100=00000010B

  即：商=00000010B，余数=11B

Unsigned Number Operations

- 无符号数的表示范围

  一个n位的无符号二进制数X，其表示范围为

  $$0 \leqslant X \leqslant 2^n-1$$

  若运算结果超出这个范围，则产生溢出。

- 溢出的判断
  - 判别方法：运算时，当最高位向更高位有进位（或借位）时则产生溢出。

Unsigned Number Operations

Example

$$\begin{array}{rr} & 11111111 \\ + & 00000001 \\ \hline 1 & 00000000 \end{array}$$

结果超出 8 位（最高位有进位），发生溢出。
（结果为256，超出 8 位二进制数所能表示的范围 255）

Unsigned Number Operations

- 思考：X位无符号二进制数可表示的真值的范围是多少？
  - X = 1时
  - X = 2时
  - X = 4时
  - X = 8时
  - X = 16时
  - X = 32时
  - X = 64时

Unsigned Number Operations

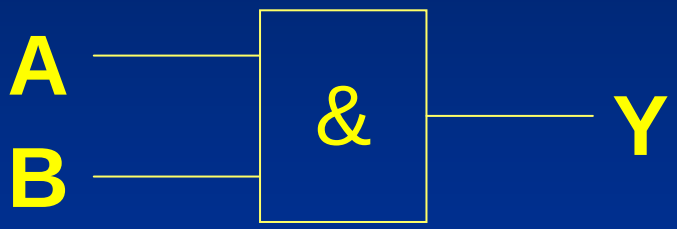
- 无符号数的逻辑运算(Logical Operations)
  - 与(∧)、或(∨)、非(¯) 、异或(⊕)
  - 特点：按位运算，无进借位
  - 运算规则

    …

Unsigned Number Operations

- 逻辑运算的数字电路实现——逻辑门
  - 与、或、非门逻辑符号和逻辑关系（真值表）
  - 与非门、或非门的应用

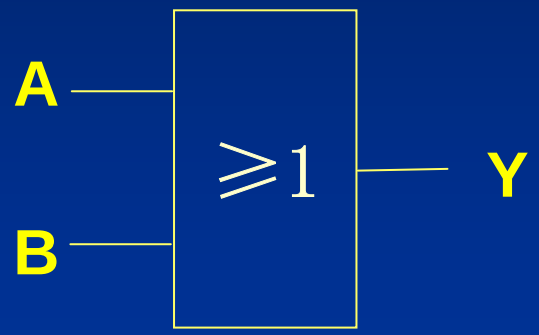
与门（AND Gate）



$Y = A \wedge B$

A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

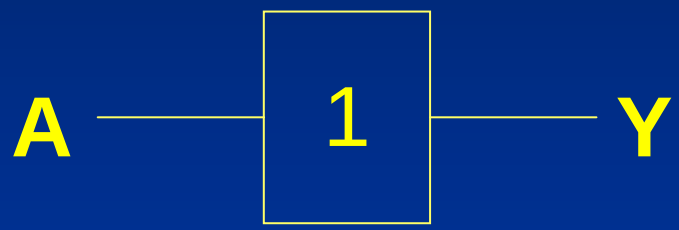
或门（OR Gate）



$Y = A \vee B$

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1

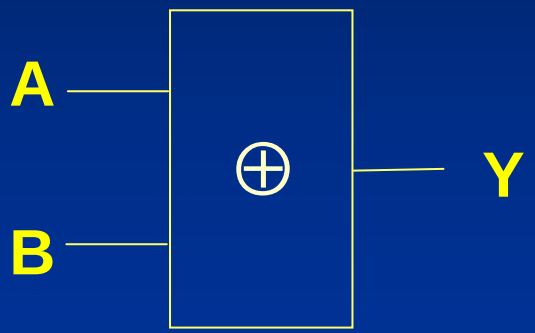
非门（NOT Gate）



$Y = \overline{A}$

A	Y
0	1
1	0

异或门（eXclusive OR Gate）

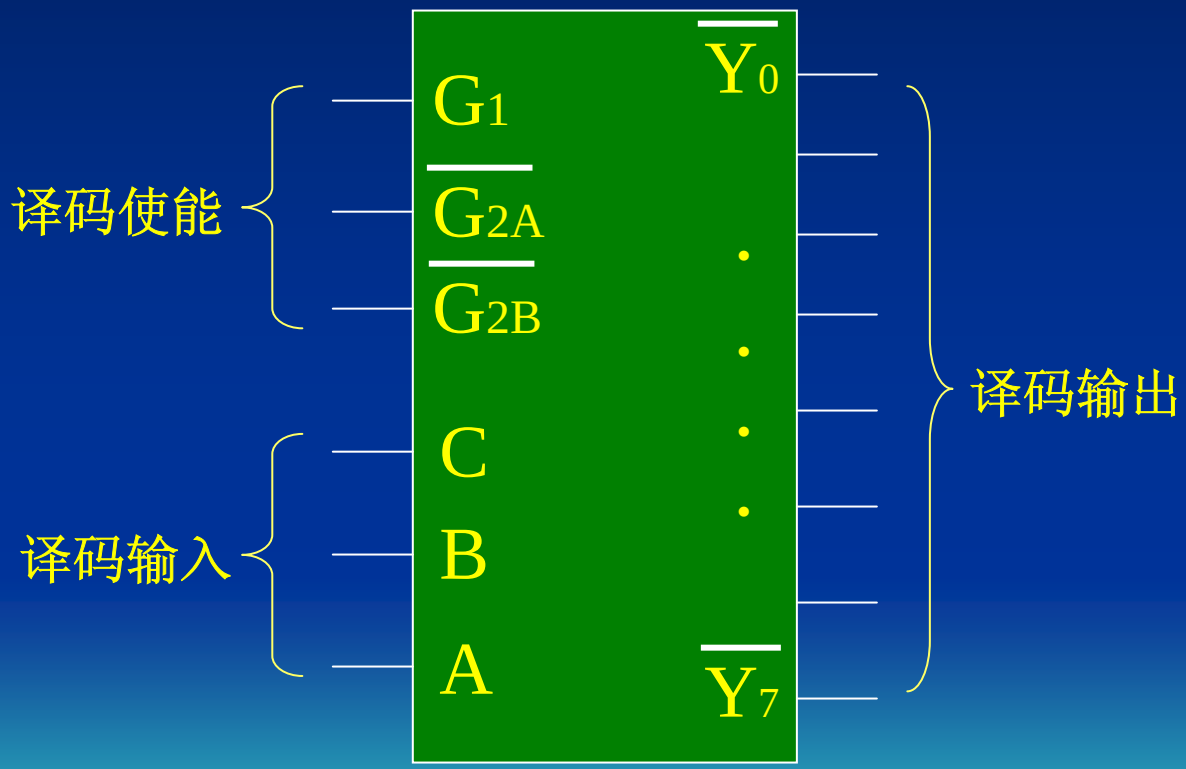


$Y = A \oplus B$

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

译码器 (Code Translator)

- 例： 74LS138译码器：



74LS138真值表

使能端			输入端			输出端							
G_1	$\#G_{2A}$	$\#G_{2B}$	C	B	A	$\#Y_0$	$\#Y_1$	$\#Y_2$	$\#Y_3$	$\#Y_4$	$\#Y_5$	$\#Y_6$	$\#Y_7$
×	0	1	×	×	×	1	1	1	1	1	1	1	1
×	1	0	×	×	×	1	1	1	1	1	1	1	1
×	1	1	×	×	×	1	1	1	1	1	1	1	1
0	×	×	×	×	×	1	1	1	1	1	1	1	1
1	0	0	0	0	0	0	1	1	1	1	1	1	1
1	0	0	0	0	1	1	0	1	1	1	1	1	1
1	0	0	0	1	0	1	1	0	1	1	1	1	1
1	0	0	0	1	1	1	1	1	0	1	1	1	1
1	0	0	1	0	0	1	1	1	1	0	1	1	1
1	0	0	1	0	1	1	1	1	1	1	0	1	1
1	0	0	1	1	0	1	1	1	1	1	1	0	1
1	0	0	1	1	1	1	1	1	1	1	1	1	0

Signed Number Operations

- 有符号数在计算机中的表示
  - 把二进制数的最高位定义为符号位
    - 符号位为 *0* 表示正数，符号位为 *1* 表示负数
  - 连同符号位一起数值化了的数，称为机器数。
  - 机器数所表示的真实的数值，称为真值。

（在以下讲述中，均以 8 位二进制数为例）

Signed Number Operations

Example: 真值和机器数

真值　　机器数

+52 = +0110100 = 0　0110100

符号位　数值位

-52 = -0110100 = 1　0110100

Signed Number Operations

- 有符号数在计算机中的表示

  对于符号数，机器数常用的表示方法有原码、反码和补码三种。数X的原码记作$[X]_{原}$，反码记作$[X]_{反}$，补码记作$[X]_{补}$。

注意：对正数，三种表示法均相同。

它们的差别在于对负数的表示。

•原码[X]原

- 定义

  符号位：0表示正，1表示负；

  数值位：真值的绝对值。

$$X=\begin{cases} X & 2^{n-1}>X\geqslant 0 \\ 2^{n-1}+|X| & 0\geqslant X>-2^{n-1} \end{cases}$$

原码的例子

n位原码表示数值的范围是

$$-(2^{n-1}-1) \sim +(2^{n-1}-1)$$

对应的原码是**111…1**～ **011…1**。

数0的原码

- 8位数0的原码： +0 = 0 0000000
  - 0 = 1 0000000

即：数0的原码不唯一。

•反码$[X]_{反}$

定义

- 若X>0，则 $[X]_{反}=[X]_{原}$
- 若X<0，则 $[X]_{反}$= 对应原码的符号位不变，数值部分按位求反

$$X=\begin{cases} X & 2^{n-1}>X\geqslant 0 \\ (2^n-1)+X & 0\geqslant X>-2^{n-1} \end{cases}$$

[例]:

- X= - 52 = -0110100

  $[X]_{原}$ = 10110100

  $[X]_{反}$ = 11001011

反码的例子

n位反码表示数值的范围是

$$-(2^{n-1}-1) \sim +(2^{n-1}-1)$$

对应的反码是**100…0**～ **011…1**。

0的反码：

$[+0]_{反}$ = 00000000

$[-0]_{反}$ = 11111111

即：数0的反码也不是唯一的。

•补码

定义：

- 若X>0，　则$[X]_{补}=[X]_{反}=[X]_{原}$
- 若X<0，　则$[X]_{补}=[X]_{反}+1$

$$X=\begin{cases} X & 2^{n-1}>X\geqslant 0 \\ 2^{n}-|X| & 0>X\geqslant -2^{n-1} \end{cases}$$

[例]:

- X= –52= – 0110100

  $[X]_{原}$ = 10110100

  $[X]_{反}$ = 11001011

  $[X]_{补}$ = $[X]_{反}$+1=11001100

n位补码表示数值的范围是

$$-2^{n-1} \sim +(2^{n-1}-1)$$

对应的补码是100…0～ 011…1。

0的补码：

- $[+0]_{补}= [+0]_{原}=00000000$
- $[-0]_{补}= [-0]_{反}+1=11111111+1$

$=1\ \ 00000000$

对8位字长，进位被舍掉

- $\therefore [+0]_{补}= [-0]_{补}= 00000000$

•特殊数10000000

- 该数在原码中定义为： -0
- 在反码中定义为： -127
- 在补码中定义为： -128
- 对无符号数：$(10000000)_2 = 128$

Signed Number Operations

- 8位二进制有符号数的表示范围(表示范围问题):
  - 原码: -127 ~ +127
  - 反码: -127 ~ +127
  - 补码: -128 ~ +127

- *思考：16位有符号数的表示范围是多少?*

Signed Number Operations

- 有符号二进制数与十进制的转换

  对用补码表示的二进制数：

  1）求出真值

  2）进行转换

思考：如何计算一个十进制数(包含整数和小数部分)的二进制补码表示?

Signed Number Operations

- **Example**：将一个用补码表示的二进制数转换为十进制数。

1) $[X]_{补} = 0\ 0101110B$　　真值为：+0101110B

正数

所以：X=+46

2) $[X]_{补} = 1\ 1010010B$

负数

$X = [[X]_{补}]_{补} = [11010010]_{补}$

$= -\ 0101110B$

所以：X = - 46

Signed Number Operations

- 补码加减法的运算规则

  通过引进补码，可将减法运算转换为加法运算。规则如下：

$$[X+Y]_{补}=[X]_{补}+[Y]_{补}$$

$$[X-Y]_{补}=[X]_{补}-[Y]_{补}$$

  其中X，Y为正负数均可，符号位参与运算，不需要特别单独处理。

Signed Number Operations

- 补码的运算原理

模（**module** ） 就是一个计数系统的最大容量，其大小等于以进位计数制基数为底，以位数为指数的幂。凡是用器件进行的运算都是有模运算，运算结果超过模的部分被运算器自动丢弃。因此，当器件为n位时，有

$$X=2^n+X \pmod{2^n}$$

不难验证，

$$[X]_{补}=2^n+X \pmod{2^n}$$

因此，

$$[X\pm Y]_{补}= 2^n+ (X\pm Y) \pmod{2^n}$$

$$= (2^n+ X)+ (2^n+ X) \pmod{2n}$$

$$= [X]_{补}+ [\pm Y]_{补}$$

Signed Number Operations

- **Example:** 基于补码表示的计算

  X=-0110100，Y=+1110100，求$[X+Y]_{补}$

  $[X]_{原}$=10110100

  $[X]_{补}$= $[X]_{反}$+1=11001100

  $[Y]_{补}$= $[Y]_{原}$=01110100

  所以： $[X+Y]_{补}$= $[X]_{补}$+ $[Y]_{补}$

  =11001100+01110100

  =01000000

Signed Number Operations

- 符号数运算中的溢出问题
  - 进(借)位
    - 在加法过程中，符号位向更高位产生进位；
    - 在减法过程中，符号位向更高位产生借位。
  - 溢出
    - 运算结果超出运算器所能表示的范围。

Signed Number Operations

- 溢出的判断方法
- 方法 1：
  - 同号相减或异号相加——不会溢出。
  - 同号相加或异号相减——可能溢出：
    - 两种情况：

      同号相加时，结果符号与加数符号相反——溢出；

      异号相减时，结果符号与减数符号相同——溢出。
- 方法 2：
  - 两个带符号二进制数相加或相减时，若

    $C_7 \oplus C_6 = 1$，

    则结果产生溢出。

    C_7为最高位的进(借)位；C_6为次高位的进(借)位。

Signed Number Operations

- 有符号数运算，有溢出表示结果是错误的
- 无符号数运算，有进位表示结果是错误的

CASE1:

```
  10110101
+ 10001111
-----------
 101000100
```

CASE2:

```
  01000010
+ 01100011
-----------
  10100101
```

CASE3:

```
  01000010
+ 11001101
-----------
 100001111
```

Floating Numbers

$N=M*R^J$

N----二进制数　　M----尾数（符号位和数值位组成）

R----基数　　J----阶码（符号位和数值位组成）

阶码符号	阶码数值	尾数符号	尾数数值

IEEE规定：

32位时	1	7	1	23
64位时	1	10	1	52

Example

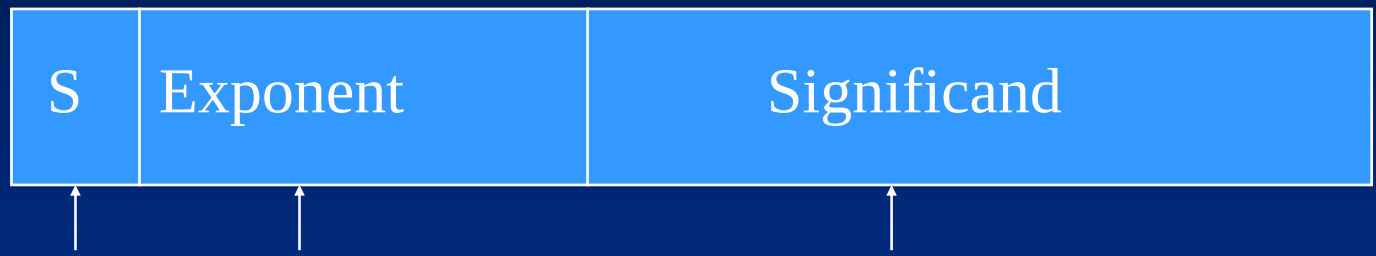
- 设阶码为4位，用补码形式表示；尾数为8位，用原码形式表示。写出数 0.0001010001B的浮点数形式。
- 解：（1）将数规格化：|M|>=1/2

  $0.0001010001B=0.1010001B*2^{-3}$

  （2）$[-3]_{补}=2^4-3=13=1101B$

  这个数的浮点制形式：1101 0 1010001

IEEE-745 standard



single-precision 1 8(移码 biased) 23(原码) 共32位

double-precision 1位 11位 52位 共64位

数J的移码=偏移量+ |J|

单精度实数，偏移量为127或1111111B，或7FH

双精度实数，偏移量为1023或1111111111B，或3FFH

尾数标准化 2> |尾数| >= 1，尾数部分不包括整数1，这个1是隐含（implied,hidden)的。

Example

- 解：12=1100B=1.1*2^3

  符号位 S=0（正数）

  指数E=7FH+3=82H=10000010B

  隐含 整数部分的1

  尾数部分M=100 0000 0000 0000 0000 0000B

  合成后的4字节数是：

  0100 0001 0100 0000 0000 0000 0000 0000B

十进制数的表示——BCD码

- 十进制数的表示——BCD码
  - 用4位二进制数表示一位十进制数。有两种表示法：压缩BCD码和非压缩BCD码。
  - 压缩BCD码的每一位用4位二进制表示，0000~1001表示0~9，一个字节表示两位十进制数。
  - 非压缩BCD码用一个字节表示一位十进制数，高4位总是0000，低4位的0000~1001表示0~9。

非数值数据的表示

- 计算机中除了能够处理数值数据以外，还可以处理文字、语音、图像等各种信息，这些信息统称为非数值数据。

- 非数值数据在计算机中也必须以二进制形式表示，非数值数据的表示本质上是编码的过程。

ASCII码—美国标准信息交换代码

Decimal	Octal	Hex	Binary	Value	
000	000	000	00000000	NUL	(Null char.)
001	001	001	00000001	SOH	(Start of Header)
002	002	002	00000010	STX	(Start of Text)
003	003	003	00000011	ETX	(End of Text)
004	004	004	00000100	EOT	(End of Transmission)
005	005	005	00000101	ENQ	(Enquiry)
006	006	006	00000110	ACK	(Acknowledgment)
007	007	007	00000111	BEL	(Bell)
008	010	008	00001000	BS	(Backspace)
009	011	009	00001001	HT	(Horizontal Tab)
010	012	00A	00001010	LF	(Line Feed)
011	013	00B	00001011	VT	(Vertical Tab)
012	014	00C	00001100	FF	(Form Feed)
013	015	00D	00001101	CR	(Carriage Return)
014	016	00E	00001110	SO	(Shift Out)
015	017	00F	00001111	SI	(Shift In)
016	020	010	00010000	DLE	(Data Link Escape)
017	021	011	00010001	DC1 (XON)	(Device Control 1)
018	022	012	00010010	DC2	(Device Control 2)
019	023	013	00010011	DC3 (XOFF)	(Device Control 3)
020	024	014	00010100	DC4	(Device Control 4)
021	025	015	00010101	NAK	(Negative Acknowledgement)
022	026	016	00010110	SYN	(Synchronous Idle)
023	027	017	00010111	ETB	(End of Trans. Block)
024	030	018	00011000	CAN	(Cancel)
025	031	019	00011001	EM	(End of Medium)

ASCII码

- 采用7位二进制代码对字符进行编码
- 数字0~9的编码是0110000~0111001，它们的高3位均是011，后4位正好与其对应的二进制代码（BCD码）相符。
- 英文字母A~Z的ASCII码从1000001（41H）开始顺序递增，字母a~z的ASCII码从1100001（61H）开始顺序递增，这样的排列对信息检索十分有利。
- 最高位通常总为0，有时也用作奇偶校验位。

Appendix 计算机中常用术语

- bit

  1Mb=1024×1024bit=2^{20}bit

  1Gb=2^{30}bit=1024Mb

  1Tb=2^{40}bit=1024Gb

- Byte

  1 Byte=8bit，1KB=1024 Byte

- Word：表示字长，有1bit,4bit,8bit,16bit等

  一般情况下为2Byte(16bit)

Summary

- 基本概念：机器数和真值，原码反码和补码表示，浮点数
- 基本原理：
  - 数的进位计数制表示及不同进制间的相互转化
  - 数在计算机内部的表示（原码反码和补码）以及有关计算
  - 浮点数德表示和有关计算