

第2章

计算机组成原理

数据的机器层次表示

2025年9月

北京理工大学计算机学院

第2章

计算机组成原理

数据是计算机加工和处理的对象，数据的机器层次表示将直接影响到计算机的结构和性能。本章主要介绍无符号数和带符号数的表示方法、数的定点与浮点表示方法、字符和汉字的编码方法、数据校验码等。熟悉和掌握本章的内容，是学习计算机原理的最基本要求。

2025年9月

北京理工大学计算机学院

第2章

计算机组成原理

2.1 数值数据的表示

2.2 机器数的定点表示与浮点表示

2.3 非数值数据的表示

2.4 十进制数和数串的表示

2.5 不同类型的数据表示举例

2.6 数据校验码

2025年9月

北京理工大学计算机学院

2.1 数值数据的表示

计算机组成原理

2.1.1 计算机中的数值数据

在计算机中常用后缀字母来表示不同的数制。

十进制数 (D)

二进制数 (B)

八进制数 (Q)

十六进制数 (H)

在C语言中，八进制常数以前缀0开始，十六进制常数以前缀0x开始。

2025年9月

北京理工大学计算机学院

2.1 数值数据的表示

计算机组成原理

R进制

- 基数：R
- 符号：0, 1, ..., r-1
- 计算规律：逢R进一或借一当R
- R进制的多项式表示：
 - $N_r = d_{n-1} \times R^{n-1} + d_{n-2} \times R^{n-2} + \dots + d_1 \times R^1 + d_0 \times R^0 + d_{-1} \times R^{-1} + d_{-2} \times R^{-2} + \dots + d_{-m} \times R^{-m}$
 - m, n为正整数，其中n为整数位数；m为小数位数。Di表示第i位的系数， R^i 称为该位的权。

2025年9月

北京理工大学计算机学院

2.1 数值数据的表示

计算机组成原理

2.1.2 无符号数和带符号数

所谓无符号数，就是整个机器字长的全部二进制位均表示数值位（没有符号位），相当于数的绝对值。

$N_1 = 01001$

表示无符号数9

$N_2 = 11001$

表示无符号数25

2025年9月

北京理工大学计算机学院

2.1 数值数据的表示

计算机组成原理

对于字长为 $n+1$ 位的无符号数的表示范
 例如：字长为8位，无符号数的表示范围是0~255。

00000000 11111111

2025年9月

北京理工大学计算机学院

2.1 数值数据的表示

计算机组成原理

所谓带符号数，即正、负数。在日常生活中，我们用“+”、“-”号加绝对值来表示数值的大小，用这种形式表示的数值在计算机技术中称为“真值”。

对于数的符号“+”或“-”，计算机是无法识别的，因此需要把数的符号数码化。通常，约定二进制数的最高位为符号位，“0”表示正号，“1”表示负号。这种在计算机中使用的表示数的形式称为机器数。

2025年9月

北京理工大学计算机学院

2.1 数值数据的表示

计算机组成原理

对于带符号数，最高位用来表示符号位，而不再表示数值位了，前例中的 N_1 、 N_2 在这里变为：

$N_1 = 01001$

表示带符号数+9

$N_2 = 11001$

根据不同的机器数表示不同的值，如：

原码时表示带符号数-9，

补码则表示-7，

反码则表示-6。

2025年9月

北京理工大学计算机学院

2.1 数值数据的表示

计算机组成原理

2.1.3 原码表示法

原码表示法是一种最简单的机器数表示法，用最高位表示符号位，符号位为“0”表示该数为正，符号位为“1”表示该数为负，数值部分与真值相同。

若真值为纯小数，它的原码形式为 $X_s.X_1X_2...X_n$ ，其中 X_s 表示符号位。

例1: $X_1 = 0.0110$, $X_2 = -0.0110$

$[X_1]_{\text{原}} = 0.0110$, $[X_2]_{\text{原}} = 1.0110$

2025年9月

北京理工大学计算机学院

2.1 数值数据的表示

计算机组成原理

若真值为纯整数，它的原码形式为 $X_sX_1X_2...X_n$ ，其中 X_s 表示符号位。

例2: $X_1 = 1101$, $X_2 = -1101$

$[X_1]_{\text{原}} = 0,1101$, $[X_2]_{\text{原}} = 1,1101$

在原码表示中，真值0有两种不同的表示形式：

$[+0]_{\text{原}} = 00000$

$[-0]_{\text{原}} = 10000$

注意

2025年9月

北京理工大学计算机学院

2.1 数值数据的表示

计算机组成原理

原码表示法的优点是直观易懂，机器数和真值间的相互转换很容易，用原码实现乘、除运算的规则很简单；缺点是实现加、减运算的规则较复杂。

2025年9月

北京理工大学计算机学院

2.1 数值数据的表示

计算机组成原理

2.1.4 补码表示法

为了克服原码在加、减运算中的缺点，引入了补码表示法，补码表示法的设想是：使符号位参加运算，从而简化加减法的规则；使减法运算转化成加法运算，从而简化机器的运算器电路。

1. 模和同余

由于设备的原因，机器数是有字长限制的，不可能容纳无限大的任意数。当运算结果超出了机器的最大表示范围，就会发生溢出（丢失进位），此时所产生的溢出量称为模，用字母M表示。

2025年9月

北京理工大学计算机学院

2.1 数值数据的表示

计算机组成原理

模实际上是一个计量器的容量。例如：一个4位的计数器，它的计数值为0~15，当计数器计满15之后再加1，这个计数器就发生溢出，其溢出量为16，也就是模等于16。

$2^4 \ 2^3 \ 2^2 \ 2^1 \ 2^0$
1 0000
↑
丢失

一个字长为n+1位的纯整数的溢出量为 2^{n+1} ，即以 2^{n+1} 为模。

一个纯小数的溢出量为2，即以2为模。

2025年9月

北京理工大学计算机学院

2.1 数值数据的表示

计算机组成原理

$2^4 \ 2^3 \ 2^2 \ 2^1 \ 2^0$
1 0000
↑
丢失

2025年9月

北京理工大学计算机学院

2.1 数值数据的表示

计算机组成原理

同余概念：即两整数A、B除以同一正整数M，所得余数相同，则称A、B对模M同余。

$A=B \pmod{M}$ ，如 $23=13 \pmod{10}$

对钟表而言，M=12。假设：时钟停在8点，而现在正确的时间是6点，这时拨准时钟的方法有两种：

倒拨

正拨

分针倒着旋转2圈，等于分针正着旋转10圈。故有： $-2=10 \pmod{12}$ ，即-2和10同余。

2025年9月

北京理工大学计算机学院

2.1 数值数据的表示

计算机组成原理

$$8-2=8+10 \pmod{12}$$

可见，只要确定了“模”，就可找到一个与负数等价的正数（该正数即为负数的补数）来代替此负数，而这个正数可以用模加上负数本身求得，这样就可把减法运算用加法实现了。

$$9-5=9+(-5)=9+(12-5)=9+7=4 \pmod{12}$$

$$65-25=65+(-25)=65+(100-25)=65+75=40 \pmod{100}$$

将补数的概念用到计算机中，便出现了补码这种机器数。

2025年9月

北京理工大学计算机学院

2.1 数值数据的表示

计算机组成原理

2. 补码表示

补码的符号位表示方法与原码相同，其数值部分的表示与数的符号有关：对于正数，数值部分与真值形式相同；对于负数，其数值部分为真值形式按位取反，且在最低位加1。

若真值为纯小数，它的补码形式为 $X_s.X_1X_2\dots X_n$ ，其中 X_s 表示符号位。

例1: $X_1=0.0110$, $X_2=-0.0110$

$[X_1]_{\text{补}}=0.0110$, $[X_2]_{\text{补}}=1.1010$

2025年9月

北京理工大学计算机学院

2.1 数值数据的表示

计算机组成原理

若真值为纯整数，它的补码形式为 $X_s X_1 X_2 \dots X_n$ ，其中 X_s 表示符号位。

例2: $X_1=1101$, $X_2=-1101$

$[X_1]_{\text{补}}=0,1101$, $[X_2]_{\text{补}}=1,0011$

在补码表示中，真值0的表示形式是唯一的。

$[+0]_{\text{补}}=[-0]_{\text{补}}=00000$

注意

2025年9月

北京理工大学计算机学院

2.1 数值数据的表示

计算机组成原理

3. 由真值、原码转换为补码

当X为正数时， $[X]_{\text{补}}=[X]_{\text{原}}=X$ 。

当X为负数时，由 $[X]_{\text{原}}$ 转换为 $[X]_{\text{补}}$ 的方法：

① $[X]_{\text{原}}$ 除掉符号位外的各位取反加“1”。

② 自低位向高位，尾数的第一个“1”及其右部的“0”保持不变，左部的各位取反，符号位保持不变。

例如： $[X]_{\text{原}}=1.1110011000$

$[X]_{\text{补}}=1.\underline{000110}1000$

不变 变反 不变
北京理工大学计算机学院

2025年9月

2.1 数值数据的表示

计算机组成原理

2.1.5 反码表示法

反码的符号位表示方法与原码相同，但其数值部分的表示与数的符号有关：对于正数，数值部分与真值形式相同；对于负数，数值部分为真值形式按位取反。

若真值为纯小数，它的反码形式为 $X_s.X_1 X_2 \dots X_n$ ，其中 X_s 表示符号位。

例1: $X_1=0.0110$, $X_2=-0.0110$

$[X_1]_{\text{反}}=0.0110$, $[X_2]_{\text{反}}=1.1001$

2025年9月

北京理工大学计算机学院

2.1 数值数据的表示

计算机组成原理

若真值为纯整数，它的反码形式为 $X_s X_1 X_2 \dots X_n$ ，其中 X_s 表示符号位。

例2: $X_1=1101$, $X_2=-1101$

$[X_1]_{\text{反}}=0,1101$, $[X_2]_{\text{反}}=1,0010$

在反码表示中，真值0也有两种不同的表示形式：

$[+0]_{\text{反}}=00000$

$[-0]_{\text{反}}=11111$

注意

2025年9月

北京理工大学计算机学院

2.1 数值数据的表示

计算机组成原理

2.1.6 3种机器数的比较

(1) 对于正数它们都等于真值本身，而对于负数各有不同的表示。

(2) 最高位都表示符号位，补码和反码的符号位可和数值位一起参加运算；但原码的符号位必须分开进行处理。

2025年9月

北京理工大学计算机学院

2.1 数值数据的表示

计算机组成原理

(3) 对于真值0，原码和反码各有两种不同的表示形式，而补码只有唯一的一种表示形式。

(4) 原码、反码表示的正、负数范围是对称的；但补码负数能多表示一个最负的数（绝对值最大的负数），其值等于 -2^n （纯整数）或 -1 （纯小数）。

注意

2025年9月

北京理工大学计算机学院

2.1 数值数据的表示

计算机组成原理

设机器字长4位（含1位符号位），以纯整数为例：
原码或反码可表示的数

补码可表示的数（多表示一个负数）

2025年9月

北京理工大学计算机学院

2.1 数值数据的表示

计算机组成原理

真值与3种机器数间的对照

| 真值 X | | [X] _原 | [X] _补 | [X] _反 | 真值 X | | [X] _原 | [X] _补 | [X] _反 |
|------|------|------------------|------------------|------------------|------|------|------------------|------------------|------------------|
| 十进制 | 二进制 | | | | 十进制 | 二进制 | | | |
| +0 | +000 | 0000 | -0 | -000 | 1000 | 0000 | 1111 | | |
| +1 | +001 | 0001 | -1 | -001 | 1001 | 1111 | 1110 | | |
| +2 | +010 | 0010 | -2 | -010 | 1010 | 1110 | 1101 | | |
| +3 | +011 | 0011 | -3 | -011 | 1011 | 1101 | 1100 | | |
| +4 | +100 | 0100 | -4 | -100 | 1100 | 1100 | 1011 | | |
| +5 | +101 | 0101 | -5 | -101 | 1101 | 1011 | 1010 | | |
| +6 | +110 | 0110 | -6 | -110 | 1110 | 1010 | 1001 | | |
| +7 | +111 | 0111 | -7 | -111 | 1111 | 1001 | 1000 | | |
| +8 | - | - | -8 | -1000 | - | 1000 | - | | |

2025年9月

北京理工大学计算机学院

2.1 数值数据的表示

计算机组成原理

如果已知机器的字长，则机器数的位数应补够相应的位。设机器字长为8位，则：

$X_1=1011$ $X_2=-1011$
 $[X_1]_{原}=0,0001011$ $[X_2]_{原}=1,0001011$
 $[X_1]_{补}=0,0001011$ $[X_2]_{补}=1,1110101$
 $[X_1]_{反}=0,0001011$ $[X_2]_{反}=1,1110100$
 $X_1=0.1011$ $X_2=-0.1011$
 $[X_1]_{原}=0.1011000$ $[X_2]_{原}=1.1011000$
 $[X_1]_{补}=0.1011000$ $[X_2]_{补}=1.0101000$
 $[X_1]_{反}=0.1011000$ $[X_2]_{反}=1.0100111$

2025年9月

北京理工大学计算机学院

第2章

计算机组成原理

2.1 数值数据的表示

2.2 机器数的定点表示与浮点表示

2.3 非数值数据的表示

2.4 十进制数和数串 的表示

2.5 不同类型的数据表示举例

2.6 数据校验码

2025年9月

北京理工大学计算机学院

2.2 机器数的定点表示与浮点表示

计算机组成原理

2.2.1 定点表示法

在定点表示法中约定：所有数据的小数点位置固定不变。通常，把小数点固定在有效数位的最前面或末尾，这就形成了两类定点数。

1. 定点小数

小数点的位置固定在最高有效数位之前，符号位之后，记作 $X_s.X_1X_2...X_n$ ，这个数是一个纯小数。定点小数的小数点位置是隐含约定的，小数点并不需要真正地占据一个二进制位。

2025年9月

北京理工大学计算机学院

2.2 机器数的定点表示与浮点表示

计算机组成原理

定点小数格式

当 $X_s=0$ ， $X_1 \sim X_n=1$ 时，X 为最大正数，即：
 $X_{最大正数}=(1-2^{-n})$ 。

注意

2025年9月

北京理工大学计算机学院

2.2 机器数的定点表示与浮点表示

计算机组成原理

定点小数格式

2^0

2^{-1}

2^{-2}

...

$2^{-(n-1)}$

2^{-n}

X_s

X_1

X_2

...

X_{n-1}

X_n

最小正数

2^0

2^{-1}

2^{-2}

...

$2^{-(n-1)}$

2^{-n}

0

0

0

...

0

1

当 $X_s=1$, $X_s \sim X_{n-1}=0$ 时, X为最小正数,
即: $X_{\text{最小正数}}=2^{-n}$.

注意

2025年9月

北京理工大学计算机学院

2.2 机器数的定点表示与浮点表示

计算机组成原理

当 $X_s=1$, 表示X为负数, 此时情况要稍微复杂一些, 这是因为在计算机中带符号数可用补码表示, 也可用原码表示。如前所述, 原码与补码所能表示的绝对值最大的负数是有区别的, 所以原码和补码的表示范围有一些差别。

2025年9月

北京理工大学计算机学院

2.2 机器数的定点表示与浮点表示

计算机组成原理

原码表示的绝对值最大负数

2^0

2^{-1}

2^{-2}

...

$2^{-(n-1)}$

2^{-n}

1

1

1

...

1

1

$X_{\text{绝对值最大负数(原码表示时)}}=-(1-2^{-n})$

补码表示的绝对值最大负数

2^0

2^{-1}

2^{-2}

...

$2^{-(n-1)}$

2^{-n}

1

0

0

...

0

0

$X_{\text{绝对值最大负数(补码表示时)}}=-1$

注意

2025年9月

北京理工大学计算机学院

2.2 机器数的定点表示与浮点表示

计算机组成原理

综上所述:
若机器字长有 $n+1$ 位, 则:
原码定点小数表示范围: $1.1111111 \sim 0.1111111$
补码定点小数表示范围: $-1 \sim (1-2^{-n})$
若机器字长有8位, 则:
原码定点小数表示范围: $-(1-2^{-7}) \sim (1-2^{-7})$
补码定点小数表示范围: $-1 \sim (1-2^{-7})$

1.0000000

0.1111111

2025年9月

北京理工大学计算机学院

2.2 机器数的定点表示与浮点表示

计算机组成原理

2.定点整数

小数点位置隐含固定在最低有效数位之后, 记作 $X_s X_1 X_2 \dots X_n$, 这个数是一个纯整数。

定点整数格式

2^n

2^{n-1}

2^{n-2}

...

2^0

X_s

X_1

X_2

...

X_n

最大正数

2^n

2^{n-1}

2^{n-2}

...

2^1

2^0

0

1

1

...

1

1

$X_{\text{最大正数}}=(2^n-1)$

注意

2025年9月

北京理工大学计算机学院

2.2 机器数的定点表示与浮点表示

计算机组成原理

注意

最小正数

2^n

2^{n-1}

2^{n-2}

...

2^1

2^0

0

0

0

...

0

1

$X_{\text{最小正数}}=1$

原码表示的绝对值最大负数

2^n

2^{n-1}

2^{n-2}

...

2^1

2^0

1

1

1

...

1

1

$X_{\text{绝对值最大负数(原码表示时)}}=-(2^n-1)$

补码表示的绝对值最大负数

2^n

2^{n-1}

2^{n-2}

...

2^1

2^0

1

0

0

...

0

0

$X_{\text{绝对值最大负数(补码表示时)}}=-2^n$

注意

2025年9月

北京理工大学计算机学院

2.2 机器数的定点表示与浮点表示

计算机组成原理

综上所述：
若机器字长有 $n+1$ 位，则：
原码定点整数的表示范围为： $-(2^n-1) \sim (2^n-1)$
补码定点整数的表示范围为： $-2^n \sim (2^n-1)$
若机器字长有8位，则：
原码定点整数表示范围为： $-127 \sim 127$
补码定点整数表示范围为： $-128 \sim 127$

10000000

01111111

01111111

2025年9月

北京理工大学计算机学院

2.2 机器数的定点表示与浮点表示

计算机组成原理

2.2.2 浮点表示法

小数点的位置根据需要而浮动，这就是浮点数。例如：
$$N = M \times r^E = M \times 2^E$$

式中： r 为浮点数阶码的底，与尾数的基数相同，通常 $r=2$ 。 E 和 M 都是带符号数， E 叫做阶码， M 叫做尾数。在大多数计算机中，尾数为纯小数，常用原码或补码表示；阶码为纯整数，常用移码或补码表示。

2025年9月

北京理工大学计算机学院

2.2 机器数的定点表示与浮点表示

计算机组成原理

浮点数的一般格式：

| | | | |
|-------|-----|-------|-----|
| 1位 | K位 | 1位 | N位 |
| e_s | e | m_s | m |
| 阶码部分E | | 尾数部分M | |

浮点数的底是隐含的，在整个机器数中不出现。阶码的符号位为 e_s ，阶码的大小反映了在数 N 中小数点的实际位置；尾数的符号位为 m_s ，它是整个浮点数的符号位，反映了该浮点数的正负。
假设阶码和尾数部分均用补码表示。

2025年9月

北京理工大学计算机学院

2.2 机器数的定点表示与浮点表示

计算机组成原理

1. 浮点数的表示范围

当 $e_s=0, m_s=0$ ，阶码和尾数的数值位各位全为1（即阶码和尾数都为最大正数）时，该浮点数为最大正数。

| | | | | | | | | |
|-------|-----------|-----|-------|-------|----------|-----|--------------|----------|
| e_s | 2^{k-1} | ... | 2^0 | m_s | 2^{-1} | ... | $2^{-(n-1)}$ | 2^{-n} |
| 0 | 1 | ... | 1 | 0 | 1 | ... | 1 | 1 |
| 阶码部分E | | | | | 尾数部分M | | | |

$X_{\text{最大正数}} = (1-2^{-n}) \times 2^{2^k-1}$

注意

2025年9月

北京理工大学计算机学院

2.2 机器数的定点表示与浮点表示

计算机组成原理

当 $e_s=1, m_s=0$ ，尾数的最低位 $m_n=1$ ，其余各位为0（即阶码为绝对值最大负数，尾数为最小正数）时，该浮点数为最小正数。

| | | | | | | | | |
|-------|-----------|-----|-------|-------|----------|-----|--------------|----------|
| e_s | 2^{k-1} | ... | 2^0 | m_s | 2^{-1} | ... | $2^{-(n-1)}$ | 2^{-n} |
| 1 | 0 | ... | 0 | 0 | 0 | ... | 0 | 1 |
| 阶码部分E | | | | | 尾数部分M | | | |

$X_{\text{最小正数}} = 2^{-n} \times 2^{-2^k}$

注意

2025年9月

北京理工大学计算机学院

2.2 机器数的定点表示与浮点表示

计算机组成原理

当 $e_s=0, m_s=1$ ，尾数的数值位为全0（即阶码为最大正数，尾数为绝对值最大的负数）时，该浮点数为绝对值最大负数。

| | | | | | | | | |
|-------|-----------|-----|-------|-------|----------|-----|--------------|----------|
| e_s | 2^{k-1} | ... | 2^0 | m_s | 2^{-1} | ... | $2^{-(n-1)}$ | 2^{-n} |
| 0 | 1 | ... | 1 | 1 | 0 | ... | 0 | 0 |
| 阶码部分E | | | | | 尾数部分M | | | |

$X_{\text{绝对值最大负数}} = -1 \times 2^{2^k-1}$

注意

2025年9月

北京理工大学计算机学院

2.2 机器数的定点表示与浮点表示

计算机组成原理

2.规格化的浮点数

为了提高运算的精度，需要充分地利用尾数的有效数位，通常采取规格化的浮点数形式，即规定尾数的最高数位必须是一个有效值。

$$1/r \leq |M| < 1$$

如果 $r=2$ ，则有 $1/2 \leq |M| < 1$ 。

2025年9月

北京理工大学计算机学院

2.2 机器数的定点表示与浮点表示

计算机组成原理

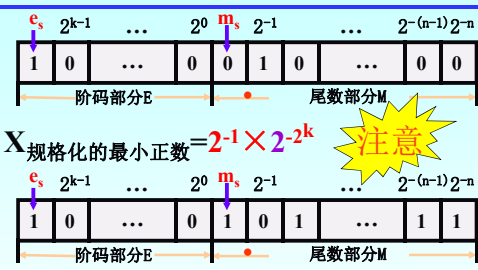
在尾数用原码表示时，规格化浮点数的尾数的最高数位总等于1。在尾数用补码表示时，规格化浮点数应满足尾数最高数位与符号位不同（ $m_s \oplus m_1 = 1$ ），即当 $1/2 \leq M < 1$ 时，应有 $0.1xx...x$ 形式，当 $-1 \leq M < -1/2$ 时，应有 $1.0xx...x$ 形式。需要注意的是当 $M=-1/2$ ，对于原码来说，是规格化数，而对于补码来说，不是规格化数；当 $M=-1$ 时，对于原码来说，这将无法表示，而对于补码来说，这是一个规格化数。

2025年9月

北京理工大学计算机学院

2.2 机器数的定点表示与浮点表示

计算机组成原理



2025年9月

北京理工大学计算机学院

2.2 机器数的定点表示与浮点表示

计算机组成原理

| | 浮点数代码 | | 真值 |
|-------------|--------|-----------|--|
| | 阶码 | 尾数 | |
| 最大正数 | 01...1 | 0.11...11 | $(1-2^{-n}) \times 2^{2^k-1}$ |
| 绝对值最大负数 | 01...1 | 1.00...00 | $-1 \times 2^{2^k-1}$ |
| 最小正数 | 10...0 | 0.00...01 | $2^{-n} \times 2^{-2^k}$ |
| 规格化的最小正数 | 10...0 | 0.10...00 | $2^{-1} \times 2^{-2^k}$ |
| 绝对值最小负数 | 10...0 | 1.11...11 | $-2^{-n} \times 2^{-2^k}$ |
| 规格化的绝对值最小负数 | 10...0 | 1.01...11 | $-(2^{-1} \cdot 2^{-n}) \times 2^{-2^k}$ |

2025年9月

北京理工大学计算机学院

2.2 机器数的定点表示与浮点表示

计算机组成原理

2.2.3 浮点数阶码的移码表示法

移码就是在真值X上加一个常数（偏置值），相当于X在数轴上向正方向平移了一段距离，这就是“移码”一词的来由，移码也可称为增码或偏码。

$$[X]_{\text{移}} = \text{偏置值} + X$$

字长 $n+1$ 位定点整数的移码形式为 $X_0X_1X_2...X_n$ 。

2025年9月

北京理工大学计算机学院

2.2 机器数的定点表示与浮点表示

计算机组成原理

最常见的移码的偏置值为 2^n 。当字长8位时，偏置值为 2^7 。

例1: $X=1011101$

$[X]_{\text{移}} = 2^7 + X = 10000000 + 1011101 = 11011101$

$[X]_{\text{补}} = 01011101$

例2: $X=-1011101$

$[X]_{\text{移}} = 2^7 + X = 10000000 - 1011101 = 00100011$

$[X]_{\text{补}} = 10100011$

2025年9月

北京理工大学计算机学院

2.2 机器数的定点表示与浮点表示

计算机组成原理

| 真值X (十进制) | 真值X (二进制) | $[X]_{\text{补}}$ | $[X]_{\text{移}}$ |
|-----------|-----------|------------------|------------------|
| -128 | -10000000 | 10000000 | 00000000 |
| -127 | -11111111 | 10000001 | 00000001 |
| ⋮ | ⋮ | ⋮ | ⋮ |
| -1 | -00000001 | 11111111 | 01111111 |
| 0 | 00000000 | 00000000 | 10000000 |
| 1 | 00000001 | 00000001 | 10000001 |
| ⋮ | ⋮ | ⋮ | ⋮ |
| 127 | 11111111 | 01111111 | 11111111 |

2025年9月

北京理工大学计算机学院

2.2 机器数的定点表示与浮点表示

计算机组成原理

偏置值为 2^n 的移码具有以下特点：

(1) 在移码中，最高位为“0”表示负数，最高位为“1”表示正数。

(2) 移码为全0时，它所对应的真值最小，为全1时，它所对应的真值最大。

(3) 真值0在移码中的表示形式是唯一的，即 $[+0]_{\text{移}} = [-0]_{\text{移}} = 100\dots 0$ 。

(4) 移码把真值映射到一个正数域，所以可将移码视为无符号数，直接按无符号数规则比较大小。

2025年9月

北京理工大学计算机学院

2.2 机器数的定点表示与浮点表示

计算机组成原理

注意

(5) 同一数值的移码和补码除最高位相反外，其他各位相同。

浮点数的阶码常采用移码表示最主要的原因有：

- 便于比较浮点数的大小。阶码大的，其对应的真值就大，阶码小的，对应的真值就小。
- 简化机器中的判零电路。当阶码全为0，尾数也全为0时，表示机器零。

2025年9月

北京理工大学计算机学院

2.2 机器数的定点表示与浮点表示

计算机组成原理

移码要在真值X上加一个偏置值，那么如何选择这个偏置值呢？假设阶码共 $n+1$ 位，则总共有 2^{n+1} 个无符号整数。这 2^{n+1} 个无符号整数可以对应于 2^{n+1} 个阶码的真值。显然，选择的偏置值应该使得阶码真值的正数和负数分布均匀。 2^{n+1} 个数中有 2^n 个正数， 2^n 个负数，居于中间的两个数是： 2^n-1 （二进制码01...11）和 2^n （二进制码10...00），可以选择这两个数中的任何一个作为偏置值，阶码真值的正数和负数分布基本均匀。

2025年9月

北京理工大学计算机学院

2.2 机器数的定点表示与浮点表示

计算机组成原理

2.2.5 IEEE754标准浮点数

大多数计算机的浮点数采用IEEE 754标准，其格式如下，IEEE754标准中有三种形式的浮点数。

| 类型 | 数符 m_s | 阶码 E | 尾数 m | 总位数 | 偏置值 |
|-------|----------|------|------|-----|---------------|
| 短浮点数 | 1 | 8 | 23 | 32 | 7FH / 127 |
| 长浮点数 | 1 | 11 | 52 | 64 | 3FFH / 1023 |
| 临时浮点数 | 1 | 15 | 64 | 80 | 3FFFH / 16383 |

2025年9月

北京理工大学计算机学院

2.2 机器数的定点表示与浮点表示

计算机组成原理

以短浮点数为例讨论浮点代码与其真值之间的关系。最高位为数符位；其后是8位阶码，以2为底，阶码的偏置值为127；其余23位是尾数。为了使尾数部分能表示更多一位的有效值，IEEE754采用隐含尾数最高数位1（即这一位1不表示出来）的方法，因此尾数实际上是24位。应注意的是，隐含的1是一位整数（即位权为 2^0 ），在浮点格式中表示出来的23位尾数是纯小数，并用原码表示。

2025年9月

北京理工大学计算机学院

2.2 机器数的定点表示与浮点表示

计算机组成原理

例1: 将 $(100.25)_{10}$ 转换成短浮点数格式。

(1) 十进制数→二进制数

$$(100.25)_{10} = (1100100.01)_2$$

(2) 非规格化数→规格化数

$$1100100.01 = 1.10010001 \times 2^6$$

(3) 计算移码表示的阶码 (偏置值+阶码真值)

$$1111111 + 110 = 10000101$$

(4) 以短浮点数格式存储该数。

符号位=0

阶码=10000101

尾数=100100010000000000000000

2025年9月

北京理工大学计算机学院

2.2 机器数的定点表示与浮点表示

计算机组成原理

短浮点数代码为

0;100 0010 1;100 1000 1000 0000 0000 0000

表示为十六进制的代码: 42C88000H。

例2: 把短浮点数C1C90000H转换成十进制数。

(1) 十六进制→二进制形式, 并分离出符号位、阶码和尾数。

C1C90000H=

1;10000011;100100100000000000000000

符号位 阶码

尾数

2025年9月

北京理工大学计算机学院

2.2 机器数的定点表示与浮点表示

计算机组成原理

(2) 计算出阶码真值 (移码-偏置值)

$$10000011 - 11111111 = 100$$

(3) 以规格化二进制数形式写出此数

$$1.1001001 \times 2^4$$

(4) 写成非规格化二进制数形式

11001.001

(5) 转换成十进制数, 并加上符号位。

$$(11001.001)_2 = (25.125)_{10}$$

所以, 该浮点数=-25.125

2025年9月

北京理工大学计算机学院

2.2 机器数的定点表示与浮点表示

计算机组成原理

2.2.6 定点/浮点表示法与定点/浮点计算机

1. 定点/浮点表示法的区别

假设定点数和浮点数的字长相同。

(1) 数值的表示范围

浮点表示法所能表示的数值范围将远远大于定点数。

2025年9月

北京理工大学计算机学院

2.2 机器数的定点表示与浮点表示

计算机组成原理

应当注意的有两点:

- 不管定点数还是浮点数, 每个数值都对应于数轴上的一个点。所谓数的表示范围实际上指的只是数的上、下限, 它们之间是一些不连续的点, 而不是一段连续的区间。
- 对于定点数而言, 各个点在数轴上的分布是均匀的; 而对于浮点数而言, 各个点在数轴上的分布是不均匀的, 越靠近数轴的原点, 两个相邻数之间的距离就越近。

2025年9月

北京理工大学计算机学院

2.2 机器数的定点表示与浮点表示

计算机组成原理

(2) 精度

浮点数虽然扩大了数的表示范围, 但这正是以降低精度为代价的, 也就是数轴上各点的排列更稀疏了。

2025年9月

北京理工大学计算机学院

2.2 机器数的定点表示与浮点表示

计算机组成原理

(3)数的运算

浮点运算要比定点运算复杂得多。

(4)溢出处理

在定点运算时，当运算结果超出数的表示范围，就发生溢出。而在浮点运算时，运算结果超出尾数的表示范围却并不一定溢出，只有当阶码超出所能表示的范围时，才发生溢出。

2025年9月

北京理工大学计算机学院

2.2 机器数的定点表示与浮点表示

计算机组成原理

2.定点机与浮点机

并不是所有的计算机都具有浮点运算功能，通常可以分为几档：

(1)定点机

以定点运算为主，浮点运算是通过软件来实现的。

(2)定点机+浮点运算部件

浮点运算部件（FPU）是专门用于对浮点数进行运算的部件。

(3)浮点机

具有浮点运算指令和基本的浮点运算器。

2025年9月

北京理工大学计算机学院

第2章

计算机组成原理

2.1 数值数据的表示

2.2 机器数的定点表示与浮点表示

2.3 非数值数据的表示

2.4 十进制数和数串的表示

2.5 不同类型的数据表示举例

2.6 数据校验码

2025年9月

北京理工大学计算机学院

2.3 非数值数据的表示

计算机组成原理

2.3.1 字符和字符串的表示方法

1.ASCII字符编码

常见的ASCII码用七位二进制表示一个字符，它包括10个十进制数字（0~9）、52个英文大写和小写字母（A~Z，a~z）、34个专用符号和32个控制符号，共计128个字符。

在ASCII码表中，数字和英文字母都是按顺序排列的，只要知道其中一个的二进制代码，不要查表就可以推导出其他数字或字母的二进制代码。

2025年9月

北京理工大学计算机学院

2.3 非数值数据的表示

计算机组成原理

| $b_7b_6b_5b_4b_3b_2b_1b_0$ | 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 | RO | RS | - | > | N | ^ | n | ~ |
| 1111 | SI | US | / | ? | O | _ | o | DEL |

2025年9月

北京理工大学计算机学院

2.3 非数值数据的表示

计算机组成原理

2.字符串的存放

字符串是指一串连续的字符。例如，字符串IF X>0 THEN READ (C)。

(1)向量法

在存储器中占用一片连续的空间，每个字节存放一个字符代码，字符串的所有元素（字符）在物理上是邻接的。在字长为32位的存储器，每一个主存单元可存放4个字符，整个字符串需5个主存单元。在每个字节中实际存放的是相应字符的ASCII码。

2025年9月

北京理工大学计算机学院

2.3 非数值数据的表示

计算机组成原理



| | | | |
|----|----|----|----|
| 49 | 46 | 20 | 58 |
| 3E | 30 | 20 | 54 |
| 48 | 45 | 4E | 20 |
| 52 | 45 | 41 | 44 |
| 28 | 43 | 29 | 20 |

2025年9月

北京理工大学计算机学院

2.3 非数值数据的表示

计算机组成原理

(2)串表法

一个存储单元有32位，仅存放一个字符代码。字符串的每个字符代码后有一个链接字，用以指出下一个字符的存储单元地址。串表法不要求串中的各个字符在物理上相邻，在对字符串进行删除和插入操作时，只需修改相应字符代码后面的链接字即可。

由于链接字占据了存储单元的大部分空间，使得主存的有效利用率下降（只有原来的25%）。

上例中整个字符串需19个主存单元。

2025年9月

北京理工大学计算机学院

2.3 非数值数据的表示

计算机组成原理

2.3.2 汉字的表示

1.汉字国标码

GB2312-80，简称国标码。该标准共收集常用汉字6763个，其中一级汉字3755个，按拼音排序；二级汉字3008个，按部首排序；另外还有各种图形符号682个，共计7445个。

每个汉字、图形符号都用两个字节表示，每个字节只使用低七位编码。

2025年9月

北京理工大学计算机学院

2.3 非数值数据的表示

计算机组成原理

2.汉字区位码

区位码将汉字编码GB2312-80中的6763个汉字分为94个区，每个区中包含94个汉字（位），区和位组成一个二维数组，每个汉字在数组中对应一个唯一的区位码。汉字的区位码定长4位，前2位表示区号，后2位表示位号，区号和位号用十进制数表示，区号从01到94，位号也从01到94。例如，“中”字在54区的48位上，其区位码为“54-48”，“国”字在25区的90位上，其区位码为“25-90”。

2025年9月

北京理工大学计算机学院

2.3 非数值数据的表示

计算机组成原理

汉字区位码并不等于汉字国标码，它们两者之间的关系可用以下公式表示：

国标码 = 区位码（十六进制） + 2020H

例如：已知汉字“春”的区位码为“20-26”，计算它的国标码。

| | | | |
|------|------|------|------|
| 区位码： | 20 | 26 | 十进制 |
| | ↓ | ↓ | |
| | 14H | 1AH | 十六进制 |
| | +20H | +20H | |
| 国标码： | 34H | 3AH | |

2025年9月

北京理工大学计算机学院

2.3 非数值数据的表示

计算机组成原理

3.汉字机内码

汉字可以通过不同的输入码输入，但在计算机内部其内码是唯一的。

因为汉字处理系统要保证中西文的兼容，当系统中同时存在ASCII码和汉字国标码时，将会产生二义性。

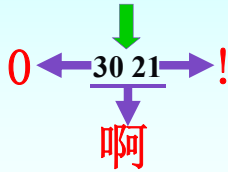
2025年9月

北京理工大学计算机学院

2.3 非数值数据的表示

计算机组成原理

例如：从主存中读出两个字节的内容，它们分别为30H和21H，这时既可能是表示汉字“啊”的国标码，又可能是表示西文“0”和“!”的ASCII码。



2025年9月

北京理工大学计算机学院

2.3 非数值数据的表示

计算机组成原理

常用的汉字机内码为两字节长的代码，它是在相应汉字国标码的每个字节最高位上加“1”。即：

汉字机内码 = 汉字国标码 + 8080H

例如，上述“啊”字的国标码是3021H，其汉字机内码则是B0A1H。

$$\begin{array}{r} 3021 \\ + 8080 \\ \hline B0A1 \end{array}$$

2025年9月

北京理工大学计算机学院

2.3 非数值数据的表示

计算机组成原理

4. 汉字字形码

汉字字形码是指确定一个汉字字形点阵的代码，又叫汉字字模码或汉字输出码。在一个汉字点阵中，凡笔画所到之处，记为“1”，否则记为“0”。

根据对汉字质量的不同要求，可有16×16、24×24、32×32或48×48的点阵结构。显然点阵越大，输出汉字的质量越高，每个汉字所占用的字节数也越多。

2025年9月

北京理工大学计算机学院

2.3 非数值数据的表示

计算机组成原理

5. 汉字编码的发展

1990年颁布了繁体字的编码标准GB12345-90，目的在于规范必须使用繁体字的各种场合，该标准共收录6866个汉字（比GB2312多103个字），纯繁体字大概有2200余个。

1995年底推出的GBK编码是中文编码扩展国家标准，该编码标准兼容GB2312，共收录汉字21003个、符号883个，并提供1894个造字码位，简、繁体字融于一库。

2000年底又颁布了GB18030大字符集标准，这个标准可以涵盖27484个汉字，繁、简体均处于同一平台。

2025年9月

北京理工大学计算机学院

2.3 非数值数据的表示

计算机组成原理

2.3.3 统一代码（Unicode）

随着国际间的交流与合作的扩大，信息处理应用对字符集提出了多文种、大音量、多用途的要求，解决问题的最佳方案是设计一种全新的编码方法，这种方法必须有足够的能力来表示任意一种语言里使用的所有符号，这就是统一代码（Unicode）。

2025年9月

北京理工大学计算机学院

2.3 非数值数据的表示

计算机组成原理

1. 编码方式

Unicode的基本方法是用一个**16位的数**来表示每个符号，这种符号集可表示65536个不同的字符或符号。被称为基本多语言平面（BMP）。这个空间已经非常大了，但设计者考虑到将来某一天它可能也会不够用，所以采用了一种可使这种表示法使用得更远的方法。

2025年9月

北京理工大学计算机学院

2.3 非数值数据的表示

计算机组成原理

当只用2字节数来表示Unicode字符时，使用的是UCS-2编码，但尽管如此，也允许在UCS-2文本中插入一些UCS-4字符。为此，在BMP中，保留了两个有1024个大小的块，这两个块中任何位置都不能用来表示任何符号。UCS-4的两个16位字每个表示一个数，这个数是UCS-2 BMP中1024个数值中的一个。这两个数的组合可以表示多达1百万多个自定义的UCS-4字符。

UCS-2: $65536 \times 2 \times 1024$

UCS-4: $2^{10} \times 2^{10} = 2^{20}$

2025年9月

北京理工大学计算机学院

2.3 非数值数据的表示

计算机组成原理

2. 实现方式

Unicode的实现方式不同于编码方式。一个字符的Unicode编码是确定的，但是在实际传输过程中，由于不同系统平台的设计不一定一致，以及出于节省空间的目的，对Unicode编码的实现方式有所不同。Unicode的实现方式称为Unicode转换格式（Unicode Translation Format，简称为UTF），目前存在的UTF格式有：UTF-7，UTF-7.5，UTF-8，UTF-16以及UTF-32。

2025年9月

北京理工大学计算机学院

第2章

计算机组成原理

2.1 数值数据的表示

2.2 机器数的定点表示与浮点表示

2.3 非数值数据的表示

2.4 十进制数和数串的表示

2.5 不同类型的数据表示举例

2.6 数据校验码

2025年9月

北京理工大学计算机学院

2.4 十进制数和数串的表示

计算机组成原理

2.4.1 十进制数的编码（二—十进制编码）

用四位二进制数来表示一位十进制数，称为二进制编码的十进制数，简称BCD码。

四位二进制数可以组合出16种代码，能表示16种不同的状态，我们只需要使用其中的10种状态，就可以表示十进制数的0~9十个数码，而其他的六种状态为冗余状态。由于可以取任意的10种代码来表示十个数码，所以就可能产生多种BCD编码。BCD编码既具有二进制数的形式，又保持了十进制数的特点。

2025年9月

北京理工大学计算机学院

2.4 十进制数和数串的表示

计算机组成原理

几种常见的BCD码

| 十进制数 | 8421码 | 2421码 | 余3码 | Gray码 |
|------|-------|-------|------|-------|
| 0 | 0000 | 0000 | 0011 | 0000 |
| 1 | 0001 | 0001 | 0100 | 0001 |
| 2 | 0010 | 0010 | 0101 | 0011 |
| 3 | 0011 | 0011 | 0110 | 0010 |
| 4 | 0100 | 0100 | 0111 | 0110 |
| 5 | 0101 | 1011 | 1000 | 1110 |
| 6 | 0110 | 1100 | 1001 | 1010 |
| 7 | 0111 | 1101 | 1010 | 1011 |
| 8 | 1000 | 1110 | 1011 | 1001 |
| 9 | 1001 | 1111 | 1100 | 1000 |

2025年9月

北京理工大学计算机学院

2.4 十进制数和数串的表示

计算机组成原理

1.8421码

8421码又称为NBCD码，其主要特点是：

(1) 它是一种有权码，四位二进制代码的位权从高到低分别为8、4、2、1。

(2) 简单直观。每个代码与它所代表的十进制数之间符合二进制数和十进制数相互转换的规则。

(3) 不允许出现1010~1111。这六个代码在8421码中是非法码。

2025年9月

北京理工大学计算机学院

2.4 十进制数和数串表示

计算机组成原理

2.2421码

其主要特点是：

(1) 它也是一种有权码，四位二进制代码的位权从高到低分别为**2、4、2、1**。

(2) 它又是一种对9的**自补码**。即某数的2421码，只要自身按位取反，就能得到该数对9之补的2421码。例如：

3的2421码是0011。3对9之补是6，而6的2421码是1100。

(3) 不允许出现**0101~1010**。这六个代码在2421码中是非法码。

2025年9月

北京理工大学计算机学院

2.4 十进制数和数串表示

计算机组成原理

3.余3码

其主要特点是：

(1) 这是一种无权码，但也可看作是一种特殊的有权码，即在8421码的基础上加**+3 (+0011)**形成的，故称余3码。在这种编码中各位的“1”不表示一个固定的十进制数值，因而不直观。

(2) 它也是一种对9的**自补码**。

(3) 不允许出现**0000~0010、1101~1111**。这六个代码在余3码中是非法码。

2025年9月

北京理工大学计算机学院

2.4 十进制数和数串表示

计算机组成原理

4. 格雷码 (Gray) 码

十进制Gray码的方案有很多种，Gray码可以避免在计数时发生中间错误，所以也被称为可靠性编码。其主要特点是：

(1) 它也是一种无权码。

(2) 从一种代码变到相邻的下一代码时，只有一个二进制位的状态在发生变化。

(3) 具有循环特性，即首尾两个数的Gray码也只有一个二进制位不同，因此Gray码又称为循环码。

(4) 十进制Gray码也有6个代码为非法码，视具体方案而定。

2025年9月

北京理工大学计算机学院

2.4 十进制数和数串表示

计算机组成原理

2.4.2 十进制数串

1. 非压缩的十进制数串

一个字节可存放一个十进制数或符号的ASCII码。

非压缩的十进制数串又根据符号所处的位置，分成**前导式数字串**和**后嵌入式数字串**。

在前导式数字串中，符号位占用单独一个字节，放在数值位之前，正号对应的ASCII码为**2BH**，负号对应的ASCII码为**2DH**。

2025年9月

北京理工大学计算机学院

2.4 十进制数和数串表示

计算机组成原理

如：**+256**，在主存中连续四个字节存放，对应的ASCII码为**2BH, 32H, 35H, 36H**。

如：**-256**，在主存中连续四个字节存放，对应的ASCII码为**2DH, 32H, 35H, 36H**。

在后嵌入式数字串中，符号位不单独占用一个字节，正号为**00H**，负号为**40H**，嵌入到最末数值位。所以正数最末位不变，负数最末位加上**40H**。

如：**+256**，在主存中连续三个字节存放，对应的ASCII码为**32H, 35H, 36H**。

如：**-256**，在主存中连续三个字节存放，对应的ASCII码为**32H, 35H, 76H**。

2025年9月

北京理工大学计算机学院

2.4 十进制数和数串表示

计算机组成原理

2. 压缩的十进制数

一个字节可存放两位BCD码表示的十进制数，既节省了存储空间，又便于直接进行十进制算术运算。

在主存中，一个压缩的十进制数串占用多个字节，每位数字仅占半个字节，通常用8421码表示。符号位也占半个字节，并放在最低数值位之后，通常用**CH**表示正号，**DH**表示负号。在这种表示中，规定数字的个数加符号位之和必须为偶数，当和为奇数时，应在最高数值位之前补一个“0”（即第一个字节的高半字节为“0000”）。

2025年9月

北京理工大学计算机学院

2.4 十进制数和数串 的表示

计算机组成原理

如：+256
两个字节：25H，6CH

| | |
|------|------|
| 0010 | 0101 |
| 0110 | 1100 |

如：-2568
三个字节：02H，56H，8DH

| | |
|------|------|
| 0000 | 0010 |
| 0101 | 0110 |
| 1000 | 1101 |

2025年9月

北京理工大学计算机学院

第2章

计算机组成原理

2.1 数值数据的表示

2.2 机器数的定点表示与浮点表示

2.3 非数值数据的表示

2.4 十进制数和数串的表示

2.5 不同类型的数据表示举例

2.6 数据校验码

2025年9月

北京理工大学计算机学院

2.5 不同类型的数据表示举例

计算机组成原理

2.5.1 C语言中的数据表示

整型数据

实型数据

字符型数据

2025年9月

北京理工大学计算机学院

2.5 不同类型的数据表示举例

计算机组成原理

2.5.2 现代微型计算机系统 中的数据表示

IA-32结构的基本数据类型

2025年9月

北京理工大学计算机学院

第2章

计算机组成原理

2.1 数值数据的表示

2.2 机器数的定点表示与浮点表示

2.3 非数值数据的表示

2.4 十进制数和数串的表示

2.5 不同类型的数据表示举例

2.6 数据校验码

2025年9月

北京理工大学计算机学院

2.6 数据校验码

计算机组成原理

数据校验码是指那些能够发现错误或能够自动纠正错误的 数据编码，又称之为“检错纠错编码”。

任何一种编码都由许多码字构成，任意两个码字之间最少变化的二进制位数，被称为数据校验码的码距。例如，用四位二进制表示16种状态，则有16个不同的码字，此时码距为1，即两个码字之间最少仅有一个二进制位不同（如0000与0001之间）。这种编码没有检错能力，因为当某一个合法码字中有一位或几位出错，就变成另一个合法码字了。

2025年9月

北京理工大学计算机学院

2.6 数据校验码计算机组成原理

具有检、纠错能力的数据校验码的实现原理是：在编码中，除去合法的码字外，再加进一些非法的码字，当某个合法码字出现错误时，就变成非法码字。合理地安排非法码字的数量和编码规则，就能达到纠错的目的。例如，若用四位二进制表示八个状态，其中只有八个码字是合法码字，而另八个码字为非法码字，此时码距为2。对于码距 ≥ 2 的数据校验码，开始具有检错的能力。码距越大，检、纠错能力就越强，而且检错能力总是大于或等于纠错能力。

2025年9月北京理工大学计算机学院

2.6 数据校验码计算机组成原理

2.6.1 奇偶校验码

1. 奇偶校验概念

奇偶校验码是一种最简单的数据校验码，它可以检测出一位（或奇数位）错误。奇偶校验码的码距等于2。

奇偶校验实现方法是：由若干位有效信息（如一个字节），再加上一个二进制位（校验位）组成校验码，然后根据校验码的奇偶性质进行校验。

奇偶校验码（N+1位）= N位有效信息+1位校验位

2025年9月北京理工大学计算机学院

2.6 数据校验码计算机组成原理

2025年9月北京理工大学计算机学院

2.6 数据校验码计算机组成原理

2025年9月北京理工大学计算机学院

2.6 数据校验码计算机组成原理

校验位的取值（0或1）将使整个校验码中“1”的个数为奇数或偶数，所以有两种可供选择的校验规律：

奇校验——整个校验码（有效信息位和校验位）中“1”的个数为奇数。

偶校验——整个校验码中“1”的个数为偶数。

| 有效信息（8位） | 奇检验码（9位） | 偶检验码（9位） |
|----------|-----------|-----------|
| 00000000 | 100000000 | 000000000 |
| 01010001 | 001010001 | 101010001 |
| 01111111 | 001111111 | 101111111 |
| 11111111 | 111111111 | 011111111 |

2025年9月北京理工大学计算机学院

2.6 数据校验码计算机组成原理

| 数据 | 奇偶校验位 | 奇偶校验码 | 结果 |
|----------|-------|----------|------|
| 01010101 | 1 | 10101011 | 正确 |
| 01010101 | 0 | 10101010 | 中断处理 |
| 01010101 | 1 | 10101011 | 正确 |
| 01010101 | 0 | 10101010 | 中断处理 |

2025年9月北京理工大学计算机学院

2.6 数据校验码

计算机组成原理

2. 简单奇偶校验（以奇校验为例）

(1) 校验位形成

当要把一个字节的代码 $D_7 \sim D_0$ 写入主存时，就同时将它们送往奇偶校验逻辑电路，该电路产生的“奇形成”信号就是校验位。它将与8位代码一起作为奇校验码写入主存。

若 $D_7 \sim D_0$ 中有偶数个“1”，则“奇形成”=1，

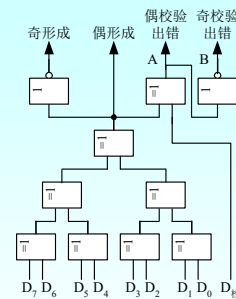
若 $D_7 \sim D_0$ 中有奇数个“1”，则“奇形成”=0。

2025年9月

北京理工大学计算机学院

2.6 数据校验码

计算机组成原理



2025年9月

北京理工大学计算机学院

2.6 数据校验码

计算机组成原理

(2) 校验检测

读出时，将读出的9位代码（8位信息位和1位校验位）同时送入奇偶校验电路检测。若读出代码无错，则“奇校验出错”=0；若读出代码中的某一位上出现错误，则“奇校验出错”=1，从而指示这个9位代码中一定有某一位出现了错误，但具体的错误位置是不能确定的。

2025年9月

北京理工大学计算机学院

2.6 数据校验码

计算机组成原理

3. 交叉奇偶校验

计算机在进行大量字节（数据块）传送时，不仅每一个字节有一个奇偶校验位做横向校验，而且全部字节的同一位也设置一个奇偶校验位做纵向校验，这种横向、纵向同时校验的方法称为交叉校验。

| | | | | | | | | | | |
|------|---|---|---|---|---|---|---|---|---|---|
| 第1字节 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | → | 1 |
| 第2字节 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | → | 0 |
| 第3字节 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | → | 0 |
| 第4字节 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | → | 0 |
| | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | | |
| | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | | |

2025年9月

北京理工大学计算机学院

2.6 数据校验码

计算机组成原理

交叉校验可以发现两位同时出错的情况，假设第2字节的 a_6 、 a_4 两位均出错，横向校验位无法检出错误，但是第 a_6 、 a_4 位所在列的纵向校验位会显示出错，这与前述的简单奇偶校验相比要保险多了。

2025年9月

北京理工大学计算机学院

2.6 数据校验码

计算机组成原理

2.6.2 海明校验码

海明码实际上是一种多重奇偶校验，其实现原理是：在有效信息位中加入几个校验位形成海明码，并把海明码的每一个二进制位分配到几个奇偶校验组中。当某一位出错后，就会引起有关的几个校验位的值发生变化，这不但可以发现错误，还能指出错误的位置，为自动纠错提供了依据。

能检测和自动校正一位错，并能发现两位错的海明码的编码原理。此时校验位的位数 K 和信息位的位数 N 应满足下列关系： $2^{K-1} \geq N+K+1$ 。

2025年9月

北京理工大学计算机学院

2.6 数据校验码

计算机组成原理

(1) 编码

一个字节由8位二进制位组成，此时 $N=8$ ， $K=5$ ，故海明码的总位数为13位，可表示为：

$$H_{13} \ H_{12} \ \dots \ H_2 \ H_1$$

五个校验位 $P_5 \sim P_1$ 对应的海明码位号应分别为： H_{13} 、 H_8 、 H_4 、 H_2 、 H_1 ，除 P_5 外，其余四位都满足 P_i 的位号等于 2^{i-1} 的关系，而 P_5 只能放在 H_{13} 上，因为它已经是海明码的最高位了。因此，有如下排列关系：

$$P_5 \ D_8 \ D_7 \ D_6 \ D_5 \ P_4 \ D_4 \ D_3 \ D_2 \ P_3 \ D_1 \ P_2 \ P_1$$

2025年9月

北京理工大学计算机学院

2.6 数据校验码

计算机组成原理

各个信息位形成 P_i 值的偶校验的结果：

$$P_1 = D_1 \oplus D_2 \oplus D_4 \oplus D_5 \oplus D_7$$

$$P_2 = D_1 \oplus D_3 \oplus D_4 \oplus D_6 \oplus D_7$$

$$P_3 = D_2 \oplus D_3 \oplus D_4 \oplus D_8$$

$$P_4 = D_5 \oplus D_6 \oplus D_7 \oplus D_8$$

依据各信息位形成 P_i ($i=1 \sim 4$) 值时，不同信息位出现在 P_i 项中的次数是不一样的，其中 D_4 和 D_7 都出现三次，而 D_1 、 D_2 、 D_3 、 D_5 、 D_6 、 D_8 仅出现两次，为此，还要补充一位 P_5 校验位，使：

$$P_5 = D_1 \oplus D_2 \oplus D_3 \oplus D_5 \oplus D_6 \oplus D_8$$

2025年9月

北京理工大学计算机学院

2.6 数据校验码

计算机组成原理

(2) 校验

将接收到的海明码按如下关系进行偶校验：

$$S_1 = P_1 \oplus D_1 \oplus D_2 \oplus D_4 \oplus D_5 \oplus D_7$$

$$S_2 = P_2 \oplus D_1 \oplus D_3 \oplus D_4 \oplus D_6 \oplus D_7$$

$$S_3 = P_3 \oplus D_2 \oplus D_3 \oplus D_4 \oplus D_8$$

$$S_4 = P_4 \oplus D_5 \oplus D_6 \oplus D_7 \oplus D_8$$

$$S_5 = P_5 \oplus D_1 \oplus D_2 \oplus D_3 \oplus D_5 \oplus D_6 \oplus D_8$$

校验得到的结果值 $S_5 \sim S_1$ （指误字），它能反映13位海明码的出错情况：

2025年9月

北京理工大学计算机学院

2.6 数据校验码

计算机组成原理

① 当 $S_5 \sim S_1$ 为00000时，表明无错。

② 当 $S_5 \sim S_1$ 中仅有一位不为0，表明是某一校验位出错或三位海明码（包括信息位和校验位）同时出错。

③ 当 $S_5 \sim S_1$ 中有两位不为0，表明是两位海明码同时出错，此时只能发现错误，而无法确定出错的位置。

④ 当 $S_5 \sim S_1$ 中有三位不为0，表明是一位信息位出错或三位校验位同时出错，出错位的位号由 $S_4 \sim S_1$ 四位编码值指明，此时不仅能检查出一位错，而且能准确地定位，因此可以纠正这个错误（将该位变反）。

2025年9月

北京理工大学计算机学院

2.6 数据校验码

计算机组成原理

⑤ 当 $S_5 \sim S_1$ 中有四位或五位不为0时，表明出错情况严重，系统工作可能出现故障，应检查系统硬件的正确性。

当五个 S_i 位有三个为1时，表示是某一信息位 D_i 出错。出错信息位的海明码位号由 $S_4 \sim S_1$ 这四位的译码值指出（分别为12、11、10、9、7、6、5、3）。例如，当 $S_5 \sim S_1 = 00111$ 时， $S_4 \sim S_1$ 的译码值为7，即对应 H_7 （也就是 D_4 ）位出错。

2025年9月

北京理工大学计算机学院

2.6 数据校验码

计算机组成原理

例：设有一个8位信息为10101100，试求海明编码的生成和校验过程。

(1) 编码生成

按偶校验有：

$$P_1 = 0 \oplus 0 \oplus 1 \oplus 0 \oplus 0 = 1$$

$$P_2 = 0 \oplus 1 \oplus 1 \oplus 1 \oplus 0 = 1$$

$$P_3 = 0 \oplus 1 \oplus 1 \oplus 1 = 1$$

$$P_4 = 0 \oplus 1 \oplus 0 \oplus 1 = 0$$

$$P_5 = 0 \oplus 0 \oplus 1 \oplus 0 \oplus 1 \oplus 1 = 1$$

∴ 可得到用二进制表示的海明码为：

$$\underline{1} \ 1 \ 0 \ 1 \ 0 \ \underline{0} \ 1 \ 1 \ 0 \ 1 \ 0 \ 1 \ 1$$

2025年9月

北京理工大学计算机学院

2.6 数据校验码

计算机组成原理

(2) 校验

假设传送后 H_{11} (D_7) 位发生了错误:

11 1 1001101011

出错

检错的过程很简单, 只要将接受到的码字重新进行偶校验:

$$S_1 = 1 \oplus 0 \oplus 0 \oplus 1 \oplus 0 \oplus 1 = 1$$

$$S_2 = 1 \oplus 0 \oplus 1 \oplus 1 \oplus 1 \oplus 1 = 1$$

$$S_3 = 1 \oplus 0 \oplus 1 \oplus 1 \oplus 1 = 0$$

$$S_4 = 0 \oplus 0 \oplus 1 \oplus 1 \oplus 1 = 1$$

$$S_5 = 1 \oplus 0 \oplus 0 \oplus 1 \oplus 0 \oplus 1 \oplus 1 = 0$$

2025年9月

北京理工大学计算机学院

2.6 数据校验码

计算机组成原理

所以指误字为**01011**, 其中低4位有效, 相应的十进制数是**11**, 指出 H_{11} 出错。现在 H_{11} 错成了“1”, 纠错就是将 H_{11} 位取反让它恢复为“0”。即:

错误码: 11 1 1001101011

↓

纠正后: 11 0 1001101011

2025年9月

北京理工大学计算机学院

2.6 数据校验码

计算机组成原理

2.6.3 循环冗余校验码

除了奇偶校验码和海明码外, 在计算机网络、同步通信以及磁表面存储器中广泛使用循环冗余校验码, 简称CRC码。

循环冗余校验码是通过除法运算来建立有效信息位和校验位之间的约定关系的。

假设, 待编码的有效信息以多项式 $M(X)$ 表示, 用另一个约定的多项式 $G(X)$ 去除, 所产生的余数 $R(X)$ 就是检验位。有效信息和检验位相拼接就构成了CRC码。

2025年9月

北京理工大学计算机学院

2.6 数据校验码

计算机组成原理

当整个CRC码被接收后, 仍用约定的多项式 $G(X)$ 去除, 若余数为0表明该代码是正确的; 若余数不为0表明某一位出错, 再进一步由余数值确定出错的位置, 以便进行纠正。

循环冗余校验码编码规律如下:

①把待编码的 N 位有效信息表示为多项式 $M(X)$ 。

②把 $M(X)$ 左移 K 位, 得到 $M(X) \times X^K$, 这样空出了 K 位, 以便拼装 K 位余数 (即校验位)。

2025年9月

北京理工大学计算机学院

2.6 数据校验码

计算机组成原理

③选取一个 $K+1$ 位的产生多项式 $G(X)$, 对 $M(X) \times X^K$ 作模2除。

④把左移 K 位以后的待编有效信息与余数 $R(X)$ 作模2加减, 拼接为CRC码, 此时的CRC码共有 $N+K$ 位。

$$M(X) \times X^K + R(X) = Q(X) \times G(X)$$

例如, 选择产生多项式为1011, 把4位有效信息1100编成CRC码。

$$M(X) = X^3 + X^2 = 1100$$

$$M(X) \times X^3 = X^6 + X^5 = 1100000$$

$$G(X) = X^3 + X + 1 = 1011$$

2025年9月

北京理工大学计算机学院

2.6 数据校验码

计算机组成原理

$$\frac{M(X) \times X^3}{G(X)} = \frac{1100000}{1011} = 1110 + \frac{010}{1011}$$

$$M(X) \times X^3 + R(X) = 1100000 + 010 = 1100010$$

2025年9月

北京理工大学计算机学院

第2章 小结

计算机组成原理

2.1 数值数据的表示

- ☞ 无符号数
- ☞ 带符号数
- 真值、机器数
- ☞ 原码、补码、反码表示
- ☞ 三种机器数对于真值0的表示方法
- ☞ 三种机器数的比较

2025年9月

北京理工大学计算机学院

第2章 小结

计算机组成原理

2.2 机器数的定点表示与浮点表示

- ☞ 定点小数表示范围（原码、补码）
- ☞ 定点整数表示范围（原码、补码）
- ☞ 浮点数表示范围
- ☞ 规格化的浮点数
- ☞ 阶码的移码表示
- ☞ IEEE 754浮点数标准

2025年9月

北京理工大学计算机学院

第2章 小结

计算机组成原理

2.3 非数值数据的表示

- ☞ ASCII码
- ☞ 汉字国标码
- ☞ 汉字区位码
- ☞ 汉字机内码
- ☞ 国标码、区位码与机内码之间的转换
- ☞ 汉字字形码

2025年9月

北京理工大学计算机学院

第2章 小结

计算机组成原理

2.4 十进制数和数串的表示

8421码、2421码、余3码

2.6 数据校验码

- ☞ 奇偶校验码和奇偶校验位
- ☞ 海明检验码

2025年9月

北京理工大学计算机学院