河北师范大学软件学院
Software College of Hebei Normal University

计算机组成原理

第二章 数据的机器层次表示

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 现代微型计算机系统中的数据表示举例

2.6 数据校验码

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 现代微型计算机系统中的数据表示举例

2.6 数据校验码

2.1 数值数据的表示

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。

2.1 数值数据的表示

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

2.1 数值数据的表示

对于字长为n+1位的无符号数的表示范围是0～$2^{n+1}-1$。

00000000

11111111

例如：字长为8位，无符号数的表示范围是0～255。

2.1 数值数据的表示

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。

2.1 数值数据的表示

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。

2.1 数值数据的表示

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

$[X_1]_{原}$=0.0110，$[X_2]_{原}$=1.0110

2.1 数值数据的表示

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

$[X_1]_{原}$=0,1101，$[X_2]_{原}$=1,1101

在原码表示中，真值0有两种不同的表示形式：

$[+0]_{原}$ =00000

$[-0]_{原}$ =10000

2.1 数值数据的表示

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。

2.1 数值数据的表示

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。

2.1 数值数据的表示

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。

2.1 数值数据的表示

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 (mod M)，如23=13（mod10）

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

时针倒着旋转2圈，等于分针正着旋转10圈。故有：-2=10 (mod 12)，即 -2和10同余。

2.1 数值数据的表示

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 $X_s.X_1X_2\ldots X_n$，其中X_s表示符号位。

例1：$X_1=0.0110$， $X_2=-0.0110$

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

2.1 数值数据的表示

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

例2：$X_1=1101$ ，　　$X_2=-1101$

$[X_1]_补=0,1101$, $[X_2]_补=1,0011$

在补码表示中，真值0的表示形式是唯一的。

$[+0]_补=[-0]_补=00000$

2.1 数值数据的表示

3.由真值、原码转换为补码

当X为正数时，$[X]_{补}=[X]_{原}=X$。

当X为负数时，由$[X]_{原}$转换为$[X]_{补}$的方法：

①$[X]_{原}$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：$[X]_{原}$ =1.1110011000

$[X]_{补}$ =1.0001101000

不变　　变反　　不变

2.1 数值数据的表示

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$X_s.X_1X_2\ldots X_n$，其中X_s表示符号位。

例1：$X_1=0.0110$， $X_2=-0.0110$

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

2.1 数值数据的表示

若真值为纯整数，它的反码形式为$X_sX_1X_2\ldots X_n$，其中X_s表示符号位。

例2：X_1=1101, X_2=-1101

$[X_1]_{反}$=0,1101，$[X_2]_{反}$=1,0010

在反码表示中，真值0也有两种不同的表示形式：

$[+0]_{反}$=00000

$[-0]_{反}$=11111

2.1 数值数据的表示

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。

2.1 数值数据的表示

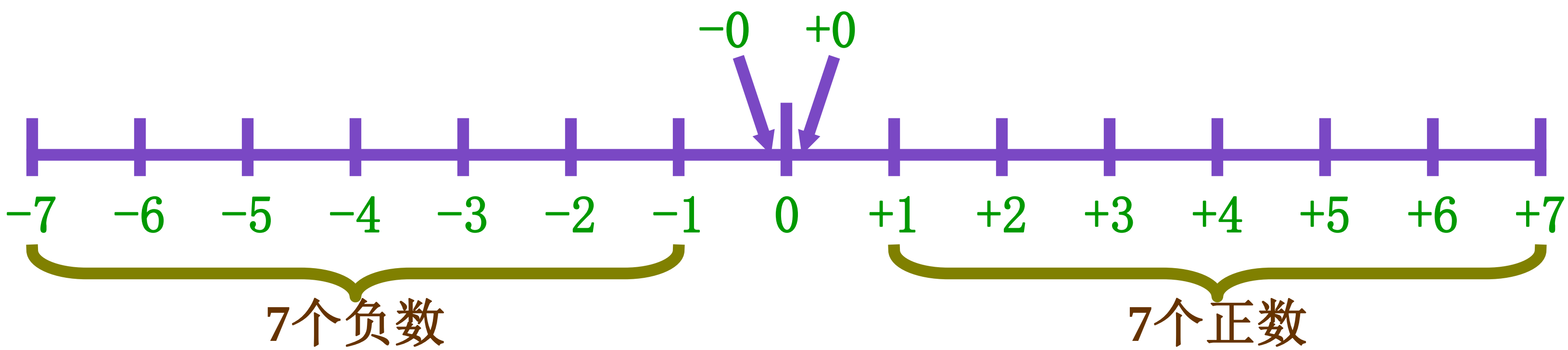
⑶ 对于真值0，原码和反码各有两种不同的表示形式，而补码只有唯一的一种表示形式。

⑷ 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）。

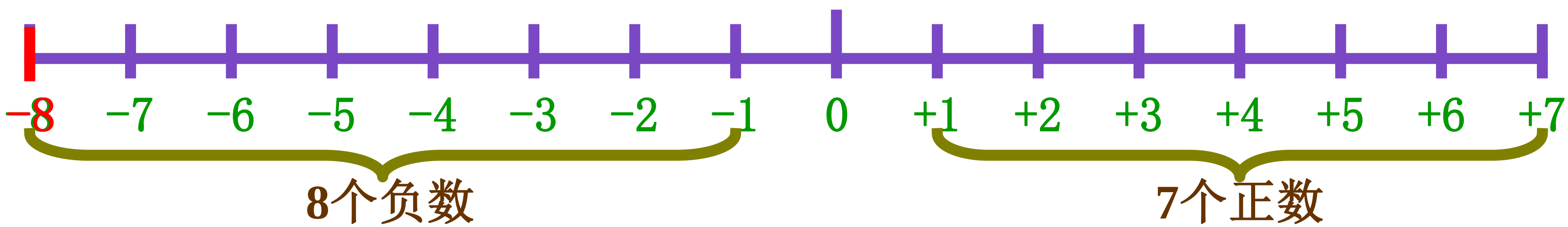
2.1 数值数据的表示

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）



2.1 数值数据的表示

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	−

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 现代微型计算机系统中的数据表示举例

2.6 数据校验码

2.2 机器数的定点表示与浮点表示

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$X_s.X_1X_2\ldots X_n$，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。

2.2 机器数的定点表示与浮点表示

	2^0	2^{-1}	2^{-2}	…	$2^{-(n-1)}$	2^{-n}
定点小数格式	X_s	X_1	X_2	…	X_{n-1}	X_n

小数点位置

	2^0	2^{-1}	2^{-2}	…	$2^{-(n-1)}$	2^{-n}
最大正数	0	1	1	…	1	1

当X_s=0，X_1～X_n=1时，X为最大正数，即：

$X_{最大正数}$ =(1-2^{-n})。

2.2 机器数的定点表示与浮点表示

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

小数点位置

最小正数

2^0	2^{-1}	2^{-2}	…	$2^{-(n-1)}$	2^{-n}
0	0	0	…	0	1

当X_n=1，X_s～X_{n-1}=0时，X为最小正数，即：$X_{最小正数}$ =2^{-n}。

2.2 机器数的定点表示与浮点表示

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。

2.2 机器数的定点表示与浮点表示

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

$X_{\text{绝对值最大负数（补码表示时）}}=-1$

2.2 机器数的定点表示与浮点表示

综上所述：

若机器字长有n+1位，则：

原码定点小数表示范围为：$-(1-2^{-n})\sim(1-2^{-n})$

补码定点小数表示范围为：$-1\sim(1-2^{-n})$

若机器字长有8位，则：

原码定点小数表示范围为：$-(1-2^{-7})\sim(1-2^{-7})$

补码定点小数表示范围为：$-1\sim(1-2^{-7})$

2.2 机器数的定点表示与浮点表示

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$X_sX_1X_2\dots X_n$，这个数是一个纯整数。

	2^n	2^{n-1}	2^{n-2}	…	2^0
定点整数格式	X_s	X_1	X_2	…	X_n

小数点位置

	2^n	2^{n-1}	2^{n-2}	…	2^1	2^0
最大正数	0	1	1	…	1	1

$$X_{最大正数}=(2^n-1)$$

2.2 机器数的定点表示与浮点表示

最小正数

2^n	2^{n-1}	2^{n-2}	…	2^1	2^0
0	0	0	…	0	1

$X_{最小正数}=\mathbf{1}$

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

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

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

$X_{绝对值最大负数（补码表示时）}=\mathbf{-2^n}$

2.2 机器数的定点表示与浮点表示

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：-127～127

补码定点整数表示范围为：-128～127

2.2 机器数的定点表示与浮点表示

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。

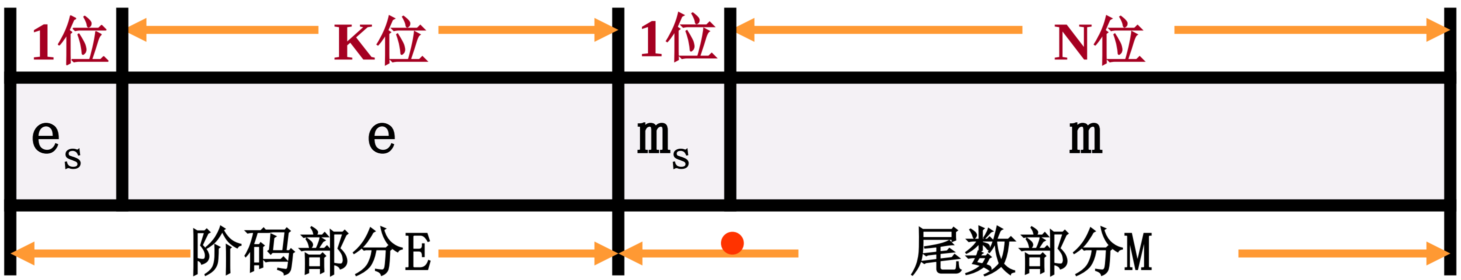
例如：

$$N=M\times r^{E}\ \ =M\times 2^{E}$$

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。

2.2 机器数的定点表示与浮点表示

浮点数的一般格式：



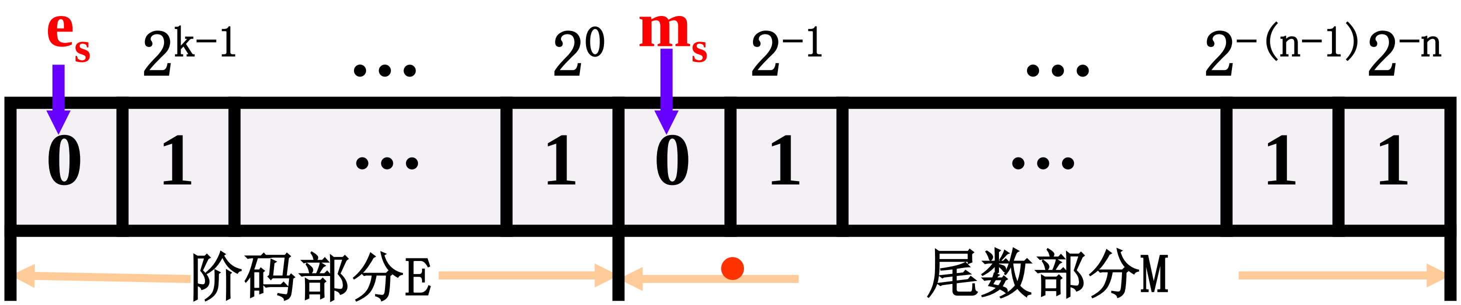
浮点数的底是隐含的，在整个机器数中不出现。阶码的符号位为e_s，阶码的大小反映了在数N中小数点的实际位置；尾数的符号位为m_s，它是整个浮点数的符号位，反映了该浮点数的正负。

假设阶码和尾数部分均用补码表示。

2.2 机器数的定点表示与浮点表示

1.浮点数的表示范围

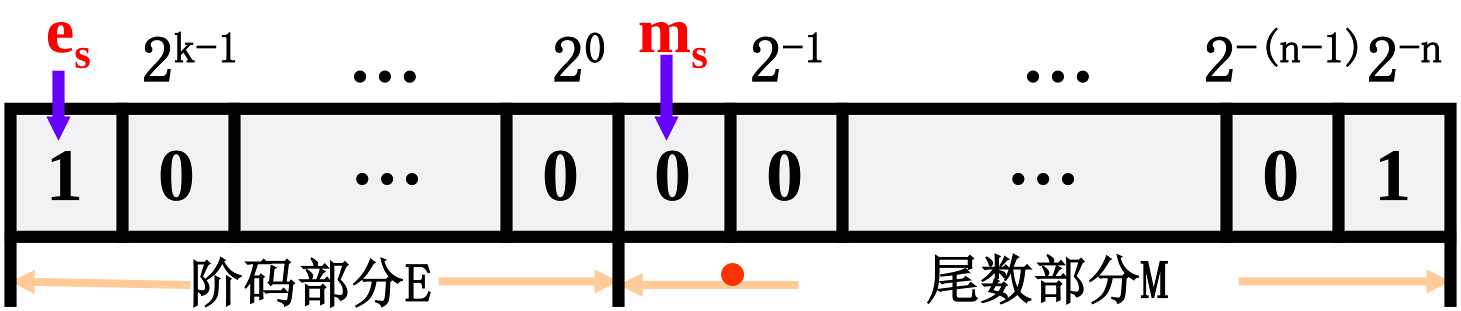
当e_s=0，m_s=0，阶码和尾数的数值位各位全为1（即阶码和尾数都为最大正数）时，该浮点数为最大正数。



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

2.2 机器数的定点表示与浮点表示

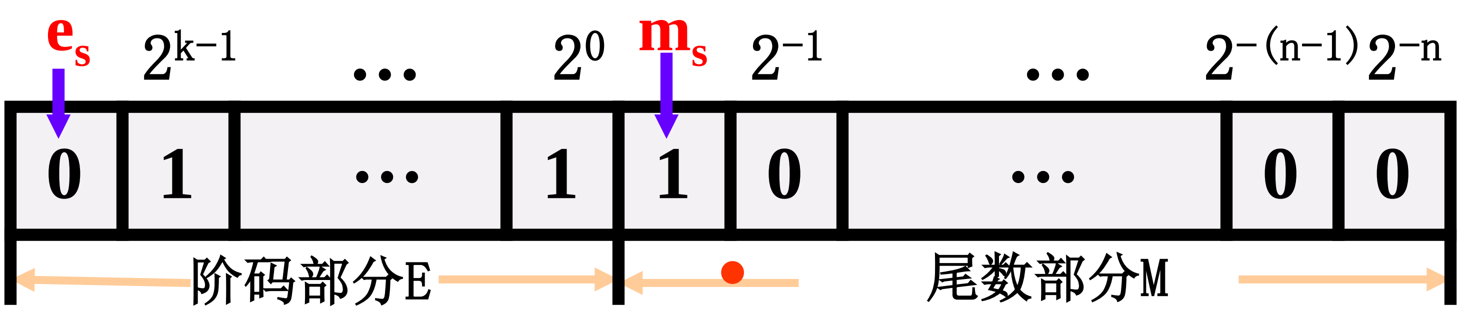
当$e_s=1$，$m_s=0$，尾数的最低位$m_n=1$，其余各位为0（即阶码为绝对值最大负数，尾数为最小正数）时，该浮点数为最小正数。



$$X_{最小正数}=2^{-n}\times 2^{-2^k}$$

2.2 机器数的定点表示与浮点表示

当$e_s=0$，阶码的数值位为全1；$m_s=1$，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。



$$X_{绝对值最大负数}=-1\times 2^{2^k-1}$$

2.2 机器数的定点表示与浮点表示

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**。

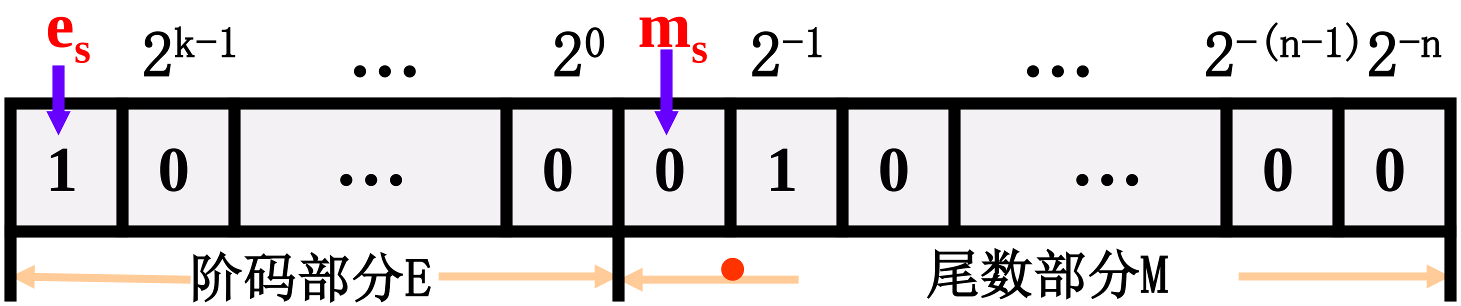
$$1/r \le |M| < 1$$

如果r=2，则有$1/2 \leqslant |M| < 1$。

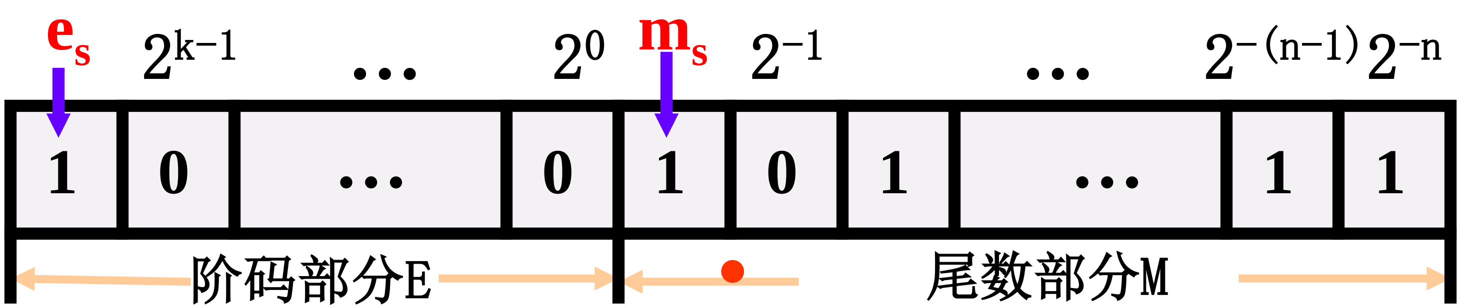
2.2 机器数的定点表示与浮点表示

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$1/2 \le M < 1$时，应有0.1xx…x形式，当$-1 \le M < -1/2$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。

2.2 机器数的定点表示与浮点表示



$$X_{规格化的最小正数}=2^{-1}\times 2^{-2^{k}}$$



$$X_{规格化的绝对值最小负数}=-(2^{-1}+2^{-n})\times 2^{-2^{k}}$$

2.2 机器数的定点表示与浮点表示

	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	$(-2^{-1}-2^{-n})\times 2^{-2^k}$

2.2 机器数的定点表示与浮点表示

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。

$[X]_{移}$=偏置值+X

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

2.2 机器数的定点表示与浮点表示

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

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

$[X]_{补}=01011101$

例2：X=-1011101

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

$[X]_{补}=10100011$

2.2 机器数的定点表示与浮点表示

真值X（十进制）	真值X（二进制）	$[X]_{补}$	$[X]_{移}$
-128	-10000000	10000000	00000000
-127	-1111111	10000001	00000001
⋮	⋮	⋮	⋮
-1	-0000001	11111111	01111111
0	0000000	00000000	10000000
1	0000001	00000001	10000001
127	1111111	01111111	11111111

2.2 机器数的定点表示与浮点表示

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$[+0]_{移}=[-0]_{移}=100\ldots0$。

(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。

2.2 机器数的定点表示与浮点表示

(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。

2.2 机器数的定点表示与浮点表示

2.2.4 实用浮点数举例

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。

m_s	E	m

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

2.2 机器数的定点表示与浮点表示

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。

2.2 机器数的定点表示与浮点表示

例1：将$(100.25)_{10}$转换成短浮点数格式。

⑴ 十进制数→二进制数

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

⑵ 非规格化数→规格化数

$1100100.01=1.10010001\times2^6$

⑶ 计算移码表示的阶码（偏置值＋阶码真值）

1111111+110=10000101

⑷ 以短浮点数格式存储该数。

符号位=0

阶码=10000101

尾数=10010001000000000000000

2.2 机器数的定点表示与浮点表示

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**转换成为十进制数。

(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;10010010000000000000000

符号位 阶码　　　　　　　　尾数

2.2 机器数的定点表示与浮点表示

⑵ 计算出阶码真值（移码－偏置值）

10000011-1111111=100

⑶ 以规格化二进制数形式写出此数

1.1001001×2^4

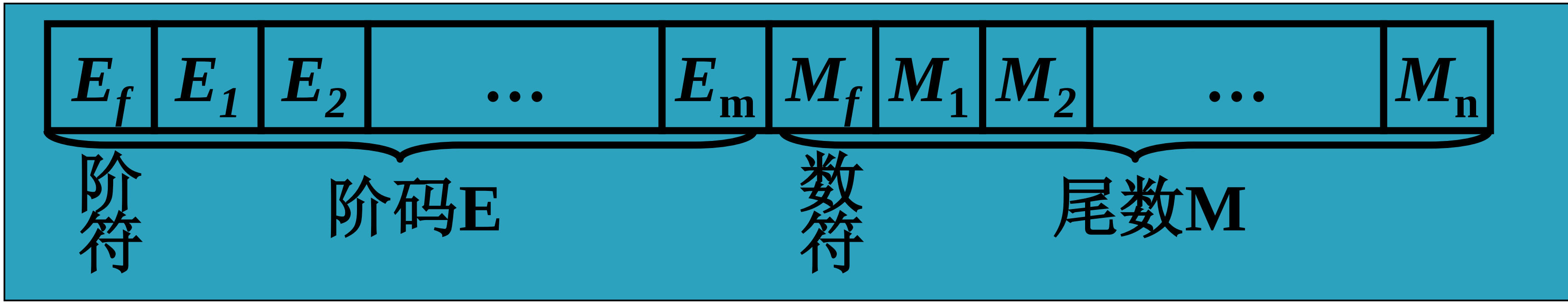
⑷ 写成非规格化二进制数形式

11001.001

⑸ 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**



例1 某浮点数格式如图示，字长32其中阶码8位，含一位阶符，补码表示，以2为底；尾数24位，含一位数符，补码表示，规格化。若浮点数代码为$(A3680000)_{16}$求其真值。

$(A3680000)_{16}=(10100011，01101000000\ldots0)_2$

$E=-(1011101)_2=-(93)_{10}$

$M=(0.11010\ldots0)_2=(0.8125)_{10}$

$N=2^{-93}\times0.8125$

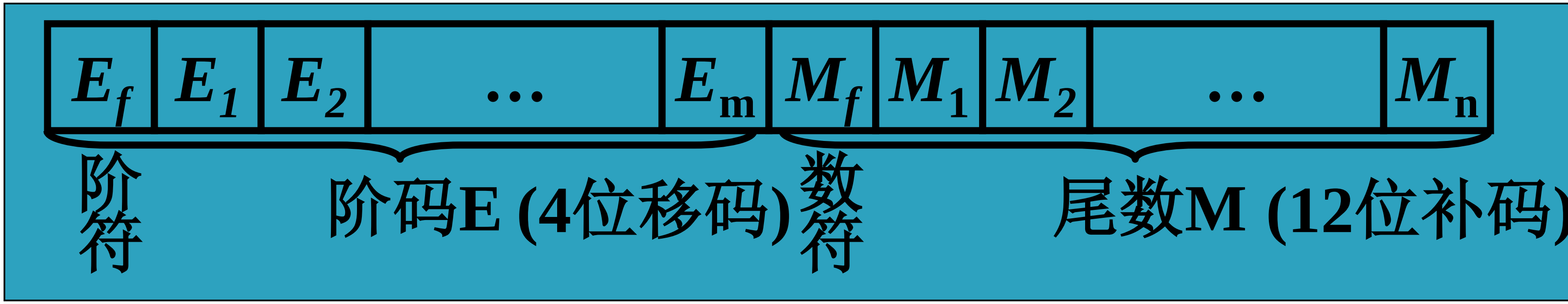
E_f	E_1	E_2	…	E_m	M_f	M_1	M_2	…	M_n
阶符	阶码E (8位补码)				数符	尾数M (24位补码)			

例2 按上述浮点格式将－（1011.11010…0）$_2$写成浮点数代码。

$$N = -(1011.11010\ldots0)_2$$
$$= -(0.101111010\ldots0)_2 \times 2^4$$
$$E = (4)_{10} = (0000100)_2$$
$$E_{补} = 00000100$$
$$M_{补} = (1.010000110\ldots0)_2$$

浮点数代码为（00000100，1010000110…0）$_2$

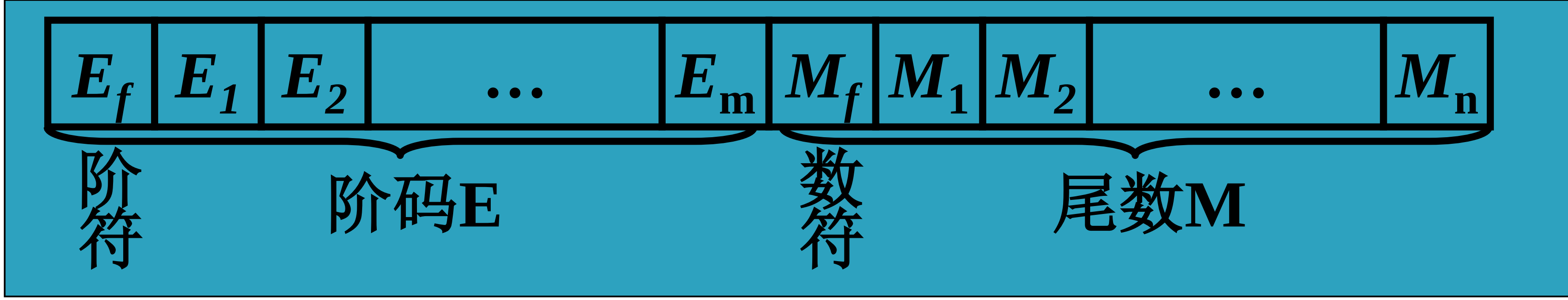
$$= (04A18000)_{16}$$



例3 按上述浮点格式将$-2^6 \times 0.4375$写成浮点数代码。

$$N = (-2^6 \times 0.4375)_{10}$$
$$= -(0.01110000000)_2 \times 2^6$$
$$= -(0.11100000000)_2 \times 2^5$$
$$E = (5)_{10} = (0101)_2$$
$$E_{移} = 1000 + E = (1101)_2$$
$$M_{补} = (1.00100000000)_2$$

浮点数代码为$(1101，100100000000)_2$
$$= (D900)_{16}$$



例4 某浮点数格式如图示，字长32，其中阶码8位，含一位阶符，移码表示，以2为底；尾数24位，含一位数符，补码表示，规格化。若浮点数代码为$(\text{BDB40000})_{16}$求其真值。

$(\text{BDB40000})_{16}=(1011\ 1101，1011\ 0100\ 0000\ldots00)_2$

$(E)_{移}=(1011\ 1101)_2$

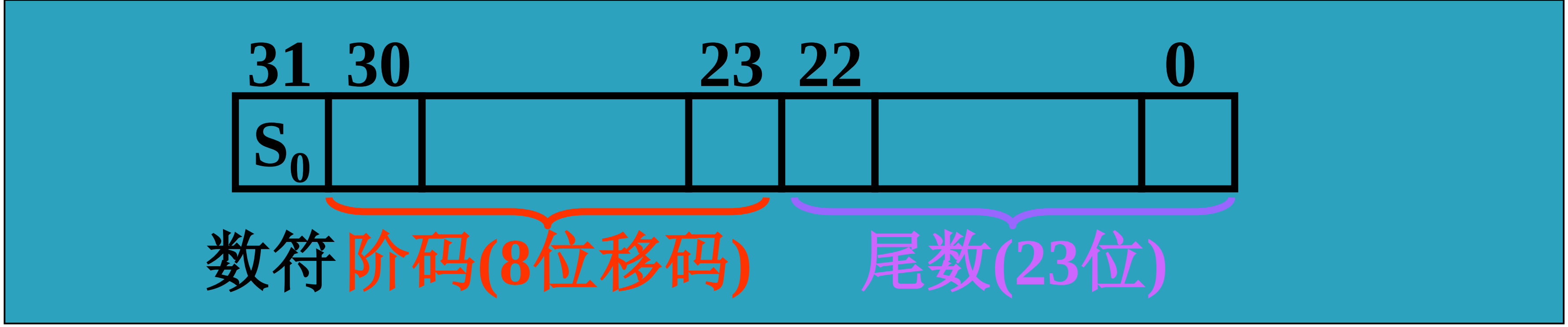
$=2^7+E$

$E=(111101)_2=(61)_{10}$

$(M)_{补}=1.011010\ldots0$

$M=-(0.100110\ldots0)_2=-(0.59375)_{10}$

$N=-(0.59375)\times 2^{61}$



例5：写出下列十进制数的IEEE754短浮点数编码

(1)0.15625;(2)-5

解：

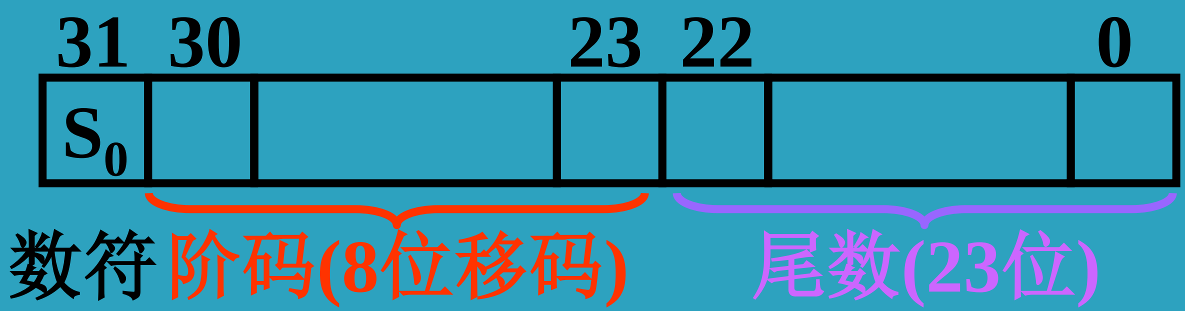
(1)$(0.15625)_{10}=(0.00101)_2$，

1.01×2^{-3}，

$E_{移}=127-3=(124)_{10}$

$=(01111100)_2$

0 01111100 010000…00



例6：写出下列十进制数的IEEE754短浮点数编码

(1)0.15625; (2)-5

解：

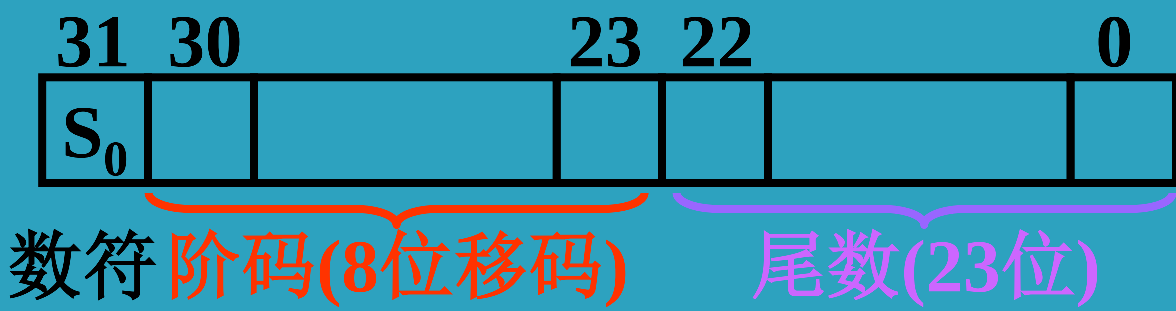
(2)$-(5)_{10}=-(101)_2$，

$-(1.01\times 2^2)$，

$E_{移}=127+2=(129)_{10}$

$=(10000001)_2$

1 10000001 01000…00



例7：若短浮点数IEEE754编码为

1011 1111 0100 0000 0000 0000 0000 0000，则其代表的十进制数为多少？

1 01111110 10000000000000000000000

阶码的真值＝E－127＝01111110－01111111＝－1

尾数(包含隐含位)＝（1.10000000…0）$_2$

其代表的十进制数为$-(1.1)\times 2^{-1}=-(0.11)_2=-(0.75)_{10}$

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 现代微型计算机系统中的数据表示举例

2.6 数据校验码

2.3 非数值数据的表示

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。

2.3 非数值数据的表示

$b_6b_5b_4$ / $b_3b_2b_1b_0$	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

2.3 非数值数据的表示

2.字符串的存放

2.3 非数值数据的表示

2.3.2 汉字的表示

1.汉字国标码

2.汉字区位码

3.汉字机内码

4.汉字字形码

2.3 非数值数据的表示

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、大字量、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**）。

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 现代微型计算机系统中的数据表示举例

2.6 数据校验码

2.4 十进制数和数串的表示

2.4.1 十进制数的编码（二一十进制编码）

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。

2.4 十进制数和数串的表示

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