计算机运算基础

复习1 常见的几种数制

- 数制

  进位计数制

  - 十进制（Decimal System）

    符合人们的习惯
  - 二进制（Binary System）

    便于物理实现
  - 八进制（Octave System）
  - 十六进制（Hexadecimal System）

    便于识别、书写

几个重要概念

重点概念1：

计算机中的数据都是以二进制形式进行存储和运算的

重点概念2：

在计算机中存储数据时，每类数据占据固定长度的二进制数位，而不管其实际长度。一般长度为字节的整倍数

例如：在八位微机中，

整数216　存储为11011000B

整数56　存储为00111000B

重点概念3：

计算机中不仅要处理无符号数，还要处理带符号和带小数点的数。

重点概念4：　机器数与真值

1）十进制

特点：每一位数有0～9十个数码，计数基数为10

进位方式为逢十进一，且一个十进制数可以用加权的形式展开。

$1998_D = 1\times10^3 + 9\times10^2 + 9\times10^1 + 8\times10^0$

其中：$10^3, 10^2, 10^1, 10^0$称为为权由数码所在位　决定

另外：$179.023_D = 1\times10^2 + 7\times10^1 + 9\times10^0 + 0\times10^{-1} + 2\times10^{-2} + 3\times10^{-3}$

2) 二进制

特点：每一位有0，1两个数码，计数基数为2，进位方式为 逢二 进

展开式:

$$1011.11_B=1\times2^3+0\times2^2+1\times2^1+1\times2^0+1\times2^{-1}+1\times2^{-2}=11.625_D$$

结论： 二进制加权展开式的和等于二进制数所对应的十进制数

3）十六进制

特点：每一位有0～9,A,B,C,D,E,F十六个数码，计数基数为16进位方式逢十六进一

$$6E8_H = 6\times16^2 + E\times16^1 + 8\times16^0 = 1768_D$$

结论:十六进制数的加权展开式之和等于它所对应的十进制数。

1 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。

将168D 转换为二进制数

方法：“除基取余倒排法”——十进制整数

2	168	···	0
2	84	···	0
2	42	···	0
2	21	···	1
2	10	···	0
2	5	···	1
2	2	···	0
2	1	···	1
	0		

余数倒排

$\therefore\ 168_D = 10101000_B$

$10101000_B = 1\times2^7+0\times2^6+1\times2^5+0\times2^4+1\times2^3+0\times2^2+0\times2^1+0\times2^0$

$=168_D$

186_D

16 | 186 ··· 10 ↑
16 | 11 ··· 11
| 0

$BA_H = 11\times16^1+10\times16^0 = 186_D$

（即乘2取整法，位数取决于要求精度）

$(0.613)_{10}\times2=1.226$ $k_{-1}=1$

$(0.226)_{10}\times2=0.452$ $k_{-2}=0$

$(0.452)_{10}\times2=0.904$ $k_{-3}=0$

$(0.904)_{10}\times2=1.808$ $k_{-4}=1$ （0. 1001）$_2$ =（0. 5625）$_{10}$

$(0.808)_{10}\times2=1.616$ $k_{-5}=1$ （0.10011）$_2$=（0. 609375）$_{10}$

$(0.616)_{10}\times2=1.232$ $k_{-6}=1$

$(0.232)_{10}\times2=0.464$ $k_{-7}=0$

2 二进制与十六进制间的转换

- 二进制数转换成十六进制数
  - 用4位二进制数表示1位十六进制数

例：(10110001001.110)B= (?)H

0101 1000 1001.1100

5 8 9 . C

- 十六进制数转换成二进制数
  - 用1位十六进制数表示4位二进制数

例：$(1863.5B)_{16}$ = $(?)_2$

(1 8 6 3. 5 B)$_{16}$

(0001 1000 0110 0011.0101 1011)$_2$

$(1863.5B)_{16}$= $(0001100001100011.01011011)_2$

3 八进制数、十六进制数与二进制数的相互转换

例：八进制： 2 5 7 · 0 5 5 4

二进制：010 101 111 · 000 101 101 100

十六进制： A F · 1 6 C

因此，$(257.0554)_8=(10101111.0001011011)_2$

$=(AF.16C)_{16}$

思考：计算机采用二进制形式表示数据和指令，
在书写，显示上引进十六进制的意义是什么？
计算机内部使用十六进制吗？

- 十进制数与二进制数之间的转换需计算，不直观；
- 二进制表示的数位多不便于书写、阅读；
- 十六进制数与二进制数间转换方便、直观，

相对于二进制数，十六进制数书写、阅读相对方便。

小结：数的转换

N进制转换为十进制——将各位之位权与对应之幂相乘展开，再累计求和即可；

十进制转换为N进制——将整数和小数分开，分别转换后再拼接。

●十进制转换为N进制转换口诀：整数部分除基取余(首次余数为代码整数最低位);小数部分乘基取整(首次整数为小数点后最高位).

●可形象记忆为：小数点两边走；

复习-2 数的表示方法

- 真值与机器数
- 数的定点和浮点表示方法
- 原码、反码、补码
- 常用编码

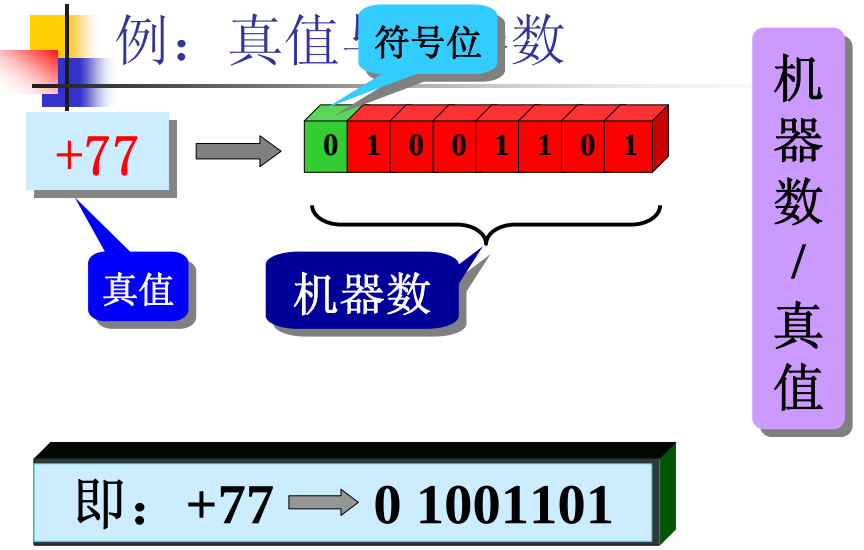
真值与机器数

1、真　值：直接用"+"和"–"表示符号的二进制数，不能在机器使用.

2、机器数：将符号数值化了的二进制数,可在机器中使用。

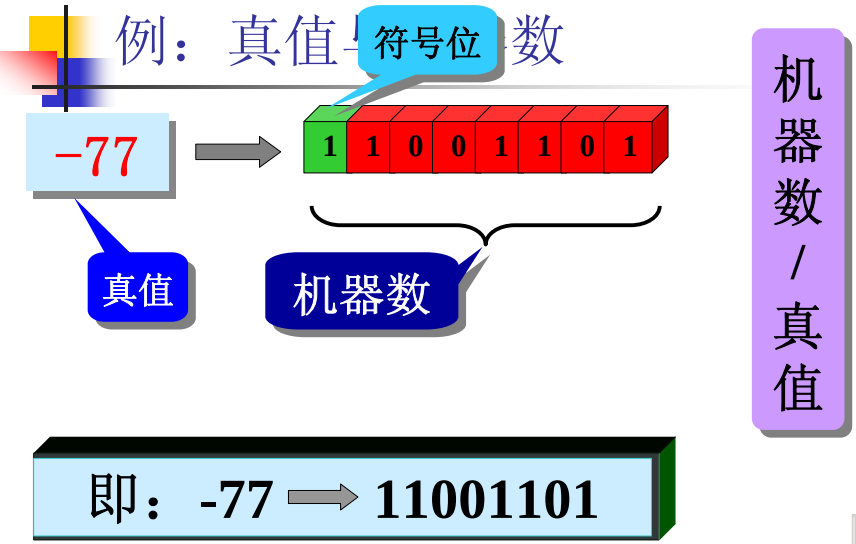
3、一般将符号位放在数的最高位。

例：真值与机器数



即：+77 ⟶ 0 1001101

例：真值与机器数



即：-77 ⟶ 11001101

2　数的定点与浮点表示

计算机中如何表示实数中的小数点呢？

计算机中不用专门的器件表示小数点，而是用数的两种不同的表示法来表示小数点的位置。

根据小数点的位置是否固定，数的表示方法分为**定点表示和浮点表示**,相应的机器数称为**定点数和浮点数**。

任意一个二进制数N均可表示为：

$$N = S \cdot 2^J$$

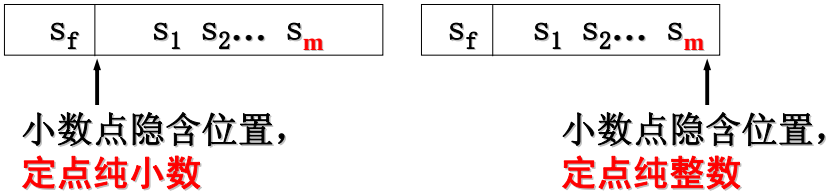
其中：

S称为数N的尾数，表示数N的全部有效数字，决定了N的精度。

J称为数N的阶码，底为2，指明了小数点的位置，决定了数N的大小范围。

（1）定点表示法

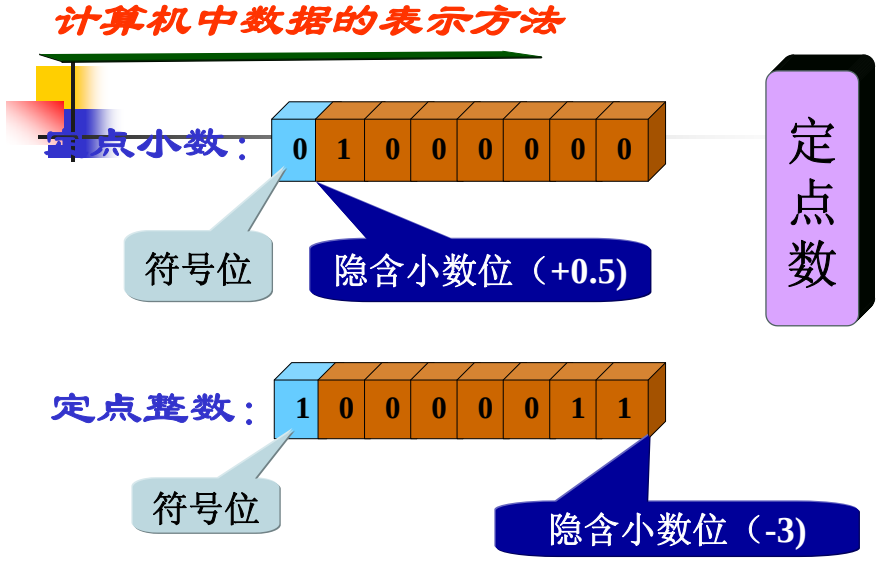
计算机在处理定点数时，常把小数点固定在数值位的最后面或最前面，即分为定点纯小数与定点纯整数两类，如图1-6所示。



例如:

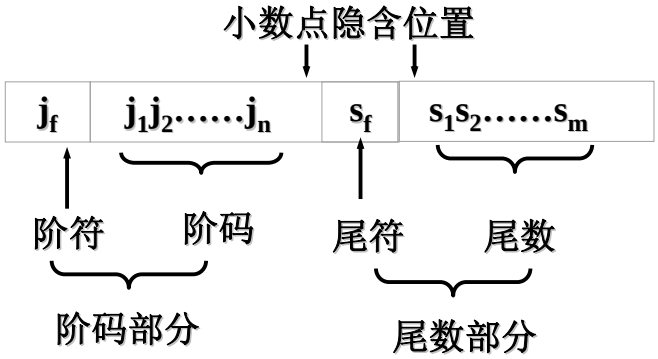
00011000B，如果看作定点纯整数，其真值为**24**

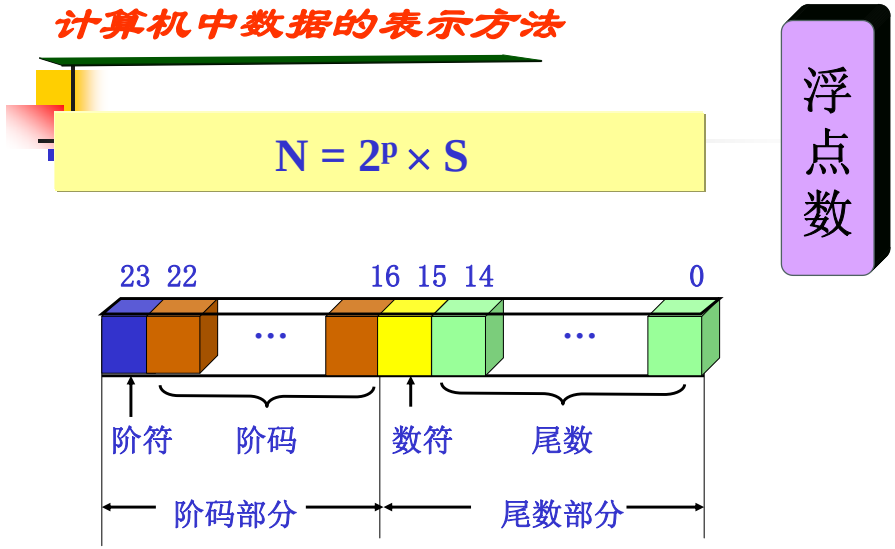
看作定点纯小数，其真值为**0.1875**



（2）浮点表示法

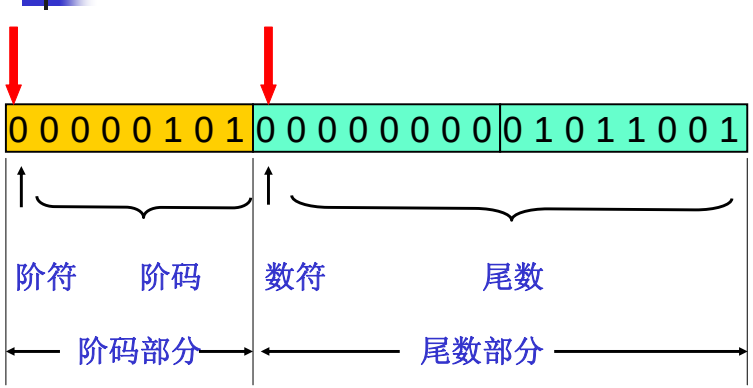
在浮点表示法中，小数点的位置是浮动的，阶码J可取不同的数值，则在计算机中除了要表示尾码S，还要表示阶码J。因此，一个浮点数表示为阶码和尾数两部分，**尾数一般是定点纯小数**，**阶码是定点纯整数**，其形式如图1-7所示。





例:X= +10110.01= 2^{+101} ×（+ 0.1011001）

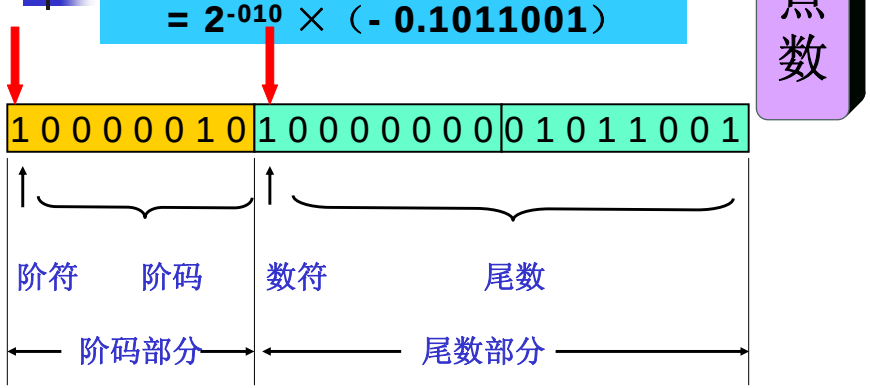
浮点数



例:X= -0.001011001

= 2^{-010} ×（- 0.1011001）

浮点数



无符号数

二进制数的各位均表示数值大小，最高位无符号意义。

例　1111 0000 B = F0H = 15 × 16 = 240D

1001 0001 B = 91H = 9 × 16 + 1= 145 D

应用场合:

处理的数全是正数时，如表示地址的数

带符号数

- 数有正、负 → 带符号数

通常数的最高位为符号位，对于字长8位机器数:

D_7为符号位: 0表示“+”，1表示“-”。符号数码化了。

D_6～D_0为数字位。

如:　X=（01011011）$_2$=+91　X=（11011011）$_2$= - 91

连同符号位在一起作为一个数称为机器数，

机器数的数值称为的真值。

如:　N1=+ 1011011　　N2= - 1011011 为真值

0 1011011　　1 101 1011 为机器数

符号数码化了，对数据进行运算时，符号位应如何处理?

把符号位和数值位一起编码：原码，反码，补码。

- 在计算机中符号也用二进制数表示

把符号位和数值位一起编码：原码，反码，补码。

原码：正数符号位用“0”表示，负数符号用“1”表示，这种表示法称为原码。

例：

符号 数值

X=+105 [X]原= 0 1101001

X=-105 [X]原= 1 1101001

原码表示简单，真值 转换方便，减法不方便。引进反码，补码。

反码：

正数反码：表示与原码相同，(最高位“0”表示正，其余位为数值位。)

负数的反码：表示为负数原码的符号位不变尾数按位取反。

例：

符号 数值

$[+4]_{反}$ = 0 0000100

$[-4]_{反}$ = 1 1111011

$[+127]_{反}$ = 0 1111111

$[-127]_{反}$ = 1 0000000

$[+0]_{反}$ = 0 0000000

$[-0]_{反}$ = 1 1111111

补码：

正数的补码表示与原码相同，(最高位 用“0”表示正，其余位为数值位.)

负数的补码表示为它的反码+1。

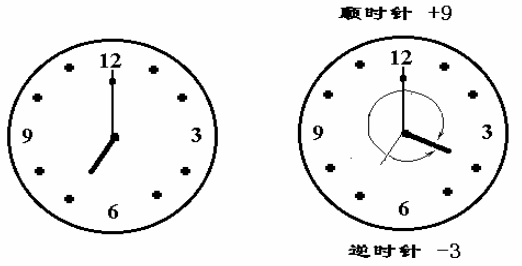
[+127]原=0 1111111 [+0]原=0 0000000

[-127]反=1 0000000 [-0]反=1 1111111

[-127]补=1 0000001 [-0]补=0 0000000

补码的含义：

以时钟对时为例来说明，现由7点钟调到4点钟。



顺时针调： 7+9 =4 （mod 12）

逆时针调： 7−3 = 4 （mod 12）

由于时钟上超过12点时就会自动丢失一个数12，这个自动丢失的数叫做“模”（module，简写为mod）

已知补码求真值：

☎ 已知正数的补码求真值

与原码相同，只要将符号位的0变为＋（正号），即得到它的真值。

☎ 已知负数的补码求真值

方法1：将负数补码的数值位按位取反再加1，将符号位的1变为－（负号），即得到它的真值。

方法2：用公式：$X=-(2^n-[X]_{补})$

已知 补码为 01111111B,其真值为+1111111B=+7FH

已知 补码为 11111111B,其真值为:

10000000B+1= 10000001B,其真值为—01H

或：X= —（ 2^8 — 11111111B)= —(00H-FFH)= —1

(1) 求补运算

对一个二进制数按位取反，最低位加1。(计算机)

等价于： 0 － 该二进制数 (人工计算)

例：对 8 位二进制数 11110001B进行求补运算

方法：按位取反，最低位加1

	1111 0001	B
取反	0000 1110	B
加1	1	
	0000 1111	B

(2) 补码

在计算机中，用补码表示带符号数。

补码的表示方法:

¬ 正数的补码: 最高位为 0,

其它各位为数字位，表示数的大小。

– 负数的补码: 通过对该数正数的补码进行求补运算得到。

负数的补码最高位为 1。

例　求 105D 的补码

```
2 |105
2 | 52   1
2 | 26   0
2 | 13   0
2 |  6   1
2 |  3   0
2 |  1   1
  0      1
```

[105D] 补

= 0110 1001B = 69 H　　(8位)

= 0000 0000 0110 1001 B= 0069 H　(16位)

正数的补码：最高位为0

其它各位为数字位，表示数的大小。

例　求 – 105D的补码

8位：[– 105D] 补 = 0 – [105D] 补

= 0 – 0110 1001B = 0 – 69H

= 10010111B = 97 H

16位：[– 105D] 补 = 0 – [105D] 补

= 0 – 0000 0000 0110 1001B = 0 – 0069H

= 1111 1111 1001 0111B = F F 9 7 H

负数的补码：通过对该数正数的补码进行求补运算得到。

(3) 补码的真值计算

真值：补码表示的数值大小。

如，用DEBUG查看到存放在内存中的一组符号数：

```
— D 2000:0 ↙              ;显示内存块2000: 0~7Fh单元的内容
2000:0000  9E 0F C9 D8 65 04 70 00-16 00 13 08 65 04 70 00
2000:0010  65 04 70 00 54 FF 00 F0-58 7F 00 F0 F5 E7 00 F0
2000:0020  00 00 00 D0 28 00 13 08-6F EF 00 F0 6F EF 00 F0
2000:0030  6F EF 00 F0 6F EF 00 F0-9A 00 13 08 65 04 70 00
2000:0040  07 00 70 D0 4D F8 00 F0-41 F8 00 F0 07 25 61 FD
2000:0050  39 E7 00 F0 40 02 5C 02-2D 04 70 00 28 0A 5C 03
2000:0060  A4 E7 00 F0 2F 00 D4 08-6E FE 00 F0 04 06 5C 03
2000:0070  1D 00 00 D0 A4 F0 00 F0-22 05 00 00 34 12 00 C0
—
```

如何知道它们表示的数值大小？

求补码真值的方法：

¬ 先判断是正数，还是负数。

由最高位判断：0 → 正数

1 → 负数

– 再求数值大小

对正数，补码的真值等于该二进制数值。

对负数，先对该数进行求补运算，再求数值大小。

例　求补码7D H 的真值:

7D H = 0111 1101B ，　最高位为0，是正数

7DH的真值 = 7 × 16 + 13 = 125 D

例　求补码 91H 的真值:

91H = 1001 0001B，　最高位为1 ，是负数。

对91H进行求补运算:

91H　求补→　00H – 91H = 6F H

91H的真值 = – 6FH = – (6 × 16 +15) = – 111D

(4)用补码表示带符号数的意义　　计算机中用补码表示带符号数。

① 将减法用加法实现，省去减法器，简化硬件。

计算机中，减法实现过程:　（补码减法）

¬ 先对减数进行求补运算（求反加1，也是加法）

– 再将求补后的数与被减数相加

® 相加的结果即为用补码表示的两数相减结果。

计算:

```
  01000100B
– 10100101B
```

计算机中实现过程:

```
¬   01011010
  +        1
    01011011
–   + 01000100
®     10011111
```

计算:

```
  01000100B
– 10100101B
```

补码减法:

```
¬   01011010
  +        1
    01011011
–   + 01000100
®     10011111
```

常规减法:

```
    01000100 B
–   10100101 B
借位 1  111111
    10011111 B
```

▲补码减法的计算结果与常规减法的结果相同。

▲人工计算时，可用常规减法

(补码减法，对人来说，相对复杂一些)

② 无符号数及带符号数的加减运算用同一电路完成。

即指令系统中加、减运算不区分无符号数或带符号数。

例: 8位运算器

在计算机中计算	看作无符号数	看作带符号数
1111 0001 + 0000 1100 1111 1101	241 + 12 253	(–15) + (+12) – 3
1111 0001 – 0000 1100 1110 0101	241 – 12 229	(–15) – (+12) – 27

思考：计算机能自动识别无符号数和符号数吗?

凡是能在计算机内存储或参与运算的都是二进制形式的机器数，计算机只能出别“0”和“1”，对于某个二进别致的最高位究竟应看做为符号位还是数值位，理论上是无法自动识别的，只能靠用户在编写程序时做到‘心中有数”。对于8位二进制数．看成字节无符导数，其表示范围是0-255(00H—FFH)；看成字节符号数，其表示范围是-128~+127(80H~7FH)。

计算机所进行的运算都是无符号数运算，既把符号数的符号位当做数值进行运算，又把所有数的运算结果当做符号数来影响溢出标志位。

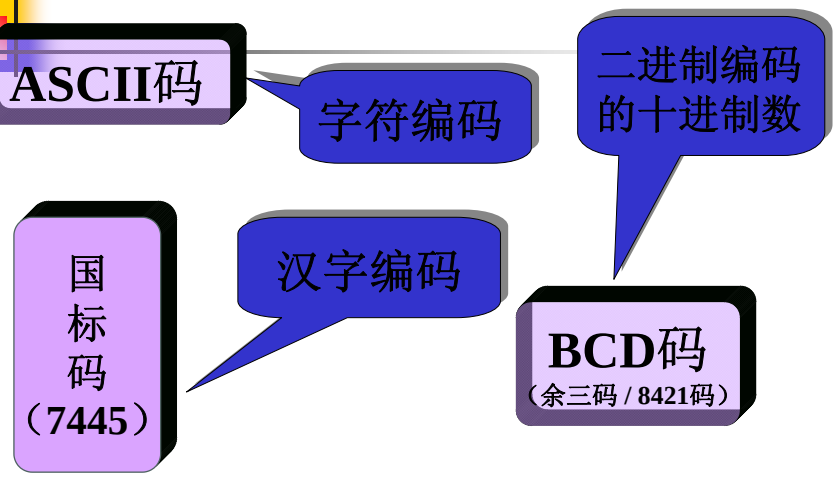
思考：计算机能自动识别无符号数和符号数吗?

但是，由于引入了补码概念，使得计算机在进行无符号数和有符号数的运算时能够实现操作的一致性，且结果合理。例如，将无符号数1FH与D0H相加，或是将符号数1FH与D0H相加，其结果都是EFH。编制无符号数加法程序的用户则会将该结果的真值认为是239，即31+208，而编制符号数加法程序的用户则会将该结果的真值认为是-17，即31+（-48）。

二进制编码

- 计算机处理的信息：数值、字符(字母、汉字等)
- 各字符在计算机中由若干位的二进制数表示
- 二进制数与字符之间一一对应的关系，称字符的二进制编码。

计算机编码



二-十进制(BCD)码

人们通常使用十进制来计数，而计算机则采用二进制，所以必须对十进制的0—9这十个数字进行二进制编码。

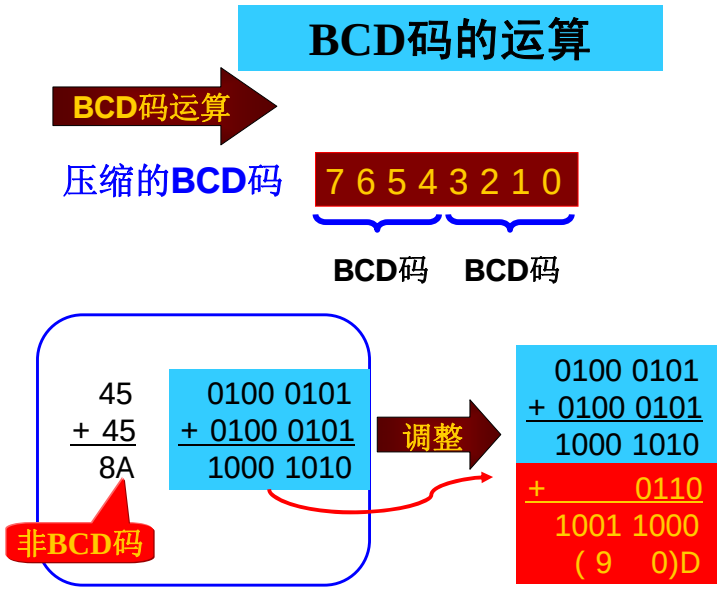
但从二进制数直接看出所对应的十进制数是比较困难的。虽然我们学过二—十进制的转换，但这种转换比较慢，这是一个明显的缺点。为此，人们提出了一个比较适合十进制的二进制码的特殊形式，即二—十进制码，通常用英文字母BCD表示，BCD码具有二进制和十进制两种数制的某些特征。

二-十进制(BCD)码

- BCD码——以二进制数表示十进制数的编码，一般采用4位二进制数来表示l位十进制数(即压缩BCD码)。
- 其特点是：4位之内为二进制关系，每4位之间为十进制关系。
- 用途：方便计算机对十进制数的直接输入、输出、存储及运算.

二-十进制(BCD)码

- BCD码有两种表示法：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。



BCD码的运算

思考:BCD码的加法运算中为什么要加6来调整?

BCD码的权值（Weights）都不相同，分别是8，4，2，1也因此，BCD码又被称为8421码。 BCD码的加减法完全模仿十进制的作法，逢十进位，可是4位元的二进位数字要达到2的4次方＝16才会进位，这中间差6，因此，在BCD码作加的运算时，我们应将其结果再加6，以强迫进位。至于减法，A-B＝A＋（B的10补数），则可用加法來完成减法的运算。

思考:BCD数是十进制数，为何也以H为单位?与十六进制数的区别何在?

BCD数的特点是用4位二进制数编码来表示1位十进制数，那么8位二进制数可以去示两位BCD数。例如十进制数29的BCD数在计算机内存放的二进制码形式为：0010100lB，为了书写方便，再将二进制数的式样写成十六进制数的式样，那就变成了29H。

有的教科书为了将BCD数29与普通十六进制数29H区分开，规定将BCDD数29写成29BCD的形式，但由于它在计算机中的存放形式仍然是00101001B，用十六进制数来记忆和书写仍然是29H，所以很少有人用麻烦的“BCD”书写形式，而是约定俗成地沿用“H“这个单位。

思考:BCD数是十进制数，为何也以H为单位?与十六进制数的区别何在?

也就是说，十进制数29的BCD数本应该写成29BCD，但因为它在计算机内存放为二进制数00101001B，即十六进制数29H，所以我们可以简单地把任意一个两位十进制数XY写成机内表示形式：XYH。

或许有人会问，当计算机内放置的是29H时，它到底是普通的十六进制数呢，还是BCD数29?这要看当初这个数据是以何种“身份”存入的，或者要根据程序的上下文来判断。

例如，欲让计算机来执行29+18＝?，可以先将两个加数分别转为二进制数(1DH和12H)之后，再调用二进制数的加法程序计算1DH+12H，最后再将结果转回十进制数；也可以用BCD形式直接存入两个加数29H和18H，再调用BCD数的加法程序来完成29H+18H。这时的结果就是BCD数，不必再转换了。

ASCII码(美国标准信息交换码)

- American Standard Code for Inform ter
- 采用7位二进制代码对字符进行编码

- ASCII码是一种8位代码，最高位一般用于奇偶校验，用7值代码来对128个字符编码，其中32个是控制字符，96个是图形字符。

ASCII（美国标准信息交换码）

微机中普遍采用的字符编码，如键盘、打印机、显示器等

- **数字0~9的编码是0110000~0111001 (30H~39H)，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。**

ASCII码—美国标准信息交换代码

ASCII 字符表

H / L	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	ENG	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

注：H 表示高 3 位，L 表示低 4 位。

课堂作业：

- 1、计算机中常用的码制有原码、反码和_____。
- 2、-54的原码是_____，反码是_____，补码是____；
- 3、十六进制数0－9,A－F对应的ASCII码 是（　　）