

# 大学计算机基础

教学课件

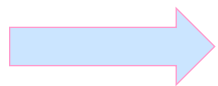
北京航空航天大学

# 课程回顾：第1次课程

- ▶ 计算机的特点、分类及应用范围
- ▶ 进位计数制
- ▶ 其它数制转换成十进制的方法
- ▶ 因故未完成：随堂测验与考勤的成绩。本次上课前先补测，然后再观看后面的课程视频。

# 十进制转换为R进制

十进制



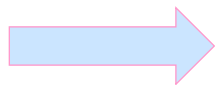
R进制

方法

- (1) 整数转换用“除基取余法”，直到商为零；每次相除所得余数为对应的R进制整数的各位数码。
- (2) 小数转换用“乘基取整法”，直到乘积的小数部分为零，或达到所要求的位数（当小数部分永不可能为零时）。

# 十进制转换为二进制

十进制



二进制

方法

- (1) 整数转换用“除2取余法”，直到商为零；每次相除所得余数为对应的二进制整数的各位数码。
- (2) 小数转换用“乘2取整法”，直到乘积的小数部分为零，或达到所要求的位数（当小数部分永不可能为零时）。

# 十进制数 125.6875 转换成二进制数

2	┌ 125
2	┌ 62
2	┌ 31
2	┌ 15
2	┌ 7
2	┌ 3
2	┌ 1

0

商为零

余数

1

0

1

1

1

1

1

二进制整数低位

二进制整数高位

结果

故整数部分 $(125)_D = (1111101)_B$

# 小数部分的转换

取整

二进制小数首位

1

0

1

1

**结果**

二进制小数末位

故小数部分  $(.6875)_D = (.1011)_B$

0.6875

$$\begin{array}{r} \times ) \quad 2 \\ \hline 1.3750 \\ 0.3750 \\ \times ) \quad 2 \\ \hline 0.7500 \\ 0.7500 \\ \times ) \quad 2 \\ \hline 1.5000 \\ 0.5000 \\ \times ) \quad 2 \\ \hline 1.0000 \\ 0000 \end{array}$$

为零，转换  
结束

# 十进制转换为十六进制

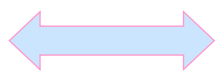
将十进制数 ( **197.734375** )<sub>10</sub> 转换成十六进制数

结果？

# 二进制与八进制、十六进制的转换

首先

二进制



八进制

方法

- (1) 二进制数转换成八进制数：以小数点为界，向左(小数点之前)或向右(小数点之后)每**3**位二进制数用相应的一位八进制数取代（不足**3**位的二进制数先用**0**补足）。
- (2) 八进制数转换成二进制数：以小数点为界，向左或向右每一位八进制数用相应的**3**位二进制数取代；如果不足**3**位，则用零补足。

一位八进制数对应着**3**位二进制数



# 二进制转换为八进制（例）

将二进制数**1101101110.11011**转换成八进制数

结果？

八	对应二	十六	对应二	十六	对 应 二
0	000	0	0000	8	1000
1	001	1	0001	9	1001
2	010	2	0010	A	1010
3	011	3	0011	B	1011
4	100	4	0100	C	1100
5	101	5	0101	D	1101
6	110	6	0110	E	1110
7	111	7	0111	F	1111

# 八进制转换为二进制（例）

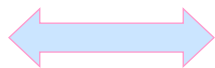
将八进制数**135.23**和**7123.56**转换成二进制数



结果？

# 二进制与十六进制转换

二进制



十六进制

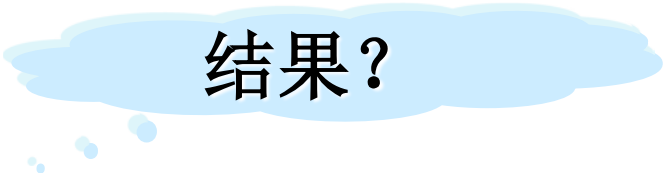
方法

- (1) 二进制数转换成十六进制数: 以小数点为界, 向左(小数点之前)或向右(小数点之后)每4位二进制数用相应的一位十六进制数取代(不足4位的二进制数先用0补足)。
- (2) 十六进制数转换成二进制数: 以小数点为界, 向左或向右每一位十六进制数用相应的4位二进制数取代; 如果不足4位, 则用零补足。

一位十六进制数对应着4位二进制数

# 二进制转换为十六进制（例）

将二进制数**1101101110.11011**转换成十六进制数



结果？

# 十六进制转换为二进制（例）

将十六进制数**6A.B1**和**2C1D.6**转换成二进制数



结果？

# 二进制数的运算

◆ 二进制加法运算：逢二进一

$$\begin{array}{r} 1011010011 \\ + \quad 111010111 \\ \hline 10010101010 \end{array}$$

# 二进制数的三种逻辑运算

## ◆ 与 (AND)

$1 \text{ AND } 1 = 1$ ,  $1 \text{ AND } 0 = 0$ ,  $0 \text{ AND } 1 = 0$ ,  $0 \text{ AND } 0 = 0$ 。即当两个参加“与”运算的逻辑变量都为“1”时，逻辑积才为“1”，否则为“0”。

## ◆ 或 (OR)

$1 \text{ OR } 1 = 1$ ,  $1 \text{ OR } 0 = 1$ ,  $0 \text{ OR } 1 = 1$ ,  $0 \text{ OR } 0 = 0$ 。即当两个参加“或”运算的逻辑变量都为“0”时，逻辑和才为“0”，否则为“1”。

## ◆ 非 (NOT)

对单一的逻辑变量进行求反运算。意思是将一个二进制数据的0变为1，1变为0。

# 二进制数的三种逻辑运算（例）

$$\begin{array}{r} \text{AND} \quad 1011010111 \\ \quad 1110010101 \\ \hline 1010010101 \end{array}$$

$$\begin{array}{r} \text{OR} \quad 1011010111 \\ \quad 1110010101 \\ \hline 1111010111 \end{array}$$

$$\begin{array}{r} \text{NOT} \quad 1111010111 \\ \hline 0000101000 \end{array}$$



# 数值的编码

机器数

用0或1表示正负号的数

真值数

机器数对应的实际数值，  
也称尾数。

例：真值数  $(-1001101)_B$   
其机器数为11001101，  
存放在计算机中。

# 问题的提出

符号位参加运算， $-5+4 = ?$

$$\begin{array}{r} \begin{array}{|c|c|c|c|c|c|c|c|} \hline 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ \hline \end{array} \\ + \begin{array}{|c|c|c|c|c|c|c|c|} \hline 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ \hline \end{array} \\ \hline = \begin{array}{|c|c|c|c|c|c|c|c|} \hline 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ \hline \end{array} \longrightarrow \text{-9} \text{ 错误的结果} \end{array}$$

为解决此类问题，提出了：  
原码、反码和补码

# 原码表示

原码表示相当于：

1. 正数的符号位是**0**，负数的符号位是**1**。
2. 数值位就是这个数的绝对值的二进制表示。

原码表示法简单、易懂，与真值的转换方便。  
缺点：加减法运算复杂。

# 原码表示（例）

以8位原码表示下列各数： $(0.25)_D$ 、  
 $(-0.8125)_D$ 、 $(228)_D$ 、 $(-12)_D$

$$(0.25)_D = (0.01)_B =$$

$$(-0.8125)_D = (-0.1101)_B =$$

$$(228)_D = (11100100)_B =$$

$$(-12)_D = (-1100)_B =$$

# 原码表示说明

原码0的表示有两种：（以4位带符号定点整数为例）  
正0：0000      负0：1000

4位带符号定点整数的原码表示范围：

1111~0111，即 -7~7

也就是  $-2^{4-1}+1 \leq X \leq 2^{4-1}-1$

n位带符号定点整数原码表示范围：

$-2^{n-1}+1 \leq X \leq 2^{n-1}-1$

# 反码表示

反码表示相当于：

1. 正数的符号位是**0**，负数的符号位是**1**；
2. 正数的数值位同原码相同，  
负数的数值位将原码的数值位各位取反。

# 反码表示（例）

以8位反码表示下列各数： $(0.25)_D$ 、 $(-0.8125)_D$ 、 $(228)_D$ 、 $(-12)_D$

$$(0.25)_D = (0.01)_B = (0\ 0100000)_{\text{原}}$$

$$(-0.8125)_D = (-0.1101)_B = (1\ 1101000)_{\text{原}}$$

$$(228)_D = (11100100)_B =$$

8位反码表示范围：  
 $-127 \leq X \leq +127$

$$(-12)_D = (-1100)_B = (10001100)_{\text{原}}$$

# 反码表示的说明

反码**0**的表示有两种：（以4位带符号定点整数为例）

正**0**：0000      负**0**：1111

4位带符号定点整数的反码表示范围：

1111~0111，即 -7~7

也就是  $-2^{4-1}+1 \leq X \leq 2^{4-1}-1$

n位带符号定点整数的反码表示范围：

$-2^{n-1}+1 \leq X \leq 2^{n-1}-1$



# 补码表示

补码表示相当于：

1. 正数的符号位是**0**，负数的符号位是**1**；
2. 正数的数值位同原码相同，负数的数值位将反码的数值位**+1**。

# 补码表示（例）

以8位补码表示下列各数： $(0.25)_D$   
 $(-0.8125)_D$   $(228)_D$   $(-12)_D$

$$(0.25)_D = (0.01)_B = (0\ 0100000)_{\text{反}}$$

$$(-0.8125)_D = (-0.1101)_B = (1\ 0010111)_{\text{反}}$$

$$(228)_D = (11100100)_B =$$

8位补码表示范围：  
 $-128 \leq X \leq +127$

$$(-12)_D = (-1100)_B = (11110011)_{\text{反}}$$

# 补码表示的说明

补码0的表示只有一种：  
以4位带符号定点整数为例 0000

4位带符号定点整数的补码表示范围：

1000~ 0111，即 -8~7

也就是  $-2^{4-1} \leq X \leq 2^{4-1}-1$

n位带符号定点整数的补码表示范围：

$-2^{n-1} \leq X \leq 2^{n-1}-1$

# 补码运算（例1）

-5的原码、反码

1	0	0	0	0	1	0	1
---	---	---	---	---	---	---	---

1	1	1	1	1	0	1	0
---	---	---	---	---	---	---	---

$$-5+4 = ?$$

	1	1	1	1	1	0	1	1
+	0	0	0	0	0	1	0	0
=	1	1	1	1	1	1	1	1

-5

4

-1

除符号位外取反加1则为原码 10000001

# 补码运算（例2）

$$(-9) + (-5) = ?$$

1	1	1	1	0	1	1	1
---	---	---	---	---	---	---	---

-9 的补码

1	1	1	1	1	0	1	1
---	---	---	---	---	---	---	---

-5 的补码

+

1	1	1	1	0	0	1	0
---	---	---	---	---	---	---	---

-14 的补码

进位丢弃

补码可方便地实现正、负数的加法运算。

补码0的表示只有一种

补码规则简单，符号位如同数值一样参加运算。

补码被广泛应用。

# 几种表示的比较

(1) 原码					
$[X]_{\text{原}} =$	0	$X$	$0 \leq X$	+127: 01111111	+0: 00000000
	1	$ X $	$X \leq 0$	-127: 11111111	-0: 10000000
(2) 反码					
$[X]_{\text{反}} =$	0	$X$	$0 \leq X$	+127: 01111111	+0: 00000000
	1	$ X $	$X \leq 0$	-127: 10000000	-0: 11111111
(3) 补码					
$[X]_{\text{补}} =$	0	$X$	$0 \leq X$	+127: 01111111	0: 00000000
	1	$ X  + 1$	$X \leq 0$	-127: 10000001	-128: 10000000

$$(+0)_{\text{补}} = (-0)_{\text{补}} = 00000000$$

# 原码、反码和补码

	符号位	数值位
原码	1表负数 0表正数	该数绝对值的二进制表示
反码		正数同原码，负数为原码的各位取反
补码		正数同原码，负数为反码的末尾+1

- 原码、反码、补码优缺点
- 整数表示范围
- 0 的表示

# 第2次课程小结

- ▶ 十进制转换成其它数制的方法
- ▶ 二进制与八进制、十六进制的转换
- ▶ 原码、反码和补码的表示与特点
- ▶ 需要完成：随堂测试及考勤



# 本次课程的测试

- ▶ 观看上课视频后，上课时间完成“随堂测验&考勤2-20210928”
- ▶ 测试后，同时记录考勤成绩（请通过查看分数，确认考勤成功）