华南理工大学

South China University of Technology

计算机与软件工程概论

Dr 林育蓓

yupilin@scut.edu.cn

软件学院

2019

主要内容

- 计算机的数制
- 数据的存储与表示
- 数据运算

计算机的数制

- 数值按预定形式在计算机中**表达**和**存储**，并按预定的规则进行**运算**。
  - 日常生活：**十进制**记数法
  - 计算机领域：**二进制**、八进制和十六进制记数法

十进制记数法

- 数码：

0	1	2	3	4	5	6	7	8	9

- 规则：逢10进1
- 展开：

$$1024 = 1 \times 10^3 + 0 \times 10^2 \times 2 \times 10^1 + 4 \times 10^0$$

- 十进制记数法是进位记数制的一种

  如何定义进制记数制？

进位记数制

- 基数：某种数制中使用的**数码**的个数
  - 十进制：10

0	1	2	3	4	5	6	7	8	9

  - 二进制：2

0	1

  - 八进制：8

0	1	2	3	4	5	6	7

  - 十六进制：16

0	1	2	3
4	5	6	7
8	9	A	B
C	D	E	F

$$(2048)_{10}$$
$$(110)_{2}$$
$$(1024)_{8}$$
$$(10AD)_{16}$$

进位记数制

- **例 2-1** 任意给定一个二进制数$(11011.101)_2$，这个数的各位权表示如下：

数	1	1	0	1	1	1	0	1
数位	4	3	2	1	0	-1	-2	-3
权	2^4	2^3	2^2	2^1	2^0	2^{-1}	2^{-2}	2^{-3}

$$1\times 2^4+1\times 2^3+0\times 2^2+1\times 2^1+1\times 2^0$$
$$+1\times 2^{-1}0\times 2^{-2}+1\times 2^{-3}$$

数制之间的转换

- 用户编程与书写：
  - 十进制、十六进制和八进制
- 计算机中存储和处理的数据通常为**二进制数**

$(400)_{16}$ $(2000)_8$ $(1024)_{10}$

$(10000001)_2$

将R进制转换成十进制

- 将R进制数**按位权展开**，再按**十进制运算规则**运算即可。
- **例 2-2** 将二进制数$(11011.101)_2$转换为十进制数。
- $(11011.101)_2$
  $=1\times2^4+1\times2^3+0\times2^2+1\times2^1+1\times2^0+1\times2^{-1}+0\times2^{-2}+1\times2^{-3}=(27.875)_{10}$
- 问题：$(101.11)_2$ 和$(10000000000)_2$？

将十进制转换成R进制

- **整数部分**和**小数部分**分别按不同规则转换
- 整数部分转换：
  - 除基取余：采用逐次除以基数R取余数的方法

其步骤如下：

(1) 将给定的十进制整数除以R，余数作为R进制数的最低位。

(2) 将前一步的商再除以R，余数作为次低位。

(3) 重复(2)步骤，记下余数，直至最后商为0，最后的余数即为R进制的最高位。

*	*	*	.	*	*	*
3	2	1	0	1	2	3
高位 ←				→ 低位		
M^3	M^2	M^1	M^0	M^{-1}	M^{-2}	M^{-3}

将十进制转换成R进制

- 小数部分转换
  - 乘基取整：将逐次乘以基数R取整数的方法

其步骤如下：

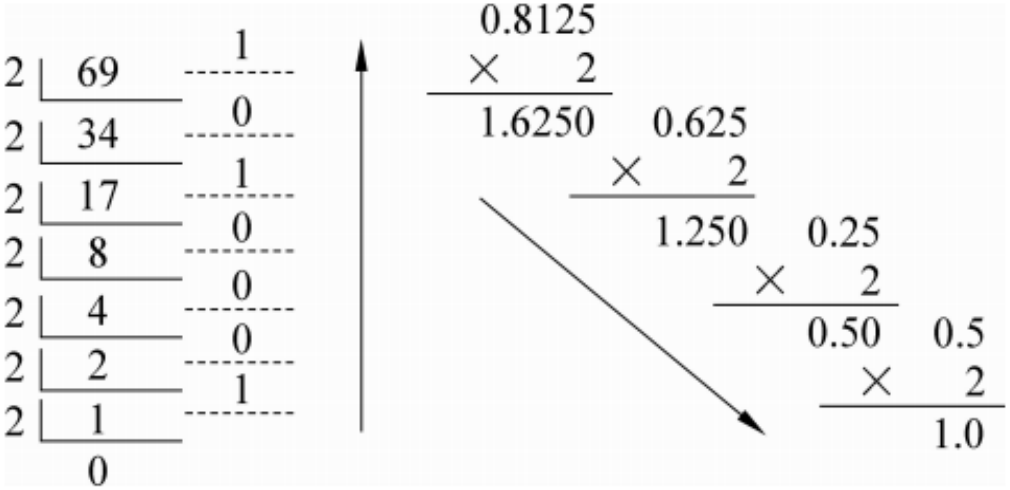
(1) 将给定的十进制数的纯小数部分乘以R，取乘积的整数部分作为R进制的最高位。

(2) 将前一步的乘积的小数部分继续乘以R，取乘积的整数部分作为R进制的次高位。

(3) 重复(2)步骤，记下整数，直至最后乘积为0或达到一定的精度为止。

将十进制转换成二进制

- **例 2-5** 把十进制数$(69.8125)_{10}$转换为二进制数。



$(69)_{10} = (1000101)_2$ $(0.8125)_{10} = (0.1101)_2$

$(69.8125)_{10} = (1000101.1101)_2$

- **问题：$(69.6)_{10}$？**

将十进制转换成二进制

- **例** 把十进制数$(0.243)_{10}$转换为二进制数。

Fraction: 0.486 0.972 1.944 1.888 1.776 1.552 1.104 …

Binary: 0. 0 0 1 1 1 1 …

➢存储单元的位数，决定二进制的精度！

数值类型

- 常用的数值类型
  - 整数和实数
- 整数
  - 无符号整数： 0～+∞
  - 有符号整数： -∞ ～+∞
- 为了有效地利用计算机的存储空间，**无符号整数**和**有符号整数**在计算机中的存储与表示方式是不同的

存储无符号整数

- N位：计算机分配用于存储与表示一个无符号整数的二进制位数，即存储单元大小为N
  - 范围：$0\sim(2^N-1)$
- 在计算机中存储无符号整数需要两个步骤。
  - (1) 首先将整数变成二进制数。
  - (2) 如果二进制位数不足N位，则在二进制整数的左边补0，使它的总位数为N位。

N-1	N-2			……			4	3	2	1	0
1	0	1	1	1	0	1	0	0	1	0	1

存储无符号整数

- **例2-13** 将7存储在8位存储单元中

$(7)_{10}$							
$(111)_2$							
?	?	?	?	?	1	1	1
0	0	0	0	0	1	1	1
7	6	5	4	3	2	1	0

译解无符号整数

- 将计算机内存中**位模式**的无符号整数，并将之转换为一个十进制数，这个过程称为无符号整数的译解
- 例子：N=8

7	6	5	4	3	2	1	0
0	**0**	**1**	**0**	**1**	**0**	**1**	**1**
$(101011)_2$							
$(43)_{10}$							

无符号整数的溢出

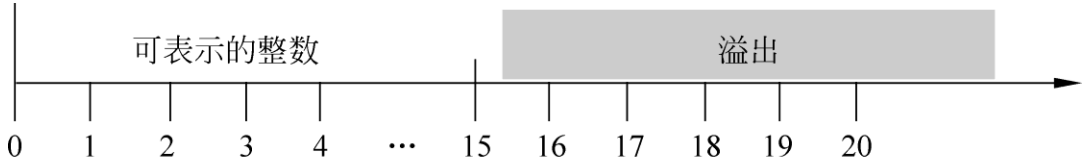
- N位存储单元：$0\sim(2^N-1)$
- 当存储超出范围的最大整数时，会发生溢出
  - 溢出的影响？
- 假设N=4，范围：$0\sim(2^4-1)$即0~15，如果存储$(16)_{10}$？

	$(16)_{10}$			
	$(10000)_2$			
1	0	0	0	0
4	3	2	1	0

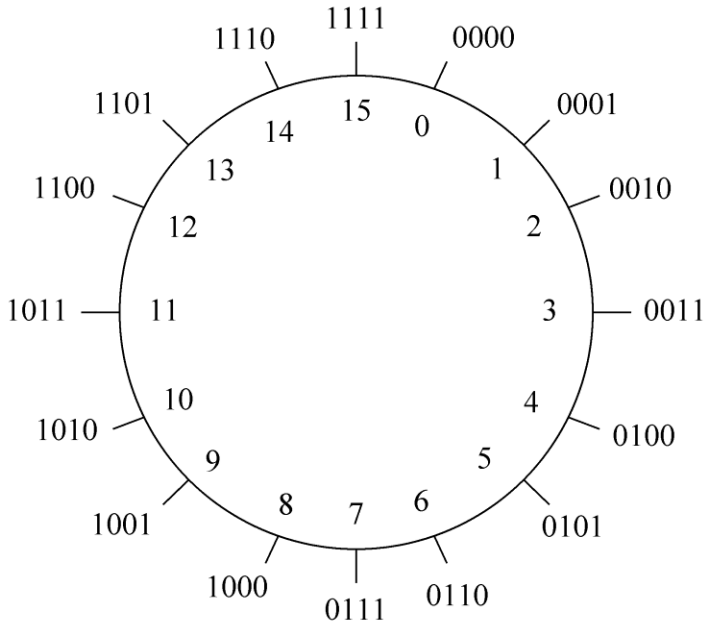
  - 溢出发生；译解时，得到$(0)_{10}$

无符号整数的溢出

- 如果N=4，那么



(a)



(b)

无符号整数的应用

- 因为不必存储整数的符号，所有分配单元都可以用来存储数字，所以无符号整数表示法可以**提高存储的效率**。
- 只要没有负整数的应用，就可以使用无符号整数表示法。
- 常见的应用有**计数、寻址**等。

存储有符号整数

- 给定N位存储单元，如何存储**符号**与**数值**？

有符号整数	
±	12358
符号	数值部分

- 为了区别符号和数值，同时又便于计算，人们对有符号整数进行了合理的编码。常见的有**原码、反码和补码**三种编码方式。
  - 给定一个数（二进制或者十进制）可求其原码、反码和补码

- 参考材料[1]和[2]

二进制补码系统

- 现在计算机中普遍采用补码表示有符号整数
- 二进制整数的补码运算有两种实现：
  - (1) 先对二进制整数序列从右边复制，直到有1被复制，然后对其余各位取反。
  - (2) 按位取反，并在最低位加1。

二进制整数补码运算

- **例2-18** 取整数10110110的补码。
  - 原模式：10110110
  - 进行补码运算： 11001010
- **例2-19** 对整数00110110进行两次补码运算。
  - 原模式： 00110110
  - 第一次补码运算： 11001010
  - 第二次补码运算： 00110110
- 可见，**对一个整数进行两次补码运算，就可以得到原先的整数。**

以补码存储整数

- N位存储单元
  - 范围：$-(2^{N-1})\sim 0\sim +(2^{N-1}-1)$ (256个)
  - 非负数：存储其二进制编码的原码
  - 负数：存储其二进制编码的补码
  - 0补码1个
- 假设N=8，

-128	1	0	0	0	0	0	0	0
-127	1	0	0	0	0	0	0	1
……	……							
-2	1	1	1	1	1	1	1	0
-1	1	1	1	1	1	1	1	1
±0	0	0	0	0	0	0	0	0
+1	0	0	0	0	0	0	0	1
……	……							
+127	0	1	1	1	1	1	1	1

译解补码

- 以二进制补码格式还原整数
  - (1) 如果最左位是1，取其补码，如果最左位是0，不操作。
  - (2)将该整数转换为十进制。
  - (3) 添加符号。

译解补码

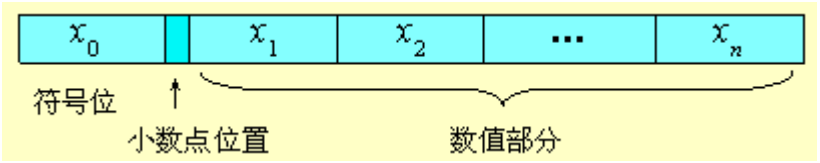
- **例2-23** 用二进制补码表示法将存储在8位存储单元中的11101010 还原成整数。
  - 最左位是1，因此符号为负。该整数需要在转换为十进制前进行补码运算。
  - 最左位是1，符号为负： 11101010
  - 进行补码运算： 00010110
  - 转换为十进制： 22
  - 加上符号： -22
  - 即$(11101010)_{补}=(-22)_{10}$
- 问题：00001110？

补码的优点

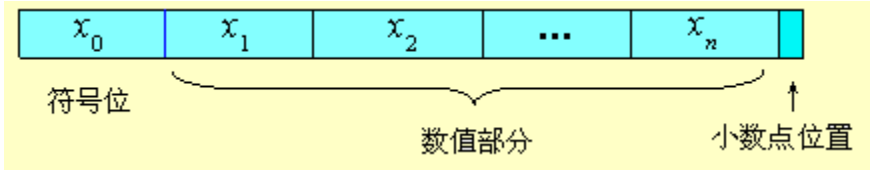
- 用补码表示有符号整数具有两个突出的优点。
  - (1) 对任意的正、负整数，可以不加区分地进行机械式的加法运算。
  - (2) 可以用减法转化为加法，减去一个数等同于加上这个数的相反数。

实数的存储与表示

- 实数：符号、整数部分和小数部分
- 两种表示方法
  - 定点表示法：小数点固定在一个位置
  - 浮点表示法：小数点位置浮动



定点小数

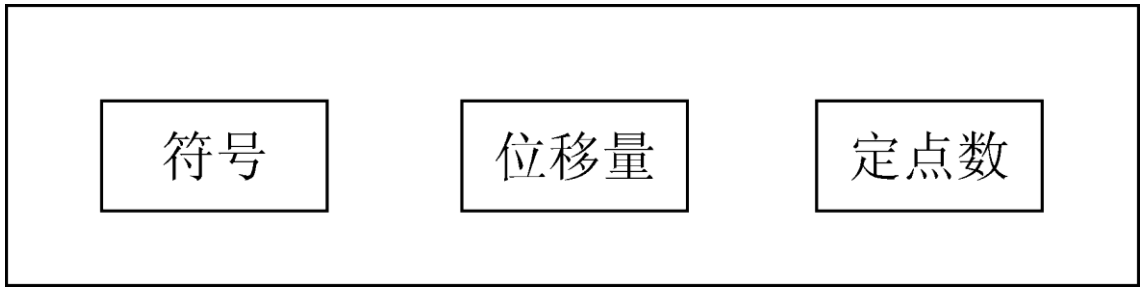


定点整数

- 计算机常用浮点表示法表示实数

浮点表示法

- 浮点表示法允许小数点浮动，即可以在小数点的左右有不同数量的数码
- 一个数字（十进制或二进制）由三部分组成：
  - **符号**：可正可负
  - 小数点位置固定的**定点表示法**
  - 小数点应该左右移动构成实际数字的**位移量**



浮点表示法

- 十进制科学计数法中，定点部分在小数点左边只有一个数码，并且位移量是10的幂次。
- **例2-25** 用科学记数法表示

  -0.00000000000316

  – 实际数字： -0.00000000000316
  – 十进制科学记数法： -3.16×10^{-12}
  – 在这个例子中，这个数字的三个部分分别是：符号(-)、位移量(-12)、定点部分(3.16)。

浮点表示法

- 规范化
  - 为了使表示法的固定部分统一，十进制科学记数法和浮点表示法（即二进制科学记数法）都**在小数点左边使用了唯一的非零数码**，这称为规范化。十进制系统中的这位非零数码可能是1~9，而二进制系统中该数码是1。则定点部分的表示方法都可以规范为以下形式：
  - 十进制 ±d.xxxxxxxxxxxxxx 注意： d是1~9，每个x是0~9
  - 二进制 ±1.yyyyyyyyyyyyyy 注意： 每个y是0或1。

浮点表示法

- 符号、指数和尾数
  - 在一个二进制数用浮点法表示并规范化之后，可以只存储该数的三部分信息：**符号S、指数E和尾数M（小数点右边的位）**
- **例如，** $+(1000111.0101)_2$规范化后变成为：

$+1.0001110101 \times 2^6$。

可以只存储以下三部分以表示这个数。

符号S： +

指数E： 6 (通常用余码表示：偏移量）

尾数M：0001110101

问题：使用多少位存储E和M？标准？

余码

- 对N位的存储单元，选取一个“魔数”(magic number, M)为2^{N-1}或者2^{N-1}-1
  - N=8，M=128，余128码：-128 ~ 0 ~ +127
  - N=8，M=127，余127码：-127 ~ 0 ~ +128

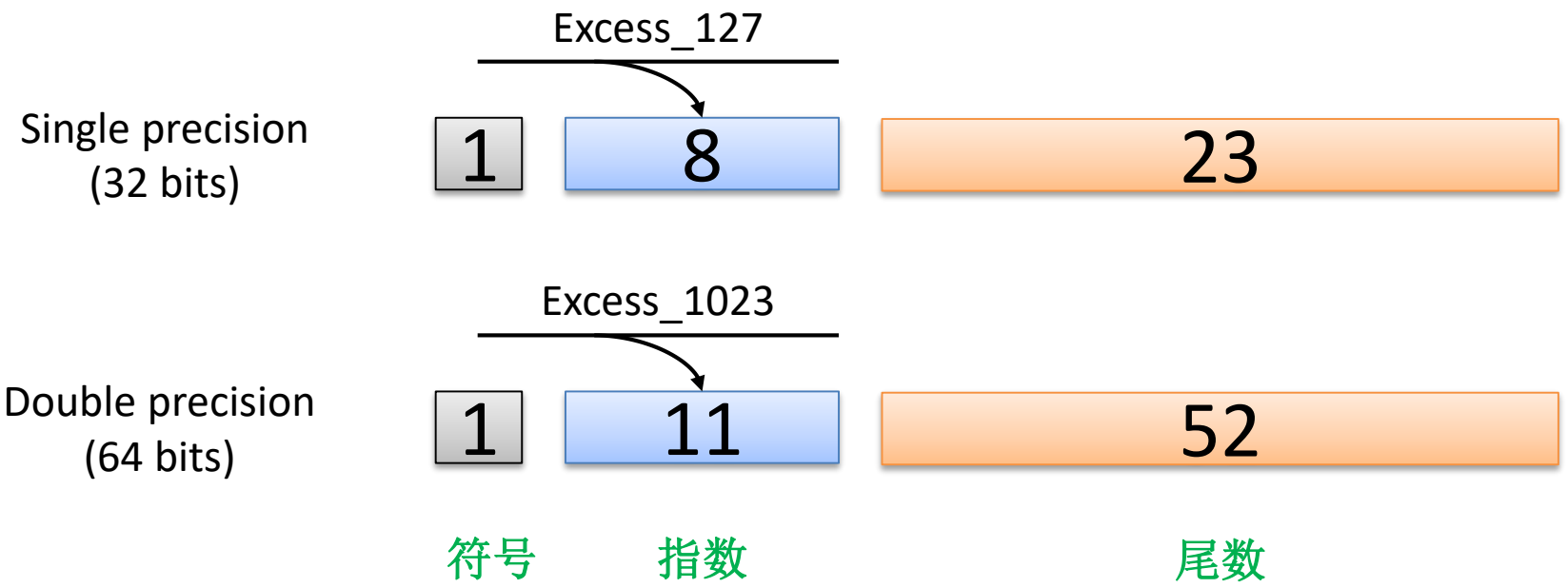
用余码表示整数

- 整数表示为余码：
  1. 整数值加上魔数M
  2. 将（1）的结果转换成二进制，如果不足N位，左边补0
- 余码译码为整数
  1. 将余码（二进制）转换成十进制数
  2. 将（1）的结果减去魔数M

Decimal	Excess_127	Excess_128
+25	1001 1000	1001 1001
-25	0110 0100	0110 0101

IEEE标准浮点数

- 美国电气和电子工程师协会(IEEE)定义了几种存储浮点数的标准。最常用的是单精度和双精度两种类型。



IEEE标准浮点数

- 一个十进制实数可以通过以下步骤存储为IEEE标准浮点数格式。
  - (1) 在符号位S中存储符号(0或1)。
  - (2) 将数字转换为二进制。
  - (3) 规范化。
  - (4) 计算指数E和尾数M的值。
  - (5) 连接符号位S、指数E和尾数M，即为IEEE标准浮点数存储格式。

IEEE标准浮点数

- **例2-28** 写出十进制数5.75的单精度(余127码)表示法。
  - （1） 符号为正，所以S=0
  - （2） 十进制转换为二进制： $5.75=(101.11)_2$
  - （3） 规范化： $(101.11)_2=(1.0111)_2\times 2^2$
  - （4） $E=2+127=129=(10000001)_2$
  - （5）M=0111。需要在M的右边增加19个0使之成为23位

S[1]	E[8]	M[23]
0	10000001	01110000000000000000000