计算机组成与系统结构

吕昕晨

lvxinchen@bupt.edu.cn

网络空间安全学院

第二章 运算方法和运算器

2.1 数据与文字的表示

2.2 定点加法、减法运算

2.5.1 逻辑运算

2.1数据与文字的表示方法

2.1.1数据格式

2.1.2数的机器码表示

2.1.3字符与字符串的表示方法

2.1.4汉字的表示方法

2.1.5校验码

2.1数据与文字的表示方法

- 计算机中使用的数据可分成两大类：
  - 符号数据:非数字符号的表示（ASCII、汉字、图形等)
  - 数值数据:数字数据的表示方式（定点、浮点）
- 计算机数字和字符的表示方法应有利于数据的存储、加工(处理)、传送；
- 编码：用少量、简单的基本符号，选择合适的规则表示尽量多的信息，同时利于信息处理（速度、方便)

2.1.1数据格式

一、定点表示法

- 所有数据的小数点位置固定不变
- 理论上位置可以任意，但实际上将数据表示有两种方法（小数点位置固定-定点表示法/定点格式）：
  - 纯小数
  - 纯整数

2.1.1数据格式

1、定点纯小数

$$\underline{x_0}\ \underline{x_1\, x_2\, x_3 \cdots x_{n-1}\, x_n}$$

符号

量值

小数点固定于符号位之后，
不需专门存放位置

2.1.1数据格式

2、纯小数的表示范围

x=0.00…0 x=1.00…0	x=0	正0和负0都是0
x=0.11…1	$x=1-2^{-n}$	最大正数
x=0.00…01	$x=2^{-n}$	最接近0的正数
x=1.00…01	$x=-2^{-n}$	最接近0的负数
x=1.11…1	$x=-(1-2^{-n})$	最小负数

2.1.1数据格式

3、定点纯整数

$$x_0\ x_1\ x_2\ x_3 \cdots x_{n-1}\ x_n$$

符号

量值

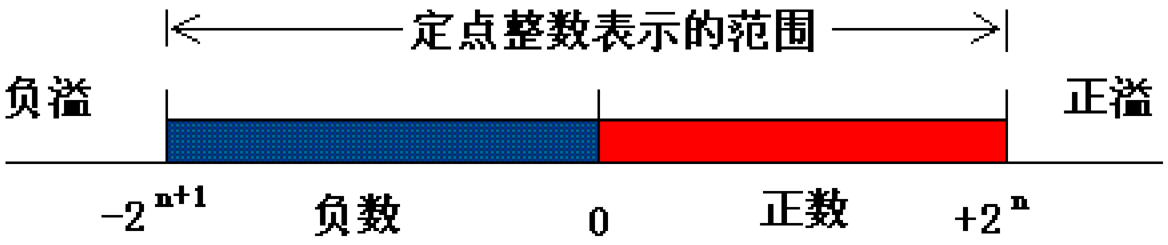
小数点固定于最后一位之后，不需专门存放位置

表示数的范围是 $-(2^n-1) \le X \le +(2^n-1)$

2.1.1数据格式

4、定点表示法的特点

- 定点数表示数的范围受字长限制，表示数的范围有限;
- 定点表示的精度有限
- 机器中，常用定点纯整数表示;



[例] 设机器字长16位,定点表示,尾数15位

(1)定点原码整数表示时,最大正数是多少?最小负数是多少?

0 111 111 111 111 111 最大正整数

$x = (2^{15} - 1)_{10} = (+32767)_{10}$

1 111 111 111 111 111 最小负整数

$x = (1 - 2^{15})_{10} = -(2^{15} - 1)_{10} = (-32767)_{10}$

(2)定点原码小数表示，最大正数是多少?最小负数是多少?

0 111 111 111 111 111 最大正小数

$x = (1 - 2^{15})_{10}$

1 111 111 111 111 111 最小负小数

$x = -(1 - 2^{-15})_{10}$

2.1.1数据格式

2、浮点表示法

电子质量(克)： $9\times10^{-28} = 0.9\times10^{-27}$

太阳质量(克)： $2\times10^{33} = 0.2\times10^{34}$

2.1.1数据格式

2、浮点表示：小数点位置随阶码不同而浮动

1、格式:$N=R^{E}.M$

基数R,取固定的值,2，隐含表示

指数E

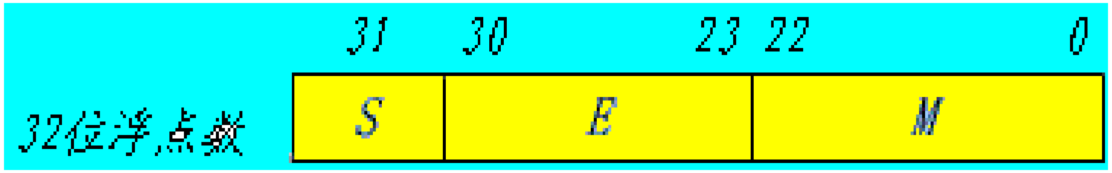
尾数M

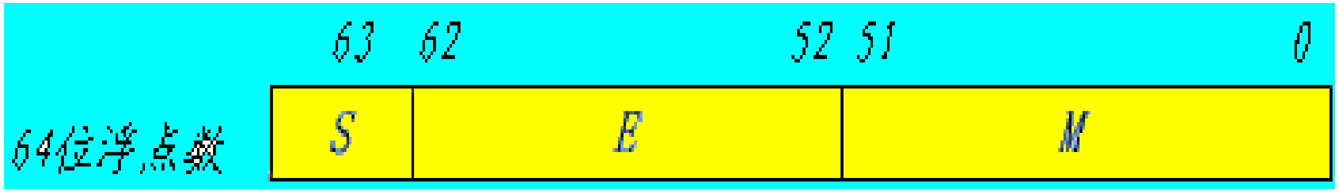
2、机器中表示

阶符	阶码	数符	尾数

2.1.1数据格式

- IEEE754标准(规定了浮点数的表示格式,运算规则等)
  - 规则规定了单精度(32)和双精度(64)的基本格式.
  - 规则中,尾数用原码,指数用移码(便于对阶和比较)





2.1.1数据格式

IEEE754标准

- 基数R=2，基数固定，采用隐含方式来表示它。
- 32位的浮点数：
  - S数的符号位，1位，在最高位，“0”表示正数，“1”表示负数。
  - M是尾数，23位，在低位部分，采用纯小数表示
  - E是阶码，8位，采用移码表示。移码比较大小方便。
  - 规格化：
    - 尾数域最左位(最高有效位)总是1，故这一位经常不予存储，而认为隐藏在小数点的左边。
    - 采用这种方式时，将浮点数的指数真值e变成阶码E时，应将指数e加上一个固定的偏移值127(01111111)，即E=e+127。

2.1.1数据格式

例1 若浮点数x的754标准存储格式为$(41360000)_{16}$，求其浮点数的十进制数值。

解：将16进制数展开后，可得二制数格式为

0 100/0001/0 011/0110/0000/0000/0000/0000

S 阶码(8位) 尾数(23位)

指数e=阶码-127=10000010-01111111=00000011=$(3)_{10}$

包括隐藏位1的尾数

1.M=1.011 0110 0000 0000 0000 0000=1.011011

于是有

$x=+(S==0)1.M\times 2^{e}=+(1.011011)\times 2^{3}=+1011.011=(11.375)_{10}$

2.1.1数据格式

例2 将数$(20.59375)_{10}$转换成754标准的32位浮点数的二进制存储格式。

解:首先分别将整数和分数部分转换成二进制数：

20.59375=10100.10011

然后移动小数点，使其在第1，2位之间

$10100.10011=1.010010011\times 2^4$

e=4于是得到：

S=0, E=4+127=131, M=010010011

最后得到32位浮点数的二进制存储格式为：

$01000001101001001100000000000000=(41A4C000)_{16}$

2.1.2数的机器码表示

- 真值:一般书写的数
- 机器码:机器中表示的数, 要解决在计算机内部数的正、负符号和小数点运算问题。
  - 原码
  - 反码
  - 补码
  - 移码

原码、反码、补码表示法

- 原码：符号位加上真值的绝对值。
- 反码：正数的反码是其本身；负数的反码是在其原码的基础上，符号位不变，其余各位取反
- 补码：正数的补码就是其本身；负数的补码是在反码的基础上+1。

填空题 6分

设置

[+1]的原码、反码、补码分别为（按序）：[填空1]、[填空2]、[填空3]；

[-1]的原码、反码、补码分别为（按序）：[填空4]、[填空5]、[填空6]。

正常使用填空题需3.0以上版本雨课堂

作答

1、原码表示法

- 定点小数$x_0.x_1x_2\ldots x_n$

$$[x]_{原}=\begin{cases} x & 1>x\geq 0 \\ 1-x & 0\geq x>-1 \end{cases}$$

符号 $\begin{cases} 0，正 \\ 1，负数 \end{cases}$

  - 有正0和负0之分
  - 范围$-(1-2^{-n}) \sim +(1-2^{-n})$

例： $x=+0.11001110$， $y=-0.11001110$

$[x]_{原}=0.11001110$ $[y]_{原}=1.11001110$

1、原码表示法

- 定点整数$X_0X_1X_2\ldots X_n$

$$[x]_{原}=\begin{cases} x & 2^n > x \geq 0 \\ 2^n - x & 0 \geq x > -2^n \end{cases}$$

符号 $\begin{cases} 0，正数 \\ 1，负数 \end{cases}$

说明:

- 有正0和负0之分
- 范围 $-(2^n-1)$ ~ $+(2^n-1)$

- 例：x=+11001110，y=-11001110

$[x]_{原}=011001110$　　$[y]_{原}=111001110$

1、原码表示法

原码特点：

- 表示简单，易于同真值之间进行转换，实现乘除运算规则简单。
- 进行加减运算十分麻烦。

2、反码表示法

- 定义：正数的表示与原码相同，负数的补码符号位为1，数值位是将原码的数值按位取反，就得到该数的反码表示。
- 电路容易实现，触发器的输出有正负之分。

2、反码表示法

- 对尾数求反，它跟补码的区别在于末位少加一个1，所以可以推出反码的定义
  - 定点小数x0.x1x2…xn

$$
[x]_{反}=\begin{cases} x & 1>x\geq 0 \\ 2+x-2^{-n} & 0\geq x>-1 \end{cases}
$$

3、补码表示法

补码表示法

- 生活例子：现为北京时间下午4点，但钟表显示为7点。有两种办法校对：

(1) 做减法 7-3 = 4 (逆时针退3格)

(2) 做加法 7+9 = 16 (顺时针进9格)

16 (mod 12) = 16-12 = 4 (以12为模，变成4)

3、补码表示法

- 定义：正数的补码就是正数的本身，负数的补码是原负数加上模。
- 计算机运算受字长限制,属于有模运算.
  - 定点小数$x_0.x_1x_2\ldots x_n$，以2为模
  - 定点整数$x_0.x_1x_2\ldots x_n$，以2^{n+1}为模
- 定点小数$x_0.x_1x_2\ldots x_n$

$$[x]_{补}=\begin{cases} x & 1>x\geq 0 \\ 2+x & 0\geq x>-1 \end{cases}$$

符号 $\begin{cases} 0, & 正小数 \\ 1, & 负小数 \end{cases}$

3、补码表示法

- 定点整数$x_0.x_1x_2\ldots x_n$

$$[x]_{补}=\begin{cases} x & 2^n>x\geq 0 \\ 2^{n+1}+x & 0\geq x>-2^n \end{cases}$$

$$符号\begin{cases} 0，正整数 \\ 1，负整数 \end{cases}$$

补码最大的优点就是将减法运算转换成加法运算。通常不按表达式求补码，而通过反码来得到

在定点数的反码表示法中，正数的机器码仍然等于其真值；而负数的机器码符号位为1，尾数则将真值的各个二进制位取反。由于原码变反码很容易实现,所以用反码做为过渡，很容易得到补码。

4、移码表示法

- 移码表示法（用在阶码中）
  - 定点整数定义　$[x]_{移}=2^n+x$　$2^n > x \geq -2^n$
  - 00000000~11111111(-2^n~2^n-1)
  - 例1 x=+1011111

原码为01011111

补码为01011111

反码为01011111

移码为11011111

4、移码表示法

例2 x=-1011111

原码为11011111

补码为10100001

反码为10100000

移码为00100001

特点：移码和补码尾数相同，符号位相反

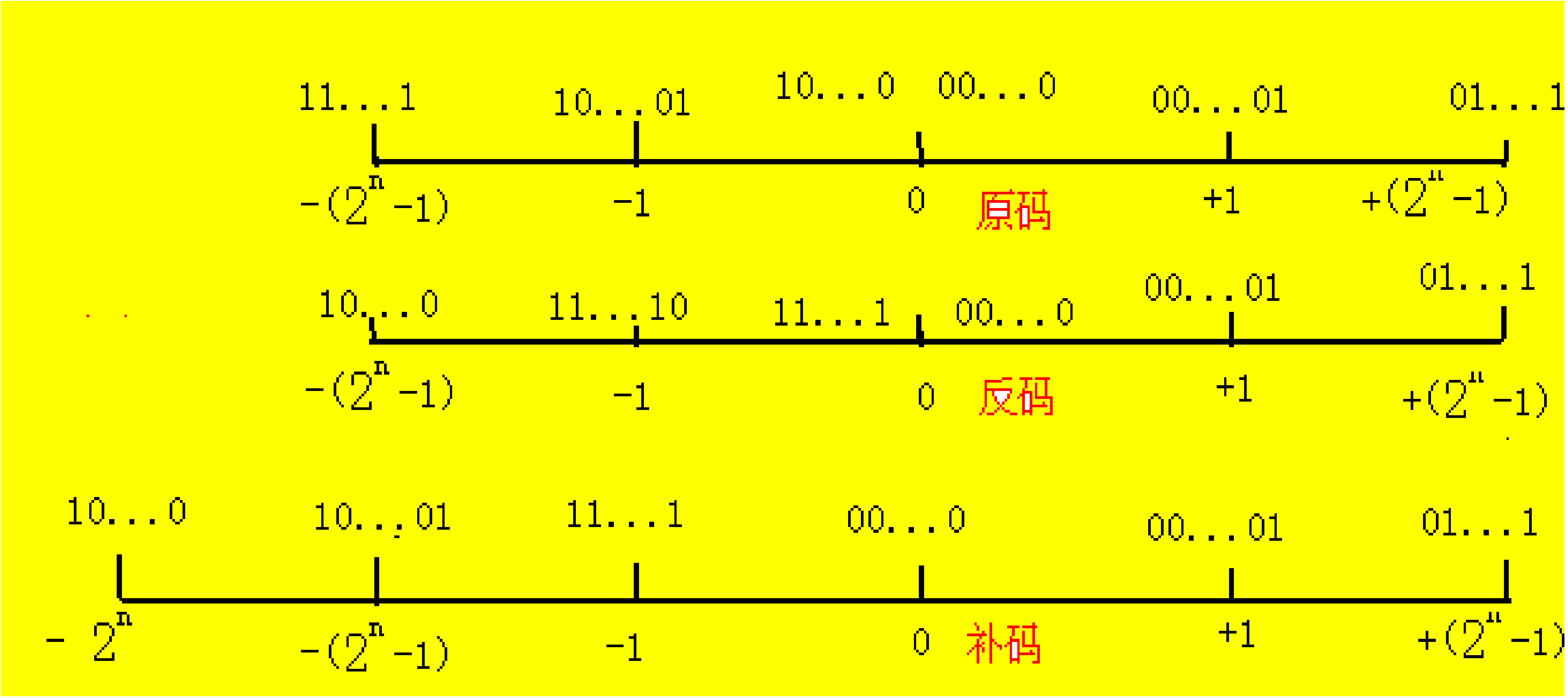
范围:$-2^n \sim 2^n-1$

00000000阶码表示数字”0”,尾数的隐含位为0

11111111阶码表示数字”无穷大” ,尾数的隐含位为0

移码方法对两个指数大小的比较和对阶操作都比较方便，因为阶码域值大者其指数值也大。

[例]以定点整数为例,用数轴形式说明原码、反码、补码表示范围和可能的数码组合情况。



[例]将十进制真值(- 127, - 1,0, + 1, + 127)列表表示成二进制数及原码、反码、补码、移码值。

真值x（十进制）	真值x（二进制）	[x]原	[x]反	[x]补	[x]移
-127	-01111111	11111111	10000000	10000001	00000001
-1	-00000001	10000001	11111110	11111111	01111111
		00000000	00000000		
0	00000000			00000000	10000000
		10000000	11111111		
+1	+00000001	00000001	00000001	00000001	10000001
+127	+01111111	01111111	01111111	01111111	11111111

2.1.3字符和字符串的表示方法

- 符号数据：字符信息用数据表示，如ASCII等；
- 字符表示方法ASCII:用一个字节来表示,低7位用来编码(128),最高位为校验位,参见教材P24表2.1

2.1.4汉字的表示方法

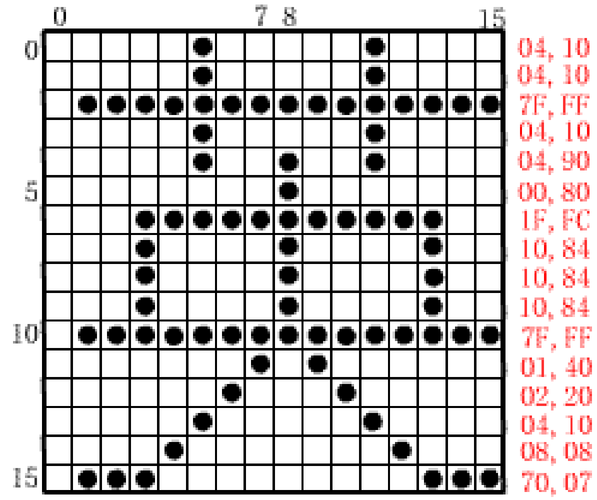
- 汉字的表示方法

(一级汉字3755个，二级汉字3008个)

  - 输入码
    - 国标码
    - 拼音、五笔
  - 汉字内码：汉字信息的存储，交换和检索的机内代码，两个字节组成，每个字节高位都为1（区别于英文字符）

2.1.4汉字的存放

- 汉字字模码：汉字字形
  - 点阵
  - 汉字库



2.1.5校验码

- 校验码（只介绍奇偶校验码）
  - 引入：信息传输和处理过程中受到干扰和故障，容易出错。
  - 解决方法：是在有效信息中加入一些冗余信息（校验位）
  - 奇偶校验位定义
  - 设 $x=(x_0x_1\ldots x_{n-1})$是一个n位字,则奇校验位C定义为:$\overline{C}=x_0\oplus x_1\oplus\ldots\oplus x_{n-1}$,式中$\oplus$代表按位加,表明只有当x中包含有奇数个1时,才使$\overline{C}=1$,即$C=0$。同理可以定义偶校验。
  - 只能检查出奇数位错；不能纠正错误。
  - 其它还有CRC

2.2 定点加法、减法运算

2.2.1 补码加法

2.2.2 补码减法

2.2.3 溢出概念与检测方法

2.2.1补码加法

- 补码加法

  公式：$[x+y]_{补}=[x]_{补}+[y]_{补} \pmod{2^{n+1}}$

$[x]_补 + [y]_补 = [x+y]_补$证明

- 假设 $|x| < 2^n-1$, $|y| < 2^n-1$, $|x+y| < 2^n-1$
- 现分四种情况来证明

(1) $x > 0, y > 0$,则 $x+y > 0$

$[x]_补=x$, $[y]_补=y$, $[x+y]_补=x+y$

所以等式成立.

(2) $x > 0, y < 0$,

$[x]_补=x$，$[y]_补=2^{n+1}+y$,

$[x]_补+[y]_补=x+2^{n+1}+y=2^{n+1}+(x+y)$

a) 当 $x+y>0$时, $[2^{n+1}+(x+y)] \bmod 2^{n+1} = x+y$,

故 $[x]_补+[y]_补= x+y=[x+y]_补$

b) 当 $x+y<0$时, $[2^{n+1}+(x+y)] \bmod 2^{n+1} =2^{n+1}+(x+y)$,

故 $[x]_补+[y]_补= 2^{n+1}+(x+y)=[x+y]_补$

所以上式成立

$[x]_{补}+[y]_{补}=[x+y]_{补}$证明

(3) $x<0$, $y>0$,

这种情况和第2种情况一样,把 x和 y的位置对调即得证。

(4) $x<0$, $y<0$,则 $x+y<0$

相加两数都是负数,则其和也一定是负数。

$\because [x]_{补}=2^{n+1}+x$, $\qquad [y]_{补}=2^{n+1}+y$

$\therefore [x]_{补}+[y]_{补}=2^{n+1}+x+2^{n+1}+y$

$=[2^{n+1}+(2^{n+1}+x+y)] \bmod 2^{n+1}$

$=2^{n+1}+(x+y)$

$=[x+y]_{补}$

2.2.1补码加法

- [例11] x=+1011 , y=+0101 , 求 x+y=?

  解：$[x]_{补} = 01001$, $[y]_{补} = 00101$

$$
\begin{array}{rll}
 & [x]_{补} & 0\ 1\ 0\ 0\ 1 \\
+ & [y]_{补} & 0\ 0\ 1\ 0\ 1 \\
\hline
 & [x+y]_{补} & 0\ 1\ 1\ 1\ 0
\end{array}
$$

$\therefore\ x+y = +1110$

2.2.1补码加法

[例12] x=+1011 , y=-0101 , 求 x+y=?

解: $[x]_{补} = 01001$, $[y]_{补} = 11011$

$$
\begin{array}{lll}
 & [x]_{补} & 0\ 1\ 0\ 0\ 1 \\
+ & [y]_{补} & 1\ 1\ 0\ 1\ 1 \\
\hline
 & [x+y]_{补} & \underline{1}\ 0\ 0\ 1\ 1\ 0
\end{array}
$$

∴ x+y = +0110

2.2.2补码减法

公式:

$[x]_{补} - [y]_{补} = [x]_{补} + [-y]_{补}$

$[-y]_{补} = -[y]_{补} \mod 2^{n+1}$

即

按位取反，末位加一

2.2.2补码减法

[例13] 已知$x_1=-1110$, $x_2=+1101$, 求:

$[x_1]_补$, $[-x_1]_补$, $[x_2]_补$, $[-x_2]_补$ 。

解:

$[x_1]_补 = 10010$

$[-x_1]_补 = -[x_1]_补 + 2^{-4} = 01101 + 00001 = 01110$

$[x_2]_补 = 01101$

$[-x_2]_补 = -[x_2]_补 + 2^{-4} = 10010 + 00001 = 10011$

2.2.2补码减法

[例14] x=+1101 , y=+0110 , 求 x-y=?

解： $[x]_{补} = 01101$

$[y]_{补} = 00110$, $[-y]_{补} = 11010$

$$
\begin{array}{lll}
 & [x]_{补} & 0\ 1\ 1\ 0\ 1 \\
+ & [-y]_{补} & 1\ 1\ 0\ 1\ 0 \\
\hline
 & [x-y]_{补} & \underline{1}\ 0\ 0\ 1\ 1\ 1
\end{array}
$$

∴ x-y = +0111

2.2.3溢出概念与检测方法

- 溢出的概念
  - 可能产生溢出的情况
    - 两正数加，变负数，正溢（大于机器所能表示的最大数）
    - 两负数加，变正数，负溢（小于机器所能表示的最小数）

下面举两个例子。

2.2.3溢出概念与检测方法

[例15] $x=+1101$, $y=+1001$, 求 $x+y$ 。

解：$[x]_{补}=01011$, $[y]_{补}=01001$

	$[x]_{补}$	0 1 0 1 1
+	$[x]_{补}$	0 1 0 0 1
	$[x+y]_{补}$	1 0 1 0 0

两个正数相加的结果成为负数，表示正溢。

2.2.3溢出概念与检测方法

[例16] x=-1101 , y=-1011 , 求 x+y 。

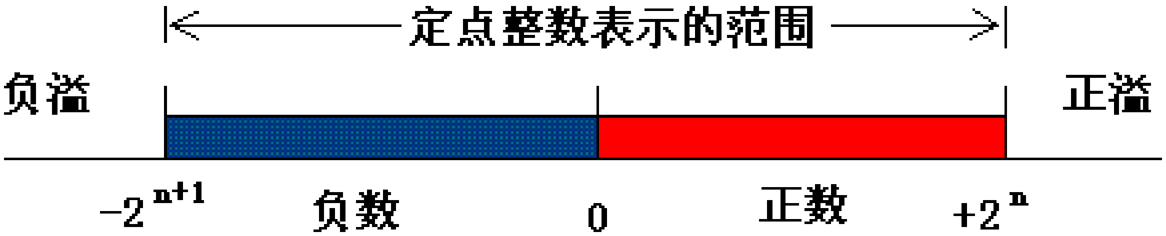
解：$[x]_{补}=10011$, $[y]_{补}=10101$

	$[x]_{补}$	1 0 0 1 1
+	$[x]_{补}$	1 0 1 0 1
	$[x+y]_{补}$	0 1 0 0 0

两个负数相加的结果成为正数，表示负溢。

2.2.3溢出概念与检测方法

溢出的概念



2.2.3溢出概念与检测方法

检测方法

1、双符号位法（变形补码）

$[x]_{补} = 2^{n+2} + x \quad (\bmod 2^{n+2})$

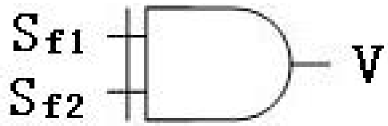
S_{f1} S_{f2}

0 0 正确（正数）

0 1 正溢

1 0 负溢

1 1 正确（负数）



S_{f1} 表示正确的符号，逻辑表达式为$V=S_{f1} \oplus S_{f2}$，可以用异或门来实现

2.2.3溢出概念与检测方法

[例17] x=+01100 , y=+01000 , 求 x+y 。

解： $[x]_{补}$ = 001100 , $[y]_{补}$ = 001000

	$[x]_{补}$	0 0 1 1 0 0
+	$[y]_{补}$	0 0 1 0 0 0
	$[x+y]_{补}$	0 1 0 1 0 0 (表示正溢)

2.2.3溢出概念与检测方法

[例18] x=-1100 , y=-1000 , 求 x+y 。

解： $[x]_{补} = 110100$, $[y]_{补} = 111000$

	$[x]_{补}$	1 1 0 1 0 0
+	$[y]_{补}$	1 1 1 0 0 0
	$[x+y]_{补}$	1 0 1 1 0 0 (表示负溢)

2.2.3溢出概念与检测方法

单符号位法

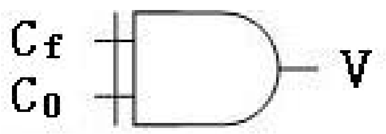
- C_f　C_0

  0　0　　正确（正数）

  0　1　　正溢

  1　0　　负溢

  1　1　　正确（负数）



- $V=C_f \oplus C_0$，其中C_f为符号位产生的进位,C_0为最高有效位产生

2.5.1 逻辑运算

1、逻辑非运算

2、逻辑加运算

3、逻辑乘运算

4、逻辑异运算

1、逻辑非运算

[例24] x_1=01001001 , x_2=11110000 , 求 $\overline{x}_1$, $\overline{x}_2$

解:

$$\overline{x}_1 = 10110100$$

$$\overline{x}_2 = 00001111$$

2、逻辑加运算

[例25] x=10100001 , y=100111011 , 求 x+y

解:

	1 0 1 0 0 0 0 1	x
+	1 0 0 1 1 0 1 1	y
	1 0 1 1 1 0 1 1	z

即 x+y = 10111011

3、逻辑乘运算

[例26] x=10111001 , y=11110011 , 求 x · y

解：

$$
\begin{array}{rl}
 & 1\ 0\ 1\ 1\ 1\ 0\ 0\ 1 \quad x \\
\cdot & 1\ 1\ 1\ 1\ 0\ 0\ 1\ 1 \quad y \\
\hline
 & 1\ 0\ 1\ 1\ 0\ 0\ 0\ 1 \quad z
\end{array}
$$

即 x·y = 10110001

4、逻辑异运算

[例27] x=10101011 , y=11001100 , 求 $x \oplus y$

解:

$$
\begin{array}{cccccccccc}
 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & x \\
\oplus & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & y \\
\hline
 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & z
\end{array}
$$

即 $x \oplus y = 01100111$

课后作业

第二章作业（1）

2.1、2.2、2.4、2.5、2.11、2.13