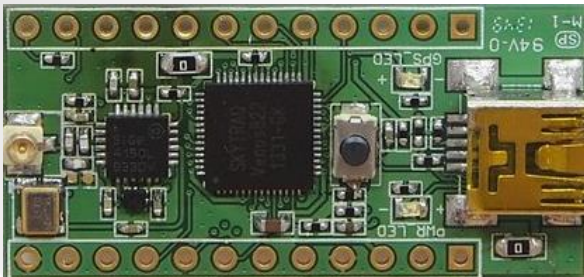


计算机组成原理

第三章 运算方法与运算器

3.1 定点数运算及溢出检测



1

定点数加法运算

$$[X]_{\text{补}} + [Y]_{\text{补}} = [X+Y]_{\text{补}} \mod 2^{n+1}$$

• 算法理解

例1 已知 $X = +10010$ $Y = -10101$ 求 $X+Y$

解: $[X]_{\text{补}} = 010010$ $[Y]_{\text{补}} = 101011$

$$\begin{aligned} [X+Y]_{\text{补}} &= [X]_{\text{补}} + [Y]_{\text{补}} = 010010 + 101011 \\ &= \underline{111101} \end{aligned}$$

所以: $X+Y = -00011$

2

定点数减法运算

$$\underline{[X-Y]_{\text{补}} = [X]_{\text{补}} - [Y]_{\text{补}} = [X]_{\text{补}} + [-Y]_{\text{补}}}$$

• 算法理解

例2 已知 $[Y]_{\text{补}} = 10011$ 求 $[-Y]_{\text{补}}$
0 1 1 0 1

解： $[Y]_{\text{补}} = 10011$

$\therefore Y = -1101$ $-Y = 1101$

$\therefore [-Y]_{\text{补}} = 01101$ ✓

对比 $[Y]_{\text{补}} = 10011$

可知：通过右向左扫描 $[Y]_{\text{补}}$ ，在遇到数字1及之前，直接输出遇到的数字，遇到1之后，取反输出，即可得到 $[-Y]_{\text{补}}$ ，反之亦然！

例3 已知 $X = +10101$ $Y = +10010$ 求 $X - Y$

解: $[X]_{\text{补}} = 010101$, $[Y]_{\text{补}} = 010010$, $[-Y]_{\text{补}} = 101110$

$$[X - Y]_{\text{补}} = [X]_{\text{补}} + [-Y]_{\text{补}} = 010101 + 101110$$

$$= \underset{\text{C}}{1} 000011$$

所以: $X - Y = +000011$

3

数溢出的概念及其判断方法

1) 溢出的概念

运算结果超出了某种数据类型的表示范围。

例4 已知 $X = +10010$ $Y = +10101$ 求 $X+Y$

解: $[X]_{\text{补}} = 010010$ $[Y]_{\text{补}} = 010101$

$$\begin{aligned}[X+Y]_{\text{补}} &= [X]_{\text{补}} + [Y]_{\text{补}} = 010010 + 010101 \\ &= 100111\end{aligned}$$

所以: $X+Y = -11001$

两个正数之和为负数!

例5 已知 $X = -10010$ $Y = -10101$ 求 $X+Y$

解: $[X]_{\text{补}} = 101110$ $[Y]_{\text{补}} = 101011$

$$\begin{aligned}[X+Y]_{\text{补}} &= [X]_{\text{补}} + [Y]_{\text{补}} = 101110 + 101011 \\ &= 1\ 010001\end{aligned}$$

所以: $X+Y = +010001$

两个负数之和为正数!

3

溢出的概念及其判断方法

2) 溢出的检测方法

- 溢出只可能发生在同符号数相加时，包括 $[X]_{\text{补}}$ 与 $[Y]_{\text{补}}$ ； $[X]_{\text{补}}$ 与 $[-Y]$ 同号；

(1) 方法1：对操作数和运算结果的符号位进行检测

当结果的符号位与操作数的符号不相同时就表明发生了溢出

(设 X_0 ， Y_0 为参加运算数的符号位， S_0 为结果的符号位)

$$V = X_0 Y_0 \bar{S}_0 + \bar{X}_0 \bar{Y}_0 S_0$$

当 $V=1$ 时，运算结果溢出，根据该逻辑表达式，容易画出相应电路。

3

溢出的概念及其判断方法

(2)方法2：对最高数据位进位和符号进位进行检测

- 设运算时最高数据位产生的进位为 C_1 ，符号位产生的进位为 C_0 ，

溢出检测电路为： $V = C_0 \oplus C_1$ ✓

$$\begin{array}{r} 0.X_1 \\ + \quad 0.Y_1 \\ \hline \end{array}$$

此时： $C_0 = 0$ ，若 $C_1 = 1$ 则改变了结果符号位，发生溢出。

$$\begin{array}{r} 1.X_1 \\ + \quad 1.Y_1 \\ \hline \end{array}$$

此时： $C_0 = 1$ ，若 $C_1 = 0$ 则改变了结果符号位，发生溢出。

(3)方法3：用变型补码

$$[X]_{\text{补}} = \underline{X_{f1}X_{f2}} \cdot X_1X_2X_3\dots X_n \quad \text{mod } 2^{n+2}$$

$$\text{溢出的判断: } V = X_{f1} \oplus X_{f2}$$

例6 已知 $X = -10010$ $Y = -10101$ 求 $X+Y$

$$\text{解: } [X]_{\text{补}} = \underline{1101110} \quad [Y]_{\text{补}} = \underline{1101011}$$

$$\begin{aligned} [X+Y]_{\text{补}} &= [X]_{\text{补}} + [Y]_{\text{补}} = 1101110 + 1101011 \\ &= \underset{\text{O}}{\textcolor{blue}{1}} \textcolor{red}{\underline{10}} 10001 \end{aligned}$$

$$V = 1 \oplus 0 = 1 \text{ 故发生溢出!}$$

上述三种方法可基于逻辑表达式画出相应电路，
在后面的运算器部分，还将具体讲解

(4) 溢出判断的软件方法

```
int tadd_ok(int x,int y) {  
    int sum=x+y;  
    int neg_over=x<0&&y<0&&sum>=0;  
    int pos_over=x>=0&&y>=0&&sum<0;  
    return !neg_over&&!pos_over; }
```

体会软/硬件功能的等效性和差异性！

体会软/硬协同的系统观！

4

无符号数运算的溢出判断

- 无符号数加法的溢出可用ALU的进位表示
- 无符号数减法的溢出也可用带加/减功能的ALU的进位取反后表示。