原码的加减法运算

一.总体思路

1. 将原码限定为 8bits
2. 从标准输入读入两个十进制数保存在两个 int 当中
3. 通过 itot 函数将两数转变成字符串形式的原码并输出
4. 通过字符串操作模拟原码的加减法运算,并输出结果

二.算法证明

1. 加法

首先判断相加两数的符号是否相同:

若相同,则取两数绝对值通过 7bits 无符号数相加,再将得数符号位赋值为相加两数的符号位

证明:

若 a,b>=0

a+b = |a| + |b|

若 a,b<0

则 a + b = -(|a|+|b|)

若不相同,则判断两数的绝对值大小,将小的一个转变成补码,通过 7bits 无符号数相加,将得数 符号位赋值为绝对值较大的数的符号

证明:

a < 0, b > 0, 且 |a| < |b|

a + b = -(|a| - |b|)

= -(|a| + |b|补)

同理可得其他情况

2. 减法

将减数去相反数,再调用加法运算即可得两数之差

证明:

a - b = a + (-b)

三.使用方法

源文件 true_form.c 在文件夹 src 当中,可直接编译运行。

输入两个在-127~127 之间的运算数,能得到 6 行结果,分别是运算数的原码,和与差的原码以及和与差的十进制数。

若在 linux 系统中,可直接使用 make 命令编译得到可执行程序,再通过命令 cat test.txt | ./true_form.out 来得到实例分析中的测试结果。

四.特殊处理

运算过程中若出现溢出,则 overflow 的值将为 1

若需要将程序进行数位扩展,则之需要将 adder_7bits 函数就位数修改,以及将存储的变量都改成需要的数位即可

五.实例分析

15 20

num1 = 00001111

num2 = 00010100

sum = 00100011 overflow = 0

difference = 10000101 overflow = 0

sum = 35

difference = -5

127 1

num1 = 01111111

num2 = 00000001

sum = 00000000 overflow = 1

difference = 01111110 overflow = 0

sum = 0

difference = 126

127 -127

num1 = 01111111

num2 = 11111111

sum = 10000000 overflow = 0

difference = 01111110 overflow = 1

sum = 0

difference = 126

-120 -10

num1 = 11111000

num2 = 10001010

sum = 10000010 overflow = 1

difference = 11101110 overflow = 0

sum = -2

difference = -110