FP16规格化数加法

该工程实现FP16规格化数的加法，输出FP16规格化数

1. Step1：pre\_operation

该阶段需要判断操作数有效性（Inf、0），并判断操作数绝对值大小关系，并分离符号、阶数、尾数各部分

* 0xffff表示Inf，任何运算皆是Inf，以标志位overflow记录，后续运算如果overflow为真就不必再计算了
* 如果有一个操作数是0，则另一个数记为更大数，并将更小数阶码、尾数都记为0
* 正常情况（操作数中无Inf、0）首先比较阶数大小，大阶数记为更大数；阶数相同比较尾数大小，尾数大记为更大数。这样可以保证high大于等于low
* 尾数左移11位是为了方便后续对阶。

？1xx\_xxxx\_xxxx|

|1xx\_xxxx\_xxxx

这是对阶的极限情况，阶数相差再大，较小数对较大数的影响就无法在精度下体现出来了，而从上面可以看出，对阶用11+11+1=23位即可

1. Step2：shift to align

该阶段需要根据阶数差距进行两个尾数的对阶

* 如step1中所说，当阶数相差大于12就无法在精度下体现出来了
* 采用小向大对齐的操作，即较大数不变，较小数移位以对齐

1. Step3：Calculate

该阶段根据符号位进行加或减的运算

* 阶数、符号位皆以较大数为准
* 符号位相同即加，不同即减

1. Step4：normalize\_pre

从该阶段开始需要进行结果的规格化，该阶段需要进行规格化初步的必要准备，寻找从高到低的第一个1从而判断其后十位保存的尾数，根据从高到低的第一个1的位置来修正阶数，以及根据抛弃部分判断舍入与否

* 从step1得知，1xx\_xxxx\_xxxx\_xxx\_xxxx\_xxxx为当前阶数的基准，

如果是 1xxx\_xxxx\_xxxx\_xxx\_xxxx\_xxxx，则阶数+1

反之 1x\_xxxx\_xxxx\_xxx\_xxxx\_xxxx，则阶数-1，以此类推

* 尾数保留从高到低的第一个1及紧接着的十位，一共11位
* 舍入原则为舍入到最近可以表示的数，如果距离一样则偏好舍入到偶数，判断条件为 [舍弃部分最高位 && （保留部分最低位 || 舍弃部分次高位及以后是否还有1）]
* ***按道理来说23位寄存器就会23种第一个1的可能，但我直观感觉最低到第11位（无数学证明），其他不会出现，待讨论？***

1. Step5：normalize\_round to nearest even

该阶段根据step4中得出的舍入与否进行尾数的修正

* 如果ifround为真，尾数+1

1. Step6：normalize\_carry

由于step5中的入可能会导致溢出，该阶段进行阶数和尾数修正

* 如果第十二位为真，尾数右移一位，阶数+1

1. Step7：result

该阶段可以得出结果

* 如果overflow标志为真，或者阶数大于30，令结果为0xffff
* 如果尾数计算得0，令0x0000
* 如果以上皆不是，则正常得拼接符号位，阶数，尾数低十位即可