FP16\_div算法说明文档

该模块实现FP16规格化数的除法，输出规格化数

这三部分分别是符号、尾数商、指数商，其中非规格化可能会在尾数商的形式上变化，另外指数商的指数需要再加FP16偏移值15，即E\_X-E\_D+15

1. Step1：input check

该步骤中需要检查除数是否为0，如果是则直接跳转输出

正常情况下（不是）分离被除数和除数的符号、指数、尾数。根据是否规格化和被除数是否0来做修正。

如果指数是0，说明该数可能是0或者非规格化数，如果尾数是0，则说明是0，不做修正；如果尾数不是0，说明是非规格化数，指数修正为1；如果指数不是0，则说明是规格化数，将尾数第11位修正为1

1. Step2：calculate sign，exp

该步骤计算商的符号和指数，具体为符号异或，指数相减再+15。另外需要把被除数X左移22位以使商也左移22位从而看到商的22位小数部分。至此因为被除数变味了22+11=33位，被除数需要左移33位。同时迭代次数刷新为32

1. Step3：calculate rm

该状态是使用不恢复余数法计算被除数X和除数D的尾数的商，选择不恢复余数法的理由是基2的SRT不会比不恢复余数法有优势，而基4的SRT不会也看不懂o.0？

不恢复余数法不多做解释，参考自：[【HDL系列】除法器(2)——不恢复余数法 - 知乎 (zhihu.com)](https://zhuanlan.zhihu.com/p/206770701)

根据我的尝试修正过算法

1. Step4：normalize\_pre

该步骤选择上一步骤的结果中相应的位数作为尾数，并选择舍入标志和修正指数

由于step2中将被除数左移了22位，所以结果中的低22位是尾数商的小数部分，低23-33这11bits是尾数商的整数部分，通过选择相应位置和修正指数将其化为1.f(f指接下来的10位作为小数)\*2^exp+comp(comp指指数修正值)的形式

1. State5：denormal operation

Step4中修正过的指数可能会小于1，这说明该值应该是一个非规格化数，需要按照非规格化的形式再次调整

通过右移尾数的形式来将指数修正到0，数学上每右移1位指数+1。其中因为最后一位是进位符号左移应该是先右移1位，再每右移1位指数+1

1. State6：output

指数大于30，上溢出，输出{符号，0x7bff}

尾数是0，输出0x0000

其余情况输出{符号，指数低5位，尾数低10位}