在浮点数的运算过程中，运算结果最多只能舍入保留 24（float）或 53(double)位尾数，为了更准确地进行舍入，硬件在浮点运算时应该额外多保留几位。

1. 保护位、舍入位

IEEE754 标准规定：所有浮点运算的中间结果的右边必须额外多保留两位，这两位分别叫保护位（G：guard）、舍入位（R：round）.

例如：将 2.56× 10^0 + 2.34× 10^2， 对阶后变成： 0.0256× 10^2 + 2.34× 10^2

```
    2.3400
+   0.0256
-------------
=   2.3656   （保护位为 5， 舍入位为 6）
```

由于多余的尾数位 56 大于 50，所以向上舍入，最终加法结果为 2.37× 10^2

但如果没有保护位和舍入位，最终加法结果位 2.36× 10^2， 相比之下，这个 2.37× 10^2离精确的结果更接近。

2. 为什么要设置两位保留位（G 和 R 位）？

在上例，只留下一位保护位（G：guard）也能达到同样的舍入效果。对浮点加法运算，一位保留位确实就够了。但乘法需要保留两位，当两个二进制尾数相乘后，得到的结果如果是小数点前面为 0，还需要规格化，将乘积左移一位。移位会将保护位移入变成最低有效位，留下舍入位精确舍入乘积。

3. 粘贴位

对.5 的舍入上, IEEE754 与四舍五入有一点不同, 它采用取偶数的方式（最近舍入模式: Round to Nearest）

举例比较:

最近舍入模式： Round(0.5) = 0; Round(1.5) = 2; Round(2.5) = 2;

四舍五入模式： Round(0.5) = 1; Round(1.5) = 2; Round(2.5) = 3;

为了支持这种舍入方式，使得计算机能舍入到最近的数字，除了保护位和舍入位之外，还要增加一个粘滞位（S：sticky）,只要舍入位的右边有非零位，就将 sticky 位设置为 1.

举例说明 G、R、S 这三个额外保留位的应用

给定 16 位的 IEEE754 编码的浮点数，1 位符号位，5 位指数，10 位尾数。

A=2.6125×10^1, B=4.150390625×10^{-1}, 计算 A+B

假定有 1 个 guard(保护位) , 1 个 round（舍入位）, 1 个 sticky（粘贴位），最近舍入模式 。

计算步骤如下:

2.6125×10^1 = 26.125 = 11010.001 = 1.1010001000×2^4

4.150390625×10^{-1} = .4150390625 = .011010100111 =1.1010100111×2^{-2}

对阶， 小阶往大阶对

```
    1.1010001000 00
+    .0000011010 10 0111 (Guard = 1, Round = 0, Sticky = 1)
-------------------------------------------------------------
=   1.1010100010 10 1   (Guard = 1, Round = 0, Sticky = 1)
```

加法结果：1.1010100011 × 2^4（检查，无溢出）

= 11010.100011 × 2^0 = 26.546875 = 2.6546875 × 10^1