Here is an example of why you need a Guard bit, in addition to the Round and Sticky bits.

Consider the subtraction problem below, first with infinite precision, then with Guard, Round and Sticky bits (like in IEEE FP), and finally just using Round and Sticky bits (not correct).

```
   1.000 x 2^5
-  1.001 x 2^1
----------------------------------------------------------------------
```

With infinite precision, we keep ALL the bits as we shift the fraction right to make the exponents equal. Note that we ALWAYS shift right, to make the logic easier to implement. After we perform the subtraction, we need to shift the result left one bit to renormalize the result (because the result has a leading 0). Then we need to round the result to have (in this example) three bits to the right of the radix point. We call the least significant bit of the rounded result the “unit of the last place” or ULP. If the bits to the right of the ULP have weight >1/2 ULP, then we need to round up. If they bits have weight <1/2 ULP, then we round down. And if the bits are exactly equal to 1/2 ULP, then we round to even. In this example, there are three bits to the right of the ULP. If these three bits are 0xx, then we round down; if the bits are 100 we round to even; if the bits are 1xx and at least one of the x’s is a one, we round up.

```
   1.000 0000  x 2^5
-  0.000 1001  x 2^5
----------------------------------------------------------------------
   0.111 0111  x 2^5   Need to shift left to normalize
   1.110 111   x 2^4   Round up, since more than half unit of the last place
   1.111       x 2^4
```

We can use the Guard, Round, and Sticky bits to reduce round the result correctly using on 3 extra bits in our adder/subtractor. As we shift the operand right, we shift the bits into the Guard, then the Round, and finally the Sticky bits. The Guard and Round bits are just standard bits, but the Sticky bit is 1 if ANY bit that shifts through it is a 1. In this example, the Sticky bit is set to 1 since the first bit that shifts into it is a 1. Note that the initial result of the computation is slightly different than the infinite precision version, but that the rounded result is the same.

```
   1.000 000  x 2^5
-  0.000 101  x 2^5   Guard is 1, Round 0, and Sticky is 1
----------------------------------------------------------------------
   0.111 011  x 2^5   Need to shift left to normalize, using Guard bit
   1.110 11   x 2^4   Round up, since more than half unit of the last place
   1.111      x 2^4   Result is correctly rounded
```

It isn't always obvious why you need the Guard bit, and can't get by with just the Round and Sticky bits. But this example shows you why you need the Guard bit. If we do the calculation with just the Round and Sticky bits, we use up the Round bit when we have to normalize the result (since the bit to the left of the radix point is 0). Thus we are left with only the Sticky bit to do the rounding. But we can't really use the Sticky bit to round, because we can't tell whether it represents 001, 100, or 111, which would all round differently.

```
   1.000 00  x 2^5
-  0.000 11  x 2^5   Round is 1, Sticky is 1
---------------------------------------------------------------------------
   0.111 01  x 2^5   Need to shift left to normalize, must use Round bit
   1.110 1    x 2^4   Can't round using Sticky, since can't tell if >/=/< 1/2 ULP
```

If it still isn't clear WHY you can't use the Sticky bit to round, consider the following SLIGHTLY DIFFERENT computation:

```
   1.000 x 2^5
-  1.111 x 2^1
---------------------------------------------------------------------------
```

With infinite precision, the calculation is:

```
   1.000 0000  x 2^5
-  0.000 1111  x 2^5
---------------------------------------------------------------------------
   0.111 0001  x 2^5   Need to shift left to normalize
   1.110 001    x 2^4   Round down, since < 1/2 ULP
   1.110          x 2^4
```

With Guard, Round and Sticky, the calculation round correctly:

```
   1.000 000  x 2^5
-  0.000 111  x 2^5
---------------------------------------------------------------------------
   0.111 001  x 2^5   Need to shift left to normalize
   1.110 01    x 2^4   Round down, since less than half unit of the last place
   1.110        x 2^4
```

With only Round and Sticky, the calculation becomes:

```
   1.000 00  x 2^5
-  0.000 11  x 2^5
---------------------------------------------------------------------------
   0.111 01  x 2^5   Need to shift left to normalize
   1.110 1    x 2^4  Can't round using Sticky, since can't tell if >/=/< 1/2 ULP
```

But notice that this last calculation is EXACTLY the same as the one we did before using just Round and Sticky, but in that case the right answer is to round UP, and in this case the right answer is to round DOWN. Thus, we need the Guard bit to round correctly.