Fast Integer Square Root

Author: Daniel Dimov <danieldimov@gmail.com>

In an integer world very few numbers has exact square roots. Therefore finding the square root of an integer number (result also integer) will be in most of the cases some approximation. If **x** is integer and **a** and **b** are two consecutive integers (**a** + 1 = **b**) we will consider that we found the square root of **x** if we find either **a** or **b** satisfying $\mathbf{a}^2 < \mathbf{x} < \mathbf{b}^2$.

If **x** has exact square root – we will of coarse must find **a** satisfying $\mathbf{a}^2 = \mathbf{x}$.

The attitude to the problem is the fact that if **y** is a guess for the square root of **x**, (**x**/**y** + **y**)/2 is a better guess. By calculating better guess several times in a row - we can reach finally to the same as previous guess – this will be the answer of the problem.

The code is:

```
int32_t fastSqrt(int32_t x, int32_t guess, byte precision)
{
    int32_t oldGuess, newGuess, sigma;

    if (x <= 0) return 0;                          // process only positive numbers
    newGuess = guess;                              // initial guess that leads to minimum # of iterations
    do
    {
        oldGuess = newGuess;                       // save the guess from previous iteration
        newGuess = (x / newGuess + newGuess) >> 1; // calculate the new guess
        sigma = (newGuess - oldGuess) >> precision; // calculate the difference and eliminate the bits we don't care of
    } while (sigma != 0 && sigma != -1);           // if the difference is small enough - we are done
    return newGuess;                               // return the latest guess
}
```

The speed of this code is determined of the number of iterations it will make. The number of iterations depends on how good is the initial `guess`. With a good guess – the code can find the square root in less than 4 iterations (average), but with a completely wrong guess it can make up to 10 iterations. The key factor for the speed of this code is to know your data and to know what guess will lead to minimum average number of iterations. One option is to pass the initial guess as a parameter (like it is above), but other option is to put it as a constant in the body of the function. General orientation for the initial guess is that it must be filled with bits up to the half of the bits of the argument. On x86 platform this code can reach a speed very close to the FPU calculation (conversion from int to float, FPU square root and conversion back from float to int). On ARM platform – it needs to be tested extensively.

The precision of this code is determined of the argument `precision`. This argument is in number of bits and must be 1 or more. It can be increased (to be 2 or more) if the specific application doesn't require precision up to the latest bit. Decreasing the precision will have little impact on the performance, but in general the lower the precision – the faster is the calculation. If the necessary precision is known in advance – it will be better to be put in the code as constant.