GEK1536, Computation and Machine, Tutorial 10 (last)

(For week 13 starting 3 Apr 06)

1. Suppose that we want to use an 8-bit char (a byte) to represent a floating point number. The 7^{th} bit will be used for sign (0 for positive, and 1 for negative). We use the next 3 bits for the biased exponent E, and the last 4 bits for the fractional part xxxx of the binary number $1.xxxx \times 2^e$. (a) What bias should be a reasonable choice for the 3-bit exponent field, $E=e+\text{bias}$? (b) How many different floating point values can you represent using this system? (c) Enumerate some of the values (in decimal), particularly those close to 0, 1, and the largest. [No need to represent the special values such as ∞ and NaN as in IEEE.]

2. Write a C program to compute the value π, using the continued fraction formula:

$$4/\pi = 1 + \cfrac{1^2}{2 + \cfrac{3^2}{2 + \cfrac{5^2}{2 + \cfrac{7^2}{2 + \cfrac{9^2}{2 + \cfrac{11^2}{2 + \cdots}}}}}}$$

   The number of terms used is given as a user input. It's good if you can try your code on your computer, but it is not a requirement.

Home Work (*Since there will be no more tutorial, handing in your work on or before Friday 14 Apr 06 to Prof. Wang S12-02-17 or Oliver in their offices*)

3. (**Homework**) Implement a simple C program to compute the square root of a number. The program reads a (double precision) number x, and compute $\sqrt{x}$ using the averaging method of Babylonians (c.f. lecture 2). The number of iterations in the averaging process should not be fixed but determined by a relative error tolerance ε (say with 10 decimal accuracy). It prints out the result to the user.

   Compiling and running of your program is not a requirement. But it does help you checking if you have made any programming errors (bugs).