Number of digits. As mentioned before, an integer n satifying $b^{k-1} \leq n < b^k$ has k digits to the base b. By the definition of logarithms, this gives the following formula for the number of base-b digits (here "[ ]" denotes the greatest integer function):

$$\text{number of digits } = \left[log_b n\right] + 1 = \left[\frac{log\, n}{log\, b}\right] + 1,$$

where here (and from now on) "log" means the natural logarithm log_e.

Bit operations. Let us start with a very simple arithmetic problem, the addition of two binary integers, for example:

$$\begin{array}{r} \scriptstyle 1\,1\,1\,1 \\ 1111000 \\ +\,\underline{0011110} \\ 10010110 \end{array}$$

Suppose that the numbers are both k bits long (the word "bit" is short for "binary digit"); if one of the two integers has fewer bits than the other, we fill in zeros to the left, as in this example, to make them have the same length. Although this example involves small integers (adding 120 to 30), we should think of k as perhaps being very large, like 500 or 1000.

Let us analyze in complete detail what this addition entails. Basically, we must repeat the following steps k times:

1. Look at the top and bottom bit, and also at whether there's a carry above the top bit.
2. If both bits are 0 and there is no carry, then put down 0 and move on.
3. If either (a) both bits are 0 and there is a carry, or (b) one of the bits is 0, the other is 1, and there is no carry, then put down 1 and move on.
4. If either (a) one of the bits is 0, the other is 1, and there is a carry, or else (b) both bits are 1 and there is no carry, then put down 0, put a carry in the next column, and move on.
5. If both bits are 1 and there is a carry, then put down 1, put a carry in the next column, and move on.

Doing this procedure once is called a *bit operation*. Adding two k-bit numbers requires k bit operations. We shall see that more complicated tasks can also be broken down into bit operations. The amount of time a computer takes to perform a task is essentially proportional to the number of bit operations. Of course, the constant of proportionality — the number of nanoseconds per bit operation — depends on the particular computer system. (This is an over-simplification, since the time can be affected by "administrative matters," such as accessing memory.) When we speak of estimating the "time" it takes to accomplish something, we mean finding an estimate for the number of bit operations required. In these estimates we shall neglect the time required for "bookkeeping" or logical steps other