Lab 02

September 13, 2017

Finding your endianness

Your “endianness” plays an important role in your computer: it dictates whether the most significant byte (big-end) or least significant byte (little-end) is stored at the address of your data. Some architectures such as Motorola 68K are big-endian while x86 (and descendants) fall into the little-endian category. Download the endian.c file to your computer. Compile it with `clang -Wall endian.c`. Now pick a three letter word and look up the hexadecimal values in the ASCII table (along with the newline character). Create the proper 4-byte int to create that string. What is the endianness of your computer???

Greatest common divisor

The greatest common divisor (GCD) of two numbers can be found by using Euclid’s algorithm. Implement the recursive Euclidean algorithm to solve the GCD of various values.

Memoization

Finding a particular Fibonacci number is easy: given F(1) = F(2) = 1, one can build successively bigger values with the recurrence F(n) = F(n-1) + F(n-2). The problem is that lots of computation is repeated using this method. One approach is to *store* the results of previous computation. We can store previously-computed Fibonacci numbers in a global array of integers that we have allocated. Filling in an array with sentinel values of zero will let you check if that value has been computed already. If, for example, F(x) is non-zero then that value has already been computed and you can return that partial solution. Otherwise, you must compute F(x) as usual, possibly utilizing some pre-computed values. Prepend the `time` command to your normal command line e.g., `time ./a.out 42` for both a naive and optimized versions. Is it a noticeable difference? What was the trade-off?