1 Optimization

Download the following C program:

http://ranger.uta.edu/~nystrom/courses/5317/hw3/nbody.c

On omega.uta.edu, compile the program with gcc:

$ gcc -lm nbody.c -o nbody.noopt

Time and run the program:

$ time ./nbody.noopt 50000000
-0.169075164
-0.169083134

real 0m49.702s
user 0m49.670s
sys 0m0.002s

Sum the user and system times and report that result.

Now, repeat the compilation and run for each of the following command-line options:

$ gcc -lm nbody.c -o nbody.noopt
$ gcc -O -lm -o nbody.opt
$ gcc -O0 -lm -o nbody.opt0
$ gcc -O1 -lm -o nbody.opt1
$ gcc -O2 -lm -o nbody.opt2
$ gcc -O3 -lm -o nbody.opt3
$ gcc -O3 -fomit-frame-pointer -lm -o nbody.opt3ofp
$ gcc -Os -lm -o nbody.optsize
$ gcc -O3 -fwhole-program -lm -o nbody.optsize
$ gcc -O3 -fno-inline -lm -o nbody.optsize
$ gcc -O3 -floop-optimize2 -o nbody.optsize

Run each benchmark twice in a row and report the second time.

At what point does applying more optimizations not improve performance significantly?

Read the man page for gcc (man gcc). What optimizations are applied with each optimization level above?

Experiment with other command-line options. Do any help? Do any hurt performance significantly?
2 Optimization

Consider the following program:

```c
is = 5;
for (i = 0; i < 100; i++) {
is += 10;
    for (j = 0; j < 200; j += 3) {
        a[is] = b[i] + c[j];
    }
}
```

Apply each of the following optimizations to successive versions of the above program. Show the results of each step and use that as the input to the next step.

(a) Unroll the inner loop a factor of 4.

(b) Common subexpression elimination.

(c) Strength reduction.

(d) Induction variable elimination.

(e) Common subexpression elimination (again).