CoE 163

Computing Architectures and Algorithms

Software Exercise 01: Profiling and Assembly

Synopsis: Formulate and profile two different algorithms to compute the number of mice pairs in a breeding simulator

Problem Statement

You are simulating an idealized version of mice breeding for a biocomputing firm. Breeding laboratory mice is such an important task since these creatures are very useful for scientific research.

There are important assumptions in the simulation, namely:

- The mice never die during the duration of the simulation. They breed forever.
- All of the mice bred or born are always fertile.
- Each pair of mice mate every 3 weeks except on the day they were born.

For this simulation, you will start with no mice. After three weeks, you will receive a pair of newly-born fertile mice. After three more weeks, these mice will mate and will produce two new mice (or a pair). Female mice generally become pregnant a day after mating and gestate for three (3) weeks. Then, after three more weeks, these two new mice will be born while the other two mice (the ones who birthed the previous two mice) will mate again to produce a new pair.

You are interested in computing how many pairs of mice end up after some amount of 3-week intervals. For reference, a year has approximately eighteen (18) 3-week intervals.

Since you expect to compute for a large number of 3-week intervals, you have decided to formulate two programs and choose the fastest one, with one made using your preferred programming language. Because of your overzealousness and also curiosity on whether you can squeeze more time from your simulation, you have decided to write the other program using a mixture of C/C++ and assembly.

Input
Input to your program would be the number of 3-week intervals N the simulation will run.

Output
Your program should output the number of pairs of mice after N 3-week intervals have passed.

Constraints
$0 < N \leq 40$

Sample Inputs/Outputs
Sample Input 1
1

Sample Output 1
1

Sample Input 2
2

Sample Output 2
1

Sample Input 3
6

Sample Output 3
8

Additional Description/Requirements
In addition to the algorithm, we expect you to have two programs for the simulation. Time each of your algorithms by first formulating a function for the breeding simulator. Then, enclose that function call in a loop.

You can use any programming language that runs at least 18 times, corresponding to one year in simulation time. The input to the function can be the index of the loop (e.g. if the loop is now on its 50th iteration, then the number of layers entered into the algorithm is 50). For example, in an ancient language named Turbo Pascal, this is how you may write your timed algorithm.

```
program BreedingSim;

var i: integer;

begin
  { TIMER start }
  for i:= 1 to 18 do begin
     { BREEDING SIMULATOR here }
  end;
  { TIMER end }
end.
```

You may use any programming language for your first program, but the second program should be written in a mixture of C/C++ and x86_64 assembly. You may either code the assembly part separately using your preferred assembler (NASM is used in the references) or use inline assembly in C/C++. Profile each of your two programs using your preferred profiler.

You also would like to write a short journal entry on which program you choose as your simulator. Explain why that algorithm was correct, how long did the two programs take to run at least 18 times, and why one of them ran faster. Also explain the differences between the two programs and which of those are better in terms of runtime, programming effort, and code organization.

Upload both your programs (as two or three single source code files; in TXT if the system does not support the file extension) and your short journal (PDF or TXT file) to your remote work repository named UVLe.

Grading Rubric

30% Algorithm code and correctness (preferred language)
15% Algorithm code and correctness (C/C++ and x86_64 asm)
20% Profiling code (preferred language)
10% Profiling code (C/C++ and x86_64 asm)
25% Short journal