Lab Sheet: Extra Exercises

This lab sheet covers some extra exercises in systems programming, to help preparing for exam and for the CW2 implementation. The tasks should be performed on one of the Linux lab (EM 2.50) machines in the department, mainly for convenience. You can try the same exercise directly on a RPi2, but expect to get quite difference performance numbers on this device, which uses an ARM rather than an Intel chip.

Exercise: Factorial computation in C and in Assembler

The factorial of an integer value n, written $n!$, is the product of all integer values from 1 to n, i.e.

$$n! = n \times (n-1) \times \cdots \times 1$$

Task: Implement 2 versions of a factorial function in C, one version that uses iteration and one version that uses recursion. Use `valgrind` (see above) to measure the performance of these versions. Which one is faster and why?

Then, again implement 2 versions of a factorial function in ARM Assembler (one as iteration the other as recursion using the stack to pass arguments). Again, compare the performance of both assembler versions, and also compare the performance with the 2 C versions that you implemented.

For sample solutions, look up the corresponding page on Rosetta Code.

Exercise: Binary digit computations

Lookup this page at Rosetta Code discussing bitwise operations, and implement this task first in C then in ARM Assembler. Compare your solution with the sample solutions on Rosetta code.

On-line material on ARM Assembler programming

A good sequence of blogs on ARM assembler in Raspberry Pi is available here. Use this as supplementary material to deepen your understanding about ARM Assembler programming, with concrete examples that you can cut-and-paste from the web page and try yourself on your Raspberry Pi.

How to use Valgrind

On the RPi2, you will need to install valgrind separately, by typing: `sudo apt-get install valgrind`

On the Linux lab machines, `valgrind` is already installed and you can run it there like this:

```
> valgrind --tool=cachegrind ./matrix2 2MAT_5_80_99
==5316== Cachegrind, a cache and branch-prediction profiler
...
```

Look out for a line like this, which summarises the *(data) cache miss rate for the level 1 cache*:

```
...
==5316== D1  miss rate:       0.3% (    0.4%     +     0.1%  )
...
```

The lower this number, the better the performance of the program will be.
If you don't have `valgrind` installed, do the following with the RPi2 connected to your local network:

```
> sudo apt-get install valgrind
```