Microprocessor Architectures [ELEC-H-473]
Lab 9: Multi-threaded execution on a multi-core architecture

BEYENS Ziad
GARCIA DIAZ Antonio
NOUGBA Hamza

May 11, 2017

1 Introduction

The aim of this laboratory was to study the pros and cons of using multi-threading in image processing. More specifically, the goal is to modify some of the previously implemented methods in C++ and assembly SIMD (from labs 7 and 8) so that they exploit a multi-thread architecture, and compare their runtime with that of the original methods (that used a single thread).

2 Presentation of the program

The image processing methods that were treated in this lab were the threshold filter (using a value of 100), and the maximum filter (applied on 3x3 pixels neighborhoods), both in their C++ and assembly SIMD versions. These filters are, as usual, applied on an 8-bit gray-scale image file called test.raw, a 1024x1024 picture of a baby's face. The program can be compiled with a simple "make" instruction, and needs no parameters at the execution.

The source code for the aforementioned methods is reused to create multi-threaded versions of themselves, exploiting 20 threads by default. Both the threshold and the maximum filter implement different data structure to define their threads (called `Thresh` and `Max`, respectively). These contain, as parameters, the first and last positions to treat in the image data, and all the necessary variables for applying the filter. The program outputs 8 image files, each processed by a different filter (threshold or maximum, standard C++ or SIMD-exploiting assembly) and either using multi-threading or not. The execution times for processing each file are displayed on the console.

3 Benchmarking of the methods

For the threshold filter:

- The C++ implementation using a single thread was executed in 0.0060 seconds.
- The SIMD implementation using a single thread was executed in 0.0020 seconds.
- The C++ implementation using 20 threads was executed in 0.0050 seconds.
- The SIMD implementation using 20 threads was executed in 0.0030 seconds.

For the maximum filter:

- The C++ implementation using a single thread was executed in 0.0970 seconds.
- The SIMD implementation using a single thread was executed in 0.0030 seconds.
- The C++ implementation using 20 threads was executed in 0.8820 seconds.
- The SIMD implementation using 20 threads was executed in 0.0080 seconds.

It appears that the multi-threaded versions of the methods tend to take more execution time than the single-threaded versions. Only the C++ threshold filter is faster when exploiting various threads than when using a single thread. This may be due either to the relative complexity of setting the threads themselves with respect to the actual execution time of the algorithm, or (most probably) to the fact that the tests were run on a computer with two cores. Perhaps, indeed, more cores are needed to observe a significant improvement in the execution times.