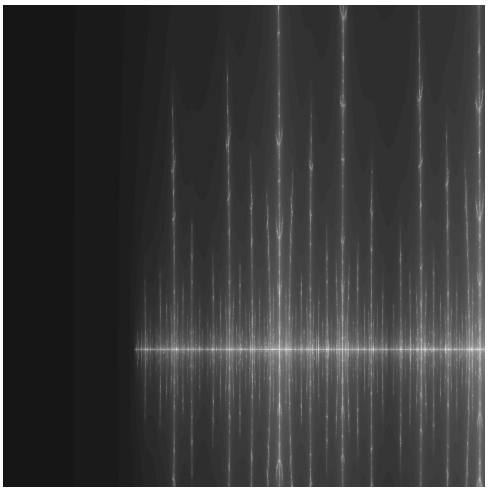
CSCM98, Assignment #1: Accelerating fractals (SIMD+MT)



Important reading

By submitting this coursework, electronically and/or hardcopy, you state that you fully understand and are complying with the university's policy on Academic Integrity and Academic Misconduct. The policy can be found at https://www.swansea.ac.uk/academic-services/academic-guide/assessment-issues/academic-integrity-academic-misconduct.

Note that course material can be used, but this is a personal piece of work and any extra material must be referenced. The course page includes a zip file to run the original version of the code. Submission instructions will be sent to you by email. The process will mainly be to reply to the email and attach your BasicApplication.cpp file. Your code should be compilable with the given project.

Some extra course material can be found at: http://cs.swansea.ac.uk/~csmora/CSCM98/

A list of Single Instruction Multiple Data (SIMD) Advanced Vector Extensions (AVX) instructions can also be found at:

https://software.intel.com/sites/landingpage/IntrinsicsGuide/

Description

We want to accelerate an algorithm that creates fractal images by combining SIMD programming and multithreading.

The algorithm creates a fractal image (see image above) by iterating for each pixel a fractal value. For each pixel, the *Iterate*(*pixel*) function is called and returns more or less the number of iterations performed. While the code uses complex numbers, the original C code to compute the iteration is already given and you just have to port the sequence of instructions to an SIMD device which shall process 8 pixels at once, as well as multithreading the code. As usual, most of the code is already written, and you will have to complete and return the .cpp file.

Marks breakdown

- 30%: Code is properly multithreaded and efficient.
- 25%: A basic use of SIMD instructions (AVX) is implemented and the code is significantly faster (SIMD accelerated).
- 20%: The SIMD code is optimised heavily, making use of more advanced instructions than just *add* or *mul*. However, **it is forbidden** to use the `_mm256_testz_ps` function, which will have to be replaced by another one.
- 15%: Workload is balanced across all threads, and therefore the algorithm scales well. Please note here that there exists a very simple solution to this problem.
- 5%: Computation times are written as comments in the code, including the computer used to do so.
- 5%: Submission follows guidelines and does not involve extra work (e.g. correcting code).

Notes:

- Any partial solution may return only partial marks. The exact marks will depend on the quality of the answers and can only be evaluated upon reception of the coursework.
- This is a strictly personal work. Any external help must be referenced as comments in the code.
- Add your student ID and Name to the code.
- You will receive an email detailing how you should submit your work soon.