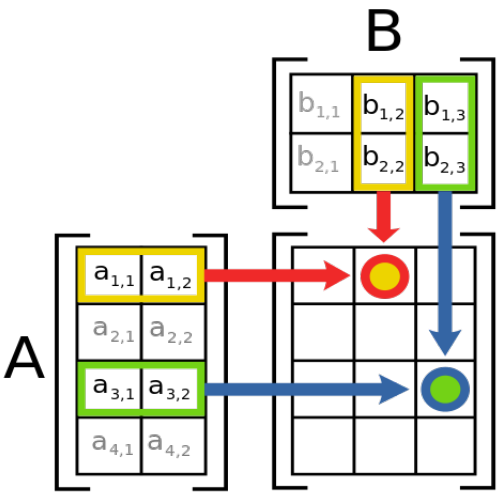
Second Assignment
Games Console Development
COMP10037

Issue Date: Monday, March 4th, 2019
Due Date: **5pm, Friday, March 29th, 2019**

Matrix Multiplication



In this assignment you will extend the serial matrix multiplication code provided, to create a parallel version for the PS4. You should build on your knowledge of SIMD parallelism; and of POSIX Threads. The code you develop can be created for testing in a cross-platform manner; allowing you to develop your code both on the PS4 Pro Development Kit, as well as a common desktop computer.

You should work individually. Submit the code you develop, along with a short pdf report of approximately 1000 words; as a single zip to Moodle by the date noted above. Submit your Visual Studio project, and any necessary

resources. Please do not submit the executable or intermediate object files: choose "Clean Solution" before zipping your submission.

Methodology

You are provided with a C++ file which multiplies two square matrices. The code is configured to multiply a simple matrix with the identity matrix; the result should therefore be the original matrix. This is checked using operator== by the expression mres==m.

Create a chip-parallel SIMD verion of the matrix multiplication routine. Refer to the slides from session #5, and the lab material from session #6, to produce a version which works with matrix elements which are either single-precision float s or double-precision doubles. These will use __m128 and __m256d respectively; and intrinsics including _mm_mul_ps and _mm256_mul_pd.

Also create a shared-memory parallel version using POSIX Threads. You may want to assign each core the task of calculating one row of the output matrix in a round-robin fashion. This should ideally work generically on single or double precision data, and upon arbitrary matrix sizes (order). To start with, you may wish to focus on an $8 * 8$ matrix for simplicity.

You should finally vary the size of the matrices, and benchmark performance on the PS4 Pro development kit. The size of the matrices is defined by the global macro value SZ. You may consider scripting the benchmarking: `orbis-clang++` preprocessor switches such as `-D SZ=1024` can help here. Ensure you are obtaining the correct results throughout.

Requirements

- Implement SIMD matrix multiplication for single-precision data
- Implement SIMD matrix multiplication for double-precision data
- Implement matrix multiplication using POSIX Threads
- Benchmark how varying the matrix size from say `(1<<3)` to `(1<<10)` affects performance.
- Benchmark how varying the number of threads/cores affects performance
- Repeat the benchmarking for both single-precision and double-precision matrices
- Ensure you run each benchmark a few times (say 6). Take the average, and calculate the standard deviation.
- Create a report to explain your approach, and the feature set you provide. Include graphs to examine all aspects of performance.

Marking Scheme

The assignment is worth 30% of the marks awarded for the entire COMP10037 module. The following provides a breakdown of the marking scheme:

SIMD single-precision implementation	5
SIMD double-precision implementation	5
Parallel POSIX Threads implementation	5
A comprehensive set of benchmark results	5
1000 word report with figures	10

Plagiarism

Ensure your work is yours alone. You can discuss ideas with your fellow students regarding how to prepare a solution, but the *copying or sharing of code is not permitted*.

Anonymity

Please use only your Banner ID to identify yourself in your submission. Be aware that version control metadata might also provide your name within Visual Studio.