Lab 1: Optimizing Serial Code

Goals

Predict the expected performance improvement for applying cache, SIMD, and data-parallelization optimizations to the color conversion (RGB to YCbCr) and downsampling portions of the serial application code and analyze the actual results compared to your predictions. Note that we are only looking at the RGB to YCbCr and downsampling parts of the application here, i.e. the `convertRGBtoYCbCr` and `downSample` functions. So all performance improvements are relative to how they performed in the baseline version, ignoring the rest of the program.

Part I: Estimating the impact

1. Read up online about the processor in the lab machines so you understand their capabilities, speeds, and design.
2. Use the profiler to analyze the behavior of the color conversion and downsampling code. Determine what you can do to improve the cache behavior, and estimate the performance impact of making that change. You should address the following:
    a. Data access pattern
    b. Cache hierarchy sizes and cache line size
    c. Prefetcher behavior
    d. Data movement between color conversion and downsampling (think blocking)
3. Analyze the code to determine where you can use SIMD instructions and estimate the performance impact of using them. You should address the following:
    a. Scalar issue width
    b. SIMD issue width
    c. Data element size
    d. Instruction dependencies
    e. What the compiler does for you
4. Analyze the code to determine where you can use OpenMP to parallelize the computation. Estimate the performance impact of parallelization. You should address the following:
    a. Available parallelism
    b. Shared cache and bandwidth effects

Note that these optimizations can be easily combined in six possible ways. You should do the above analysis and fill in the table below. Please make the estimates and explain why you expect those speedups before continuing to part 2. Make your best guess here. What is important is your analysis, not your absolute result. You should address each of the issues listed above as well as any others you feel will have an impact on the speedup.

B = Baseline (provided code)
C = Cache Improvements
S = SIMD Improvements
P = Parallel Improvements (4 cores)

Predicted Performance Impact

Optimization	Predicted Performance Impact	Why? (How did you come up with this impact? Address the issues above and any others you feel are important.)
B	Speedup = 1.0	This is the baseline. The speedup is B/B, which will always be 1.0. I'm sure you can do better below!
C		
S		Hint: SIMD is 256 bits, which is 8 floating point operations. All of the math is fully SIMDizable, so we would expect 8x performance on the math. What about data movement? Do you have the bandwidth for 8x the math performance?
P		
C+S		
C+P		
S+P		
C+S+P		

Part 2: Measuring the impact

Implement each of the combinations above and measure the actual performance improvement. (Note that most of the changes are orthogonal, so apart from some copying and pasting it shouldn't be a lot of extra work to do all 7 combinations.)

Please make sure you've completed part 1 before you do this part.

Actual Performance

	B	C	S	P	C+S	C+P	S+P	C+S+P
Measured Time								
Predicted Speedup	1.0							
Measured Speedup	1.0							

In addition to the above combinations, for the parallel ones you should investigate the parallel scaling by looking at the speedup as a function of the number of cores you allow OpenMP to use.

Actual Parallel Scaling

	1 core	2 cores	4 cores
P			
C+P			
S+P			
C+S+P			

Analysis

You have now tried to predict the impact of three very different optimizations as well as implemented them and measured the results. The point of this lab is twofold: 1) analyze code to predict the performance benefit of an optimization and 2) get better performance by optimizing. Your lab report should address these two issues.

Note: The most important thing is to show that you understand what is going on. If your estimates were far off the actual performance then you should analyze the real results and use the profiler to figure out what is really happening. (As well as talk to the TA.) Your report should explain what is really going on and what you missed when you did your first estimates. You will not lose any points for having plausible, but incorrect, initial estimates if you plausibly identify what was wrong about them from the actual implementation. (Indeed, that's the whole point!)

You should produce a 2-page, concise report with the following 4 sections:

- **1. Performance Estimates**: Your estimates of the performance benefits of each of the optimizations and how you came up with them.
- **2. Performance Achieved**:
  - Your measured performance results from implementing the optimizations.
  - The parallel scaling of your implementation.
- **3. Discussion**:
  - A discussion of the measured results and how they differ from your predictions.
  - A discussion of what you learned about these optimizations from implementing them and measuring the results.
  - Comment on any unexpected or odd results.
  - Comments on the difficulty of each optimization.
  - Comment on what the compiler did for you.
- **4. Lab comments**: Any feedback on the lab itself.

The comparative results should be presented in a graphical form to make it easy to see trends and analysis.