# Programming Assignment

Due April 10th, Friday, 11:59 PM CST

**Overview**

This assignment is intended to help you develop an understanding of parallel programming. You will also gain experience measuring and reasoning about the performance of parallel programs (a challenging, but important, skill).

JPEG (http://en.wikipedia.org/wiki/JPEG) is a commonly used method of lossy compression for digital images. In this project, your job is to parallelize the JPEG encoding computation of the images using pthreads. A C++ program for JPEG encoding algorithm is provided. BMP images are also provided as the input to the program. You will need to make use of pthread API calls such as `pthread_create and pthread_join` in this assignment.

The encoder normally includes the following steps:

Divide the image into 8*8 blocks and do the following for each block
1. Shift the block
2. Perform a DCT on the block
3. Quantize the block
4. Subtract the last DC coefficient from the current DC coefficient
5. Zigzag the block
6. Zero run length encode the block
7. Break down the non-zero coefficients into variable-length binary numbers & their lengths
8. Entropy encode the run lengths & binary number lengths9.

Write the entropy encoded information & binary numbers to the output

**What you need to do:**

1. Modify the starter code to parallelize the JPEG encoding using two processor cores. Specifically, compute the top half of the image in thread 0, and the bottom half of the image in thread 1. This type of problem decomposition is referred to as *spatial decomposition* since different spatial regions of the image are computed by different processors.
2. Extend your code to utilize 2, 3, and 4 threads, partitioning the encoding work accordingly. In your write-up, produce a graph of **speedup compared to the reference sequential implementation** as a function of the number of cores used. Is speedup linear in the number of cores used? In your write-up hypothesize why this is (or is not) the case?
3. To confirm (or disprove) your hypothesis, measure the amount of time each thread requires to complete its work by inserting timing code at the beginning and end

of each thread. How do your measurements explain the speedup graph you previously created?

4. Try to modify the mapping of work to threads to achieve to improve speedup to at **about 3.5x**. We are expecting you to come up with a single work decomposition policy that will work well for all thread counts---no hard coding a solution specific to each configuration. In your writeup, describe your approach and report the final 4-thread speedup obtained.