Assignment 3
Homework (no programming)

Release Time	Due Date
October 27, 2016	November 3, 2016

Objectives

- Gain more in-depth understanding of sorting techniques
- Design HPC solutions – in particular merge-sort
- Start to get a flavor (theoretical design) of multicore shared memory solutions
-

General Instructions: As I have mentioned in class, for many of the homework or programming problems we give in the class, solutions are readily available on the web; however, if you want to learn problem solving, designing HPC solutions (sequential or multicore or parallel or distributed), then please refrain from looking up solutions or discussing it with a colleague. First exercise your brain by thinking of possible solutions yourself. Only if you come to a dead-end, discuss with someone or look-up. Of course, even after you found a solution elsewhere, understand it thoroughly and then express it in your own words (*Pause*: changing sentence structure, variable names, method names etc. is all too easy and easily identifiable, further there are even automated ways that can detect similarity, do you want to risk of being accused of plagiarizing?). When you have found the solution elsewhere, you must give credit to the original contributor.

Problem Specifications

1. In class we discussed how to convert the recursive version of merge-sort by unfolding the recursive steps to an iterative version when n, the number of elements to sort, is a power of two (i.e., $n = 2^d$, for some $d \geq 0$). Using similar approach, extend the design and analyze an efficient iterative solution to merge sort for any value of $n \geq 0$. You need to give pseudo-code and document the code so a layman can understand what your code is trying to do.

2. Let us get an intuitive start on designing multi-core HPC solutions when single-core solutions may not be sufficient. Assume that you are given an array A[0..n-1] of n elements in shared - memory and that you have eight PEs (aka processing elements / cores / CPUs) in your machine. Each PE can access a shared memory location in $\Theta(1)$-time. (In other words, each PE can read or write an element from the shared memory in $\Theta(1)$-time; of course the type of elements has to be same as A or one of the primitive types, e.g., int, float, Boolean, double, char.) At most one PEs is allowed to access a memory location in the shared memory at any given time (i.e., two PEs can't access the same memory location x at any given time t, so if they need to access the memory location x, they will have to do that at different times). Each PE has a small amount of fast local memory (such as registers) to execute a basic instruction in $\Theta(1)$-time on the data that is in its local memory. Design and analyze an efficient solution to sort the list A using the eight PEs efficiently using merge-sort. Again pseudo-code is sufficient but explain your design so anyone (with a reasonable programming expertise) can understand your solution easily. You may want to give a high-level description of your idea first and then give pseudo-code. For 20 points extra credit, generalize your solution so it can work for p-PEs where $p = 2^d$, for some $d \geq 0$.

Assignment Submission

- A single pdf file using appropriate naming conventions.
- Beginning of the file should clearly identify you, the class, submission date, and the main goals of the homework. Also add credit to others if you had to resort to looking up the solution elsewhere. Finally add one of the sentences: “I give permission to the instructor to share my solution(s) with the class." or “I do NOT give permission to the instructor to share my solution(s) with the class."