Problem 1. (20 points) [CS&G 2.3] There are two dominant models for how parent and children processes relate to each other in a shared address space. In the heavyweight so-called process model, when a process creates another process, the child gets a private copy of the parent's image; that is, if the parent had allocated a variable \( x \), then the child also finds a variable \( x \) in its address space, and the variable is initialized to the value that the parent had for \( x \) when it created the child. However, any modifications that either process makes subsequently are to its own copy of \( x \) and are not visible to the other process. In the lightweight threads model, the child process or thread gets a pointer to the parent's image, so that it and the parent now see the same storage location for \( x \). All data that any process or thread allocates is shared in this model, except the data on a procedure's stack.

(a) Consider the problem of a process having to reference its process identifier \( p.i.d \) in various parts of a program, in different routines called (in a call chain) by the routine at which the process begins execution. How would you implement this in the first model? In the second? Do you need private data per process, or could you do this with all data being globally shared?

(b) A program written in the process model may rely on the fact that a child process gets its own private copies of the parent's data structures. What changes would you make to port the program to the threads model for data structures that are (i) only read by processes after creation of the child and (ii) are both read and written?

Problem 2. (25 points) Under the data-parallel model, many processors perform operations in parallel on separate elements of a larger data structure. Once each processor has calculated the corresponding values, it must exchange information before continuing execution. One method for communication is the reduction operation where the values from many processors are reduced into one value.

Consider a data-parallel architecture consisting of an 8x8 grid of processing elements. There are up to 64 individual processing elements that can execute instructions at the same time, and a control processing element to perform the control. Let the variables \( i \) and \( j \) denote the row and column, respectively. The top left processor would have indices (0,0) and the bottom right would have indices (7,7).
(a) Write pseudo-code for the reduction operation for the data parallel model assuming logarithmic convergence. For example, with 64 elements, only 6 reduction operations would be necessary for the values to be reduced to a single processor because $2^6$ is 64.

The **operation** keyword can be used in the pseudo-code for any generic operation that may be applied to a reduction. Accessing values can be accomplished by referencing another processor in the grid via the $i$ and $j$ ordered pair. You may assume that the specifics of broadcasting values to other processors will be handled by the control processor. Assume that each processor in the grid contains data within the variable `var` that needs to be reduced.

(b) What types of operations may be substituted in the **operation** keyword for successful reduction? List a couple of examples and describe characteristics that allow them to be reduced.

**Question 3.** (25 points) Here are four ways cache can be organized:

- Direct
- Fully associative
- Set associative
- Sectored

Let us assume that the cache contains 2048 words, 16 words per line, and 128 lines.

(a) Which organization strategy(s):

- (i) Has the least amount of tags needed to be compared for word look-up? Most?
- (ii) Benefits *least* from using a least recently used (LRU) replacement policy? Most?

(b) If I have a Main-Memory address of [11 0011 0011 0011 0011], what line is the data fetched into with each cache organization? (The answer has to be between 0 and 127).

- (i) Direct
- (ii) Fully associative
- (iii) Set associative (with 64 sets)
- (iv) Sectored (with 8 sectors)
Problem 4. (30 points) Suppose that a program takes $n^3 + n^2$ units of time to execute sequentially.

(a) Supposing that the $O(n^2)$ work may be done in parallel, and that we have $p$ processors, what is the maximum speedup?

(b) As $p \to \infty$ in part a, the highest speedup approaches what value? *Hint:* Consider L'Hôpital's rule.

(c) Suppose that the $O(n^3)$ work may be done in parallel, and that we have $p$ processors, what is the maximum speedup?

(d) As $p \to \infty$ in part (c), the highest speedup approaches what value?

(e) Explain why there is such a large discrepancy between the answers for speedup between part (b) and part (d).

(f) Draw the concurrency profiles for (i) when the entire program is serial; (ii) when only the $O(n^2)$ work can be done in parallel, and (iii) when only the $O(n^3)$ work can be done in parallel.

(g) Are the concurrency diagrams obtained in part (f) regular or irregular?