1. (36 points total)
This problem is about computer performance modeling as developed in the Stone text. Stone's model for unoverlapped communication assumes that each task communicates with every other task. We will assume a different communication pattern. Suppose each processor only needs to communicate with two other processors, regardless of the number of tasks assigned to each processor. For example, in each processor, one task receives an incoming communication and one task sends outgoing communication. Tasks internal to a processor may pass information to each other, but do not require communication. Communication and computation are not overlapped, and communication occurs sequentially.

Suppose we have $N$ processors and $M$ tasks. Let $R$ represent computation time for each task. Let $C$ represent communication overhead between tasks on different processors. These parameters are as in Stone. State any additional assumptions you make.

Initially we do not know the number of processors that will be used. It could be any number from 1 to $N$.

(a) (9 points)
If the goal is to minimize total execution time, whether running on 1 or 2 or any number of processors up to $N$, what pattern of assignment of tasks to processors would you apply? Justify your answer.

(b) (9 points)
Using the assignment of tasks to processors that you selected in part (a), develop an equation for total execution time as a function of $P$, the number of processors in use. This should be a general equation, not one for specific values of $R$ or $C$. 
(c) (9 points)
Derive an equation for the number of processors that minimizes total execution time.

(d) (9 points) Suppose C is fixed, but R can be changed by changing the number of tasks. How does the granularity of the application affect the selection of the number of processors that minimizes the total execution time?

2. (36 points total)
Consider an MIMD computer system having a physically distributed memory with a global address space. Each processor has an 8 MB local memory that is part
of the total address space. Each processor also has a 512 KB cache that is accessible only to it. Remote accesses take much longer than local or cache accesses.

<table>
<thead>
<tr>
<th>Type of Access</th>
<th>Relative Access Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>cache</td>
<td>1</td>
</tr>
<tr>
<td>local</td>
<td>10</td>
</tr>
<tr>
<td>remote</td>
<td>100</td>
</tr>
</tbody>
</table>

Suppose an application requires a data set of size 32MB for execution. We are interested in speedup and scalability of this application as run on various numbers of processors, thus we assume the data set is loaded into memory before execution and the start of timing. Assume also that the application can be partitioned into identical pieces operating on different parts of the data set without requiring communication between processors. This falls in the SPMD (single program, multiple data) mode of operation.

The application was run on various numbers of processors with the following empirical results.

<table>
<thead>
<tr>
<th>Number of Processors Used</th>
<th>Execution Time</th>
<th>(a) Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2512</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>896</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>88</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>44</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>22</td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>11</td>
<td></td>
</tr>
<tr>
<td>64</td>
<td>5.05</td>
<td></td>
</tr>
<tr>
<td>128</td>
<td>2.525</td>
<td></td>
</tr>
</tbody>
</table>

(a) (9 points)
Fill the Speedup column in the table with speedup relative to using one processor. Two decimal places is sufficient.

(b) (9 points)
Why is the speedup superlinear? That is, why is the speedup greater than the number of processors used? Justify your answer.

(c) (9 points)
Speedup appears to be linear in the range from 4 to 32 processors. That is, doubling the number of processors doubles the speedup. Explain why speedup deviates from the linear pattern as you move from 32 to 64 and 128 processors. (This deviation may be easier to see if you compute speedup relative to using 16 processors.)

(d) (9 points)
Let us define scalability as the ability to gain significant performance improvement for an application from an increase in the number of processors used. Is this application scalable on this system? Suppose interprocessor communication were required, with an R/C ratio of 1. Would scalability improve or not? Justify your answer.
3. (28 points total)
A multiprocessor (MIMD) computer has a shared memory accessed by all
processors through a single shared bus. Each processor also has a write-back
cache. A processor must implement cache coherence by "snooping" on the
memory bus. For this cache, we define the following terms:

- A cache block is dirty if it has been updated, but not yet written back to
  memory.
- An invalidate signal commands a cache to purge a particular block.

The algorithm for processor P to read memory block m is shown below:
If block m is in P's cache, then
    Read block m from the cache
else {
    Replace a block in P's cache if necessary and write it back to
    memory if it is Dirty
    Read block m from memory into P's cache, and mark it Clean
}

The algorithm for processor P to write memory block m is shown below:
If block m is in P's cache and is Dirty, then
    Update the value of block m in P's cache,
else {
    If block m is not in P's cache, then {
        Replace a block in P's cache if necessary and write it back
        to memory if it is Dirty
        Read block m from memory into P's cache
    }
    Signal other caches to Invalidate block m
    Update the value of block m in P's cache, and mark it Dirty
}

The "snooping" algorithm (responding to bus traffic) for Processor P for a block
m which is in P's cache is:
If block m is read by another processor Q, and m is Dirty in P's cache,
then {
    Write back block m to memory
    Mark block m Clean in processor P's cache
}
If block m is Invalidated by another processor Q, then {
  If block m is Dirty in processor P's cache, then
    Write it back to memory
    Invalidate block m in processor P's cache
}

(a) (14 points)
Write a formula for the effective memory access time in this computer. Define and use whatever parameters you need, and state all assumptions you make.

(b) (14 points)
The above algorithms define how a cache block is controlled. This can be implemented by a controller (state machine) with states: Invalid (no block present), Clean (block present, not updated), and Dirty (block present, and updated). Inputs to the controller are: PRead, PWrite, QRead, Hit (in cache), and InvalidateIn. Outputs of the controller are InvalidateOut, ReadMemory, and WriteMemory. Draw the state machine diagram for this controller, or write the equations for its operation.