EE 7700-4 **Homework 4: 25 November 1998, 17:00 CST** **Due 4 December 1998**

Problem 1: Determine an approximate run time for each phase of the program below, taking into account only the time for accessing the array, `a`. The system has the following characteristics:

- The system consists of 16 processors (not all are used). Each processor has a 2^{14}-set 4-way set-associative cache with a line size of 16 bytes.
- The array is mapped so that the first 160 elements are at node 0, the next 160 elements are at node 1, and the last 160 elements are at node 2. Each array element is four bytes.
- The directory protocol described in Section 8.4 of the text is used.
- Initially the array is not in any cache.
- The processor can issue one load or store instruction per cycle and zero time is needed for other instructions.
- A load or store instruction is not executed until the previous one completes.
- A load or store hitting the cache takes 1 cycle.
- The protocol processor takes 5 cycles to process each received message or miss, plus, if necessary, another 10 cycles to access memory (including reading and writing the directory). (No additional time is needed for the protocol processor to access the cache.)
- The time needed to send a protocol message is 10 plus $\lceil L/4 \rceil$ cycles, where L is the length of the message. The message length is 4 bytes (all messages) plus 4 bytes if an address is present plus the space needed for any data. The same protocol processor is used for a node's cache and memory; there is no need to send a message from the cache to memory or memory to cache of the same node.

```
/* proc: processor number.
   i:    Local variable.
   a:    Pointer to shared array. */

/* Phase 1 */

if( proc == 0 )
  for(i=0; i<480; i++) a[i] = i;

barrier();

/* Phase 2 */

if( proc == 1 )
 for(i=0; i<480; i++) a[i] = a[i] + 1;
```

Problem 2: Extend the directory protocol presented in Section 8.4 so that lines can be upgraded. A line is said to be *upgraded* when a cache having a shared copy of the line gets an exclusive one. Because the cache already has the line, the message granting the upgrade contains no data, reducing traffic volume.

Consider two possibilities: in one, the cache can request an upgrade; in the other, the cache sends an ordinary write miss message to the memory. Choose and justify the better approach. The protocol must work correctly even if the line being upgraded is replaced or invalidated between the write miss and the upgrade (by re-issuing the write request).

Indicate the new protocol messages needed and give state transition diagrams for the cache and memory controllers. Give an example illustrating an upgrade.

Problem 3: The code below is used by all 16 tasks in a parallel program to obtain a unique id number. The compiler will use a fetch-and-increment instruction for the code, guaranteeing atomicity. Estimate the shortest execution time for the code for a memory-only and a modify-in-cache fetch-and-increment implementations. (Execution starts when the first processor executes the code and ends when the last processor finishes execution.) Specify when each processor starts to execute the fetch-and-increment instruction and any other conditions needed for best-case timing. The memory controller sends NAK (negative acknowledgment, also called busy) messages in response to Write Miss messages for a block if it is waiting for Data Write Back messages needed for an earlier Write Miss message. Use the timings given in the first problem. State any assumptions.

```
myid = id_counter++;
```

Describe why the program running on a system with modify-in-cache fetch-and-increment instructions would take longer than the best case time.