CS 740, Fall 1997
Assignment 6:
Part II: Directory-Based Cache Coherence

Assigned: Wed., Nov. 19
Due: Fri., Dec. 5

Policy

You may work in a group of up to 3 people in solving the problems for this assignment. You should turn in a single report for your entire group, identifying all of the group members.

Logistics

Any clarifications and revisions to the assignment will be posted on the class bboard and Web page. There is no electronic handin for this assignment.

Background

The following problems are modified versions of ones that appeared on last year's final exams. Overall, the students didn't do very well. We attribute this to a failure on our part to cover the material adequately in class. Hopefully, things will be better this year.

The purpose of this part of the assignment is for you to reason about the relation between a programmer's view of the memory system and memory consistency model, and the mechanisms used to realize this model. You'll find Sections 8.4 and 8.6 of the book helpful.

Consider a distributed shared memory system using a directory-based cache coherency scheme, similar to the one described in pp. 679–685 of the text. Each memory block has a "home", i.e., a cache controller responsible for managing its status. All of the processors follow a common protocol for accessing the shared memory state that guarantees that at any given time each block is in one of the following states:

Uncached: There are no remote copies of the block.

Shared: There are one or more read-only copies of the block in remote caches.

Exclusive: There is a single, writeable copy of the block in a remote cache.

As an example of the protocol, suppose three processors P1, P2, and P3 have shared copies of some block, and P1 wants to gain exclusive access in order to write to a word in this block. A series of transactions to obtain this privilege is illustrated in Figure 1. In this figure, time moves downward, with each vertical line denoting the actions of an individual agent, and the arrows denoting communications among them. The communications are:

IReq P1 sends a invalidation request to the home.

ICmd The home sends invalidation commands to the other processors having copies of the block

IAck The other processors acknowledge the invalidation command

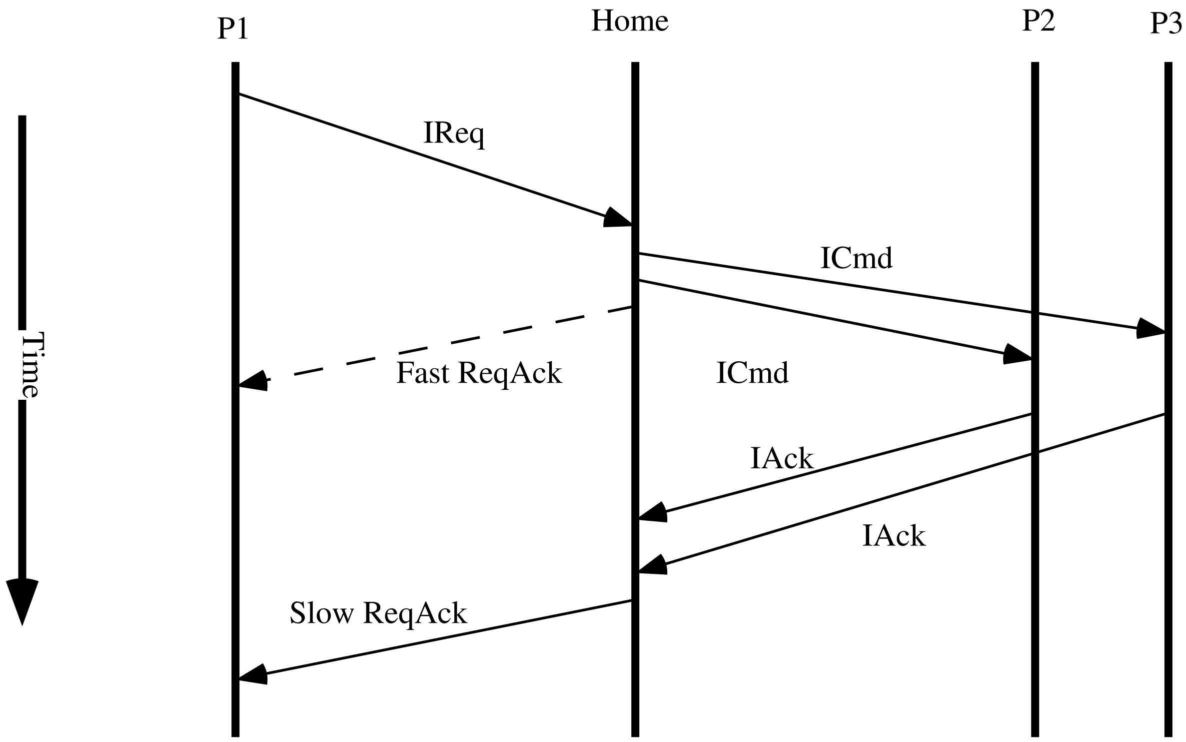

Figure 1: Transactions for P1 to gain exclusive access to previously-shared block

ReqAck The home acknowledges the original request.

In this diagram, two versions of ReqAck are diagrammed: "Fast" where the request is acknowledged immediately, and "Slow" where the home waits until it is sure all other copies have been invalidated.

Problem 1

Suppose that the protocol were such that a processor could begin writing to its block before it receives any acknowlegement. Describe a scenario in which this could have disasterous consequences, where two processors accessing independent memory words could interfere.

Problem 2

Suppose we use the "Fast" version of ReqAck, such that a processor can begin writing to its block as soon as it receives this acknowledgement. Describe a scenario in which a concurrent program (e.g., the one shown at the bottom of H&P page 708) could violate sequential consistency.

Problem 3

The MIPS instruction set includes an instruction `SYNC` which, when executed on a processor, guarantees that all preceding memory operations by the processor are completed before any successive ones are begun. This is a form of "memory fence" operation described on H&P page 713.

Describe how such an instruction could be implemented. State what resources the processor would require, how they would be managed, and and how it would behave with respect to the transactions diagrammed in Figure 1. In particular, describe how a processor can determine whether its memory operations have been "completed."