5.9 a. P0,0: read 100 — L1 hit returns 0x0010, state unchanged (M)

b. P0,0: read 128 — L1 miss and L2 miss will replace B1 in L1 and B1 in L2 which has address 108.

L1 will have 128 in B1 (shared), L2 also will have it (DS, P0,0)

Memory directory entry for 108 will become <DS, C1>

Memory directory entry for 128 will become <DS, C0>

c, d, …, h: follow same approach

5.10 a. P0,0: write 100 ← 80, Write hit only seen by P0,0

b. P0,0: write 108 ← 88, Write "upgrade" received by P0,0; invalidate received by P3,1

c. P0,0: write 118 ← 90, Write miss received by P0,0; invalidate received by P1,0

d. P1,0: write 128 ← 98, Write miss received by P1,0.

5.11

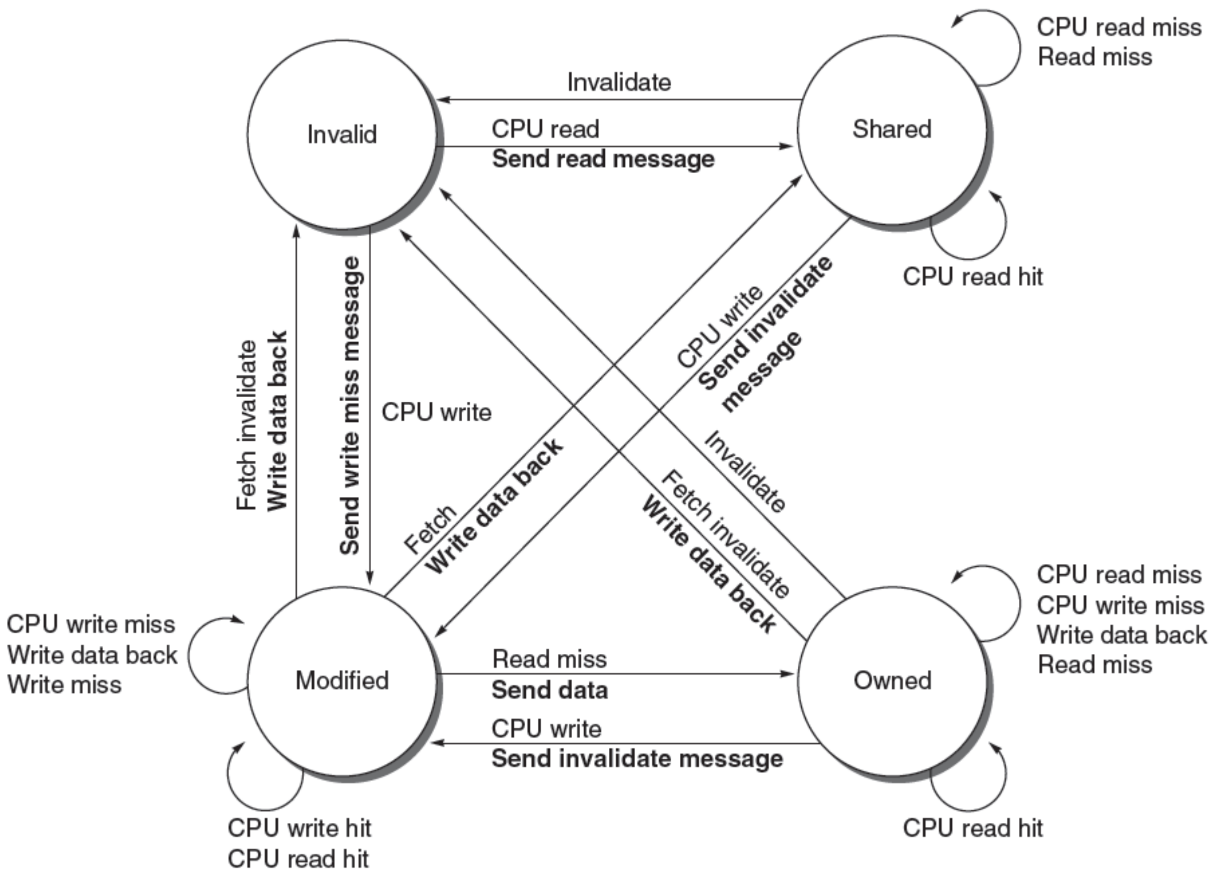

Figure S.30 Cache states.

5.12 The Exclusive state (E) combines properties of Modified (M) and Shared (S). The E state allows silent upgrades to M, allowing the processor to write the block without communicating this fact to memory. It also allows silent downgrades to I, allowing the processor to discard its copy with notifying memory. The memory must have a way of inferring either of these transitions. In a directory-based system, this is typically done by having the directory assume that the node is in state M and forwarding all misses to that node. If a node has silently downgraded to I, then it sends a NACK (Negative Acknowledgment) back to the directory, which then infers that the downgrade occurred. However, this results in a race with other messages, which can cause other problems.

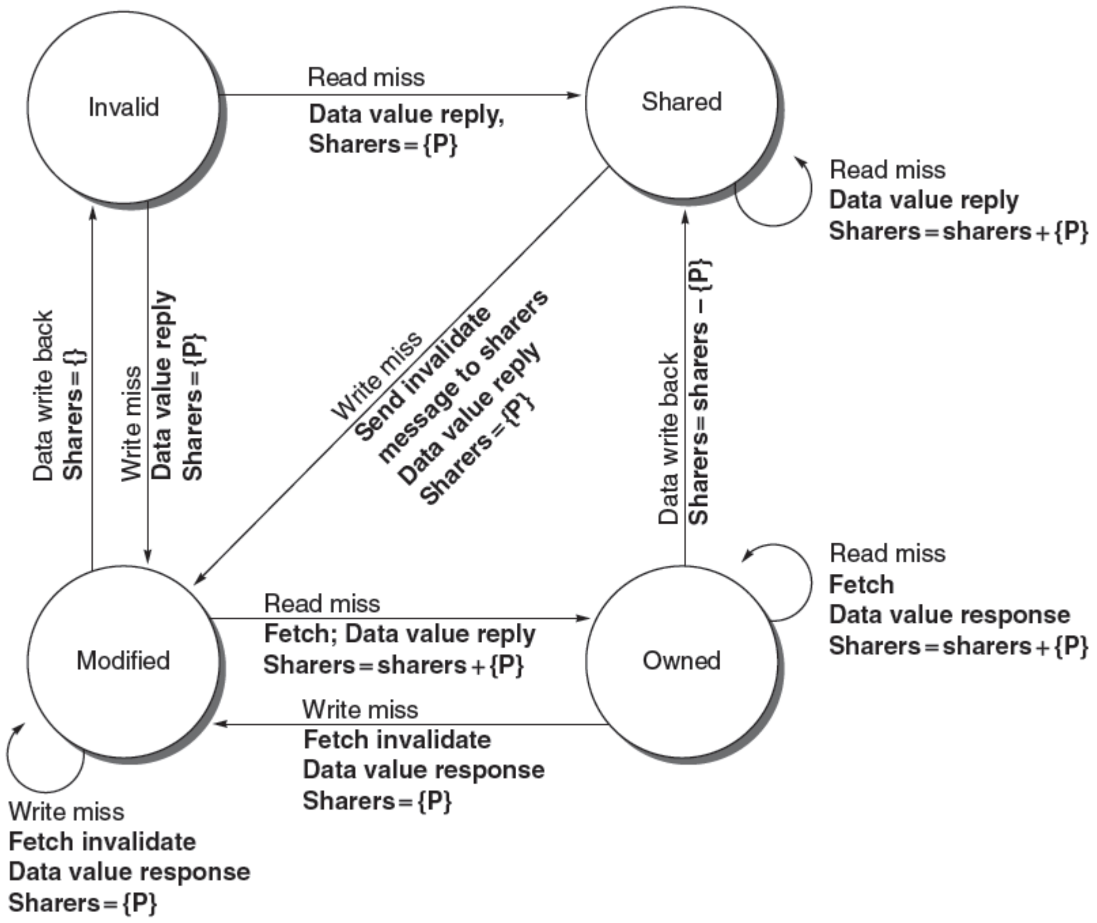

Figure S.31 Directory states.

5.19 The general form for Amdahl's Law is

$$Speedup = \frac{Execution\ time_{old}}{Execution\ time_{new}}$$

all that needs to be done to compute the formula for speedup in this multiprocessor case is to derive the new execution time.

The exercise states that for the portion of the original execution time that can use *i* processors is given by *F(i,p)*. If we let *Execution time*$_{old}$ be 1, then the relative time for the application on *p* processors is given by summing the times required for each portion of the execution time that can be sped up using *i* processors, where *i* is between 1 and *p*. This yields

$$Execution\ time_{new} = \sum_{i=1}^{p} \frac{f(i,p)}{i}$$

Substituting this value for *Execution time*$_{new}$ into the speedup equation makes Amdahl's Law a function of the available processors, *p*.

5.20

a. (i) 64 processors arranged a as a ring: largest number of communication hops = 32 → communication cost = $(100 + 10 \times 32)$ ns = 420 ns.

(ii) 64 processors arranged as 8x8 processor grid: largest number of communication hops = 14 → communication cost = $(100 + 10 \times 14)$ ns = 240 ns.

(iii) 64 processors arranged as a hypercube: largest number of hops = 6 ($\log_2$ 64) → communication cost = $(100 + 10 \times 6)$ ns = 160 ns.

b. Base CPI = 0.5 *cpi*

(i) 64 processors arranged a as a ring: Worst case CPI = $0.5 + 0.2/100 \times (420) \times 3.3 = 3.272$ *cpi*

(ii) 64 processors arranged as 8x8 processor grid: Worst case CPI = $0.5 + 0.2/100 \times (240) \times 3.3 = 2.084$ *cpi*

(iii) 64 processors arranged as a hypercube: Worst case CPI CPI = $0.5 + 0.2/100 \times (160) \times 3.3 = 1.556$ *cpi*

The average CPI can be obtained by replacing the largest number of communications hops in the above calculation by $\hat{h}$, the average numbers of communications hops. That latter number depends on both the topology and the application.

c. Since the CPU frequency and the number of instructions executed did not change, the answer can be obtained by the CPI for each of the topologies (worst case or average) by the base (no remote communication) CPI.

5.32 a. Because flag is written only after A is written, we would expect C to be 2000, the value of A.

b. Case 1: If the write to flag reached P2 faster than the write to A.

Case 2: If the read to A was faster than the read to flag.

c. Ensure that writes by P1 are carried out in program order and that memory operations execute atomically with respect to other memory operations.

To get intuitive results of sequential consistency using barrier instructions, a barrier need to be inserted in P1 between the write to A and the write to flag.