COE 501: Computer Architecture

Problem Set 6: Cache Coherence

Solution

1) (10 pts) The bus-based dual-core multiprocessor shown in Figure 1 represents a symmetric shared memory architecture. Each processor has an L1 write-back private cache. Coherence is maintained using the MSI write-invalidate snooping protocol. For simplicity, each cache is directly-mapped with four blocks, and each block holds two words (8 bytes). For clarity, the tag contains the full address in hexadecimal, while the data is shown in decimal.

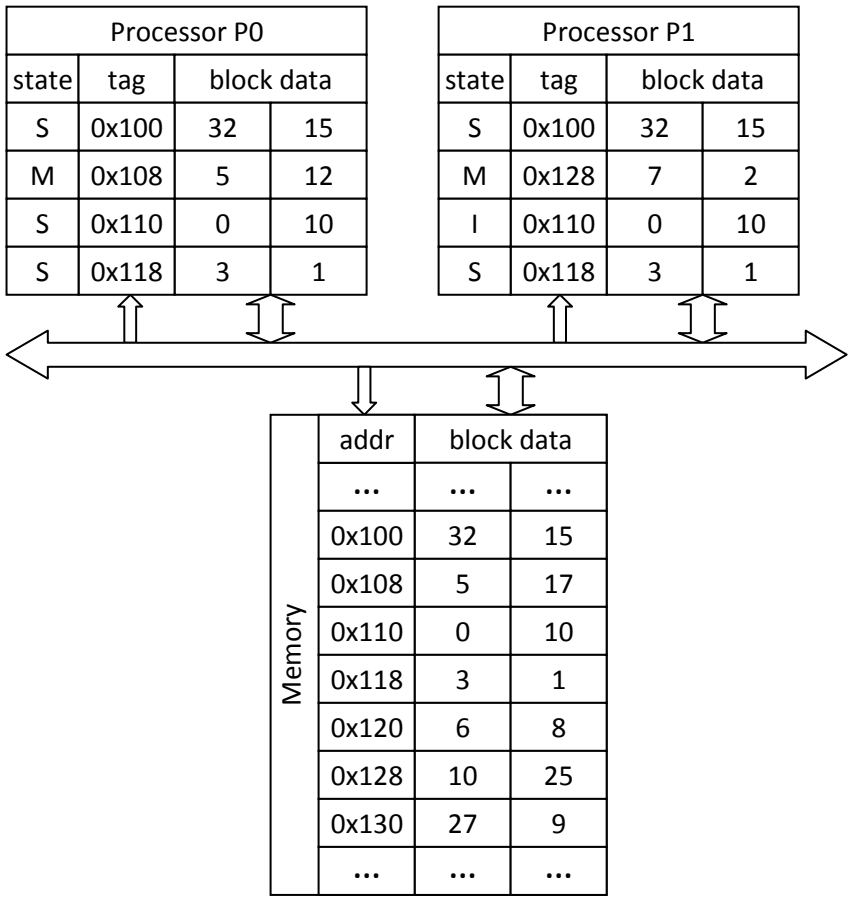


Figure 1: Bus-Based Dual Core multiprocessor

Each part of this exercise specifies a memory operation. Treat each operation as independently applied to the initial state given in Figure 1. What value is returned (by a read) and the bus transaction (if any), the resulting state, tag, and value of the caches and memory after the given operation? Show the content of the relevant cache blocks.

a) P0 reads address 0x120
b) P0 writes address 0x120 ← 38
c) P0 reads address 0x128
d) P1 reads address 0x114
e) P1 writes address 0x11C ← 14

Solution:

a) (1.5 pts) P0 reads address 0x120
P0 places Read Miss on bus at address 0x120 ➔ Transfer block at 0x120 in memory to P0
Processor P0 reads Block 0: (S, 0x120, 6, 8)
Returns 6

b) (1.5 pts) P0 writes address 0x120 ← 38
P0 places Write Miss on bus at address 0x120 ➔ Transfer block at 0x120 in memory to P0
Processor P0 writes Block 0: (M, 0x120, 38, 8)

c) (3.5 pts) P0 reads address 0x128
P0 Write-Back Block 1 which is replaced: (M, 0x108, 5, 12)
P0 places Read-Miss Transaction on bus at address 0x128
P1 Write-Back Block 1 in modified state: (M, 0x128, 7, 2)
P1 transfers Block 1 at address 0x128 to P0 (cache-to-cache transfer)
P1 changes state of Block 1 to shared: (S, 0x128, 7, 2)
Processor P0 gets Block 1 in shared state: (S, 0x128, 7, 2)
Returns 7

d) (1.5 pts) P1 reads address 0x114
P1 places Read Miss on bus at address 0x110 ➔ Transfer block in memory at 0x110 to P1
Processor P1, Block 2: (S, 0x110, 0, 10)
Returns 10

e) (2 pts) P1 writes address 0x11C ← 14
P1 places Invalidate transaction on bus, at block address 0x118
Processor P0 invalidates block 3: (I, 0x118, 3, 1)
Processor P1 writes block 3: (M, 0x118, 3, 14)

2) (10 pts) The performance of a snooping-cache coherent multiprocessor depends on many detailed implementation issues that determine how quickly a cache responds with data in the modified (M) state. In some implementations, a CPU read miss to a block in the M state in another processor's cache is faster than a read miss to a block in memory. This is because caches are smaller and faster than main memory. Consider the following latencies:

- CPU read and write hits generate no stall cycles
- CPU read and write misses generate N_{memory} and N_{cache} stall cycles if satisfied by memory and cache, respectively.
- CPU write hits that generate an invalidate incur $N_{invalidate}$ stall cycles.
- A write-back of a block, due to replacement or another processor's request, incurs an additional $N_{writeback}$ stall cycles.

For the multiprocessor of Figure 1, consider the following parameters:
N_{memory} = 80 stall cycles, N_{cache} = 10 stall cycles, $N_{invalidate}$ = 4 stall cycles, $N_{writeback}$ = 5 stall cycles.

Consider the following sequences of operations. Assuming the initial state of Figure 1, how many stall cycles are generated? Explain your answer and show the modified blocks.

a) P1 reads address 0x120
P1 reads address 0x124
P1 reads address 0x128

b) P0 reads address 0x100
P0 writes address 0x108 ← 37
P0 writes address 0x130 ← 29

c) P0 reads address 0x120
P0 reads address 0x100
P1 writes address 0x100 ← 24
P0 reads address 0x104

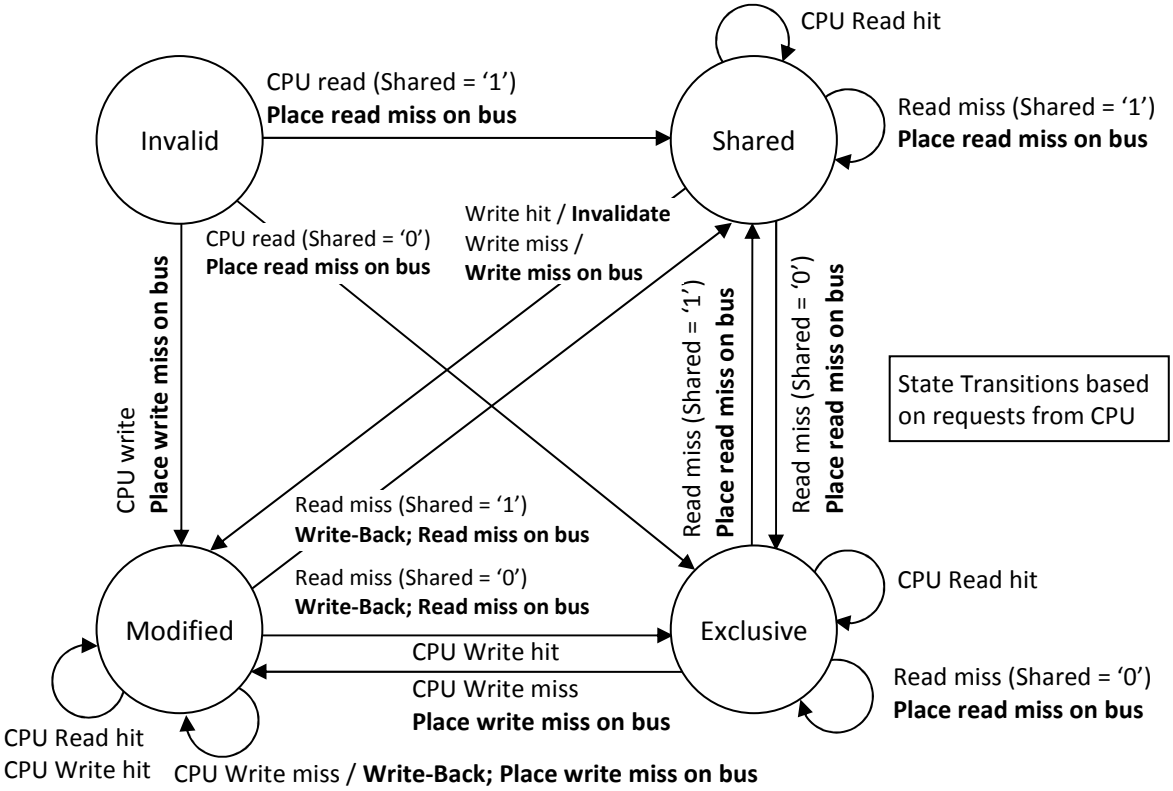
Solution:

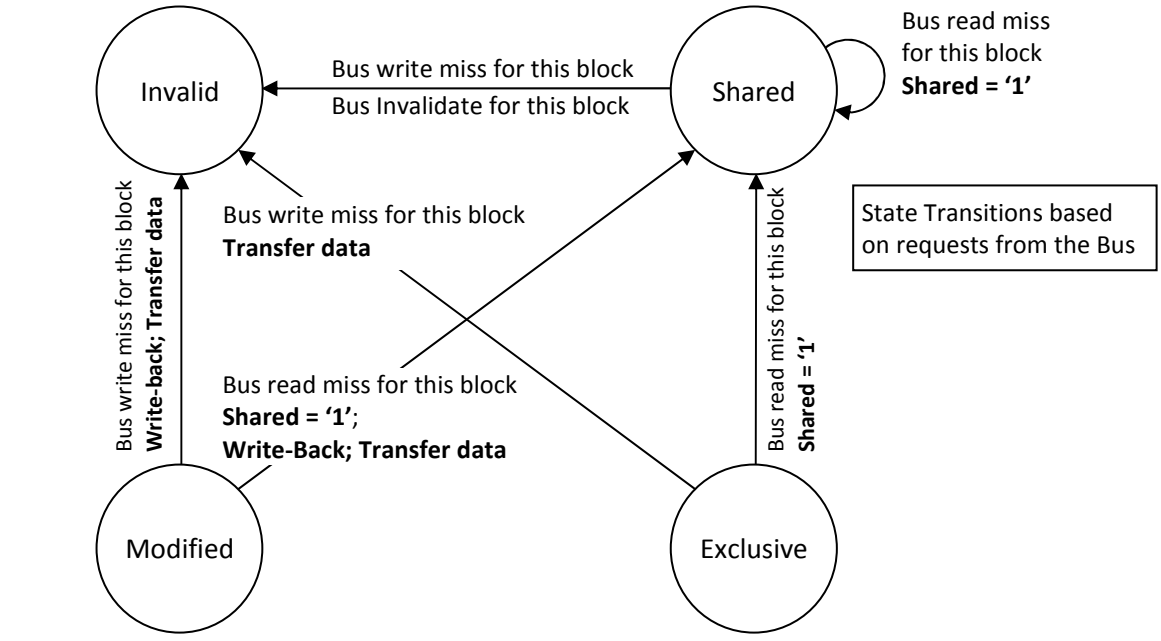
a) (3 pts) P1 reads address 0x120 → P1 places Read Miss on bus at address 0x120
Transfer block in memory at 0x120 to P1 → P1 block 0: (S, 0x120, 6, 8) → return 6
N_{memory} = 80 stall cycles
P1 reads address 0x124 → read hit → return 8 (zero stall cycles)
P1 reads address 0x128 → read hit → return 7 (zero stall cycles)
Total stall cycles = 80

b) (3 pts) P0 reads address 0x100 → read hit → return 32 → zero stall cycles
P0 writes address 0x108 ← 37
Write hit → P0 writes block 1: (M, 0x108, 37, 12) → zero stall cycles
P0 writes address 0x130 ← 29
P0 places write miss on bus at address 0x130
Transfer block in memory at 0x130 to P0 → P0 block 2: (M, 0x130, 29, 9)
N_{memory} = 80 stall cycles, Total stall cycles = 80

c) (4 pts) P0 reads address 0x120 → P0 places Read Miss on bus at address 0x120
Transfer block in memory at 0x120 to P0 → P0 block 0: (S, 0x120, 6, 8) → return 6
N_{memory} = 80 stall cycles
P0 reads address 0x100 (block 0 replaced) → P0 places Read Miss on bus at address 0x100
Transfer block in memory at 0x100 to P0 → P0 block 0: (S, 0x100, 32, 15) → return 32
N_{memory} = 80 stall cycles
P1 writes address 0x100 ← 24
P1 places invalidate transaction on bus at address 0x100
P0 invalidates block 0: (I, 0x100, 32, 15)
P1 writes block 0: (M, 0x100, 24, 15)
$N_{invalidate}$ = 4 stall cycles
P0 reads address 0x104 (block 0 invalidated) → P0 places Read Miss on bus at address 0x100
P1 writes-back block 0 and changes state to shared → P1 block 0: (S, 0x100, 24, 15)
$N_{writeback}$ = 5 stall cycles
P1 transfers block at address 0x100 to P0 (cache-to-cache transfer) → N_{cache} = 10 stall cycles
P0 block 0: (S, 0x100, 24, 15) → return 15 (address 0x104)
Total stall cycles = 80 + 80 + 4 + 5 + 10 = 179

3) (10 pts) Many snooping coherence protocols have additional states, state transitions, or bus transactions to reduce the overhead of maintaining cache coherence. An optimization to the MSI snooping protocol is to add the Exclusive (E) state, indicating that no other node has a copy of the block, but the block has not yet been modified. A cache block enters the Exclusive state when a read miss is satisfied by memory and no other node has a valid copy. CPU reads and writes to that block proceed with no further bus traffic, but CPU writes causes the state of the block to transition from Exclusive to Modified. There is a *Shared* wire on the bus that signals '1' when a read miss appears on the bus and at least one other processor has a copy of the block. Otherwise, the *Shared* wire signals '0', indicating that the block is exclusively read by one processor. Draw state transition diagrams for the MESI cache coherence protocol. Two diagrams are needed: first diagram should be based on requests from the processor and the second diagram should be based on requests from the bus.





4) (10 pts) The quad-core multiprocessor shown in Figure 2 uses directory-based cache coherence implemented in the shared L2 cache. Each processor has an L1 write-back private cache. Three states are defined in the L1 cache: M (Modified), S (Shared), and I (invalid). For simplicity, each L1 cache is directly-mapped with four blocks, and each block holds two words (8 bytes). For clarity, the tag contains the full address in hexadecimal, while the data is shown in decimal.

The L2 cache is directly-mapped, shared by all cores, and split into two banks. Blocks whose physical addresses are multiple of 16 (0x10 in hexadecimal) are mapped to L2 cache bank 0, while the other blocks are mapped to L2 cache bank 1. Four presence bits are associated with each L2 cache block, to indicate which L1 cache has a copy of the L2 cache block. The L2 cache is sufficiently large and contains a superset of the L1 blocks (inclusion policy). Four states are associated with each L2 cache block: M (Modified by a processor), O (Owned by L2 and can be shared in L1), S (Shared clean block), and I (Invalid). The L2 Owned state indicates that a block was modified in L1 and later written back to L2. The L2 cache is updated but not memory.

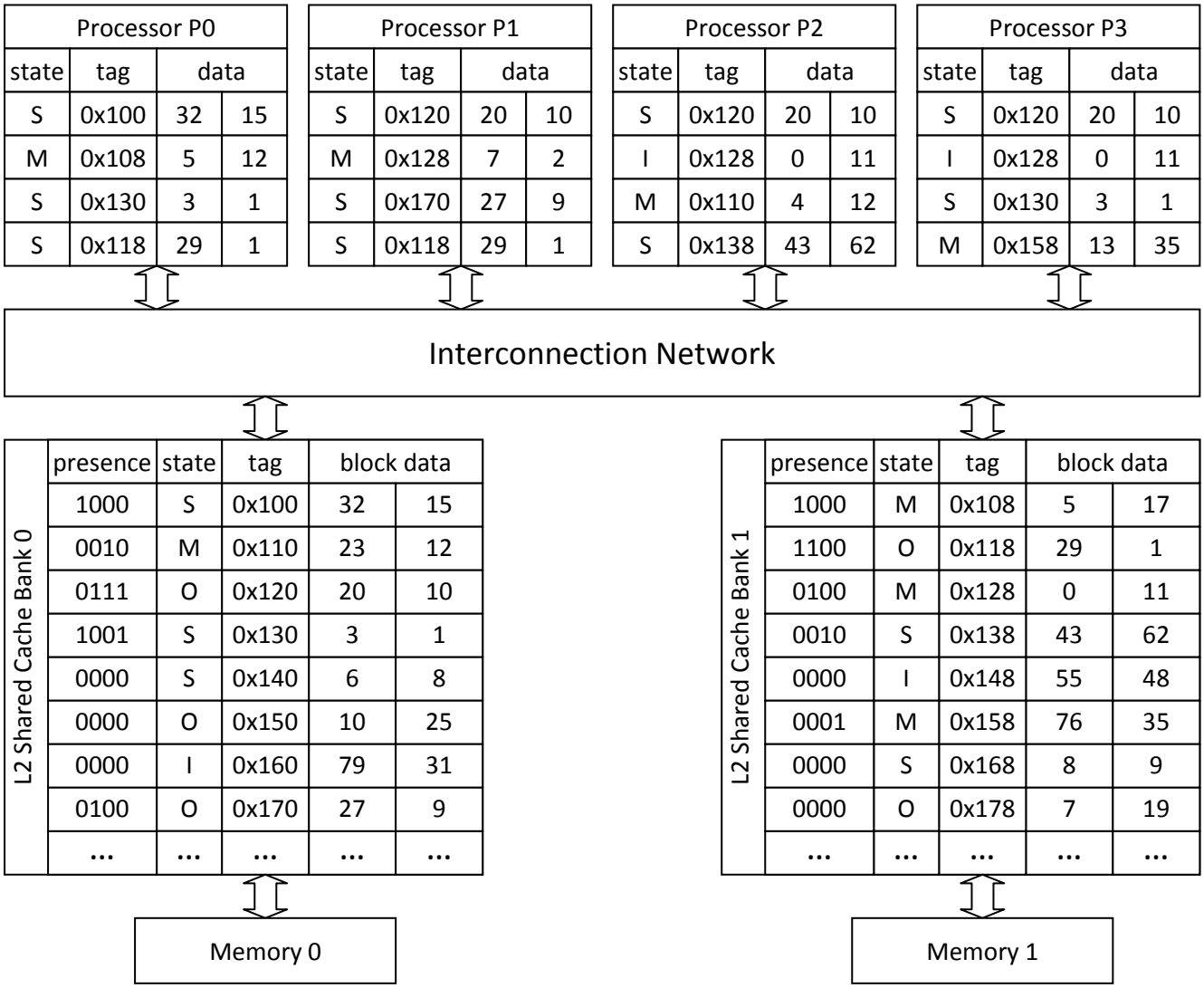


Figure 2: Directory-Based Quad-Core multiprocessor

Each part of this exercise specifies a memory operation. Treat each operation as applied to the initial state given in Figure 2. What value is returned (by a read) and the resulting state, tag, and value in L1 and L2 caches? Show the presence bits and content of the relevant cache blocks.

a) P0 reads address 0x120
b) P3 writes address 0x124 ← 38
c) P2 reads address 0x128
d) P1 reads address 0x158
e) P1 writes address 0x11C ← 14

Solution:

a) (2 pts) P0 reads address 0x120
Read miss, P0 replaces block 0 with: (S, 0x120, 20, 10) → return 20
Block is owned by shared cache, set presence bit of P0: (1111, O, 0x120, 20, 10)

b) (2 pts) P3 writes address 0x124 ← 38
P3 sends invalidate message to invalidate all shared copies of block 0x120
P1 and P2 invalidate block 0: (I, 0x120, 20, 10)
Shared cache (directory) changes presence bits and state: (0001, M, 0x120, 20, 10)
P3 writes block 0 and changes its state: (M, 0x120, 20, 38)

c) (2 pts) P2 reads address 0x128
P2 sends Read miss message to shared cache (directory), which sends fetch message to P1
P1 writes-back block 1 and change its state to shared: (S, 0x128, 7, 2)
Shared cache is the owner of block 0x128 and sets presence bit of P2: (0110, O, 0x128, 7, 2)
P2 receives block 0x128 in shared state: (S, 0x128, 7, 2) → returns 7

d) (2 pts) P1 reads address 0x158
P1 sends read miss message to shared cache (directory), which sends fetch message to P3
P3 writes-back block 3 and change its state to shared: (S, 0x158, 13, 35)
Shared cache is the owner of block 0x158 and sets presence bit of P1: (0101, O, 0x158, 13, 35)
P1 receives block 0x158 in shared state: (S, 0x158, 13, 35) → returns 13

e) (2 pts) P1 writes address 0x11C ← 14
P1 sends invalidate message to invalidate all shared copies of block 0x118
P0 invalidates block 3: (I, 0x118, 29, 1)
Shared cache (directory) changes presence bits and state: (0100, M, 0x118, 29, 1)
P1 writes block 3 and changes its state: (M, 0x118, 29, 14)

5) (10 pts) Directory protocols are more scalable than snooping protocols because they send explicit request and invalidate messages to those nodes that have copies of a block, while snooping protocols broadcasts all requests to all nodes. Consider again the directory-based quad-core multiprocessor shown in Figure 2. Show all the messages that are sent in the network for each of the following requests:

a) P0 reads address 0x120
b) P3 writes address 0x124 ← 38
c) P2 reads address 0x128
d) P1 writes address 0x11C ← 14

Solution:

a) (2 pts) P0 reads address 0x120
Message1: (from P0, to shared bank 0, address 0x120, Read miss)
Message2: (from shared bank 0, to P0, address 0x120, Data reply = 20, 10)

b) (3 pts) P3 writes address 0x124 ← 38
Message1: (from P3, to shared bank 0, address 0x120, Invalidate)
Message2: (from shared bank 0, to P1 (then P2), address 0x120, Invalidate)
Message3: (from P1, to P2, address 0x120, Invalidate)
Message4: (from P2, to shared bank 0, address 0x120, Ack-Invalidate)
Message5: (from shared bank 0, to P3, address 0x120, Ack-Invalidate)

c) (2.5 pts) P2 reads address 0x128
Message1: (from P2, to shared bank 1, address 0x128, Read miss)
Message2: (from shared bank 1, to P1, address 0x128, Fetch)
Message3: (from P1, to shared bank 1, address 0x128, Write back data = 7, 2)
Message4: (from shared bank 1, to P2, address 0x128, Data reply = 7, 2)

d) (2.5 pts) P1 reads address 0x158
Message1: (from P1, to shared bank 1, address 0x158, Read miss)
Message2: (from shared bank 1, to P3, address 0x158, Fetch)
Message3: (from P3, to shared bank 1, address 0x158, Write back data = 13, 35)
Message4: (from shared bank 1, to P1, address 0x158, Data reply = 13, 35)