Computer Architecture, Problem Set #5A

Note: We have procured permission to post these problems from Computer Architecture: A Quantitative Approach from the publisher.

Problem #1 (10 Points): Page 254 in H&P5, Problem 3.13

For problem #1: Assume that 3 stall cycles after load means load has a latency of 4.

3.13 [25] <3.13> In this exercise, you will explore performance trade-offs between three processors that each employ different types of multithreading. Each of these processors is superscalar, uses in-order pipelines, requires a fixed three-cycle stall following all loads and branches, and has identical L1 caches. Instructions from the same thread issued in the same cycle are read in program order and must not contain any data or control dependences.

- Processor A is a superscalar SMT architecture, capable of issuing up to two instructions per cycle from two threads.
- Processor B is a fine MT architecture, capable of issuing up to four instructions per cycle from a single thread and switches threads on any pipeline stall.
- Processor C is a coarse MT architecture, capable of issuing up to eight instructions per cycle from a single thread and switches threads on an L1 cache miss.

Our application is a list searcher, which scans a region of memory for a specific value stored in R9 between the address range specified in R16 and R17. It is parallelized by evenly dividing the search space into four equal-sized contiguous blocks and assigning one search thread to each block (yielding four threads). Most of each thread's runtime is spent in the following unrolled loop body:

(Continued on next page)

Copyright © 2011, Elsevier Inc. All rights Reserved

```
loop:  LD R1,0(R16)
       LD R2,8(R16)
       LD R3,16(R16)
       LD R4,24(R16)
       LD R5,32(R16)
       LD R6,40(R16)
       LD R7,48(R16)
       LD R8,56(R16)
       BEQAL R9,R1,match0
       BEQAL R9,R2,match1
       BEQAL R9,R3,match2
       BEQAL R9,R4,match3
       BEQAL R9,R5,match4
       BEQAL R9,R6,match5
       BEQAL R9,R7,match6
       BEQAL R9,R8,match7
       DADDIU R16,R16,#64
       BLT R16,R17,loop
```

Assume the following:

- A barrier is used to ensure that all threads begin simultaneously.
- The first L1 cache miss occurs after two iterations of the loop.
- None of the BEQAL branches is taken.
- The BLT is always taken.
- All three processors schedule threads in a round-robin fashion.

Determine how many cycles are required for each processor to complete the first two iterations of the loop.

Copyright © 2011, Elsevier Inc. All rights Reserved

Problem # 10 (10 Points): Page 420 in H&P5, Problem 5.11

d. [10] <5.4>

5.11 [25] <5.4> Exercise 5.3 asked you to add the Owned state to the simple MSI snooping protocol. Repeat the question, but with the simple directory protocol above.

Case Study 2: Simple Directory-Based Coherence

Concepts illustrated by this case study

- Directory Coherence Protocol Transitions
- Coherence Protocol Performance
- Coherence Protocol Optimizations

Consider the distributed shared-memory system illustrated in Figure 5.37. It consists of two four-core chips. The processor in each chip share an L2 cache (L2$), and the two chips are connected via a point-to-point interconnect. The system memory is distributed across the two chips. Figure 5.38 zooms in on part of this system. `Pi,j` denotes processor *i* in chip *j*. Each processor has a single direct-mapped L1 cache that holds two blocks, each holding two words. Each chip has a single direct-mapped L2 cache that holds two blocks, each holding two words. To simplify the illustration, the cache address tags contain the full address and each word shows only two hex characters, with the least significant word on the right. The L1 cache states are denoted M, S, and I for Modified, Shared, and Invalid. Both the L2 caches and memories have directories. The directory states are denoted DM, DS, and DI for Directory Modified, Directory Shared, and Directory Invalid. The simple directory protocol is described in Figures 5.22 and 5.23. The L2 directory lists the local sharers/owners and additionally records if a line is shared externally in another chip; for example, `P1,0;E` denotes that a line is shared by local processor `P1,0` and is externally shared in some other chip. The memory directory has a list of the chip sharers/owners of a line; for example, `C0,C1` denotes that a line is shared in chips 0 and 1.

Copyright © 2011, Elsevier Inc. All rights Reserved

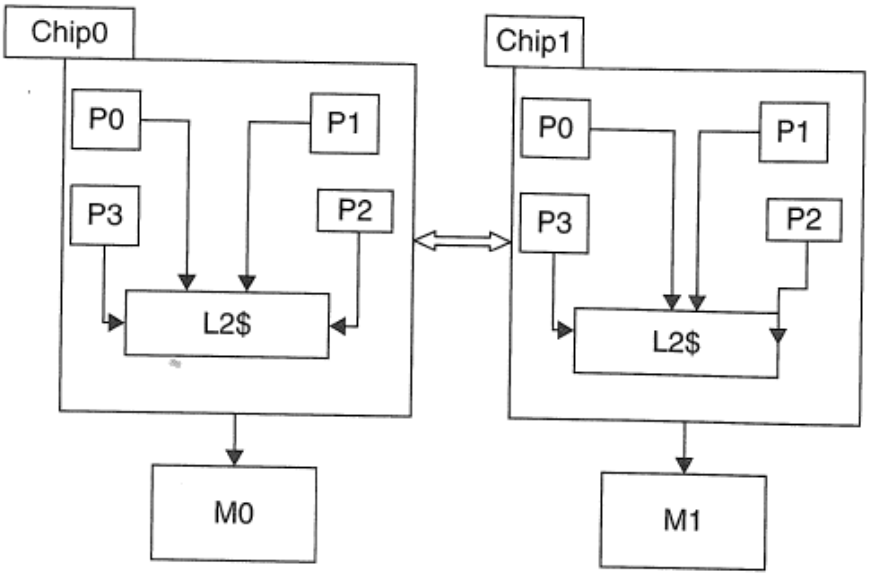
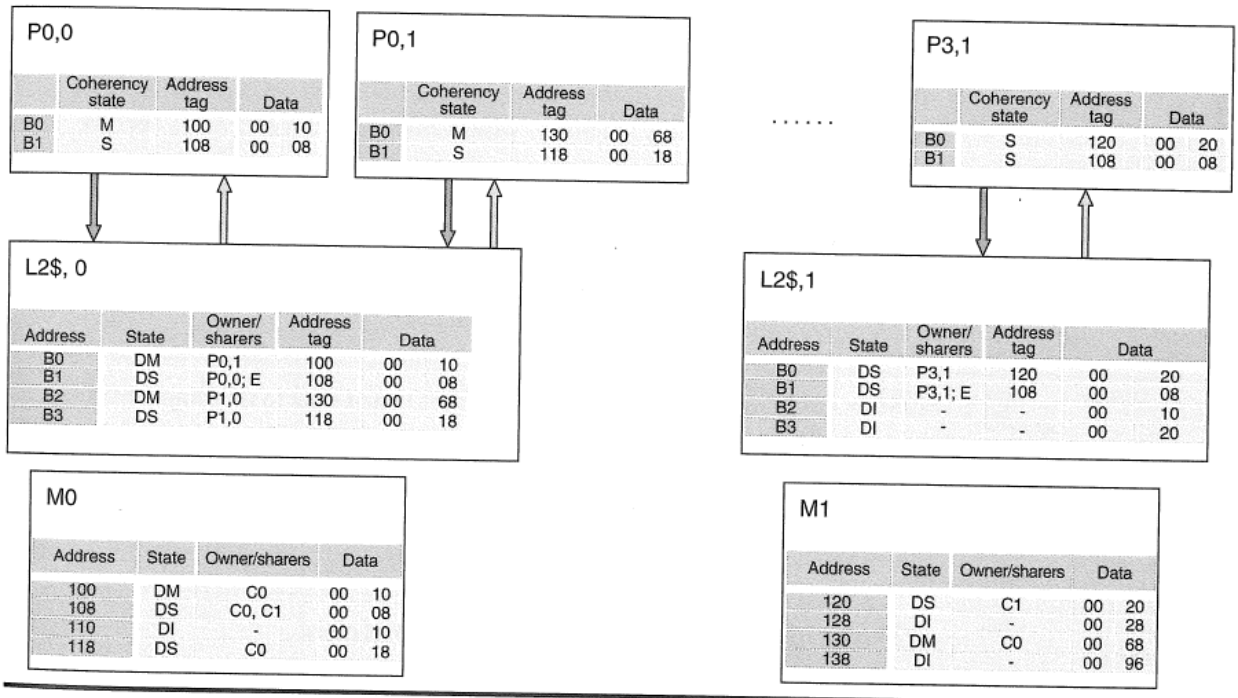


Figure 5.37 Multichip, multicore multiprocessor with DSM.



P0,0

	Coherency state	Address tag	Data	
B0	M	100	00	10
B1	S	108	00	08

P0,1

	Coherency state	Address tag	Data	
B0	M	130	00	68
B1	S	118	00	18

P3,1

	Coherency state	Address tag	Data	
B0	S	120	00	20
B1	S	108	00	08

L2$, 0

Address	State	Owner/ sharers	Address tag	Data	
B0	DM	P0,1	100	00	10
B1	DS	P0,0; E	108	00	08
B2	DM	P1,0	130	00	68
B3	DS	P1,0	118	00	18

L2$,1

Address	State	Owner/ sharers	Address tag	Data	
B0	DS	P3,1	120	00	20
B1	DS	P3,1; E	108	00	08
B2	DI	-	-	00	10
B3	DI	-	-	00	20

M0

Address	State	Owner/sharers	Data	
100	DM	C0	00	10
108	DS	C0, C1	00	08
110	DI	-	00	10
118	DS	C0	00	18

M1

Address	State	Owner/sharers	Data	
120	DS	C1	00	20
128	DI	-	00	28
130	DM	C0	00	68
138	DI	-	00	96

Figure 5.38 Cache and memory states in the multichip, multicore multiprocessor.

Copyright © 2011, Elsevier Inc. All rights Reserved

5.3 [20] <5.2> Many snooping coherence protocols have additional states, state transitions, or bus transactions to reduce the overhead of maintaining cache coherency. In Implementation 1 of Exercise 5.2, misses are incurring fewer stall cycles when they are supplied by cache than when they are supplied by memory. Some coherence protocols try to improve performance by increasing the frequency of this case. A common protocol optimization is to introduce an Owned state (usually denoted O). The Owned state behaves like the Shared state in that nodes may only read Owned blocks, but it behaves like the Modified state in that nodes must supply data on other nodes' read and write misses to Owned blocks. A read miss to a block in either the Modified or Owned states supplies data to the requesting node and transitions to the Owned state. A write miss to a block in either state Modified or Owned supplies data to the requesting node and transitions to state Invalid. This optimized MOSI protocol only updates memory when a node replaces a block in state Modified or Owned. Draw new protocol diagrams with the additional state and transitions.

Copyright © 2011, Elsevier Inc. All rights Reserved