16.482 / 16.561: Computer Architecture and Design

Fall 2013

Lecture 11: Key Questions
December 2, 2013

1. Explain the benefits of using small arrays of disks, rather than a single large disk.

2. What impact do disk arrays have on reliability?

3. Explain each of the following levels of RAID:
a. RAID 1

b. RAID 3

c. RAID 4

d. RAID 5 (& 6)

1. What are the four classes of processors according to Flynn's Taxonomy?

2. What are the two classes of multiprocessors based on memory organization?

3. What are the two classes of multiprocessors based on communication mechanisms?

4. Define coherence and consistency.

5. What are the two classes of coherence protocol?

6. What are the two ways of handling writes in a coherence protocol?

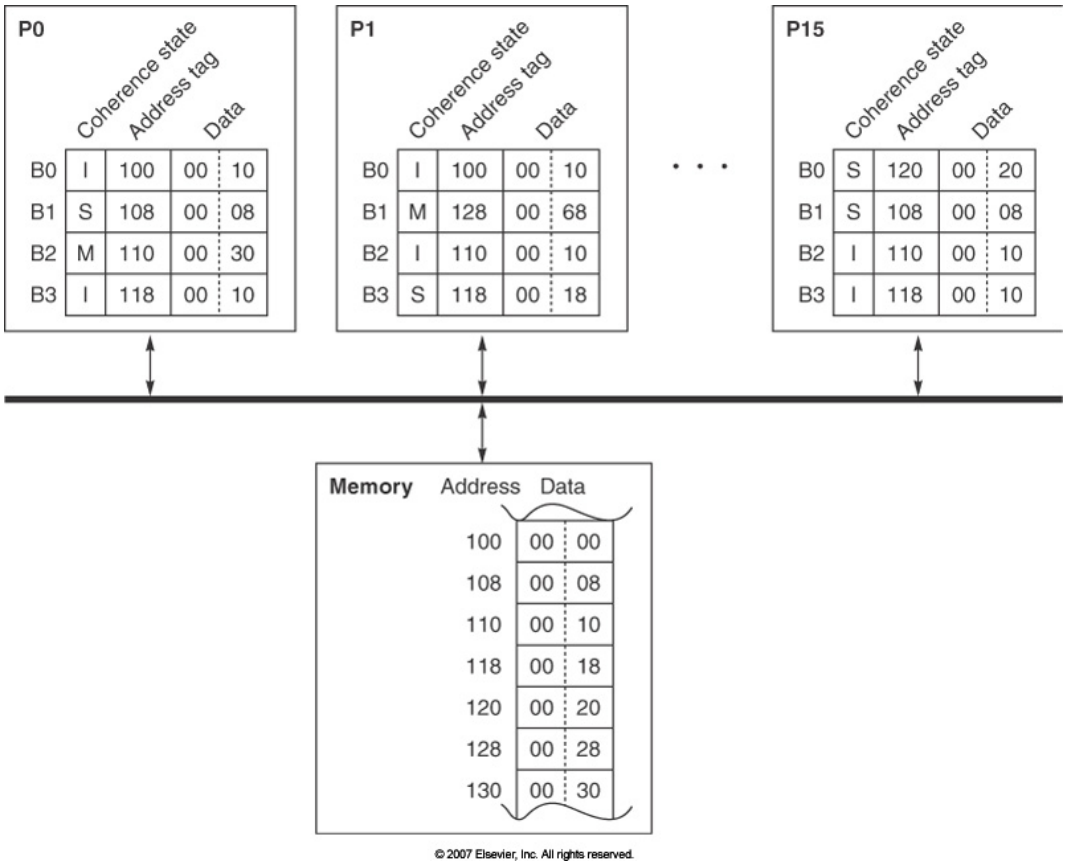
7. What do we need to implement a coherence protocol?

8. Draw the state transition diagram for a cache block based on requests from the CPU.

9. Draw the state transition diagram for a cache block based on requests seen on the bus.

10. <u>Snooping protocol example</u>

Assume you are given a system with the following initial cache and memory states:



Each cache is direct-mapped, and each block contains two 32-bit words of memory—only the least significant byte of each word is shown. Addresses are written in hexadecimal. In this figure, the address tag shows the full address for simplicity. The coherence states are M (Modified—both exclusive and dirty), S (Shared), and I (Invalid). The system uses write-back caches and a write-invalidate protocol.

Each of the operations below is independently performed on the system using the initial state above—in other words, changes do not carry over from one part to the next, so your answer to part (b), for example, should not be affected by your answer to part (a). Each step specifies a CPU operation as follows:

```
P#: <read/write> <address> [ ← <value>]
```

Where `P#` is a processor number, `<read/write>` tells you which operation is performed, `<address>` gives the address of the word being accessed, and the optional `<value>` field lists a value to be written to that address. For each part, answer the following questions:

- What is the resulting state—coherence state, tags, and data—for the caches and memory after each given action? Show only the blocks that change, using the following format:

  ```
  P#.B#: (<state>, <tag>, <data>)
  ```

  So, for example: `P0.B0: (I, 138, 00 01)` indicates that block `B0` on CPU `P0` is invalid, contains a tag of `138`, and data words `00` and `01`. If an access changes memory, list the state of the changed memory block using the format:

  ```
  M[<address>]: <data>
  ```

  Remember, a single access may affect multiple blocks, depending on their state.

- If the operation is a read, what value is returned?

a. `P1: read 108`

b. `P0: write 108 ← 83`

c. `P1: write 108 ← 83`

d. `P15: read 118`

e. `P0: write 110 ← 52`

f. `P1: write 118 ← 99`

g. `P0: write 118 ← 99`

11. Define coherence misses, and explain the two types of coherence misses

12. What are the benefits of directory-based coherence over snooping coherence?

13. Explain the basic operation of a directory protocol.

14. Draw the state machine for requests sent to the directory for a given block.

15. Directory protocol example

Say we have a dual-processor system that uses a write-invalidate, directory coherence protocol. The system contains 4 memory blocks, as shown in the directory state below:

Block #	P0	P1	Dirty
0	1	0	1
1	0	0	0
2	1	1	0
3	0	1	0
4	1	1	0

What messages are sent between the nodes and directory for each of the following transactions to ensure the processor has the most up-to-date block copy, the appropriate invalidations are made, and the directory holds the appropriate state? Assume the same initial state for every part of the problem—your answer to (b) does not depend on (a).

a. P0: read block 1

b. P1: write block 2

c. P1: read block 0

d. P0: read block 3