STAM Center — SECURE, TRUSTED, AND ASSURED MICROELECTRONICS
ASU Ira A. Fulton Schools of Engineering — Arizona State University

# CSE 520
# Computer Architecture II

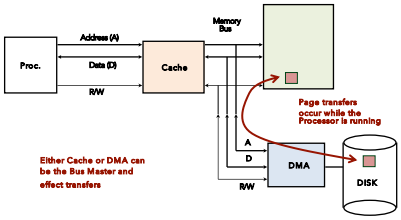
## Cache Coherence Protocols

Prof. Michel A. Kinsy

1

STAM Center — SECURE, TRUSTED, AND ASSURED MICROELECTRONICS
ASU Ira A. Fulton Schools of Engineering — Arizona State University

## Warmup: Parallel I/O

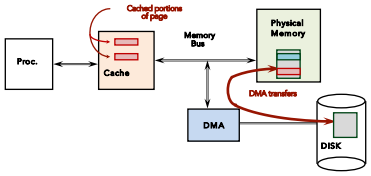



- DMA stands for Direct Memory Access

2

STAM Center — SECURE, TRUSTED, AND ASSURED MICROELECTRONICS
ASU Ira A. Fulton Schools of Engineering — Arizona State University

## Problems with Parallel I/O

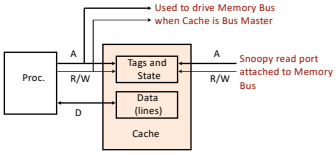



- Memory → Disk: Physical memory may be stale if cache copy is dirty
- Disk → Memory: Cache may have data corresponding to the memory

3

## Snoopy Cache *Goodman 1983*

- Idea: have the cache watch (or snoop upon) DMA transfers, and then "do the right thing"
- Snoopy cache tags are dual-ported



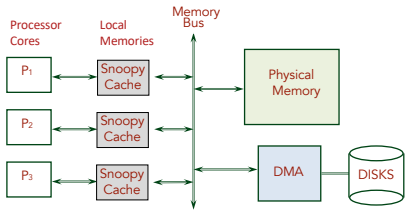

4

## Snoopy Cache Actions

| Observed Bus Cycle | Cache State | Cache Action |
|---|---|---|
| Read Cycle, i.e., Memory → Disk | Address not cached | No action |
| | Cached, unmodified | No action |
| | Cached, modified | Cache intervenes |
| Write Cycle, i.e., Disk → Memory | Address not cached | No action |
| | Cached, unmodified | Cache purges its copy |
| | Cached, modified | ??? |

5

## Shared Memory Multiprocessor



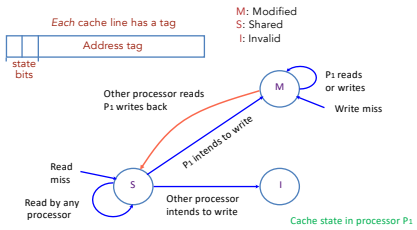

- Use snoopy mechanism to keep all processors' view of memory coherent

6

STAM Center — SECURE, TRUSTED, AND ASSURED MICROELECTRONICS
ASU Ira A. Fulton Schools of Engineering — Arizona State University

## Cache State Transition Diagram

- The MSI protocol

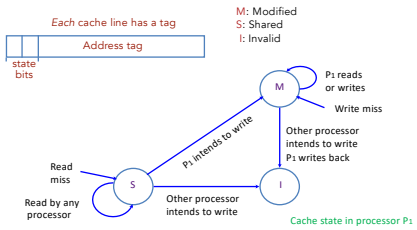



7

STAM Center — SECURE, TRUSTED, AND ASSURED MICROELECTRONICS
ASU Ira A. Fulton Schools of Engineering — Arizona State University

## Cache State Transition Diagram

- The MSI protocol

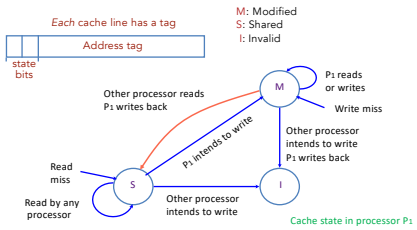



8

STAM Center — SECURE, TRUSTED, AND ASSURED MICROELECTRONICS
ASU Ira A. Fulton Schools of Engineering — Arizona State University

## Cache State Transition Diagram

- The MSI protocol

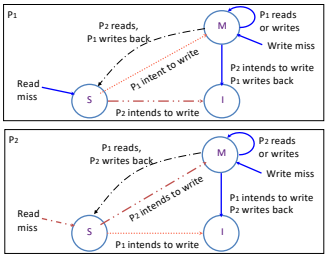



9

STAM Center — SECURE, TRUSTED, AND ASSURED MICROELECTRONICS
ASU Ira A. Fulton Schools of Engineering — Arizona State University

## 2 Processor Example

P1 reads
P1 writes
P2 reads
P2 writes
P1 reads
P1 writes
P2 writes
P1 writes

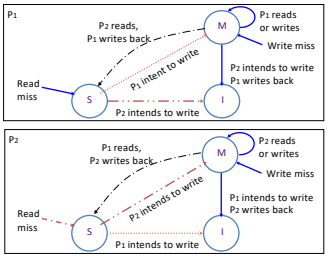



10

STAM Center — SECURE, TRUSTED, AND ASSURED MICROELECTRONICS
ASU Ira A. Fulton Schools of Engineering — Arizona State University

## 2 Processor Example

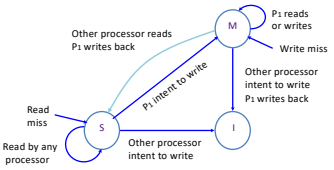
P1 reads
P1 writes
P2 reads
P2 writes
P1 reads
P1 writes
P2 writes
P1 writes




11

STAM Center — SECURE, TRUSTED, AND ASSURED MICROELECTRONICS
ASU Ira A. Fulton Schools of Engineering — Arizona State University

## Observation

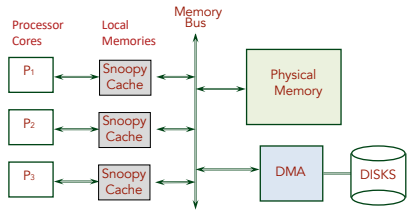



- If a line is in the M state then no other cache can have a copy of the line!
  - Memory stays coherent, multiple differing copies cannot exist

12

STAM Center — SECURE, TRUSTED, AND ASSURED MICROELECTRONICS — ASU Ira A. Fulton Schools of Engineering, Arizona State University

## Shared Memory Multiprocessor



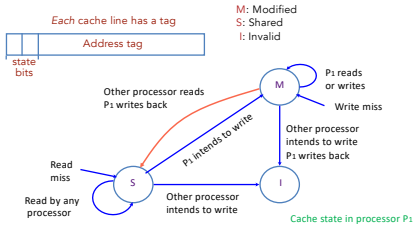

- Use snoopy mechanism to keep all processors' view of memory coherent

13

STAM Center — SECURE, TRUSTED, AND ASSURED MICROELECTRONICS — ASU Ira A. Fulton Schools of Engineering, Arizona State University

## Cache State Transition Diagram

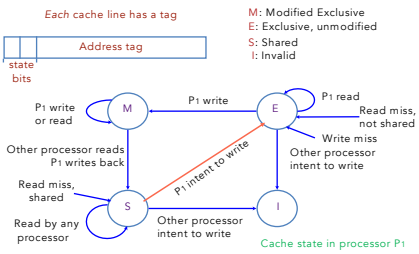
- The MSI protocol




14

STAM Center — SECURE, TRUSTED, AND ASSURED MICROELECTRONICS — ASU Ira A. Fulton Schools of Engineering, Arizona State University

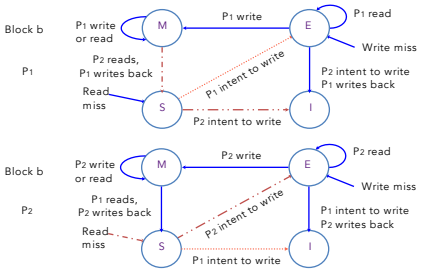
## MESI: An Enhanced MSI protocol




15

STAM Center — SECURE, TRUSTED, AND ASSURED MICROELECTRONICS — ASU Ira A. Fulton Schools of Engineering, Arizona State University

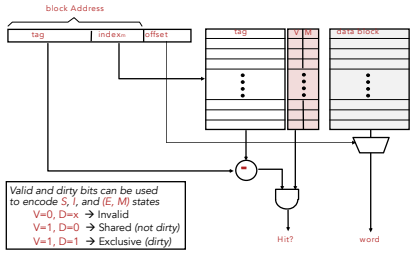
## 2 Processor Example




16

STAM Center — SECURE, TRUSTED, AND ASSURED MICROELECTRONICS — ASU Ira A. Fulton Schools of Engineering, Arizona State University

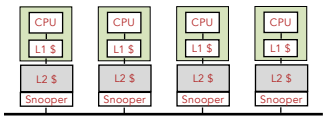
## Cache Coherence State Encoding




17

STAM Center — SECURE, TRUSTED, AND ASSURED MICROELECTRONICS — ASU Ira A. Fulton Schools of Engineering, Arizona State University

## 2-Level On-chip Caches

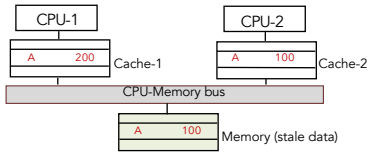



- Inclusion property: entries in L1 must be in L2
  - Invalidation in L2 → invalidation in L1
- Does snooping on L2 affect CPU-L1 bandwidth?
  - Yes -- to check if a dirty copy is stored in L1
- How can this be avoided?
  - Write-through L1 cache

18

STAM Center — SECURE, TRUSTED, AND ASSURED MICROELECTRONICS — ASU Ira A. Fulton Schools of Engineering, Arizona State University

## Intervention




- When a read-miss for A occurs in cache-2, a read request for A is placed on the bus
  - Cache-1 needs to supply & change its state to shared
  - The memory may respond to the request also!

19

STAM Center — SECURE, TRUSTED, AND ASSURED MICROELECTRONICS — ASU Ira A. Fulton Schools of Engineering, Arizona State University

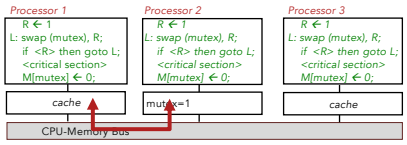
## False Sharing


- A cache block contains more than one word
- Cache-coherence is done at the block-level and not word-level
- Suppose $M_1$ writes $word_i$ and $M_2$ writes $word_k$ and both words have the same block address.
- What can happen?
  - Block may be invalidated many times unnecessarily

20

STAM Center — SECURE, TRUSTED, AND ASSURED MICROELECTRONICS — ASU Ira A. Fulton Schools of Engineering, Arizona State University

## Synchronization and Caches




- Performance Issues
  - Cache-coherence protocols will cause mutex to ping-pong between P1's and P2's caches.
  - Ping-ponging can be reduced by first reading the mutex location (non-atomically) and executing a swap only if it is found to be zero.

21

STAM Center — SECURE, TRUSTED, AND ASSURED MICROELECTRONICS
ASU Ira A. Fulton Schools of Engineering — Arizona State University

## Performance Related to Bus Occupancy

- In general, a read-modify-write instruction requires two memory (bus) operations without intervening memory operations by other processors
- In a multiprocessor setting, bus needs to be locked for the entire duration of the atomic read and write operation
  - Expensive for simple buses
  - Very expensive for split-transaction buses
- Modern processors use
  - Load-reserve and store-conditional

22

STAM Center — SECURE, TRUSTED, AND ASSURED MICROELECTRONICS
ASU Ira A. Fulton Schools of Engineering — Arizona State University

## Load-reserve & Store-conditional

- Special register(s) to hold reservation flag and address, and the outcome of store-conditional

```
Load-reserve R, (a):
        <flag, adr> ← <1, a>;
        R ← M[a];
```

```
Store-conditional (a), R:
        if <flag, adr> == <1, a>
        then cancel other procs' reservation on a;
                M[a] ← <R>;
                status ← succeed;
        else status ← fail;
```

- If the snooper sees a store transaction to the address in the reserve register, the reserve bit is set to 0
  - Several processors may reserve 'a' simultaneously
  - These instructions are like ordinary loads and stores with respect to the bus traffic

23

STAM Center — SECURE, TRUSTED, AND ASSURED MICROELECTRONICS
ASU Ira A. Fulton Schools of Engineering — Arizona State University

## Performance

- Load-reserve & Store-conditional
  - The total number of memory (bus) transactions is not necessarily reduced, but splitting an atomic instruction into load-reserve & store-conditional:
    - Increases bus utilization (and reduces processor stall time), especially in split-transaction buses
    - Reduces cache ping-pong effect because processors trying to acquire a semaphore do not have to perform stores each time

24

STAM Center — SECURE, TRUSTED, AND ASSURED MICROELECTRONICS
ASU Ira A. Fulton Schools of Engineering — Arizona State University

## Maintaining Cache Coherence

- Hardware support is required such that
  - Only one processor at a time has write permission for a location
  - No processor can load a stale copy of the location after a write
- write request:
  - The address is invalidated in all other caches before the write is performed
- read request:
  - If a dirty copy is found in some cache, a write-back is performed before the memory is read

25

STAM Center — SECURE, TRUSTED, AND ASSURED MICROELECTRONICS
ASU Ira A. Fulton Schools of Engineering — Arizona State University

## Software Cache Coherence

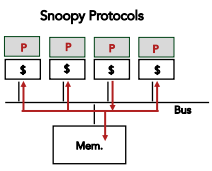
- Exclude hardware support for cache coherence, e.g., Cray T3D
- Systems with caches mark shared data as uncacheable
- Software can explicitly cache value of stored data
- Disadvantages?
  - Compiler mechanisms for CC very limited
  - Need to be conservative: every block that might be shared is treated as shared
  - Doing things at the cache block level more efficient

26

STAM Center — SECURE, TRUSTED, AND ASSURED MICROELECTRONICS
ASU Ira A. Fulton Schools of Engineering — Arizona State University

## Directory-Based Coherence

- By Censier and Feautrier, 1978
- Snoopy schemes broadcast requests over memory bus
  - Totally ordered interconnect
- Difficult to scale to large numbers of processors
- Requires additional bandwidth to cache tags for snoop requests

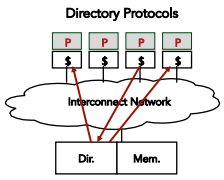



27

STAM Center — SECURE, TRUSTED, AND ASSURED MICROELECTRONICS
ASU Ira A. Fulton Schools of Engineering — Arizona State University

## Directory-Based Coherence

- By Censier and Feautrier, 1978



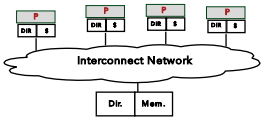

- Directory schemes send messages to only those caches that might have the line
- Can scale to large numbers of processors
- Requires extra directory storage to track possible sharers
- Directory may become bottleneck

28

STAM Center — SECURE, TRUSTED, AND ASSURED MICROELECTRONICS
ASU Ira A. Fulton Schools of Engineering — Arizona State University

## Directory Protocols

- Directory keeps the state of every block that is cached
  - Which caches have copies, which do not?
- Directory entries can be distributed, so different directory accesses go to different locations
  - Sharing status of a block always in single known location
  - Will not consider this in this class




29

STAM Center — SECURE, TRUSTED, AND ASSURED MICROELECTRONICS
ASU Ira A. Fulton Schools of Engineering — Arizona State University

## Directory Protocol Basics

- Handling a read miss
- Handling a write to a shared, clean block
- Handing a write miss

Directory state for address A

S — One or more processors have the block cached and value in memory is up to date

U — No processor has a copy of the cache block

M — Exactly one processor has a copy of the cache block and it has written the block

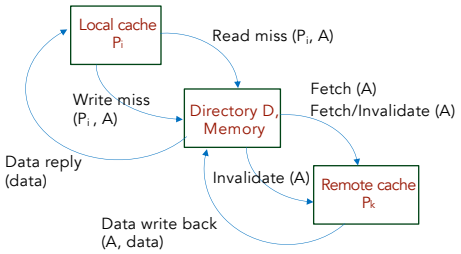
30

## Snoopy vs. Directory

- States and transitions on the cache side are similar but actions on transition are different
- Cannot use interconnect as a single point of arbitration in directory scheme
- Interconnect is message-oriented (rather than a transaction-oriented bus) and many messages have explicit responses

31

## Message Catalog




32

## Some Assumptions

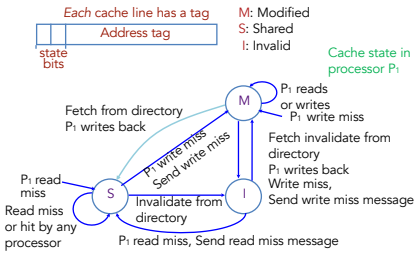
- Attempts to write data that is not exclusive in the cache always generate write misses
- Processors block until access completes
- Messages will be received and acted upon in the same order that they are sent
  - Invalidates sent by a processor are honored immediately

33

STAM Center — SECURE, TRUSTED, AND ASSURED MICROELECTRONICS

ASU Ira A. Fulton Schools of Engineering — Arizona State University

## Cache State Transition Diagram

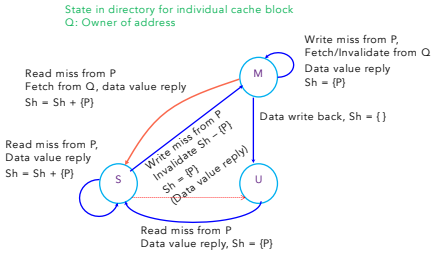



34

STAM Center — SECURE, TRUSTED, AND ASSURED MICROELECTRONICS

ASU Ira A. Fulton Schools of Engineering — Arizona State University

## Directory State Transition Diagram

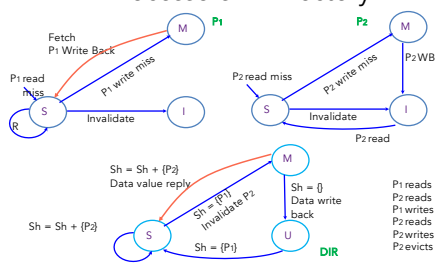



35

STAM Center — SECURE, TRUSTED, AND ASSURED MICROELECTRONICS

ASU Ira A. Fulton Schools of Engineering — Arizona State University

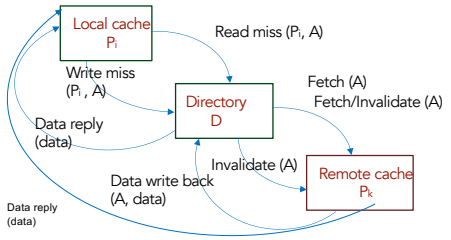
## 2 Processors + Directory




36

STAM Center — SECURE, TRUSTED, AND ASSURED MICROELECTRONICS
ASU Ira A. Fulton Schools of Engineering — Arizona State University

## Message Catalog Including Optimization




37

STAM Center — SECURE, TRUSTED, AND ASSURED MICROELECTRONICS
ASU Ira A. Fulton Schools of Engineering — Arizona State University

## Two Remaining Implementation Issues

- In snoopy protocol, we assumed bus transactions were atomic
  - Write miss cannot be atomic
  - This assumption is impossible to maintain in a general interconnection network that is message-oriented
- Assumed infinite buffers
  - Finite buffering to hold message requests and replies causes additional possibilities for deadlock

38

STAM Center — SECURE, TRUSTED, AND ASSURED MICROELECTRONICS
ASU Ira A. Fulton Schools of Engineering — Arizona State University

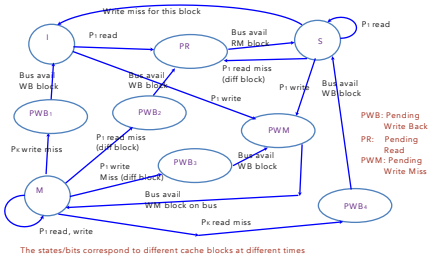
## Write Misses (in Snoopy Protocol)

- Two steps in a write miss
  1. Detect the miss and request the bus
  2. Acquire the bus, place the miss on the bus, get the data and complete the write
- Do not change the block to exclusive or allow the cache update to proceed before bus is acquired
  - Writes to the same cache block will serialize at second step assuming bus transactions are atomic once the bus is acquired (what does this imply?)
    - No split transactions
- 2-step write implies more complex protocol!

39

STAM Center — SECURE, TRUSTED, AND ASSURED MICROELECTRONICS
ASU Ira A. Fulton Schools of Engineering — Arizona State University

## "Real" Snoopy Cache Protocol




The states/bits correspond to different cache blocks at different times

40

STAM Center — SECURE, TRUSTED, AND ASSURED MICROELECTRONICS
ASU Ira A. Fulton Schools of Engineering — Arizona State University

## Finite Buffering

- Finite buffers could cause deadlock if there are no buffers to send replies in a directory protocol
  - P1 waiting for reply from directory on write miss
  - Directory waiting for data reply from P1 on P2 read of P1 modified data
- Don't initiate transaction unless resources are available
  - Separate network for requests and replies
  - Every request that expects a reply allocates space for the reply
  - Controller can reject a request but never a reply
  - Any request that is rejected is retried

41

STAM Center — SECURE, TRUSTED, AND ASSURED MICROELECTRONICS
ASU Ira A. Fulton Schools of Engineering — Arizona State University

## Next Learning Module

- On-chip Interconnect Network

42