电子科技大学
格拉斯哥学院
Glasgow College, UESTC

UESTC4019: Real-Time Computer Systems and Architecture

Lecture 19
Parallel Processing

Multiple Processor Organization (1 of 2)

- Single instruction, single data **(SISD)** stream
  - Single processor executes a single instruction stream to operate on data stored in a single memory
  - Uniprocessors fall into this category
- Single instruction, multiple data **(SIMD)** stream
  - A single machine instruction controls the simultaneous execution of a number of processing elements on a lockstep basis
  - Vector and array processors fall into this category

Multiple Processor Organization (2 of 2)

- Multiple instruction, single data **(MISD)** stream
  - A sequence of data is transmitted to a set of processors, each of which executes a different instruction sequence
  - Not commercially implemented
- Multiple instruction, multiple data **(MIMD)** stream
  - A set of processors simultaneously execute different instruction sequences on different data sets
  - SMPs, clusters and NUMA systems fit this category

A Taxonomy of Parallel Processor Architecture

With the MIMD organization, the processors are general purpose; each is able to process all of the instructions necessary to perform the appropriate data transformation.

MIMDs can be further subdivided by the means in which the processors communicate

If the processors share a common memory, then each processor accesses programs and data stored in the shared memory, and processors communicate with each other via that memory. The most common form of such system is known as a symmetric multiprocessor (SMP)

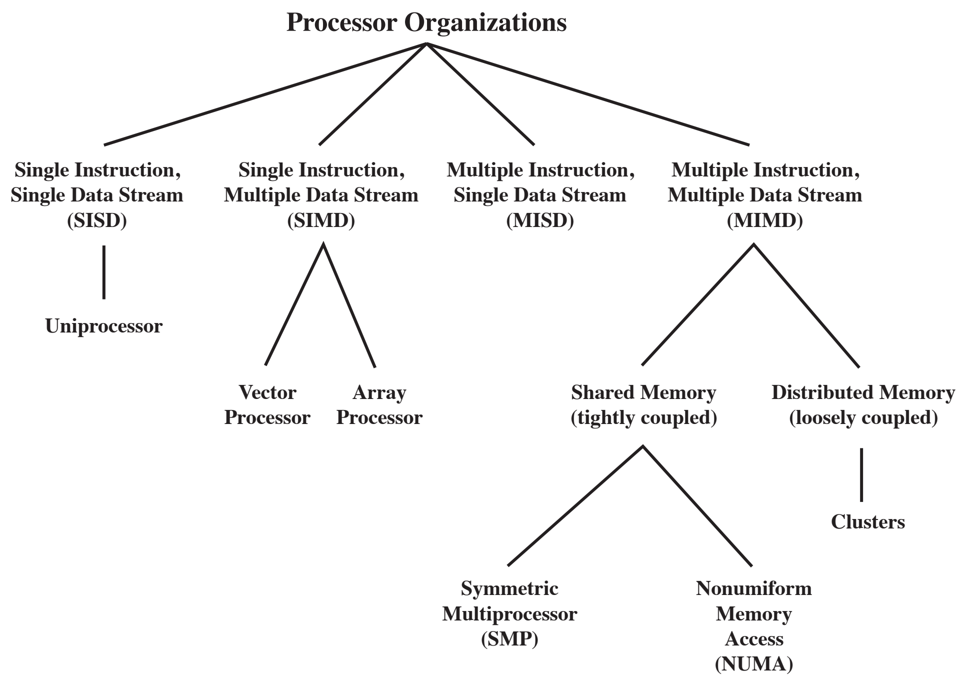
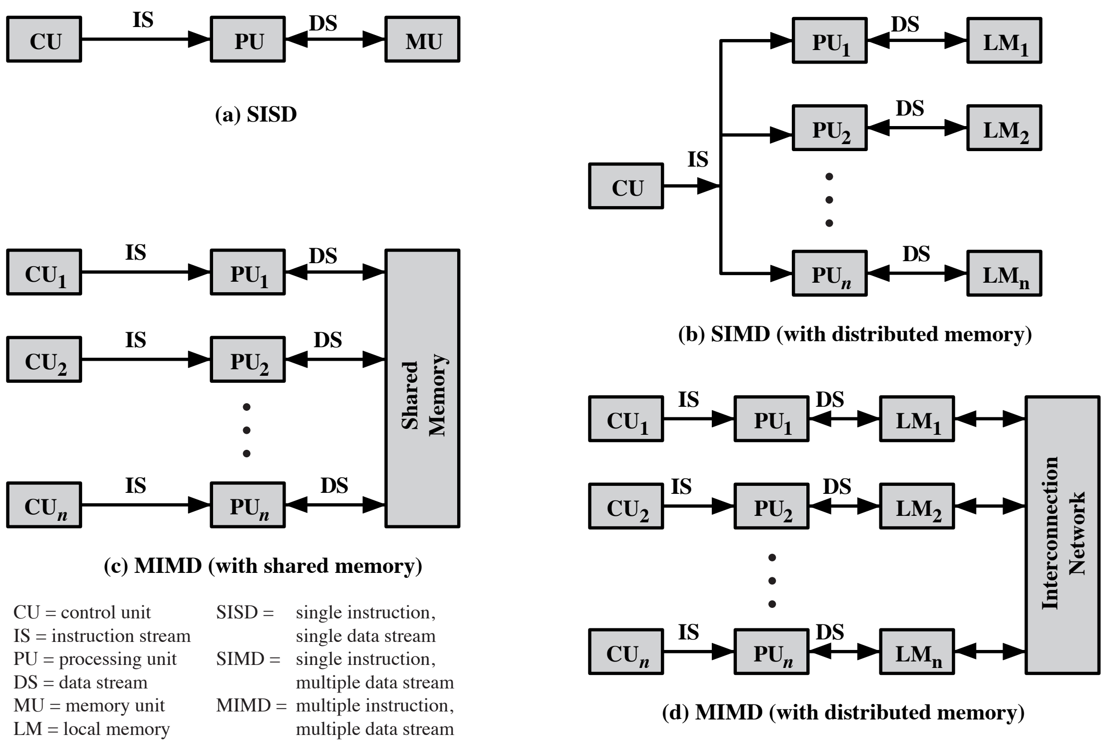

Processor Organizations
Single Instruction, Single Data Stream (SISD)
Single Instruction, Multiple Data Stream (SIMD)
Multiple Instruction, Single Data Stream (MISD)
Multiple Instruction, Multiple Data Stream (MIMD)
Uniprocessor
Vector Processor
Array Processor
Shared Memory (tightly coupled)
Distributed Memory (loosely coupled)
Clusters
Symmetric Multiprocessor (SMP)
Nonumiform Memory Access (NUMA)

Figure 17.1 A Taxonomy of Parallel Processor Architectures

Alternative Computer Organizations



In **SISD,** there is some sort of control unit (CU) that provides an instruction stream (IS) to a processing unit (PU). The processing unit operates on a single data stream (DS) from a memory unit (MU). With an **SIMD**, there is still a single control unit, now feeding a single instruction stream to multiple PUs. Each PU may have its own dedicated memory, or there may be a shared memory. Finally, with the **MIMD**, there are multiple control units, each feeding a separate instruction stream to its own PU. The MIMD may be a shared-memory multiprocessor or a distributed-memory multicomputer.

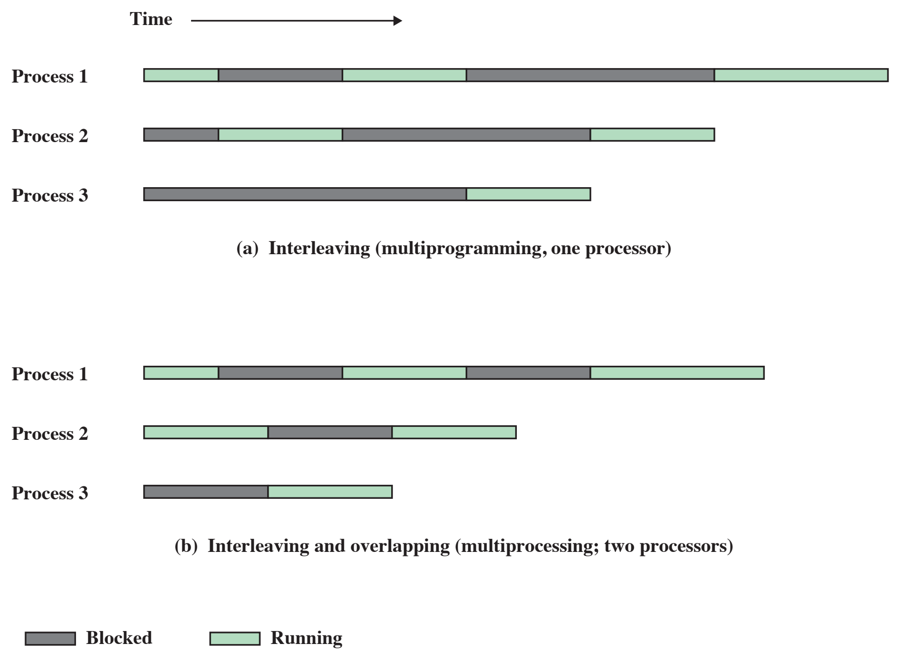
Symmetric Multiprocessor (SMP) (1 of 2)

- A stand alone computer with the following characteristics:
- Two or more similar processors of comparable capacity
- Processors share same memory and I/O facilities
  - Processors are connected by a bus or other internal connection
  - Memory access time is approximately the same for each processor
- All processors share access to I/O devices
  - Either through same channels or different channels giving paths to same devices

Symmetric Multiprocessor (SMP) (2 of 2)

- All processors can perform the same functions (hence “symmetric”)
- System controlled by integrated operating system
  - Provides interaction between processors and their programs at job, task, file and data element levels

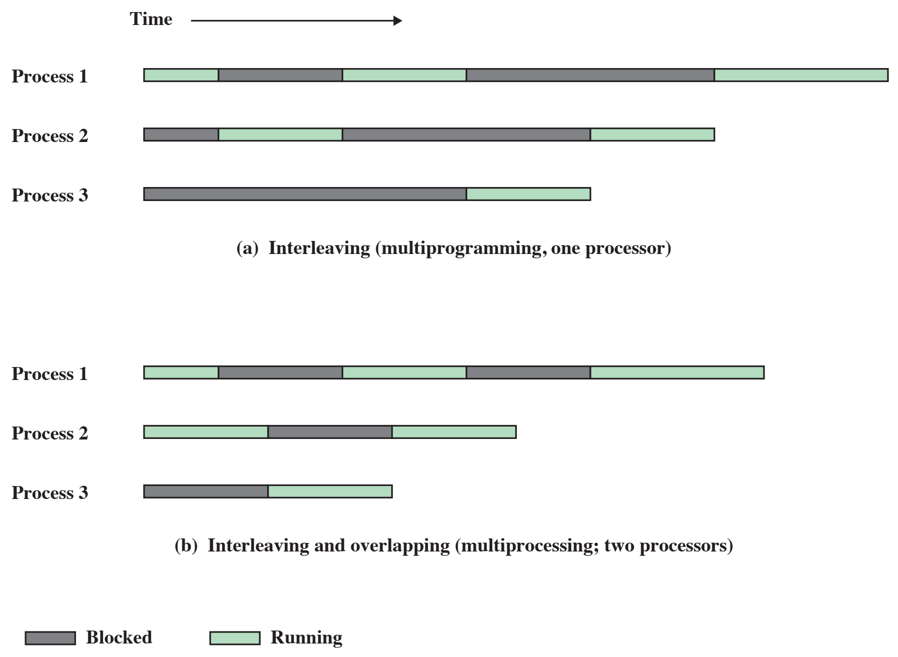
Multiprogramming and Multiprocessing



The operating system of an SMP schedules processes or threads across all of the processors. An SMP organization has a number of potential advantages over a uniprocessor organization, including the following:

Performance: If the work to be done by a computer can be organized so that some portions of the work can be done in parallel, then a system with multiple processors will yield greater performance than one with a single processor of the same type.

Multiprogramming and Multiprocessing


Time
Process 1
Process 2
Process 3
(a) Interleaving (multiprogramming, one processor)
Process 1
Process 2
Process 3
(b) Interleaving and overlapping (multiprocessing; two processors)
Blocked
Running

Availability: In a symmetric multiprocessor, because all processors can perform the same functions, the failure of a single processor does not halt the machine. Instead, the system can continue to function at reduced performance.

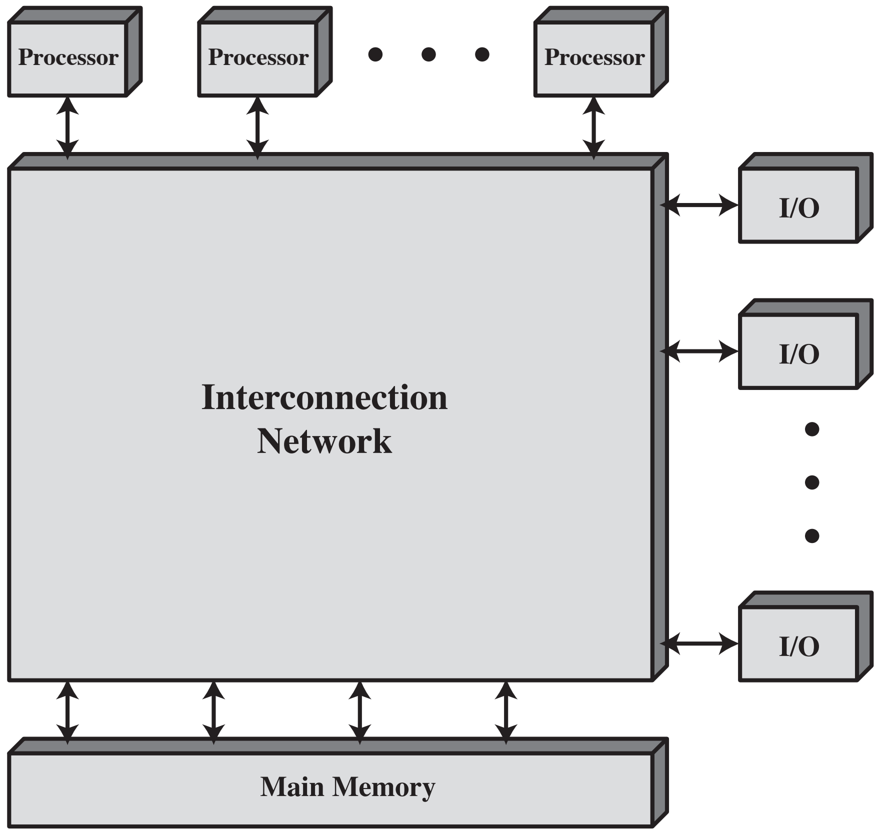
Incremental growth: A user can enhance the performance of a system by adding an additional processor.

Scaling: Vendors can offer a range of products with different price and performance characteristics based on the number of processors configured in the system.

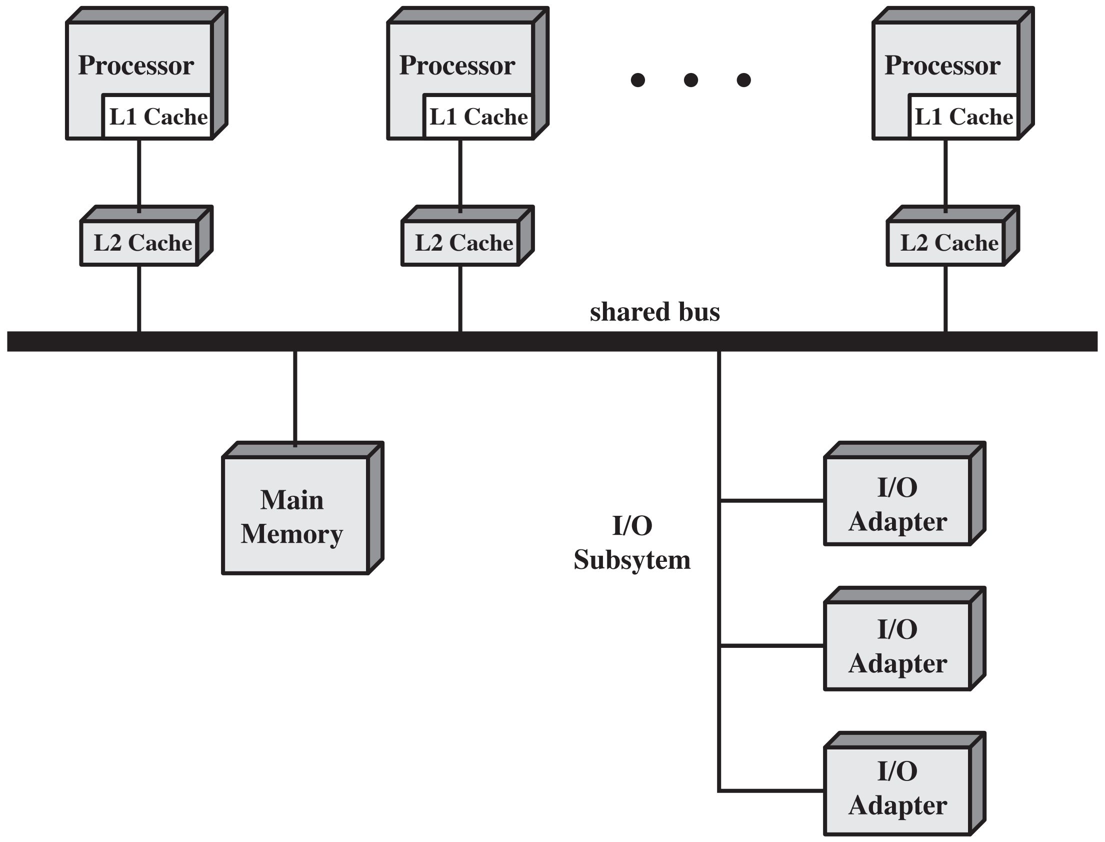
An attractive feature of an SMP is that the existence of multiple processors is transparent to the user. The operating system takes care of scheduling of threads or processes on individual processors and of synchronization among processors.

Generic Block Diagram of a Tightly Coupled Multiprocessor

There are two or more processors. Each processor is self-contained, including a control unit, ALU, registers, and, typically, one or more levels of cache. Each processor has access to a shared main memory and the I/O devices through some form of interconnection mechanism. The processors can communicate with each other through memory (messages and status information left in common data areas). It may also be possible for processors to exchange signals directly. The memory is often organized so that multiple simultaneous accesses to separate blocks of memory are possible. In some configurations, each processor may also have its own private main memory and I/O channels in addition to the shared resources.



Symmetric Multiprocessor Organization



The most common organization for personal computers, workstations, and servers is the time-shared bus. The time-shared bus is the simplest mechanism for constructing a multiprocessor system. The structure and interfaces are basically the same as for a single-processor system that uses a bus interconnection. The bus consists of control, address, and data lines.

The Bus Organization Has Several Attractive Features

- Simplicity
  - Simplest approach to multiprocessor organization
- Flexibility
  - Generally easy to expand the system by attaching more processors to the bus
- Reliability
  - The bus is essentially a passive medium and the failure of any attached device should not cause failure of the whole system

Disadvantages of the Bus Organization

- Main drawback is performance
  - All memory references pass through the common bus
  - Performance is limited by bus cycle time
- Each processor should have cache memory
  - Reduces the number of bus accesses
- Leads to problems with cache coherence
  - If a word is altered in one cache it could conceivably invalidate a word in another cache
    - To prevent this the other processors must be alerted that an update has taken place
  - Typically addressed in hardware rather than the operating system

Multiprocessor Operating System Design Considerations (1 of 4)

- **Simultaneous concurrent processes**
  - OS routines need to be reentrant to allow several processors to execute the same IS code simultaneously
  - OS tables and management structures must be managed properly to avoid deadlock or invalid operations
- **Scheduling**
  - Any processor may perform scheduling so conflicts must be avoided

Multiprocessor Operating System Design Considerations (2 of 4)

  - Scheduler must assign ready processes to available processors
- **Synchronization**
  - With multiple active processes having potential access to shared address spaces or I/O resources, care must be taken to provide effective synchronization
  - Synchronization is a facility that enforces mutual exclusion and event ordering

Multiprocessor Operating System Design Considerations (3 of 4)

- **Memory management**
  - In addition to dealing with all of the issues found on uniprocessor machines, the OS needs to exploit the available hardware parallelism to achieve the best performance
  - Paging mechanisms on different processors must be coordinated to enforce consistency when several processors share a page or segment and to decide on page replacement

Multiprocessor Operating System Design Considerations (4 of 4)

- **Reliability and fault tolerance**
  - OS should provide graceful degradation in the face of processor failure
  - Scheduler and other portions of the operating system must recognize the loss of a processor and restructure accordingly

Cache Coherence (1 of 3)

Software Solutions

- Attempt to avoid the need for additional hardware circuitry and logic by relying on the compiler and operating system to deal with the problem
- Attractive because the overhead of detecting potential problems is transferred from run time to compile time, and the design complexity is transferred from hardware to software
  - However, compile-time software approaches generally must make conservative decisions, leading to inefficient cache utilization

Cache Coherence (2 of 3)

Hardware-Based Solutions

- Generally referred to as cache coherence protocols
- These solutions provide dynamic recognition at run time of potential inconsistency conditions
- Because the problem is only dealt with when it actually arises there is more effective use of caches, leading to improved performance over a software approach
- Approaches are transparent to the programmer and the compiler, reducing the software development burden

Cache Coherence (3 of 3)

- Can be divided into two categories:
  - Directory protocols
  - Snoopy protocols

Directory Protocols

- Collect and maintain information about copies of data in cache
- Directory stored in main memory
- Requests are checked against directory
- Appropriate transfers are performed
- Creates central bottleneck
- Effective in large scale systems with complex interconnection schemes

Snoopy Protocols (1 of 2)

- Distribute the responsibility for maintaining cache coherence among all of the cache controllers in a multiprocessor
  - A cache must recognize when a line that it holds is shared with other caches
  - When updates are performed on a shared cache line, it must be announced to other caches by a broadcast mechanism
  - Each cache controller is able to snoop on the network to observe these broadcast notifications and react accordingly

Snoopy Protocols (2 of 2)

- Suited to bus-based multiprocessor because the shared bus provides a simple means for broadcasting and snooping
  - Care must be taken that the increased bus traffic required for broadcasting and snooping does not cancel out the gains from the use of local caches
- Two basic approaches have been explored:
  - Write invalidate
  - Write update (or write broadcast)

Write Invalidate (1 of 2)

- Multiple readers, but only one writer at a time
- When a write is required, all other caches of the line are invalidated
- Writing processor then has exclusive (cheap) access until line is required by another processor
- Most widely used in commercial multiprocessor systems such as the x86 architecture
- State of every line is marked as modified, exclusive, shared or invalid
  - For this reason the write-invalidate protocol is called **MESI**

Write Invalidate (2 of 2)

- Can be multiple readers and writers
- When a processor wishes to update a shared line the word to be updated is distributed to all others and caches containing that line can update it
- Some systems use an adaptive mixture of both write-invalidate and write-update mechanisms

MESI Protocol (1 of 2)

To provide cache consistency on an SMP the data cache supports a protocol known as MESI:

- Modified
  - The line in the cache has been modified and is available only in this cache
- Exclusive
  - The line in the cache is the same as that in main memory and is not present in any other cache

MESI Protocol (2 of 2)

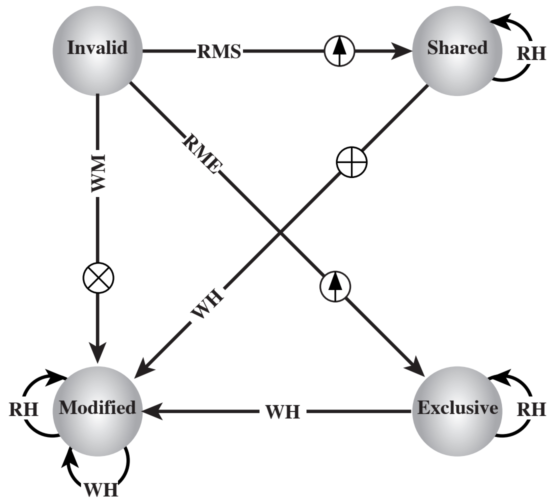
- Shared
  - The line in the cache is the same as that in main memory and may be present in another cache
- Invalid
  - The line in the cache does not contain valid data

MESI Cache Line States

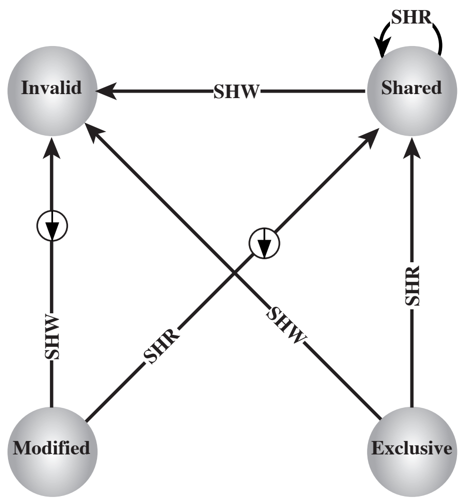
	M Modified	**E Exclusive**	**S Shared**	**I Invalid**
This cache line valid?	Yes	Yes	Yes	No
The memory copy is …	out of date	valid	valid	-
Copies exist in other caches?	No	No	Maybe	Maybe
A write to this line …	does not go to bus	does not go to bus	goes to bus and updates cache	goes directly to bus

Summarizes the meaning of the four states.

MESI State Transition Diagram (1 of 2)



(a) Line in cache at initiating processor



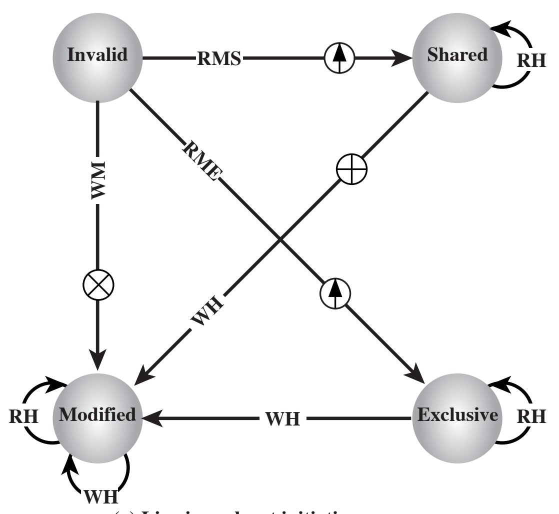
(b) Line in snooping cache

RH	Read hit	⇩	Dirty line copyback
RMS	Read miss, shared	⊕	Invalidate transaction
RME	Read miss, exclusive	⊗	Read-with-intent-to-modify
WH	Write hit	⇧	Cache line fill
WM	Write miss		
SHR	Snoop hit on read		
SHW	Snoop hit on write or read-with-intent-to-modify		

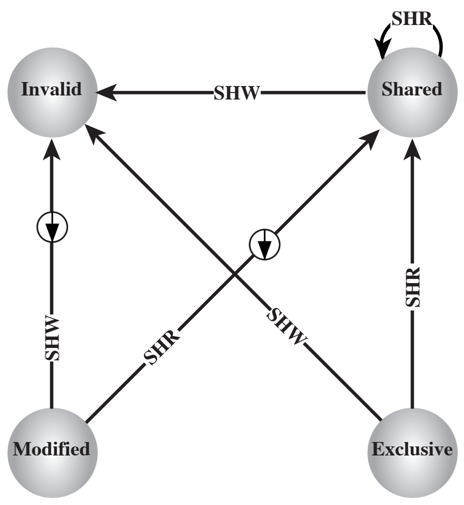
Figure above displays a state diagram for the MESI protocol. Keep in mind that each line of the cache has its own state bits and therefore its own realization of the state diagram.

Figure (a) shows the transitions that occur due to actions initiated by the processor attached to this cache. Figure (b) shows the transitions that occur due to events that are snooped on the common bus. This presentation of separate state diagrams for processor-initiated and bus-initiated actions helps to clarify the logic of the MESI protocol.

MESI State Transition Diagram (2 of 2)



(a) Line in cache at initiating processor



(b) Line in snooping cache

RH	Read hit
RMS	Read miss, shared
RME	Read miss, exclusive
WH	Write hit
WM	Write miss
SHR	Snoop hit on read
SHW	Snoop hit on write or read-with-intent-to-modify

↓	Dirty line copyback
⊕	Invalidate transaction
⊗	Read-with-intent-to-modify
↑	Cache line fill

At any time a cache line is in a single state. If the next event is from the attached processor, then the transition is dictated by Figure (a) and if the next event is from the bus, the transition is dictated by Figure (b).