1. Compare centralized shared memory multiprocessors, distributed shared memory multiprocessors and distributed message passing multiprocessors in terms of organization, interconnect, programming environment, scalability, and cache coherence.

Centralized shared memory multiprocessor: 1) has one bank of DRAM connected to all processors over a single shared bus, 2) Common global memory address space in which any processor can access any byte in memory using a unique address value for each byte, 3) Very limited scalability due to single bus and single DRAM block becoming a bottleneck as the number of processors increase and the traffic to memory increases, 4) typically uses snoopy cache coherence protocol.

Distributed shared memory multiprocessor: 1) multiple banks of DRAM connected to multiple processors over a interconnect network, with no single shared bus anywhere in the network, 2) Common global memory address space in which any processor can access any byte in memory using a unique address value for each byte, 3) highly scalable since it eliminates the bottlenecks of single memory bank and single shared bus, 4) uses directory based cache coherence protocol.

Distributed message passing multiprocessor: 1) multiple banks of DRAM connected to multiple processors over a interconnect network, with no single shared bus anywhere in the network, 2) each processor has its own memory address space, which means there is no global memory that allows one processor to address data belonging to another processor. Data communication between processors happens through send() and get() library function calls placed by the programmer in the parallel program. 3) highly scalable since it eliminates the bottlenecks of single memory bank and single shared bus, 4) since there is no global memory address space or data sharing between processors, it does not require maintaining cache coherence.

1. A distributed shared memory multiprocessor and a distributed message passing multiprocessor both use 10 32-bit processors. What is the maximum memory capacity of each system?

Shared memory multiprocessor capacity: 232 bytes.

Message passing multiprocessor capacity: 10 x 232 bytes.

1. List and briefly describe the 4 classes of computers in Flynn’s classical taxonomy.

Single instruction single data stream (SISD): one cpu computer that has no thread parallelism support in the hardware.

Single instruction multiple data streams (SIMD): one cpu with one program counter to fetch instructions sequentially, but each instruction operates on operands where each operand contains more than one data element. This is why SIMD computers are also called vector computers.

Multiple instruction single data stream (MISD): multiple cpus executing different programs on the same data, e.g., running different decryption algorithms on an encrypted file.

Multiple instruction multiple data streams (MIMD) multiple cpus processing different data at the same time.

1. Briefly discuss OpenMP parallel environment execution model and its 3 components.

OpenMP is a shared memory parallel programming model that executes multiple threads in selected parallel regions in a program. Pragmas are used to define parallel regions and other thread and data objects properties. Threads are forked from a master thread at the beginning of a parallel region and join into a master thread at the end of the parallel region. The environment consists of 3 components: compiler directives, runtime library routines and environment variables.

1. How can a programmer determine the ID of a thread and the number of threads in a parallel region?

omp\_get\_thread\_num() and omp\_get\_num\_threads() respectively.

1. Explain the problem in the following code segment. How do you fix it?

#pragma omp parallel

{

x = x + 1;

}

The above statement is compiled into a sequence of load, add, and store instructions. Since each parallel thread can execute its sequence interleaved in any time order relative to another thread sequence, the final computed value of x at the end of the parallel region is nondeterministic. To fix the problem, critical region or atomic directive has to be used.

1. A uniform memory access multiprocessor uses a snoopy MSI cache coherence protocol. Describe what happens when:

a) A store misses the cache in one processor and the cache block is in exclusive state in another processor’s cache.

b) A store hits a block in shared state in some processor.

c) A load misses the cache in one processor and the cache block is in exclusive state in another processor’s cache.

1. The processor that misses puts a write miss request on the shared bus. The other processor that has the block in exclusive state snoops the write-miss bus request, writes back its block to memory, invalidates the block in its cache, and sets “abort memory read” signal. Processor one grabs the written back block from the bus and puts the block in its cache in exclusive state.
2. The processor that hits puts invalidate request on the shared bus, writes its block with the store instruction, and changes the block state to exclusive. The other processor invalidates the shared block in its cache.
3. The processor that misses puts a read miss request on the shared bus. The other processor writes back the block from its cache and changes the block to shared state. The first processor grabs the written back block from the bus and puts it in its cache in shared state.
4. Which of the following statement about SIMD computers is **False?**

a. All processing units execute the same instruction at any given clock cycle

b. Each processing unit can operate on a different data element.

c. Vector processors are examples of SIMD architectures.

d. SIMD is rarely used in modern multicore processors.

e. One of the above is False.

1. Which of the following computers does not exploit parallel execution?

a. Pipelined computer.

b. Superscalar microprocessor.

c. NUMA multiprocessor.

d. SISD computer running a multitasking operating system.

e. All of the above exploit parallel execution.

1. A cache has a total capacity of 16K bytes. It is implemented as 2-way set associative, with block size of 32 bytes. The physical address on the machine consists of 32 bits. Which of the following statements is **TRUE**?

a. Number of set index bits = 8 and number of tag bits = 19

b. Number of set index bits = 8 and number of tag bits = 20

c. Number of set index bits = 9 and number of tag bits = 18

d. Number of set index bits = 9 and number of tag bits = 19

e. None of the above

1. Which of the following statements is **False**?

a. OpenMP specifies nothing about parallel I/O.

b. In OpenMP, a parallel region is a block of code that will be executed by multiple threads.

c. In OpenMP, a parallel region is a block of code that does not contain any shared data objects.

d. It is illegal to branch out of a parallel region.

e. In OpenMP, work-sharing constructs do not launch new threads

1. A cache has a total capacity of 32K bytes. It is implemented as 4-way set associative, with block size of 32 bytes. The physical address on the machine consists of 32 bits.

How many bits is each of the byte offset, set index and address tag?

Byte offset: 5 bits, Set index: 8 bits, Tag: 19 bits

1. List the sequence of messages in an 8-node distributed multiprocessor system that uses the MSI directory based invalidate cache coherence protocol, when:

A store to memory block of address A misses in processor 1 due to a set conflict with a block of address B in exclusive state and memory block with address A is in shared state in processors 4 and 5.

Assume that memory blocks A and B directory entries are in nodes 2 and 8 respectively. Recall that this means that address A belongs to the DRAM array in node 2 and address B belongs to the DRAM array in node 8. Specify for each message its type as well as the source node and destination node.

Here, the nodes are numbered 1-8.

Message 1: Write miss from node 1 to node 2.

Message 2: Invalidate from node 2 to node 4.

Message 3: Invalidate from node 2 to node 5.

Message 4: Write-in data from node 2 to node 1.

Message 5: Write-back data from node 1 to node 8.

Final block A state in directory 2: 10 00000001.

Final block B state in directory 8: 00 00000000.

1. List the sequence of messages and the final block state in an 8-node distributed multiprocessor system that uses the MSI directory based invalidate cache coherence protocol, when:

Block in node 0 directory in state 10 00000010, Node 3 has CPU read miss to Invalid State.

While here and in later problems, the nodes are numbered 0-7. Sorry about the inconsistency.

Message 1: Read miss from node 3 to node 0.

Message 2: Fetch from node 0 to node 1.

Message 3: Write-back data from node 1 to node 0.

Message 4: Write-in data from node 0 to node 3.

Final block state in directory 0: 01 00001010.

1. List the sequence of messages and the final block state in an 8-node distributed multiprocessor system that uses the MSI directory based invalidate cache coherence protocol, when:

Block in node 0 directory in state 10 00000010, Node 3 has CPU write miss to Invalid State.

Message 1: Write miss from node 3 to node 0.

Message 2: Fetch-Invalidate from node 0 to node 1.

Message 4: Write-back data from node 1 to node 0.

Message 4: Write-in data from node 0 to node 3.

Final block state in directory 0: 10 00001000.

1. List the sequence of messages and the final block state in an 8-node distributed multiprocessor system that uses the MSI directory based invalidate cache coherence protocol, when:

Block in directory 0 in state 10 00000010, Node 3 has CPU Write miss to Exclusive State and Exclusive block home node is node 5.

Message 1: Write miss from node 3 to node 0.

Message 2: Fetch from node 0 to node 1.

Message 3: Write-back data from node 1 to node 0.

Message 4: Write-in data from node 0 to node 3.

Message 5: Write-back data from node 3 to node 5.

Final block A state in directory 0: 10 00001000.

Final block B state in directory 5: 00 00000000.

1. List the sequence of messages and the final block state in an 8-node distributed multiprocessor system that uses the MSI directory based invalidate cache coherence protocol, when:

Block in node 0 directory in state 01 00000110, Node 1 has CPU Write Hit.

Message 1: Invalidate from node 1 to node 0.

Message 2: Invalidate from node 0 to node 2.

Final block in directory 0: 10 00000010.

1. List the sequence of messages and the final block state in an 8-node distributed multiprocessor system that uses the MSI directory based invalidate cache coherence protocol, when:

Block in node 0 directory in state 01 00000110, Node 4 has CPU Read Miss.

 Message 1: Read miss from node 4 to node 0.

Message 2: Write-in data from node 0 to node 4.

Final block in directory 0: 01 00010110.

1. List the sequence of messages and the final block state in an 8-node distributed multiprocessor system that uses the MSI directory based invalidate cache coherence protocol, when:

Block in node 0 directory in state 01 00000110, Node 4 has CPU Write Miss

 Message 1: Write miss from node 4 to node 0.

Message 2: Invalidate from node 0 to node 1.

Message 3: Invalidate from node 0 to node 2.

Message 4: Write-in data from node 0 to node 4.

Final block in directory 0: 10 00010000.

TABLE . MSI snoopy cache coherence transitions due to cpu requests

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **CPU Request** | **Current State** | **Hit/Miss** | **Action** | **Next State** |
| Load  | Invalid | Miss | Place read for load on bus | Shared |
| Store | Invalid | Miss | Place read for store on bus | Modified |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

TABLE 2. MSI snoopy cache coherence transitions due to other CPU bus transactions

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Bus transaction** | **Current State** | **Hit** | **Action** | **Next State** |
| Read for load | Modified | Yes | Writeback, abort memory read | Shared |
| Read for store | Modified | Yes | Writeback, abort memory read | Invalid |
| Invalidate | Shared | Yes | None | Invalid |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

1. The snoopy bus protocol discussed in the class is commonly known as M/S/I snoopy cache coherence protocol, where:

M = Dirty-Exclusive state: block is dirty and is only in one cache.

S = Replicated state: block is clean in one or more caches.

I = invalid: block is not in the cache.

Dirty means block has been written by a store from by cache’s CPU.

Clean means block has been read but has not been written by the cache’s CPU.

A snoopy cache coherence protocol state machine could be represented as a table, as shown in Tables 1 and 2 above. Complete the above tables for all possible internal CPU requests and external bus transactions events.

1. Create similar tables for M/E/S/I snoopy cache coherence protocol, where:

M = Dirty-Exclusive state: block is dirty and is only in one cache.

E = Clean-Exclusive block is clean and is only in one cache

S = Replicated state: block is clean in one or more caches

I = invalid: block is not in the cache.

1. Recall that the MESI cache coherence protocol defines 4 coherence states:

M = Dirty-Exclusive state: block is dirty and is only in one cache.

E = Clean-Exclusive block is clean and is only in one cache

S = Replicated state: block is clean in one or more caches

I = invalid: block is not in the cache.

Write a short piece of assembly code that benefits in performance from the E state. Explain where the performance benefit comes from.

1. A data cache uses MESI snoopy cache coherence protocol. A store instruction from the same CPU hits a block in exclusive state. Which of the following statements is **TRUE?**

a. The other processors invalidate their blocks that have the same address.

b. This processor writes its block and places an invalidate transaction on the external bus.

c. This processor writes its block and changes the block to modified state.

d. All of the above.

e. None of the above.

1. What is the difference between a shared memory multiprocessor computer and a centralized memory multiprocessor computer?

Shared memory multiprocessor is a multiprocessor where any byte in memory can be accessed by any processor using one unique address. Centralized memory multiprocessor means there is one bank of memory connected to multiple processors over a shared bus.

1. The following OpenMP program has two bugs. What are these two bugs and how would you fix them?

#include <omp.h>

#include <stdio.h>

#include <stdlib.h>

int main (int argc, char \*argv[])

{

int nthreads, i, tid;

float total;

/\*\*\* Spawn parallel region \*\*\*/

#pragma omp parallel

{

/\* Obtain thread number \*/

tid = omp\_get\_thread\_num();

/\* Only master thread does this \*/

if (tid == 0) {

 nthreads = omp\_get\_num\_threads();

 printf("Number of threads = %d\n", nthreads);

}

printf("Thread %d is starting...\n",tid);

#pragma omp barrier

/\* do some work \*/

total = 0.0;

#pragma omp for schedule(dynamic,10)

for (i=0; i<1000000; i++)

 total = total + i\*1.0;

printf ("Thread %d is done! Total= %e\n",tid,total);

} /\*\*\* End of parallel region \*\*\*/

}

1. tid should be private.
2. Reduction clause should be used for total in the statement total = total + i\*1.0;