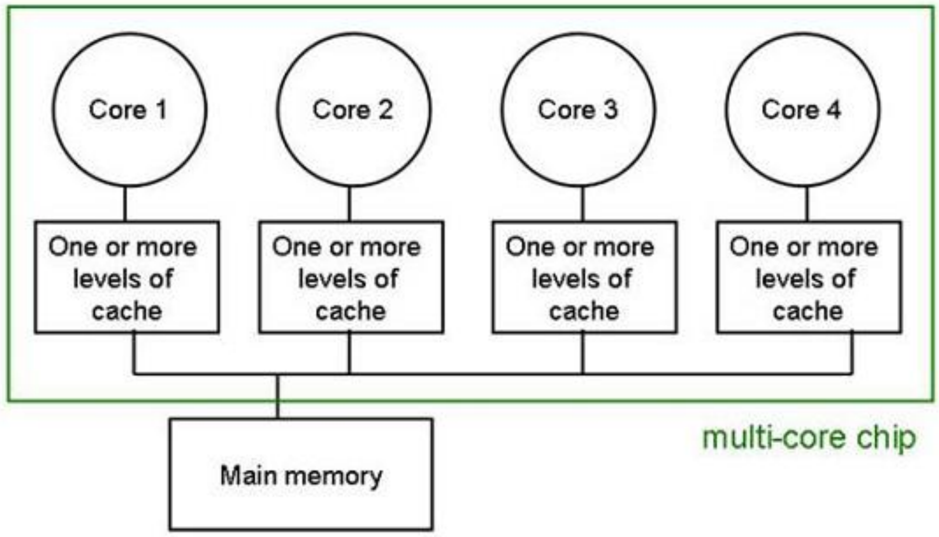
CPS5401 Fall 2012 Name ______________________

Shirley Moore, Instructor

Take-home quiz (Practice test)

Due Tuesday, October 23

1. Consider the multicore processor shown below (figure from csail.mit.edu).


Core 1
Core 2
Core 3
Core 4
One or more levels of cache
One or more levels of cache
One or more levels of cache
One or more levels of cache
Main memory
multi-core chip

a) Explain why some sort of cache coherence protocol is needed on this processor.

b) Assume the processor uses a basic MSI cache coherence protocol where the three states that a cache line can be in are Modified, Shared, and Invalid. Assume that the caches use a write back policy. Give a possible correct set of states for the cache line on each core containing the variable x and for the value of x on each core and in main memory for the sequence of events in the table below. Assume that initially the value of x is 7 and the cache line is not in any of the caches.

Event	Core 1		Core 2		Core 3		Core 4		Value of x in memory
	State	x	State	x	State	x	State	x	
Cores 1,2,3 read x									
Core 4 adds 1 to x									
Core 2 reads x									

2. a) What is bisection bandwidth and why is it used as a measure of network performance?

b) The bisection bandwidth of an N-dimensional torus network with P processors is given by $2P^{(N-1)/N}B$, where B is the bandwidth of a single link. Suppose you are the designer of a line of supercomputers that use a 3D torus network. Assume that your current products all have at least 100,000 processors. If you were to design a new line with a 5D torus network and double the number of processors, how would the bisection network of the product change? Please express your answer in terms of the number of processors P of the original product. Would the bisection bandwidth improve or degrade with the change?

3. Why does a direct mapped cache have a lower hit latency than a set associative cache of the same capacity?

4. Consider the following two loops, written in C, that calculate the sum of the entries in a 128-by-64 matrix of 32-bit integers.

Loop A	***Loop B***

`sum = 0;`
`for (i = 0; i < 128; i++)`
`  for (j = 0; j < 64; j++)`
`    sum += A[i][j];`
	`sum = 0;`
	`for (j = 0; j < 64; j++)`
	`  for (i = 0; i < 128; i++)`
	`    sum += A[i][j];`

The matrix A is stored contiguously in memory in row-major order. Row-major order means that elements in the same row of the matrix are adjacent in memory as shown in the following memory layout:

0	4		252	256		4*(64*127+63)
A[0][0]	A[0][1]	...	A[0][63]	A[1][0]	...	A[127][63]

Assume that only accesses to matrix A cause memory references and that all other variables are stored in registers and that instructions are in a separate instruction cache. Consider a direct-mapped cache with 32-byte cache lines. Assume the cache is initially empty.

a) Calculate the minimum cache size required if loop A is to run without any cache misses other than initial compulsory misses. Please explain your answer.

b) Calculate the minimum cache size required if loop B is to run without any cache misses other than compulsory misses. Please explain your answer.

5. Explain what is wrong with the OpenMP code fragment below and provide a correct and efficient version. Note that only the outer loop will be parallelized by the parallel directive and that local variables within the parallelized loop (e.g., offset) are automatically private.

```
#pragma omp parallel for private(j,k) shared(sum1,sum2)
      for(i=0; i<threads; i++) {
            int offset=i*array_size;
            for(j=0; j<(iterations*(vec_ratio)); j++) {
                  for(k=0; k<array_size; k++) {
                        sum1 += a[k+offset] * s1;
                        sum2 += a[k+offset] * s2;
                  }
            }
      }
```

6. In an MPI program, how do you assign different work to each processor if all processors are running the same program?

7. What three lines are required in any MPI program and what is the purpose of each? (You may describe them rather than giving exact C or Fortran code)

8. Could the point-to-point communication sketched below ever lead to deadlock? Explain why or why not.

```
Process 0     Process 1
ISend(1)      ISend(0)
IRecv(1)      IRecv(0)
Waitall       Waitall
```