

CS 194-2: Quiz 1 Review (10 Oct 2007)

1. What is false sharing? Why is the following code a victim of it? List two different ways to avoid false sharing in this example. Identify the method that requires some knowledge about the hardware; what do you need to know?

```
double sum[num_threads];

void ThreadFunc(void *data)
{
    .
    .
    for (i=0; i<N; i++)
        sum[data->myID] += p[i];
    .
    .
}
```

2. Identify the race conditions in the two examples below. What is causing the race condition? What thread construct(s) can be applied to make the code thread-safe? What are some other methods of eliminating race conditions?

a. Assume that `consume_thing()` is thread-safe.

```
nthings = n;

p0: while(nthings > 0) { consume_thing(); nthings--; }
p1: while(nthings > 0) { consume_thing(); nthings--; }
```

b. Assume that you are using the system PRNG `srand()` and `rand()`.

```
p0: x = rand();
p1: y = rand();
```

3. Describe the differences between the threads model of shared-memory parallelism, and the SIMD model of parallelism. Focus on the instruction-set architecture and the hardware, rather than the high-level programming model presented by a language or library.
4. What are some of the optimizations used for matrix-matrix multiply? Describe the potential benefit of each optimization. Which of these optimizations would help with matrix-vector multiply?

CS 194-2: Quiz 1 Review (10 Oct 2007)

5. Define sequential consistency. Based on the following two concurrent tasks T1 and T2, which values for X' and Y' are possible after execution? Assume that initially X=0 and Y=10, and that R1 and R2 are registers.

```
T1:
store(X), 1    //(X=1)
store(Y), 11   //(Y=11)
```

```
T2:
load R1, (Y)
store(Y'), R1  //(Y'=Y)
load R2, (X)
store(X'), R2  //(X'=X)
```

(X',Y') -> {(1,11), (0,11), (0,10), (1,10)} ?

6. Suppose that you have a stack data structure implemented as a singly linked list (with a pointer to the head). The stack is visible to all the threads, but each thread only has access to the head. Show an interleaving of instructions between two threads that would result in corruption of the data structure or loss of an invariant. How would you apply threading constructs to make the stack thread-safe? Now suppose that a thread can read values from the middle of the stack. How would you then ensure that the data structure is thread-safe?
7. Name two different types of task queues. Which one probably performs the best under the most general circumstances? What are the disadvantages of the other type? List some potential pitfalls or sources of (correctness and / or performance) bugs when implementing the solution you think is better. Also, identify a situation in which you might prefer the other solution.