CS4961: Parallel Programming Midterm Exam
October 20, 2011

Instructions:
This is an in-class, open-book, open-note exam. Please use the paper provided to submit your responses. You can include additional paper if needed. The goal of the exam is to reinforce your understanding of issues we have studied in class.

CS4961: Parallel Programming
Midterm Quiz
October 20, 2011

I. Definitions (10 points)

Provide a very brief definition of the following terms:

a. Barrier

b. Candidate Type Architecture

c. #pragma omp task

d. Bisection bandwidth

e. Snooping cache coherence

II. Short Answer (30 points)

a. Does the following code have a loop-carried dependence? If so, identify the dependence or dependences and which loop carries it/them.

```
for (i=1; i < n-1; i++)
  for (j=1; j < n-1; j++)
    A[i][j] = A[i][j-1] + A[i-1][j+1];
```

b. Briefly compare OpenMP and Pthreads by describing the advantages and disadvantages of each language.

c. What sort of cache locality is achievable in the following code and how would you obtain it? (Assume row major ordering in memory for the arrays.)

```
for (j=0; j<n; j++)
    for (i=0; i<n; i++)
        A[i][j] += B[i][j];
```

d. Suppose you had a Pthreads program where most of the time you were performing read-only operations on a specific data structure but occasionally you were updating the data structure. How could you synchronize accesses to that data structure to avoid race conditions and still mostly access the data structure in parallel.

e. If you were to parallelize the following loop in OpenMP, what directives would you provide to achieve a static scheduling of iterations in block-cyclic order where each thread would execute 8 consecutive iterations of the loop?

```
for (i=0; i<n; i++)
      A[i] += B[i];
```

III. Problem Solving (60 points)

In this set of three questions, you will be asked to provide code solutions to solve particular problems. This portion of the exam may take too much time if you write out the solution in detail. I will accept responses that sketch the solution, without necessarily writing out the code or worrying about correct syntax. Just be sure you have conveyed the intent and issues you are addressing in your solution.

a. Rewrite the following sequential code in OpenMP for producer-consumer task parallelism.

```
for (i=0; i<INPUT_SIZE; i++) {

   // TASK 1: Finite Impulse Response (FIR) filter
   for (j=0; j<TAP_SIZE; j++) {
       sum += sample[i+j] * coeff[i];
   }
   data_out[i] = sum;

  // TASK 2: Multiply by coefficient
   final[i] = data_out[i];
   for (j=0; j<n; j++) {
       final[i] *= coeff2[j];
   }
}
```

b. How would you rewrite the following code to perform as many operations as possible in SIMD mode for SSE and achieve good SIMD performance? Assume the data type for all the variables is 32-bit integers, and the superword width is 128 bits. Briefly justify your solution.

```
k = ?; // k is an unknown value in the range 0 to n, and b is declared to be of size 2n
for (i= 0; i<n; i++) {
    x1 = i1 + j1;
    x2 = i2 + j2;
    x3 = i3 + j3;
    x4 = i4 + j4;
    x5 = x1 + x2 + x3 + x4;
    a[i] = b[n+i]*x5;
}
```

c. Sketch out how to rewrite the following code to improve its cache locality and locality in registers.

```
for (i=1; i<n; i++)
    for (j=1; j<n; j++)
        a[j][i] = a[j-1][i-1] + c[j];
```

Briefly explain how your modifications have improved memory access behavior of the computation.