TRINITY COLLEGE DUBLIN

THE UNIVERSITY OF DUBLIN

Faculty of Engineering, Mathematics & Science
School of Computer Science & Statistics

Integrated Computer Science Programme **Trinity Term 2015**
Year 3 Annual Examinations

Concurrent Systems 1 (CS3014)

Tuesday 28th April, 2015 **Exam Hall** **09:30 – 11:30**

Dr David Gregg

Instructions to Candidates:

- Answer 2 out of the 3 questions
- All questions are marked out of 50
- All program code should be commented appropriately

Materials permitted for this examination:

- Non-programmable calculator

1a) As computing moves away from the desktop to tablets, phones and wearable computing devices, arguably raw computing performance has become less important than performance/Watt. To what extent do multicore architectures have a role in low power computing? Given the increasing importance of low power, please describe what you consider in your opinion to be the likely main types of processors in 5-10 years. How will they differ from today?

[10 marks]

b) Examine each of the following pieces of code. State if each individual piece of code can be vectorized using SSE. If not, state clearly why. If the code can be vectorized, use SSE intrinsics to vectorize it. Write a short note explaining the parallelization strategy on any code you vectorize.

```
/* code segment 1 */
void mul_neighbour(float * array, int size)
{
      for (int i = 0; i < size-1; i++ ) {
            array[i] = array[i] * array[i+1];
}
}

/* code segment 2 */
/* note that the restrict keyword indicates that arrays
   a and b do not overlap */
float dot_product(float *restrict a, float *restrict b)
{
      float sum = 0.0;
      for ( int i = 0; i < 4096; i++ ) {
            sum = sum + a[i] * b[i];
      }
      return sum;
}
```

(Question 1 continues on next page)...

...(Question 1 continued from previous page)

```
/* code segment 3 */
float rgb_sum_product(float * pixels) {
        for ( int i = 0; i < 1011; i += 3 ) {
                sum_red = sum_red + pixels[i];
                sum_green = sum_green + pixels[i+1];
                sum_blue = sum_blue + pixels[i+2];
    }
    return sum_red * sum_green * sum_blue;
}

    /* code segment 4 */
int max_index(float * array)
    {
        float max = array[0];
        int max_idx = 0;
    for ( int i = 1; i < 4096, i++ ) {
        if ( array[i] > max ) {
                max_idx = i;
        }
                }
                return max_idx;
    }
```

[10 marks for each segment]

2. Computers are always designed with typical applications in mind. The result is that computers to solve different problems have very different features. The following C code implements a simplified version of a key part of the code for Cholesky decomposition, which divides a square input matrix into a lower triangular part and another factor.

```
void cholesky(float ** input, float ** result, int size)
{
for ( int i = 0; i < size; i++ ) {
      for ( int j = 0; j <= i; j++ ) {
            float sum = 0.0;
            for ( int k = 0; k < j; k++ ) {
                  sum += result[i][k] * result[j][k];
            }
            // if this is on the diagonal of the matrix
            if ( i == j ) {
                  result[i][j] = sqrt(input[i][i] - sum);
            }
            // else not on diagonal
            else {
                  result[i][j]= 1/result[j][j]*(input[i][j]s);
            }
      }
}
}
```

a. Identify any potential parallelism in the above code.

[10 marks]

(Question 2 continues on next page)...

...(Question 2 continued from previous page)

b. Discuss the suitability of various parallel computer architectures for executing this code, assuming a large array. The parallel architectures you should discuss are:

i. out-of-order superscalar

ii. very long instruction word (VLIW)

iii. vector processor

iv. multithreaded/simultaneous multithreaded

v. shared memory multi-core processor

vi. multiple chip symmetric multiprocessor

vii. NUMA multiprocessor

viii. distributed memory multicomputer.

You should explain which aspects of each architecture suit the code well and identify any likely bottlenecks or problems in the running of the code. For each architecture you should also describe any programmer intervention that may be required to parallelize the code for the particular architecture.

[5 marks per architecture]

3. Detecting collisions between objects in computer games is an important problem. A common technique is to place each object in a *bounding sphere*, and check whether the spheres have collided. At any point in time, two spheres have collided if the distance between their centres is less than the sum of their radii. An outline of a simple sphere abstract data type is as follows:

```
struct sphere {
        double centre_x;
        double centre_y;
        double centre_z;
        double radius;
};
// add any functions here that operate on the sphere
```

Write a parallel function using OpenMP and C that takes an array of sphere objects as a parameter, and returns the set of pairs of objects that have collided. Your function should be parallel, and should make efficient use of machine resources on a modern multi-core computer. The prototype of your function should look like:

```
int detect_collisions(struct sphere * spheres, int * collisions,
int num_spheres);
```

The parameter *spheres* is an array of spheres that you need to check against one another for collisions. The number of sphere is *num_spheres*. The parameter *collisions* is an array of integers. Each adjacent pair of integers in the *collisions* array represent the fact that the corresponding *spheres* in the spheres array have collided. For example, if *collisions[0]* has the value 10, and *collisions[1]* has the value 14, it means that the spheres at positions 10 and 14 in the *spheres* array have collided. The order in which the pairs of colliding items appear in the *collisions* array is not important. You should assume that collisions are common. Finally, the function should return the number of collisions.

The distance between two points, *p* and *q*, with x, y and z coordinates is:

$$\sqrt{(p.x-q.y)^2+(p.y-q.y)^2+(p.z-q.z)^2}$$

[30 marks]

(Question 3 continues on next page)...

...(Question 3 continued from previous page)

Write a short commentary on your function explaining how it works, and why you believe it to be efficient. You should comment on the time and space complexity of your solution, and also explain why you think it is likely to run efficiently on a modern multi-core architecture.

[20 marks]