THE AUSTRALIAN NATIONAL UNIVERSITY
First Semester Examination – June 2014

COMP3320/6464/HONS
High Performance Scientific Computing

Study Period: 15 minutes
Time Allowed: 3 hours
Permitted Materials: Non-Programmable Calculator

Exam questions total 100 marks.
All students answer all questions
Clarity and conciseness in answers will be highly valued
Marks may be lost for supplying irrelevant information
Question 1 [20 marks]

Short Questions

(a) The pharmaceutical industry is a large user of high performance computing. Give one example of how you think high performance computing might be used in the design of a new drug. Explain why this task is computationally demanding. [2 marks]

(b) Virtually all new computers are NUMA. What is NUMA? Draw a diagram to illustrate a NUMA computer system. [2 marks]

(c) When is it better to have a few powerful computers, rather than many less powerful computers? When would the opposite be true? [2 marks]

(d) Give one advantage of the Python programming language over C, and one example of C over Python. [2 marks]

(e) Each of the following statements are either true or false. Which are they? Give a few sentences to explain each of your answers.

   (i) If on a computer two real numbers are exactly representable as floating-point numbers, then the result of adding them together will also be exactly representable. However, this is NOT always true if the numbers are subtracted due to cancellation.

   (ii) Truncation or approximation error is the difference between the true value and the value computed using a given algorithm with infinite precision arithmetic.

   (iii) In a floating point number system the underflow limit is the smallest positive machine number that perturbs the number 1 when added to it. [6 marks]

(f) Order the following events according to increasing execution time on a typical computer. For each you should provide a rough estimate of the time it takes together with a brief justification of your estimate.

The time to:

1. Perform a single floating point multiplication between two data items that must be fetched from main memory.

2. Perform a single floating point division on two data items that are currently available in registers.

3. Complete a dot product between two vectors of length 100 with data available in level 1 cache.

4. Add a value to a shared variable within a critical section of an OpenMP program running with 10 threads.

5. Perform one timestep of a molecular dynamics simulation with 1000 particles interacting with each other via a pair potential (like in Assignment 1). [6 marks]
Question 2 [20 marks]

High Performance Computer Architecture

(a) The typical memory hierarchy of a modern processor includes registers, cache, main memory and disk storage. For each of these components outline its typical size, access time, bandwidth and what part of the compute environment is responsible for managing this resource.

[5 marks]

(b) What is a cache line conflict? Illustrate your answer by outlining a small code fragment that will exhibit a cache line conflict on a computer with a 16KB 2-way set associate cache that has a 32byte cache line size.

[5 marks]

(c) Over time there has been a tendency for all microprocessors to increase the number of stages in their execution pipeline. Why? What is the potential disadvantage of long pipelines and how have microprocessors designers sought to minimize this disadvantage?

[4 marks]

(d) On a shared memory parallel computer the words x[0] and x[1] map to the same cache line. At time T0 both processors P1 and P2 issue read requests bringing this cache line into their respective caches. In subsequent time steps the following instructions are issued:

<table>
<thead>
<tr>
<th>Time</th>
<th>P1</th>
<th>P2</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>read x[0]</td>
<td>read x[1]</td>
</tr>
<tr>
<td>1</td>
<td>write x[0]</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>write x[0]</td>
<td>read x[1]</td>
</tr>
<tr>
<td>3</td>
<td>read x[1]</td>
<td>write x[1]</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>read x[1]</td>
<td></td>
</tr>
</tbody>
</table>

(i) On each processor what happens to the cache line containing x[0] and x[1] at the various time steps.

(ii) The above example shows one (or more) misses due to false sharing. What is a false sharing miss, and when do such misses occur in the above access sequence?

[6 marks]
Question 3  [20 marks]

OpenMP

(a) What is a race condition? Sketch a few lines of OpenMP code that exhibit a race condition. Explain your code and why it may exhibit a race condition.  

[6 marks]

(b) For the following code, what value of `icount` do you expect to be printed by each thread if the program is run using 3 threads? Explain your answer specifying any assumptions you make about the behavior of OpenMP.

```c
#pragma omp parallel
{
    int icount=0;
    int iam=omp_get_thread_num();
#pragma omp for
    for (int k=0; k<10; k++) icount = icount + k;
    printf("Thread %d icount %d\n",iam,icount);
}
```

[7 marks]

(c) The following function uses recursion to compute Fibonacci numbers (recall $F_n = F_{n-1} + F_{n-2}$ where $F_1 = 1$ and $F_2 = 1$). Show how you might parallelise this routine using OpenMP tasks. Discuss how you expect your code to perform as a function of $n$, number of OpenMP threads, and the architecture of the machine your code is running on.

```c
int fib(int n)
{
    if (n<2)
        return n;
    else{
        int i=fib(n-1);
        int j=fib(n-2);
        return i+j;
    }
}
```

[7 marks]
Question 4 [20 marks]

Application of High Performance Computing

The following matrix:

\[
\begin{bmatrix}
-4 & 2 & 0 & 0 & 0 \\
2 & 0 & 8 & 0 & 0 \\
0 & 8 & 0 & -5 & 10 \\
0 & 0 & -5 & 0 & 0 \\
0 & 0 & 10 & 0 & -6
\end{bmatrix}
\]

is sparse, in that it contains many zero elements. Sparse matrices are frequently stored in Compressed Sparse Row (CSR) format. For example the above matrix may be stored using the following data structures:

\[
\text{val} = [-4, 2, 2, 8, 8, -5, 10, -5, 10, -6] \\
\text{col_idx} = [0, 1, 0, 2, 1, 3, 4, 2, 2, 4] \\
\text{row_ptr} = [0, 2, 4, 7, 8, 10]
\]

Using this representation the product of sparse matrix \text{val} containing \text{n_row} rows with vector \text{b} and placing the result in vector \text{c} can be computed using the following code.

```c
for (int i=0; i<n_row; i++){
    for (int j=row_ptr[i]; j<row_ptr[i+1]; j++){
        c[i]=c[i]+val[j]*b[col_idx[j]];
    }
}
```

(a) Discuss the performance of this code. In your answer consider the effect of iteration count for each of the two loops, the memory access patterns of each data structure, and the sensitivity of performance to cache characteristics. [8 marks]

(b) Sketch equivalent code for performing a matrix vector product that does NOT use CSR storage. That is, code to perform a standard dense matrix vector product. (Assume there are \text{n_col} columns, and that the matrix is stored in an array \text{I[r_row][n_col]}). [6 marks]

(c) Comment on how you think your code would perform compared to the CSR code as a function of overall matrix sparsity (the relative number of zero to non-zero elements in the matrix), the exact distribution of non-zero elements within each row, and the overall matrix dimensions (\text{n_row/n_col}). [6 marks]
Question 5  [20 marks]

SIMD Operations

(a) Intel SSE instructions allow a user to vectorize their code. Obstacles to using SSE instructions include i) non-contiguous memory accesses, ii) unaligned data structures and iii) data dependencies. For each of these items explain what is meant and why it can be an obstacle to the use of SSE.

[6 marks]

(b) Intel have progressively increased the size of their SIMD registers from the original 128 bit wide SSE registers to the recent 512 bit wide AVX-512 registers. What are the advantages and disadvantages of having longer SIMD registers? How might you need to change the cache and memory hierarchy in order to accommodate these larger register sizes?

[6 marks]

(c) The following text and code is taken directly from the course text book.

“The compiler, guided by the rules of the programming language and its interpretation of the source, must make certain assumptions that may limit its ability to generate optimal machine code. The typical example arises with pointer (or reference) formal parameters in the C (and C++) language:"

```c
void scale_shift(double *z, double *b, double s, int n){
    for (int i=1; i<n; i++)
        a[i] = s*b[i-1];
}
```

The issue being referred to is aliasing.

(i) Explain what aliasing is.

(ii) Why does possible aliasing prevent the compiler from vectorizing the above loop?

(iii) Assuming you force the compiler to vectorize the above loop. Give one example of how you might call this routine and expect it to produce identical results to when you call the same but unvectorized code, and one example of when you might call this routine and expect it to report different results. Explain the reasons behind your answers.

[8 marks]