

THE AUSTRALIAN NATIONAL UNIVERSITY
First Semester Examination – June 2003

COMP3320/COMP6464

High Performance Scientific Computing

Study Period: 15 minutes

Time Allowed: 3 hours

Permitted Materials: Calculator

Exam questions total 100 marks.

COMP3320 Answer Questions 1-5

COMP6464 Answer Questions 1-4 and 6

*Clarity and conciseness in answers will be highly valued
Marks may be lost for supplying irrelevant information*

Question 1 [30 marks]

(a) The following code illustrates the effects of both truncation and rounding errors:

```
#include <stdio.h>
#include <math.h>
int main(void)
{ double x=0.5,h;
  int i;
  for (i=0, h=0.1; i<14; i++, h*=0.1)
    printf("%16.14f %8.1e %16.14f \n",cos(x),h,
          (sin(x+h)-sin(x-h))/(2*h));
  return 0;
}
```

It computes the derivative of $\sin(x)$ both analytically and numerically using the finite difference scheme:

$$\frac{d \sin(x)}{dx} = \cos(x) \approx \frac{\sin(x+h) - \sin(x-h)}{2h}$$

where h is the step size used in the finite difference. The code uses $x = 0.5$ and various step sizes. It gives the following output:

```
0.87758256189037 1.0e-01 0.87612065543192
0.87758256189037 1.0e-02 0.87756793558747
0.87758256189037 1.0e-03 0.87758241562663
0.87758256189037 1.0e-04 0.87758256042766
0.87758256189037 1.0e-05 0.87758256187399
0.87758256189037 1.0e-06 0.87758256189266
0.87758256189037 1.0e-07 0.87758256156663
0.87758256189037 1.0e-08 0.87758256325554
0.87758256189037 1.0e-09 0.87758254889393
0.87758256189037 1.0e-10 0.87758256838246
0.87758256189037 1.0e-11 0.87758224352839
0.87758256189037 1.0e-12 0.87757202221241
0.87758256189037 1.0e-13 0.87752532020383
0.87758256189037 1.0e-14 0.87723339876888
```

- (i) Define rounding and truncation error.
- (ii) Describe how this program illustrates both sources of error.
- (iii) In evaluating the numerical derivative of $\sin(x)$ at which step size is the error minimized? For this step size, and in terms of the numerical values given, write down an expression for the relative error in the numerical derivative.

[5 marks]

- (b) In a guest lecture given during this course it was stated that “Physics drove computing capability in the last 20 years, biology will do the same in the next 20 years”. Discuss this quote. **[3 marks]**
- (c) What is the difference between subroutine profiling and basic block profiling? How would you use each? **[3 marks]**
- (d) Give two examples of events you might seek to measure using hardware performance counters. **[1 mark]**
- (e) Out-of-order execution is considered to be one of the key features of post-RISC architectures. What is out-of-order execution and how does it relate to register renaming? **[2 marks]**
- (f) Write a few lines of code that demonstrate a cache unfriendly operation. Include in your answer a detailed explanation as to why your code is cache unfriendly. **[2 marks]**
- (g) Most compilers translate code into an intermediate representation (IR) that is then subjected to optimisation and eventual code generation. Usually the IR incorporates the core features of the RISC architecture.
- Give two core RISC features that the IR might try to model.
 - Illustrate your answer with small bits of C code and their corresponding IR.
- [4 marks]**
- (h) Give two reasons why compilers unroll loops. **[2 marks]**
- (i) What factors have contributed to the success of Beowulf clusters? **[2 marks]**
- (j) What is meant by i) control dependency and ii) data dependency. Give a few lines of code to illustrate each. **[4 marks]**
- (k) What are array operations in Fortran 95? Do you consider them to be a useful feature for computational science applications? Justify your answer. **[2 marks]**

Question 2 [20 marks]

- (a) Explain the difference between a process and a thread. Is OpenMP based on threads or processes? [2 marks]
- (b) The OpenMP programming model is often referred to as a *fork/join* model and considered to be well suited to *fine grain parallelism*. What is meant by a “fork/join model” and “fine grain parallelism”? [2 marks]
- (c) To evaluate the expression:

$$x = 2.785 \times \prod_{i=1}^N \frac{(3i - 1)}{(3i + 1)}$$

you have written the following OpenMP code:

```
#include <stdio.h>
#include <stdlib.h>
#include <omp.h>
int main(int argc, char* argv[])
{
    double x;
    int nterms,ncpus,istart,iend,nthrd,iam;
    nterms = atoi(argv[1]);
    ncpus = atoi(argv[2]);
    if (nterms <= 0 || ncpus <= 0)return -1;

    omp_set_num_threads(ncpus);
    x=2.785;
#pragma omp parallel default(none) firstprivate(nterms) \
    reduction(*:x) private(iam,nthrd) shared(istart,iend)
    {
        nthrd=omp_get_num_threads();
        iam=omp_get_thread_num();
        istart = iam*(nterms/nthrd)+1;
        iend = (iam+1)*(nterms/nthrd);
        do {
            x*=(3.0*istart-1.0)/(3.0*istart+1.0);
#pragma omp atomic
            istart++;
        } while (istart <= nterms);
    }
    printf(" Number of Terms = %d Value = %20.12e\n",nterms,x);
    return 0;
}
```

The code compiles and runs correctly with `ncpus = 1`, but gives wrong results when `ncpus > 1`.

- (i) Detail as many errors as possible with the above code. (You will get negative marks for giving false errors).
- (ii) Modify the code so that it works correctly with `ncpus > 1`, while also partitioning work across the available CPUs. Your revised code **MUST** maintain the use of the `do/while` construct.
- (iii) Comment on the likely scalability of your code as a function of both input parameters, ie. `nterms` and `ncpus`.

[8 marks]

(d) All modern shared memory parallel computers are cache coherent

- What is cache coherency?
- How might cache coherency impact on the performance of the following parallel loop:

```
#pragma omp parallel for(static,1)
for (i=0;i<N;i++)a[i] = a[i] + b[i];
```

- Suggest how this loop might be modified to give better cache behaviour.

[6 marks]

(e) Distributed memory parallel computers are often programmed using message passing. Outline two fundamental differences between the message passing paradigm and OpenMP. [2 marks]

Question 3 [20 marks]

The following code to performs a matrix vector product:

```
void matvec(int m, int n, double x[], double a[],
            double b[]){
    int i,j;
    for (i=0; i<n; i++){
        for (j=0; j<m; j++){
            x[i]+=a[i*m+j]*b[j];
        }
    }
}
```

For the purpose of this question the code is run on a 500MHz UltraSPARC II processor and you should assume the following:

- A group of up to 4 instructions can be issued per cycle, with (i) up to 2 integer instructions, (ii) up to 1 load or store instruction, (iii) up to 2 *different* floating point instructions and (iv) up to 1 branch instruction.
 - The latency of a double precision load is two cycles, and store operations take two cycles to complete. The latency of a floating point instruction is 3 cycles and all integer instructions take at most 1 cycle.
 - There is no aliasing between the function arguments.
- (a) Is the above code written using a *dot product* or *daxpy* formulation? Write code to illustrate the ALTERNATIVE formulation. **[2 marks]**
- (b) What is the minimum cycle time required to complete the floating point operations associated with just one iteration of the inner j loop. Assume no loop unrolling and ignore all integer operations associated with the loop index. Give the cycle time from issuing the first floating point load until completion of the final floating point store. Based on this time what MFLOP rate will this loop achieve? Show exactly how you derive your answers. **[5 marks]**
- (c) The routine is compiled with Sun compilers using the `-fast` and `-xrestrict` options; `-fast` sets a high level of compiler optimisation and `-xrestrict` states that there is no aliasing between function arguments. Inspection of the assembly shows that the compiler has unrolled the inner loop. What is the maximum possible floating point performance than can be achieved by unrolling just the inner loop AND how does use of `-xrestrict` relate to loop unrolling? **[3 marks]**
- (d) Write C-code showing how you would perform 4-way unrolling of the outer i loop. Is manual unrolling of this loop index likely to be beneficial? Justify your answer. **[5 marks]**
- (e) The dimension of the matrix vector product is determined by the input parameters m and n . These can range anywhere between 1 and 10^6 . Discuss the issues that must be considered in optimizing the performance of this routine for this wide range of possible input dimensions. **[5 marks]**

Question 4 [20 marks]

- (a) As an example of an N-Body simulation method this course has considered molecular dynamics. Briefly outline what molecular dynamics is and give TWO other examples of computational applications that use dynamical simulation. [3 marks]
- (b) You are writing a N-Body simulation code for bodies interacting via the following potential:

$$V_{ij} = \left(\frac{\sigma_i \sigma_j}{r_{ij}^{12}} - \frac{\lambda_i \lambda_j}{r_{ij}^6} \right)$$

where r_{ij} is the distance separating the two bodies i and j , and σ_i and λ_i are constants dependent only on body i . The simulation is in Cartesian space with the distance defined in terms of the x , y and z Cartesian components as:

$$r_{ij}^2 = (x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2$$

The total interaction potential is given as:

$$V_{total} = \sum_{i < j} V_{ij}$$

- (i) Sketch C code to illustrate how you would efficiently evaluate V_{total}
- (ii) How does the computational cost of your code vary with the number of bodies involved (N)?
- (iii) Do you expect the performance of your code to be limited by the floating point performance of your computer or by memory traffic. Give detailed justification for your answer.
- (iv) Discuss how your potential behaves as a function of r_{ij} and how this might be used to reduce the computational cost of evaluating V_{total} .
- (v) Would it be worth parallelising your routine on a shared memory parallel computer? Justify your answer.

[11 marks]

- (c) An essential parameter for all dynamics methods is the *timestep*. What is the timestep, what has to be considered in choosing a timestep, and what happens if too large or small a timestep is chosen? [2 marks]
- (d) N-Body simulation methods often include *periodic boundary conditions*. What are “periodic boundary conditions” and why are they used? [2 marks]
- (e) The work of Barnes and Hutt and a method known as Ewald summation are two approaches that have been used to reduce the scaling of N-Body methods that include a potential of the form $1/r_{ij}$. For ONE of these methods outline what you know about it, eg. the philosophy behind the approach, what it is applied to, etc. [2 marks]

COMP3320 STUDENTS ONLY

Question 5 [10 marks]

(a) Java for computational science:

- (i) From a computational science perspective give two potential benefits AND two current limitations of Java.
- (ii) When computational scientists made early attempts to use Java they found it to be very slow. Why was this? What recent advances have meant that the performance difference between native C or Fortran code and the equivalent Java code is no longer as great?

[4 marks]

(b) Vector processing:

- (i) What is a vector operation and what are the key components of a vector processor.
- (ii) General purpose vector processors are no longer common, but where might you find a specialised vector processor?
- (iii) How does Amdahls law apply to vector processing?

[6 marks]

COMP6464 STUDENTS ONLY

Question 6 [10 marks]

(a) Memory Bandwidth:

- (i) Define memory bandwidth and give one example of an application for which memory bandwidth is important.
- (ii) *Stream* is a widely used benchmark for measuring memory bandwidth. Discuss the issues involved in designing a code to measure memory bandwidth.

[5 marks]

(b) Memory Latency:

- (i) Define memory latency and give one example of an application for which memory latency is important.
- (ii) How does memory latency differ from memory cycle time?
- (iii) Can you suggest how you might design a benchmark to measure memory latency?

[5 marks]
