The questions are followed by labelled, framed blank panels into which your answers MUST be written. Extra boxes are given at the end of the paper. No additional paper will be provided. Writing outside of the boxes may not be marked.

The following are for use by your friendly examiner!

<table>
<thead>
<tr>
<th>Q1 Mark</th>
<th>Q2 Mark</th>
<th>Q3 Mark</th>
<th>Q4 Mark</th>
<th>Q5 Mark</th>
<th>Total Mark</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

This exam is marked out of 100. You should answer all questions.

Your mark for this exam will contribute at most 15% of your total course mark, according to the marking scheme given on the course web page.
Question 1 [20 marks] Performance Measurement

(a) In the standard distributions of Linux/Unix the following two library calls:

```
#include <sys/times.h>
clock_t times(struct tms *buf);

#include <sys/time.h>
int gettimeofday(struct timeval *tv, struct timezone *tz);
```

are used to obtain the “process time” and the “wall clock time” respectively.

(i) What is meant by “process time” and “wall clock time”?

(ii) In timing your application code, when would you use process time and when would you use wall clock time?

(iii) The process time is usually broken down into “user” and “system” components. What is included in each of these components?
(b) When using a timer to measure the performance of your application code, it is important to know the “resolution” and “overhead” associated with that timer.

(i) Define timer resolution and timer overhead.

(ii) Timers that report process time typically have a lower resolution and a higher overhead than timers that report wall clock time. Explain why this is and provide typical values for the resolution and overhead associated with each type of timer.

[10 marks]
Question 2 [20 marks] Timing

The following C routine compiles and runs without error.

```c
void mycode(double a[], double b[], double c[], int N){
    int i, j;
    int count;

    count=0;
    for (i=0; i<N; i=i+1){
        a[i] = 0.0;
        count++;
    }
    printf("end loop 1 \%d\n",count);

    count=0;
    for (i=0; i<N; i=i+1){
        for (j=0; j<N; j=j+1){
            a[i] = a[i] + b[j];
            count++;
        }
    }
    printf("end loop 2 \%d\n",count);

    count=0;
    for (i=0; i<N; i=i+1){
        for (j=1; j<N*N; j=j*2){
            a[i] = a[i] + c[i+j]*2.0;
            count++;
        }
    }
    printf("end loop 3 \%d\n",count);

    return;
}
```

(a) We are interested in the scaling of each loop.

(i) If \( N \) has a value of 10, what is the value of \( \text{count} \) printed out by each \text{printf} statement (3 values)?

(ii) Write general expressions for the scaling of each loop as a function of \( N \) (3 expressions).
(b) Assuming that the code is executed exactly as written (no compiler optimisation) complete the following table as a function of $N$:

<table>
<thead>
<tr>
<th>Loop</th>
<th>Total load/store operations on</th>
<th>Total floating point operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Loop 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Loop 2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Loop 3</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

[6 marks]

(c) If there is a 1 MB data cache, how many of the above load/store requests would you expect to be serviced from cache? Consider both small and large values for $N$.

[4 marks]

(d) In the above code all lines containing the variable `count` are deleted (including the `printf` statements). Detail how you would re-write this version of the code so that it produces the same overall result but with less computational effort. You can change the loop structure in any way and use new temporary storage, but the effect of executing the routine must remain the same.

[4 marks]
Question 3 [20 marks] Floating Point Computation

(a) Assume a decimal (base 10) floating-point number system having machine precision $\epsilon_{\text{mach}} = 10^{-5}$ and an exponent range of $\pm 20$. In this number system the value $1.23456789$ cannot be exactly represented since $\epsilon_{\text{mach}} = 10^{-5}$. In a similar manner provide specific examples and explanations to support each of the following:

(i) An addition operation where there will be a rounding error.

(ii) An addition operation where there will NOT be a rounding error.

(iii) An operation involving two or more valid floating point numbers that results in an underflow.

(iv) An operation involving two or more valid floating point numbers that results in an overflow.

(v) An operation involving two or more valid floating point numbers that results in a significant cancellation error.
Your assignment has involved writing a basic molecular dynamics code.

(i) In your molecular dynamics code truncation errors can become evident when using a large timestep. Why is this the case and how is this evident in the printed and/or visual output of your simulation code?

(ii) In your molecular dynamics code rounding errors become more evident after executing many timesteps. Why is this the case and how is this evident in the printed and/or visual output of your simulation code?
Question 4 [20 marks] Memory Architecture

(a) Sketch C code that illustrates i) a cache friendly operation ii) a cache unfriendly operation. In both cases provide a brief explanation as to why you consider the code to be cache friendly/unfriendly.

(b) Provide two fundamentally different reasons why a data item previously referenced and brought into cache may no longer be present in cache when referenced a second time.
(c) Consider a machine that supports byte-addressable virtual memory and uses 32-bit addressing.

(i) In virtual memory systems what is the purpose of the “page table”?

(ii) If there are 4KB pages, what will be the largest number of entries in the page table?

(iii) Give one advantage and one disadvantage of using 4MB pages rather than 4KB pages.

(iv) The Translation Lookaside Buffer (TLB) is a cache that stores page table entries. How many entries does this special purpose cache typically hold?

(v) On a computer system what happens if your code encounters a TLB miss, and what is the performance implication of this?

[10 marks]
Question 5  [20 marks]  CPU Architecture

(a) Modern processors make extensive use of pipelining. Explain the concept of pipelining using an example that is unrelated to computing.

(b) Data dependencies impact on the ability to pipeline.

(i) Give a few lines of code containing a for loop where there is a data dependency between iterations of the loop. Clearly indicate the dependency.

(ii) Explain how a loop can contain a dependency between different iterations of the loop but can still be amenable to pipelining and producing the correct answers. (Hint: consider pipeline length)
Another form of dependency is a control dependency.

(i) Explain with example what a control dependency is.

(ii) In what way do all loop iterations involve a control dependency?

(iii) Why do control dependencies impact on pipelining, and how does modern hardware attempt to limit this impact?

[8 marks]