Question 1  [20 marks]

Short Questions

(a) Define the term machine precision. [2 marks]

(b) Give two fundamentally different examples of computational science applications that require high performance computing resources. [2 marks]

(c) Explain cancellation error. Include in your explanation a specific example. [2 marks]

(d) Compilers usually translate a high level language into an intermediate level representation that has features that resemble the operation of the target hardware architecture. Give two examples of hardware features that you might expect to be reflected in the intermediate level representation. [2 marks]

(e) In the context of computer architecture, what is “out-of-order execution”? Give one argument in favour of including out-of-order execution in a new processor design and one argument against. [3 marks]

(f) Most modern processors are superscalar with multiple functional units that can execute simultaneously. Name four different functional units that you might expect to find on a modern superscalar processor. [4 marks]

(g) Intel uses a “Tick-Tock” strategy for processor development. Explain how this strategy operates. [2 marks]

(h) In the context of parallel programming define what is meant by i) control dependency, ii) data dependency and iii) output dependency. [3 marks]
Question 2  [20 marks]

Performance Modeling and Measurement

(a) You are using a dual-socket shared memory computer. Each socket contains a four-core Intel processor. Each core can execute vector SSE instructions.

You are trying to develop a performance model for a single-precision (32 bit) floating-point intensive code running on this system. Parts of the code are sequential executing on only one core, other parts run on one core but have been vectorized using 128 bit wide SSE2 instructions, while other parts have been both vectorized and parallelized to run over multiple cores.

Associated with each of these three parts of the code you define three computation rates, \( R_1, R_2 \) and \( R_3 \). You denote the fraction of work that can be executed at rate \( R_i \) as \( f_i \), and \( f_1 + f_2 + f_3 = 1 \).

(i) Write an expression for the average computation rate (\( R_a \)) achieved by the code in terms of the three different \( R \) and \( f \) values. \([4 \text{ marks}]\)

(ii) The three floating-point execution rates are 300, 800 and 5200 Mflops for execution on a single core, a single core using SSE2, and on multiple cores respectively. Comment on the ratio of these values and the relative efficiency of the vectorized and parallelized portions of your code. \([4 \text{ marks}]\)

(iii) When running the code for a particular input data set you estimate that \( f_1 = 0.3, f_2 = 0.2 \) and \( f_3 = 0.5 \). Using the values of \( R_i \) given above how much faster is this application when run over the entire system, compared to when it is run entirely on a single core without SSE vectorization? Provide detailed computations to support your answer. \([4 \text{ marks}]\)

(b) Memory bandwidth and memory latency can significantly impact application code performance

(i) Define memory bandwidth and memory latency. \([4 \text{ marks}]\)

(ii) Explain how the complex cache memory hierarchy of a modern processor helps to reduce the impact of memory bandwidth and latency limitations on overall application performance. \([4 \text{ marks}]\)
Question 3 [20 marks]

Vectorization and SSE

(a) In the context of computer architecture, define what is meant by a vector operation? [2 marks]

(b) 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]

(c) Pipelining and vectorization are similar but not identical concepts. Provide one example of the similarity between these two concepts and one example of their difference. [2 marks]

(d) The following operation:

\[
\text{for (} i = 0; \ i < \text{size}; \ i++) \ c[i] = a[i] + b[i];
\]

is implemented using SSE2 instructions in the following routine:

```c
void VectorAddSSE( float *a, float *b, float *c, 
    size_t size){
    for (size_t i = 0; i < (size/4) * 4; i+=4){
        _m128 sse_a = _mm_load_ps(&a[i]);
        _m128 sse_b = _mm_load_ps(&b[i]);
        _m128 sse_c = _mm_add_ps(sse_a, sse_b);
        _mm_store_ps(&c[i], sse_c);
    }
    for (size_t i = (size/4) * 4; i < size; i++){
        c[i] = a[i] + b[i];
    }
}
```

Provide a detailed explanation of the operations being performed in this routine. [5 marks]

(e) The following code computes a dot product.

\[
\text{result}=0.0
\]

\[
\text{for (} i = 0; \ i < \text{size}; \ i++) \\text{result} = \text{result} + a[i] \times b[i];
\]

Outline how you would write a routine to implement this operation using SSE instructions. You are not required to reproduce the SSE syntax exactly, but you must make your intention clear. Assume 4-byte Float data types. [5 marks]
**Question 4 [20 marks]**

*Shared Memory Programming with OpenMP*

The following C routine was found via profiling to dominate the computation time in a larger program. Your colleague has attempted to parallelize the routine using OpenMP. (For the purpose of this question you should assume that arrays \(f, d\) and \(val\) correspond to distinct memory regions, i.e. they are not aliased.)

```c
void mystery(double f[], double d[], double val[], int n)
{
    int i,j,k,l,ij,kl,ixf;
    ij=0;
    for (i=0; i<n; i++)
    {
        for (j=0; j<i; j++)
        {
            ij++;
            f[i*(i-1)/2+j]=0.0;
            kl=0;
            #pragma omp parallel default(shared)
            for (k=0; k<n; k++)
            {
                #pragma omp for schedule(static,10)
                for (l=0; l<k; l++)
                {
                    kl++;
                    ixf=i*(i-1)/2+j;
                    f[ixf] += d[k*(k-1)/2+l]*val[ij*(ij-1)/2+kl];
                    /* question requests values of ij, kl and ixf if printed here */
                }
            }
        }
    }

    return;
}
```

(a) What would be the values of \(ij\), \(kl\) and \(ixf\) at the point indicated by the comment line above when \(i=3\), \(j=2\), \(k=7\) and \(l=4\)? (Your answer should be three integer values.) [3 marks]

(b) Exactly how many unique elements of arrays \(f\), \(d\) and \(val\) are accessed? (Your answer should be three expressions written in terms of \(n\).) [3 marks]

(c) Explain the precise meaning of each of the OpenMP pragma directives. [4 marks]

(d) The code when compiled with OpenMP enabled and run in parallel gives incorrect results. Explain why this is the case. (There are multiple errors.) [4 marks]

(e) Rewrite the routine so that it runs correctly in parallel using OpenMP and with high efficiency. You are free to change the code as you like, adding additional local variables if needed, The effect on arrays \(f\), \(d\) and \(val\) of calling routine mystery must remain the same. You are not required to have totally correct C/OpenMP syntax, but must make your intention clear. [6 marks]
Question 5 [20 marks]

Application of High Performance Computing

One of the simplest operations in digital image processing is correlation. This operation computes the two-dimensional cross-correlation of two input matrices. The first matrix is an image where each element is a pixel. The second matrix, which is normally much smaller than the image, is a mask with values that correspond to weights. The correlation operation involves overlaying the mask on top of each pixel in the image and using it to compute a new pixel value that is a weighted sum of the surrounding “covered” pixels.

Consider an input image (I) and mask (M) defined as follows:

\[
I = \begin{bmatrix}
17 & 14 & 18 & 4 & 7 \\
10 & 41 & 29 & 3 & 38 \\
12 & 29 & 38 & 26 & 23 \\
31 & 27 & 19 & 39 & 0 \\
24 & 46 & 28 & 47 & 19
\end{bmatrix} \quad M = \begin{bmatrix}
1 & 2 & 1 \\
0 & 0 & 0 \\
-1 & -2 & -1
\end{bmatrix}
\]

When the mask is positioned over pixel I(3,2)

the new pixel value of \(I_{new}(3,2)\) after the correlation is: \(1*10 + 2*41 + 1*29 + 0*12 + 0*29 + 0*38 + (-1)*31 + (-2)*27 + (-1)*19 = 17\).

Suppose that we use the same mask as above but for an image size of 1,024*1,024 pixels. The correlation operation (without processing the image boundary pixels) is performed using the following code (assume \texttt{Inew} is initialized to zero):

\begin{verbatim}
for(i=1; i<1023; i++)
    for(j=1; j<1023; j++)
        for(x=0; x<3; x++)
            for(y=0; y<3; y++){
                Inew[j][i] = Inew[j][i] + I[j+x-1][i+y-1]*M[x][y];
            }
\end{verbatim}

Both the image and the mask are of type double (8 byte). The image data and the mask data are stored in row major form in contiguous memory locations.
Answer the following questions:

(a) How many times is each element of the $I$ and $M$ arrays accessed? Note special cases. [4 marks]

(b) The above code is executed on a computer with a 16KB level 1 cache that has a 32 byte line size. Comment on the level 1 cache usage. For example you might want to consider the percentage of the $I_{\text{new}}$, $I$ and $M$ array accesses that are likely to be serviced from level 1 cache. In your discussion detail any assumptions you make. [5 marks]

(c) Rewrite the above code to make it more efficient. Consider both possibilities for reducing the complexity of the algorithm as well as making it more cache friendly. [5 marks]

(d) Your colleague suggests that to improve execution time further you might i) use 4 byte data types instead of 8 byte, ii) exploit streaming SIMD extensions, iii) parallelize the code to run over the six cores that are present on your computer. Discuss the merits of each of these options in turn. Include in your discussion comments on the ease of implementing each suggestions and its likely performance gain. [6 marks]