1. (20 points) Consider the following pseudo-code:

   for k = 1,N
   for j = 1,k
   for i = 1,N
      C(j,k) = C(j,k) + A(i,k)*B(i,j)
   end for
   end for
   end for

(a) Show a 2-way k-loop unrolled form (assume N is even).

(b) Would any of the other permutations of the nested loops be preferable for this computation? If so, which would be best (show pseudo-code)? Assume row-major storage of arrays.

2. (30 points) Consider the following 2D and 3D forms of tiling for matrix-matrix product.

   for it = 1,1024,512
   for jt = 1,1024,512
   for k = 1,4096
      for i = it,it+511
         for j = jt,jt+511
            C(i,j) = C(i,j) + A(i,k)*B(k,j)
         end for
      end for
   end for
   end for

   for it = 1,1024,256
   for jt = 1,1024,256
   for k = 1,1024
      for i = it,it+255
         for j = jt,jt+255
            C(i,j) = C(i,j) + A(i,k)*B(k,j)
         end for
      end for
   end for
   end for

2D Tiling

   for it = 1,1024,512
   for jt = 1,1024,512
   for k = 1,4096,256
      for i = it,it+511
         for j = jt,jt+511
            C(i,j) = C(i,j) + A(i,k)*B(k,j)
         end for
      end for
   end for
   end for

3D Tiling

Assuming a fully associative cache with linesize 8 words and capacity of 512 Kwords, which of the above tiled forms is preferable? Assume row-major storage of arrays.

3. (30 points) A first-order model for communication cost for point-to-point messages in a distributed-memory system is

   \[ t = t_s + l \cdot t_w, \]

   where \( t_s \) is the start-up cost, \( t_w \) is the per-word transmission cost, and \( l \) is the length of the message in words. Consider repeated matrix-vector multiplication similar to that implemented in Assignment 4:

   Initialize each component of x to 1.0
   Initialize A[i,j] to ((i+j) mod N)/(N*(N-1)/2)
   start timer
   repeat k times
   \{ 
      y[1:N] = Ax[1:N]
\[ x[1:N] = Ay[1:N] \]

\textit{stop timer}

\textit{output min and max components of x}

Assume that each arithmetic operation takes 1 unit of time as long as the operands are in register or cache, and that \( t_s = 10,000 \) time units, and \( t_w = 10 \) time units. The cache capacity is 256 Kwords and linesize is 8 words. Main memory access time is 32 time units. Predict the execution time and achieved speed-up (relative to sequential execution) on 2, 4, 8, and 16 processors with a row-block partitioning of \( A \) (i.e. \( N/P \) rows of \( A \) are mapped to each processor) and a dot-product form of the matrix-vector multiply algorithm, for \( N=1024 \). Assume that \( k \) is large and that there is negligible cache interference between \( x, y, \) and \( A \). A P-1 step ring algorithm with point-to-point communication is used for the all-to-all-broadcast.

Assume that the processors are connected by a cross-bar switch (i.e. it is possible for each processor to be simultaneously communicating, as long as no processor is the destination of more than one simultaneous message). Assume that all links are bidirectional and that sending and receiving at a processor can be concurrent, but that two sends from the same processor are completely serialized and cannot have any overlap.

4. (20 points) Consider the loops (L1 and L2) for Dot-product forms of lower-triangular solve and matrix-vector multiplication:

\begin{verbatim}
for (i=0;i<1024;i++) for (i=0;i<1024;i++)
{ b[i] = 0;
  for (j=0;j<i;j++) for (j=0;j<1024;j++)
    b[i] = b[i] - a[i][j]*x[j]; b[i] = b[i] + a[i][j]*x[j];
  x[i] = b[i]/a[i][i];
}
\end{verbatim}

\begin{verbatim}
L1: Lower-triangular Solve  L2: matrix-vector multiply
\end{verbatim}

(a) Show how you would parallelize each of them using work-sharing constructs in OpenMP.

(b) Assuming that each arithmetic operation takes one unit of time and scheduling the iterations of a work-sharing construct takes 100 units of time, estimate the speedup you would expect on 4 processors for each code. Assume loop counter incrementing and testing takes negligible amount of time. For simplifying the estimation of speedup of the lower-triangular solve (but not for the actual code), consider the upper-loop bound for \( j \) to be \( N \) instead of \( i \).