Problem 1. (10 points) Write the code for an average_columns function for the data-parallel model of lecture 8.

The function should divide up the data so that each processor operates on columns of data. The average of each column of data should be written into the corresponding element of the 1-dimensional array column_averages_array. The calling function is already written for you.

```c
int nprocs;
int row_count;
int col_count;

double **A;
double *column_averages_array;

procedure main()
begin
    read(nprocs);
    read(row_count);
    read(col_count);
    A <= G_MALLOC( row_count * col_count );
    column_averages_array <= G_MALLOC( col_count );
    read(A);
    average_columns(A, column_averages_array, row_count, col_count);
    print_1d_array( "Column Averages:\n", column_averages_array );
end

procedure average_columns(A, col_avgs, row_count, col_count)
double **A, double* col_avgs, int row_count, int col_count;
begin
*** your code here. ***
end
```

Problem 2. (25 points) Given a 128 byte cache, which has a block size of 16 bytes, and the following list of memory reference addresses hexadecimal:

0x4F5C2
0x012D5
0x89637
0xBB0A7
0x12045
0x508AB
0xBB0A5
0x4F5CF
0xB453C
0x89633
(a) Label each reference in the list in a table showing the binary address, block number, tag value, tag status before and after referring to the data, and whether each reference was a hit or miss. Then show the final contents of the cache for
   (i) A direct mapped cache
   (ii) A 2-way set associative cache
   (The cache is initially empty in both cases)

(b) Which cache performs better? Why?

**Problem 3. (25 points)** Gaussian elimination is a well-known technique for solving simultaneous linear systems of equations. Variables are eliminated one by one until there is only one left, and then the discovered values of variables are back-substituted to obtain the values of other variables. In practice, the coefficients of the unknowns in the equation system are represented as a matrix $A$, and the matrix is first converted to an upper-triangular matrix (a matrix in which all elements below the main diagonal are 0). Then back-substitution is used. Let us focus on the conversion to an upper-triangular matrix by successive variable elimination. Pseudocode for sequential Gaussian elimination is shown below. The diagonal element for a particular iteration of the $k$ loop is called the **pivot element**, and its row is called the **pivot row**.

\[
\text{procedure Eliminate (}A\text{)} \quad /*\text{triangularize the matrix } A**/
\begin{align*}
& \text{begin} \\
& \text{for } k \leq 0 \text{ to } n-1 \text{ do } /*\text{loop over all diagonal(pivot)elements}*/ \\
& \quad \text{begin} \\
& \quad \text{for } j \leq k+1 \text{ to } n-1 \text{ do } /*\text{for all elements in the row of, and to the right of, the pivot element}*/ \\
& \qquad A[k,j] = A[k,j]/A[k,k]; /*\text{divide by pivot element}*/ \\
& \qquad A[k,k] = 1; \\
& \quad \text{for } i \leq k+1 \text{ to } n-1 \text{ do } /*\text{for all rows below the pivot row}*/ \\
& \qquad \text{for } j \leq k+1 \text{ to } n-1 \text{ do } /*\text{for all elements in the row}*/ \\
& \quad \text{endfor} \\
& \quad \text{A[i,k] = 0;} \\
& \text{endfor} \\
& \text{endfor} \\
& \text{end procedure}
\]

(a) Draw a simple figure illustrating the dependences among matrix elements.

(b) Assuming a decomposition into rows and an assignment into blocks of contiguous rows, write a shared address space parallel version using the primitives used for the equation solver in the chapter.

(c) Write a message-passing version for the same decomposition and assignment, first using synchronous message passing and then any form of asynchronous message passing.

(d) Can you see obvious performance problems with this partitioning?

(e) Modify both the shared address space and message-passing versions to use an interleaved assignment of rows to processes.

(f) Discuss the trade-offs (programming difficulty and any likely major performance differences) in programming the shared address space and message-passing versions.

**Problem 4. (20 points)** Consider the following memory hierarchy:
(a) For the given memory hierarchy, what are the hit and miss rates in the cache and the memory if the processor executes a total of 1,000,000 memory references, and 945,000 of which hit in the cache and 45,000 of which hit in the main memory?

(b) If the memory size is 4000MB, and the cache is of size 64KB, and the cache has lines of size 128B each, and is 4-way set-associative, then

(i) The cache contains how many numbers of lines, sets and tags?

(ii) The address used in this system contains total how many bits, and each tag in the cache contains how many bits?

(iii) If the cache is write-through, how many bits are required for each tag if an LRU replacement policy is used? What if the cache is write-back?

(c) The cache has a hit rate of 95% and a cache hit latency of 5 ns. The main memory takes 100 ns to return the first word (32 bits) of the line, and 10 ns to return each subsequent word.

(i) What is the cache miss time for this cache? (Assume that the cache waits until the line has been fetched into the cache and then re-executes the memory operation, resulting in a cache hit. Neglect the time required to write the line into the cache once it has been fetched from the memory. Also, assume that the cache takes the same amount of time to detect a miss has occurred as to handle a cache hit.)

(ii) If doubling the cache line length reduces the miss ratio to 3 percent, does it reduces the average memory access time? (Remember: Average access time for each level in memory hierarchy contains the time taken because of hit + time taken because of miss)

(d) Similar to the 1st bit of the question, the processor executes 1,000,000 memory references. The cache has a miss rate of 7%, of which 1/4 are compulsory misses or cold misses, 1/4 are capacity misses, and 1/2 are conflict misses.

(i) If the only change you are allowed to make to the cache is to increase the associativity, what is the maximum number of misses that you can hope to eliminate?

(ii) If you are allowed to both increase the cache size and increase the associativity, what are the maximum number of misses that you can hope to eliminate?

Problem 5. (20 points) The following table shows the memory operations of three processors. All three processors access two variables (v1, v2) in sequence either for a read or write operation (see table). Complete the table for the scenarios in a) – d) by entering the state of each variable in each processor's cache, the action occurring on the bus, from where the data is provided and if the data is flushed to main memory. Determine also how many bus transactions and data reads over the bus are performed for each cache/protocol combination.

a) a write-through cache with two-state invalidation protocol is used

b) a write-back cache with the MSI protocol is used
c) a write-back cache with the MESI protocol is used 

d) which of the cache/protocol combination provides the best performance for this sequence of read/write operations? Why?

Table of read/write operations:

<table>
<thead>
<tr>
<th>Bus Action</th>
<th>Data by</th>
<th>Data flushed to memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>P2</td>
<td>P3</td>
</tr>
<tr>
<td>v1</td>
<td>v2</td>
<td>v1 v2 v1 v2</td>
</tr>
</tbody>
</table>

P1  Read v1
P2  Read v1
P2  Write v1
P3  Read v2
P2  Write v1
P3  Write v2
P1  Write v1
P1  Read v2
P2  Read v1
P2  Read v2