CSC/ECE 506: Architecture of Parallel Computers
Problem Set 1
Due Thursday, February 7, 2002

Problems 1, 5, and 6 will be graded. There are 60 points on these problems. Note: You must do all the problems, even the non-graded ones. If you do not do some of them, half as many points as they are worth will be subtracted from your score on the graded problems.

Problem 1. (15 points) Page 12 of the lecture notes for Lecture 3 (titled Communication architectures, cont.) presents an assembly-language version of the inner loop of a vector-processor matrix-multiplication program whose pseudocode is given on page 10 of the same lecture. Write the assembly code for the entire program (i.e., the program on page 10), using the code already given on page 12. Please make sure that your program is well commented; however, you do not have to comment portions that you use without change from page 12.

Problem 2. (15 points) Below is a section of pseudocode that subtracts two $N \times N$ matrices, and sets any negative values in the result to zero. Using the vector instructions from Lecture 3 to set the appropriate mask bits, implement the pseudocode. A vector of size $N$ containing all -1’s is loaded at location P, to help perform the subtraction and a vector of size $N$ containing all 0’s is loaded at location Q.

```plaintext
for (i=1 to N)
{
    C[i] = A[i] – B[i] ; /*subtract the entire row*/
    for all C[i] < 0 set C[i] = 0 ;
} /*end for loop*/
```

Problem 3. (15 points) [Culler, Singh, & Gupta 1.6] In message-passing models, each process is provided with a special variable or function that gives its unique number or rank among the set of processes executing the program. Most shared memory programming systems provide a `fetch&inc` (fetch and increment) operation, which reads the value of a location and atomically increments the value at the location.

(a) Write a little pseudocode the show how this `fetch&inc` instruction can be used to assign each process a unique number. Assume that the number of processes is known, and is stored in the variable `number_procs`, and that each process waits till all processes have been assigned a rank before proceeding forward. Comment your pseudocode to clearly indicate the variable that holds the rank, and the variable on which the `fetch&inc` is performed.

(b) If the number of processes is not known, can you determine it in a manner similar to part (a) above?

Problem 4. (15 points) Suppose a program that was being run on one processor is now run on a 100-processor machine. If a speedup of 80 is desired (on the 100-processor machine, as compared to the single processor), what fraction of the program can be serial? Use Amdahl’s law.

Problem 5. (25 points) Suppose we have an MIMD computer system with 16 independent processors accessing a shared memory through an interconnection network. Let the time to perform one add operation on one processor be $T_a$. This means that the system can perform 16 additions in time $T_a$, 32 in time $2T_a$, and so on. In the following parts, ignore the time necessary for memory access, communication, and synchronization.
(a) Plot the time required to perform $n$ additions, where $n$ ranges from 1 to 64. Mark time along the vertical axis, and $n$ along the horizontal. Mark at least 4 points on the graph, and explicitly specify these points in a table of values of $n$ against the corresponding values of time.

(b) Plot the speedup $S_n$ resulting from the use of the 16-processor system as compared with a one-processor system that uses a processor of the same type, for $n$ additions, as in part (a) above. Mark speedup $S_n$ along the vertical axis and $n$ along the horizontal. Mark at least 4 points and explicitly specify these in a table. Of values of $n$ against corresponding values of time and the speedup $S_n$.

(c) Now consider a more specific problem. We want to sum the elements of a vector of length 64. That is, form a single number by adding all the values from a 1-dimensional array with 64 elements. If a binary-tree-structured algorithm is used (that is, by splitting the set of numbers to be added into two disjoint sets of equal size, and adding corresponding elements of each set), how much time is required for the entire addition process? Note that data dependencies have to be taken into consideration for this algorithm.

**Problem 6. (20 points)** [Culler, Singh, & Gupta 1.12] If we consider loading a cache line from memory, the transfer time is the time taken to actually transmit the data across the bus. The startup includes the time to obtain access to the bus, convey the address, access the memory, and possibly place the data in the cache before responding to the processor. However, in a modern processor with dynamic instruction scheduling, the overhead may include only the portion spent in accessing the cache to detect the miss and placing the request on the bus. The memory access portion contributes to the latency, which can potentially be hidden by the overlap with execution of instructions that do not depend on the result of the load.

Suppose we have a machine with a 64-bit wide bus running at 40MHz. It takes two bus cycles to arbitrate for the bus and present the address. The cache line size is 32 bytes and the memory access time is 100ns.

(a) What is the latency for a read miss?

(b) What bandwidth does this transfer obtain?