EECS 587 Homework due September 29, 2015

As always, this homework is due at the start of class.
Throughout this class, you are to do all homework on your own unless told otherwise.

1. Start with the serial program:

```
1   initialize
2   for i = 0 to n-1
3      statement 3
4      for j = 0 to n-1
5         statement 5
6      end for j
7   end for i
8   finalize
```

   where `finalize`, `statement 3` and `statement 5` each take constant time and `initialize` takes $\Theta(n)$ time. Suppose you parallelize for a distributed memory machine with p processors. All processors execute the outer i-loop, and the inner j-loop can be perfectly parallelized, i.e., each processor does n/p iterations and no overhead is added except for the communication mentioned below. Suppose for any given $0 < c < 1$ you want to achieve efficiency c by weak scaling. Given c, what is the fastest that p can increase with n, or is it impossible to achieve arbitrarily high efficiency, if a communication step is added

   (a) between lines 5 and 6, taking $\Theta(1)$ time?

   (b) between lines 6 and 7, taking $\Theta(p)$ time?

   (c) between lines 6 and 7, taking $\Theta(\sqrt{n/p})$ time?

   By the way, each of these occurs in a realistic application.

2. You have an array `A(0:n-1)` of real numbers and want to compute `reduce(sum,A)`, which is $\sum_{j=0}^{n-1} A(i)$. You have a distributed memory system of p processors, with ids $0 \ldots p-1$, where p evenly divides n and each processor starts with n/p numbers.

   (a) The processors are arranged in a 1-dimensional mesh, with processor i connected to $i-1$ and $i+1$, if they exist. Briefly sketch an efficient algorithm to compute the reduction. You only need a few English sentences since you are just outlining an algorithm for a human to understand. You can only use blocking receives and nonblocking sends.

   (b) In O-notation, how fast should `p` grow with `n` to minimize the total time? Assume that standard computer instructions, and sending a single value to a neighbor, take constant time. Note that if there are too few processors then there isn't much advantage in parallelizing, while if there are too many processors then too much time is spent communicating.

   (c) The same question as (b), but now the processors are completely connected. You don't need to write the algorithm.

3. In this problem you will write a program for a distributed memory SIMD hypercube. It has p processors, where p is a power of 2. Each processor starts with p and its processor number pid in local variables, where $0 \leq \mathsf{pid} \leq p-1$. For processor i, the coordinates of i are its value expressed in $\lg(p)$ bits. E.g., if $p = 32$, then processor 6 has coordinates 00110. Two processors are connected if and only if they differ in exactly one coordinate. E.g., 6 is connected to 10110, 01110, 00010, 00100, 00111. Note that this is the same as saying that i and j are connected if and only if $|i - j| = 2^{dim}$, where *dim* is the coordinate they differ by. To help, there is a function bit(i, *dim*) which returns the coordinate (bit) of i in dimension *dim*. E.g., bit(6,0)=0 and bit(6,1)=1.

   The only processor communication function is exchange(dim,x,y), where every processor sends the value of its local variable x to its neighbor along its dim coordinate, and receives the value sent in y. E.g., if initially processor 00 has a x value of 1 and processor 01 has x value of 2, then exchange(0,x,y) results in processor 00 having a y value 2 and processor 01 having y value of 1.

   There is a real array $V(0 : p-1)$, where processor i starts with $V(i)$ stored in a local variable V. Each processor also has a local variable M. At the end, the maximum value in V is stored in every processor's M variable. This is known as all_reduce(max,V).

   In pseudo-code, write an efficient program to compute all_reduce(max,V) and analyze the time required as a function of p. Write the program that the controller would run. For each instruction, indicate whether it would be executed in the controller, or transmitted to the processors, and for every variable, indicate if it is used solely by the controller, or is in all of the processors. The processors have as many scalar local variables as needed.

A 4-dimensional hypercube. Numbers are pids, lines indicate connections.