0907521 Parallel and Distributed Systems (Fall 2013)
Midterm Exam

الاسم: رقم التسلسل: رقم الشعبة: ١

==========

Instructions: Time **70** minutes. Open book and notes exam. No electronics. Please answer all problems in the space provided and limit your answer to the space provided. **No questions are allowed**.

<Good Luck>

Q1. As a programmer, give one technique to achieve each of the following objectives.

A) Efficient utilization of a computer with a multi-threaded CPU.

<2 marks>

Write multi-threaded programs.

B) Reduce capacity and compulsory misses in a serial program.

<2 marks>

Access data in small strides.

C) Reduce misses due to false sharing in a shared-memory parallel program.

<2 marks>

Insert enough space between one shared variable and the next.

D) Improve the speedup of a message-passing programing that has multiple processes that exchange frequent short messages.

<2 marks>

Aggregate small messages in larger ones.

E) Improve the IPC of your program that runs on a dynamically-scheduled superscalar processor.

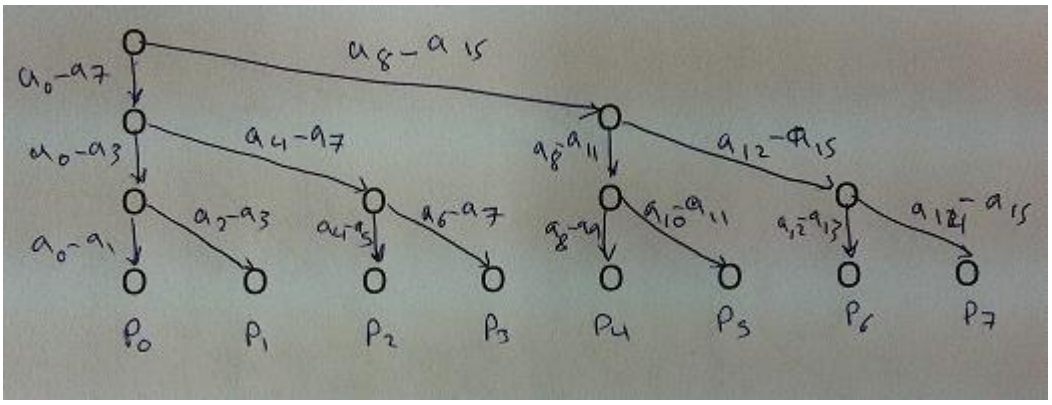
<2 marks>

Merge loops to get many instructions that the processor can schedule together.

Q2. Suppose `comm_sz` = 8 and n = 16.

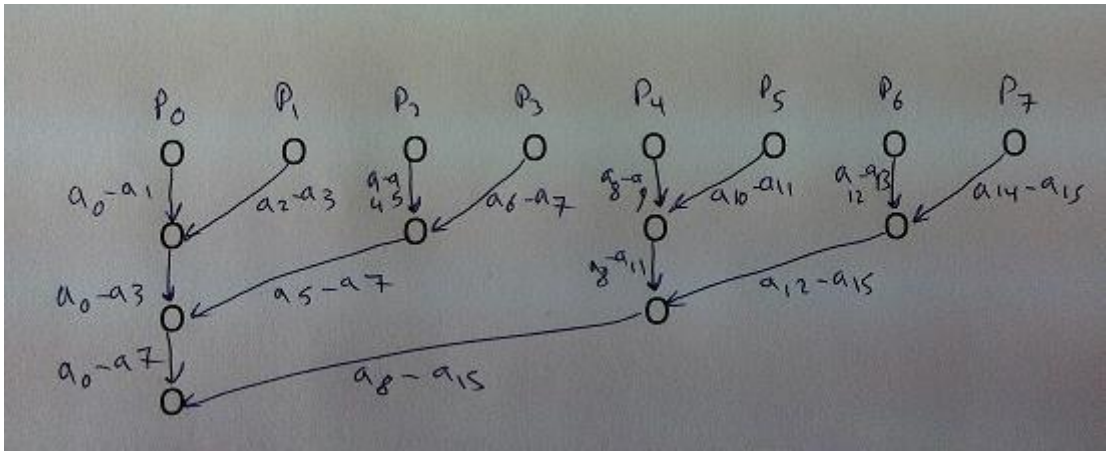
A) Draw a diagram that shows how `MPI_Scatter` can be implemented using tree-structured communication with `comm_sz` processes when process 0 needs to distribute an array containing n elements.

<5 marks>



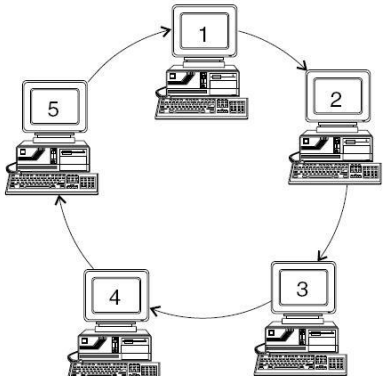
B) Draw a diagram that shows how `MPI_Gather` can be implemented using tree-structured communication when an n-element array that has been distributed among `comm_sz` processes needs to be gathered onto process 0.

<5 marks>



Q3. Write an MPI program where the processes are organized in a virtual ring according to their MPI communication ranks. Each processor should send its rank ID to one of its neighbors and receive the rank ID of the other neighbor. Then for each process, a message originating from that process should be displayed on the standard output device showing the process ID and its neighbor, e.g., "`I am process 1 and my neighbor is 2`".

<10 marks>




```
#include <string.h
#include <mpi.h>

const int MAX_STRING = 100;

int main(void) {
   char       message[MAX_STRING];
   int        comm_sz;
   int        my_rank;
   int        my_partner;

   MPI_Init(NULL, NULL);
   MPI_Comm_size(MPI_COMM_WORLD, &comm_sz);
   MPI_Comm_rank(MPI_COMM_WORLD, &my_rank);

   /* send to the next process */
   MPI_Send(&my_rank, 1, MPI_INT, (my_rank+comm_sz-1)%comm_sz, 0,
         MPI_COMM_WORLD);

   /* receive from the previous process */
   MPI_Recv(&my_partner, 1, MPI_INT, (my_rank+1)%comm_sz,
         0, MPI_COMM_WORLD, MPI_STATUS_IGNORE);

   if (my_rank != 0) {
      /* Create message */
      sprintf(message, "I am process %d and my neighbor is %d",
            my_rank, my_partner);
      /* Send message to process 0 */
      MPI_Send(message, strlen(message)+1, MPI_CHAR, 0, 0,
            MPI_COMM_WORLD);
   } else {
      /* Print my message */
      printf("I am process %d and my neighbor is %d\n",
            my_rank, my_partner);
      for (int q = 1; q < comm_sz; q++) {
         /* Receive message from process q */
         MPI_Recv(message, MAX_STRING, MPI_CHAR, q,
            0, MPI_COMM_WORLD, MPI_STATUS_IGNORE);
         /* Print message from process q */
         printf("%s\n", message);
      }
   }

   MPI_Finalize();

   return 0;
}
```