Chapter 1

1. What is SPMD programming style? Why is it popular?

   Write one program which works for multiple data streams. Each processor executes the same code, but operates on different data based on processor ID.

   It is popular because it simplifies parallel programming. We don’t need to worry about different program codes for different processors.

Chapter 2

1. What is the difference between bisection width and bisection bandwidth?

   - It is the minimum number of links that must be removed to partition the network into two equal halves. To compute bisection bandwidth, we sum bandwidth of links that cut the network into two equal halves and then choose the minimum. The difference between these two concepts is that the bisection width counts each link as 1 and does not consider the bandwidth difference of these links.

2. What is the Amdahl’s law?

   This law shows the limitation of parallel performance. The maximum speedup is limited if part of the serial code cannot be parallelized.

Chapter 2 extra topics (Parallel Task Computation and Communication Operations)

1. Given a summation binary tree with n nodes, what is the length of the critical path? What is the maximum speedup based on the span law and work law? Assume all nodes have equal weights.

   The length of the critical path (longest path) is about log n. The total workload is n for this tree. Given p processors, the shortest parallel time is log n based on the span law. It is n/p based on the work law. Thus speedup = sequential time/parallel time <= min(n/log n, p).

2. Design a broadcast algorithm for a 2D torus architecture and what is the cost of broadcasting a message?

   Let p be the number of processors and this torus is a square shape. Let a+bm be the cost of communication between two neighbor processors where a is the startup cost, b is the transmission speed, and m is the message size. The broadcast algorithm contains two stages.
Stage 1: Perform a row-wise horizontal broadcast within the row of the processor which initiates broadcasting. Such broadcast can be done easily by propagating the message from the center (the source of broadcasting) to its row neighbors linearly one by one. The total cost is about \( \sqrt{p}(a + bm)/2 \).

Stage 2 Perform a vertical broadcasting in each column in parallel. Since each column has a ring topology and the communication algorithm is similar as Stage 1. The total cost is about \( \sqrt{p}(a + bm)/2 \).

Total cost is about \( \sqrt{p}(a + bm) \).

Chapter 3

1. In this program, Process 2 tries to receive a message from Process 1 with 2 numbers and Process 0 with 1 number. What is the output of printf()?

```c
int i, rank; int buf[2]; MPI_Status status;
MPI_Comm_rank(MPI_COMM_WORLD, &rank); buf[0]=buf[1]=rank;

if (rank== 0) {MPI_Send( buf, 1, MPI_INT, 2, 0, MPI_COMM_WORLD); }
else if (rank == 1) { MPI_Send( buf, 2, MPI_INT, 2, 0, MPI_COMM_WORLD); }
else if (rank == 2) {
  for (i= 0; i<2; i++) {
    MPI_Recv( buf, 2, MPI_INT, MPI_ANY_SOURCE, 0, MPI_COMM_WORLD, &status );
    printf( "buf = %d %d\n", buf[0],buf[1]); }
}
```

Two possible cases
1) If Process 0 message is received first, the output is: buf = 0 2
   buf = 1 1

2) If Process 1’s message is received first, the output is
   buf = 1 1
   buf = 0 1

2. With 3 processes running, what is the output of printf()?

```c
int rank; int buf[2];
MPI_Comm_rank(MPI_COMM_WORLD, &rank); buf[0]=buf[1]=rank;

if(rank==0) MPI_Reduce ( buf, &buf[1], 1, MPI_INT, MPI_SUM, 0, MPI_COMM_WORLD);
else MPI_Reduce ( &buf[1], buf, 1, MPI_INT, MPI_SUM, 0, MPI_COMM_WORLD);
printf( “Process %d: buf = %d %d\n", rank, buf[0],buf[1]);
```
Three processes can output the following 3 messages in any order:
Process 0 3
Process 1 1
Process 2 2

Chapter 3 Extra – Program parallelization

1. Derive dependence graph of the following sequential program segment?

   S1: A = B + D
   S2: C = A + D
   S3: E = A + B
   S4: A = C + B

Flow dependence: S1 → S2. S1 → S3. S2 → S4

Anti dependence: S2 → S4. S3 → S4

Output dependence: S1 → S4

3. Illustrate the iteration space and dependence among iteration statements

   For i = 3 to n
   For j = 3 to n
   X[i-1, j-1] = X[i-2, j-1] + 3

4. Write pseudo message-passing SPMD code to parallelize the above 2D loop on a distributed memory architecture using block mapping and illustrate the partitioning and mapping process in your design.

   Data partitioning: we divide 2D array X in a column wise manner and map columns to processors in a block manner. Column-to-processor mapping and column-to-local-address mapping are:

   \[
   \text{Proc\_map(\text{Column } j)} = \text{floor}(j/r) \quad \text{where } r \text{ is the block size and } r = \text{ceiling}(n/p)
   \]
Local(Column $j$) = $j \mod r$

Code partitioning and mapping is arranged consistently as data mapping. SPMD code:

```
me=mynode()
For j = 3 to n
    If(Proc_map(j-1) == me ) {
        For i = 3 to n
            $X[i-1, Local(j-1)] = X[i-2, Local(j-1)] + 3$
    }
```