Name :DuZhi

Q1.a (2 marks). What is the maximum possible speedup possible if the hardware has 20 stages?

20n/(n+19)

Q. 1 b. (1 marks) True/False. Multithreading solves bw problem but exacerbates latency problem

False  
Q.1 c. (1 marks) True/False. EREW model is the strongest PRAM architecture.

False  
Q. 2 (2 marks). Following MPI program with 2 processes has deadlock. State the reason of the deadlock.  
The arguments are in this order : memory address, tag, process\_id  
P0 P1  
receive(&a, 1, 1); receive(&a, 1, 0);  
send(&b, 1, 1); send(&b, 1, 0);

Answer:Cause p0 and p1 both need use &a to continue the result, when p0 get the memory address ,p1 needs to wait until the address is released.But p0 needs p0 to finish the “send” function.They both were waiting,so it’s gonna has deadlock.

Q3 (2 marks). For the following block of code running on two processors P1 and P2  
if(A == 4)  
printf(“hi”);  
else  
printf(“bye”);  
Describe how will the code block execution by P1 and P2 vary in SIMD and MIMD architecture ?

ANSWER: SIMD judge if A is 4,then output hi or bye, MIMD print “bye” and judge if A is 4“hi”,if it is, output hi.

Q4. (3 marks). Prove that the diameter of a hypercube is O(log p) where p is the number of processors.

Answer: The diameter of a full-binary-tree network with Diameter D = q (**p**1/q – 1) = log2 **p** \* = **O**(**log p**)

Q5. (4 marks). Given a collection of line segments as input and a function bool  
isIntersecting(line\_segment seg1, line\_segment seg2) which returns true/false based on  
whether seg1 and seg2 intersects. Here line\_segment contains start vertex and end\_vertex.  
Provide a parallel algorithm and state its time complexity for counting the number of segment  
intersections by p processes.

Answer :Suppose there have n lines. So we need to count n\*(n-1)/2 = Olog(n\*n) possibility for segment.

isIntersecting(line\_segment seg(1), line\_segment seg2)

true/false

Islntersecting(line\_segment seg1,line\_segment seg3)

True/fause

......

Islntersecting(line\_segment seg 1,line\_segment seg n)

True/fause

Islntersecting(line\_segment seg 2,line\_segment seg 3)

True/fause

...

Islntersecting(line\_segment seg 2,line\_segment seg n)

True/fause

...

Islntersecting(line\_segment seg n-1,line\_segment seg n)

True/fause  
​ Q6) (2 marks) Given below is ​ a function ​insertElement to insert a value to a shared array ​arr.  
This function does not work correctly in a multi-threaded environment. State potential bugs and  
a method to make the code thread-safe.

|  |  |
| --- | --- |
| #define MAX 1000 int arr[MAX]; int position = 0; int main()  { insertElement(10); return 0; } | void insertElement(int info)  {  Lock(&position) if(position < MAX) { arr[position] = info; position++; } else { printf(“arr is full”); }  UNCLOCK(&position) } |

### Answer :It might has a Race condition problem in a multi-threaded environment, because in the circle,might execute before .

Use lock to make sure each process operation orderly.

Q7.(6 marks) Write a pseudo-code (algorithm) using MPI to add the elements of an ​m ​ by ​n  
matrix with the help of ​p ​ processes. Each process gets matrix elements in a cyclic fashion as  
described below.  
Here is an example ​: A input matrix with each cell having a number to be divided among 3  
processes as shown:  
Processor P0 should add ​red ​ cells, P1 should add white cells and P2 should add ​blue ​cells.  
P0 adds the red cells locally = 1 + 4 + 7 + 10 + 13 + 16 = 51  
P1 adds the white cells locally = 2 + 5 + .. + 14 = 40  
P2 adds the blue cells locally = 3 + 6 + .. + 15 = 45  
The final output should be the summation of the local sums 51, 40 and 45 to yield 136.  
Assume:  
1<p<8  
row<10  
col<10

ANSWER:

numThreads = number of threads

threadId = unique id of each thread

arr[N] = array of N numbers

Int sum=0;

int localAdd(int threadId, int numThreads, int arr[], int N)

{

int start\_index = (threadId \*N) / numThreads

int end\_index = ((threadId +1)\*N) / numThreads

for(i = int start\_index; i<end\_index; i++)

  local\_sum = local\_sum + arr[i]

return local\_sum;

sum= sum + local\_sum;

}

Printf(“%d”,sum);

Return 0;

Q8. Book question (chapter 2, Q 2.16)  
If each processor has 1 element, provide a parallel algorithm to find the minimum element.  
More info:  
When all switches are closed, all n^2 processors are connected by the bus.  
If all processors disconnect the switch on their north, then we obtain row buses.  
Column buses are obtained by having each processor disconnect the switch on its east.

ANSWER:

Partitioning the mesh into two equal parts of p/2 processors . Thus bisection width = sqrt(p). By configuring the mesh appropriately, the distance between any two processors can be made to be independent of the number of processors. Therefore, the diameter of the network is O(1). Each processor has a reconfigurable set of switches associated with it. we see that each processor has six switches. Therefore, the total number of switching elements is 6p. The number of communication links is identical to that of a regular two-dimensional mesh, and is given by 2(p − √p).

Q9. LU decomposition : How to parallelize this problem ?  
Identify the tasks that could be executed in parallel and the task dependencies. Also, show how to efficiently map these tasks to processes.  
LU decomposition ​ (where 'LU' stands for 'lower upper') decomposes a ​matrix ​ as the product of a  
lower ​triangular matrix ​ and an upper triangular matrix.  
A = LU  
Attachment in d2l.mu.edu

Answer:

