1. **What are the two main styles of parallelism? Explain**.

Parallelismis the demonstration of performing at least two activities all the while. Parallelism emerges at each degree of a cutting edge PC framework. There are two main styles of parallelism they are Instruction level parallelism and task parallelism, explained as follows.

***Instruction-level parallelism*: A processor can just address short of one instruction for each clock cycle stage. These instructions can be re-requested and assembled which are later on executed simultaneously without influencing the after effect of the program. This is called instruction level parallelism**

***Task Parallelism:* It divides the tasks into subtasks, then allocates each of task for execution and finally processors concurrently performs subtask execution.**

1. **What are the two main types of locality? Explain.**

Locality is the aptness of a processor to access the same set of memory locations tediously over a brief timeframe. There are two types of reference locality, they are temporal locality and spatial locality, explained as follows.

***Temporal Locality:*** The idea that an resource that is referenced at one point in time will be referenced again at some point in near future. For example when you run a program and it access a random subset of array locations before accessing another subset randomly then it is said to have temporal locality.

***Spatial Locality:*** When you consider a large array, if a program is accessing every element in that array while reading it only once and then moves to next element, but it doesn’t repeat the access to already visited locations until it finishes visiting every other untouched locations then it is said to be spatial locality.

1. **What are the three basic programming paradigms for parallel processing? Explain.**

There are three basic programming paradigms for parallel processing they are Explicit parallelism, Implicit parallelism and Ideal system. Explained as follows.

***Explict Parallelism***: It is a feature of programming language for parallel processing system which makes the programmer to write his program to differentiate which parts should be executed as independent parallel tasks. It gives high performance but more work to the programmer to indicate independent parallel tasks***.***

***Implict Parallelism:*** It is a feature of a programming language for a parallel processing system which automatically decides which tasks should run in parallel. Low performance compared to explicit parallelism.

***Ideal System:*** The Ideal system is a situation in parallelism which allows to use both explicit and implicit parallelism techniques***.***

1. **Discuss the difference between shared address space machines and distributed address space machines. Discuss the advantages and disadvantages of both architectures.**

***Shared Address Space:***

* In this shared address space processors will be directly accessing all the data in the system.
* It naurally fits the shared address
* ***Advantages***:
* It is typically time shared.
* Access to job queue can be centralized or decentralized.
* Here in shared memory it offers a unified address space in which all data can be found.
* ***Disadvantages***:
* Access to data is delayed.
* Primary disadvantage is the lack of scalability between the memory’s and CPU’s.
* It becomes increasingly difficult and expensive to design and produce shared memory machines with ever increasing number of processors

***Distributed Address Space:***

* In this Distributed address space each processor has a private memory and nothing is shared between them they can only directly access the local data.
* It naturally fits the private address.
* ***Advantages:***
* Simple to develop.
* Address space is easily expandable.
* Any CPU failure doesn’t affect the whole system, it is highly reliable.
* Nodes of parallel machine have simple version of OS.
* ***Disadvantages:***
* It is difficult to program for distributed address space
* Usually the access to parallel machine is via host computer running a serial OS.
* Computational tasks can only operate on local data

1. **What is parallel I/O? Why do need parallel I/O?**

Parallel I/O is a subset of parallel computing that performs multiple input/output operations simultaneously, rather than processing I/O requests serially one by one it accesses data on disk simultaneously.

We need parallel I/O because this allows a system to achieve higher write speeds and it maximizes bandwidth based on the principle that larger issues can be divided into multiple, smaller issues that can be solved at the same time. Used mostly in high-performance computing, parallelism can help run application efficiently and quickly.

1. **Give an example of anti-dependence and give a corresponding solution to remove the dependence.**

***Anti-dependence:*** A variable used in a statement is assigned to in a subsequent statement.

***Eg 1;*** Y= Z\*Z

Z=9

***Eg 2:*** 1. Z = 5

2. X = Z + 1

3. Z = 9

An anti-dependency is an example of a *name dependency*. That is, renaming of variables could remove the dependency, as in the next example:

1. Z = 3

N. Z2 =Z

2. X = Z2 + 1

3. Z = 9

Here to remove anti dependency between 2 and 3 a new variable Z2 has been declared and value of Z is copied to it. Now we can execute this instructions in parallel. However this modification has given new dependency, now instruction 2 is now truly dependent on instruction N and instruction N is truly dependent on instruction 1. Due to this flow dependencies these new dependencies are impossible to remove safely.

1. **Consider the search tree shown in the following figure, in which the dark node represents the solution.**

******

1. **If a sequential search of the tree is performed using the standard depth-*first search (DFS) algorithm, how much time does it take to find the* solution if traversing each arc of the tree takes one unit of time? Note: check this on how DFS works**.

If the Depth first search algorithm is performed for above graph, first it will visit root node then it’s child left node and then right child node, but here if left node is having any child nodes then it will visit them before visiting the right node. This process continues until it reaches the solution. So if we consider that each arc takes one unit of time it takes 11 units of time as running time for solution.

1. **Assume that the tree is partitioned between two processing elements that are assigned to do the search job, as shown in figure b. If both processing elements perform a DFS on their respective halves of the tree, how much time does it take for the solution to be found? What is the speedup? Is there a speedup anomaly? If so, can you explain the anomaly?**

If the tree is partitioned between two processing elements P0 & P1 and DFS is performed then it takes 4 units of time to get the solution node, so running time will be 4. Speedup = Ts/Tp = 11/4 = 2.75. There is a speedup anomaly. This is called superlinearity effects due to exploratory decomposition. The cause for this superlinearity is that the work performed by parallel and serial algorithms is different.

1. **Derive the formula for calculating the average access time for a word in a system with three levels of cache. Assume the following values for a theoretical system containing an L1, L2, and L3 cache.**

Average Memory-access time (AMAT) = Hit time + Miss rate \* Miss Penalty

AMAT for L1 cache can be written as = Hit timeL1 + Miss RateL1 \* Miss PenaltyL1

L2 Cache will reduce Miss PenaltyL1 as

Miss PenaltyL1 = Hit TimeL2 + Miss RateL2 \* Miss PenaltyL2

So AMAT is going to be as follows

AMAT= Hit TimeL1 + MissRateL1 + ( Hit TimeL2 +Miss RateL2 \* Miss PenaltyL2)

Again we can derive Miss PenaltyL2  as follows

Miss PenaltyL2 = Hit TimeL3 + Miss RateL3 \* Miss PenaltyL3

AMAT= Hit TimeL1 +MissRateL1\*( Hit TimeL2 +Miss RateL2 \* ( Hit TimeL3 +

Miss RateL3 \* Miss PenaltyL3 ) )

= 5 + 0.55 \* ( 10 + 0.25 \* ( 35 + 0.10 \* 100) )

= 5 + 0.55 \* ( 10+ 0.25 \* ( 45 ) )

= 5 + 0.55 \* ( 21.25 )

= 5 + 11.6875 = 16.69 ns

1. **Why is it difficult to construct a true shared-memory computer? What is the minimum number of switches for connecting p processors to a shared memory with b words (where each word can be accessed independently)?**

Shared memory computer uses one or more multiple core processors. In this case it is not difficult to construct a shared memory computer but it is difficult to construct an *efficient* shared memory computer. For a true shared-memory computer with p processors and a shared memory with b words, where each word to be accessed we need to have ( p\*b ) switches. Which makes the switching network complex and very expensive in this case.

1. **A cycle in a graph is defined as a path originating and terminating at the same node. The length of a cycle is the number of edges in the cycle. Show that there are no odd-length cycles in a d-dimensional hypercube**.

We'll have vertices that are d-bit binary numbers for a d-dimensional hypercube.

The vertex parity is0 if the number of 1s in the label is even and if the number of 1s in the label is odd then the parity is 1. If a given vertex parity is 1 then the adjacent vertex parity is 0. Now, let's see what the cycle's overall length will be if we start from a vertex with a parity of 0 and want a cycle to a vertex with the same parity. From 0 we'll go to the next vertex with parity 1 and from there we'll go to the next vertex with a parity of 0 by that we'll have a cycle in the graph and the cycle length will be 2. The same thing will happen when we begin with a vertex with a parity of 1. Therefore, in a d-dimensional hypercube it is not feasible to have an odd-length cycle.