Course Code : MCSE-011
Course Title : Parallel Computing
Assignment Number : MCA(5)/E011/Assign/2013
Maximum Marks : 100
Weightage : 25%
Last Dates for Submission : 31st October, 2013 (For July 2013 Session)
30th April, 2014 (For January 2014 Session)
20 marks are for viva voce. You may use illustrations and diagrams to enhance the explanations. Please go through the guidelines regarding assignments given in the Programme Guide for the format of presentation.
The answers are to be given in your own words an

Quesiton1: Discuss each of the following concepts, with at least one **(10 marks)** appropriate example not discussed in course material.
(i) Granularity in parallel/ concurrent environment
(ii) Speed-up
(iii) Data-flow computing
(iv) Scalability

Answer:

(i) **Granularity in parallel/ concurrent environment:**

Granularity is the extent to which a system is broken down into small parts, either the system itself or its description or observation. It is the extent to which a larger entity is subdivided. For example, a yard broken into inches has finer granularity than a yard broken into feet. **Coarse-grained** systems consist of fewer, larger components than **fine-grained** systems; a **coarse-grained** description of a system regards large subcomponents while a **fine-grained** description regards smaller components of which the larger ones are composed. The terms **granularity**, **coarse**, and **fine** are relative, used when comparing systems or descriptions of systems. An example of increasingly fine granularity: a list of nations in the United Nations, a list of all states/provinces in those nations, a list of all cities in those states, etc. The terms *fine* and *coarse* are used consistently across fields, but the term *granularity* itself is not. For example, in investing, more granularities refer to more positions of smaller size, while photographic film that is more granular has fewer and larger chemical "grains". In parallel computing, granularity means the amount of computation in relation to communication, i.e., the ratio of computation to the amount of communication. Fine-grained parallelism means individual tasks are relatively small in terms of code size and execution time. The data is transferred among processors frequently in amounts of one or a few memory words. Coarse-grained is the opposite: data is communicated infrequently, after larger amounts of computation. The finer the granularity, the greater the potential for parallelism and hence speed-up, but the greater the overheads of synchronization and communication. Granularity is also used to describe the division of data. Data

with low granularity is divided into a small number of fields, while data with high granularity is divided into a larger number of more specific fields. For example, a record of a person's physical characteristics with high data might have separate fields for the person's height, weight, age, sex, hair color, eye color, and so on, while a record with low data would record the same information in a smaller number of more general fields, and an even lower record would list all of the information in a single field. Greater granularity
makes data more flexible by allowing more specific parts of the data to be processed separately, but requires greater computational resources.

(ii)**Speed-up:**
In parallel computing, **speedup** refers to how much a parallel algorithm is faster than a corresponding sequential algorithm. Speedup is defined by the following formula: Sp= T1/T2

Where:
p is the number of processors

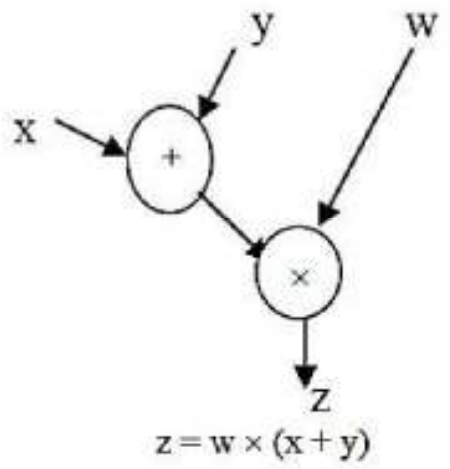
T_1 is the execution time of the sequential algorithm

T_2 is the execution time of the parallel algorithm with *p* processors

The speedup of a program using multiple processors in parallel computing is limited by the time needed for the sequential fraction of the program. The concept of speed up is used as a measure of the speed up that indicates up to what extent to which a sequential program can be parallelised. Speed up may be taken as a sort of degree of inherent parallelism in a program. For example if we talk about the computation of addition of natural numbers up to n sequence. The time complexity of the sequential algorithm for a machine with single processor is O(n) as we need one loop for reading as well as computing the output. However, in the parallel computer, let each number be allocated to individual processor and computation of shortest path used being a tree. In such a situation, the total number of steps required to compute the result is log n i.e. the time complexity is O(log n).

(iii) **Data-flow computing:** Data-flow computing plays an important role in parallel computing. It is described by the data-flow model. An alternative to the von Neumann model of computation is the dataflow computation model. In a dataflow model, control is tied to the flow of data. The order of instructions in the program plays no role on the execution order. Execution of an instruction can take place when all the data needed by the instruction are available. Data is in continuous flow independent of reusable memory cells and its availability initiates execution. Since, data is available for several instructions at the same time; these instructions can be executed in parallel. For the purpose of exploiting parallelism in computation Data Flow Graph notation is used to represent computations. In a data flow graph, the nodes represent instructions of the program and the edges represent data dependency between instructions.

As an example, the dataflow graph for the instruction z = w × (x+y) is shown in Figure below.



DFG for z =w x (x+y)

Data moves on the edges of the graph in the form of data tokens, which contain data values and status information. The asynchronous parallel computation is determined by the firing rule, which is expressed by means of tokens: a node of DFG can fire if there is a token on each of its input edges. If a node fires, it consumes the input tokens, performs the associated operation and places result tokens on the output edge. Graph nodes can be single instructions or tasks comprising multiple instructions. The advantage of the dataflow concept is that nodes of DFG can be self-scheduled. However, the hardware support to recognize the availability of necessary data is much more complicated than the von Neumann model. The example of dataflow computer includes Manchester Data Flow Machine, and MIT Tagged Token Data Flow architecture.

(iv) **Scalability: scalability** is the ability of a system, network, or process to handle a growing amount of work in a capable manner or its ability to be enlarged to accommodate that growth. For example, it can refer to the capability of a system to increase total throughput under an increased load when resources (typically hardware) are added. It refers to a parallel system's (hardware and/or software) ability to demonstrate a proportionate increase in (parallel) speedup with the addition of more processors. Factors that contribute to scalability include:
Hardware - particularly memory-cpu bandwidths and network communications:
Application algorithm
Parallel overhead related
Characteristics of your specific application and coding.

Question 2:
(a) Use Bernstein's conditions for determining the maximum **(3 marks)**

parallism between the instructions in the following segment.
S1: Y = X + Z
S2: Z = U + X
S3: S = R + V
S4: Z = Y + R
S5: P = N + Z

Answer:
R1 = {Y,Z} W1 = {X}
R2 = {U,V} W2 = {Z}
R3 = {S,V} W3 = {R}
R4 = {X,R} W4 = {Z}
R5 = {M,Z} W5= {Q}
Thus, S1, S3 and S5 and S2 & S4 are parallelizable.

(b) Discuss essential features of each of the following schemes for **(7 marks)**
classification of parallel computers:
(i) Flynn's (ii) Handler's (iii) Structural (iv) Based on grain-size

Answer:
(i) Flynn's Classification: Flynn's classification describes the behavioural concept of computer structure. It is based on multiplicity of instruction stream and data streams observed by the CPU during program execution. The term 'stream' refers to a sequence or flow of either instructions or data operated on by the computer. In the complete cycle of instruction execution, a flow of instructions from main memory to the CPU is established. This flow of instructions is called **instruction stream.** Similarly, there is a flow of operands between processor and memory bi-directionally. This flow of operands is called **data stream**.
Through the Flynn's Classification the computer organisation is classified into four types as follows:
(a) **Single Instruction and Single Data stream (SISD)**
(b) **Single Instruction and Multiple Data stream (SIMD)**
(c) **Multiple Instruction and Single Data stream (MISD)**
(d) **Multiple Instruction and Multiple Data stream (MIMD)**

(ii) Handler's Classification: Handler's classification elaborate notation for expressing the pipelining and parallelism of computers. Handler's classification is best explained by showing how the rules and operators are used to classify several machines.

Handler's classification addresses the computer at three distinct
levels: Processor control unit (PCU),
Arithmetic logic unit (ALU),
Bit-level circuit (BLC).

The PCU corresponds to a processor or CPU, the ALU corresponds to a functional unit or a processing element and the BLC corresponds to the logic circuit needed to perform onebit operations in the ALU. The direct use of numbers in the nomenclature of Handler's classification's makes it much more abstract and hence difficult. Handler's classification is highly geared towards the description of pipelines and chains. While it is well able to describe the parallelism in a single processor, the variety of parallelism in multiprocessor computers is not addressed well.

(iii) Structural Classification: Structural classification describes the structural concept of computer. Structural classification is based on the multiple processors with memory being globally shared between local copies of the memory. It can be classified into two types: (a) Loosely Coupled System and (b) Tightly Coupled System. It is basically described the memory management system while process are running in various way and with many memory address. In other word we can also say that this classification is used for shared memory management system.

(iv) Based on grain-size: The Based on Grain-Size classification is based on the recognizing the parallelism in a program to be executed on a multi-processor system. The Grain or Granularity is a measure which determines how much computer is involved in a process. The Grain size is determined by counting the number of instructions in a program segment. This classification is categorized in the three size those are Fine Grain, Medium Grain and Coarse Grain.

Through the Grain-Size classification parallelism can be classified at various levels as follows: Level 1 : Instruction Level
Level 2 : Loop Level
Level 3 : Procedure or Subprogram Level
Level 4 : Program Level

Question 3:
(a) How the following properties can be used in determining the **(2 marks)**

quality of an interconnection network:
(i) Network diameter (ii) Latency (iii) Bisection bandwidth

Answer:

(i) Network Diameter: It is the minimum distance between the farthest nodes in a network. The distance is measured in terms of number of distinct hops between any two nodes.
(ii) Latency: It is the delay in transferring the message between two nodes. In interconnection networks various nodes may be at different distances depending upon the topology. The network latency refers to the worst-case time delay for a unit message when transferred through the network between farthest nodes.
(iii) Bisection Bandwidth: Number of edges required to be cut to divide a network into two halves is called bisection bandwidth. Bisection bandwidth of a network is an indicator of robustness of a network in the sense that if the bisection bandwidth is large then there may be more alternative routes between a pair of nodes; any one of the other alternative routes may be chosen. However, the degree of difficulty of dividing a network into smaller networks is inversely proportional to the bisection bandwidth.

(b) Discuss relative merits and demerits of Tree Interconnection **(2 marks)**
Network vis-à-vis Systolic Array Network.

Answer:
Merits: Both interconnection networks can form a big network. In the tree interconnection network, processors are arranged in a complete binary tree pattern which can be many levels whereas the Systolic Array Network has also pipelined array architecture designed, for performing matrix multiplication which can form a big network with many levels.
Demerits: As we know that as big the network is the data transfer rate gone down. Both networks can form the many levels, hence both needs the high bandwidth for data transfer. Both networks are expensive due to its multi-connection design. Both are not easily maintainable.

(c) For K-ary n-cube network calculate each of the following **(6 marks)**
(i) Number of nodes in the network
(ii) The Network diameter (iii)Bisection
bandwidth of the network.

(i) Number of nodes in the network: In a k-ary n-cube network, the number of nodes $N = k_n$ for the torus and $N = 2_n$ for the hypercube. In the dual network, the number of nodes $N' = n' k'^{n'}$ for the torus and $N' = n' 2^{n'-1}$ for the hypercube. To have the same number of nodes $N = N'$ in the k-ary n-cube network and its dual:

$K^n = n' k'^{n'}$ for the torus, and Equation 1

$2^n = n' 2^{n'-1}$ for the hypercube Equation 2

Solving Equation 1 with n = n' gives:

$\frac{k}{k'} = \sqrt[n]{n}$ Equation 3

Hence, there are more nodes per dimension in a torus than that of its dual with equation dimensionality and number of nodes.

Solving Equation 2 yields:

n = (n'2/2)+log Equation

Thus the dimensionality of a hypercube is great example, there are-Dhypercube,1024nodesasinwella10as in the d

(ii) The Network diameter: The network diameter is defined as the maximum distance between any two nodes in the network. It is calculated by counting the number of hops between the two most distance nodes in the network.

In k-ary n-cube network, the diameter D= nk/2 for a torus, and D= n for a hypercube, in the dual network the diameter D' = n' k' / 2 for a torus, and D' = n' for a hypercube.

If the dimensionalities of a torus and its dual

$$\frac{D}{D'} = \frac{k}{k'} = \sqrt[n]{n}$$

For a hypercube:

$$\frac{D}{D'} = \frac{n}{n'} = \frac{n' + \log_2(n'/2)}{n'}$$

Hence, the -diameterary-cubennetworkofak is larger than that of

(iii) Bisection bandwidth of the network: The bisection width of the network is defined as the number of channels that must be crossed in order to cut the network into two equal sub-networks. The bisection width of a k-ary n-cube networks torus is $b = 2k^{n-1}$. The factor 2 is due to the ring arrangement in all dimensions. The bisection width of a hypercube is $b = 2k^{n-1}$. In the dual network, the bisection width is $b' = 2k'^{n'-1}$ for a torus, and $b' = 2^{n'-1}$ for a hypercube.

For a torus and its dual network with an equal number of nodes and same dimensionality:

$$\frac{b}{b'} = \frac{k^{n-1}}{k'^{n-1}} = \frac{k^n}{k'^n} X \frac{k'}{k} = \frac{n}{\sqrt[n]{n}}$$

For a hypercube network and its dual with an equal number of nodes:

$$\frac{b}{b'} = \frac{2^{n-1}}{2'^{n-1}} = \frac{n'}{k2}$$

Therefore, the -bisectionary-cubenetworkwidth isof largerkan equalthan thanu nodes.

Question 4: Write brief notes on any five of the following: **(10 marks)**
(i) Pipeline processing
(ii) Array processing
(iii)Associative Array Processing
(iv) VLIW architecture
(v) Multi-threaded processor
(vi) Superscalar processor

Answer:
(i) **Pipeline processing:** In an economical way on the digital computer the pipeline is the method to realize the overlapped parallelism in the proposed solution of the problem. By taking an example of an automobile assembling plant, where the production done in an automated way, we can easily understand the pipelining. The assembling line of the plant, every item is assembled from the separate stages (parts) and then output goes of one stage (part). By taking the analogy of the assembly line, pipelining is the method to introduce temporal parallelism in computer operations. Assembly line is the pipeline and the separate parts of the assembly line are different stages through which operands of an operation are passed.

To introduce pipelining in a processor P, the following steps must be followed:
Sub-divide the input process into a sequence of subtasks. These subtasks will make stages of pipeline, which are also known as segments.

Each stage Si of the pipeline according to the subtask will perform some operation on a distinct set of operands. When stage Si has completed its operation, results are passed to the next stage Si+1 for the next operation.

•The stage Si treceof ivesnputa fromnew-1. seprevious stage Si

The pipeline processors can classified into these types:
(a) Level of Processing
(b) Pipeline Configuration
(c) Type of Instruction and data

Further the classification according to the level of processing can be classified into these types:

Instruction Pipeline: We know that an instruction cycle may consist of many operations like, fetch opcode, decode opcode, compute operand addresses, fetch operands, and execute instructions. These operations of the instruction execution cycle can be realized through the pipelining concept. Each of these operations forms one stage of a pipeline. The overlapping of execution of the operations through the pipeline provides a speedup over the normal execution. Thus, the pipelines used for instruction cycle operations are known as instruction pipelines.

•**Arithmetic**The**Pipelin**compl**ex:** arithmetic operations like consume much of the time of the ALU. These opera of the ALU and as a consequence, high speed per arithmetic operationslinares .known as arithmetic pi

The classification according to the pipeline configuration can be classified into these types:

Unifunction Pipelines: When a fixed and dedicated function is performed through a pipeline, it is called a Unifunction pipeline.

Multifunction Pipelines: When different functions at different times are performed through the pipeline, this is known as Multifunction pipeline. Multifunction pipelines are reconfigurable at different times according to the operation being performed.

The classification according to the type of instruction and data also classified into these types:

Scalar Pipelines: This type of pipeline processes scalar operands of repeated scalar instructions.

•**Vector Pipelines:**Thistype of pipeline processes vector i

A pipeline processor can be defined as a processor that consists of a sequence of processing circuits called segments and a stream of operands (data) is passed through the pipeline. In each segment partial processing of the data stream is performed and the final output is received when the stream has passed through the whole pipeline. An operation that can be decomposed into a sequence of well-defined sub tasks is realized through the pipelining concept.

(ii) **Array processing:** Array processing is another method of vector processing. If we have an array of n processing elements (PEs) i.e., multiple ALUs for storing multiple operands of the vector, then an n instruction, for example, vector addition, is broadcast to all PEs such that they add all operands of the vector at the same time. That means all PEs will perform computation in parallel. All PEs are synchronised under one control unit.

of Array processing is same as in SIMD. An array processor can handle one instruction and multiple data streams as same in case of SIMD organisation. Therefore, array processors are also called SIMD array computers.
The organisation of an array processor is shown in Figure below:

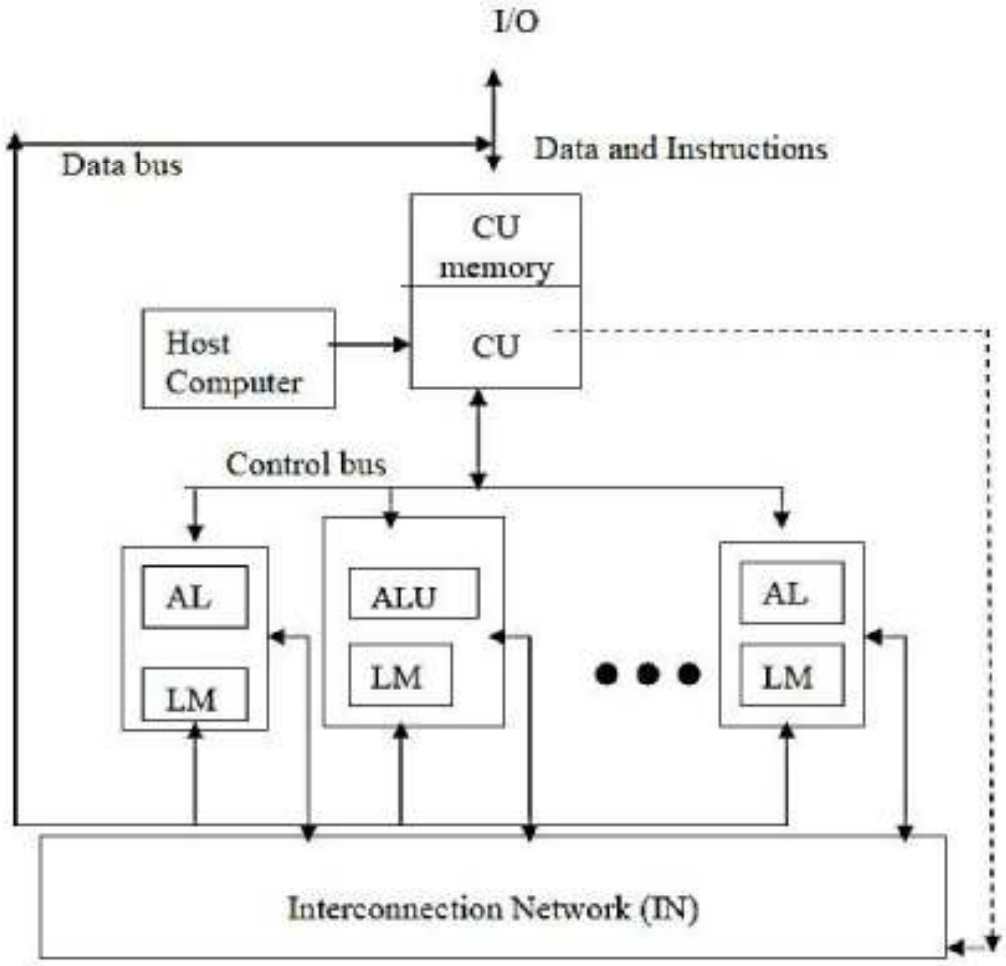


Figure: Organisation of SIMD Array Processor

Control Unit (CU): CU controls the inter communication between the PEs. The user programs are loaded into the CU memory. The vector instructions in the program are decoded by CU and broadcast to the array of PEs. Instruction fetch and decoding is done by the CU only.

Processing elements (PEs) : Each processing element consists of ALU, its registers and a local memory for storage of distributed data. These PEs have been interconnected via an interconnection network. All PEs receive the instructions from the control unit and the different component operands are fetched from their local memory. **Interconnection Network (IN):** IN performs data exchange among the PEs, data routing and manipulation functions. This IN is under the control of CU.
Host Computer: An array processor may be attached to a host computer through the control unit. The purpose of the host computer is to broadcast a sequence of vector instructions through CU to the PEs.
(iii) Associative Array Processing: The Array processor built with associative memory is called Associative Array Processor. An associative memory is content addressable memory, by which it is meant that multiple memory words are accessible in parallel. The parallel accessing feature also support parallel search and parallel compare.

This capability can be used in many applications such as:
Storage and retrieval of databases which are changing rapidly
Radar signal tracking
Image processing
•Artificial Intelligence

In the organisation of an associative memory, following registers are used:
• **Comparand Register (C):** This register is used to hold the operands, which are being searched for, or being compared with.
• **Masking Register (M):** It may be possible that all bit slices are not involved in parallel operations. Masking register is used to enable or disable the bit slices.
• **Indicator (I) and Temporary (T) Registers:** Indicator register is used to hold the current match patterns and temporary registers are used to hold the previous match patterns.
Types of Associative Processor:
Based on the associative memory organisations, we can classify the associative processors into the following categories:
1) **Fully Parallel Associative Processor**: This processor adopts the bit parallel memory organisation. There are two type of this associative processor:
Word Organized associative processor: In this processor one comparison logic is used with each bit cell of every word and the logical decision is achieved at the output of every word.
Distributed associative processor: In this processor comparison logic is provided with each character cell of a fixed number of bits or with a group of character cells. This is less complex and therefore less expensive compared to word organized associative processor.
2) **Bit Serial Associative Processor:** When the associative processor adopts bit serial memory organization then it is called bit serial associative processor. Since only one bit slice is involved in the parallel operations, logic is very much reduced and therefore this processor is much less expensive than the fully parallel associative processor.

PEPE is an example of distributed associative processor which was designed as a special purpose computer for performing real time radar tracking in a missile environment. STARAN is an example of a bit serial associative processor which was designed for digital image processing. There is a high cost performance ratio of associative processors. Due to this reason these have not been commercialised and are limited to military applications.

(iv) **VLIW architecture:** To improve the speed of the processor is to exploit a sequence of instructions having no dependency and may require different resources, thus avoiding resource conflicts. The idea is to combine these independent instructions in a compact long word incorporating many operations to be executed simultaneously. That is why; this architecture is called **very long instruction word (VLIW) architecture**. In fact, long instruction words carry the opcodes of different instructions, which are dispatched to different functional units of the processor. In this way, all the operations to be executed simultaneously by the functional units are synchronized in a VLIW instruction. The size of the VLIW instruction word can be in hundreds of bits. VLIW instructions must be formed by compacting small instruction words of conventional program. The job of compaction in VLIW is done by a compiler. The processor must have the sufficient resources to execute all the operations in VLIW word simultaneously.

(v) **Multi-threaded processor:** The use of distributed shared memory has the problem of accessing the remote memory, which results in latency problems. This problem increases in case of large-scale multiprocessors like massively parallel processors (MPP). In case of large-scale MPP systems, the following two problems arise:

Remote-load Latency Problem: When one processor needs some remote loading of data from other nodes, then the processor has to wait for these two remote load operations. The longer the time taken in remote loading, the greater will be the latency and idle period of the issuing processor.

Synchronization Latency Problem: If two concurrent processes are performing remote loading, then it is not known by what time two processes will load, as the issuing processor needs two remote memory loads by two processes together for some operation. That means two concurrent processes return the results asynchronously and this causes the synchronization latency for the processor.

The concept of Multithreading offers the solution to these problems. When the processor activities are multiplexed among many threads of execution, then problems are not occurring. In single threaded systems, only one thread of execution per process is present. But if we multiplex the activities of process among several threads, then the multithreading concept removes the latency problems.If we provide many contexts to multiple threads, then processors with multiple contexts are called multithreaded processor (systems). These systems are implemented in a manner similar to multitasking systems. A multithreaded processor will suspend the current context and switch to another. In this way, the processor will be busy most of the time and latency problems will also be optimized. Multithreaded architecture depends on the context switching time between the threads. The switching time should be very less as compared to latency time.

(vi) **Superscalar processor:** In scalar processors, only one instruction is executed per cycle. That means only one instruction is issued per cycle and only one instruction is completed. But the speed of the processor can be improved in scalar pipeline processor if multiple instructions instead of one are issued per cycle. This idea of improving the processor's speed by having multiple instructions per cycle is known as Superscalar processing. In superscalar processing multiple instructions are issued per cycle and multiple results are generated per cycle.

superscalar processing is how many instructions we can issue per cycle. If we can issue k number of instructions per cycle in a superscalar processor, then that processor is called a k-degree superscalar processor. If we want to exploit the full parallelism from a superscalar processor then k instructions must be executable in parallel.

For implementing superscalar processing, some special hardware must be provided which is as follows:
The requirement of data path is increased with the degree of superscalar processing. Suppose, one instruction size is 32 bit and we are using 2-degree superscalar processor, then 64 data path from the instruction memory is required and 2 instruction registers are also needed.
Multiple execution units are also required for executing multiple instructions and to avoid resource conflicts.
Many popular commercial processors have been imp DEC 21064, MIPS R4000,m,etcPower. PC, Pentiu

Question 5:
(a) Using sorting algorithm for combinational circuit given in **(5 marks)**

Section 1.7 of Block 2, sort the following sequence of numbers in increasing order.
3, 8, 5, 10,90,9, 40,12, 95,20, 0,14 60, 23, 83

Answer:

The sorted form of the given sequence is: **0, 3, 5, 8, 9, 10, 12, 14, 20, 23, 40, 60, 83, 90, 95**

(b) Using matrix multiplication algorithm given in Section 1.10, compute the following product: **(5 marks)**

$$\begin{pmatrix} 1 & 3 \\ 5 & 7 \end{pmatrix} \begin{pmatrix} 8 & 4 \\ 6 & 2 \end{pmatrix}$$

Answer:

$$= \begin{pmatrix} 8+18 & 4+6 \\ 40+42 & 20+14 \end{pmatrix}$$

$$= \begin{pmatrix} 26 & 10 \\ 82 & 34 \end{pmatrix}$$

Question 6:
(a) Discuss relative merits and demerits of three types of **(5 marks)**

implementations, viz.,
(i) Message passing (ii) Shared memory

(iii)Data parallel of PRAM model

Answer:
(i) Merits of Message Passage:
Provides excellent low-level control of parallelism;
Portable;
Minimal overhead in parallel synchronisation and data distribution; and
It is less error prone.
Drawbacks Message-passing code generally requires more software overhead than parallel shared-memory code.
(ii) Merits of Shared Memory:
Global address space provides a user-friendly programming perspective to memory.
Data sharing between processes is both fast and uniform due to the proximity of memory to CPUs.
No need to specify explicitly the communication of data between processes.

Negligible process-communication
overhead. More intuitive and easier to learn.

Drawbacks

Not portable.
Difficult to manage data locality.
Scalability is limited by the number of access pathways to memory.
User is responsible for specifying synchronization, e.g., locks.

***(iii) Merits of* Data parallel of PRAM model:**

The number of operations executed per one cycle on *p* processors is at most *p*.
Any processor can read or write **any** shared memory cell in unit time.
It abstracts from any communication or synchronization overhead, which makes the complexity and correctness analysis of PRAM algorithms easier.
All the processors have read and write access to a shared global memory.
In the *PRAM* the access can be simultaneous.
A body of algorithms exist for this shared memory model.
The model ignores algorithmic details of synchronization and communication. It makes explicit the association between operations and processors.
PRAM algorithms are robust – network-based algorithms can be derived from them. PRAM is MIMD model.

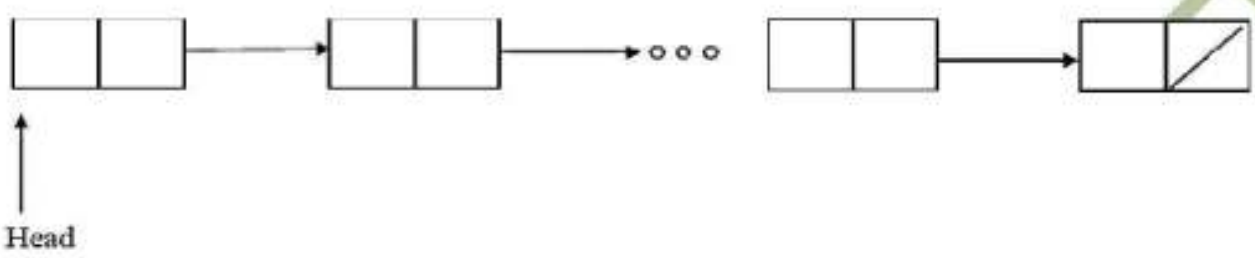
Drawbacks

•Each processor shares llonlyexecutiononeglobalat thmemorysamefot

(b) Write short notes for any two of the following data structures **(5 marks)**
for parallel algorithms
(i)Linked list (ii) Array pointers (iii) Hypercu

Answer:

(1) Linked List: A linked list is a data structure composed of zero or more nodes linked by pointers. Each node consists of two parts, as shown in *Figure below*: *info* field containing specific information and *next* field containing address of next node. First node is pointed by an external pointer called *head*. Last node called tail node does not contain address of any node. Hence, its next field points to null. Linked list with zero nodes is called null linked list.



A large number of operations can be performed using the linked list. For some of the operations like insertion or deletion of the new data, linked list takes constant time, but it is time consuming for some other operations like searching a data. We are giving here an example where linked list is used:
Given a linear linked list, rank the list elements in terms of the distance from each to the last element.
A parallel algorithm for this problem is given h

Algorithm:
Processor j, 0 j<p, do
if next[j]=j then
rank[j]=0
else rank[j]
=1 endif
while rank[next[first]]0 Processor j, 0 j<p, do
rank[j]=rank[j]+rank[next[j]]
next[j]=next[next[j]]
endwhile

The working of this algorithm is illustrated by

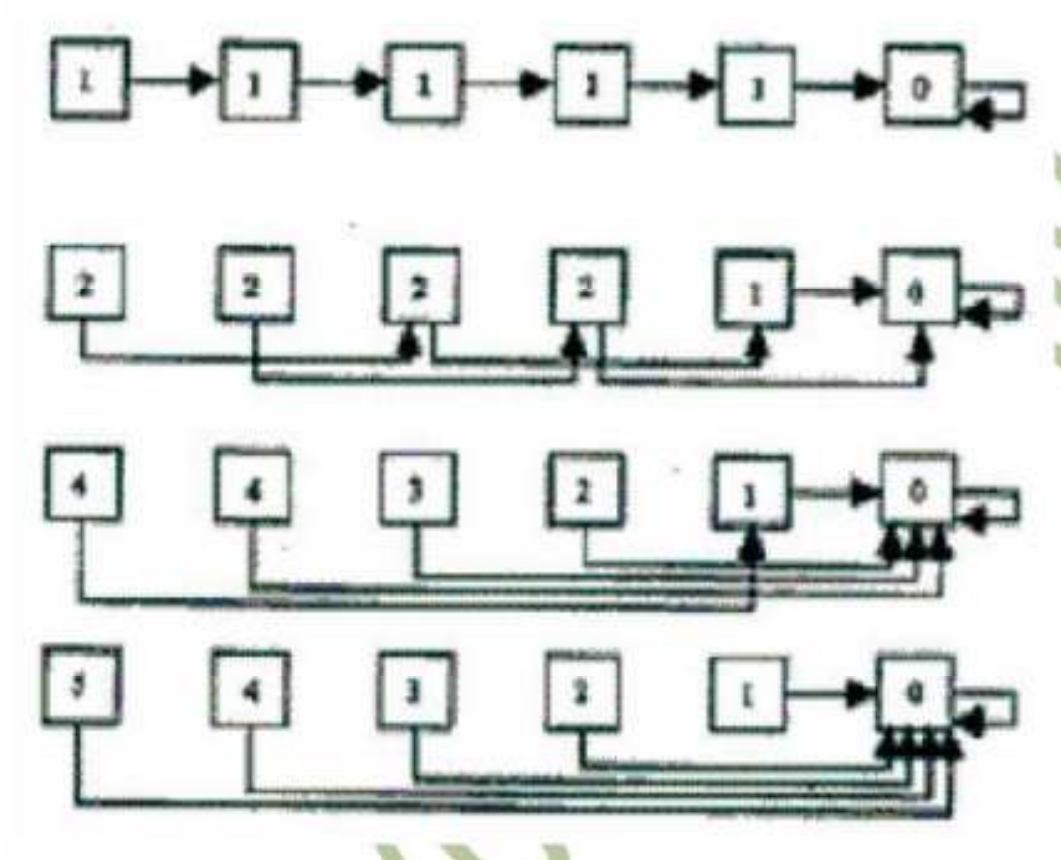


Figure: Finding Rank of elements

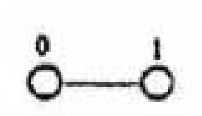
(ii) Arrays Pointers: An array is a collection of the similar type of data. At the one hand, arrays can be used as a common memory resource for the shared memory programming; on the other hand they can be easily partitioned into sub-arrays for data parallel programming. This is the flexibility of the arrays that makes them most frequently used data structure in parallel programming.

Consider the array shownyisbelow10.. The size of the

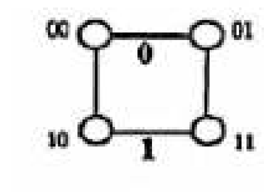
5	10	15	20	25	30	35	40	45	50

(ii)**Hypercube**Th**Network:**hypercube architecture has playe

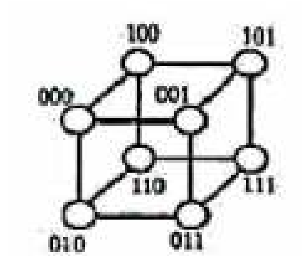
the hypercube supports a variety of elegant called-cubes,n where n indicates the number of d depicted below:



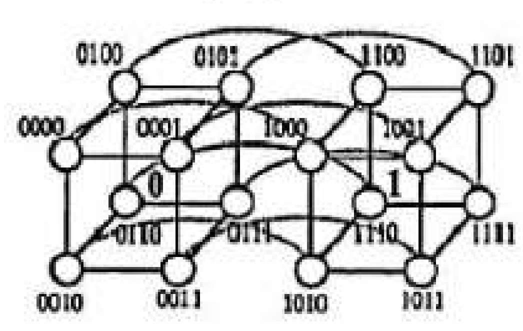
1-cube built of 2 0-cubes
Figure 8(a): 1-cube



2-cube built of 2 1-cubes
Figure 8(b): 2-cube



3-cube built of 2 2-cubes
Figure 8(c): 3-cube



4-cube built of 2 3-cubes

Properties of Hypercube:

A node p in a n-cube has a unique label, its binary ID, that is a n-bit binary number.
The labels of any two neighbouring nodes differ in exactly 1 bit.
Two nodes whose labels differ in *k* bits are connected by a shortest path of length *k*. Hypercube is both node- and edge- symmetric.

Hypercube structure can be used to implement man that is, algorithms innicatewhich witheach everytask mustothercot computation-to-allrequiringcommunicationall*P*tasks toamongbe performed in polynomial time using other data structures like

Question 7:
(a) Enumerate different steps to write a general parallel **(3 marks)**

programme

Answer: Steps to write a general parallel program:
i) Understand the problem thoroughly and analyze that portion of the program that can be parallelized;
ii) Partition the problem either in data centric way or in function centric way depending upon the nature of the problem;
iii) Decision of communication model among processes:
iv) Decision of mechanism for synchronization of processes,
v) Removal of data dependencies (if any),
vi) Load balancing among processors,
vii) Performance analysis of program. to set upper triangle of a matrix to zero.

(b) In High-performance Fortran, write a FOR ALL statement **(3 marks)**

Answer:
The following statements set each element of matrix X to the sum of its indices. FORALL (i=1:m, j=1:n) X(i,j) = i+j
and the following statement sets the upper right triangle of matrix Y to zero .
FORALL (i=1:n, j=1:n, i<j) Y(i,j) = 0.0

(c) Write a pscudo-code to find the product f(a) * f(B) of two **(4 marks)**

functions in shared memory programming using lib

Answer:
Pseudo-code to find the product f(a) * f(B) ofibrartwo

Process A	Process B
mul = 0	:
:	:
fork B	mul = mul + f(B)
	:
Mul = mul X f(a)	end B
:	
Join B	
:	
end A	

Pseudo-code to find the product f(a) * f(B) of two routines**with Locking:**

Process A	Process B
mul = 0	:
:	lock mul
fork B	mul = mul + f(B)
:	unlock mul
lock mul	
mul = mul X f(a)	end B
:	
Join B	
:	
end A	

Question 8:
Discuss in detail synchronization problem and its possible solutions for performance and correctness of execution in parallel computing environment. **(10 marks)**

Answer:
In multiprocessing, various processors need to communicate with each other. Thus, synchronisation is required between them. The performance and correctness of parallel execution depends upon efficient synchronisation among concurrent computations in multiple processes. The synchronisation problem may arise because of sharing of writable data objects among processes. Synchronisation includes implementing the order of operations in an algorithm by finding the dependencies in writable data. Shared object access in MIMD architecture requires dynamic management at run time, which is much more complex as compared to that of SIMD architecture. Low-level synchronization primitives are implemented directly in hardware. Other resources like CPU, Bus and memory unit also need synchronisation in Parallel computers.
To understand the synchronization, the following dependencies are identified:
i) **Data Dependency:** These are WAR, RAW, and WAW dependency.
ii) **Control dependency:** These depend upon control statements like GO TO, IF THEN, etc.
iii) **Side Effect Dependencies:** These arise due to exceptions, Traps, I/O accesses.

For the proper execution order as enforced by correct synchronization, program dependencies must be analysed properly. Protocols like wait protocol, and sole access protocol are used for doing synchronisation.

Wait protocol

The wait protocol is used for resolving the conflicts, which arise because of a number of multiprocessors demanding the same resource. There are two types of wait protocols: busy-wait and sleep-wait. In busy-wait protocol, process stays in the process context register, which continuously tries for processor availability. In sleep-wait protocol, wait protocol process is removed from the processor and is kept in the wait queue. The hardware complexity of this protocol is more than busy-wait in multiprocessor system; if locks are used for synchronization then busy-wait is used more than sleep-wait.

Execution modes of a multiprocessor: various modes of multiprocessing include parallel execution of programs at (i) Fine Grain Level (Process Level), (ii) Medium Grain Level (Task Level), (iii) Coarse Grain Level (Program Level).

For executing the programs in these modes, the following actions/conditions are required at OS level.

i) Context switching between multiple processes should be fast. In order to make context switching easy multiple sets should be present.

ii) The memory allocation to various processes should be fast and context free.

iii) The Synchronization mechanism among multiple processes should be effective.

iv) OS should providermancesoftwaremonitoringtools. for perfo

Sole Access Protocol: The atomic operations, which have conflicts, are handled using sole access protocol. The method used for synchronization in this protocol is described below:

1) **Lock Synchronization:** In this method contents of an atom are updated by requester process and sole access is granted before the atomic operation. This method can be applied for shared read-only access.

2) **Optimistic Synchronization:** This method also updates the atom by requester process, but sole access is granted after atomic operation via abortion. This technique is also called post synchronisation. In this method, any process may secure sole access after first completing an atomic operation on a local version of the atom, and then executing the global version of the atom. The second operation ensures the concurrent update of the first atom with the Updation of second atom.

3)**Server synchronization:**Itupdatesthe atom by the server pro atom behavesuniqueas update server. A process reques request to the atom's update server.