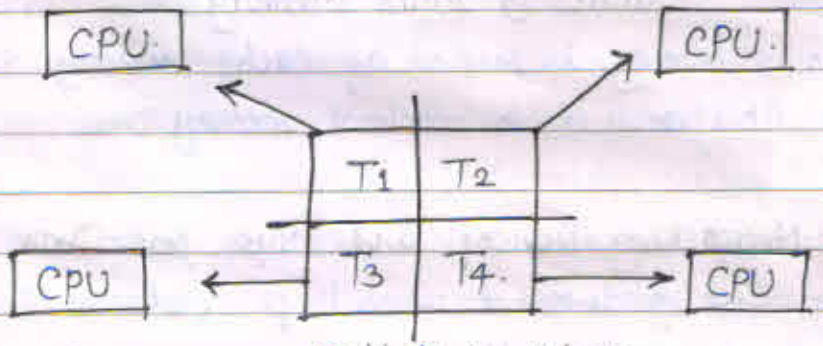
PROF. ANAND N. GHARU
ASSISTANT PROFESSOR
PVGCOE, NASHIK.
COMPUTER DEPARTMENT
HIGH PERFORMANCE COMPUTING

UNIT -I

Name : Prof. Gharu
Assistant Professor.
Computer Department.
8087777708
gharu.anand@gmail.com

1. Parallel Processing Concepts

* Introduction of Parallel Computing :
  - parallel computing is basically used to describe about the solving of a single problem using two or more processor.
    e.g. Supercomputer
  - Definition :
    "Parallel Computing is a type of computing/computation or the exe. of processes are carried out simultaneously. Large problems can often be divided into smaller ones, which can then be solved at the same time."



Probl. to be solved.

Ifel Computation of tasks.

* Motivation for parallelism :

→ 1] Increase nos of transistors in integrated chips enhances Computing power.
2] Improvements in storage technology (Memory/Disk.)
3] Improve n/wing devices and systems for Data communications.

1] Increase nos of transistors in integrated chips. enhances. Computing power.

→ - The power of CPU increases because of the additional transistors in the ICs.
- The modern processing elements are equipments with multiple units. and these unit can be assigned to perform processing task individually.

2] Improvement in storage technology (Memory / Disk)

→ - As we may know, processor alone is not responsible for the enhancement in computation speed but primary memory as well as disk speed also play very significant role.
- As the no of instn executed in per clk cycle by the processor has increased but at the same time memory should be able to feed the required data for execution.
- The tech called locality of references is used to manage the mismatch bet the memory & the processor speed

Cache & Main Memory

- The fastest memory is termed as cache memory is used to cope with situation. it is fastest memory

3] Improved Networking Devices and Sys. for Data Comm :

- Now a days, use of Internet as a large platform for IIel and distributed Computation.
- Internet is used to provide the environment for performing large Computation.
- transfer information from one node to another node through use of Internet.

* Scope of Parallel Computing :

→ - The parallel computing is suitable for the problems require much more time for Computation completion.
- It is used based on high end engg & scientific problems
- This includes Computer based simulations and Computational fluid dynamics (CFD) as well as other computer based digital image processing and security algorithms.

* Application of Parallel Computing in Engg :

→ - parallel computing is basically for speedup of computation.
- This phenomenon is adopted in several engg application domain such as aerodynamics, optimization algo., like branch & bound and genetic programming, clustering etc.

* Application in Scientic area :
- To show the simulated behaviour of real world entities by using mathematics & mathematical formulas.
- The scientific appli^n are major candidates for the Parallel computation eg. weather forecasting and climate modeling, oil exploration and energy research, drug discovery & genomic research etc.

* Commercial Appli^n for Parallel computing. +
- It require more procesing power in the current market trends because performing many activities simultaneously.
- we can consider multimedia application as commercial application.

* Parallel computation application in Computer system ?
- The computation perform in Parallel, by the use of collection of low power computing devices in the form of clusters.
- The group of computer connected together to solve complex problem is termed as cluster computing system.
- It is basically applied in the field of computer security.

* Parallel Programming Platforms :
- The basic component of the sequential computer are memory unit and the procesor.
- In some system, memory unit is connected with the procesor thro the data path.
- The multiplicity is used for performance improvement.
The multiplicity is achieved by introducing multiple elements. eg memory unit, procesing element and data path.
- There are 2 variation in multiplicity.
One is in implicit Parallelism where it is not seen by the programer whereas it can be exposed to the programmer in diff form.

* pipelining and Superscalar Execution ?

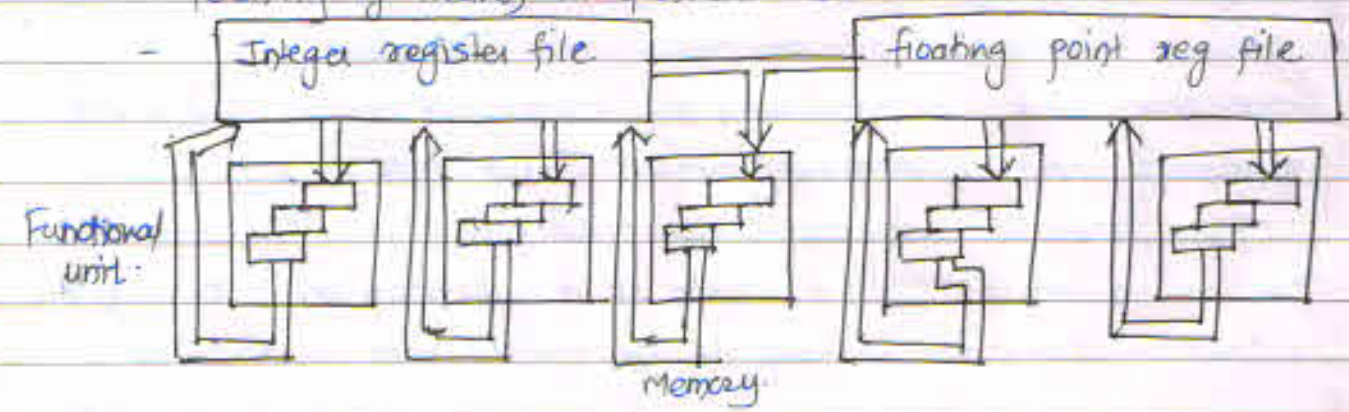
→ - pipelining is the proces ~~executing~~ fetching the ~~current~~ next instruction when current instruction is being executed

- The pipeline processor is based on pipelining used in car assembly line. The whole task to be performed in an assembly line is divided into sub-tasks. It was invented by Henry Ford.
- Henry Ford got an idea that it took much more time to build a car physically but he could actually build a car in a minute.
- he did some step, car was divided into diff. stages on assembly line., one stage is assigned to put engine., another stage is assigned wheels, next stage is used to put cabinet & so on.

Instruction Pipeline.

* Superscalar Execution :

- A processor designed to execute more than one instru at a time during single clock cycle is considered as superscalar executions.
- Superscalar processor fetches & decodes several instru at a time
- The superscalar architecture exploits the potential of ILP. (Instru level Parallelism)
- The typical behaviour of a superscalar archi. is described with the fetching of multiple instru at a time and the attempt for fetching of nearby independent instru.
-



Superscalar Organization.

- Superscalar m/c is based on von-neumann architecture But it can issue more than one instru per clock cycle.
- A superscalar m/c uses multiple pipelines because with a single pipeline it is not possible to issue multiple instru.
- The classifications of the superscalar processors are based upon the maximum number of instru can be issued at the same time.
- This is depend on the pipeline structure used in the processor.

* Very Long Instru Word processors (VLIW) :

→ - VLIW (Very-Long instru word) Architectures are considered as one of the suitable alternatives to achieve Instru level parallelism (ILP) in programs.
- VLIW architecture are used for exe. of more than one basic instru at a time in prog.
- VLIW architecture can store multiple instru in single word. In VLIW based system a parallel compiler is used to generate operation to be executed in parallel in the same word.
- The compiler is responsible for resolving dependancies among instru at compile time.
- VLIW in this architecture indicates that the prog to be executed in such processors is to be recompiled in a way that the instru runs sequentially w/o existense of stall in the pipeline.
- In this architecture, it is not required for the h/w to examine the Instru stream about the instru to be executed in parallel.
- The compiler determines which oper to be executed in parallel.
- The performance of VLIW architecture is very good when sequential prog written in C or Fortran lang are executed after recompilation for such system.
- The VLIW Compiler ensures that all oper in executing unit can perform simultaneously.
- The VLIW architecture (processor) are used in application area where high performance is required with less cost.

e.g. DSP (Digital signal processing)

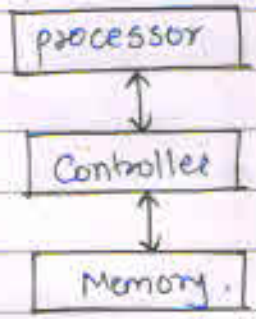
* Basic Working of VLIW Processor :- (Principle)
- VLIW procenor's basic aim to speeding up computation by exploiting ILP.
- VLIW uses same h/w core as superscalar processors with multiple execution unit (EU) working in ||el.
- In VLIW procenor an instru consist g multiple operations in which a typical word length considered from 52 bits to 1 Kbits.
- In this, all oper in an instru are executed in lock-step-mode.
- The VLIW processor relies on compiler to find ||elism and schedule dependancy free prog. code.

* Advantages g VLIW :-
- do not need complicated logic to check for dependancies.
- Eliminated Complicated Instru scheduling
- Compiler is critical to performance.

* Disadvantages g VLIW :-
- Increased code size to empty "slot"
- Increased Memory bandwidth.
- Compiler is critical to performance :- must do all dependancy resolution.
- Cache misses in one pipeline will force all pipelines to stall in a "pure" VLIW machine.

* Limitation of Memory System Performance :-
- As we know, performance g compute not only depend on procening element but memory also does matter.
- The use g cache memory shows a very useful impact on the overall performance g the system.
- A cache is a high-speed memory to be used as the buffer memory. It is logically placed bet CPU & main memory

- Controller resides bet^n processor and memory
- The accen g the memory is handled by the controller as an intermediate ~~entry~~ entity.
- The procenor send it's read/write request to the ~~processor~~ Controller.
- The controller performs the translation g memory addresses and request into appropriate signals for the underlying memory.
- Finally controller panes these signal to the memory chips.



Controller bet^n Memory and processor.

- The memory system performance can be assened by measuring the speed at which a seq. g operation performed by the system.
- Latency is measure limitation g memory because when we execute nes instn thro memory, then processor should be able to provide speed as fast as needed. otherwise there can be delay bet^n procenor & memory speed.
- In this, dependancies must be resolving while executing procen.
- procenor & memory accen speed should be compatible.

* Use of Caches for Improvement of Latency :
* Memory Bandwidth.
* Memory Latency hiding tech^n.
* Effect g Multi-threading & Prefetching.

* Basic Working of VLIW Processor :- (Principle)

- VLIW procesor's basic aim to speeding up computation by exploiting ILP.
- VLIW uses same h/w core as superscalar processors with multiple execution unit (EU) working in ||el.
- In VLIW procesor an instru consist of multiple operations in which a typical word length considered from 52 bits to 1 Kbits.
- In this, all oper in an instru are executed in lock-step-mode.
- The VLIW processor relies on compiler to find ||elism and schedule dependancy free prog. code.

* Advantages of VLIW :-

- do not need complicated logic to check for dependancies
- Eliminated Complicated Instru scheduling
- Compiler is critical to performance.

* Disadvantages of VLIW :-

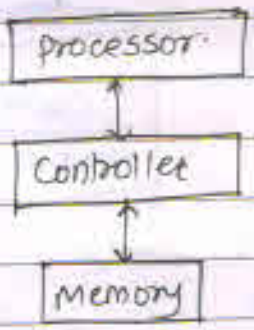
- Increased code size to empty "slot"
- Increased memory bandwidth.
- Compiler is critical to performance :- must do all dependancy resolution.
- Cache misses in one pipeline will force all pipelines to stall in a "pure" VLIW machine.

* Limitation of Memory System Performance :-

- As we know, performance of compute not only depend on procesing element but memory also does matter.
- The use of cache memory shows a very useful impact on the overall performance of the system.
- A cache is a high-speed memory to be used as the buffer memory. It is logically placed bet CPU & main memory

* Limitation of Memory System performance :-

- The effective performance of a (computer) program on a computer relies not just on the speed of processor but also on the ability of the memory system to feed data to the processor.
- Execution speed highly depend upon the speed at which instruction and data supplied to the processor by memory components
- A cache is high speed memory to be used as buffer memory. It is logically placed bet CPU & main memory.
- Latency :-
  It is the time elapses bet the start of operation and complete of operation.
  Latency does not provide complete information about performance of the memory system.
- The controller resides bet processor & physical memory. The access of memory by the processor is handled by the controller as an intermediate entity.



Controller bet Processor & Memory.

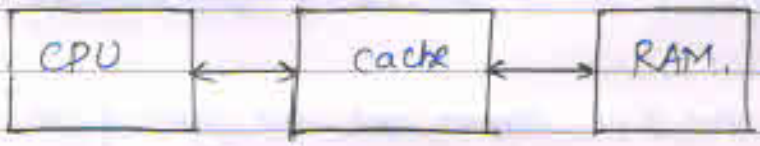
- Bandwidth :-
  The rate at which data can be pumped from the memory to the processor determines the bandwidth of memory system

* Use of Cache for Improvement of latency :-

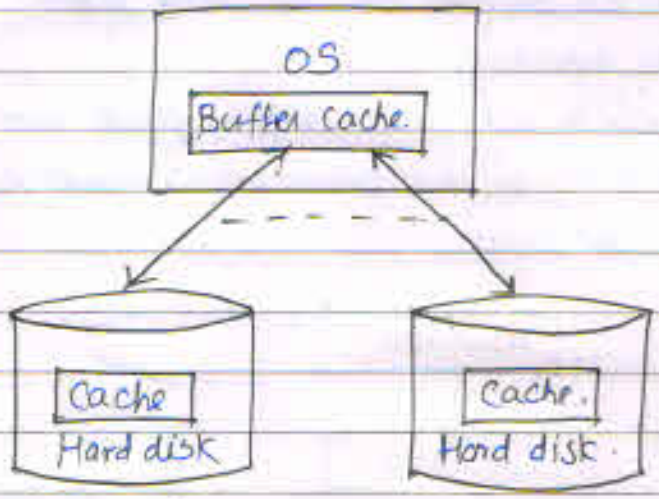
- The Cache memory is used to enhance the access speed of any storage devices
  e.g. disk drive, main memory, tape storage, web server :- for other cache also.
- The principle upon which cache works is called locality of references.

- The Cache are divided into 2 caches - hardware & Software.
- The Cache is memory with low latency & high bandwidth properties.

If one page/appli? is already fetched by processor then it is saved in Cache memory so next time it can be fetched by cache mem so it improves latency of computer.



Hardware Cache.



Software Cache.

- There are 2 terms used here Cache-hit and cache miss. When the requested data references in satisfied by the cache is called cache-hit otherwise cache miss.
- The cache memory comes with different System., Actually amount of data that can be stored in the cache is considered as the capacity of that cache,
  e.g. 32 KB Cache.

1) Cache block - contain multiple byte/words of data.
2) Cache set - row in the cache.
3) Tag - prefix group of data uniquely

* Memory Bandwidth & (Double data Rate (DDR))

- Memory Bandwidth is the rate at which data can be read from or stored into a semiconductor memory by a processor.
- Memory Bandwidth usually expressed in units of bytes per second.
- The memory Bus & memory unit are used to determine memory bandwidth.
- The memory Bandwidth can be improved by increasing size of memory block.

* Memory Latency hiding techniques:

- The increase in memory latency typically occures when need arises to access Remote memory.
  e.g. Distributed Shared memory based System.

The important Latency hiding tech are as follow:

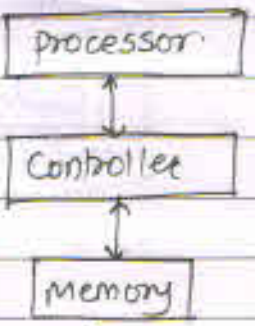
1) Using Prefetching techniques:

- The prefetching is either software / hardware. Controlled.
- In s/w controlled prefetching, explicit "prefetch" instruction are issued for data that is known to be remote.
- In h/w controlled prefetching, it is done thro use of long cache line to capitalize on spatial locality or through the instru lookahead.
- The prefetching tech is used for latency hiding because it brings instru or data close to the processor befor their actual requirement.
- The direct effect of this scheme is that the time duration bet the issues of instru and its actual refelences is increased. This is very significant impact when latencies are large.

2) Use of Coherent Cacheing tech
3) Relaxing the memory consistency Requirement
4) Using Multiple Context to hide latency.

* Limitation of Memory System performance :-

- The effective performance of a (computer) program on a computer relies not just on the speed of processor but also on the ability of the memory system to feed data to the processor.
- Execution speed highly depend upon the speed at which instruction and data supplied to the processor by memory components.
- A cache is high speed memory to be used as buffer memory. It is logically placed between CPU & main memory.
- Latency :-
  It is the time elapses between the start of operation and complete of operation.
  Latency does not provide complete information about performance of the memory system.
- The controller resides between processor & physical memory. The access of memory by the processor is handled by the controller as an intermediate entity.



Controller between Processor & Memory.

- Bandwidth :-
  The rate at which data can be pumped from the memory to the processor determines the bandwidth of memory system.

* Use of Cache for Improvement of latency :-

- The Cache memory is used to enhance the access speed of any storage devices
  e.g. disk drive, main memory, tape storage, web server. for other cache also.
- The principle upon which cache works is called locality of references.

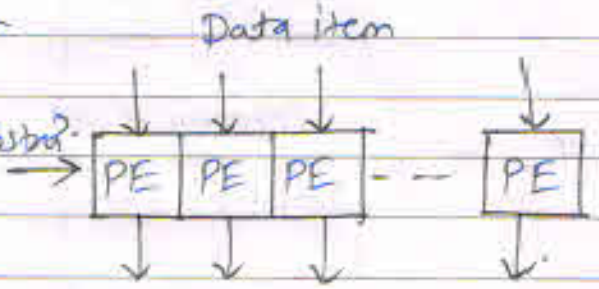
* Parallel Computing platforms :-

→ - To facilitate IIel platform, we need to study physical & Logical org
- physical org means actual h/w org of the platform
logical org means: programmers view of the platform.

- * Control structure of IIel platform :-
- In this, sys. is divided into multiple. Smaller part is granular in structure.
- There are 2 types 1) Cource grained 2) Fine grained.
- When sys is divided into large nos & small part is called as fine grained.
Sys. is divided into smaller nos & large parts is called as Course grained.
- The term granularity is used to describe about the divisio of a task into nos & smaller subtasks.
- The granularity in IIel prog. is considered as different leve. e.g. prog. level and instn level.
- There are 2 approaches for working & procesing unit in a IIel computer:
- First, Single Control unit is used to co-ordinate all procesing unit centrally
- Second, it is based on working & various processing units independantly.

* SIMD :- (single Instn stream & Multiple Data stream) Architecture :-

→ In SIMD processor, one instruction works on several data item Simultaneously by using Several procesing elements (PEs) all of which carry out the Same operation as shown in fig



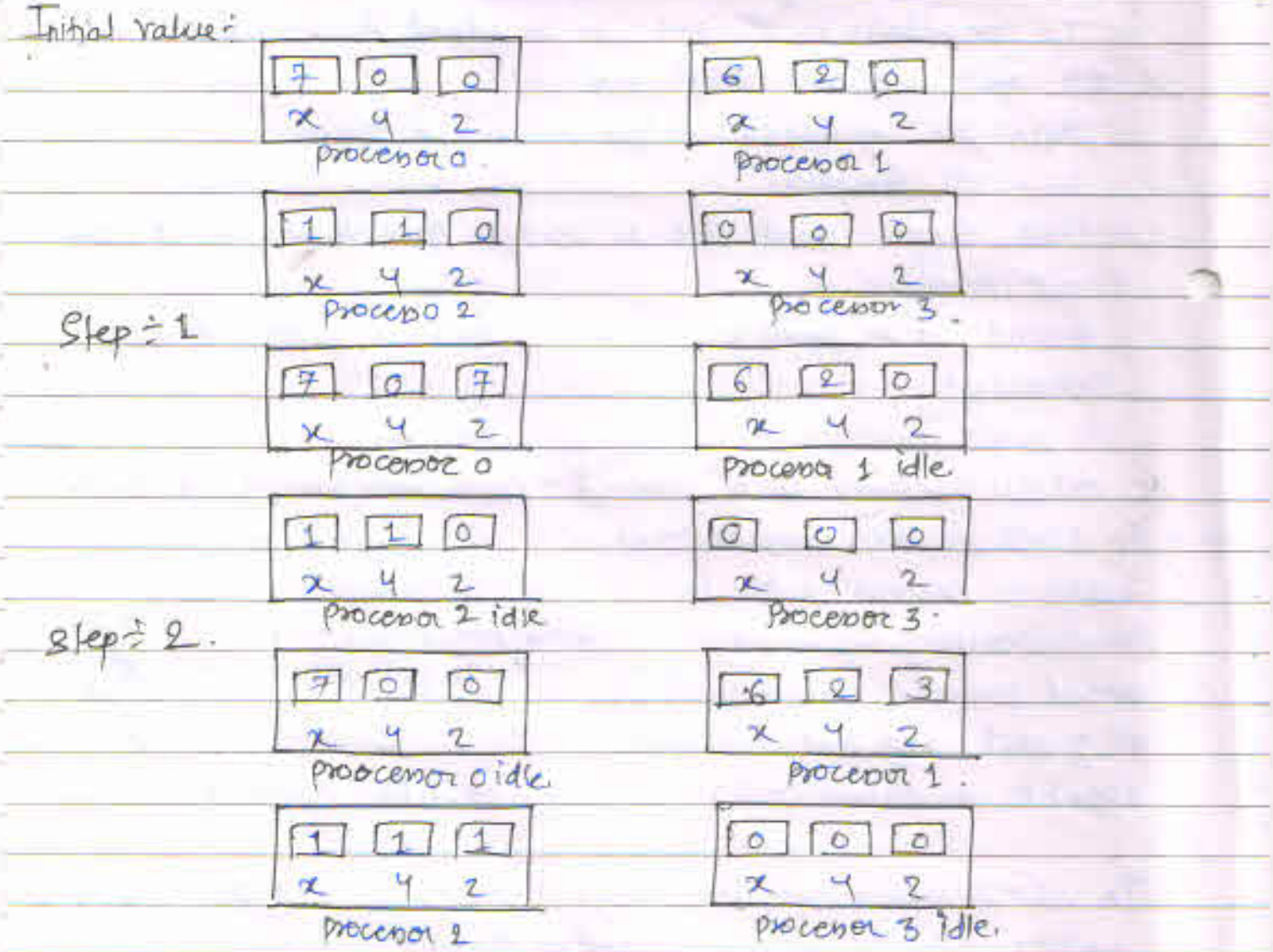
SIMD Architecture.

- The SIMD model sys. uses single control unit to dispatch multiple instn to various procesing element

- In SIMD computing model, single control unit is used to read instr^{n} by single prog counter (PC)., decode them and send control signal to the PE's.
- The nos of data paths are depend upon nos of PE.
- Data are supplied to & derived from PE by the memory.
- The diff. unit like PE and memory module are connected by interconne^{n} n/w.
- e.g. exe- of cond^{al} statements on SIMD architecture.

```
if (y==0)
 {
    z=x
 }
else
 {
   z=x/y;
 }
```

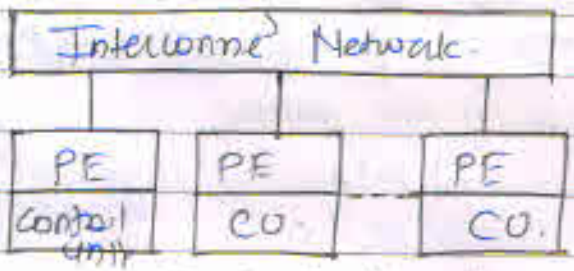
Conditional statement



Execution of Conditional Statement

* MIMD. (Multiple Instru stream Multiple data stream) Architecture:-

- This model represent the sys. capable of executing multiple instru on multiple data sets simultaneously.
- In fact, MIMD is used to describe a ||el mic able to perform independant computation at same time.
- The MIMD class of mic can execute independant program at the same time. The procesing elements included in MIMD class of mic execute different progs at a time.
- This means in MIMD class of mic each processor fetches it's own instru and operates on it's own data.



MIMD Architecture.

* Comm model for ||el platform:-

- There are diff form of transfer among nos of ||el tasks running concurrently.
- The basic 2 forms are - i) Shared data approach
  ii) message passing approach.

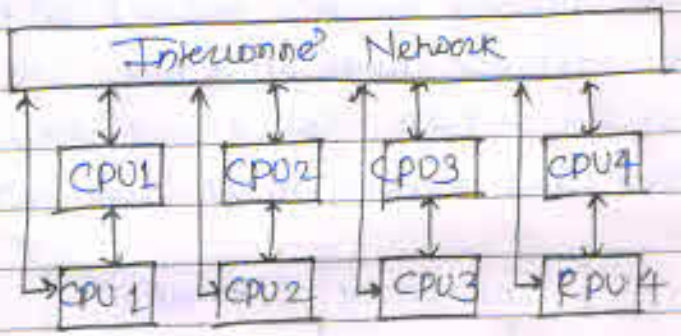
* Share-Address space :-

-> In this, it provides common data space accesible by all processor included in the system.

- This shared space is used to provide interaction among nos of processor by modifying data objects.
- The sys is also called as multiprocessor system. bcz it support the ||el proging approach team as single program multiple data (SPMD) proging.
- This type of platform uses seperate mem associated with the processor. or common memory unit globally available for all the processors included in multiprocessing environment.
- The share address space platform is classified as: NUMA & UMA.

NUMA : Non Uniform Memory Access :

- It allows memory access to every processor w/o any restriction.
- A block of is attached to the processor and all block of mem can be accessed thro the path provided by the use of interconnect n/w



NUMA.

- A processor has direct path to the block of memory attached to it.

eg. If accessing MEM block MEM1 from CPU 1 will be much faster than accessing block MEM2 from CPU 1.

- It has significant implication :

i.e. If we map address carefully it may be possible to keep most of the info required by a processor in the block attached to it.

- Therefore, The CPU can access that memory directly and reducing the contention for the common Bus. Since the time to access a mem loc depends on whether it is attached to invoking CPU or not

This model is called NUMA.

* UMA : (Uniform Memory Access) :

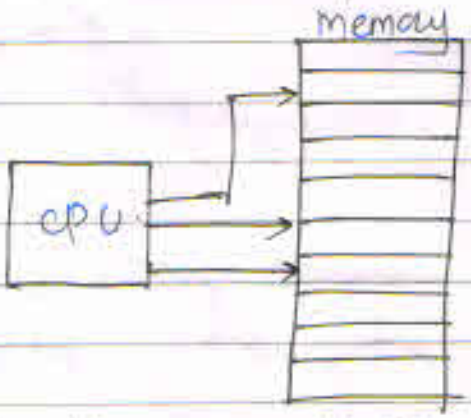
- In this, each processor gets equal priority to access the main memory & the m/c is called as UMA.
- In fact, memory access by each processor is identical in UMA platform. as shown in fig.

UMA

- The prog'ing is much easier in such platform because & availability & global memory space in the system.
- The coding for read only instn among the nos & progs running in different processor is not at all seen by the programmers.
- This happens bcz coding for such platform is similar to coding usually done in serial prog for single m/c.
- The interaction among the read/write operation are handled requires the use & mutual exclusion or some other tools for synchronization

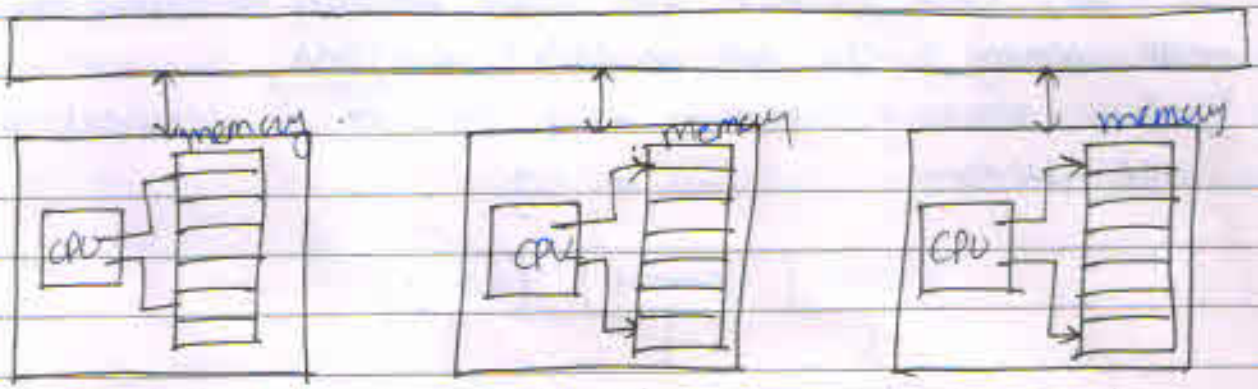
* physical Org of ||el platforms :

1) Architecture & Ideal ||el computer



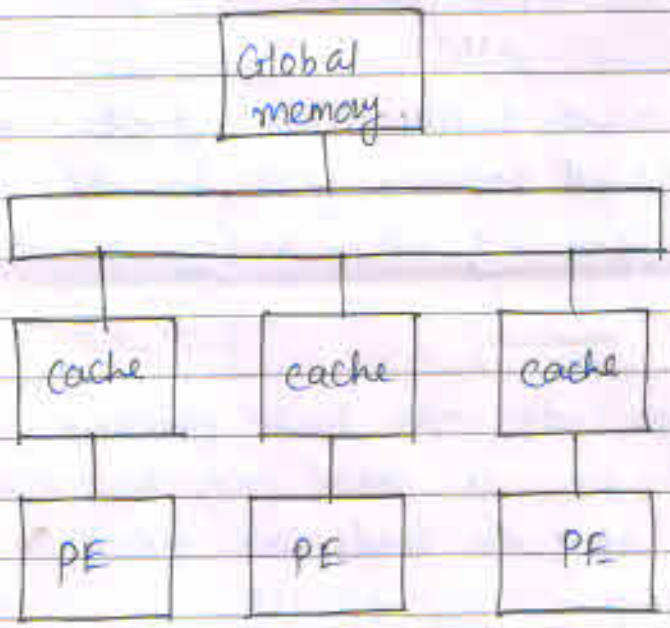
Von Neumann Architecture Machine.

* Multicomputer based ||el computing m/c model



Multicomputer based ||el computing m/c model.

* Ideal Model for ||el Computing :



Ideal Model for ||el Computing System.

* Interconnection n/w for Parallel Computer :-

→ Def :-

" Parallel Computing system consist of more than one processor and these processing element is connected to memory unit directly or indirectly.

Interconn n/w is needed to route data when processor needs to access memory structure.

- classification of Interconnection Networks :-
- It is divided into static & dynamic classes.
- The connection bet i/p & o/p nodes are fixed for entire duration of comm is called as static n/w.
- e.g. linear array, ring, tree, star, mesh, hypercube etc.
- The connection bet the i/p and output is not fixed (variable) is called as Dynamic n/w.
- e.g. Buses, Crossbar switches & multistage n/w, mesh n/w etc
- In static interconnection n/w, nodes are connected using point to point comm links.

Static n/w is also called as direct network.

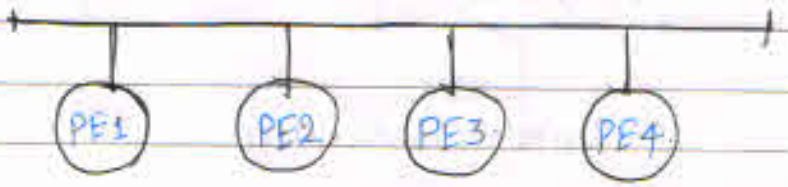
- In the other hand, dynamic interconn n/w, is built up using the switches and comm link.
- Dynamic n/w are refered as Indirect n/w.

* Network topologies :-

- N/w topology are graphical representation of various arrangement of nodes and comm links among nodes.

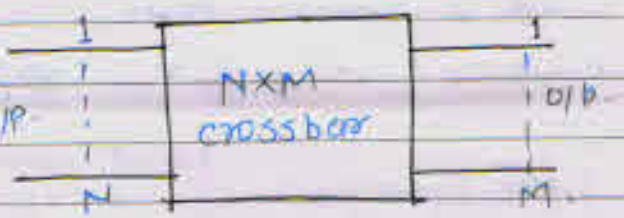
* Bus Based Network :-

- This is simplest type of n/w (interconn) used and static by nature.
- In this n/w, all elements share common comm link as shown in fig.



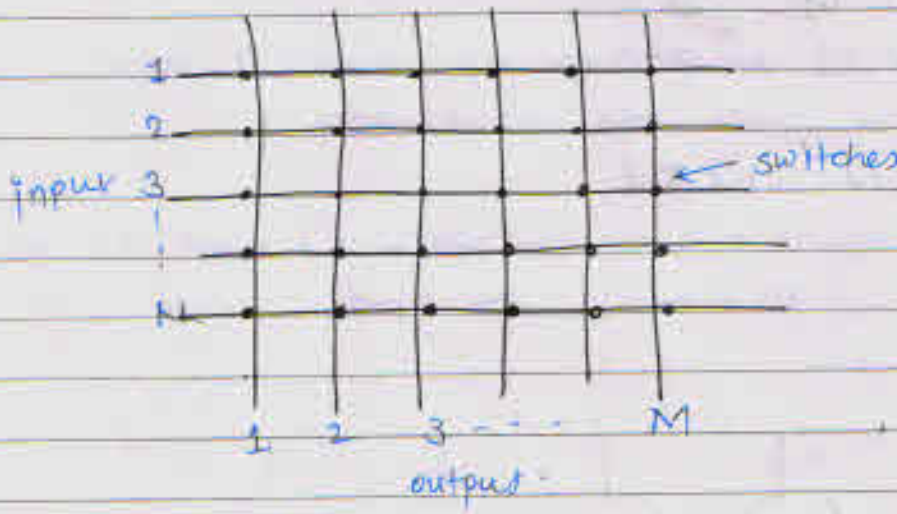
- this is most inexpensive n/w. Ne nodes can easily be added & deleted from this n/w.
- this n/w always has requirement for handling a conflict when several node elements request for the Bus at the same time.
- Ne Bus Controller mechanism is used to provide bus access on a FCFS basis.

* Crossbar Network :-
- This is one of the simplest n/w topology used for interconnection among computing nodes.
- It consist of 2-dimensional grids of switches.
- this n/w provides non-blocking connectivity bet i/p and o/ps.
- This type of interconnection n/w provides the characteristics in which any of the i/p can join to any of the o/p.



NXM Crossbar.

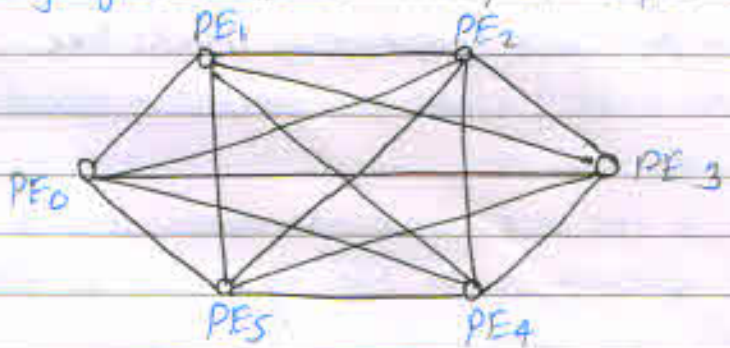
- The connection establishment is done thro' a cross-point for particular row i/p and a particular column o/p.
- The no of switch requirements for a n/w with N i/ps o/p is N^2



Switch Connection.

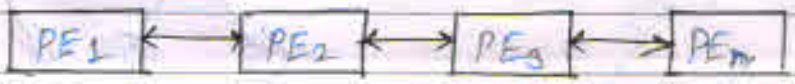
* Fully- Connected Network :

- This is one of the most powerful interconnection topology used for providing connectivity among nodes.
- In this type of topology, each node is directly connected to all other nodes.
- The shortcoming of this n/w is that, it requires too many connections.



- Linear Array :

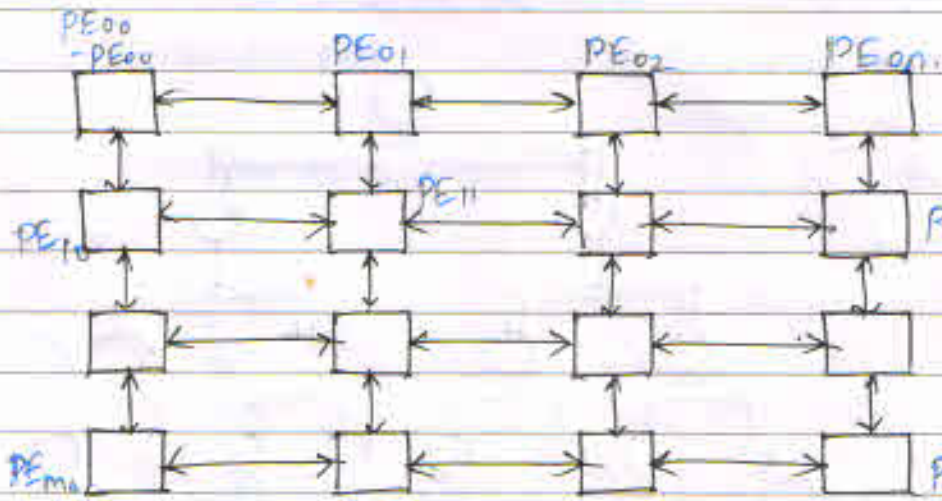
- It is one of the fundamental interconnection n/w pattern.
- In this type of n/w, processors are connected in 1-D linear array.



Linear Array

- This is 1-D n/w, where 1st & last node connected with only one adjacent processor and all the intermediate elements are connected with 2 adjacent processing element.

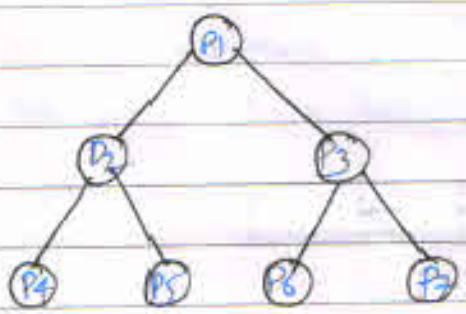
* Meshes : A mesh is 2-D n/w. In which processing element are arranged in 2-D grid. The row & col. posi? are used to denote a particular processor in the mesh n/w.



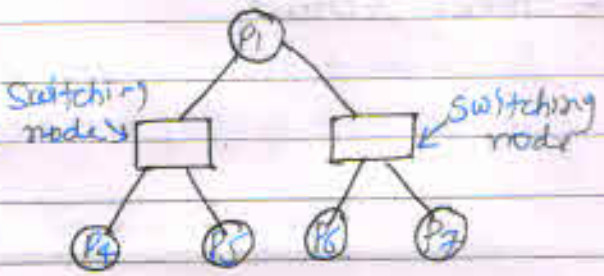
Mesh Network.

* Tree Based Network :-

- In this, only one path is used bet' any pair of nodes.
- In this, linear array & star connected n/w are considered as special case of tree n/w.
- In this n/w, processor are arranged in a completely like Binary-tree pattern.
- It is divided into 2 category. 1) static tree 2) Dynamic tree.
- When each node of a tree are processing elements we call it as static tree n/w.
- Whereas, in the case, when the nodes included at the intermediate level are switching nodes the tree n/w is considered as a dynamic tree interconnect n/w.



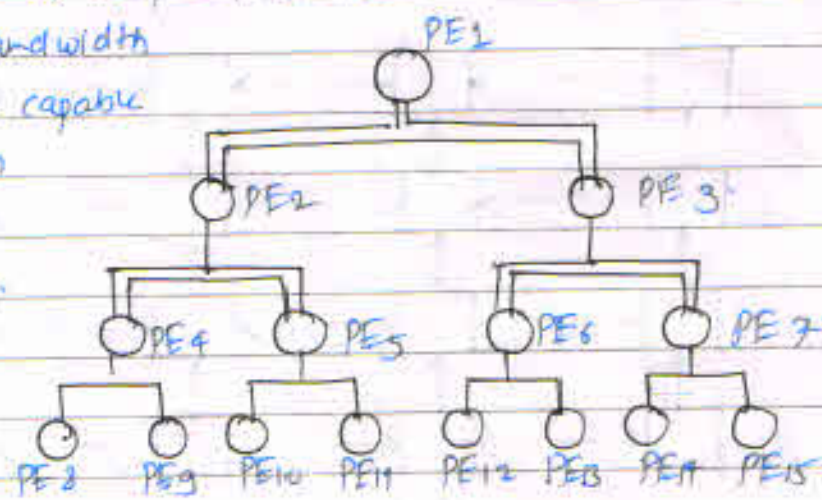
A tree n/w static in nature.



A tree n/w dynamic in nature

* Fat Tree :-

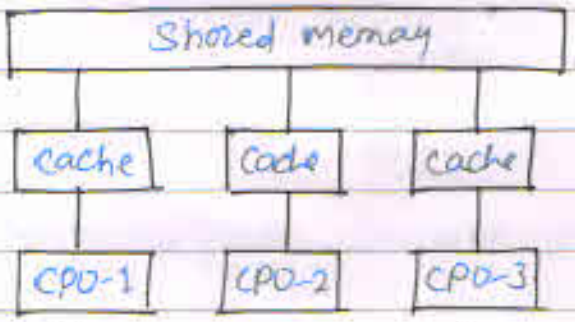
- This type of n/w is modified form of the original tree n/w.
- This has increased bandwidth of edges into root direction.
- This simulated on the basis of actual trees where branches get thicker towards root. It provides more adaptability bcz practically more traffic occurs towards the root as compared to leaves.
- therefor, the larger bandwidth provided at root side is capable to handle the traffic in effective ways and No bottleneck can occurs.



Fat Tree.

* Cache Coherence in Multi-processor Systems :

- The shared memory multiprocessor system equipped with a seperate cache memory for each processor.
- Such sys creeps many copies of data and instns in the fashion that one copy is kept in memory and one copy in each cache memory.
- when copy is changed by the processor other copies must also reflect with changes.
- Here cache coherency deals with the changes in the shared data should propagate throughout the sys in a timely fashion.



Multiprocessor system with shared Memory.

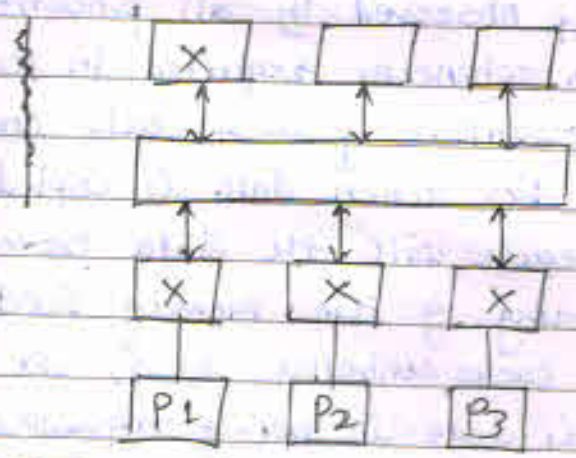
* Cache Coherence Problem :

- In the shared memory, private cache is associated with each processor. In such situation each cache contains multiple copies of the same data.
- The problem occurs when all processor are allowed to update the data independently. This is called as cache coherence problem.
- The sys is called coherent only when every read operation results in the value which is updated by previous write operation, even by the process at any other processor of that system.
- The sys must be able to maintain a coherent view of memory. It is required to maintain consistency of data in shared resources & individual local cache memory in the system.
- The dealing with this consistency issues is called as cache coherence.
- The coherence problems may happen in multiprocessor system where each processor have local cache as well as can access global shared cache of the system.

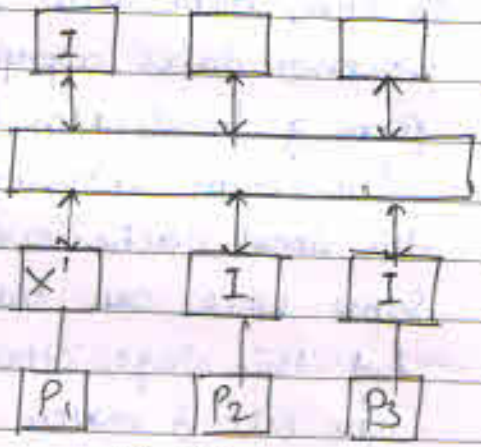
- The basic things included with coherency associated in multiprocessor system are migration and Replication.
- The genuine drawback while accessing shared data that is not available locally are latency & bandwidth.
- Cache coherence protocol are implemented at h/w level instead of the s/w because of the efficiency of the access.
- The cache coherency protocol is used to keep records about the shared data blocks and their associated status.
- Snooping and directory based are 2 types of coherence protocol.

* Hardware-based protocol :- The basic policies used for maintaining Cache Consistency are Invalidate & update

* Write - Invalidate.

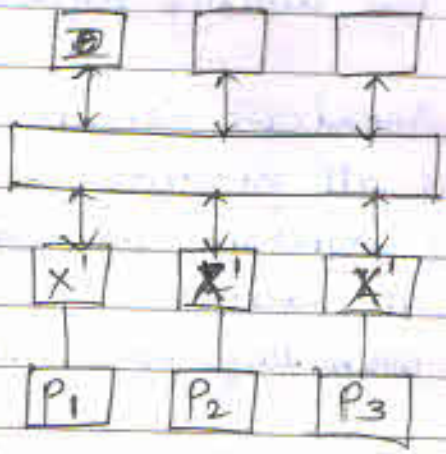


Four copies of block X



Copies of P2 & P3 are invalidated.

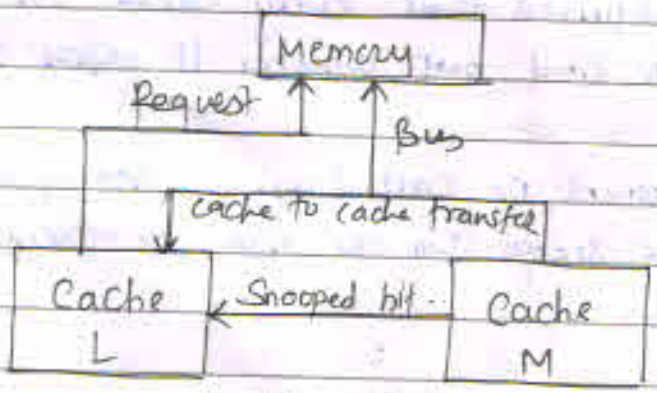
* Write - Update :-



Write update policy

1) Memory copy is updated or not - it depends on imple^n of protocol.
2) In this, if processor modifies data, it reflects to the all processor / system.
3) Actually processor cache has ability to broadcast changes to others.

* Snoopy Cache protocol :-



Cache Coherence.

- It is suited for multiprocessor sys. with shared memory & interconn^n among the nodes

- It is used in bus based System. It is suited for Multiprocessor System
- In this, mem. trans' is carefully observed by all processor. processor takes appropriate action whenever required in the form of invalidation & update operation of local cache content.
- The term snooping is used bcz when data is copied to the local cache every other cache with the data from same block can track the sharing of the memory block.
- At this stage, other cache or cache controller snoops via. the bus & watch about their data is being requested by other cache.
- In diag, cache L & M are connected via a bus.
- The data transfer occur bcz a cache when cache M receive a Snoop hit from cache L, which has actually requested it from memory.
- The shared bus has property to broadcast Coherent info' in very fast manner among all processors.
- This multiprocessor system strictly maintain consistency of data by updating info' among all processors
- The shared bus become bottleneck when large nos of process is included.
  This situation can be handled by increasing bandwidth.
- In some interconne' n/w, like Bus n/w
  it is feasible to broadcast Consistency cmds to all Cache..

In this it is required that every cache should execute Consistency cmd to find out whether it refers to data in cache.

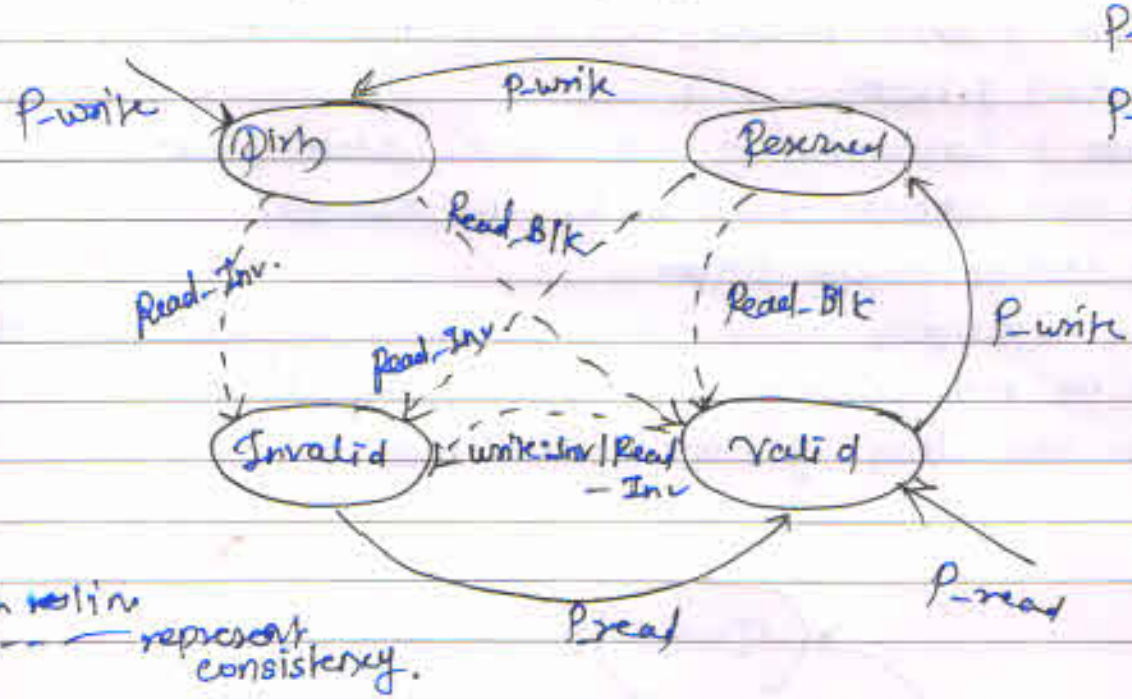
- This type of protocol is called as "Snoopy Cache Protocol" because each Cache Snoops On the n/w for every incoming Consistency cmds.

* Write-Invalidate Snoopy Cache protocol :

→ 1st write-Invalidate Snoopy protocol is Write-once protocol.

- It provides assumption of State with exact copy of block. These State are named as Invalid, Valid, Reserved & Dirty for copy of blocks.

1) Invalid - When copy is inconsistent
2) Valid - copy is consistent
3) Reserved - when data is written exactly once and & copy is consistent with memory.
4) Dirty - Copy is only one in the system but data is modified more than One.



Write-Once Protocol State transi diag of cache copies.

- The copy-back mem update policy is used in write-once.
- It indicates that entire copy of data blocks must be written back to the memory when it is replaced.
- This oper is performed only when the data has been modified during its cache residence time. (i.e. state is dirty)
- The normal mem. cmd read block (Read_blk) and write block (write_blk) are applied & besides these cmds other Consistency cmd required to maintain consistency are used

* These cmds are :

1) Write_Inv - invalidates all other copies of blocks.
2) Read_Inv - used to read a block and all other copies are invalidated.

* Write-Update Snoopy Cache protocol :

This protocol has 3 state.

1) valid-exclusive :

it indicates about only cache copy and this copy is consistent with the memory copy

2) Shared :

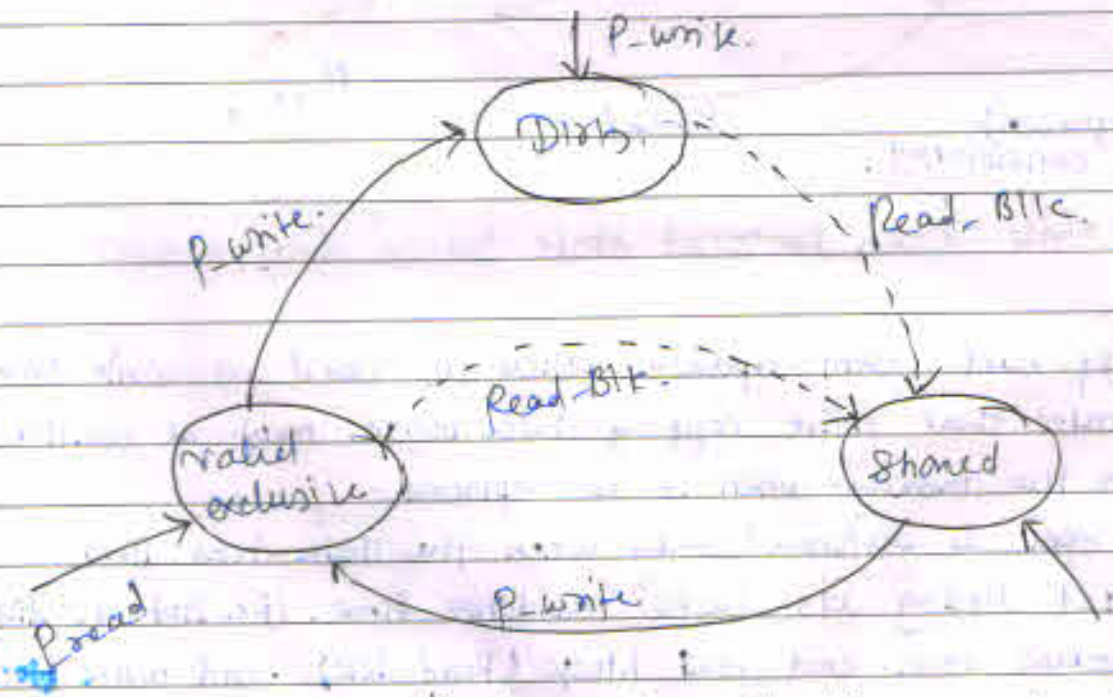
The copy of data to be shared is consistent and there are existence of other consistence copies also.

3) Dirty :

This is the only available copy of data & in this state the inconsistency of the memory copy of data is assumed.

There are 2 diff policies used by Firefly protocol for diff types of blocks

- it has implementation of copy-back-update policy for private block whereas shared blocks are supported during write-through policies option.
- write consistency update cmd is used to update all the copies for maintaining of consistency in the system.
- State trans diag of this model is shown in fig



State trans diag. for Firefly protocol

1) Instr level ||elism (ILP) :-

→ modern procer has abilit to execute several opers g a prog simultaneously.

- ILP is the measures about how many oper associated with a prog. can be performed simultaneously.
- This is handled by overlapping the instr g progs during exe. and is actully termed as ILP.
- In this, multiple instr from same instr stream execute concurrently.
- Such instr are generated & mnged by h/w level e.g. Superscalar procers / compiles it generates instr to be executed Concurrently. as an example g VLIW.

2) Thread-level ||elism :-

- Thread-level ||elism occurs when multiple thread or instr sequences in a same applicatn executed Concurrently
- In this, prog. is divided into independent parts in which each part is called thread. and these threads execute Concurrently.
- The ||elism occurs here bcz thread can have the effect g seperate independent prog. running together and seperate activities are being performed inside the same ||el prog.
- This type g ||elism is used in MIMD class g m/c.
- There are 2 types 1) fine grained 2) Course grained multithreading
- Thread level ||elism comes under task-level ||elism, bcz it is based upon org g prog. or Computing sol into set g thread for simultaneous exe. on different procesor.

* Transachon level ||elism :-

A trans is seq. g infor exchange and related work considered as a unit in prgming.

The term trans level ||elism is used to describe Concurrent exe g multiple procesors and threads from different transactions.

* Models :-

1) SIMD :- 2) MIMD. 3) NUMA 4) UMA.

we have already discussed this point

* Comparision of SIMD & MIMD.

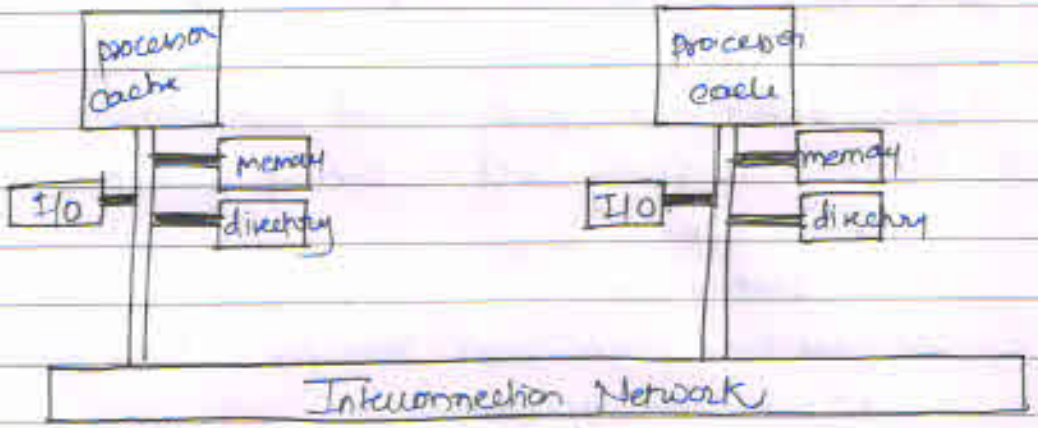
SIMD	MIMD
1] SIMD archi. are simple.	1] MIMD archi are complex.
2] low cost	2] medium cost
3] only one prog. copy is required to be stored	3] No of prog. copies depends on the no of PE. Each PE store it's own copy of prog.
4] control unit contains only one decoder.	4] control unit is availble in every P.E.
5] provides scalability in terms of size & performance.	5] It is complex in size. but provides good performance.
6] Sync is required implicitly at prog level.	6] Handling of synch. requires explicit data structure & operation

* Dataflow model — Asynch exe, no sequencing, Dataflow Repres
* Dataflow graph — operator, decide, switch actor.

* multithreaded Archi - single, multi-core.

* Directory Based Scheme:

- This is cache coherence protocol that is not based on broadcasting of data. This protocol stores the loc of all cached copies of every block of shared data.
- The cache loc can be kept at centralized place or at distributed manner and are called directories.
- used in scalable cache-coherent distributed memory multiprocessor system where cache directories are used to keep record on where copies of cache block reside.
- The directory based protocol is shown in fig.



Directory Based Protocol Organization.

- In this two basic opn called handling read miss and handling a write to a shared cache blocks are implemented in this protocol.
- The impln of these opn are possible if the directory track the state of each cache block.

The diff. state considered here are

1) Shared - The value in the mem. is up to date of the block cached by one/more processor.
2) Uncached - processor do not have copy of cache block
3) Exclusive - Exactly one processor own copy of the cache block & it is designated as owner. In this state, the memory copy becomes out of date bcz the owner has updated the block.

* Comm Cost in ||el machines :-
- The ||el computing platform provides facility for executions of ||el progs.
- The various module of these ||el progs execute in diff nodes of distributed Computing based ||el system or in diff processing elements of a shared memory ||el system.
- The module need to communicate with each other for completion of computing tasks.
- The comm cost among these task depend on several factors like interconnn n/w used for connecting nodes, modes used for ||el praming, protocol used in comm etc.

--Issues affect the Overall Comm :-
1) Message passing Cost :- Theory you can write your own. factors are :-
startup-time, selc of routing algo., interface establishment per-hop time, per-word transfer, Store & forward-routing and cut-through routing.
2) stored & forward Routing
3) packet Routing
4) Cut-through routing.

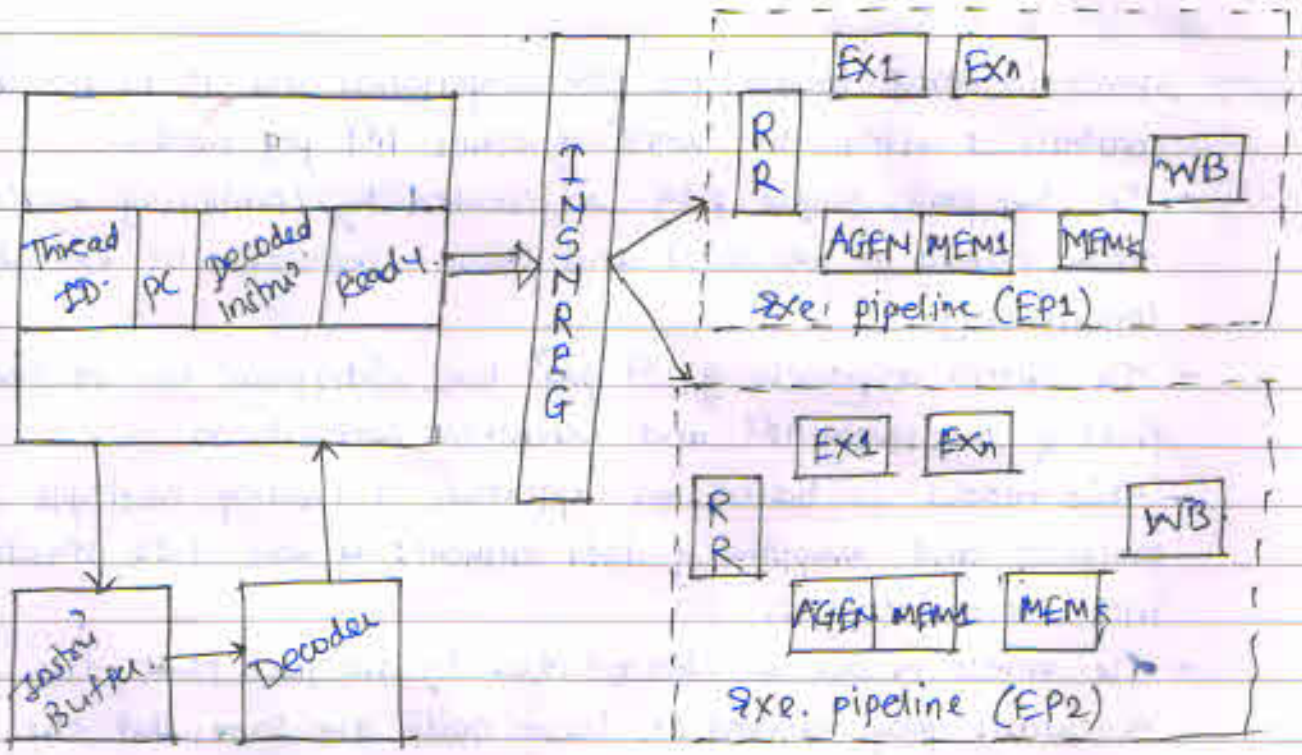
* Level of ||elism :-
- There are 2 types 1) H/W ||elism 2) S/W ||elism.
- In h/w, it includes activities provided at architecture level itself.
- In s/w, it is divided into ||elism based on task & data level.
- The types of ||elism in applications are -
1) ~~level~~ Instruction level
2) Thread level or Task level
3) Transaction level ||elism.

* SIMT :-

- The single instru multiple thread archi. has been developed to achieve throughout Computing with high energy efficiency.
- In todays computing field, SIMT processors are used most of the times for Graphics Procesor Units. (GPUs)
- The SIMT archit. is created with the combined supports from h/w & s/w sides.
- The h/w archi. side in SIMT provides an approach for the Conversion of scalar instru into vector-style single-instru-multiple data [SIMD] procesing for the achievement of energy efficiency.
- On the other hand s/w side, the SIMT prg model for e.g CUDA and OpenCL provides facility to achieve data lplism. to be implemented as task-level lplism.

* Micro-archi of SIMT :-

- The micro archi of an SIMT core is shown in fig. An instru is selected & issues from ready warps in warp scheduler to the multiple exe. pipelines.



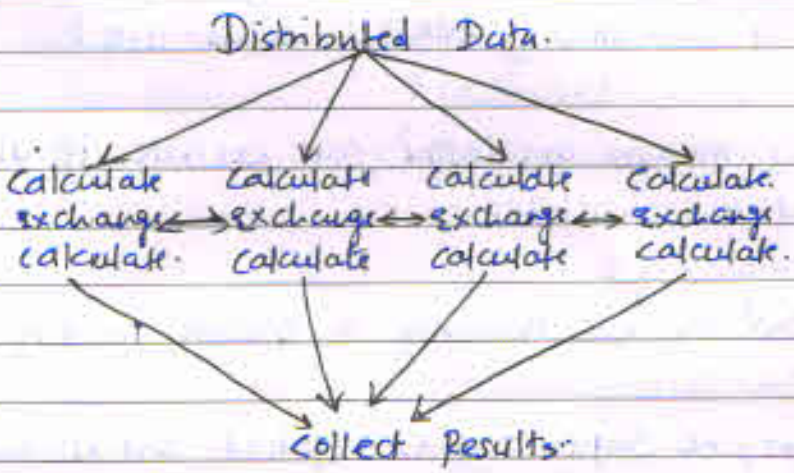
Micro-Archi. of an SIMT core.

- These multiple exe. pipelines are also called as streaming processors or processing elements (PEs)
- A warp is considered as a ready warp when all dependancies requires for the exe. of next instr^{n} have been resolved.
- In the archi. each entery of warp scheduler contain the instr^{n} of warp. The instr^{n} from instr^{n} cache is read using Prog. Counter.

* SIMT prg^{m} model:-
- The data 11^{el}ism is expressed as task level according to the SIMT progn model.
- The prg model here follows the - Single prog. multiple thread paradigm this indicates that all threads shares the same progs.
- The scalar code commonly called as kernel code is written by an appli^{n} developed to utilize the paves of SIMT machines.
- All thread execute the same kernel fu^{n} and are also refered as work items. Each thread identifies the data to be operated upon using the unique identifier.

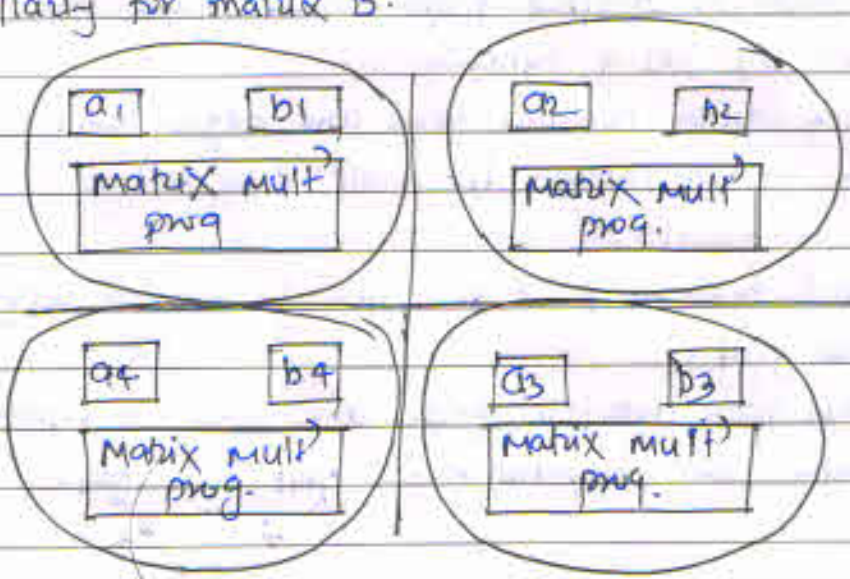
* SPMD (Single prog Multiple Data):-
- SPMD is prog model for 1^{el} computation and it is possible to construct it using the comb^{n} of other 11^{el} prg model.
- In this prog, single prog. is created by combining multiple task (thread or process) and these task can be executed concurrently.
- The SPMD approach of 11^{el} prg^{m} has widespread use in the field of massively 1^{el} and scientific computation.
- This model is based on approach of having multiple processor and mapping of data elements of the data structure into these processors.
- The SPMD model is charachized by the fact that the executable prog running on these node are same but the data upon which the step of these prog. operate are different.
- In this, it splits appli^{n} data among various processor. This type of 11^{el}ism is refered as geometric 11^{el}ism, domain decomposition or data 11^{el}ism.

- The overall data used by appli is divided and given to a set of processes included to solve the problem.
- One can easily understand SPMD by the use of an e.g. Consider matrix multi prog on 4 nodes of ~~dist~~ computing sys.



Base structure of SPMD model of computing.

- In this arrangement, 3 nodes are given exclusive responsibilities to act as slaves whereas one node act as master as well as involve in computation as a slave also.
- Also assume 2 matrixes A & B of size 100X100 for multip. The whole matrix can be divided into 4 chunks with 25 elements in each bcz ~~each~~ overall 4 nodes are available for computation.
- Matrix A is divided into 4 chunks as a1[25][25], a2[25][25], a3[25][25] a4[25][25] where a1 contains 1st 25 element, a2 next 25 & so on. Similarly for matrix B.



Matrix Multiplication as a SPMD example.

* Data flow Model :-

- The dataflow model of comput is based on graphical repres of prog. in which oper are represented by nodes and arcs are used to represent data dependancies.
- The ||el activities under dataflow model of computing provides various property to solve the computing prob in efficient way.
- Some of properties of dataflow model are :-

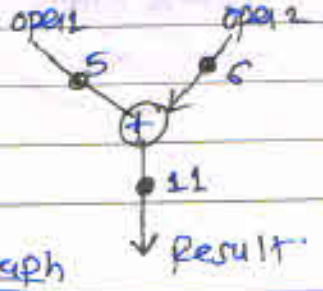
1) Asynchronous Executions :-
   - This means an instn can execute if all its required operand are available.
2) No sequencing :-
   - Instn are not necessary to execute in any sequential fashion.
3) Dataflow Representation :-
   - In this, no sequencing is required and it makes the possibility of dataflow repres of a prog.
   - The dataflow repres of prog. provides the use of all forms of ||el prog. exe. w/o the assistance of any explicit tools of ||el exe.

- Dataflow Graph :-
  - It is called of nodes & arcs. The dataflow computation works on dataflow graph. As we know, compute understand only m/c level lang. In similar, dataflow computer also works but m/c level lang repres of dataflow computer is dataflow graph.
  - Dataflow graph is directed graph which shows the data dependancies betn nos of function.
  - A dataflow graph contains node and edges where each node of dataflow graph contain i/p & o/p data ports.

    Result = (oper1) + (oper2).

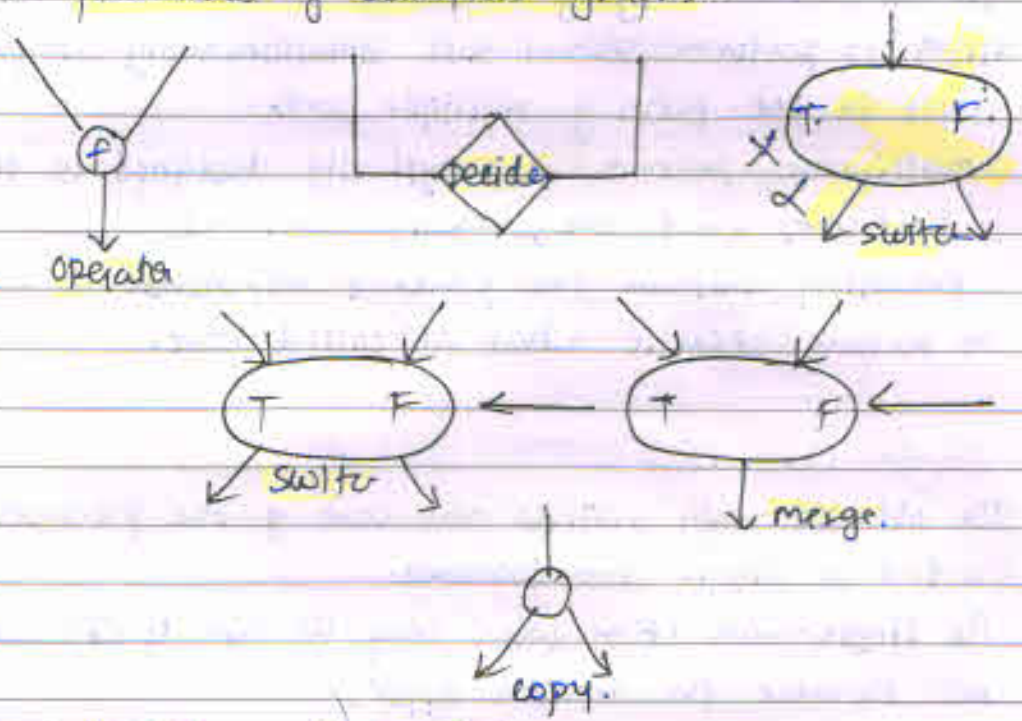
  - The connn betn o/p & i/p port are represented using the edges of dataflow graph.

In this way dataflow graph are used to represent prog for data flow computation.



Dataflow graph

* Basic primitives of dataflow graphs:



- operator :- It perform some oper?
- decider :- It generates true/false value
- switch (merge) :- used for directing data values. either by switch or merge
- copy :- An identity operator called as copy is used to duplicate i/p token.

* Architectures - N-wide Superscalar architecture, Multi-core and Multi-threaded :-

* Multi-threaded Architecture :-

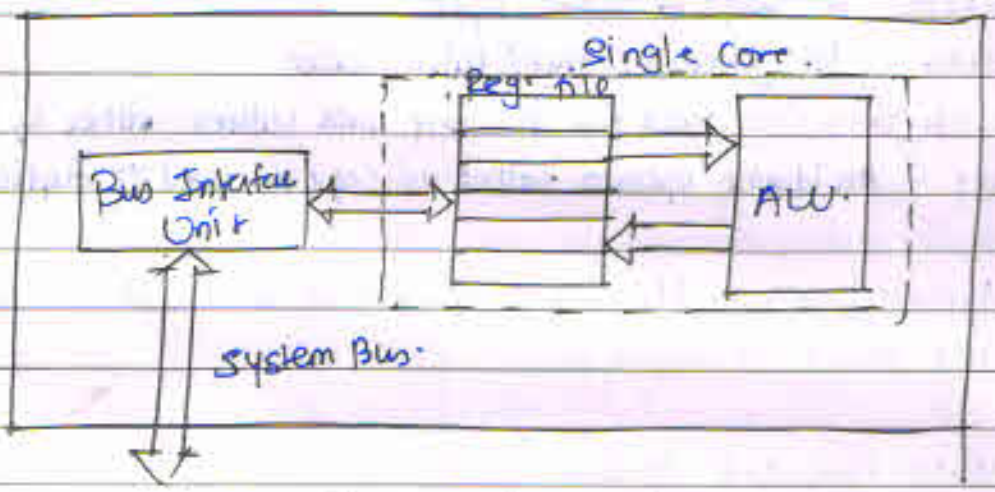
- The thread in association with the h/w context is diff. than the s/w threads in multithreaded o/s. The h/w supported thread depends on specific form of multithreaded procesor.
- The h/w based multi-threaded tech enables TLP at process level. This is made up of duplication of architectural state on each procesor while sharing one set of procesor exe. states.
- The diff. archi. set are considered as seperate logical procesor by the OS during exe. of multiple threads.
- The OS schedules the diff. threads as unit of exe on logical procesor.
- The logical procesor are created thro. h/w for sharing the procesor's functional unit among diff. threads.
- The sharing of resources is crucial in multithreaded archi. bcz more nos of sharing resources bet logical procesors provides more efficient oper of multithreading level.

* Multi-core Architecture :-

- In the modern day g computing procesor are constructed ~~itself~~ to perform several task simultaneously at procesor level itself in the form g multiple cores.
  multi-core procesor are typically designed in the form g. dual-core, quad-core, many-cores etc.
- Procesing unit in the procesor responsible to read instn & to perform specific actn is called core.

1) Single-Core CPU :-

- The old CPU only utilizes one core g the procesor is called as Single Core procesor.
  The single-core CPUs were very 1st type g CPU and not suitable for modern appl?.

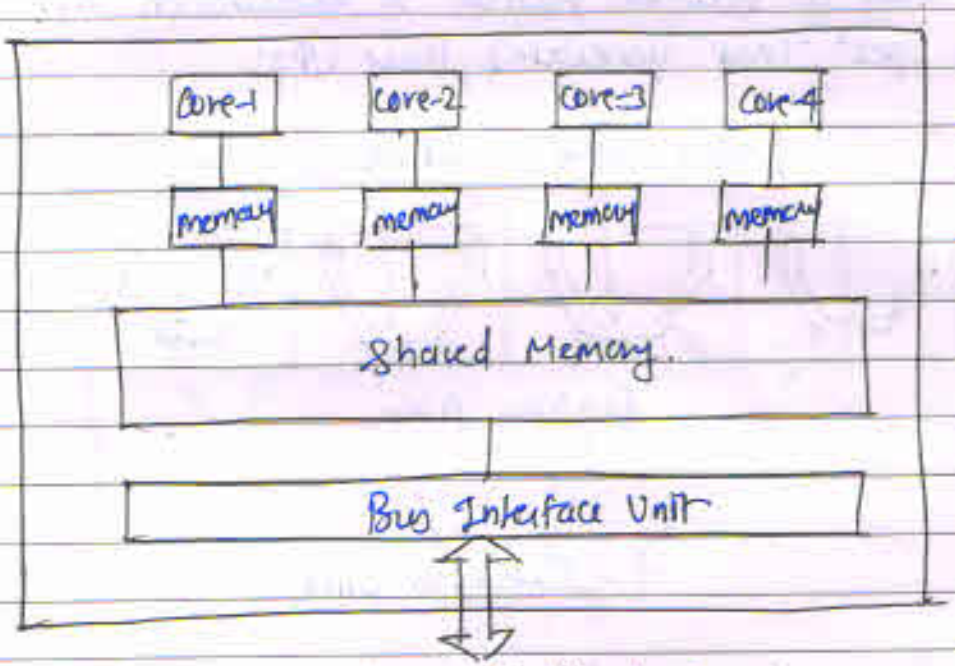


Single-Core chip.

2) Multi-core CPU :-

- The multi-core procesor contain more than one distinct core in a chip. In this way, if a chip has 2 cores. it is named as dual procesor Similar quad core procesor it contains 4 procesors in it (core).
- The multi-core CPU design implements seperate exe. pipelines for each & every cores.
- This means each core can run thread g exe. w/o. requiring any resources from other cores.
- multi-core procesors are considered as MIMD category g. mc bcz. multiple instn are executed by diff cores.

- It also supports Shared memory multiprocessor because all cores can share the same memory.



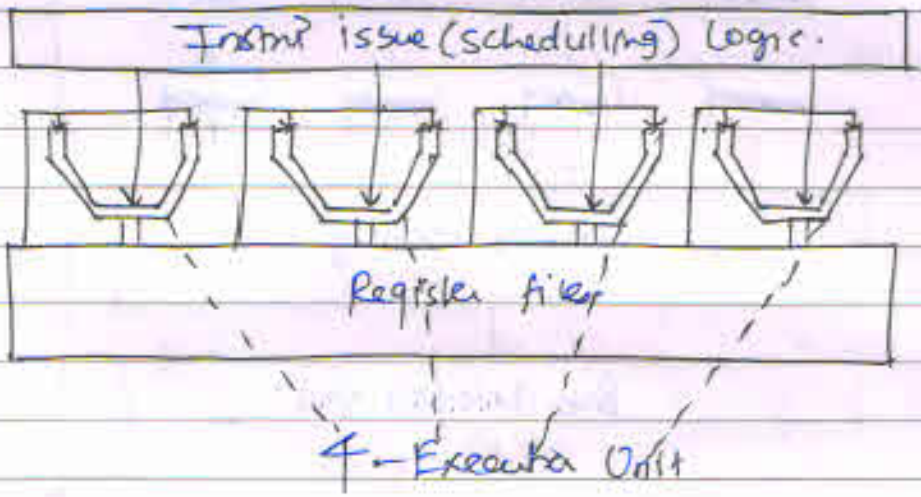
Multi-core processor.

- Intel xeon is multi-core processor packaged with the logic & circuitary for 2 or more Intel xeon processor.
- In fact, multi core processor is based on packaging of more such processor in single physical processor.
- The advantage of multi-core processor is the improvement in performance by using multiple cores to run multiple tasks simultaneously.

* N-wide Super scalar Architecture:
- The superscalar archit implemented to support issue of one, two etc instru and accordingly 4-wide, 6-wide. Superscalar architecture are designed by various vendors.

* Superscalar archi. processor:
- The superscalar archi. contains multiple exe. unit for instru exe.
- In this, single centralized reg. file is used to read operands from it and write results into it by each exe. unit.

- The result written back to the register file by the exe. unit becomes visible to all of the exe unit on the next cycle. This way it becomes possible to executes on diff. units from opers that generates their i/ps.



Instruction issue logic & 4 Exe. Unit in a Superscalar.

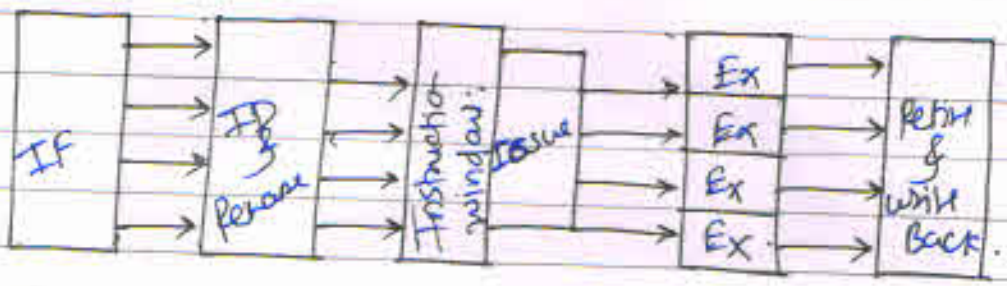
* ILP in superscalar processor :
- The superscalar arch. uses complex bypassing h/w to reduce delay bet dependent instru.
- The delay is reduce bcz bypassing h/w forwards result in instru exe. to all exe. units.
- The superscalar processor contains the instru issue logic which is used to store instru of program. This instru issue logic provides instru to the unit in H/W.

* Instru issue logic in superscalar processor :
- The changes in control flow in prog. exe. due to branches occur simultaneously across all of the units.
- The prog. development under superscalar processor becom much easier bcz. the occurances of simultaneous activities across all units.

* H/w used in superscalar processor :
→ The ILP is extracted from a sequential prog from the h/w unit used in superscalar processor.
- The instru issue logic of superscalar processor examines the instru in the seq. prog. to determine which instru may be issued on that cycle.

* ILP strength & weakness :-
- The superscalar processor executes instru in ilel and in this way it is possible to achieve significant speedup while executing diff. types of progs.
- The max improvement in the performance depend upon the limitation imposed by instru dependancies.

* pipeline in superscalar :-



Ex - Execute/ address calculation.
IF - Instru fetch.
ID - Instru decode / Register fetch.

Superscalar Pipelines.

- The branch prediction logic used in superscalar pipelines provides control independancies for instru in the instru window.
- The reg. renaming is used here to achieve name independancies in the pipeline. This way only true dependancy like data dependancies and structural conflict remain in the architecture to be handled.

PAGE NO.
DATE / /

* Section of superscalar pipeline :-
it is partitioned in distinct section based on the ability to issue & execute instrn.

In-order section :-
With the instrn fetch, decode, rename stages - the issue is also part of the in-order section. in case of in-order issue.

Out of order section :-
Instrn can be executed in an order different from that specified in the prog.