Assignment-3 (CA)

(1) Explain briefly the network characteristics & Hierarchical bus System

Network characteristics

The network are designed with many choices like timing Switching & Control Strategy like in case of dynamic network the multiprocessor interconnections are under program control

Timing

* Synchronous - Controlled by a global clock which Synchronizes all network activity.
* Asynchronous - use handshaking or interlock mechanisms for communication & especially for coordinating devices for with speed.

Switching method

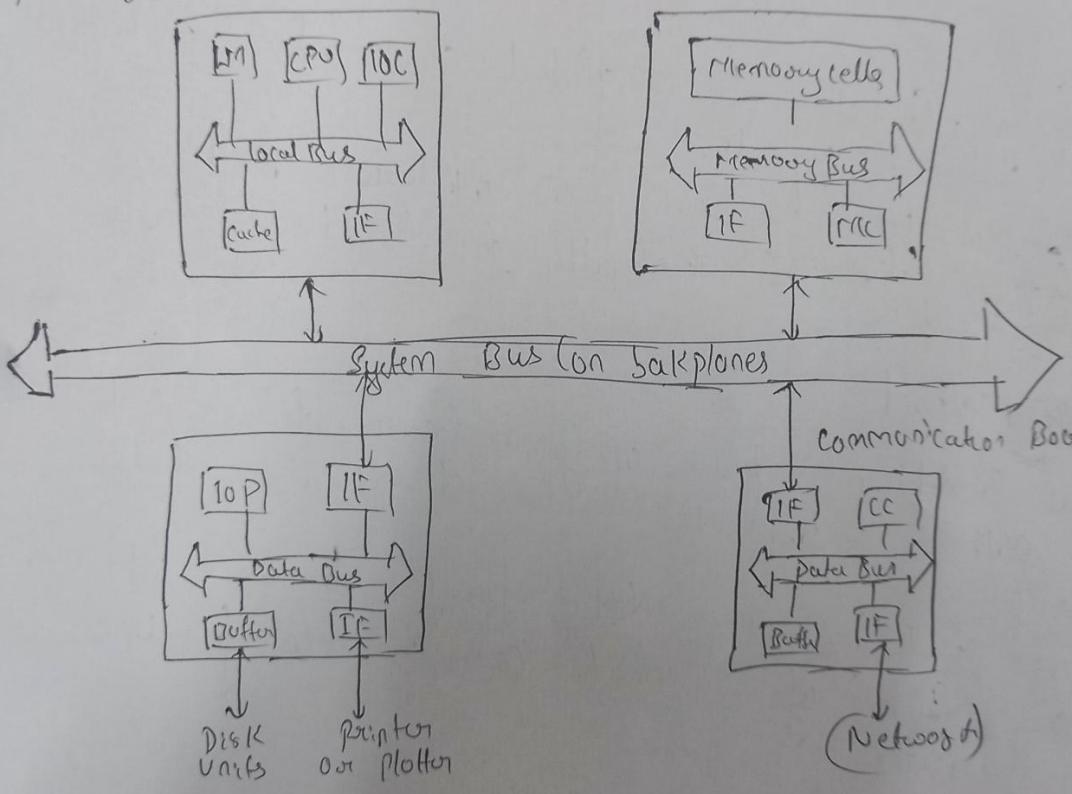
* Circuit Switching - a pair of communicating devices control the path for the entire duration of data transfer
* packet Switching - large data transfer broken into smaller pieces each of which can compete for use of the path

Network Control

* Centralized - global controller receiver and acts on requests
* distributed - requests handled by local devices independently

Heirarchical Bus System

* A Bus System consists of a hierarchy of buses connecting various system and subsystem components in a computer
* Each bus is formed with a number of signal, control, & power lines. Different buses are used to perform different interconnection functions
* In general the Hierarchy of bus system are packaged at different levels as depicted

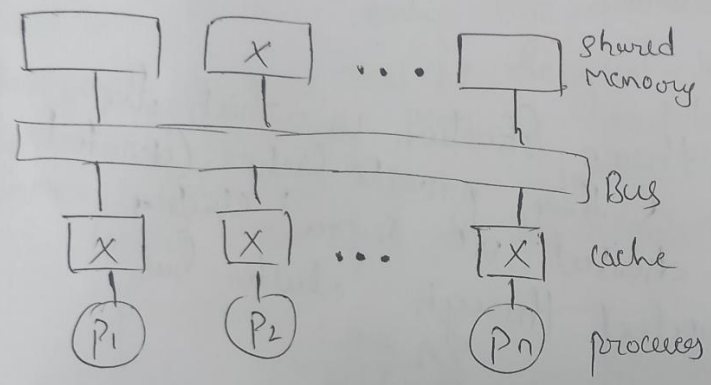


IF (Interface logic) LM (Local Memory)
IOC (I/O Controller) MC (Memory Controller)
IOP (I/O Processor), CC (Communication Controller)

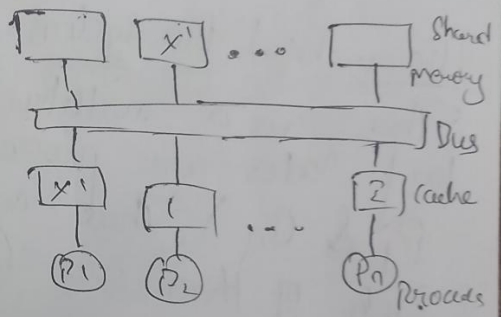
* Local Bus: Buses implemented on printed-circuit boards are called local buses
* On processor board one finds a local bus which provides a common communication path among major components (chips) mounted on board.

(a) Explain the Snoopy bus protocol approach for Cache.

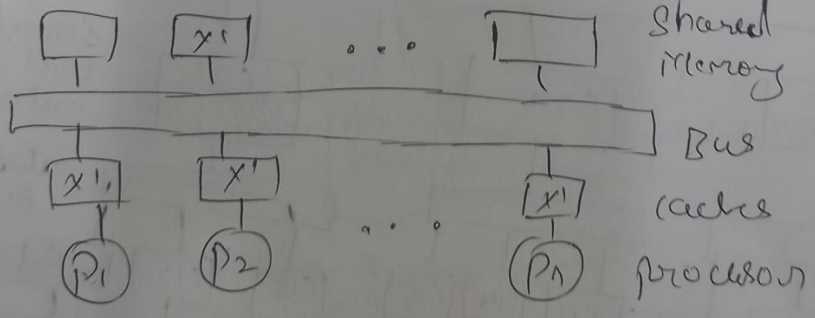
* Snoopy protocol achive data consistency among the cache and shared memory through a bus watchdog mechanism
* In following diagram, two snoopy bus protocols create different results Consider 3 processors (P_1, P_2, P_3) maintaining consistent copies of block X in their local block X in the their local cache.



a) Consistent copies of block X are in shared memory & three processor caches



b) After write - invalidate operation by P_1 ...



* using a write - invalidate protocol, the processor P_1 modifies (write) its cache from X to X' and all other copies are invalidated via the bus Invalidated blocks are called dirty meaning they should not be used
* The write - update protocol demands the new blocks content X' be broadcast to all caches via the bus

Backplane Bus

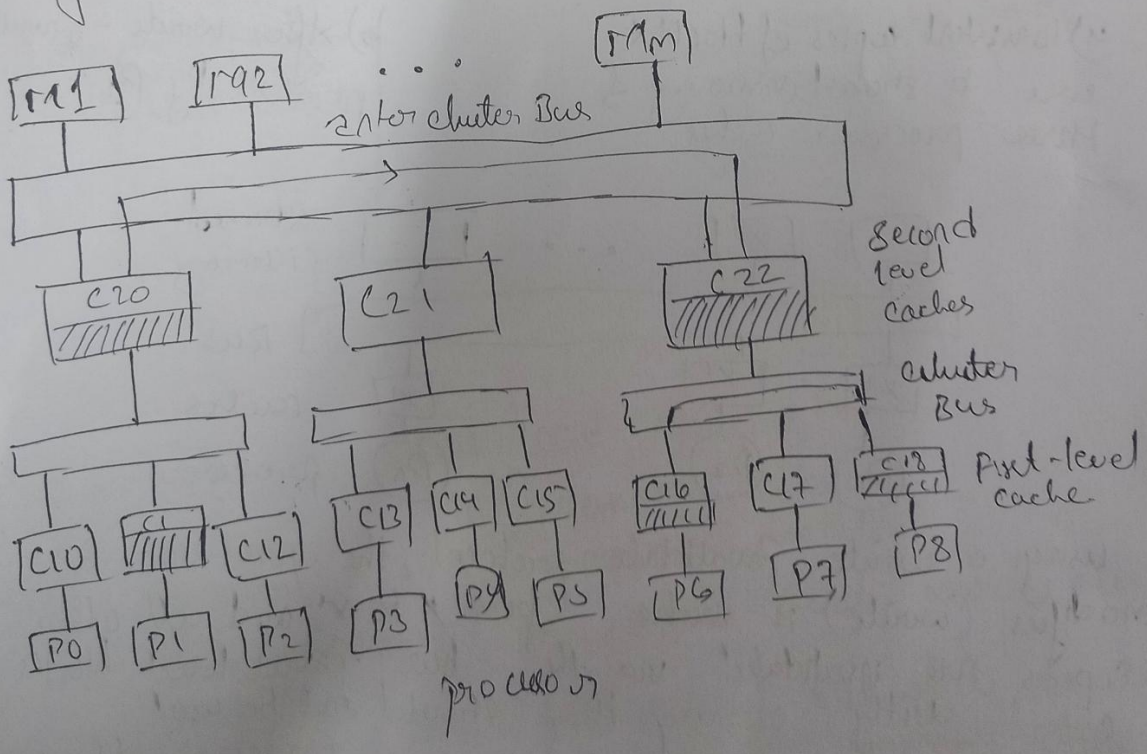
- A Backplane is printed circuit on which many connection are used to plug in functional boards.
- A Systems bus, consisting of shared signal pathsand utility lines

I/p Bus

I/O Bus devices are connected to a Computer Systems through I/O bus such as SCSI (small Computer Systems Interface) bus. This bus is made of coaxial cables with taps connecting disks, printer and other devices to a processor Through an I/O controller

Hierarchical Bus System:

This is a multilevel three structure in which the leaf nodes are processor & their private caches (denoted P_j & C_{ij}) These are divided into several clusters each of them are connected through cluster bus



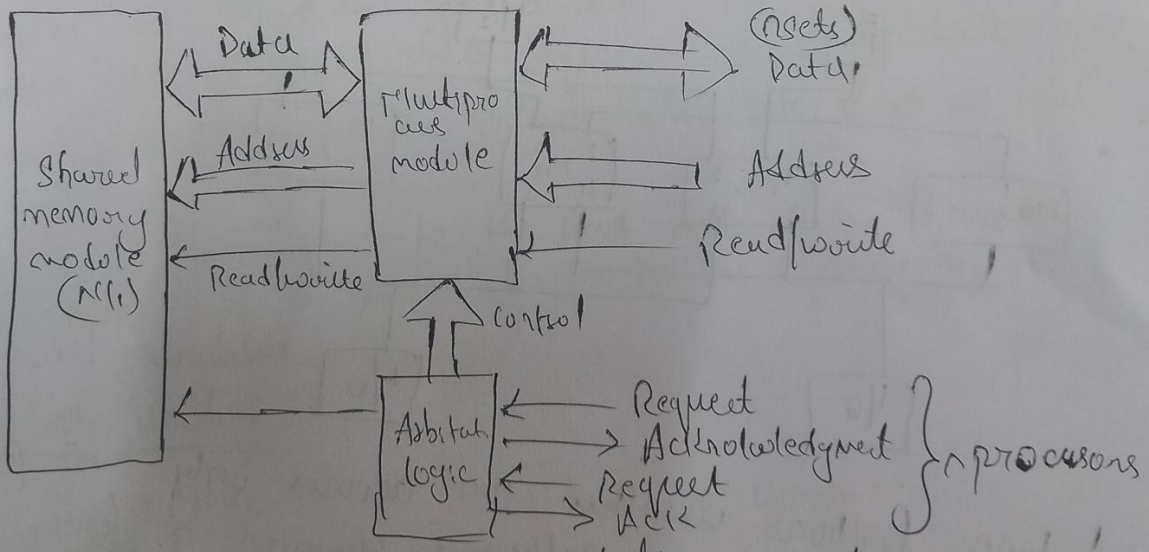
3) Show the architectural characteristics of cross bus switch & multiport memory system

Crossbar Network

Crossbar network connect every input to every output through a crosspoint switch. A crossbar network is a single stage, non-blocking permutation network.

Crosspoint switch Design

Out of n crosspoint switches in each column an n×□n crossbar mech. only one can be connected at a time. Crosspoint switches must be designed to handle the potential contention for each memory module. A crossbar switch avoid competation for bandwidth by O(N) switches to connect N input to N outputs



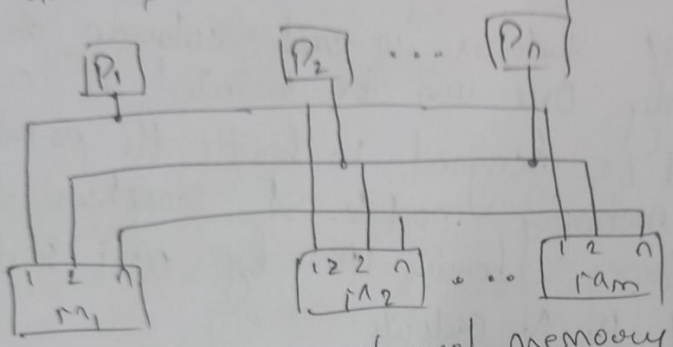
Each processor provides a request line, read/write line. a set address line & a set of datalines line to a crosspoint switch for a single column.

The crosspoint switch eventually responds with an acknowledgement when the access has been completed

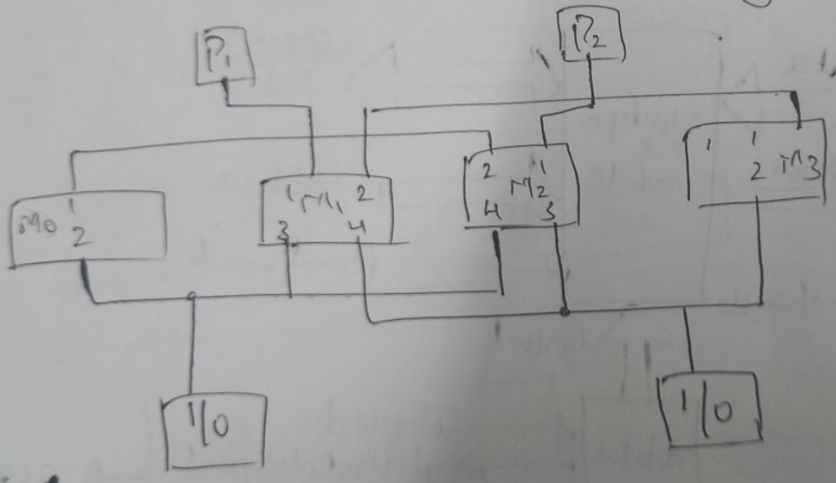
Multiport Memory

Since Crossbar switches are expensive and not suitable for systems with many processors or memory modules, multiport memory modules may be used instead.

A multiport memory module has multiple connection ports for processors (or I/O devices). The memory controller in the module handles the arbitration & switching that might otherwise have been accomplished by crosspoint switches.



M shared memory modules

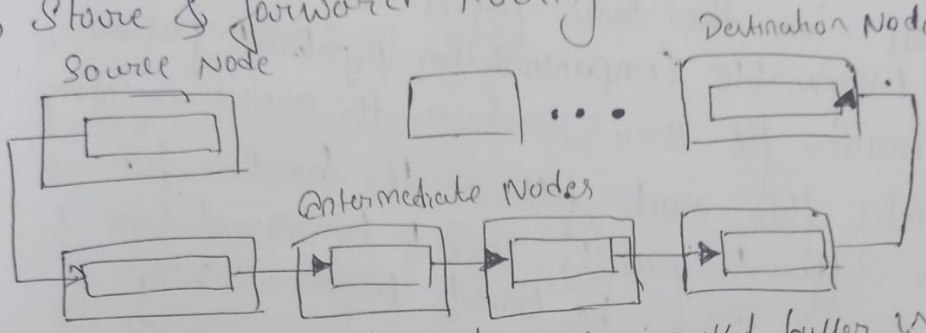


A two functions switches can assumes only two possible state namely state or exchanges However a four function switch box can be any of four possible states.

An multistage network is capable of connecting any input terminal to any output terminal.

4) Illustrate the following terms associated with multicore network.
i) store & forward routing at packet level
ii) Wormhole routing at flit level

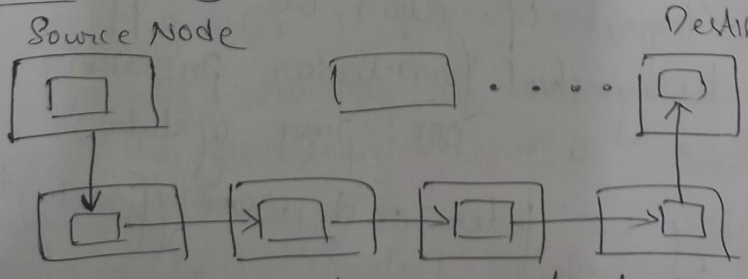
i) Store & forward Routing



a) stored-forward routing using packet buffer in successive nodes

* Packets are the basic unit of information flow in store & forward network
* Each node is required to use a packet buffer
* A packet is transmitted from a source node to a destination node through a sequence of intermediate nodes
* When a packet reaches an intermediate node, it is first stored in buffer
* Then it is forwarded to the next node if the desired output channel & a packet buffer in the receiving node are both available

2. Wormhole Routing



b) wormhole routing using flit buffer in successive routes

* Packets are subdivided into smaller flits. Flit buffer are used in the hardware router attached to nodes.
* The transmission form the source node to the destination node is done through a sequence of router.
* All the flits in the same packet are transmitted in order as inseparable companions in pipelined fashion
* Only the header flit knows where the packet is going
* All the data flits must follows the header flit
* Flits from different packets cannot be mixed up. Otherwise they may be towed to the wrong destination.

(5) Explain the vector processing principle with compound vector processing

Vector : A vector is a set of scalar data items, all of the same type, stored in memory

Vector processor : A vector processor is an ensemble of hardware resources, including vector register, functional pipeline processing elements & register counters for performing vector operations.

Vector processing : A vector processing occurs when arithmetic or logical operation are applied to vector. It is distinguished from scalar processing, which operates on one or one pair of data.

Vector processing is faster and more efficient than than scalar processing

Vectorization: The conversation from scalar code to vector code is called vectorization

Vectorizing Compiler: A Compiler capable of vectorizations is called a vectorizing Compiler (vectorizer)

Vector Instruction Types

1. vector - vector instructions
2. vector Scalar Instructions
3. vector - Memory instructions
4. vector - reduction instructions
5. Gather and Scatter instructions

Compound vector processing

A compound vector function (CVF) is defined as a composite function of vector operations converted from a looping structure of linked scalar operations

```
Do 10 I = 1, N
Load R1, X(1)
Load R2, Y(1)
Multiply R1, S
Add R2, R1
Store Y(1), R2
10 Continue
```

Where X(1) & Y(1) I = 1, 2 ... N are two source vector orginally residing in memory. After the Computation. the resulting vector is stored is stored back to the Memory. & an immediate Constant Supplied to the multiply instruction

After vectorization the above scalar SAXPY Code is Converted to a Sequences of five vector instructions

M(X:X+N-1) → V1 Vector Load
M(Y:Y+N-1) → V2 Vector Load
S x V1 → V1 vector Multiply
V2 x V1 → V2 vector Add
V2 → M(Y:Y+N-1) vector Store

X and Y are starting memory addresses of the X & Y vectors, respectively; V1 & V2 are two N-element vector registers in the vector processor

CVF : Y(1:N) = S x (1:N) + Y(1:N) or

Y(I) = S x (I) + Y(I)

where index I implies that all vector operations involve N elements.

* Typical CVF for one dimensional arrays are load, store multiply, divide logical & shifting operations
* The number of available vector register & functional Pipelines impose some restrictions on how many CVFs can be executed simultaneously.

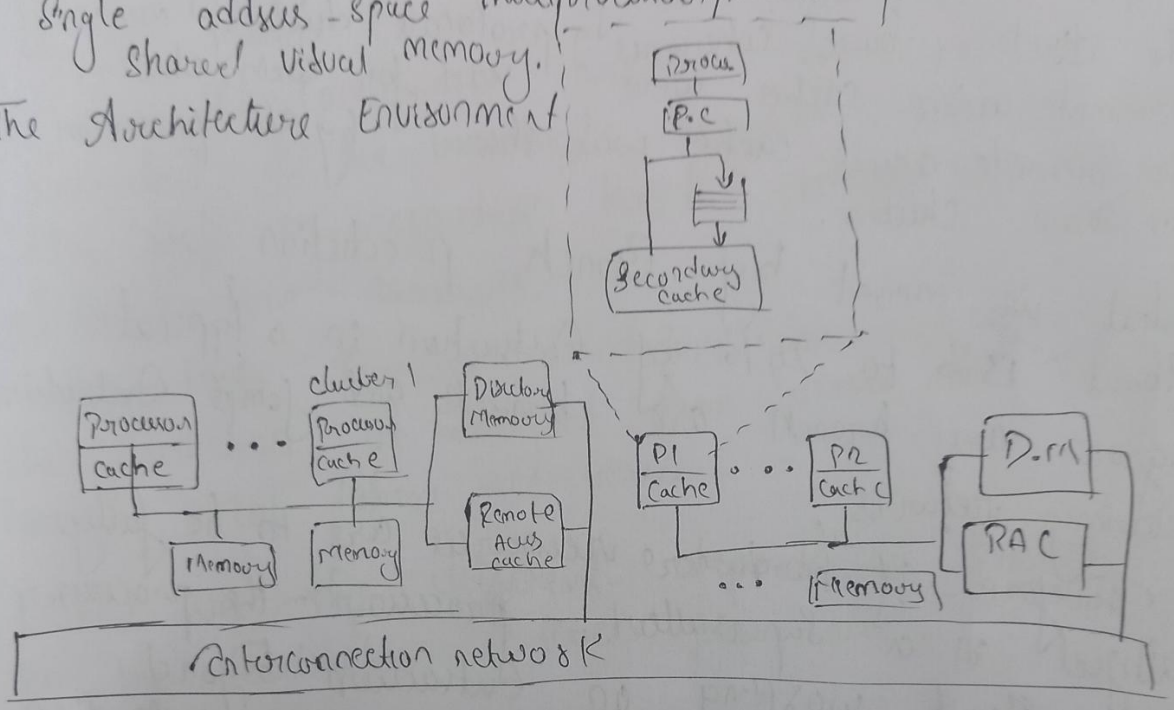
6) List & Explain the various approaches of Latency Hiding technique.

Latency Hiding Techniques

Shared Virtual Memory

Single address space multiprocessor/multicomputer must use shared virtual memory.

The Architecture Environment



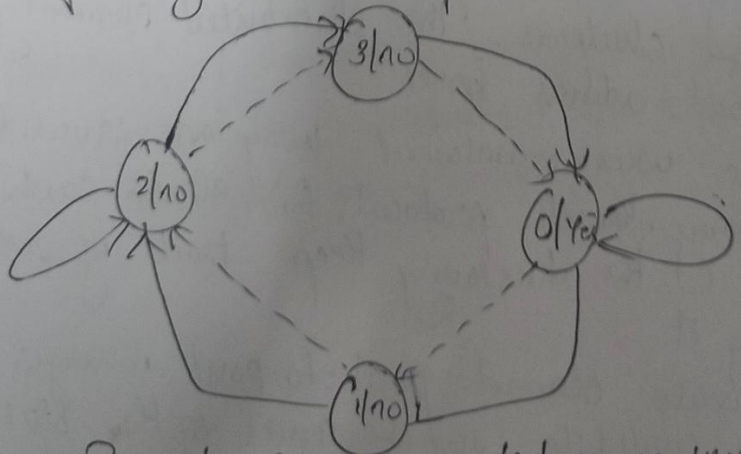
* It consist of multiple multiprocessor clusters connected through scalable low latency interconnection network
* Physical memory was distributed among the processing nodes in various clusters. The distributed memory formed a global address space
* Cache Coherence was maintained using an invalidating distributed directory-based protocol. For each memory block, the directory keep track of remote nodes caching it
* When we write occured, point-to-point messages were sent to invalidate remote copies of the block
* Acknowledgment messages were used inform for originating node when an invalidation was completed

* Two level of local cache were used per processing node Loads & writes were Seperated with the use of write buffers for implementing weather memory consistency models.
* The main memory was shared by all processing nodes in the same cluster. To facilitate prefetching & the directory-based coherence protocols directory memory & remote-access cache were used for each cluster.
* The remote-access cache was shared by all processor in same cluster.

(7) what is meant by Branch prediction

* About 15% to 20% of instruction in a typical program are branch and jump instruction including returns
* Therefore - if hardware resources are to be fully utilized in a superscalar - processor - the processor must start working on instruction beyond a branch, even before the branch instruction itself has completed. This is only possible through some form of branch prediction

*



* A basic Branch prediction techniques uses a so-called two bit predictor. A two bited counter is maintained for every conditional branch instruction in the program.

* The two bit counter has four possible states; These four states & possible transitions blw these states are shown Fig. 12.15
* When the counter state is 0 or 1, the respective branch is predicted is taken; when the counter state is 2 or 3, the branch is predicted is not taken.
* when the conditional branch instruction is executed & the actual branch outcome is known. the state of respective two-bit counters is changed & broken line arrowes.
* When two successive predictions come out wrong to the prediction is changed from branch taken to branch not taken, vice versa

(8) What are the different dependencies existing in the instruction level parallelism

1) Data Dependencies

A RAW dependence - ie. true data dependence - will hold up the execution of the dependent instruction if the result value required as its input operand is not available.

Operand forwarding can be added to this schema to speed up the supply of needed o/p operand as soon as its value have been completed

WAR and WAW dependencies

anti dependence & o/p dependences respectively also hold up the execution of dependent instruction & create a possible pipeline stall.

2. Control dependences

The instruction in reorder buffer belong to branch in the program which should not have been taken there has been a mis predicted branch. Clearly then the reordered buffer should be flushed along with other elements of pipeline

* Therefore the performance impact of control dependence in running program is determined by accuracy of branch prediction technique employed
* The reorder buffer plays no direct role in handling of control dependences

3. Resource Dependences

(3) on instruction needs a functional unit to execute, but the unit is not free, then instruction must wait for unit to become free - clearly no technique in the world can change that

* The processor designer can aim to achive at least this: if a subsequent instructions needs to use another functional unit which is free then the subsequent instruction can be executed out of order.

(9) Demonstrate How the Thread level parallelism is Implemented to reduce the limitation in the Exploiting Instruction level parallelism.

Sol:

* One way to reduce the burden of dependences is to Combine with hardware Support within the Processor. Instruction from multiple independent threads of Execution
* Such hardware Support for multi-threading would provide the processor with pool of instructions in various Stages of execution, which have a relatively smaller number of dependences amongst them, since the threads are independent of another
* Let us Consider once again the processor with Instruction pipeline of depth eight, & width targeted Superclas performance of four Instruction Completed in Every clock cycle
* Now, Suppose that these Instructions Come from four independent threads of Execution. Then on average the Number of Instruction in the processor at any one of time from one thread would be $4 \times 8/4 = 8$
* With the thread being independent of one another there is a Smaller total number of data dependences amongst the Instruction in processor
* Further, Multi Controll dependences also being Separated into four threads, less agressive branch prediction Need.

* Another major benifit of hardware support in multithreading is that pipeline stall are very effectively utilized.
* If one thread runs into pipeline stall for accessing main memory. say - then another thread makes use of corresponding processor clock cycles which would otherwise wasted
* Thus hardware support for multi threading the processor must be designed to switch bw threads either on the occurence of pipeline stall.

Dependency that are switching threads, hardware support for multi threading may be classified as

1. Coarse grain multi threading
2. Fine grain multi threading
3. Simultaneous multi threading

(10) Relate how the Operand fowarding & reorder buffering techniques is implementing in instruction level parallelism

Operand Foswarding

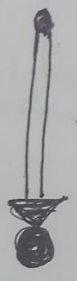
It helps in reducing the Impact of true data dependences in instruction stream. Consider the following sequence

ADD R1, R2, R3
SHIFTR #4, R3, R4

The result of ADD instruction is stored in destination register R3. & then shifted by right for four bits in the second instruction with shifted value being placed R4.

ADD R1, R2, R3

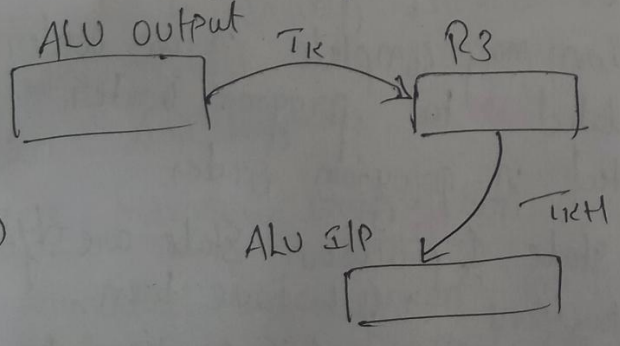
RAW (R3)

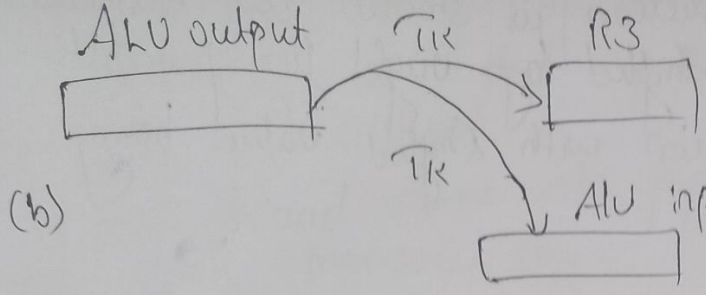


SHIFTR #4, R3, R4

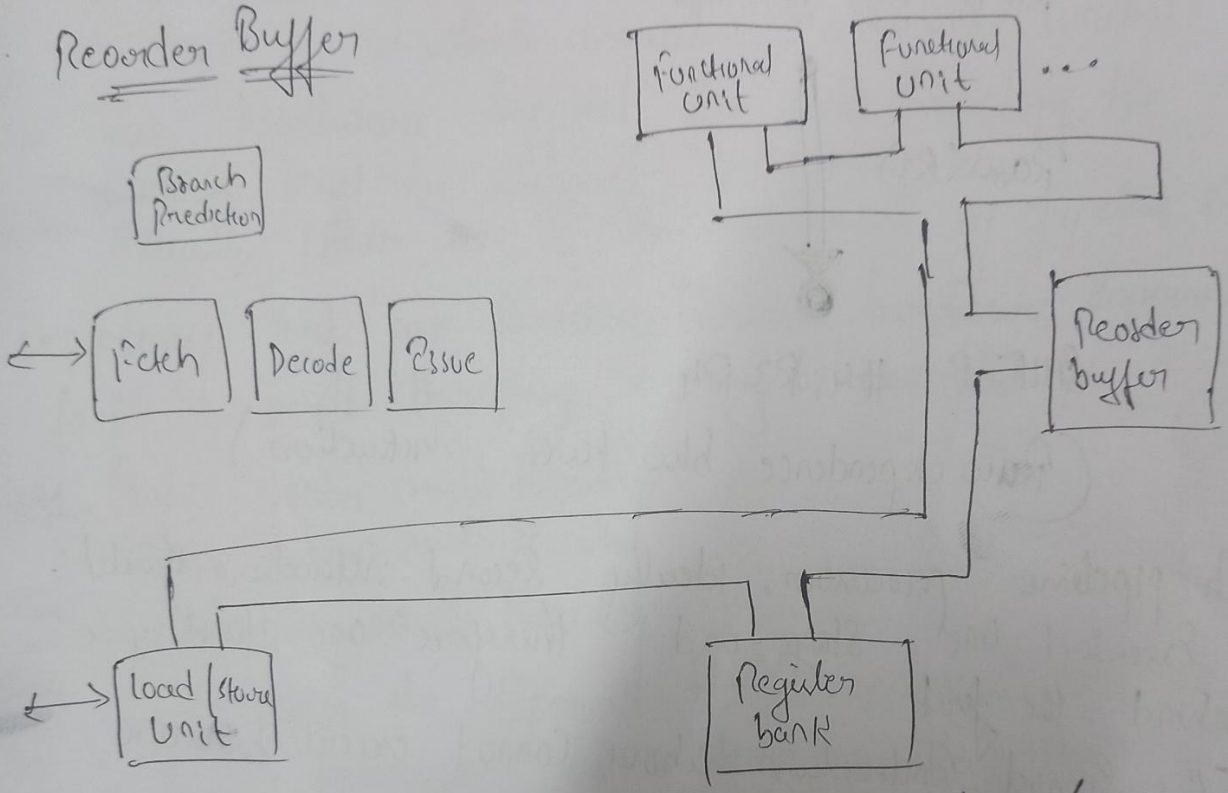
(Raw dependence b/w two instruction)

* In pipeline processor, ideally second instruction should be executed one stage - and therefore one clock cycle behind the first
* The second instruction above cannot executed in one cycle behind the first
* when carried out in this order, clearly two data transfer operations take two cycles





Reorder Buffer



* Since instruction execute in parallel on multiple functional units the reorder buffer serves the function of bringing completed instructions back into a order which is consistent with program order
* Note that instruction may complete in an order which is not related to program order, but must be commited in program order
* At any time, program state & processor state are defined in terms of instructions which have been committed their results are reflected in appropriate register and / or memory locations

* Entries in the reorder buffer are completed instruction which are re queed in program order
* However, since instruction do not necessarily complete in program order we also need a flag with each reorder buffer entry to indicate whether the instruction in the position has completed

(11) Discuss the different vector access memory schemas.

1) C-Access memory Organization

The M-way low-order interleaved memory structure allows m-memory words to be accessed together in overlapped structure. This concurrent access has been known as C-access

The access cycles in various memory modules are staggered. The lower-order bits choose the modules, & the high order bits selected the word within every module, where m = 2^n & n+b = n is the address length.

S-access memory Organization

The low-order interleaved memory can be rearranged to enable simultaneous access or S-access in the method all memory modules are created simultaneously in a synchronized method

C-S access memory organization

A memory organization in which C-access & S-access are combined is called C/S access. The scheme where n access buses are applied with m interleaved memory modules connected to every bus

The C/s - access memory is suitable for use in vector multiprocessor configurations

* It provides parallel pipelined access to a vector data set with high bandwidth
* A particular vector cache design is required with each processor to maintain smooth data movement blw the memory multiple vector processors

(12) Illustrate processor Consistency models.

Sln:-

* Every write operation can be divided into several sub-write to all memories
* A read from one such memory can happen before the write to this memory completes
* Therefore data can read be stable.
* A processor under PC can execute a younger load when an older store needs to be stalled
* Read before write, read after read & write before write ordering is still preserved in this model
* The processor Consistency model is similar to PRAM Consistency model with a stronger condition that defines all writes to same memory locations must be seen in the same sequential order by all processors.
* Processor Consistency is weaker than Sqential Consistency but stronger than PRAM Consistency model.

* The Stanford DASH multiprocessor System implements the variation of multiprocessor consistency which is incomparable to Goodman's definations.
* All processor needs to be Consistent in order in which they see writes by one processor & in the way they see writes by different processor to the same location.

(13) Define Parallel programming models. Discuss any two models.

Programming model →
Simplified and transparent view of Computer hardware/ Software System.
Parallel programming model is specifically designed for multiprocessors, multicomputer / vector / SIMD Computers
we have 5 programming models
1) Shared variable model
2) Message-Passing model
3) Data Parall model
4) Object Oriented model
5) Function & Logic model

Data Parallel model
* Used in SIMD Computers. Parallelism handled by Hardware Synchronization & flow Control.
* Fortran 90 → data parallel lang.

* Require predistributed data sets

Data parallelism

* This technique used in array processor (SIMD)
* Issue → Match problem size with machine size

Array language Extension

* various data parallel ~~used~~ language used
* Represented by high level data types
* CFD for illac. 4, DAP fortan for distributed array processor
* Target to make the number of PE's of problem size

Function & Logic Model

* Functional programming Language → Lisp, Sisal & Stand 88
Logic programming Languages → Concurrent Prolog & parlog

Functional programming model

- should not produce any side effects
- Non Concep of storage, assignment & branching
- Single assignment & data flow language, functional in nature

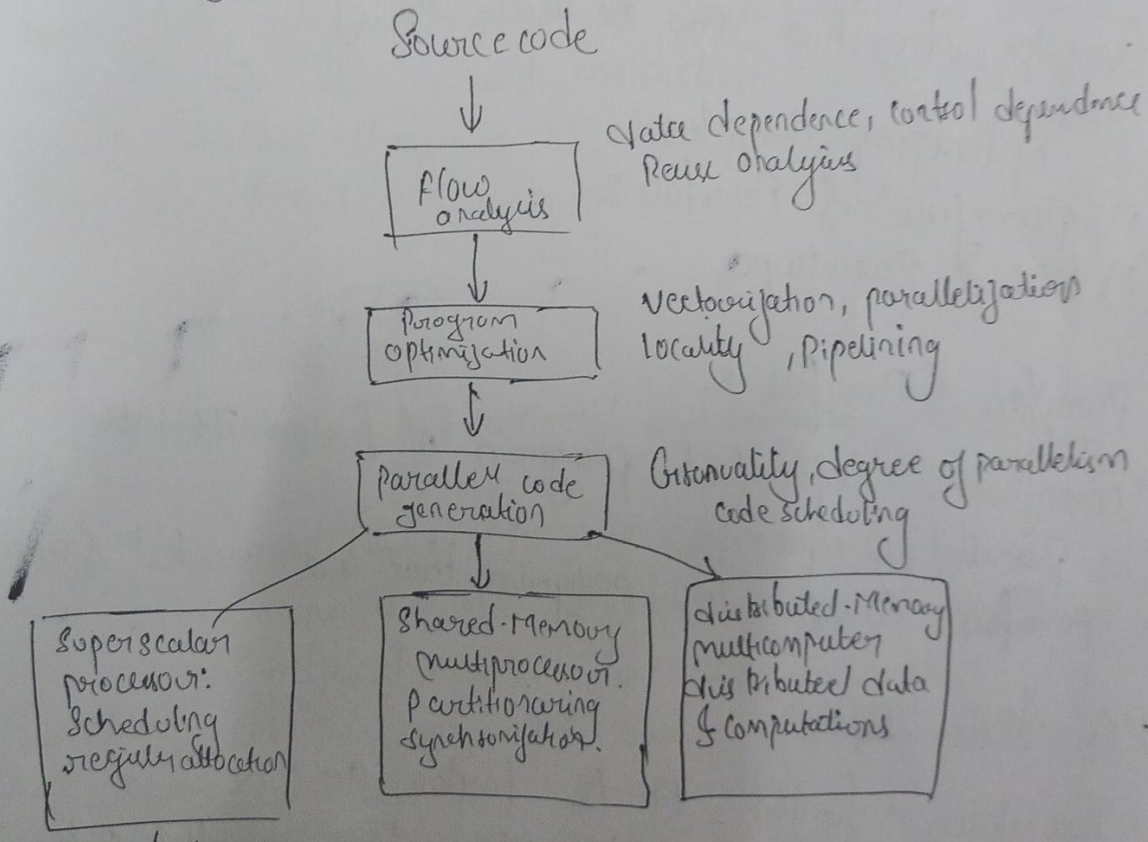
Logic Programming Model

- Used for Knowledge processing from large database
- Support implicity search Statergy
- And parallel execution and Or Parallel Reduction technique used.
- Used in artificial intelligence

(14) With neat Labelled diagram Illustrate different Paralellizing Compiler.

* Role of Compiler to remove burden of optimization & generation

3 phases:
1) Flow analysis
2) Optimization
3) Code generation



1) Flow analysis

- Reveals design Flow patters to determine data & Control dependences
- Flow analysis carried at various Execution levels.
  1) Instruction level -> VLSI or Superscalar processor.
  2) Loop Level -> Simd & Systolic Computer
  3) Task level -> Multiprocessor

Optimization

- Transformation of user programs to explore hardware capability
- Explore better performance
- Goal to maximize speed of code execution
- To minimize code length
- To local & global optimizations

Parallel Code generation

Compiler directive can be used to generate parallel code:

- 2 optimizing Compilers:
  1) Parafuse & Parafuse 2
  2) PFC & Parascope

Parafuse & Parafuse 2

- Transforms sequential program of fortan 77 into parallel programs
- parafuse consists two program that are encoded & passed.

PFC AND Parascope

- Translates fortan 77 to Fortan 90 code
- PFC package extended to PFC + For parallel code generation on shared memory multiprocessor.
- PFC performs analysis as following steps below
  1) Enter. procedure Flow analysis
  2) Transformation
  3) Dependence analysis
  4) Vector code Generation

15) Illustrate the Dynamic Scheduling of Pipeline using Tomasulo's Algorithm

-> Tomasulo's Algorithm is another method of implementing dynamic scheduling

-> This scheme is invented by Robert Tomasulo

Ex:-

```
MULTD   F4,F2,F2
ADDD    F2,F0,F6
```

Contains anti-dependence since the first instruction reads from F2 & the second instruction writes F2 (a WAR hazard). However, there is no data dependences as is shown by code below

```
MULTD   F4,F2,F2
ADDD    F8,F0,F6.
```

* The anti-dependency is removed without changing the semantics of code simply by changing the F2 to an F8. However we don't have to use F8.

* we can use any available register. In fact suppose there were some physical register in the chip that instruction set was not aware of ie there were some extra registers

* we can use those extra registers & leave the rest of the compiler to play with.

* This is what register renaming is: it allows the hardware to detect the name dependence and eliminate it by storing the result of an instruction somewhere else.

Thus, the register renaming

```
MULTD   F4,F2,F2
ADD     F2,F0,F6
```

The Add an execute & finish before the multiply starts even though looking at the code would result in erroneous
Suggests that answers.