UNIT- 2

Reduced Instruction Set computer (RISC) :- An important aspect of computer architecture is the design of the instruction set for the processor. Early computers had small and simple instruction sets, forced mainly by the need to minimize the hardware used to implement them.

A computer with a large number of instructions is classified as a complex instruction set computer (CISC)

A computer with fewer instructions with simple constructs so they can be executed much faster within the cpu without having to use memory as often called as Reduced Instruction Set computer (RISC)

CISC Characteristics :-

1) A large no. of instructions - typically from 100 to 250 instructions.
2) Some instructions that perform specialized tasks and are used infrequently.
3) A large variety of addressing modes - typically from 5 to 20 different modes.
4) Variable - length instruction formats
5) Instructions that manipulate operands in memory.

RISC characteristics :-

1) Relatively few Instructions.
2) Relatively few addressing Modes.
3) Memory access limited to load & Store Instructions.
4) All operations done within the registers of the cpu.
5) Fixed-length, easily decoded instruction format.
6) Single-cycle Instruction execution.
7) Hardwired rather than microprogrammed control.

Pipeline and vector processing :

Parallel Processing :- Parallel processing indicates that the system is able to perform several data-processing tasks at a time. It is able to perform concurrent data processing to achieve faster execution time. The system may have two / more ALU's and be able to execute two/more ~~processors~~ instructions at a time.

The purpose of parallel processing is to speed up the computer processing capability & increase its throughput, i.e the amount of processing that can be done during a given interval of time.

Parallel processing at a higher level of complexity can be achieved by having a multiplicity of functional Units that perform identical / different operations.

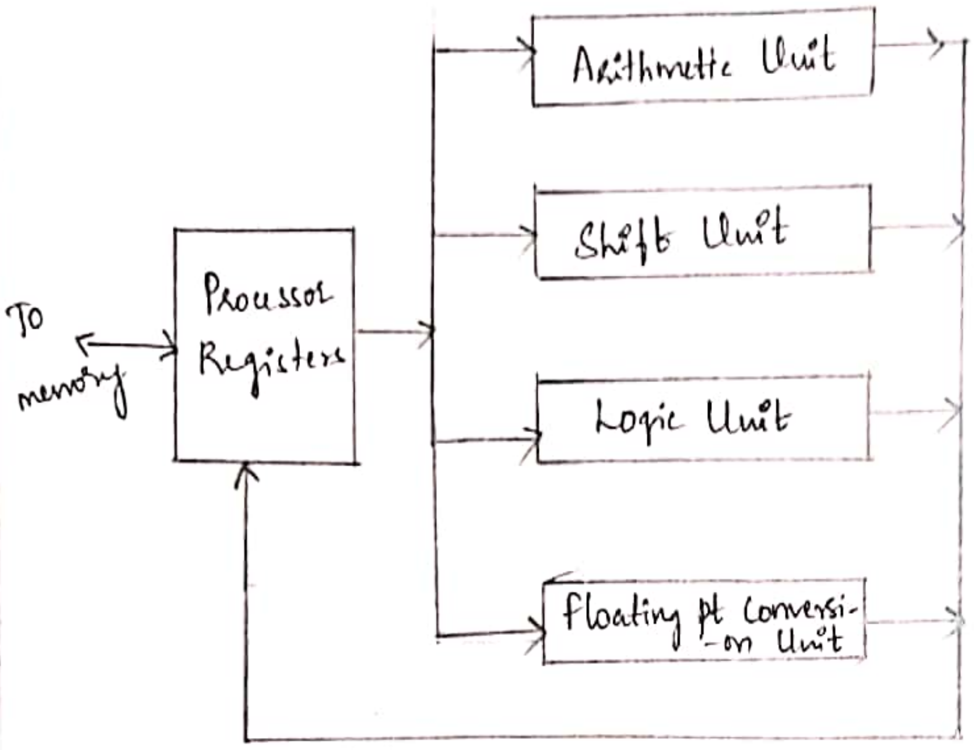


fig1: Processor with multiple functional Units.

Here we can see that, the data stored in the processor registers is being sent to separate devices based on the operation needed on the data. If the data is requesting an arithmetic operation, the data will be sent to arithmetic unit, similarly logic & shift Units, parallely executing arithmetic operations.

Instruction stream :- The sequence of instructions read from the memory is called as an Instruction stream.

Data Stream :- The operations performed on data in the processor is called a Data Stream.

The computers are classified into 4 types based on Instruction Stream & Data Stream. They are called as Flynn's classifi-cation of computers.

1) Single Instruction Stream & Single Data stream (SISD) :- Represents the organization of a single computer containing a control unit, a processor unit & a memory unit. Instru-ctions are executed sequentially & the system may/may not have internal parallel processing capabilities. Parallel processing may be achieved by means of multiple functional units (or) by pipeline processing.

2) Single Instruction Stream & Multiple Data stream: (SIMD): Represents an organization that includes many processing units under the supervision of a common control unit. All processors receive the same instruction from the control unit but operate on different items of data. The shared memory unit must contain multiple modules so that it can communicate with all the processors simultaneously.

3) Multiple Instruction streams & Single Data stream: (MISD): structure is only of theoretical interest since no practical system has be constructed using this organization because

multiple Instruction means more no. of Instructions means performing multiple instructions on same data at a time which is impossible.

4) Multiple Instruction stream and Multiple Data stream:- (MIMD): This refers to system capable of processing several programs at the same time.

Pipelining :- Pipelining is a technique of decomposing a sequential process into sub operations, with each sub process being executed in a special dedicated segment that operates concurrently with all other segments.

The pipe lining organization can be explained by the below example:

(A*B) + C

$R_1 \leftarrow A$, $R_2 \leftarrow B$

$R_3 \leftarrow R_1 * R_2$, $R_4 \leftarrow C$

$R_5 \leftarrow R_3 + R_4$

Input A and B

Multiply & input C

Add C to product.

In this process, 5 pipeline registers are used implemented in a segment. Each Segment has one/ two registers and a combinational circuit. R_1 through R_5 are Registers that recive new data with every clock pulse.

→ The contents of the Register in the below pipeline concept are shown as follows:

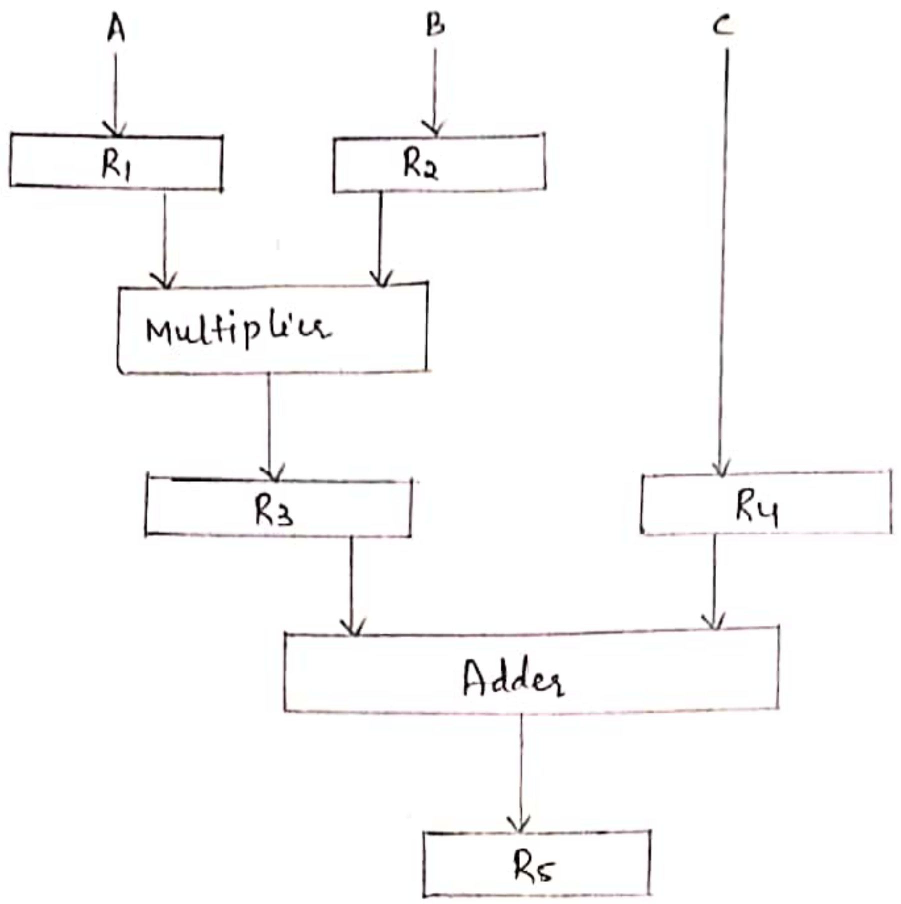


fig 2: Example of pipeline processing.

Clock pulse Number	Segment 1		Segment 2		Segment 3
	R_1	R_2	R_3	R_4	
1	A_1	B_1	-	-	-
2	A_2	B_2	$A_1 * B_1$	C_1	-
3	A_3	B_3	$A_2 * B_2$	C_2	$A_1 * B_1 + C_1$
4	A_4	B_4	$A_3 * B_3$	C_3	$A_2 * B_2 + C_2$
5	A_5	B_5	$A_4 * B_4$	C_4	$A_3 * B_3 + C_3$
6	A_6	B_6	$A_5 * B_5$	C_5	$A_4 * B_4 + C_4$
7	A_7	B_7	$A_6 * B_6$	C_6	$A_5 * B_5 + C_5$
8	—	—	$A_7 * B_7$	C_7	$A_6 * B_6 + C_6$
9					$A_7 * B_7 + C_7$

Contents of Registers in pipeline

Segment Representation :-

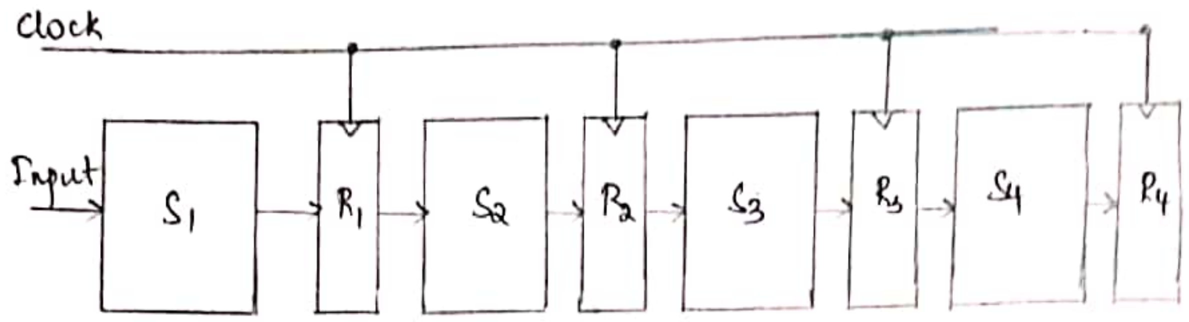


fig 3: Four Segment pipeline

The below table is the space diagram for the execution of 6 tasks in the 4 segment pipeline. The speedup of a pipeline processing over an equivalent non-pipeline processing is defined by the ratio

	1	2	3	4	5	6	7	8	9
Segment: 1	T_1	T_2	T_3	T_4	T_5	T_6			
2		T_1	T_2	T_3	T_4	T_5	T_6		
3			T_1	T_2	T_3	T_4	T_5	T_6	
4				T_1	T_2	T_3	T_4	T_5	T_6

→ clock cycles

$$\boxed{S = \frac{n t_n}{(k+n-1)t_p}}$$

fig 4: Space-time diagram for pipeline

Arithmetic pipeline :- The i/p's to the floating point adder pipeline are two normalized floating point binary numbers.

$$x = A \times 2^a$$

$$y = B \times 2^b$$

A & B are two fractions that represent the mantissas & a & b are the exponents. The floating point addition & substraction is performed in 4 segments and the sub-operations that are performed in 4 segments are:

1) Compare the exponents.
2) Align the mantissa.
3) Add/ subtract the mantissa's
4) Normalize the result.

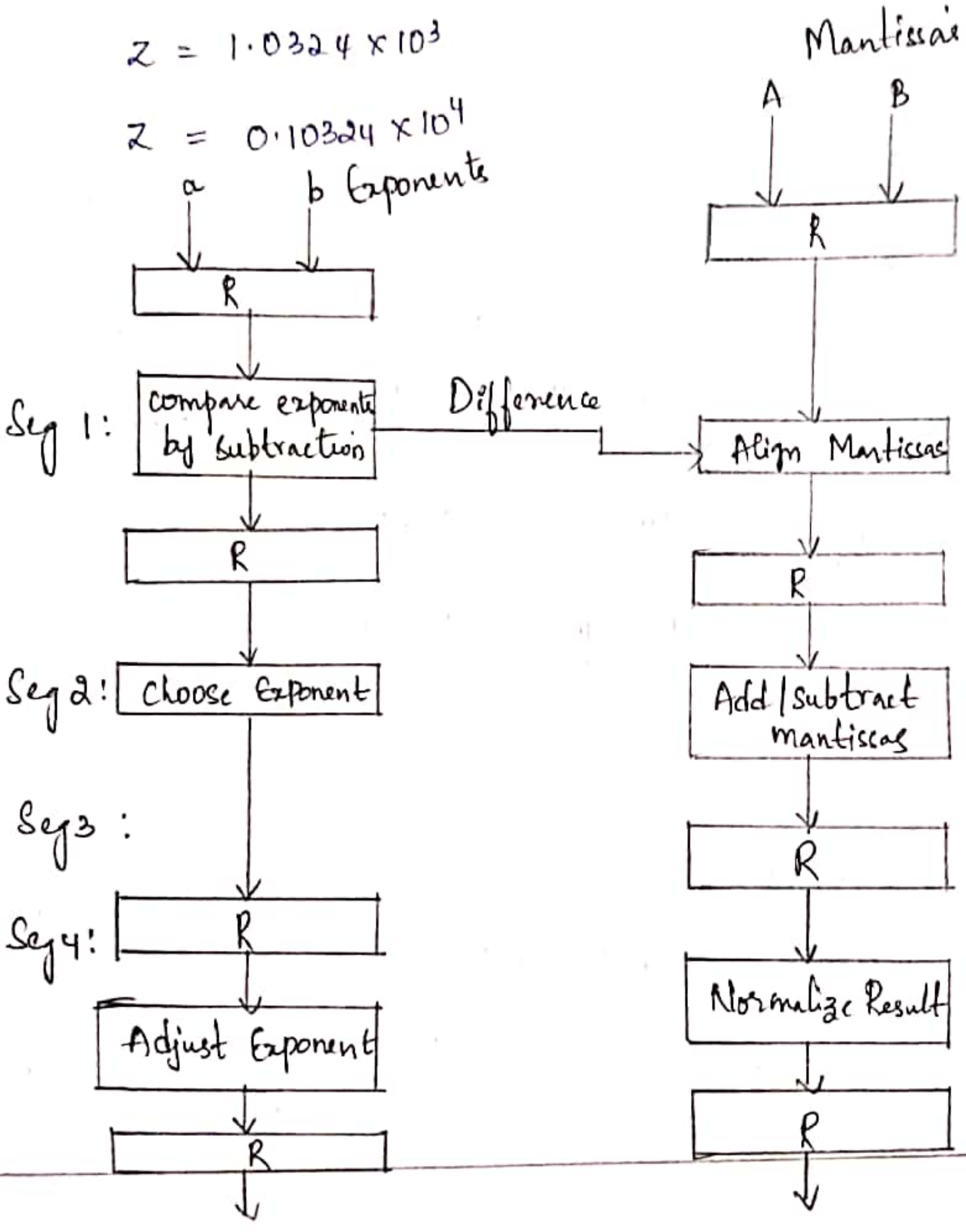
Example :- $x = 0.9504 \times 10^3$ $y = 0.8200 \times 10^2$

$\Rightarrow$ $x = 0.9504 \times 10^3$

$y = 0.0820 \times 10^3$

$z = 1.0324 \times 10^3$

$z = 0.10324 \times 10^4$



Instruction pipeline :- pipelining concept is not only limited to the data stream, but can also be applied on the instruction stream. The instruction pipeline execution will be the queue execution. In the queue the data that is entered first, will be the data retrieved first. Therefore when an instruction is first placed, the instruction will be placed in the queue & will be executed in the system. Finally, the results will be passing on to the next instruction in the queue. The Instruction cycle is given below.

1) Fetch the instruction from the memory.
2) Decode the instruction.
3) Calculate the effective address
4) Fetch the operands from memory
5) Execute the Instruction.
6) store the result in the proper place.

Four Segment Instruction pipeline :-
While an Instruction is being executed in segment 4, the next instruction in sequence is busy fetching an operand from memory in segment 3. The effective address is calculated separately in ~~seg~~ a separate arithmetic circuit. The fourth & all subsequent instructions can be fetched & placed in an instruction FIFO. The four segments are represented in the below figure.

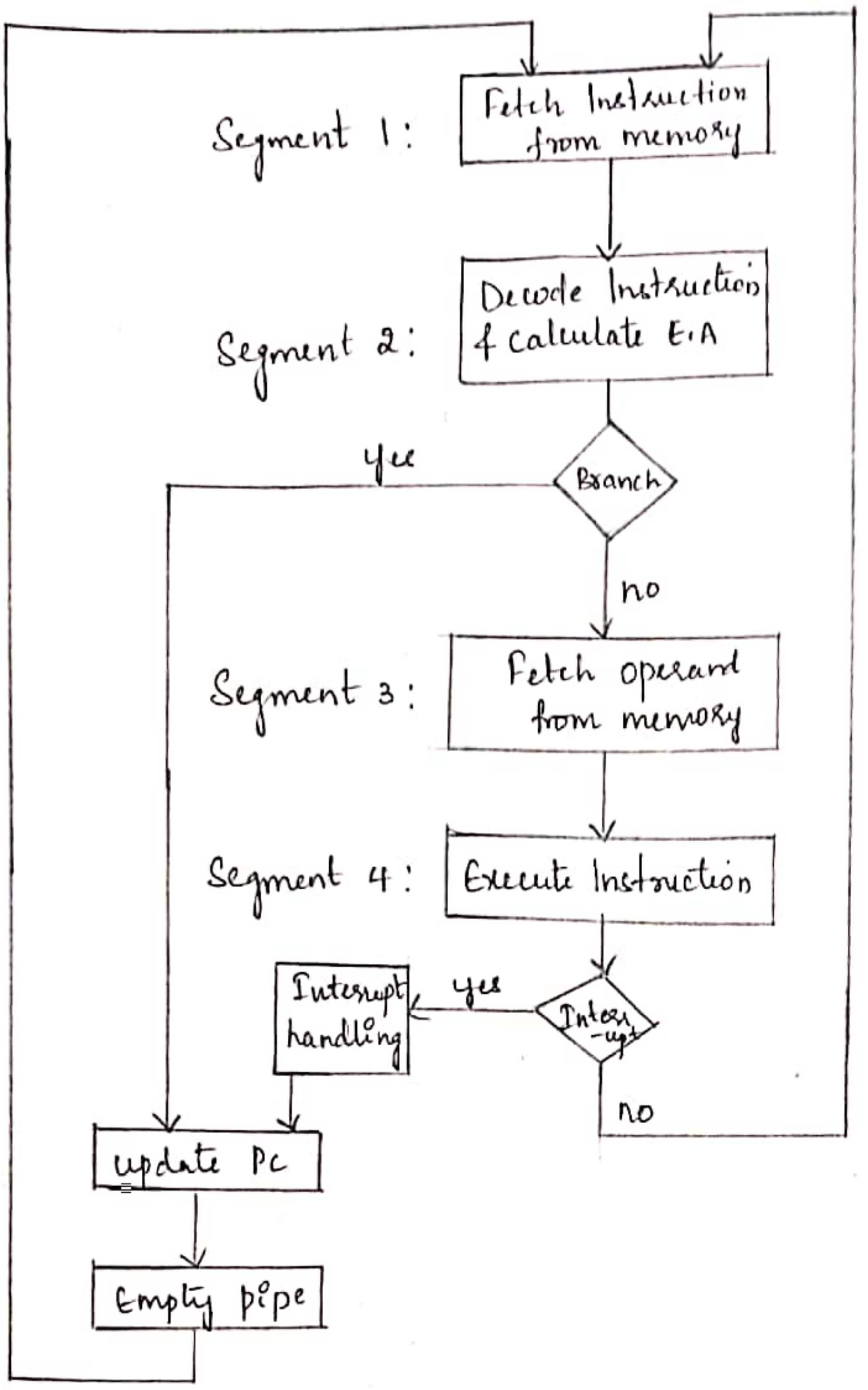


fig 5 : Four-segment cpu pipeline.

1) FI is the segment that fetches an Instruction.
2) DA is the segment that decodes the Instruction & calculates E.A
3) FO is the Segment that ~~decodes~~ fetches the operand.
4) Ex is the Segment that executes the instruction.

Timing of Instruction pipeline :-

Step:	1	2	3	4	5	6	7	8	9	10	11	12	13
Instruction: 1	FI	DA	FO	EX									
2		FI	DA	FO	EX								
(Branch) 3			FI	DA	FO	EX							
4				FI	—	-	FI	DA	FO	EX			
5					-	-	-	FI	DA	FO	EX		
6									FI	DA	FO	EX	
7										FI	DA	FO	EX

In general, there are three majos difficulties that cause the instruction pipeline to deviate from its normal operation.

1) Resource conflicts :- These are caused by access to memory by two segments at the same time. Most of these conflicts can be resolved by using separate instruction & data memories.

2) Data Dependency : Arise when an instruction depends on the result of a previous instruction, but this result is not yet available.

3) Branch difficulties :- Arise ~~when an instruction depends on the result of a previous instruction,~~ from branch and other Instructions that change the value of PC.

Data Dependency conflict can be solved by the following methods :

Hardware Interlocks : The most straight forward method is to insert hardware interlocks. An interlock is a circuit that detects instructions whose source operands are destination of instructions up in the pipeline. Detection of this situation causes the instruction whose source is not available to be delayed by enough clock cycles to resolve the conflicts.

Operand forwarding :- This technique uses special hdw to detect a conflict & avoid the conflict path by using a special path to forward the values b/t the pipeline segments.

Delayed Load :- When executing an instruction in the pipeline, simply delay the execution starting of the instruction such that all the data that is needed for the instruction can be successfully updated before execution.

Branch conflicts are solved by the following concepts :

Pre-fetch Target Instruction :- Branch Instructions which are to be executed are prefetched to detect if any errors are present in the branch before Execution.

Branch Target Buffer :- BTB is the associative memory implementation of the branch conditions

Loop Buffer :- It is very high speed memory buffer device. Whenever a loop is to be executed in the computer, the complete loop will be transferred into the loop buffer memory & will be executed in the cache memory.

Branch Prediction :- Before a branch is to be executed, the instructions along with the error checking conditions are checked to avoid unnecessary branch loops.

Delayed Branch :- In this, Execution of a branch process is delayed, before all the data is fetched by the system from the beginning of the cpu.

Risc Pipeline :- The simplicity of the instruction set can be utilized to implement an instruction pipeline using a small no. of sub operations, with each being executed in one clock cycle.

→ Due to fixed length instruction format, the decoding of the operation can occur at the same time as the register selection.

→ Since the arithmetic, logic & shift operations are done on register basis, there is no need of extra fetching or E.A decoding.

→ Therefore, the total operations can be categorized as one segment will be fetching the instruction from pgm memory, the other segment executie the instruction in the ALU & the third may be used to store the result of the ALU operation in a destination register.

→ The data transfer instructions in Risc are limited to only Load & Store instructions. To prevent conflicts in data transfer, two separate buses one for storing instruction & other for storing the data.

Example of three segment Instruction pipeline :- An operation with arithmetic, logic & shift operations are performed the Instruction cycle has the following steps:

I - Instruction Fetch

A - ALU operation

E - Execute Instruction

→ The I segment will be fetching the Instruction from pgm memory. The Instruction is decoded as an ALU operation is performed in the A segment.

→ In the A segment the ALU operation, instruction will be fetched & the E.A will be retrieved.

→ Finally in the E segment, the instruction will be Execu-ted.

Delayed Load :-
1. LOAD : $R_1 \leftarrow M[address\ 1]$
2. LOAD : $R_2 \leftarrow M[address\ 2]$
3. ADD : $R_3 \leftarrow R_1 + R_2$
4. STORE : $M[address\ 3] \leftarrow R_3$

(a) Pipeline timing with data conflict :

Clock Cycle	1	2	3	4	5	6
1. Load R1	I	A	E			
2. Load R_2		I	A	E		
3. Add R_1+R_2			I	A	E	
4. Store R_3				I	A	E

(b) pipeline timing with delayed load :

Clock cycles:	1	2	3	4	5	6	7
1. Load R1	I	A	E				
2. Load R2		I	A	E			
3. No operation			I	A	E		
4. Add R1+R2				I	A	E	
5. Store R3					I	A	E

The concept of delaying the use of the data loaded from the memory is referred to as delayed load.

Vector Processing :- Special processing systems like artificial Intelligence systems & some weather forecasting systems, terrain analysis, the normal systems are not sufficient. In such systems, the data processing involves on very high amount of data usually classified as big arrays. To process these data the vectors are considered as the large one-dimensional array of data.

~~Mult~~ Machine Level program :-

```
    Initialize I = 0
20  Read A(I)
    Read B(I)
    Store C(I) = A(I) + B(I)
    Increment I = I+1
    If I ≤ 100 go to 20
      continue
```

The same pgm written in the vector processing statement as:

$C(1:100) = A(1:100) + B(1:100)$

The vector Instruction includes the initial address of the operands, the length of the vectors, & the operation to be performed, all in one composite Instruction.

Operation code	Base Address Source 1	Base Address Source 2	Base Address destination	Vector length

fig 6: Instruction Format for Vector processor.

Matrix Multiplication :- In this, Row of Matrix 'A' is multiplied with Column of the Matrix 'B' elements, individually but adding the result finally.

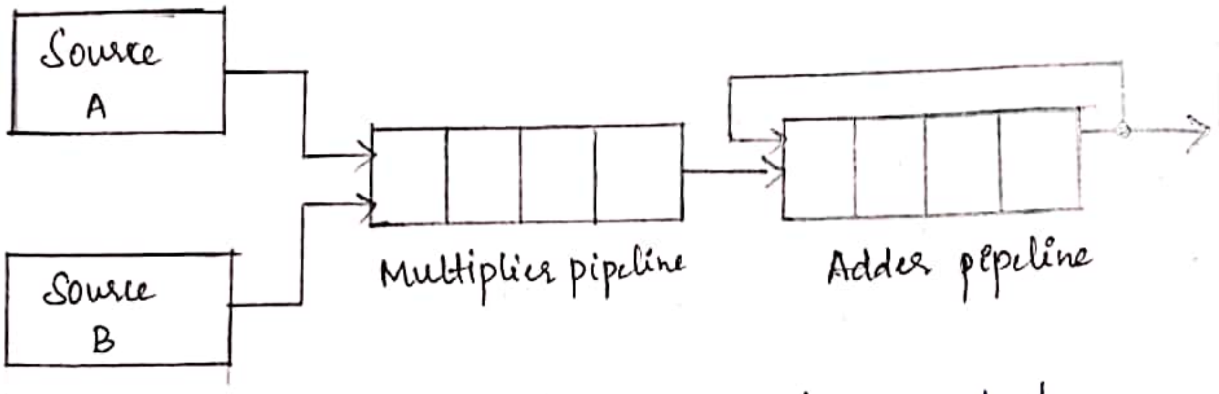


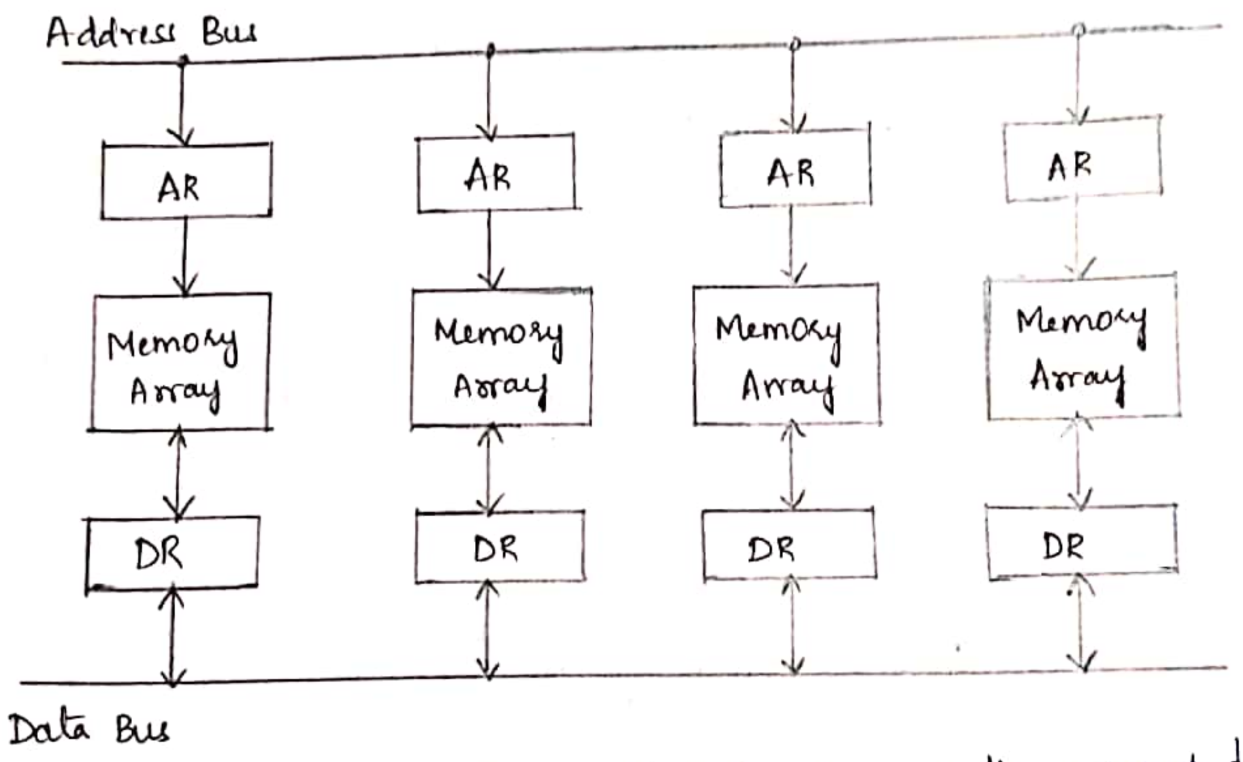
fig 7: Pipeline for calculating an Inner product.

→ A 4x4 matrix A and B are considered. From the Source A vector first 4 values are taken & will be sent to multiplier pipeline along with the 4 values from the vector B. The resultant '7' value is stored in the adder pipeline.

Likewise remaining values from a row & column multiplication will be brought into the adder pipeline, which will be perform -ing the addition of all things finally we will have the result of one row to column multiplication.

→ when addition operation is taking place in the adder pipeline the next set of values will be brought into the multiplier pipeline, so that all the operations can be performed simultaneously using the parallel processing concepts by the implementation of pipeline.

Memory Interleaving :- Multiple Memory Module Organization



Pipelining & vector processing naturally requires the several data elements for processing. The modular system permits one module to initiate a memory access while other modules are in the process of reading or writing a word & each module can honors a memory request independent of the state of the other modules. The advantage of a modular memory is that it allows the use of a technique called interleaving.

→ In an interleaved memory, different sets of addresses are assigned to different memory modules.

Array Processors :- It is a processor that performs computations on large arrays of data. An attached array processor is an auxiliary processor attached to a general-purpose computer. An SIMD array processor is a processor that has a single-instruction multiple-data organizations. It manipulates vector instructions by means of multiple functional units responding to a common instruction.

Attached Array Processor :-

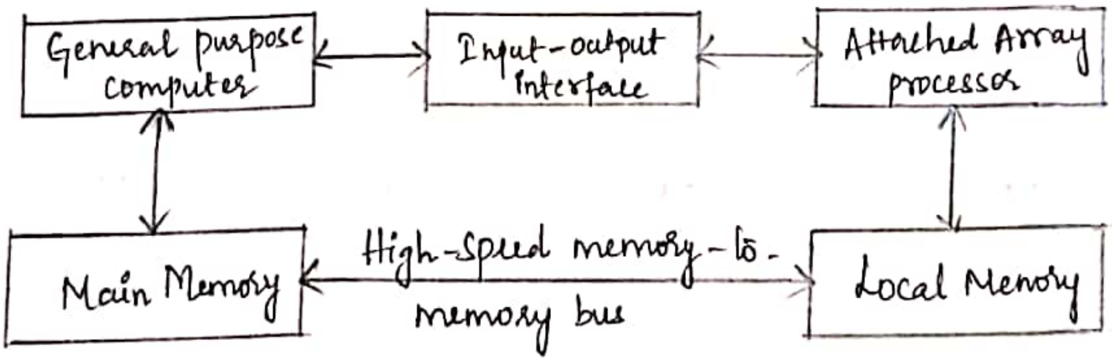


fig 8 : Attached Array Processor with host computer

Scanned by CamScanner

The above fig shows the interconnection of an attached array processor to a host computer. The host computer is a general-purpose commercial computer & the attached processor is a backend machine driven by the host computer. The array processor is connected to an I/p-o/p controller which computer treats like an interface. The data for the attached processor are transferred from main memory to a local memory, through a high speed bus.

SIMD Array Processor :-

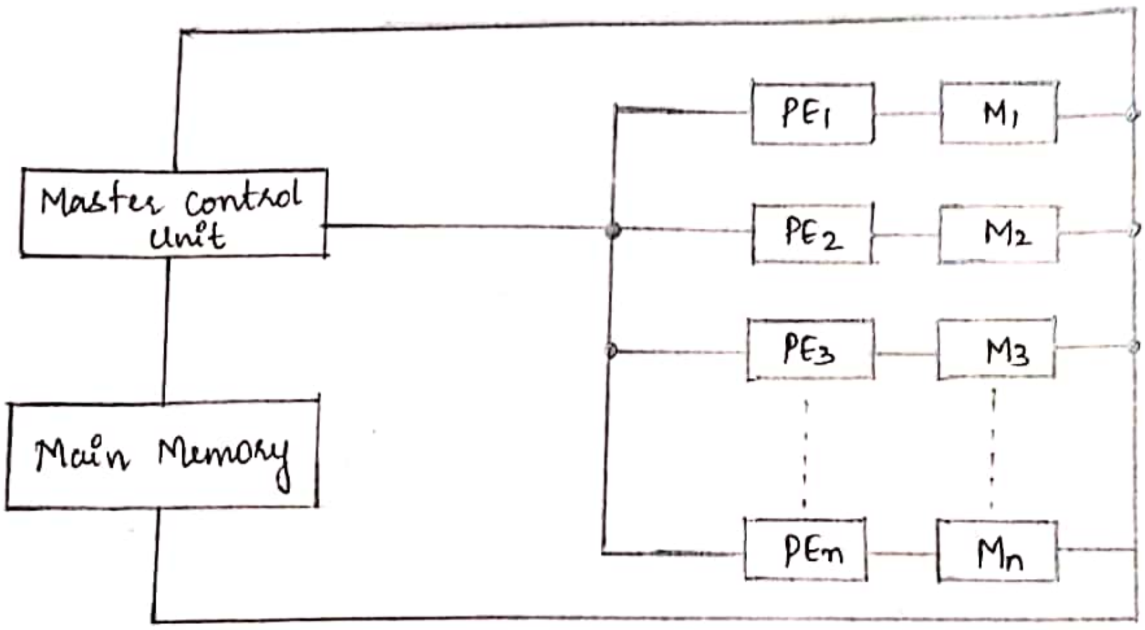


fig 9: The working of SIMD array processor.

→ In this, a master control Unit will be coordinating all the processes in the array processor. Each processing (PE) unit in the array processor is having a local memory [M] unit as in the memory interleaving concept on which

It performs the operations. Finally a main memory in which the original source data & the results ~~a~~ that are obtained from the array processor will be stored.

Multiprocessors :- Characteristics of Multiprocessors :-

A multiprocessor system is an interconnection of two/more cpu's with memory & Input-output equipment. A multiprocessor system implies the existence of multiple cpu's although usually there will be one/more Iop's as well. A multiprocessor system is controlled by one operating system that provides interaction b/t processors & all the components of the system.

1) occupies very less space and these are of low cost :
Though a single system supports the existence of more than one processor, it's widely used ~~so~~ due to its low cost & small size of the processor. This is possible due to VLSI chip which facilitate the integration of several millions of transistors into a single minute chip at a very low cost.

2) Enhances the Reliability of the Existing system to a Great Extent :
Multiprocessors improves the reliability of the system so that a failure/error in one part has a limited effect on the rest of the computer. If a fault causes one processor to fail, a second processor can be assigned

to perform the functions of the disabled processor. The system as a whole can continue to function correctly with perhaps some loss in efficiency.

3) Enhance the overall performance of the system & hence Increasing the throughput:
The given multiprocessor system can be effectively utilized in two ways: either by dividing a single large task into multiple task blocks & assigning each of them to a separate processor (or) Assigning these processors with multiple independent tasks.

(i) By analyzing first consideration, following can be done.
→ When a single large task block is divided into multiple task blocks & are assigned to individual processor in an independent manner, systems performance can be enhanced to a large extent. This is because the time required to solve this task using a single processor is extremely high when compared to multiprocessor system.

(ii) By analyzing second consideration, "multiple, independent tasks can be assigned to each of these processors", following can be done.
→ Processor can guard different functionalities of a single organization.

→ The processor maintains a given set of ALU's that can govern the operations of the processors which frequently ~~co~~ interacte with the interface.

4) Multiprocessors are classified by the way. their memory is organized :

A multiprocessor system with common shared memory is classified as a shared memory (or) tightly coupled multi-processor. This does not prelude each processor from having its own local memory.

An Alternative model of microprocessor is the distributed memory (or) loosely coupled system. Each processor element in a loosely coupled system has its own private local memory. The processors are tied together by a switching scheme designed to route infromation from one processor to another through a message-passing scheme. The processors relay pgm & data to other processors in packets. A packet consists of an address, the data content & some error detection code. The packets are addressed to a specified processor (or) taken by the first available processor, depending on the communication channel used.

Loosely coupled systems are most efficient when the interaction b/t tasks is less, where as tightly coupled systems can tolerate a higher degree of interaction b/t the tasks.

Ins

Interconnection Structures :- The interconnection between the components can have different physical configurations, depen-ding on the no. of transfer paths that are available b/t the processors and memory. These are several physical forms available for establishing an interconnection network. Some of the th schemes are presented as follows :

1) Time-shared Common bus
2) Multiport memory
3) Crossbar Switch
4) Multistage Switching network
5) Hypercube System

1) Time-shared Common bus :- A common bus multiprocessor system consists of a no. of processors connected through a common path to a memory unit. A time-shared common bus for five processors is shown below. only one processor can communicate with the memory / another processor at any given time. There is a single memory unit which is also Connected to the common bus. whenever any of these processors intends to communicate either with the memory or with other processors, it has to initially check the availability of the given bus. If the bus is busy, the processor either waits till the bus is available. If it is available, the processor issues a command & release it on the bus. The command is checked by all the

attached devices & response is made only by the correspond-
ing device whose address matches with the cmd address.
while one device is using the bus, all the other devices
should either switch on to other process / remain idle.
Also only one processor, is authorized to use the bus at
any given instant of time. If conflict arises, separate
bus controller is introduced.

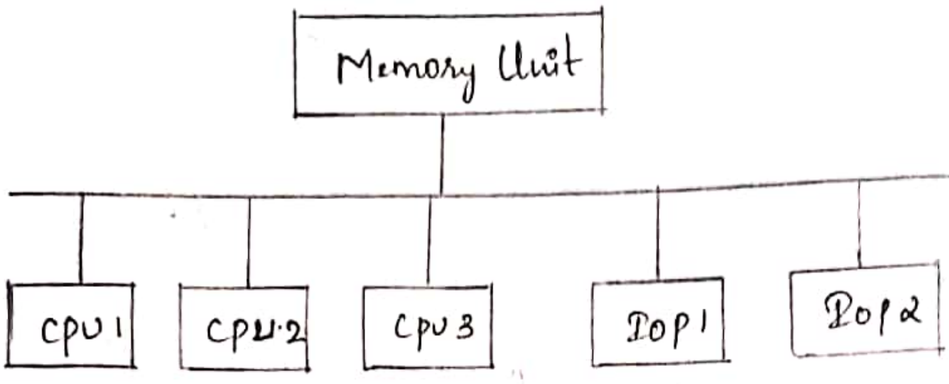


fig 10 : Time-shared common bus organization

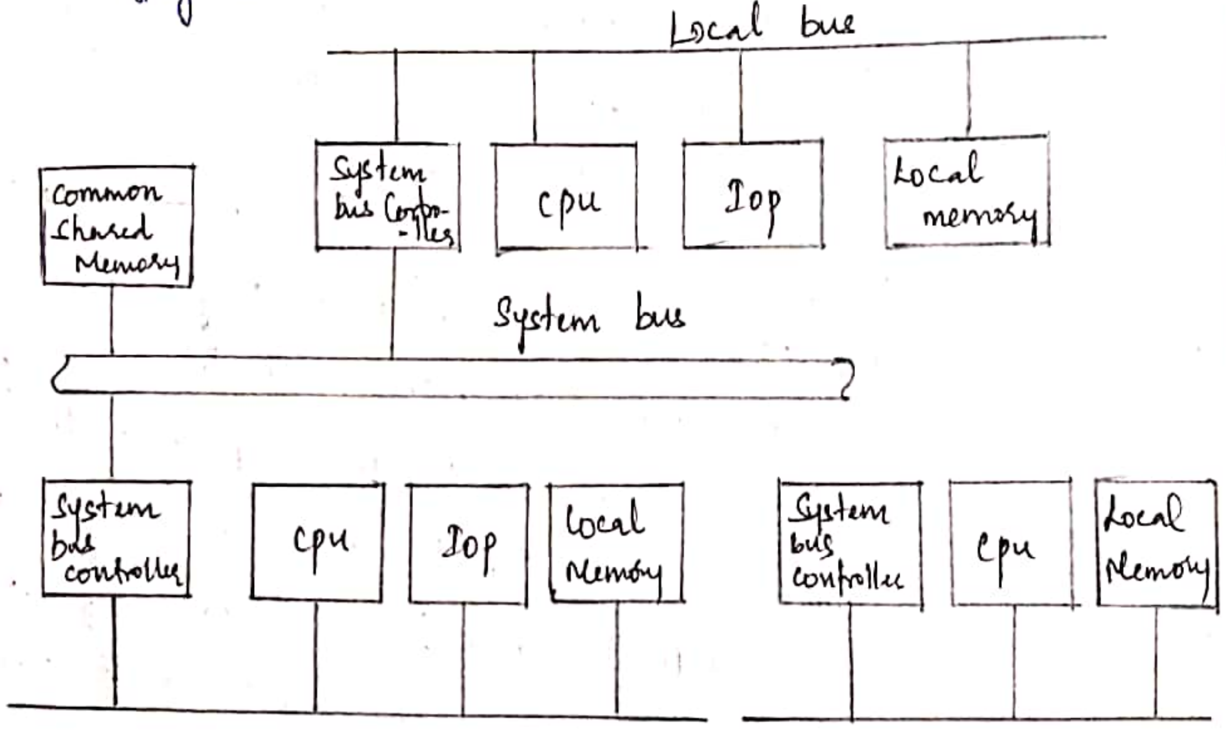


fig 11 : System bus structure for multiprocessor

2) Multiport Memory :- A multiport memory system employs separate buses b/t each memory module & each cpu. The bus which originates from cpu is a combination of address, data & control lines resp, which connects to the respective memory units through their ports. Each memory unit maintains three ports, with each port taking single processor bus. The memory unit must have an internal control logic so as to determine which processor can access the memory. Since, all the cpu's are connected to all the memory modules, there exist conflicts, to avoid conflicts, each cpu is assigned with some priority based on its port positions.

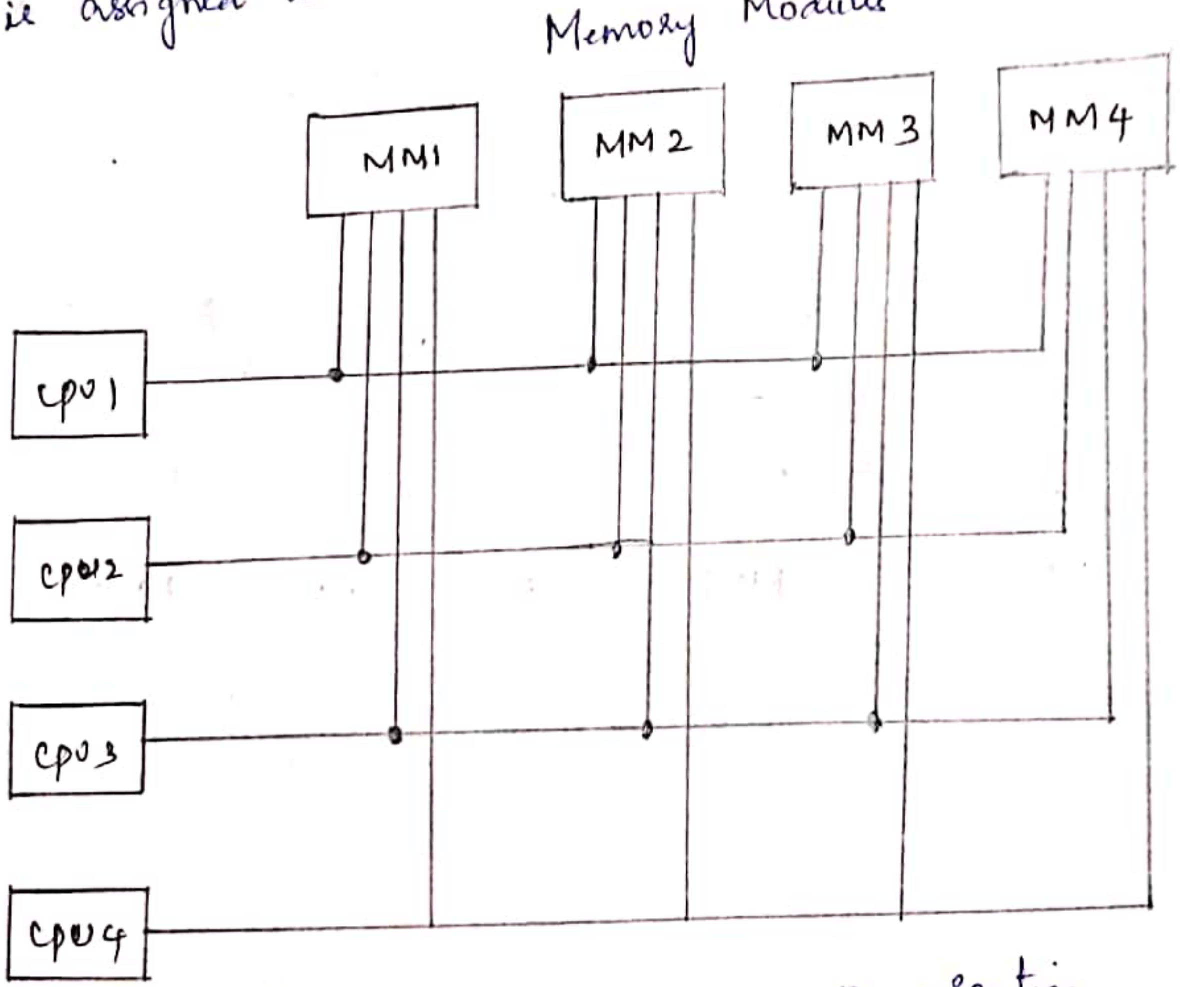


fig 12: Multiport Memory Organization

3) Cross bar Switch :- The cross bar switch organization consists of a no. of crosspoints that are placed at intersections b/t processor buses & memory module paths. The small square in each crosspoint is a switch that determines the path from a processor to a memory module. Each switch point has control logic to set up the transfer path b/t a processor & memory. It examines the address that is placed in the bus to determine whether its particular module is being addressed.

The functional design of cross bar switch connected to one memory module consists of multiplexers that select the data, address & control lines from one cpu for communication with the memory module. Priority levels are established by the arbitration logic to select one cpu when two / more cpu's attempt to access the same memory. A cross bar switch organization supports simultaneous transfers from all memory modules because there is a separate path associated with each module.

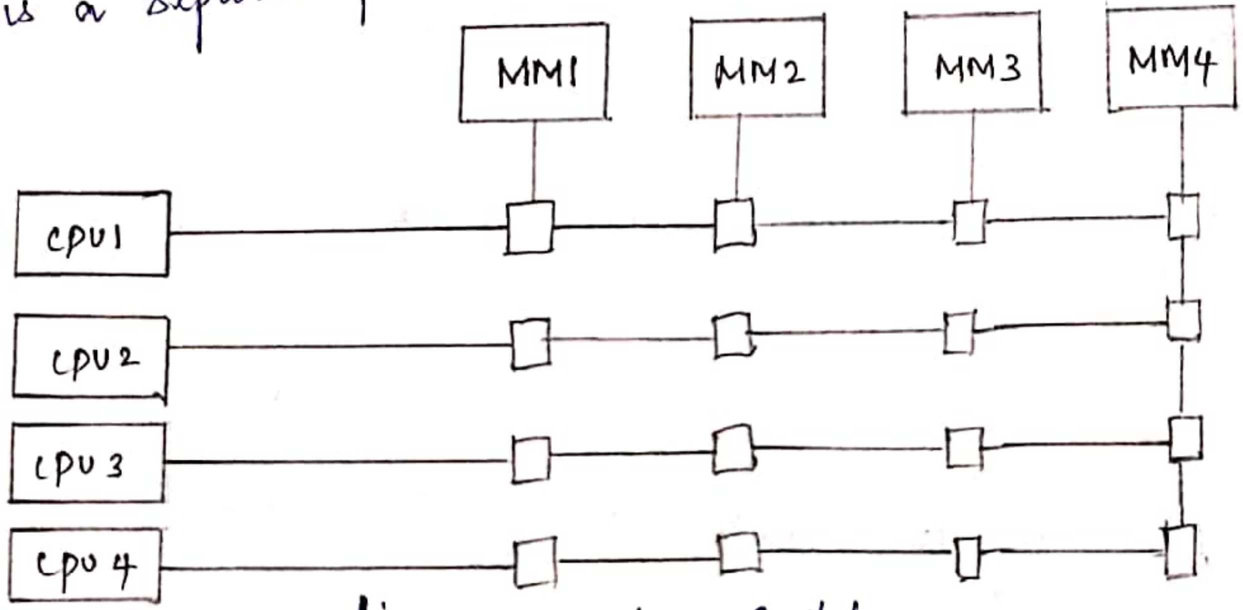


fig 15: Cross bar Switch

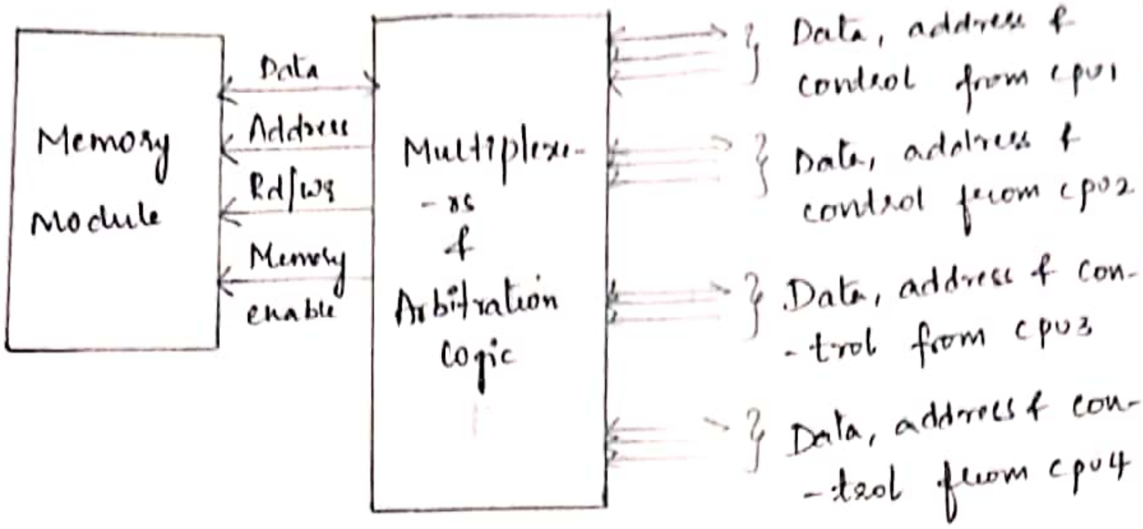

Memory Module
Data
Address
Rd/wr
Memory enable
Multiplexers & Arbitration logic
Data, address & control from cpu1
Data, address & control from cpu2
Data, address & control from cpu3
Data, address & control from cpu4

fig 14: Block diagram of crossbar Switch.

4) Multistage Switching Network :- The basic component of a multistage n/w is a 2-i/p, 2-o/p interchange switch. The 2x2 switch has 2 i/p's, labeled A & B, & i/p 2 o/p's labeled '0' & '1'. There are control signals associated with the switch that establish the ~~con~~ interconnection b/t the i/p & o/p terminals. The Switch also has the capability to arbitrate b/t conflicting requests.

using the 2x2 switch as a building block, it is possible to build a multistage network to control the communication b/t a no. of sources & destinations. The two processors P1 & P2 are connected through Switches to eight memory modules marked in binary from 000 through 111.

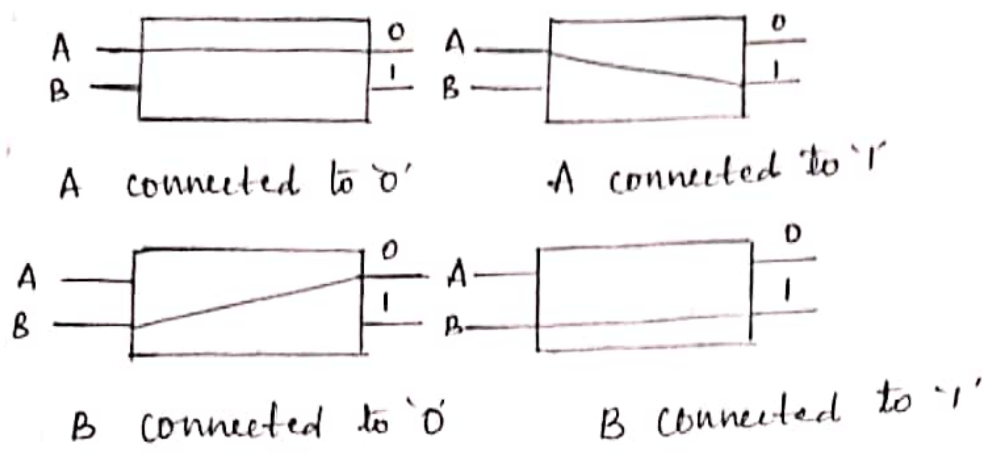


fig 15: Operation of a 2x2 interchange Switch

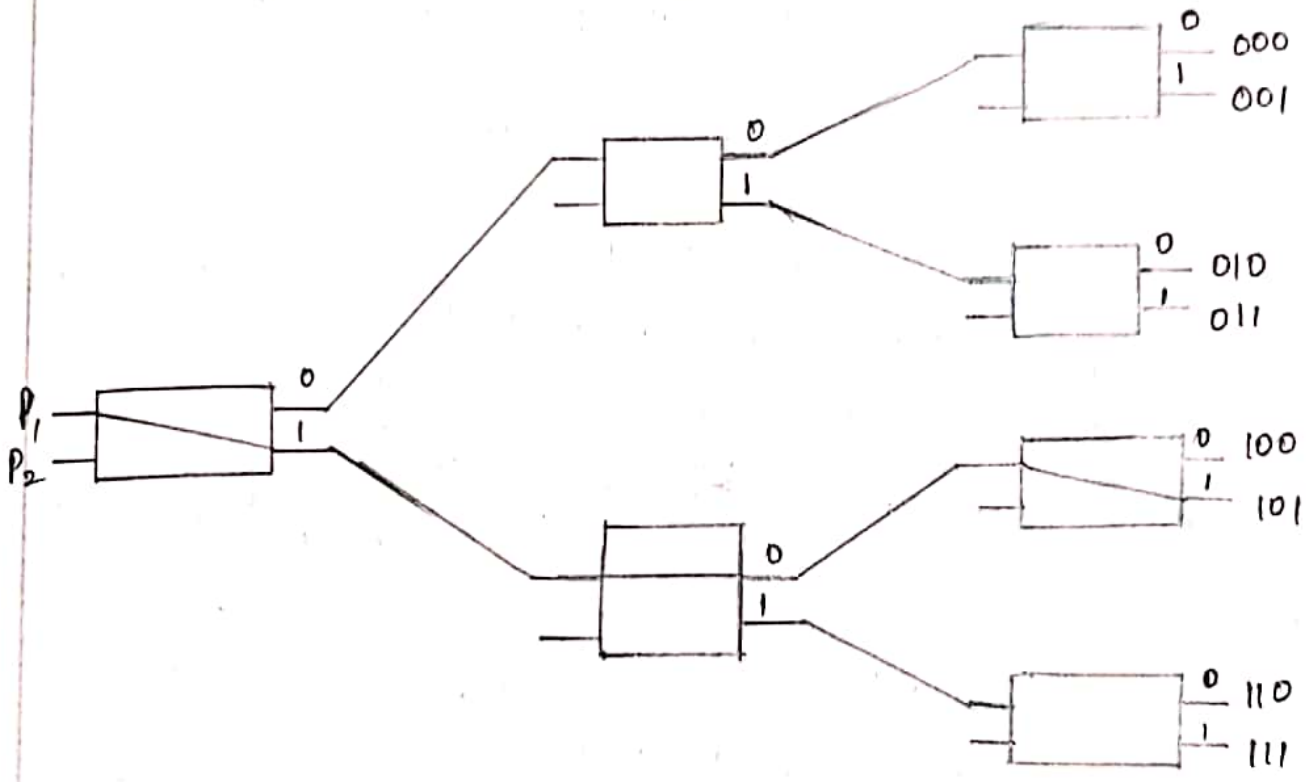


fig 16: Binary tree with 2x2 Switches.

→ The paths from a source to a destination is determined from the binary bits of the destination number.

Many different topologies have been proposed for multistage switching n/w's to control processor - memory communication in a tightly coupled multiprocessor system or to control the communication b/t the processing elements in a loosely coupled system. one such topology is the omega switching n/w. In this, there is exactly one path from each source to any particular destination.

A particular request is initiated in the switching n/w by the source, which sends a 3-bit pattern representing the destination number. Level '1' inspects the 'MSB', Level '2' inspects the middle bit, & Level '3' inspects the 'LSB'.

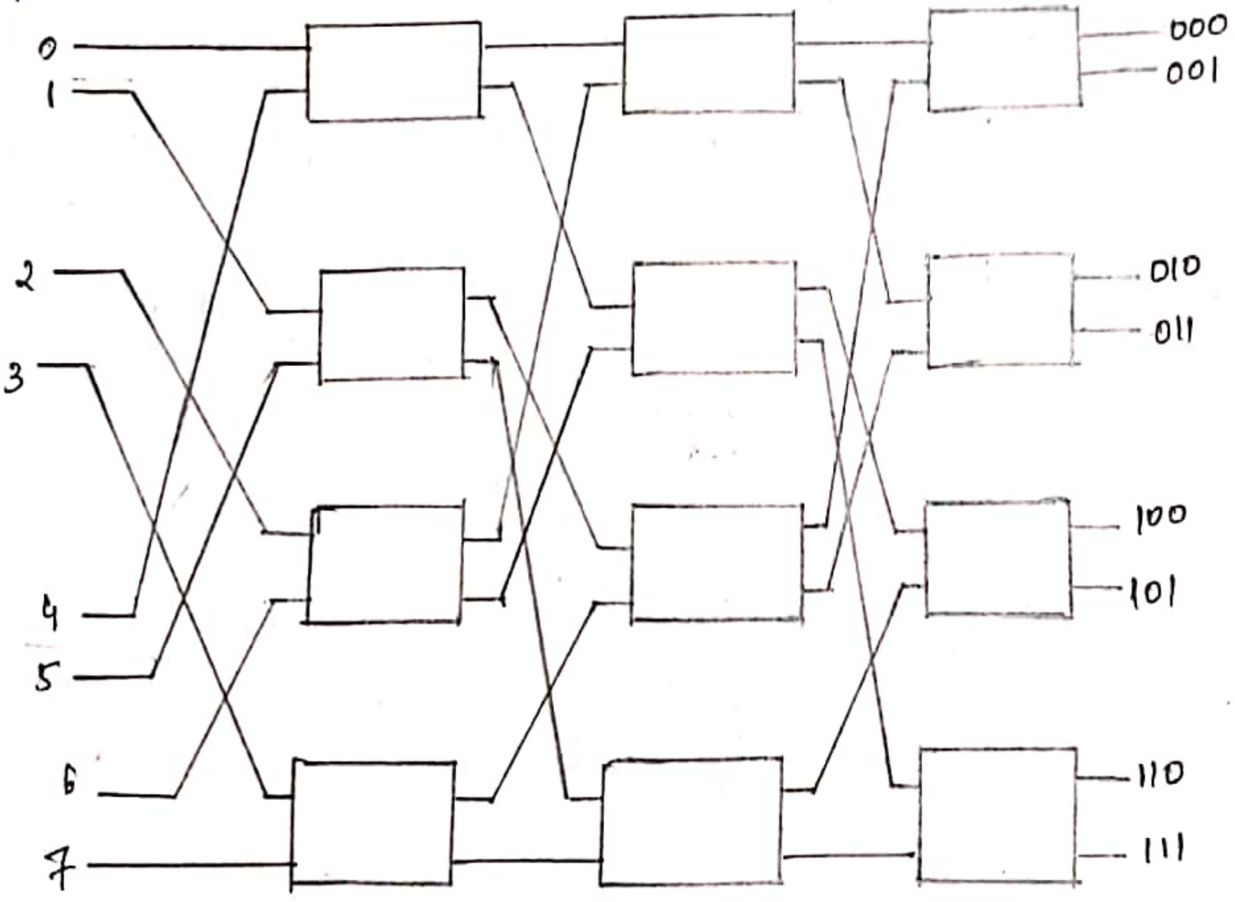
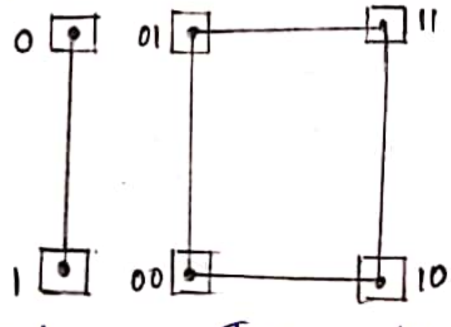


fig 17: 8×8 omega switching network

5) Hypercube Interconnection :- The hypercube or binary n-cube multiprocessor structure is a loosely coupled system composed of $N = 2^n$ processors interconnected in an n-dimensional binary cube. Each processor forms a 'node' of the cube. Each processor has direct communication paths to 'n' other neighbour processors. These paths correspond to the edges of the cube. There are 2^n distinct n-bit binary addresses that can be assigned to the processors. Each processor address differs from that of each of its 'n' neighbors by exactly one bit position.

The hypercube structure for $n = 1, 2, 4\ 3$ has n. The hypercube structure interconnected $n=1$ & $2^n = 2$. It contains two processors by a single path. A two cube structure has $n=2$ & $2^n=4$. It contains 4 nodes interconnected as a square. A 3 cube structure has eight nodes interconnected as a cube. An n-cube structure has 2^n nodes with a processor residing in each node. Each node is assigned a binary address in such a way that the addresses of two neighbors differ in exactly one bit position.



one-cube Two-cube

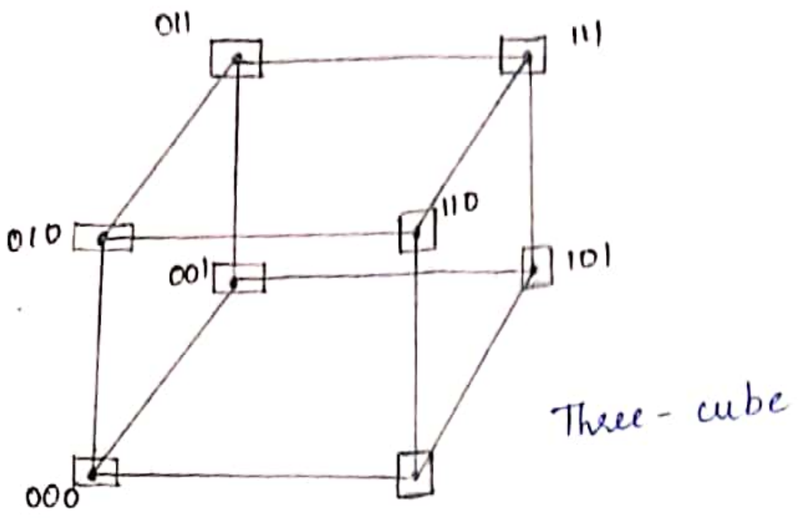


fig 15: Hypercube Structures for n= 1, 2, 3 . . ,

Inter processor Arbitration :-

Computer systems contain a no. of buses at various levels to facilitate the transfer of information b/t components. The cpu contains a no. of internal buses for ~~the transf~~ transferring info. b/t ~~-erring~~ info. b/t ~~components,~~ processor registers & ALU. A memory bus consists of lines for transferring data, address & read/ write information. A bus that connects major components in a ~~multi~~ multiprocessor system, such as Cpu's, Iop's & memory is called a System bus.

→ Data transfers over the system bus may be synchronous or asynchronous. In a synchronous bus, each data item is transferred during a time slice known in advance to both source & destination units.

→ Synchronization is achieved by driving both units from a common clock source.

→ The control lines provide signals for controlling the information transfer between units &

→ Timing signals indicate the validity of data & address information. command signals specify operations to be performed.

→ The six bus arbitration signals are used for inter-processor arbitration.

Serial Arbitration Procedure :

A hardware bus priority resolving technique can be established by means of serial | parallel connection of the units requesting control of the system bus.

→ The serial priority resolving technique is obtained from a daisy-chain connection of bus arbitration circuits similar to the priority interrupt logic.

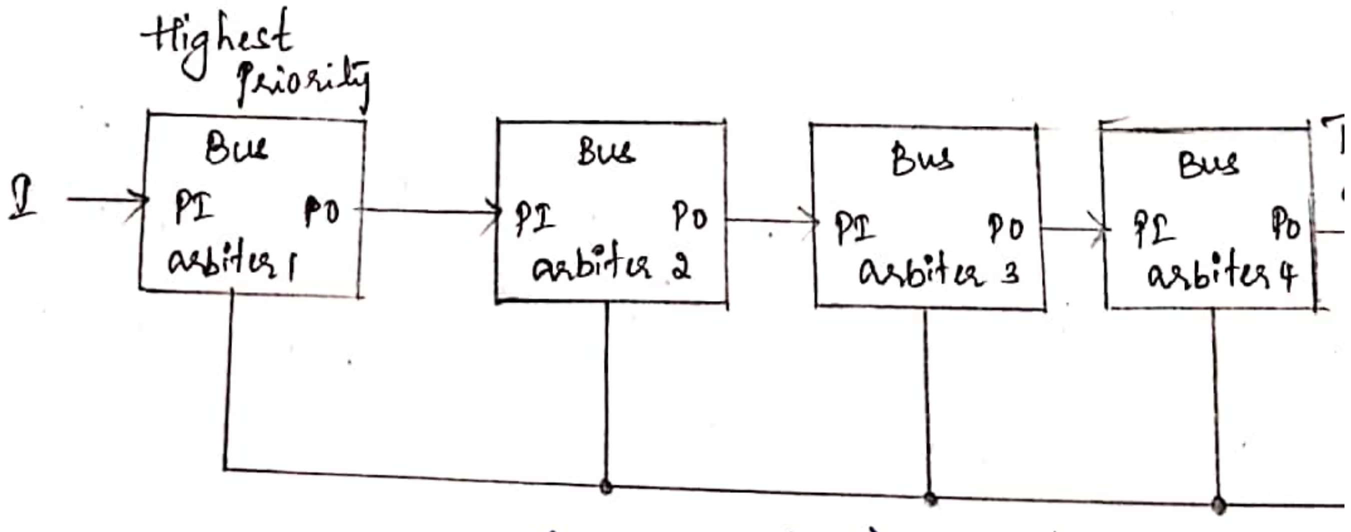


fig 19: Serial (daisy-chain) Arbitration