UNIT-5

Q-1)

a) Define pipeline processing.

Pipelining is a technique used in modern processors to improve performance by executing multiple instructions simultaneously. It breaks down the execution of instructions into several stages, where each stage completes a part of the instruction.

b) What is hardware interlock?

An interlock is a circuit that detects instructions whose source operands are destinations of instructions farther up in the pipeline.

c) Define the Arithmetic Pipeline and Instruction pipeline.

- An arithmetic pipeline divides an arithmetic problem into various sub-problems for execution in various pipeline segments. It is used for floating point operations, multiplication, and various other computations.
- An instruction pipeline reads instructions from the memory while previous instructions are being executed in other segments of the pipeline. Thus we can execute multiple instructions simultaneously. The pipeline will be more efficient if the instruction cycle is divided into segments of equal duration.

d) What do you mean by synchronous bus and asynchronous bus?

- In a synchronous bus, bus operations are synchronized with reference to a clock signal. The bus clock is generally derived from the computer system clock, however, often it is slower than the master clock. For instance, 66MHz buses are used in systems with a processor clock of over 500MHz.
- asynchronous bus that interconnects devices of a computer system where information transfers between devices are self-timed rather than controlled by a synchronizing clock signal. A connected device indicates its readiness for a transfer by activating a request signal.

e) Define LRU and FIFO.

- The least recently used (LRU) algorithm gives the highest priority to the requesting device that has not used the bus for the longest interval. The priorities are adjusted after a number of bus cycles according to the LRU algorithm.
- In the first-come, first-serve scheme, requests are served in the order received. To implement this algorithm, the bus controller establishes a queue arranged according to the time that the bus requests arrive. Each processor must wait for its turn to use the bus on a first-in, first-out (FIFO) basis.

Q-2)

a) What are the three major pipeline conflicts?

There are mainly three types of dependencies possible in a pipelined processor. These are : 1) Structural Dependency
2) Control Dependency
3) Data Dependency

Structural Dependency:

Hardware resource conflicts among the instructions in the pipeline cause structural hazards. Memory, a GPR Register, or an ALU might all be used as resources here. When more than one instruction in the pipe requires access to the very same resource in the same clock cycle, a resource conflict is said to arise. In an overlapping pipelined execution, this is a circumstance where the hardware cannot handle all potential combinations.

Control Dependency:

Branch Dependency are caused by branch instructions and are known as control hazards in computer architecture. The flow of program/instruction execution is controlled by branch instructions. Remember that conditional statements are used in higher-level languages for iterative loops and condition testing (correlate with while, for, and if case statements). These are converted into one of the BRANCH instruction variations. As a result, when the decision to execute one instruction is reliant on the result of another instruction, such as a conditional branch, which examines the condition's consequent value, a conditional hazard develops.

Data Dependency:

Data Dependency in pipelining emerge when the execution of one instruction is dependent on the results of another instruction that is still being processed in the pipeline. The order of the READ or WRITE operations on the register is used to classify data threats into three groups.

b) What are hardware interlocks?

c) Discuss the difference between tightly coupled and loosely coupled multiprocessors.

Loosely Coupled Multiprocessor System	Tightly Coupled Multiprocessor System
In this system, every processor has its own memory module.	In this system, the processors share memory modules.
It is efficient when there is less interaction between tasks running on different processors.	It is efficient when used with real−time processing.
There are no memory conflicts in general.	It has memory conflicts.
It is considered as a Message transfer system (MTS).	They are connected through networks such as PMIN, IOPIN, and ISIN.
It is less expensive.	It is expensive.
It has a low data rate.	It has a high data rate.
It provides comparatively slow speed.	It provides high speed.
They are usually seen in distributed computing systems	It is usually seen in parallel processing systems.

d) Give any three dynamic arbitration algorithms.

Dynamic priority algorithm gives the system the capability for changing the priority of the devices while the system is in operation. Few dynamic arbitration procedures that use dynamic priority algorithms: *Time Slice, Polling, LRU, FIFO*

Time Slice: In this algorithm allocates a fixed-length time slice of bus time that is offered to each processor in sequentially manner, in round-robin fashion. The service provide to each processor with this scheme is independent of its location along the bus. No preference is given to any particular device since each is allotted the same amount of time to communicate with the bus.

Polling: In a bus system that uses polling, the bus-grant signal is replaced by a set of lines called poll lines, which are connected to all units. Poll lines are used by the bus controller to define an address for each device connected to the bus. The bus controller, arrange address in a sequence through prescribed manner. When a processor that recognizes its address, it activates the bus busy-line and then accesses the bus. After a number of bus cycles, the polling process continues by choosing a different processor. The polling sequence is normally programmable, and as a result, the selection priority can be randomly under program control.

LRU: The LRU (least recently used) algorithm gives the highest priority to the requesting device that has not used the bus for the longest interval. The priorities are adjusted after a number of bus cycles according to the LRU algorithm. With this procedure, no processor is favoured over any other since the priorities are dynamically changed to give every device an opportunity to access the bus.

FIFO: In the first-come, first-serve scheme, requests are served in the order received. To implement this algorithm, the bus controller establishes a queue arranged according to the time that the bus requests arrive. Each processor must wait for its turn to use the bus on a first-in, first-out (FIFO) basis.

e) A non-pipeline system takes 50 ns to process a task. The same task can be processed in a six-segment pipeline with a clock cycle of 10 ns. Determine the speedup ratio of the pipeline for 100 tasks. What is the maximum speedup that can be achieved?

Speed up factor (S) is defined as the ratio of time required for non-pipelined execution to that of time received for pipelined execution. i.e.,

$$S = \frac{T_n}{T_p}$$

Time for non-pipelined execution per task = tn = 50 ns
Time for pipelined execution per task = tp = 10 ns
Number of stages in the pipeline = k = 6
Number of tasks = 500
Calculation:
Time for non-pipelined = Tn = 50 × 500

Time for pipelined = Tp = 1st task × k × t_p + (All Remaining Tasks (k - 1)) × t_p

Time for pipelined = T_p = 1 × 6 × 10 + (500 - 1) × 10

The speed up factor will be,

$$S = \frac{T_n}{T_p} = 4.95$$

Q-3)

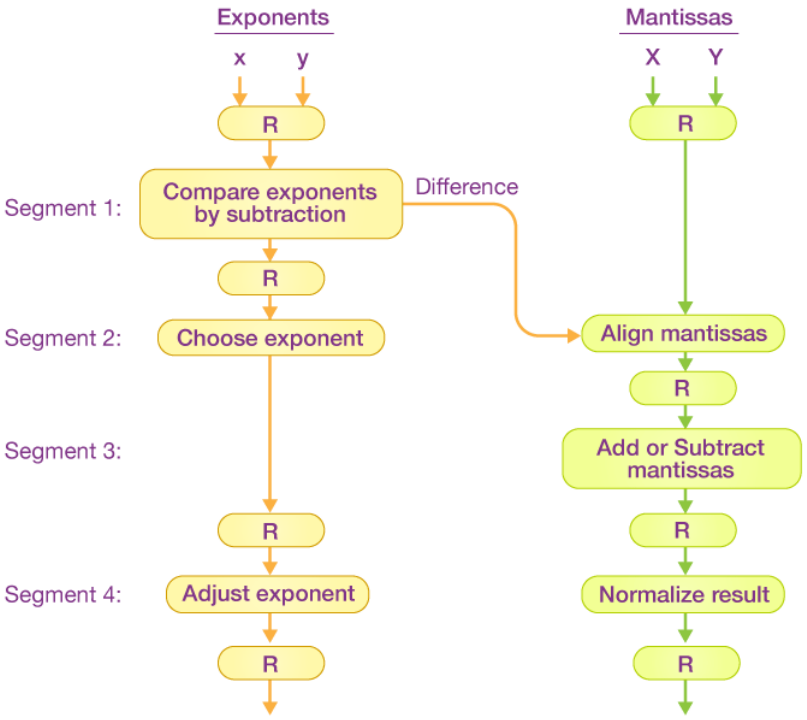
a) Discuss the arithmetic pipeline with a numerical example.

Arithmetic Pipelines are commonly used in various high-performance computers. They are used in order to implement floating-point operations, fixed-point multiplication, and other similar kinds of calculations that come up in scientific situations.
For example we perform addition and subtraction of floating points on a unit of the pipeline here.
The inputs are as follows:
A = X * 2^x = 0.9504 * 10^3
B = Y * 2^y = 0.8200 * 10^2
Where x and y refer to the exponents and X and Y refer to two fractions representing the mantissa.

The floating-point addition and subtraction process is broken into four pieces. The matching sub-operation to be executed in the specified pipeline is contained in each segment. The four segments depict the following sub-operations:

1. Comparing the exponents using subtraction
2. Aligning the mantissa
3. Adding or subtracting the mantissa
4. Normalizing the result

The sub-operations conducted in each pipeline section are depicted in the block diagram below:



1. Comparing Exponents by Subtraction

The difference between the exponents is calculated by subtracting them. The result's exponent is chosen to be the larger exponent.

The exponent difference, 3 – 2 = 1, defines the total number of times the mantissa associated with the lesser exponent should be shifted to the right.

2. Aligning the Mantissa

As per the difference of exponents calculated in segment one, the mantissa corresponding with the smaller exponent would be moved.

$A = 0.9504 * 10^3$

$B = 0.08200 * 10^3$

3. Adding the Mantissa

Both the mantissa would be added in the third segment.

$C = A + B = 1.0324 * 10^3$

4. Normalizing the Result

After the process of normalization, the result would be written as follows:

$C = 0.1324 * 10^4$

b) Discuss about SIMD Array processors.

An SIMD array processor is a computer with multiple processing units operating in parallel.

• A general block diagram of an array processor is shown below, It contains a set of identical processing elements (PEs), each having a local memory M. Each PE includes an ALU, a floating-point arithmetic unit, and working registers. Vector instructions are broadcast to all PEs simultaneously.

• Masking schemes are used to control the status of each PE during the execution of vector instructions. o Each PE has a flag that is set when the PE is active and reset when the PE is inactive.

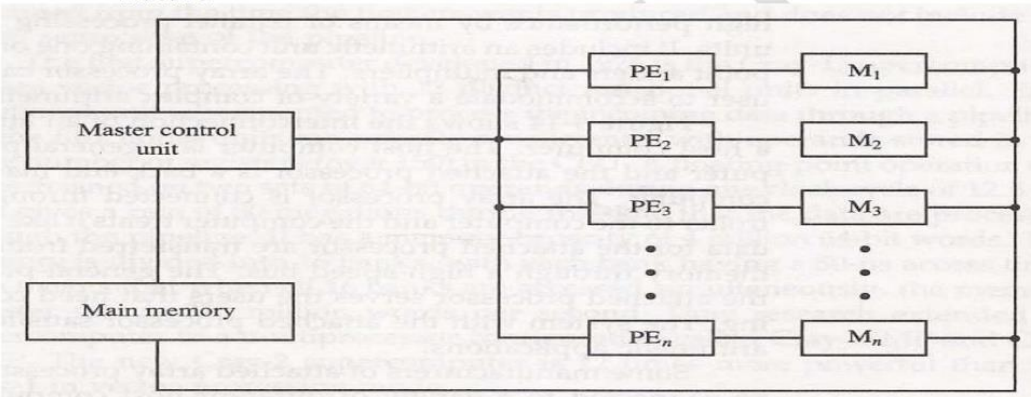


Figure 9-15 SIMD array processor organization.

• For example, the ILLIAC IV computer developed at the University of Illinois and manufactured by the Burroughs Corp. o Are highly specialized computers. o They are suited primarily for numerical problems that can be expressed in vector or matrix form.

c) Discuss the serial arbitration Procedure.

Arbitration procedures service all processor requests on the basis of established priorities. A hardware bus priority resolving technique can be established by means of a serial or 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. The processors connected to the system bus are assigned priority according to their position along the priority control line. The device closest to the priority line is assigned the highest priority. When multiple devices concurrently request the use of the bus, the device with the highest priority is granted access to it.

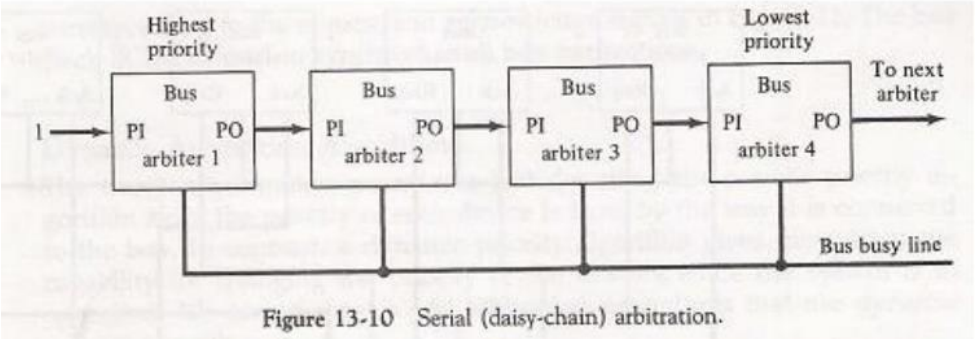


Figure 13-10 Serial (daisy-chain) arbitration.

Above figure shows the daisy-chain connection of four arbiters. It is assumed that each processor has its own bus arbiter logic with priority-in and priority-out lines. The priority out (Po) of each arbiter is connected to the priority in (PI) of the next-lower-priority arbiter. The PI of the highest-priority unit is maintained at logic 1 value. The highest-priority unit in the system will always receive access to the system bus when it requests it. The Po output for a particular arbiter is equal to 1 if its PI input is equal to 1 and the processor associated with the arbiter logic is not requesting control of the bus. This is the way that priority is passed to the next unit in the chain. It the processor requests control of the bus and the corresponding arbiter finds its PI input equal to 1, it sets its PO output to 0. Lowerpriority arbiters receive a 0 in PI and generate a 0 in Po. Thus the processor whose arbiter has a PI = 1 and Po = 0 is the one that is given control of the system bus.

A processor may be in the middle of a bus operation when a higher-priority processor requests the bus. The lower-priority processor must complete its bus operation before it relinquishes control of the bus. The bus busy line shown in Fig. 13-10 provides a mechanism for an orderly transfer of control. The busy line comes from open-collector circuits in each unit and provides a wired-OR logic connection. When an arbiter receives control of the bus (because its PI = 1 and Po = 0) it examines the busy line. If the line is inactive, it means that no other processor is using the bus. The arbiter activates the busy line and its processor takes control of the bus. However, if the arbiter finds the busy line active, it means that another processor is currently using the bus. The arbiter keeps examining the busy line while the lower-priority processor that lost control of the bus completes its operation. When the bus busy line returns to its inactive state, the higherpriority arbiter enables the busy line, and its corresponding processor can then conduct the required bus transfers.

d) Explain the different dynamic arbitration algorithms.

Dynamic priority algorithm gives the system the capability for changing the priority of the devices while the system is in operation. Few dynamic arbitration procedures that use dynamic priority algorithms: *Time Slice, Polling, LRU, FIFO*

Time Slice: In this algorithm allocates a fixed-length time slice of bus time that is offered to each processor in sequentially manner, in round-robin fashion. The service provide to each processor with this scheme is independent of its location along the bus. No preference is given to any particular device since each is allotted the same amount of time to communicate with the bus.

Polling: In a bus system that uses polling, the bus-grant signal is replaced by a set of lines called poll lines, which are connected to all units. Poll lines are used by the bus controller to define an address for each device connected to the bus. The bus controller, arrange address in a sequence through prescribed manner. When a processor that recognizes its address, it activates the bus busy-line and then accesses the bus. After a number of bus cycles, the polling process continues by choosing a different processor. The polling sequence is normally programmable, and as a result, the selection priority can be randomly under program control.

LRU: The LRU (least recently used) algorithm gives the highest priority to the requesting device that has not used the bus for the longest interval. The priorities are adjusted after a number

of bus cycles according to the LRU algorithm. With this procedure, no processor is favoured over any other since the priorities are dynamically changed to give every device an opportunity to access the bus.

FIFO: In the first-come, first-serve scheme, requests are served in the order received. To implement this algorithm, the bus controller establishes a queue arranged according to the time that the bus requests arrive. Each processor must wait for its turn to use the bus on a first-in, first-out (FIFO) basis.

Rotating Daisy-Chain: The rotating daisy-chain procedure is a dynamic extension of the daisy-chain algorithm. In this scheme there is no central bus controller, and the priority line is connected from the priority-out of the last device back to the priority-in of the first device in a closed loop. This is similar to the connections shown in below Fig. except that the PO output of arbiter 4 is connected to the PI input of arbiter 1. Whichever device has access to the bus serves as a bus controller for the following arbitration. Each arbiter priority for a given bus cycle is determined by its position along the bus priority line from the arbiter whose processor is currently controlling the bus. Once an arbiter releases the bus, it has the lowest priority.

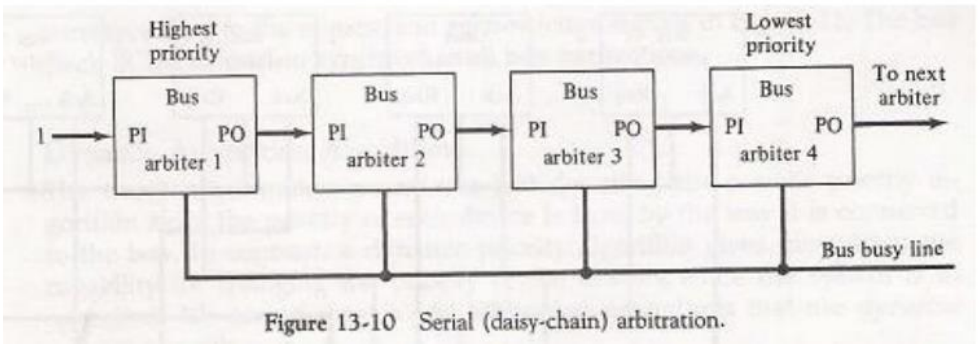


Figure 13-10 Serial (daisy-chain) arbitration.

e) Explain Mutual Exclusion with a Semaphore.

A properly functioning multiprocessor system must provide a mechanism that will guarantee orderly access to shared memory and other shared resources. This is necessary to protect data from being changed simultaneously by two or more processors. This mechanism has been termed mutual exclusion.

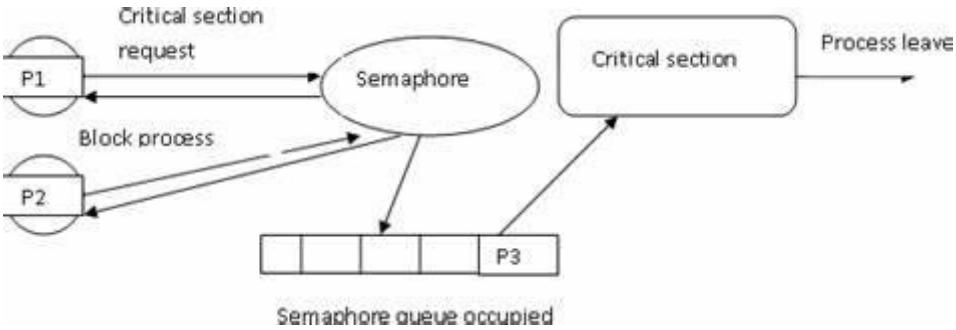
A binary variable called a semaphore is often used to indicate whether or not a processor is executing a critical section. A semaphore is a software- controlled flag that is stored in a memory location that all processors can access. When the semaphore is equal to 1, it means that a processor is executing a critical program, so that the shared memory is not available to other processors. When the semaphore is equal to 0, the shared memory is available to any requesting processor. Processors that share the same memory segment agree by convention not to use the memory segment unless the semaphore is equal to 0, indicating that memory is available. They also agree to set the semaphore to 1 when they are executing a critical section and to clear it to 0 when they are finished.

Testing and setting the semaphore is itself a critical operation and must be performed as a single indivisible operation. If it is not, two or more processors may test the semaphore simultaneously and then each set it, allowing them to enter a critical section at the same time. This action would allow simultaneous execution of critical section, which can result in erroneous initialization of control parameters and a loss of essential information.

A semaphore can be initialized by means of a test and set instruction in hardware lock conjunction with a hardware lock mechanism. A hardware lock is a processor generated signal that serves to prevent other processors from using the system bus as long as the signal is active. The test-and-set instruction tests and sets a semaphore and activates the lock mechanism during the time that the instruction is being executed. This prevents other processors from changing the semaphore between the time that the processor is testing it and the time that it is setting it. Assume that the semaphore is a bit in the least significant position of a memory word whose address is symbolized by SEM. Let the mnemonic TSL designate the "test and set while locked" operation. The instruction TSL SEM will be executed in two memory cycles (the first to read and the second to write) without interference as follows:

R←M[SEM] Test semaphore

M[SEM] ←1 Set semaphore.



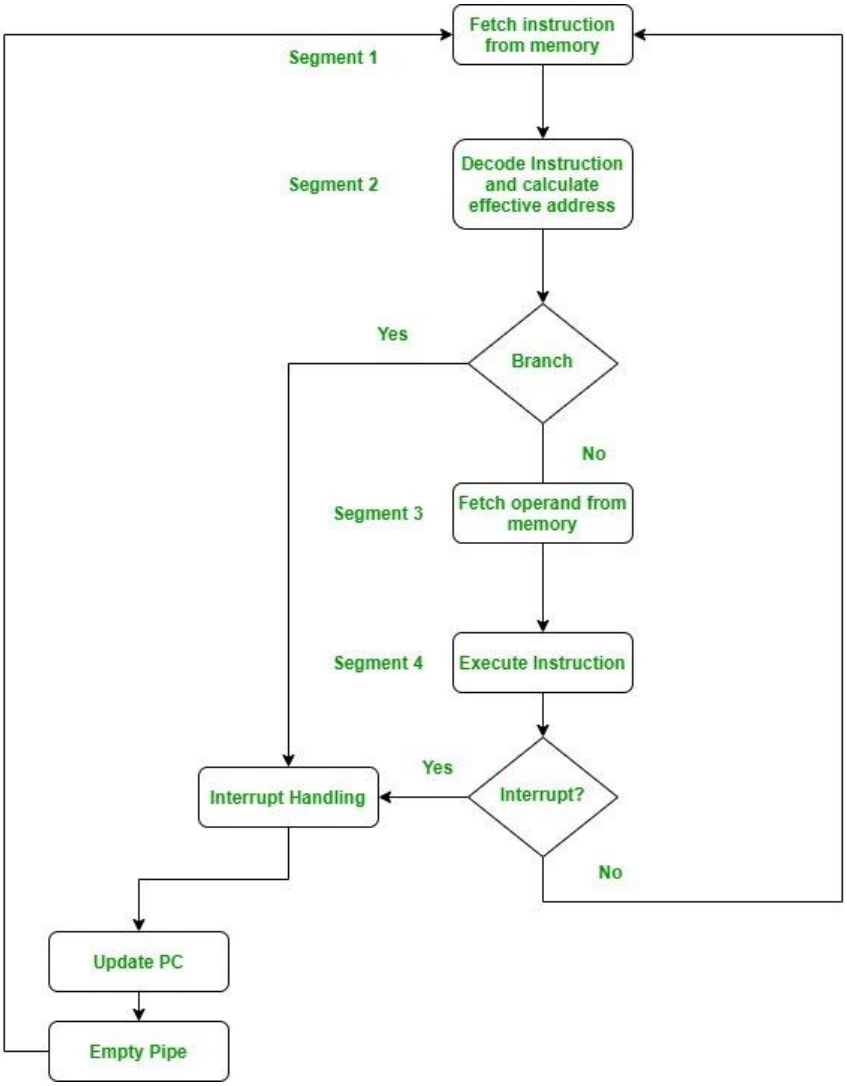
Q-4)

a) Explain the instruction pipeline in detail with a suitable example. Also, discuss the difficulties that cause the instruction pipeline to deviate from its normal operation.

In this a stream of instructions can be executed by overlapping fetch, decode and execute phases of an instruction cycle. This type of technique is used to increase the throughput of the computer system. An instruction pipeline reads instruction from the memory while previous instructions are being executed in other segments of the pipeline. Thus we can execute multiple instructions simultaneously. The pipeline will be more efficient if the instruction cycle is divided into segments of equal duration. In the most general case computer needs to process each instruction in following sequence of steps:

1. **Fetch the instruction from memory (FI)**
2. **Decode the instruction (DA)**
3. **Calculate the effective address**
4. **Fetch the operands from memory (FO)**
5. **Execute the instruction (EX)**
6. **Store the result in the proper place**

The flowchart for instruction pipeline is shown below.



Example:

Instruction Branch

Stage	1	2	3	4	5	6	7	8	9	10	11	12	13
1	FI	DA	FO	EX									
2		FI	DA	FO	EX								
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

Here the instruction is fetched on first clock cycle in segment 1. Now it is decoded in next clock cycle, then operands are fetched and finally the instruction is executed. We can see that here the fetch and decode phase overlap due to pipelining. By the time the first instruction is being decoded, next instruction is fetched by the pipeline. In case of third instruction we see that it is a branched instruction. Here when it is being decoded 4th instruction is fetched simultaneously. But as it is a branched instruction it may point to some other instruction when it is decoded. Thus fourth instruction is kept on hold until the branched instruction is executed.

When it gets executed then the fourth instruction is copied back and the other phases continue as usual.

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

- Resource conflicts caused by access to memory by two segments at the same time. It can be resolved by using separate instruction and data memories.
- Data dependency conflicts arise when an instruction depends on the result of a previous instruction, but this result is not yet available.
- Branch difficulties arise from branch and other instructions that change the value of PC.

b) Discuss the different physical forms available for interconnection structures.

There are different physical forms available for establishing an interconnection network between various components of the computer system. These physical forms are known as Interconnection structures.

There are five types of Interconnection Structures:

1. **Time-shared common bus.**
2. **Multiport memory.**
3. **Crossbar switch.**
4. **Multistage switching network.**
5. **Hypercube system.**

Time-shared common bus

In any multiprocessor system, the time-shared common bus interconnection structures provide a common communication path by connecting all the functional units like I/O processor, processor, memory unit, etc. The figure below displays a multiprocessor system with a common communication path (single bus).

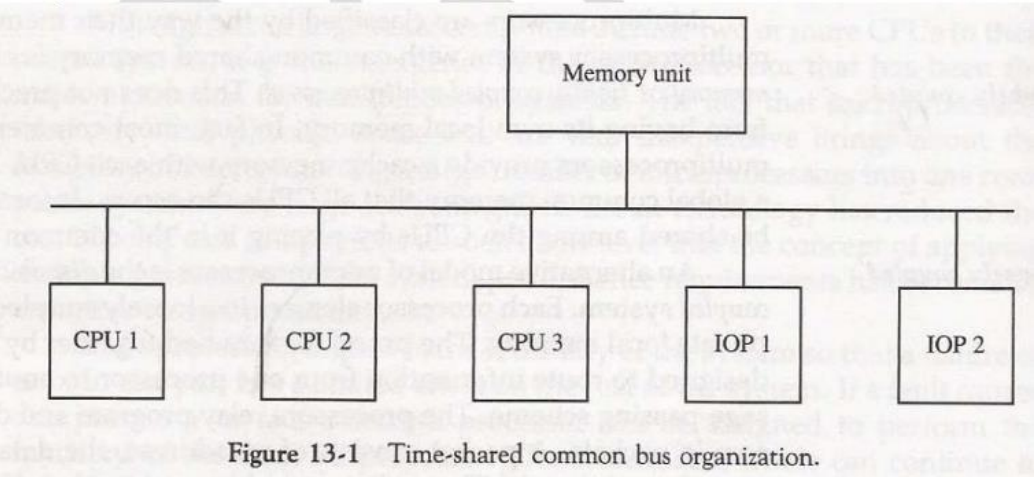


Figure 13-1 Time-shared common bus organization.

The processor needs the common bus to communicate with a functional unit to transfer data. To do so, the processor first checks if the bus is available or not. The processor can only use the common bus if it is free. The processor puts the destination unit address on the common bus, and the destination unit identifies it. A command is issued to tell the receiver unit what

work is to be done to communicate with a functional unit. The other functional units at that time will be either busy in internal operations or will sit free, waiting to get the common bus. We can resolve memory access conflict with methods such as First-In-Out (FIFO) queues, static & fixed priorities, and daisy chains can be used.

Multiport memory

A multiport memory structure employs separate buses for every memory module and CPU. Every processor in a multiport memory is connected to each memory unit. The below figure shows multiport memory interconnection structures.

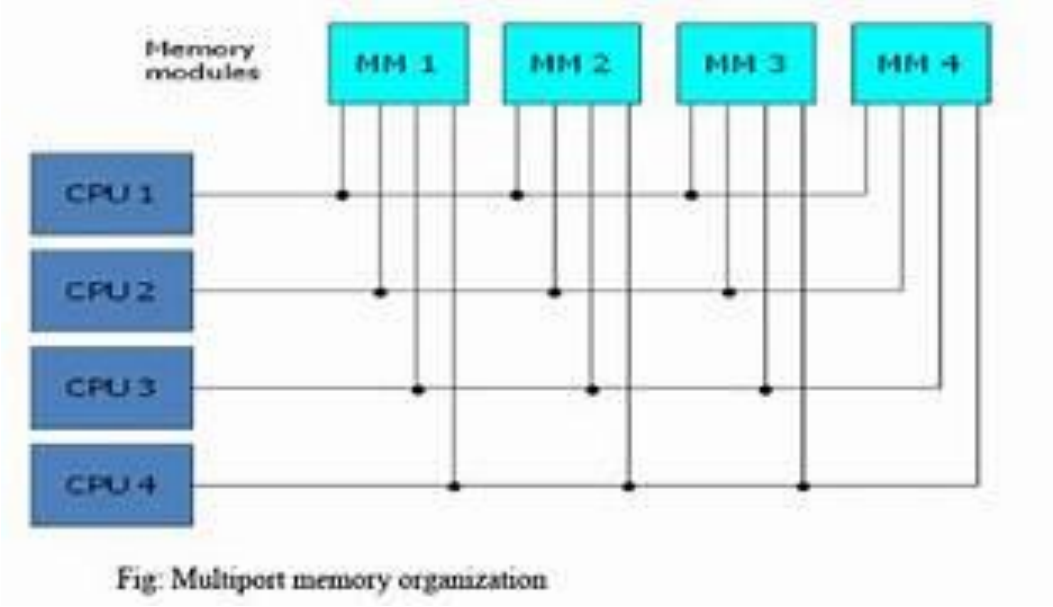


Fig: Multiport memory organization

A processor bus consists of the data, control lines, and address required for communication with the memory. The memory unit is said to have three ports, and each port connects with one of the buses. The module should have internal control logic to determine which port will have access to memory at the given time.

Crossbar switch

The crossbar switch system consists of several intersection crosspoints between processor buses and memory module paths. Let's understand its structure with the below figure.

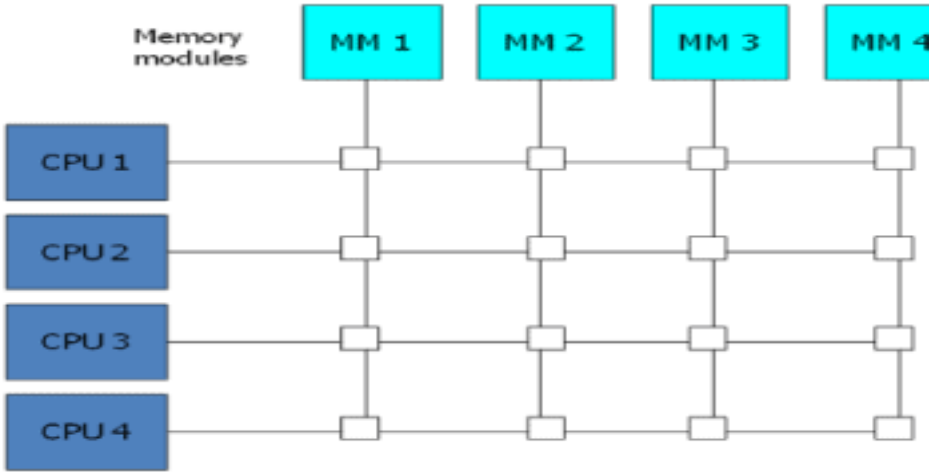


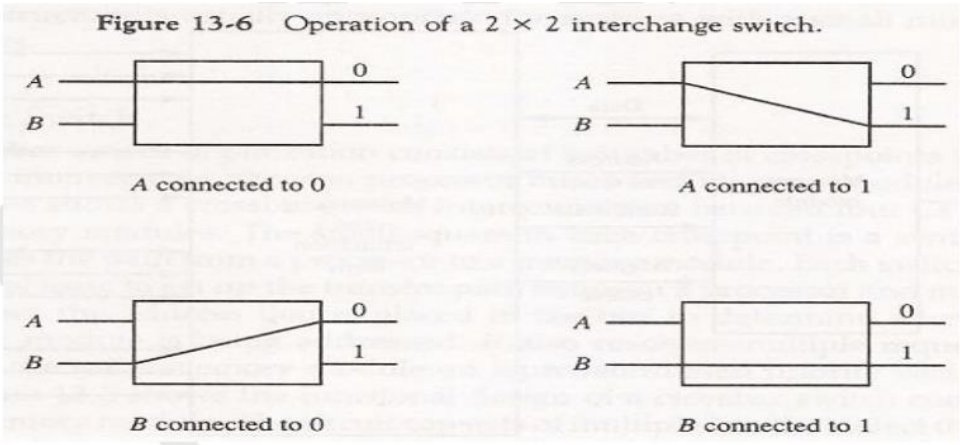
Fig: Crossbar switch

In every crosspoint, the tiny box represents a switch that can obtain the path from a processor to the memory module. Every switch point has to control logic for setting up the transfer path among a processor and memory unit. It can calculate the address placed in the bus to obtain if its specific module is addressed. In addition, it can eliminate multiple requests to access the same memory module on a priority basis.

Multistage Switching Network

The basic component of a multistage network is a two-input, two-output interchange switch interchange switch. As shown in below Fig. the 2 X 2 switch has two input labeled A and B, and two outputs, labeled 0 and 1. There are control signals (not shown) associated with the switch that establish the interconnection between the input and output terminals. The switch has the capability connecting input A to either of the outputs. Terminal B of the switch behaves in a similar fashion. The switch also has the capability to arbitrate between conflicting requests. If inputs A and B both request the same output terminal only one of them will be connected; the other will be blocked.

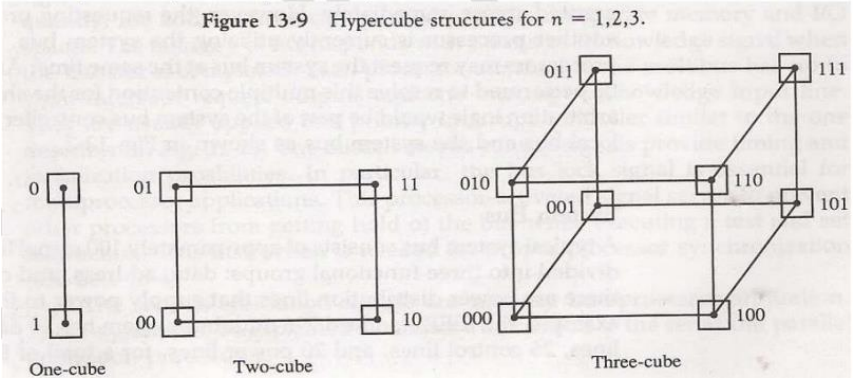
Figure 13-6 Operation of a 2 × 2 interchange switch.



Hypercube Interconnection

The hypercube or binary n-cube multiprocessor structure is a loosely coupled system composed of N = 2n processors interconnected in an n-dimensional binary cube. Each processor forms a node of the cube. Although it is customary to refer to each node as having a processor, in effect it contains not only a CPU but also local memory and I/O interface. Each processor has direct communication paths to n other neighbor processors. These paths correspond to the edges of the cube. There are 2n 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.

Figure 13-9 Hypercube structures for $n = 1,2,3$.



c) What are the characteristics of RISC and CISC? Also, give the differences between them.

Characteristics of Reduced Instruction Set Architecture (RISC):

- Simpler instruction, hence simple instruction decoding.
- Instruction comes undersize of one word.
- Instruction takes a single clock cycle to get executed.
- More general-purpose registers.
- Simple Addressing Modes.
- Fewer Data types.
- A pipeline can be achieved.

Characteristics of Complex Instruction Set Architecture (CISC):

1. The length of the code is shorts, so it requires very little RAM.
2. CISC or complex instructions may take longer than a single clock cycle to execute the code.
3. Less instruction is needed to write an application.
4. It provides easier programming in assembly language.
5. Support for complex data structure and easy compilation of high-level languages.
6. It is composed of fewer registers and more addressing nodes, typically 5 to 20.
7. Instructions can be larger than a single word.
8. It emphasizes the building of instruction on hardware because it is faster to create than the software.

Difference between the RISC and CISC Processors

RISC	CISC
It is a Reduced Instruction Set Computer.	It is a Complex Instruction Set Computer.
It emphasizes on software to optimize the instruction set.	It emphasizes on hardware to optimize the instruction set.
It is a hard wired unit of programming in the RISC Processor.	Microprogramming unit in CISC Processor.
It requires multiple register sets to store the instruction.	It requires a single register set to store the instruction.
RISC has simple decoding of instruction.	CISC has complex decoding of instruction.

Uses of the pipeline are simple in RISC.	Uses of the pipeline are difficult in CISC.
It uses a limited number of instruction that requires less time to execute the instructions.	It uses a large number of instruction that requires more time to execute the instructions.
It uses LOAD and STORE that are independent instructions in the register-to-register a program's interaction.	It uses LOAD and STORE instruction in the memory-to-memory interaction of a program.
The execution time of RISC is very short.	The execution time of CISC is longer.
RISC architecture can be used with high-end applications like telecommunication, image processing, video processing, etc.	CISC architecture can be used with low-end applications like home automation, security system, etc.
The program written for RISC architecture needs to take more space in memory.	Program written for CISC architecture tends to take less space in memory.
Example of RISC: ARM, PA-RISC, Power Architecture, Alpha, AVR, ARC and the SPARC.	Examples of CISC: VAX, Motorola 68000 family, System/360, AMD and the Intel x86 CPUs.