SECOND YEAR MSC COMPUTER SCIENCE

PARALLEL PROCESSING - P16CSE3A

Objective:

To study the Parallel computer Architecture, theories of parallel computing, interconnection networks and applications of cost effective computer systems.

UNIT I

Introduction to Parallel Processing – Evolution of Computer Systems – Parallelism in Uniprocessor Systems – Parallel Computer Structures – Architectural Classification Schemes– Parallel Processing Applications.

UNIT II

Memory and Input-Output Subsystems – Hierarchical Memory Structure – Virtual Memory System – Memory Allocation and Management – Cache Memories and Management – Input-Output Subsystems.

UNIT III

Principles of Pipelining and Vector Processing – Pipelining : An Overlapped Parallelism – Instruction and Arithmetic Pipelines – Principles of Designing Pipelined Processors – Vector Processing Requirements.

UNIT IV

Vectorization and Optimization methods – Parallel Languages for Vector Processing – Design of Vectorizing Compiler – Optimization of Vector Functions – SIMD Array Processors – SIMD Interconnection Networks

UNIT V

Multiprocessors Architecture and Programming – Functional Structures – Interconnection Networks - Parallel Memory Organizations – Multiprocessor Operating Systems – Language Features to Exploit Parallelism – Multiprocessor Scheduling Strategies.

Text Book:

Kai Hwang and Faye A. Briggs,Computer Architecture and Parallel Processing,McGraw Hill International Edition, 1985.

UNIT I Chapter 1 Section 1.1 – 1.5

UNIT II Chapter 2 Sections 2.1 – 2.5

UNIT III Chapter 3 Sections 3.1 – 3.4

UNIT IV Chapter 4 Sections 4.5 , Chapter 5 Sections 5.1 ,5.2 , 5.4

UNIT V Chapter 7 7.1 – 7.4, 7.5-7.5.1, Chapter 8 Sections 8.3

Books for Reference:

1. Richard Kain, Advanced Computer Architecture, PHI, 1999.

2. V. Rajaraman and C. Siva Ram Murthy, Parallel Computers, Architecture and Programming, PHI, 2000.

UNIT I

Introduction to Parallel Processing – Evolution of Computer Systems – Parallelism in Uniprocessor Systems – Parallel Computer Structures – Architectural Classification Schemes– Parallel Processing Applications.

Introduction to parallel processing.

Content:

- Evolution of Computer Systems
- Parallelism in Uniprocessor system
- Parallel Computer Structures
- Architectural classification schemes
- Parallel processing application

1.1 **Evolution of Computers / Generation of Computers:**

- A computer is an electronic device that manipulates information or data.
- It has the ability to store, retrieve, and process data. Nowadays, a computer can be used to type documents, send email, play games, and browse the Web.
- It can also be used to edit or create spreadsheets, presentations, and even videos.
- But the evolution of this complex system started around 1940 with the first Generation of Computer and evolving ever since.

There are five generations of computers:

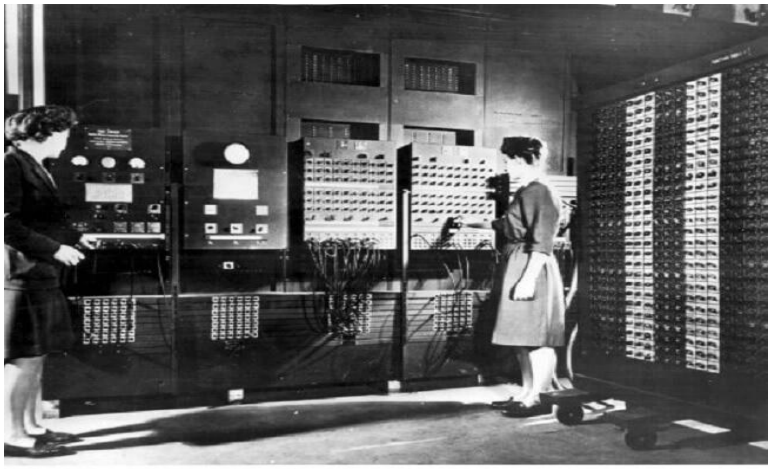
- First Generation Of Computer: Vacuum Tubes (1940–1956)
- Second Generation Of Computer: Transistors (1956–1963)
- Third Generation Of The Computer: Integrated Circuits (1964–1971)
- Fourth Generation Of Computer: Microprocessors

  (1971–2010)
- Fifth Generation Of Computer: Artificial Intelligence

  (2010 — Present)

First GenTubeseration Of Computer:

Vacuum (1940–1956)



Introduction:

1. 1946-1959 is the period of first generation computer.
2. J.P.Eckert and J.W.Mauchy invented the first successful electronic computer called ENIAC, ENIAC stands for "Electronic Numeric Integrated And Calculator".
3. During this generation, computers were built with vacuum tubes.
4. These electronic tubes were made of glass and were about the size of a light bulb.were

Vaccum Age

- 1951-1953– IBM sells over 1,000 IBM 650 systems.
- 1952– Dr. Grace Hopper introduces the A6 Compiler, which is the first example of software that converts high-level language symbols into instructions that a computer can execute

Advantage:

- ✓ The computer was very fast to calculate.
- ✓ The vacuum tube technology made possible the advent of electronic computers.
- ✓ Those days this was the only electronic.

Disadvantage:

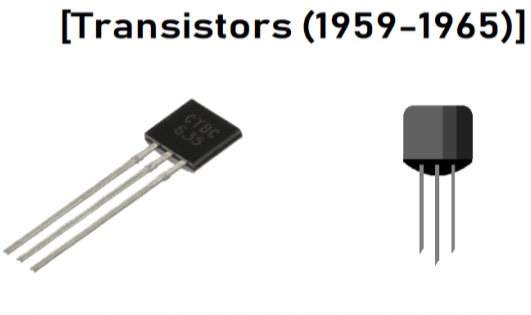
- ✓ The first generation of computer is not portable.
- ✓ It is not reliable devices.
- ✓ An air-conditioned is required.

Examples:

1. ENIAC
2. EDVAC
3. UNIVAC
4. IBM-701
5. IBM-650

Second Generation Of Computer:

Transistors (1956–1963)



Introduction:

1. 1959-1965 is the period of second-generation computer.
2. Second generation computers were based on Transistor instead of vacuum tubes.
3. This generation begins with the first computers built with transistors— small devices that transfer electronic signals across a resistor.
4. Because transistors are much smaller, use less power, and create less heat than vacuum tubes, the new computers were faster, smaller, and more reliable than the first-generation machines were.

Transistors Age

- 1959—Introduction of the removable disk pack, providing users with fast access to stored data.
- 1963--ASCII (American Standard Code for Information Interchange) introduced which enables computers to exchange information.

Advantage:

- ✓ More reliable than the first generation.
- ✓ Good speed and can calculate the data in the microseconds.
- ✓ Also used assembly languages.
- ✓ Smaller in the size as compared to the first generation.
- ✓ Use less amount of energy.

- ✓ Portable
- ✓ Accuracy is improved than its predecessor.

Disadvantage:

- ✓ Constant maintenance is required to work properly.
- ✓ Commercial production was very difficult.
- ✓ Still punched cards were used for input.
- ✓ The cooling system is required.
- ✓ More expensive and non-versatile.
- ✓ Used for specific purposes.

Examples:

1. Honeywell 400
2. IBM 7094
3. CDC 1604
4. CDC 3600
5. UNIVAC 1108

Third Generation Of The Computer:

Integrated Circuits (1964–1971)

Introduction:

1. 1965-1971 is the period of third generation computer.
2. These computers were based on Integrated circuits.
3. IC was invented by Robert Noyce and Jack Kilby In 1958-1959.
4. IC was a single component containing number of transistors.
5. In 1964, computer manufacturers began replacing transistors with integrated circuits.
6. An integrated circuit (IC) is a complete electronic circuit on a small chip made of silicon.
7. These computers were more reliable and compact than computers made with transistors, and they cost cost less to manufacture..

Integrated circuit age:

- 1965—Digital Equipment Corporation (DEC) introduces the first minicomputer.
- 1969—Introduction of ARPANET and the beginning of the Internet.

Advantages:

- ✓ These computers were cheaper as compared to second-generation computers.
- ✓ They were fast and reliable.
- ✓ Use of IC in the computer provides the small size of the computer.
- ✓ This generation of computers has big storage capacity.
- ✓ Instead of punch cards, mouse and keyboard are used for input.

Disadvantages:

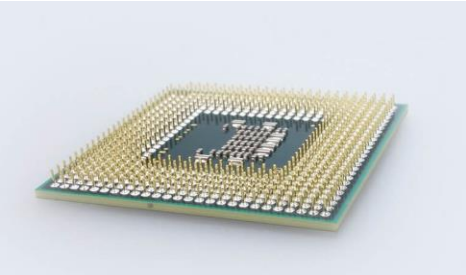
- ✓ IC chips are difficult to maintain.
- ✓ The highly sophisticated technology required for the manufacturing of IC chips.
- ✓ Air conditioning is required.

Examples:

1. PDP-8
2. PDP-11
3. ICL 2900
4. IBM 360
5. IBM 370

Fourth Generation Of Computer:

Microprocessors (1971–2010)



Introduction:

1. 1971-1980 is the period of fourth generation computer.
2. This technology is based on Microprocessor.
3. A microprocessor is used in a computer for any logical and arithmetic function to be performed in any program.

4. Graphics User Interface (GUI) technology was exploited to offer more comfort to users.
5. There are many key advancements that were made during this generation, the most significant of which was the use of the microprocessor—a specialized chip developed for computer memory and logic.
6. This revolutionized the computer industry by making it possible to use a single chip to create a smaller "personal" computer (as well as digital watches, pocket calculators, copy machines, and so on).

Microprocessor Age

- 1977—Apple Computer Inc. was Established.
- 1981—Introduction of the IBM PC, which contains an Intel microprocessor chip and Microsoft's MS-DOS operation system.
- 1990—Microsoft releases Windows 3.0 with the ability to run multiple applications.

Advantage:

- ✓ Fastest in computation and size get reduced as compared to the previous generation of computer.
- ✓ Heat generated is negligible.
- ✓ Small in size as compared to previous generation computers.
- ✓ Less maintenance is required.
- ✓ All types of high-level language can be used in this type of computers.

Disadvantage:

- ✓ The Microprocessor design and fabrication are very complex.
- ✓ Air conditioning is required in many cases due to the presence of ICs.
- ✓ Advance technology is required to make the ICs.

Examples:

1. IBM 4341
2. DEC 10
3. STAR 1000

Fifth Generation Of Computer:

Artificial Intelligence (2010 — Present)



Introduction:

1. The period of the fifth generation in 1980-onwards.
2. This generation is based on artificial intelligence.
3. The aim of the fifth generation is to make a device which could respond to natural language input and are capable of learning and self-organization.
4. This generation is based on ULSI(Ultra Large Scale Integration) technology resulting in the production of microprocessor chips having ten million electronic component.
5. Our current generation has been referred to as the "Connected Generation" because of the industry's massive effort to increase the connectivity of computers.
6. The rapidly expanding Internet, World Wide Web, and intranets have created an information superhighway that has enabled both computer professionals and home computer users to communicate with others across the globe.

Artificial Intelligence

- 1993-2000—Intel and Microsoft lead the way in computer upgrades.
- 1997 of Internet and WWW users estimated at 50 million.
- By 2004—A private Internet—Internet2—expected to be completed w/ higher speed, limited access, & tighter security is in the works. Expected to include advanced virtual reality interfaces called nanomanipulators.

Advantage:

- ✓ It is more reliable and works faster.
- ✓ It is available in different sizes and unique features.
- ✓ It provides computers with more user-friendly interfaces with multimedia features

Disadvantage:

- ✓ They need very low-level languages.
- ✓ They may make the human brains dull and doomed.

Examples:

1. Desktop
2. Laptop
3. NoteBook
4. UltraBook
5. Chromebook

1.2 **PARALLEL PROCESSING IN UNIPROCESSOR SYSTEM.**

- Most General Uniprocessor system have the same basic structure
  - Uniprocessor Architecture
  - Parallel Processing Mechanisms
  - Balancing of System Bandwidth
  - Multiprogramming and Time sharing

Uniprocessor Architecture:

- The system with a single processor is a uniprocessor system.
- It consists of three major components
  - The Main Memory
  - The Central Processing Systems
  - The Input-Output Subsystem

1. **Main Memory**

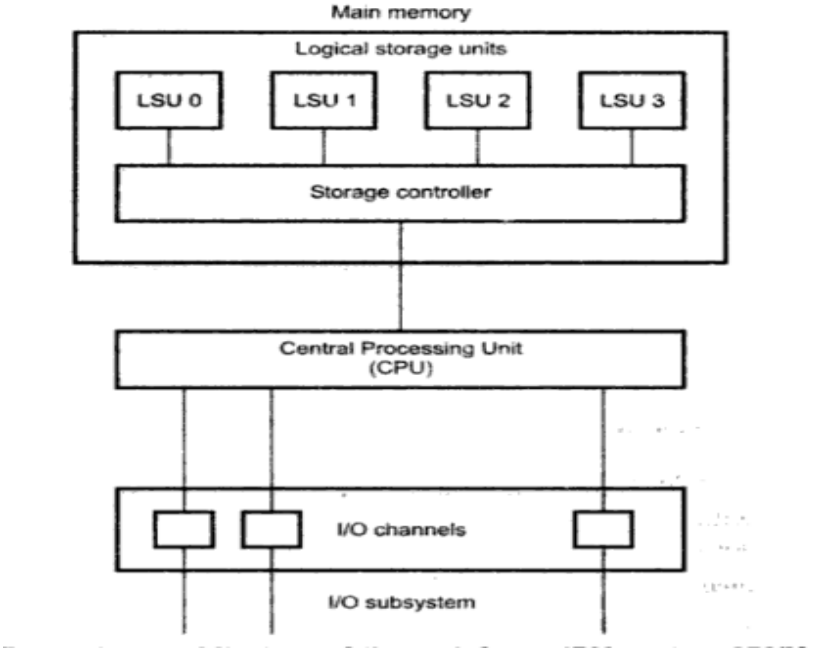
- It is divided into four units, referred to as Logical Storage Units (LSU), that are four way interleaved
- The storage controller Provides multiport connections between the CPU and the four LSU's

2. CPU

- The CPU contains the instruction decoding and execution units as well as a cache

3. I/O SubSystem

- Peripherals are connected to the system via high-speed I/O channels which operate asynchronously with the CPU

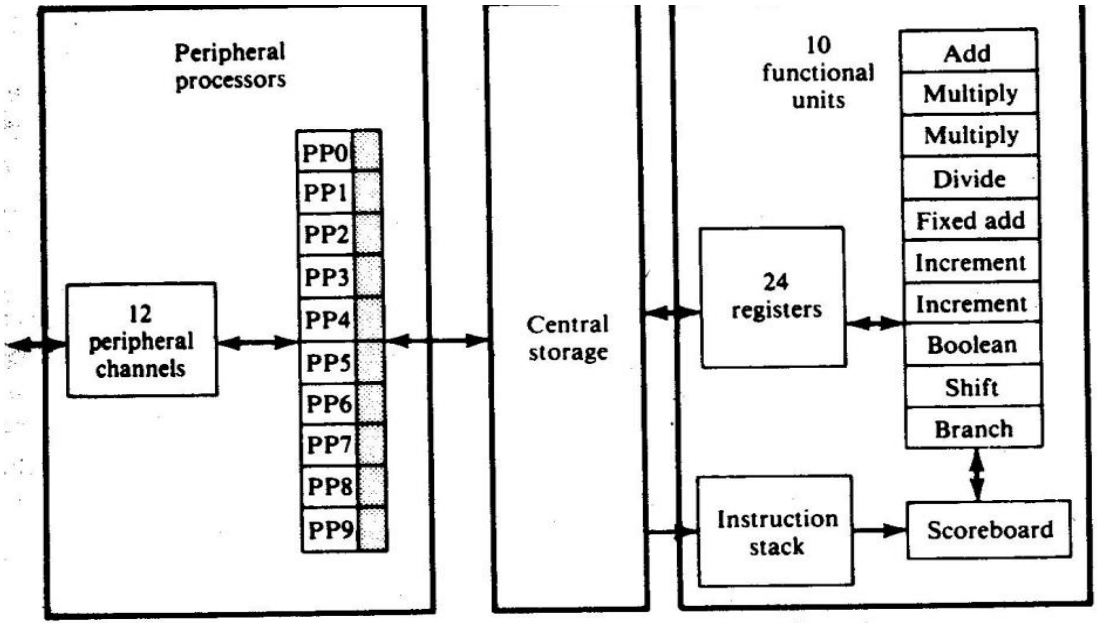


Parallel Processing Mechanisms:

- A number of parallel processing mechanism have been developed in uniprocessor computers. Those are,
  - Multiple functional units
  - Parallelism and pipelining within the CPU
  - Overlapped CPU and IO operation
  - Use of hierarchical memory system

Multiple functional units:

- Early computers
  - one ALU that perform one operation at a time.
  - Slow process
- Multiple and specialized functional units.
  - operate in parallel.
- IBM 360/91 →
  - two parallel execution units (fixed and floating point arithmetic)
- CDC-6600 →
  - 10 functional units



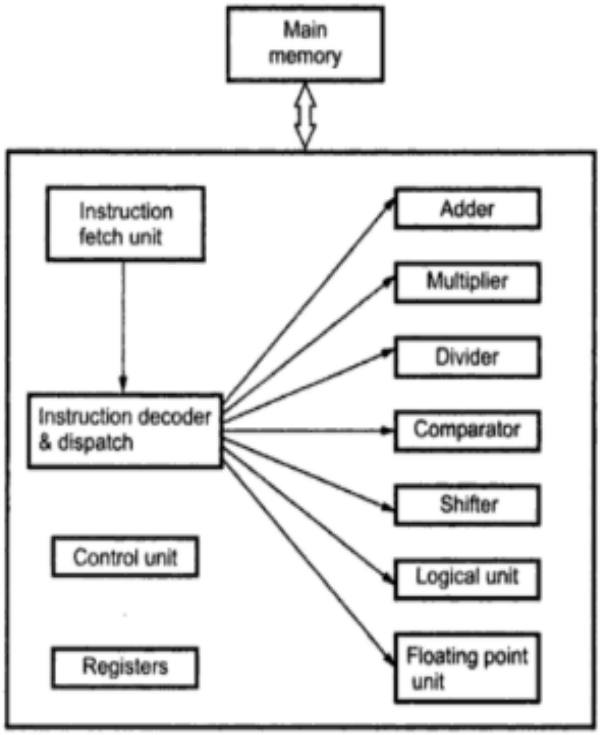


Fig. 1.3 Multiple functional units in a single processor

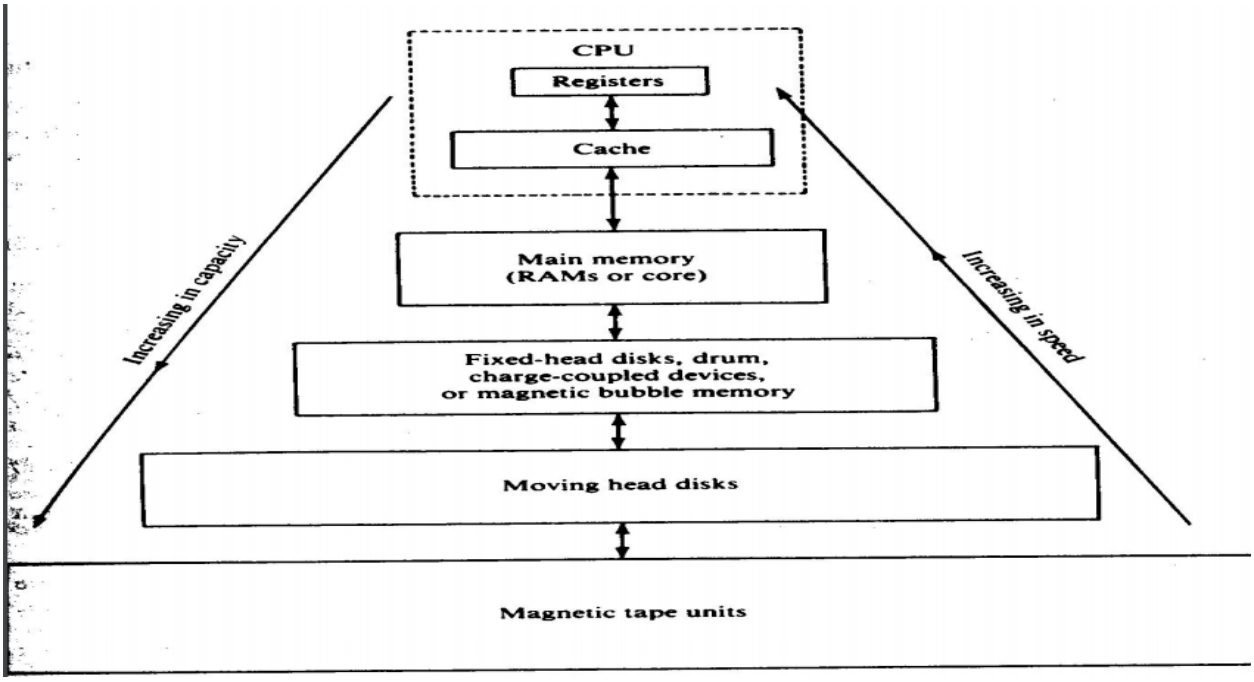
Parallelism and pipelining within the CPU:

- Parallel Adders
  - bit serial adders.
  - carry-lookahead and carry save adders.
- Multiplier recoding and convergence division.
- Phases of instruction execution are pipelined
  - Instruction fetch, decode, operand fetch, arithmetic logic execution, store result
- Instruction Prefetch and data buffering.

Overlapped CPU and IO operation:

- I/O operations performed simultaneously with CPU computations
- separate I/O controllers, channels or I/O processors.
- DMA channels – cycle stealing.

Use of hierarchical memory system:



Balancing of subsystems:

- CPU is the fastest unit in computer. The bandwidth of a system is defined as the number of operations performed per unit time. In case of main memory the memory bandwidth is measured by the number of words that can be accessed per unit time.
- Bandwidth Balancing Between CPU and MemoryThe speed gap between the CPU and the main memory can be closed up by using fast cache memory between them. A block of memory words is moved from the main memory into the cache so that immediate instructions can be available most of the time from the cache.
- Bandwidth Balancing Between Memory and I/O DevicesInput-output channels with different speeds can be used between the slow I/O devices and the main memory. The I/O channels perform buffering and multiplexing functions to transfer the data from multiple disks into the main memory by stealing cycles from the CPU.

Multiprogramming:

- Within the same interval of time, there may be multiple processes active in a computer, competing for memory, I/O and CPU resources. Some computers are I/O bound and some are

- CPU bound. Various types of programs are mixed up to balance bandwidths among functional units.
  Example Whenever a process P1 is tied up with I/O processor for performing input output operation at the same moment CPU can be tied up with an process P2. This allows simultaneous execution of programs. **The interleaving of CPU and I/O operations among several programs is called as Multiprogramming.**

Time-Sharing:

- The mainframes of the batch era were firmly established by the late 1960s when advances in semiconductor technology made the solid-state memory and integrated circuit feasible. These
- advances in hardware technology spawned the minicomputer era. They were small, fast, and
- inexpensive enough to be spread throughout the company at the divisional level. Multiprogramming mainly deals with sharing of many programs by the CPU. Sometimes high priority programs may occupy the CPU for long time and other programs are put up in queue.
- This problem can be overcome by a concept called as Time sharing in which every process is allotted a time slice of CPU time and thereafter after its respective time slice is over CPU is allotted to the next program if the process is not completed it will be in queue waiting for the second chance to receive the CPU time

1.3 **PARALLEL COMPUTER STRUCTURES.**

- Pipeline Computers
- Array Processors
- Multiprocessors systems

Pipelined processor:

- It contain four stages
  - IF - Instruction Fetch
  - ID – Instruction Decoding
  - OF – Operation Fetch
  - EX – Execution
  -

Functional Structure of Pipeline Computer:

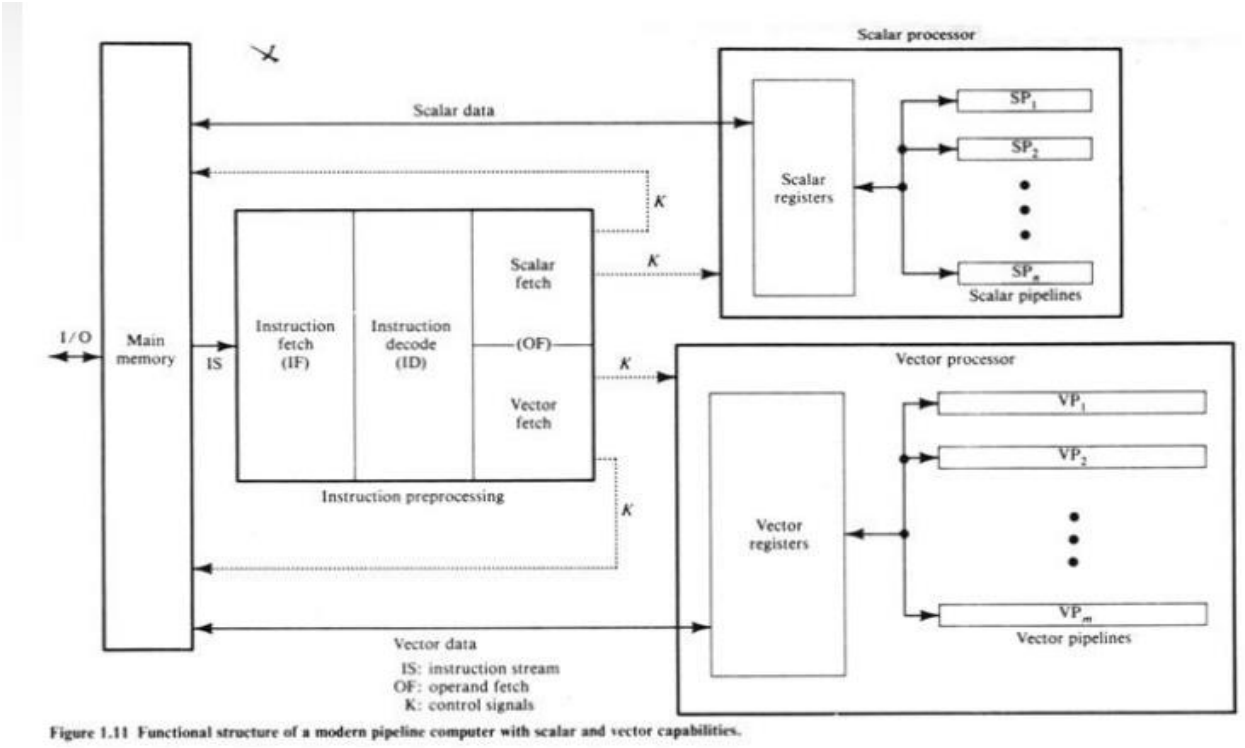


Figure 1.11 Functional structure of a modern pipeline computer with scalar and vector capabilities.

Array Processor:

- Array processor is a synchronous parallel computer with multiple ALU , called as Processing elements PE, these PE's can operate in parallel mode.
- An appropriate data routing algorithm must be established among PE's

Multiprocessor System:

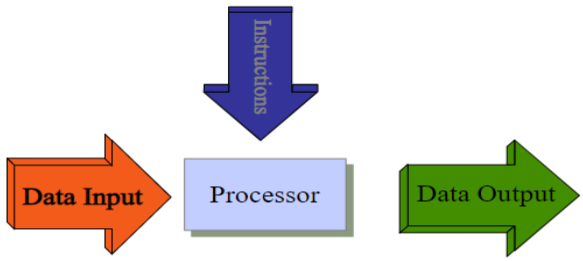
- This systems achieves asynchronous parallelism through a set of interactive processors with shared resources

Flynn's Classification

- Based on multiplicity of instruction streams and data streams observed by the CPU during program execution.

1) Single Instruction and Single Data stream (SISD)
2) Single Instruction and multiple Data stream (SIMD)
3) Multiple Instruction and Single Data stream (MISD)
4) Multiple Instruction and Multiple Data stream (MIMD)

SISD : A Conventional Computer



➔Speed is limited by the rate at which computer can transfer information internally.

Ex: PCs, Workstations

Single Instruction and Single Data stream (SISD)

- sequential execution of instructions is performed by one CPU containing a single processing element (PE)
- Therefore, SISD machines are conventional serial computers that process only one stream of instructions and one stream of data. Ex: Cray-1, CDC 6600, CDC 7600
-

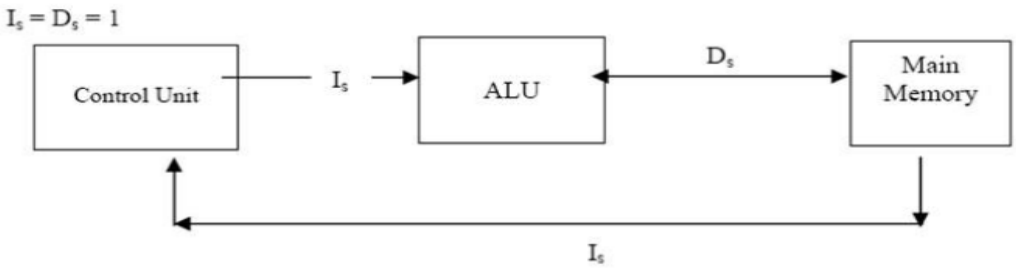
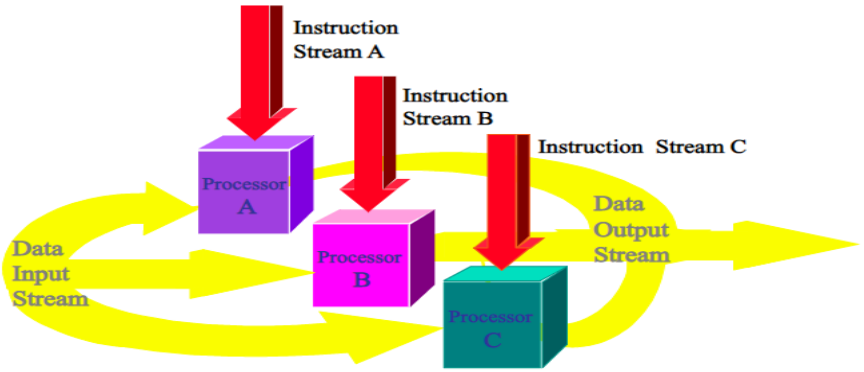


Figure 4: SISD Organisation

The MISD Architecture



➔More of an intellectual exercise than a practical configuration.
Few built, but commercially not available

Multiple Instruction and Single Data stream (MISD)

- multiple processing elements are organized under the control of multiple control units.
- Each control unit is handling one instruction stream and processed through its corresponding processing element.
- each processing element is processing only a single data stream at a time.
- Ex:C.mmp built by Carnegie-Mellon University.

$I_s > 1$
$D_s = 1$ All processing elements are interacting with the common shared memory for the organization of single data stream

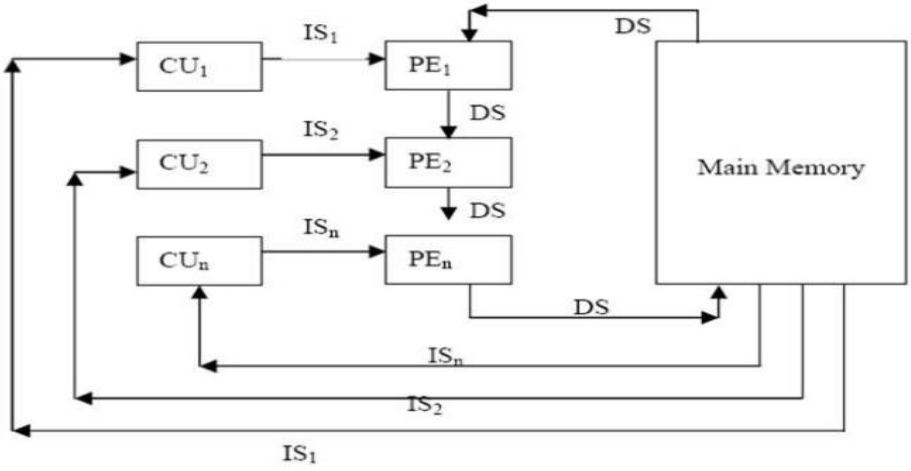


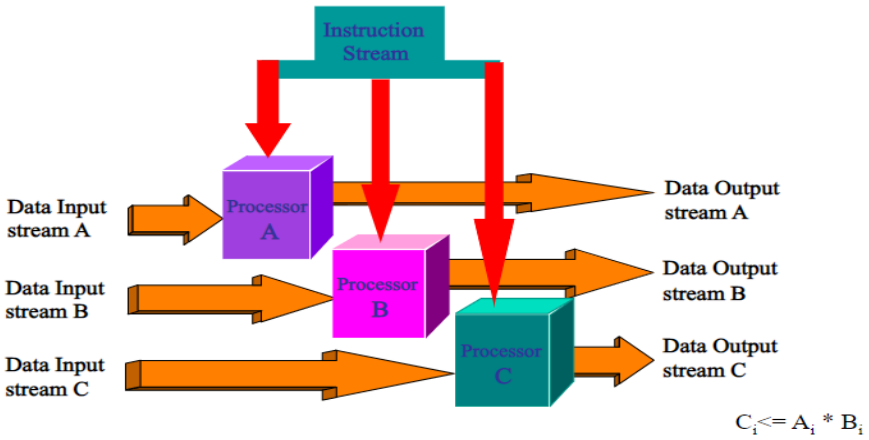
Figure 6: MISD Organisation

Advantages of MISD

– for the specialized applications like

- Real time computers need to be fault tolerant where several processors execute the same data for producing the redundant data.
- All these redundant data are compared as results which should be same otherwise faulty unit is replaced.
- Thus MISD machines can be applied to fault tolerant real time computers.

SIMD Architecture



Ex: CRAY machine vector processing, Intel MMX (multimedia support)

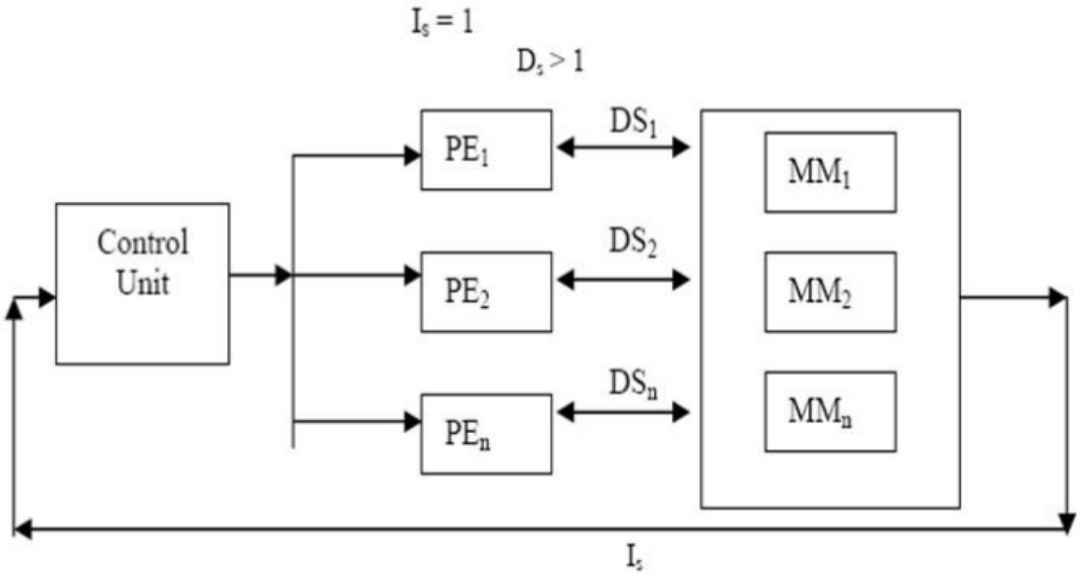


Figure 5: SIMD Organisation

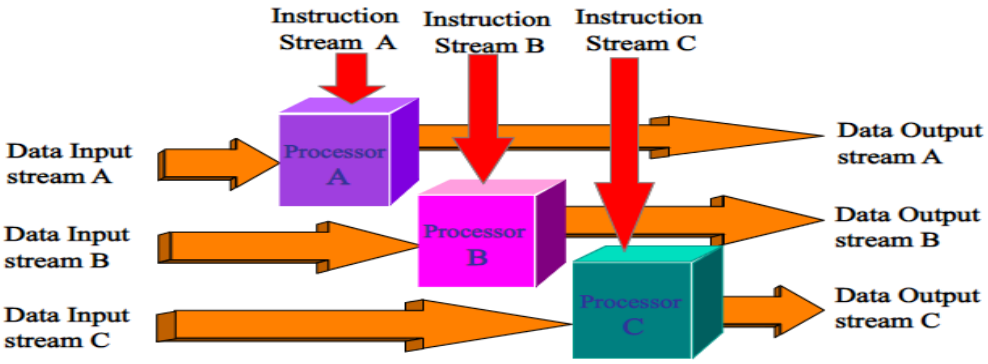
Single Instruction and multiple Data stream (SIMD)

- multiple processing elements work under the control of a single control unit.
- one instruction and multiple data stream.
- All the processing elements of this organization receive the same instruction broadcast from the CU.
- Main memory can also be divided into modules for generating multiple data streams.
- Every processor must be allowed to complete its instruction before the next instruction is taken for execution.
- The execution of instructions is synchronous

SIMD Processors

- Some of the earliest parallel computers such as the Illiac IV, MPP, DAP, CM-2 are belonged to this class of machines.
- Variants of this concept have found use in co-processing units such as the MMX units in Intel processors and IBM Cell processor.
- SIMD relies on the regular structure of computations (such as those in image processing).
- It is often necessary to selectively turn off operations on certain data items. For this reason, most SIMD programming architectures allow for an ``activity mask'', which determines if a processor should participate in a computation or not.

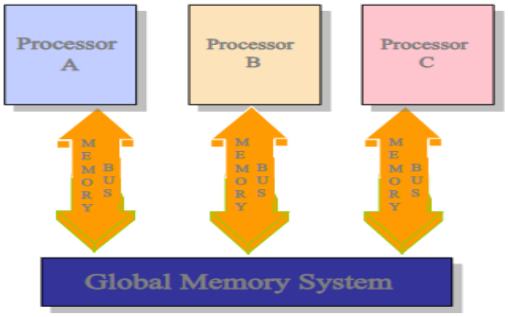
MIMD Architecture



Unlike SISD, MISD, MIMD computer works asynchronously.

- Shared memory (tightly coupled) MIMD
- Distributed memory (loosely coupled) MIMD

Shared Memory MIMD machine



Communication : Source PE writes data to GM & destination retrieves it

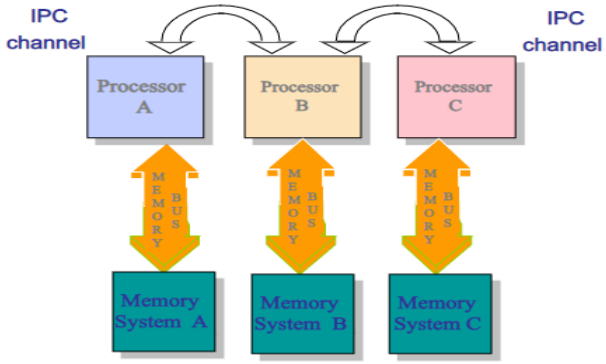
➔ Easy to build, conventional OSes of SISD can be easily ported

➔ Limitation : reliability & expandability. A memory component or any processor failure affects the whole system.

➔ Increase of processors leads to memory contention.

Ex. : Silicon graphics supercomputers....

Distributed Memory MIMD



- Communication : IPC (Inter-Process Communication) via High Speed Network.
- Network can be configured to ... Tree, Mesh, Cube, etc.
- Unlike Shared MIMD
  - ➔ easily/ readily expandable
  - ➔ Highly reliable (any CPU failure does not affect the whole system)

MIMD Processors

- In contrast to SIMD processors, MIMD processors can execute different programs on different processors.
- A variant of this, called single program multiple data streams (SPMD) executes the same program on different processors.
- It is easy to see that SPMD and MIMD are closely related in terms of programming flexibility and underlying architectural support.
- Examples of such platforms include current generation Sun Ultra Servers, SGI Origin Servers, multiprocessor PCs, workstation clusters.

Multiple Instruction and Multiple Data stream (MIMD)

- multiple processing elements and multiple control units are organized as in MISD.
- for handling multiple instruction streams, multiple control units are there and For handling multiple data streams, multiple processing elements are organized.
- The processors work on their own data with their own instructions.
- Tasks executed by different processors can start or finish at different times.

- in the real sense MIMD organization is said to be a **Parallel computer.**

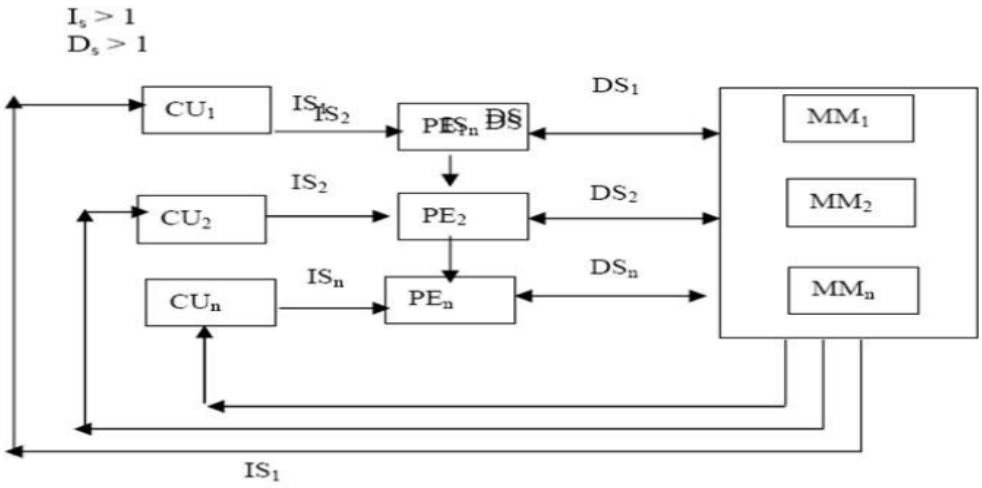


Figure 7: MIMD Organisation

- All multiprocessor systems fall under this classification.
- Examples :C.mmp, Cray-2, Cray X-MP, IBM 370/168 MP, Univac 1100/80, IBM 3081/3084.
- MIMD organization is the most popular for a parallel computer.
- In the real sense, parallel computers execute the instructions in MIMD mode

SIMD-MIMD Comparison

- SIMD computers require less hardware than MIMD computers (single control unit).
- However, since SIMD processors are specially designed, they tend to be expensive and have long design cycles.
- Not all applications are naturally suited to SIMD processors.
- In contrast, platforms supporting the SPMD paradigm can be built from inexpensive off-the-shelf components with relatively little effort in a short amount of time.

UNIT 2

Memory and Input-Output Subsystems – Hierarchical Memory Structure – Virtual Memory System – Memory Allocation and Management – Cache Memories and Management – Input-Output Subsystems.

Content:

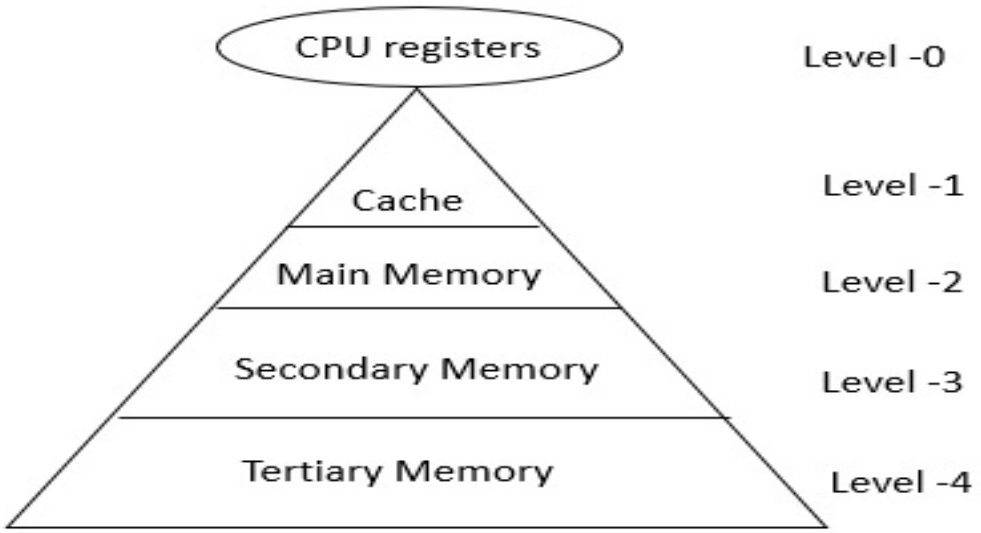
- Hierarchical Memory Structure
  - Memory Hierarchy
  - Optimization of memory hierarchy
  - Addressing schemes for main memory
- Virtual memory system
  - The concept of virtual memory
  - Paged Memory System
  - Segmented Memory system
  - Memory with paged segment
- Memory allocation and management
  - Classification of memory policies
  - Optimal load control
  - Memory management policies
- Cache Memory and management
  - Characteristics of cache
  - Cache memory management
- Input – Output Subsystems
  - Characteristics of I/O subsystems
  - Interrupt Mechanism and Special Hardware
  - I/O processors and I/O Channels

2.1 **Hierarchical Memory Structure.**

Memory Hierarchy:

- The Computer memory hierarchy looks like a pyramid structure which is used to describe the differences among memory types. It separates the computer storage based on hierarchy.
- In Memory Hierarchy the cost of memory, capacity is inversely proportional to speed. Here the devices are arranged in a manner Fast to slow, that is form register to Tertiary memory

Let us discuss each level in detail:



Level-0 − Registers

The registers are present inside the CPU. As they are present inside the CPU, they have least access time. Registers are most expensive and smallest in size generally in kilobytes. They are implemented by using Flip-Flops.

Level-1 − Cache

Cache memory is used to store the segments of a program that are frequently accessed by the processor. It is expensive and smaller in size generally in Megabytes and is implemented by using static RAM.

Level-2 − Primary or Main Memory

It directly communicates with the CPU and with auxiliary memory devices through an I/O processor. Main memory is less expensive than cache memory and larger in size generally in Gigabytes. This memory is implemented by using dynamic RAM.

Level-3 − Secondary storage

Secondary storage devices like Magnetic Disk are present at level 3. They are used as backup storage. They are cheaper than main memory and larger in size generally in a few TB.

Level-4 − Tertiary storage

Tertiary storage devices like magnetic tape are present at level 4. They are used to store removable files and are the cheapest and largest in size (1-20 TB).

Let us see the memory levels in terms of size, access time, bandwidth.

Level	**Register**	**Cache**	**Primary memory**	**Secondary memory**
Bandwidth	4k to 32k MB/sec	800 to 5k MB/sec	400 to 2k MB/sec	4 to 32 MB/sec
Size	Less than 1KB	Less than 4MB	Less than 2 GB	Greater than 2 GB
Access time	2 to 5nsec	3 to 10 nsec	80 to 400 nsec	5ms
Managed by	Compiler	Hardware	Operating system	OS or user

Optimization of Memory Hierarchy:

- Larger block size
  - -Reduces compulsory misses
  - -Increases capacity and conflict misses, increases miss penalty
- Larger total cache capacity to reduce miss rate
  - -Increases hit time, increases power consumption
- Higher associativity
  - -Reduces conflict misses
  - -Increases hit time, increases power consumption
- Higher number of cache levels
  - -Reduces overall memory access time
- Give priority to read misses over writes
  - -Reduces miss penalty
- Avoid address translation in cache indexing
  - -Reduces hit time

Advanced optimizations:

Metrics:

- Reducing the hit time
- Increase cache bandwidth
- Reducing miss penalty
- Reducing miss rate
- Reducing miss penalty or miss rate via parallelism

Addressing Schemes:

Addressing schemes is the mechanism employed for specified operands. The arrangements of opcodes and operands and their number within the instruction determine the form or format of instruction. a hypothetical machine which does not have any registers but very large cache and main memory can have the following addressing modes.

- Immediate Addressing
- Direct Addressing
- Indirect Addressing
- Stack Addressing

➢ Immediate Addressing

Under this scheme actual operand D is A the content of the operand field D=A. This addressing is used to initialized value of variable requiring no memory access for execution.

➢ Direct addressing

In this the content A of operand field specify effective address of the operand and the next statement below implies the data is stored in the memory location specified by effective address,

EA = A, D=(EA)

In this scheme only memory references are required. If the addressing scheme has n bits then address space addressable is 2n memory address.

➢ Indirect addressing:

In this addressing effective address EA and the contents of operand field are related by

EA=(A) , D=(EA)

The disadvanteg of this addressing scheme is that it requires 2 memory references to fetch the effective address from the memory and second for fetching the operand using EA. In this addressing the addressed space is determined by word length.

➢ Stack Addressing:

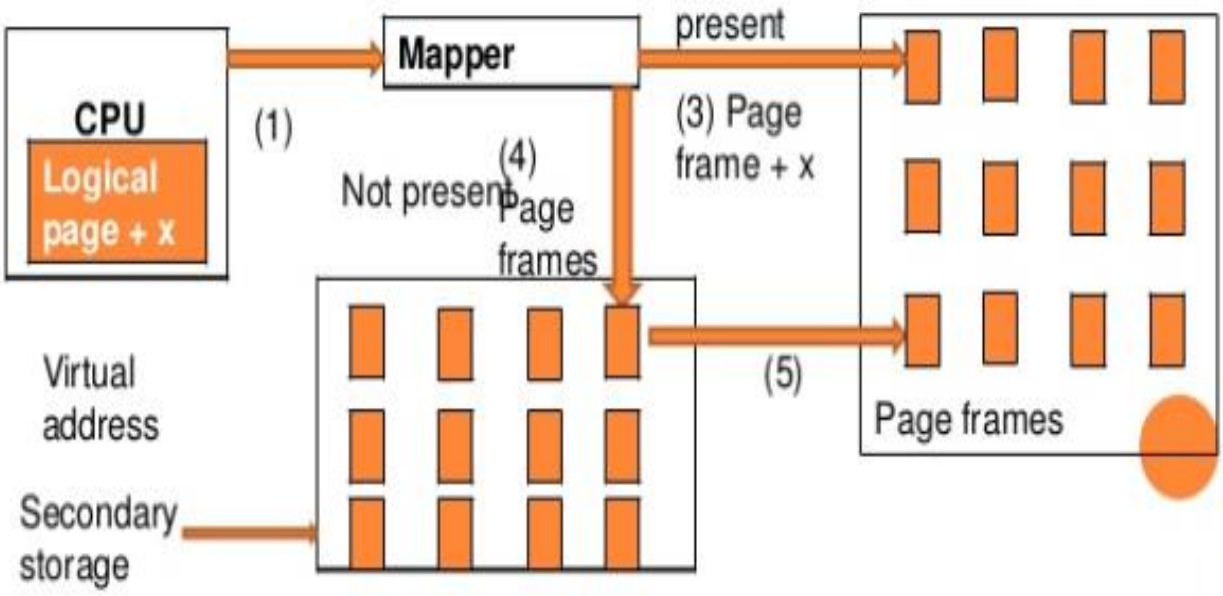
In this the address of operand is not specified explicitly . it is found on the top of the stack, in the absence of registers we can preserve the top of stack pointer in the memory.

All the four addressing schemes are using memory locations and not registers their disadvantage is that the addressable memory location is dependent on the instruction code length i.e., if the code is of n bits the addressable memory is 2n Locations. The other disadvantage is the multiple accesses of locations for fetching a single data.

2.2 **VIRTUAL MEMORY.**

- The term *virtual memory* refers to something which appears to be present but actually it is not.
- The virtual memory technique allows users to use more memory for a program than the real memory of a computer.
- So, virtual memory is the concept that gives the illusion to the user that they will have main memory equal to the capacity of secondary storage media.
- A programmer can write a program which requires more memory space than the capacity of the main memory. Such a program is executed by virtual memory technique.
- The program is stored in the secondary memory. The *memory management unit* (MMU) transfers the currently needed part of the program from the secondary memory to the main memory for execution.
-
- This to and fro movement of instructions and data (parts of a program) between the main memory and the secondary memory is called **Swapping.**

- Virtual address is the address used by the programmer and the set of such addresses is called the *address space* or *virtual memory*.
- An address in main memory is called a *location* or *physical address*. The set of such locations in main memory is called the *memory space* or *physical memory*.



Paged Memory

- Memory is divided into *pages*
  - Conceptually like a cache block
  - Typically 2K to 16K in size
  - A page can live anywhere in main memory, or on the disk
- Like cache, we need some way of knowing what page is where in memory
  - Page table instead of tags
  - Page table base register stores the address of the page table

Page Faults

- If the memory address desired cannot be found in the TLB, then a *page fault* has occurred, and the page containing that memory must be loaded from the disk into main memory.
- Page faults are costly because moving 2K – 16K (page size) can take a while.
- What if you have too many page faults?

Page Replacement Algorithms

- In a computer operating system that uses paging for virtual memory management, *page replacement algorithms* decide which memory pages to page out (swap out, write to disk) when a page of memory needs to be allocated.

- Paging happens when a page fault occurs and a free page cannot be used to satisfy the allocation, either because there are none, or because the number of free pages is lower than some threshold.

First - In - First - Out (FIFO)

- First-in-first-out is very easy to implement. The FIFO algorithm selects the page for replacement that has been in memory the longest time.

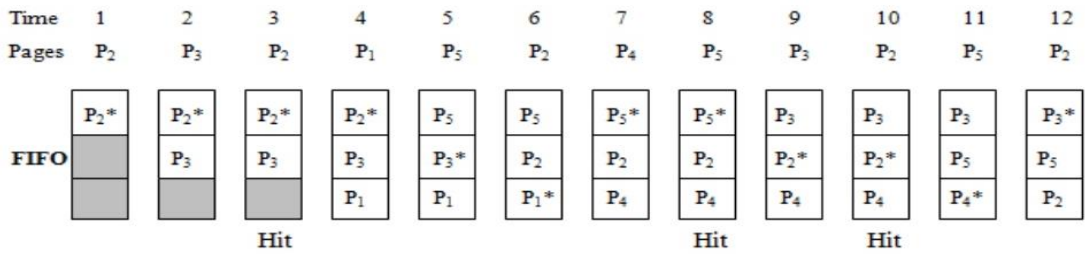
- When a new page must be loaded, the page recently brought in is removed. The page to be removed is easily determined because its identification number is at the top of the FIFO stack.

- The FIFO replacement policy has the advantage of being easy to implement.

- It has the disadvantage that under certain circumstances pages are removed and loaded from memory too frequently.

FIFO Algorithm

- Consider a paging system having capacity of 3 pages. The execution of a program requires references to five distinct pages P_1, P_2, P_3, P_4 and P_5. The pages are executed in the following sequence:
- P_2 P_3 P_2 P_1 P_5 P_2 P_4 P_5 P_3 P_2 P_5 P_2



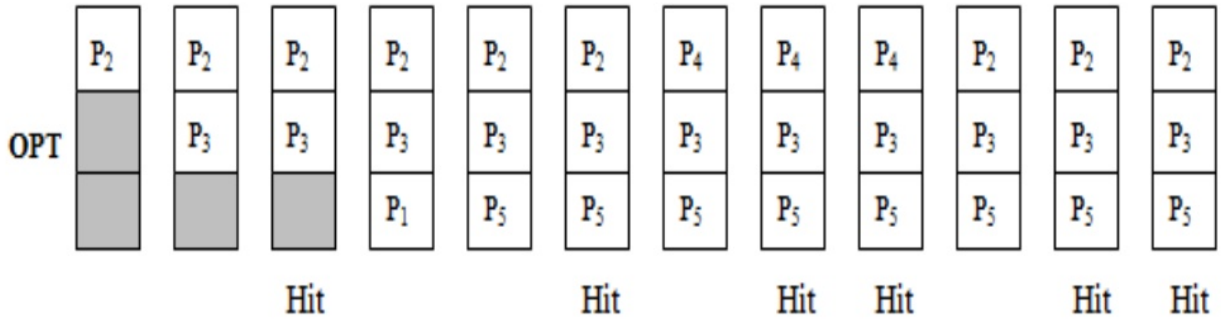
2.3 **Memory Allocation and Management.**

Optimal (OPT)

- The optimal policy selects that page for replacement for which the time to the next reference is longest.
- This algorithm results in fewest number of page faults. But, this algorithm is impossible to implement.
- At the time of page fault , the operating system has no way of knowing when each of the pages will be referenced next. However , it does serve as a standard against which to judge other algorithms.

Optimal (OPT)

- Consider a paging system having capacity of 3 pages. The execution of a program requires references to five distinct pages P_1, P_2, P_3, P_4 and P_5. The pages are executed in the following sequence:
- P_2 P_3 P_2 P_1 P_5 P_2 P_4 P_5 P_3 P_2 P_5 P_2



Memory management police:

Memory management keeps track of the status of each memory location, whether it is allocated or free. It allocates the memory dynamically to the programs at their request and frees it for reuse when it is no longer needed. Memory management meant to satisfy some requirements that we should keep in mind.

These Requirements of memory management are:

Relocation:

The available memory is generally shared among a number of processes in a multiprogramming system, so it is not possible to know in advance which other programs will be resident in main memory at the time of execution of his program. Swapping the active processes in and out of the main memory enables the operating system to have a larger pool of ready-to-execute process.

Protection:

There is always a danger when we have multiple programs at the same time as one program may write to the address space of another program. So every process must be protected against unwanted interference when other process tries to write in a process whether accidental or incidental. Between relocation and protection requirement a trade-off occurs as the satisfaction of relocation requirement increases the difficulty of satisfying the protection requirement.

Sharing:

A protection mechanism must have to allow several processes to access the same portion of main memory. Allowing each processes access to the same copy of the program rather than have their own separate copy has an advantage.

Logical organization:

Main memory is organized as linear or it can be a one-dimensional address space which consists of a sequence of bytes or words. Most of the programs can be organized into modules, some of those are unmodifiable (read-only, execute only) and some of those contain data that can be modified. To effectively deal with a user program, the operating system and computer hardware must support a basic module to provide the required protection and sharing.

Physical organization:

The structure of computer memory has two levels referred to as main memory and secondary memory. Main memory is relatively very fast and costly as compared to the secondary memory. Main memory is volatile. Thus secondary memory is provided for storage of data on a long-term basis while the main memory holds currently used programs.

2.4 **Chache Memories and Management:**

Cache Memories:

- Cache memory is nothing more than a small part of the storage system which is the solution to performance problems in the memory system. This is usually incorporated in processors and RAM, being a really small piece in terms of physical space. Its function is not complicated at all, but it is really important because it takes care of storing the most used data by the user and keep them close to the processor so that, it can have easy and quick access to them.
- Cache memory is taken as a special buffer of the memory that all computers have, it performs similar functions as the main memory. One of the most recognized caches are internet browsers which maintain temporary downloads made from the internet to have available information for internal system.

Charactreistics and Organization of cache memory:

- ✓ Cache memory is used to reduce the average time to access data from the Main memory.
- ✓ The cache is a smaller and faster memory which stores copies of the data from frequently used main memory locations.
- ✓ There are various different independent caches in a CPU, which stored instruction and data.

Levels of memory:

Level 1 or Register –
It is a type of memory in which data is stored and accepted that are immediately stored in CPU. Most commonly used register is accumulator, Program counter, address register etc.

Level 2 or Cache memory –
It is the fastest memory which has faster access time where data is temporarily stored for faster access.

Level 3 or Main Memory –
It is memory on which computer works currently it is small in size and once power is off data no longer stays in this memory

Level 4 or Secondary Memory –
It is external memory which is not fast as main memory but data stays permanently in this memory

Types of Cache :

Primary Cache –
A primary cache is always located on the processor chip. This cache is small and its access time is comparable to that of processor registers.

Secondary Cache –
Secondary cache is placed between the primary cache and the rest of the memory. It is referred to as the level 2 (L2) cache. Often, the Level 2 cache is also housed on the processor chip.

Organization:

- ✓ Allows you to quickly and organizationally have certain data, readings or files in common use, or data that have been recently added as downloads from the Internet.
- ✓ It presents an order system according to the importance of the file depending on the requirement of each operating system.
- ✓ It allows the processor to improve its performance and the results of the tasks by disposing its information as if it were a continuous use tool.
- ✓ Unwind internal processes of random storage memories, better known as RAM.
- ✓ Despite being an important part of the operating system, does not take up much space within the hardware or software.

Fetch and Mani memory:

The instruction cycle (also known as the fetch–decode–execute cycle, or simply the fetch-execute cycle) is the cycle that the central processing unit (CPU) follows from boot-up until the computer has shut down in order to process instructions. It is composed of three main stages: the fetch stage, the decode stage, and the execute stage.

Fetch Stage:

The next instruction is fetched from the memory address that is currently stored in the program counter and stored into the instruction register. At the end of the fetch operation, the PC points to the next instruction that will be read at the next cycle.

Steps:

The fetch step is the same for each instruction:

1. The CPU sends the contents of the PC to the MAR and sends a read command on the control bus
2. In response to the read command (with address equal to PC), the memory returns the data stored at the memory location indicated by the PC on the data bus
3. The CPU copies the data from the data bus into its MDR (also known as MBR; see section Role of components above)
4. A fraction of a second later, the CPU copies the data from the MDR to the instruction register for instruction decoding
5. The PC is incremented so that it points to the next instruction. This step prepares the CPU for the next cycle.

2.5 **INPUT OUTPUT SUBSYSTEM.**

Characteristic of Input output subsystem:

- Input-output (I/O) systems transfer information between computer main memory and the outside world. An I/O system is composed of I/O devices (peripherals), I/O control units, and software to carry out the I/O transaction(s) through a sequence of I/O operations.
- I/O devices can be classified as serial, i.e. able to transfer bit streams one bit at a time, or parallel. Parallel devices have a wider data bus and can therefore transfer data in words of one or more bytes.
- I/O is a concerted work of both hardware and software. The software which is executed to carry out an I/O transaction for a specific I/O device is called a device driver.

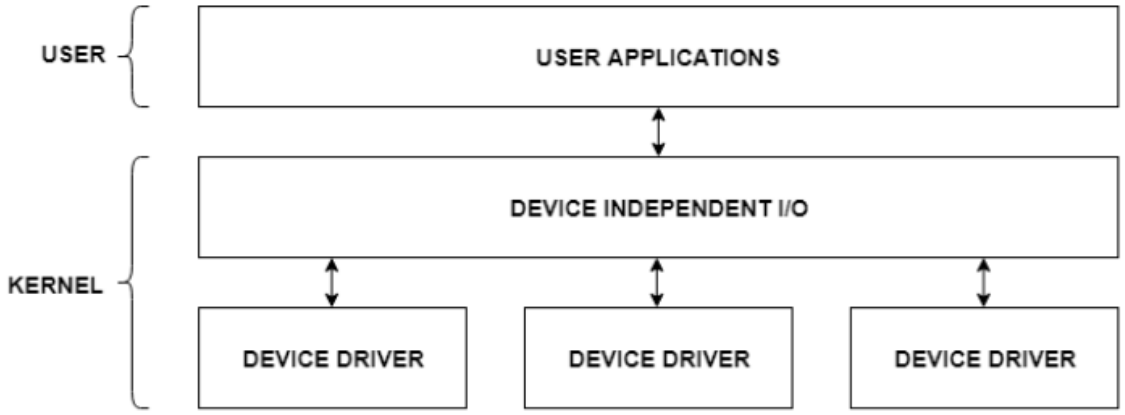
Interrupt Mechanisms and Special Hardware:

I/O devices are very important in the computer systems.They provide users the means of interacting with the system. So there is a separate I/O system devoted to handling the I/O devices.

I/O Application Interface:

The user applications can access all the I/O devices using the device drivers, which are device specific codes. The application layer sees a common interface for all the devices.

This is illustrated using the below image −



I/O Software:

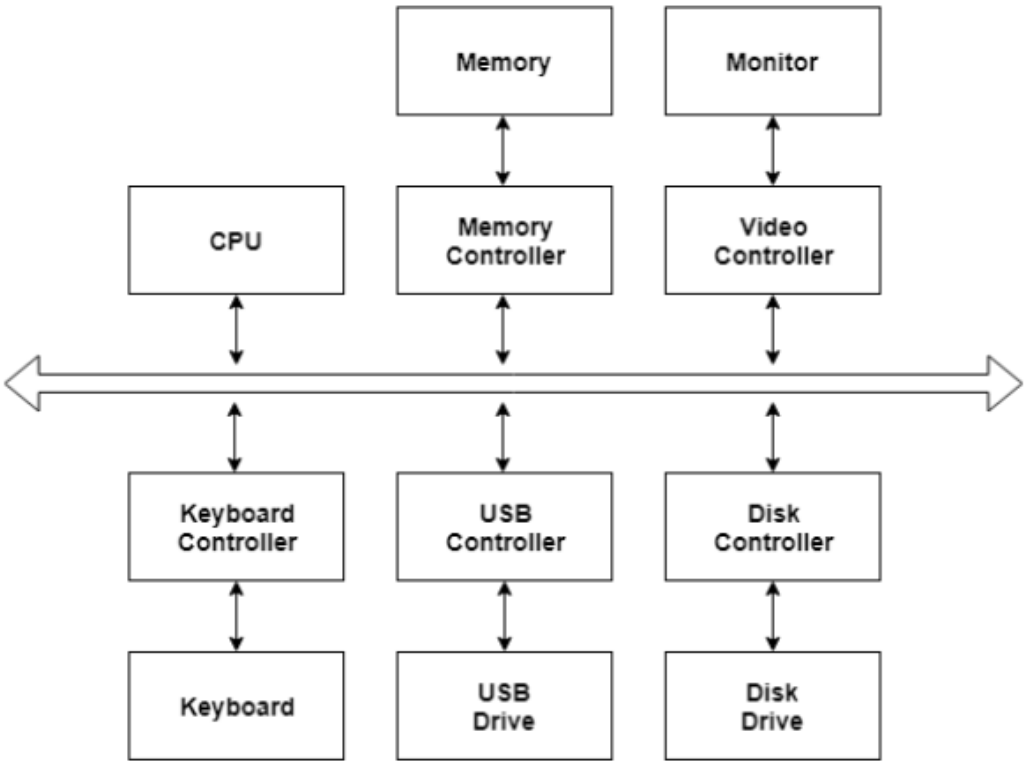
The I/O software contains the user level libraries and the kernel modules. The libraries provide the interface to the user program to perform input and output. The kernel modules provides the device drivers that interact with the device controllers.

The I/O software should be device independent so that the programs can be used for any I/O device without specifying it in advance. For example - A program that reads a file should be able the read the file on a hard disk, floppy disk, CD-ROM etc. without having to change the program each time.

I/O Hardware:

There are many I/O devices handled by the operating system such as mouse, keyboard, disk drive etc. There are different device drivers that can be connected to the operating system to handle a specific device. The device controller is an interface between the device and the device driver.

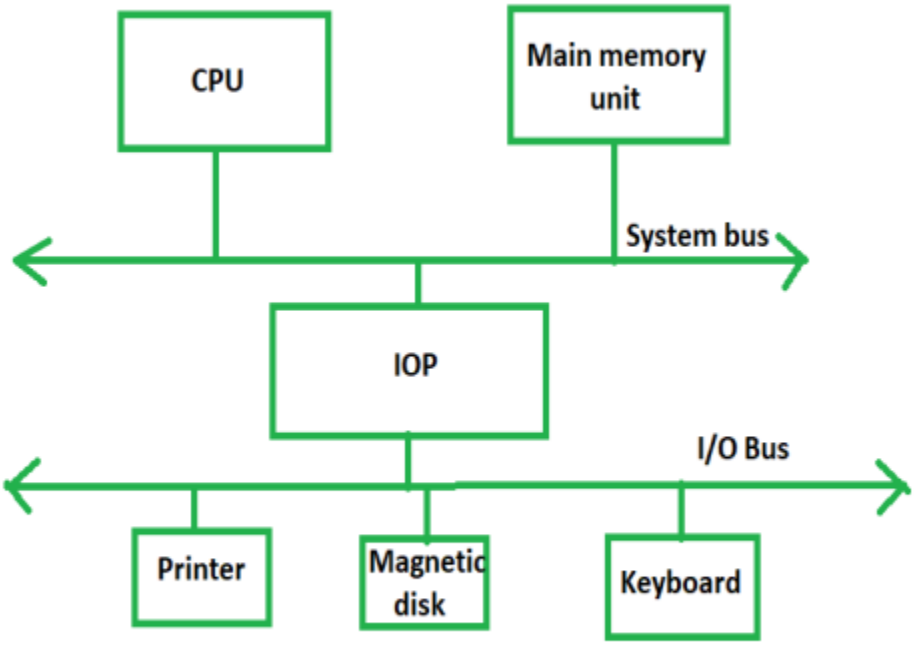
A diagram to represent this is −



I/o Processors or i/o Channels:

The DMA (Direct memory access) mode of data transfer reduces CPU's overhead in handling I/O operations. It also allows parallelism in CPU and I/O operations. Such parallelism is necessary to avoid wastage of valuable CPU time while handling I/O devices whose speeds are much slower as compared to CPU.

The concept of DMA operation can be extended to relieve the CPU further from getting involved with the execution of I/O operations. This gives rises to the development of special purpose processor called Input-Output Processor (IOP) or IO channel.



Input-Output Processor

The Input Output Processor is a specialized processor which loads and stores data into memory along with the execution of I/O instructions. It acts as an interface between system and devices. It involves a sequence of events to executing I/O operations and then store the results into the memory.

Advantages –

The I/O devices can directly access the main memory without the intervention

- by the processor in I/O processor based systems.
- It is used to address the problems that are arises in Direct memory access method.

****

UNIT 3

Principles of Pipelining and Vector Processing – Pipelining : An Overlapped Parallelism – Instruction and Arithmetic Pipelines – Principles of Designing Pipelined Processors – Vector Processing Requirements.

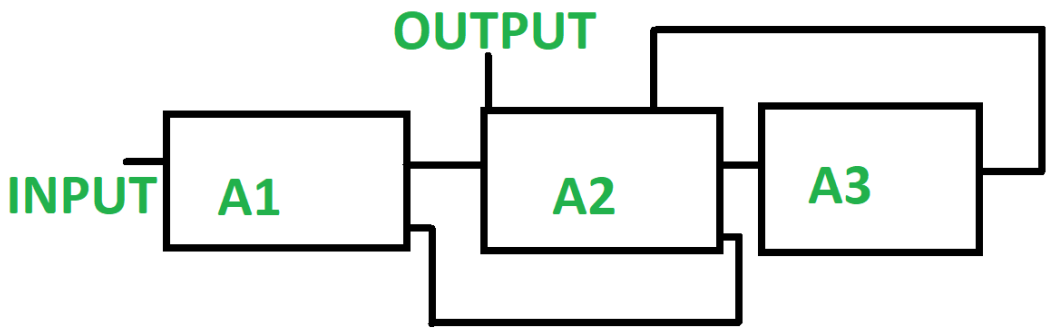
Pipelining:

- Pipelining is a parallel processing strategy in which an operation or a computation is partitioned into disjoint stages. Them stages must be executed in a particular order (could be a partial order) for the operation or computation to complete successfully.
- Each stage is implemented as a component which could be a hardware device or a software thread. When a stage completes, it becomes available to do other work.
- Parallelism results from the execution of a sequence of operations or computations so that at any given time several components of the sequence are under execution and each one of these is at a different stage of the pipeline.
- Pipelining is pervasive in today's machines. Processor control units and arithmetic units are typically pipelined. Also, programs take advantage of pipelined parallelism by partitioning computations into stages.

PRINCIPLES OF LINEAR PIPELINING:

Linear Pipeline :

- Linear pipeline is a pipeline in which a series of processors are connected together in a serial manner. In linear pipeline the data flows from the first block to the final block of processor.
- The processing of data is done in a linear and sequential manner. The input is supplied to the first block and we get the output from the last block till which the processing of data is being done.
- The linear pipelines can be further be divided into synchronous and asynchronous models.



Linear pipelining - is a technique of that decomposes any sequential process into small sub-processes, which are independent of each other so that each sub-process can be executed in a special dedicated segment and all these segments operates concurrently. Thus whole task is partitioned to independent tasks and these sub-task are executed by a segment.

Classification of Pipelining processors:

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. We can consider the pipelining concept as a collection of several segments of data processing programs which will be processing the data and sending the results to the next segment until the end of the processing is reached.

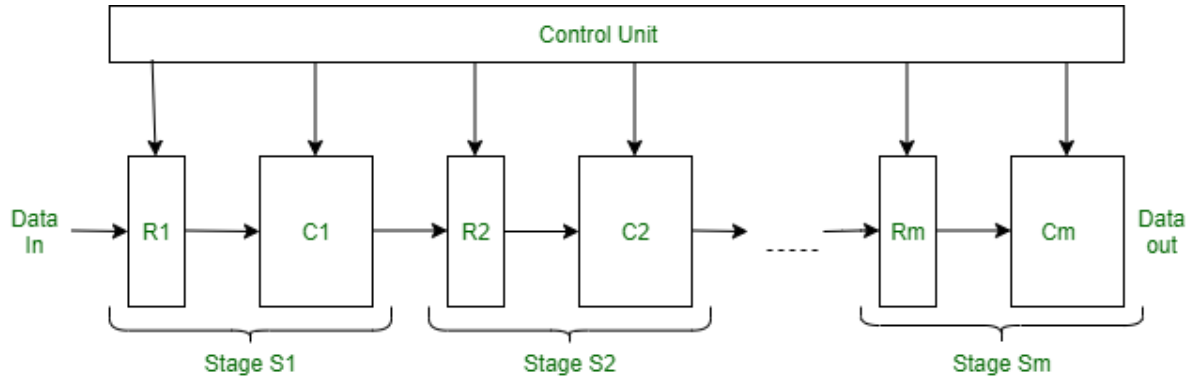
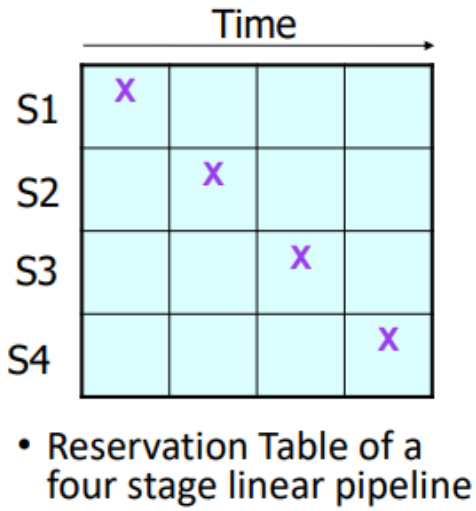


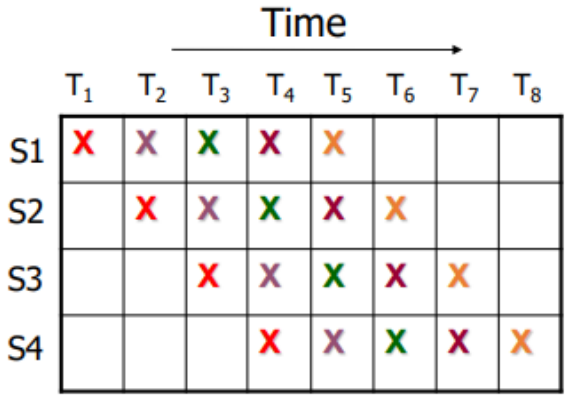
Figure - Structure of a Pipeline Processor

Reservation Table:

The utilization pattern of successive stages in a pipeline is specified by a Reservation Table



• Reservation Table of a four stage linear pipeline



5 tasks on 4 stages

Reservation table for function X

S1	X					X		X
S2		X		X				
S3			X		X		X	

Reservation table for function Y

S1	Y				Y	
S2			Y			
S3		Y		Y		Y

There are two reservation tables corresponding to a function X and a function Y, respectively. Each function evaluation is specified by one reservation table.

• A dynamic pipeline may be specified by more than one reservation table.

• Each reservation table displays the time-space flow of data through the pipeline for one function evaluation.

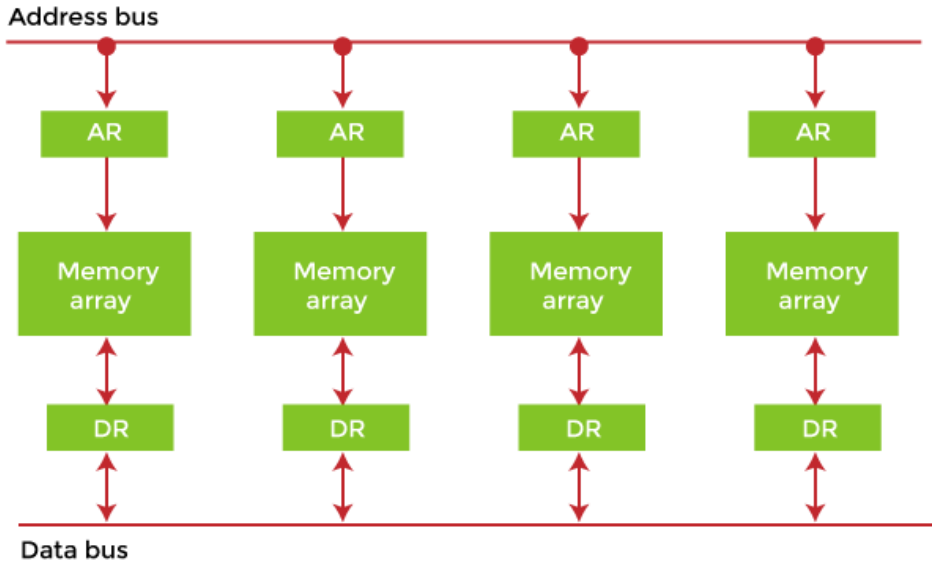
• Different functions follow different paths through the pipeline.

• The number of columns in a reservation table is called the Evaluation time of a given function.

Interleaved Memory Organization:

Interleaved memory is designed to compensate for the relatively slow speed of dynamic random-access memory (DRAM) or core memory by spreading memory addresses evenly across memory banks.

In this way, contiguous memory reads and writes use each memory bank, resulting in higher memory throughput due to reduced waiting for memory banks to become ready for the operations.



3.2 **Instruction and Arithmetic pipeline**

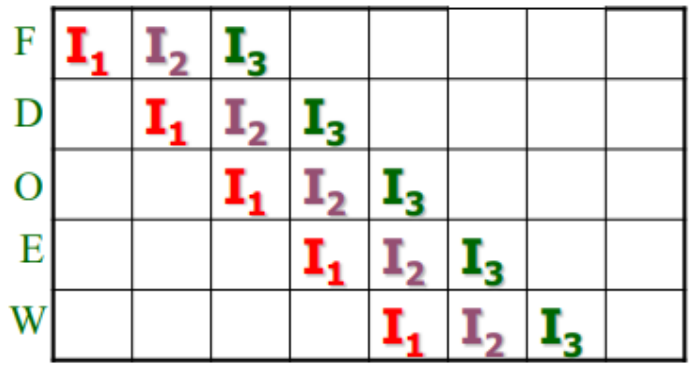
Instruction Pipeline:

Instruction pipelining is a technique for implementing instruction-level parallelism within a single processor. Pipelining attempts to keep every part of the processor busy with some instruction by dividing incoming instructions into a series of sequential steps (the eponymous "pipeline") performed by different processor units with different parts of instructions processed in parallel.

Linear Instruction Pipelines:

Assume the following instruction execution phases:

- Fetch (F)
- Decode (D)
- Operand Fetch (O)
- Execute (E)
- Write results (W)



Arithmetic Pipeline:

Pipeline arithmetic units are generally discovered in very large-speed computers. It can execute floating-point operations, multiplication of fixed-point numbers, and the same computations encountered in mathematical problems.

The inputs to the floating-point adder pipeline are two normalized floating-point binary numbers represented as −

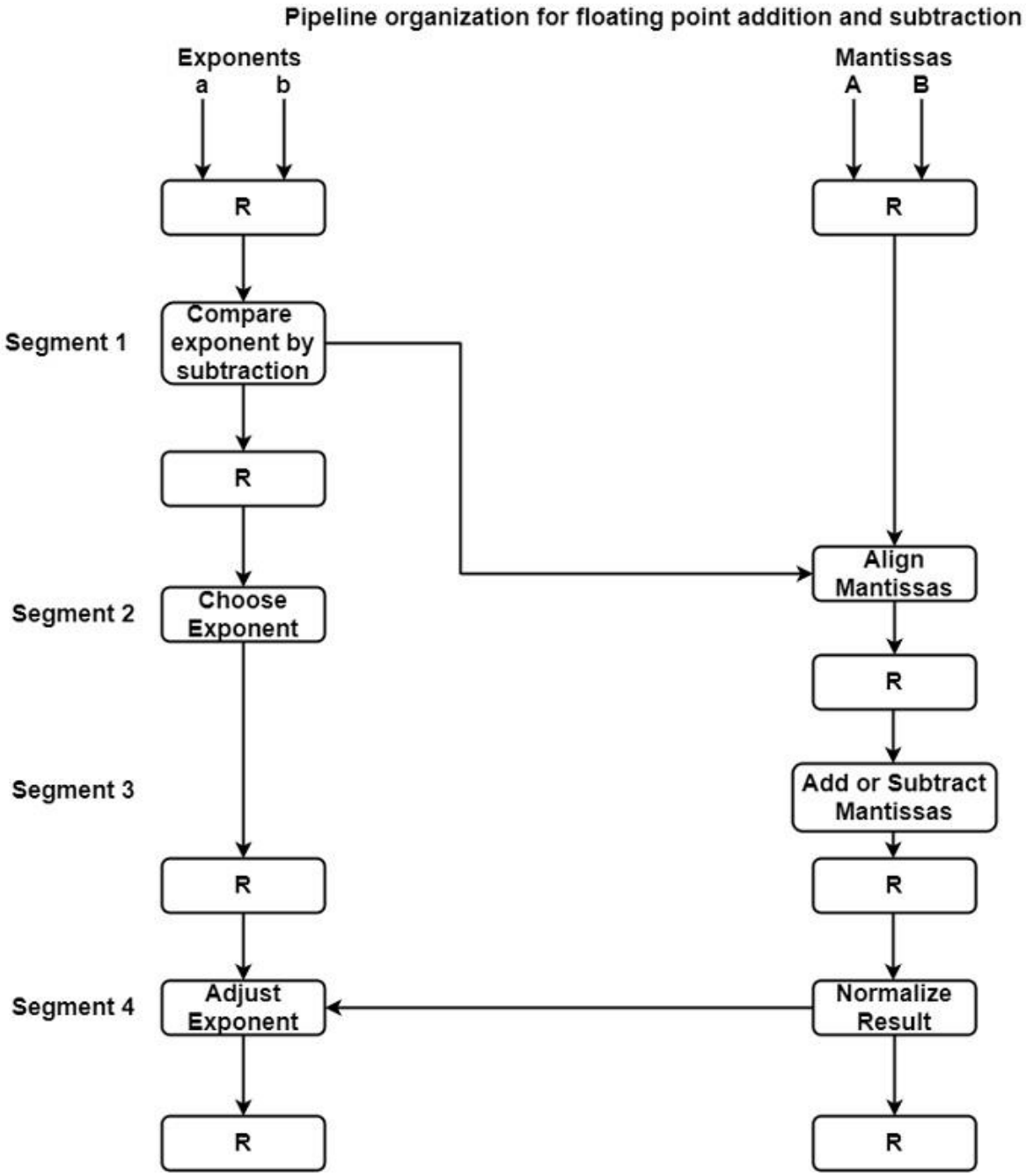
$X = A \times 2^a$

$Y = B \times 2^b$

Where A and B are two fractions that define the mantissa and a and b are the exponents. The floating-point addition and subtraction can be implemented in four segments, as a displayed figure. The registers labeled R is located between the segments to save intermediate outcomes. The sub-operations that are implemented in the four segments are −

- It can compare the exponents.
- It can align the mantissa.
- It can add or subtract the mantissa.
- It can normalize the result.

The following block diagram describes the sub-operations implemented in each segment of the pipeline.



Multifunction and Array Pipeline:

Multifunction Pipeline:

- There are two types of pipelines:
  - Static
  - Dynamic

- A static pipeline can perform only one function at a time, whereas a dynamic pipeline can perform more than one function at a time. A pipeline reservation table shows when stages of a pipeline are in use for a particular function.
- Static pipelines are **linear**, dynamic ones are **non-linear**
- A linear pipeline processor is a series of processing stages and memory access. A non linear pipelining (also called dynamic pipeline) can be configured to perform various functions at different times. In a dynamic pipeline there is also feed forward or feedback connection. Non-linear pipeline also allows very long instruction words.

Array Pipeline:

- Pipelining computations over a large array of cells has been an important feature of systolic arrays. To achieve even higher degrees of concurrency, it is desirable to have cells of a systolic array themselves be pipelined as well.

- The resulting two-level pipelined systolic array would enjoy in principle a k-fold increase in its throughput, where k is the ratio of the time to perform the entire cell computation over that to perform just one of its pipeline stages.

- This describes such a two-level pipelined systolic array that is capable of performing convolutions of any dimension. The designs take full advantages of the pipelining assumed to be available at each cell.

- Multi-stage pipelined arithmetic units built from discrete components have been used in most of high-performance computers. With the advent of VLSI, these pipelined units will surely be implemented in one or few chips. This paper shows for the first time how a large number of these pipelined chips can be efficiently combined to form a systolic array.

3.3 **Principles of Designing Pipelined processors.**

Instruction Prefetch:

The instructions in computer programs can be classified into 4 types:

- ✓ Arithmetic/Load Operations

- ✓ Store Type Instructions
- ✓ Branch Type Instructions
- ✓ Conditional Branch Type

Arithmetic/Load Operations:

-These operations require one or two operand fetches.

-The execution of different operations requires a different number of pipeline cycles

Store Type Instructions:

-It requires a memory access to store the data.

Branch Type Instructions:

-It corresponds to an unconditional jump.

Conditional Branch Type:

-Yes path requires the calculation of the new address

-No path proceeds to next sequential instruction.

Branch Handling:

- Arithmetic-load and store instructions do not alter the execution order of the program.
- Branch instructions and Interrupts cause some damaging effects on the performance of pipeline computers.

Handling Example – Interrupt System of Cray1;

Cray-1 System

•The interrupt system is built around an exchange package.

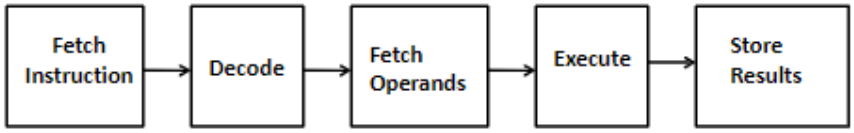
•When an interrupt occurs, the Cray-1 saves 8 scalar registers, 8 address registers, program counter and monitor flags.

•These are packed into 16 words and swapped with a block whose address is specified by a hardware exchange address register

•In general, the higher the percentage of branch type instructions in a program, the slower a program will run on a pipeline processor.

Effect of Branching on Pipeline Performance;

•Consider a linear pipeline of 5 stages



Data Buffering:

- A data buffer is a region of a physical memory storage used to temporarily store data while it is being moved from one place to another. Typically, the data is stored in a buffer as it is retrieved from an input device or just before it is sent to an output device.

- Buffers can be implemented in a fixed memory location in hardware or by using a virtual data buffer in software, pointing at a location in the physical memory.
- In all cases, the data stored in a data buffer are stored on a physical storage medium. A majority of buffers are implemented in software, which typically use the faster RAM to store temporary data, due to the much faster access time compared with hard disk drives.

- A buffer is a fast cycle added to a pipeline. It is typically, but not necessarily, non functional except for its buffering duties. There are three reasons for adding buffer cycles to a pipeline.

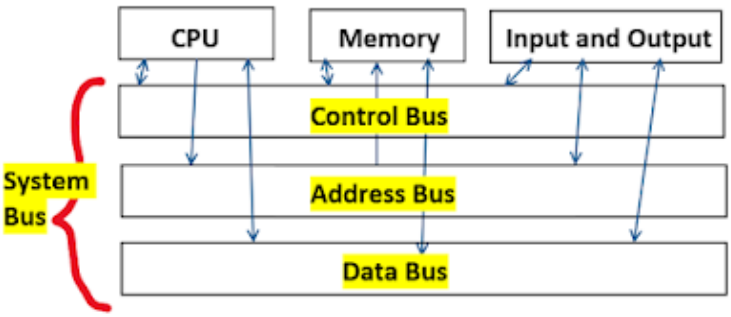
- The first is to enhance the throughput of a pipeline composed of variable throughput cycles. The second is to synchronize variable throughput behaviour with constant throughput behaviour.

Bussing Structure:

- A Bus is a collection of wires that connects several devices. Buses are used to send control signals and data between the processor and other components.
- This is to achieve a reasonable speed of operation. In computer system all the peripherals are connected to microprocessor through Bus.

Types of Bus structure:

- Address bus
- Data bus
- Control bus



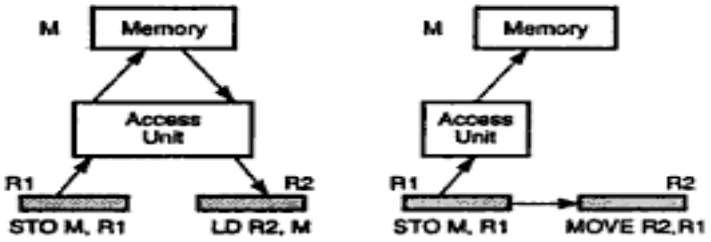
Types of Buses in Computer Architecture

Internal Forwarding:

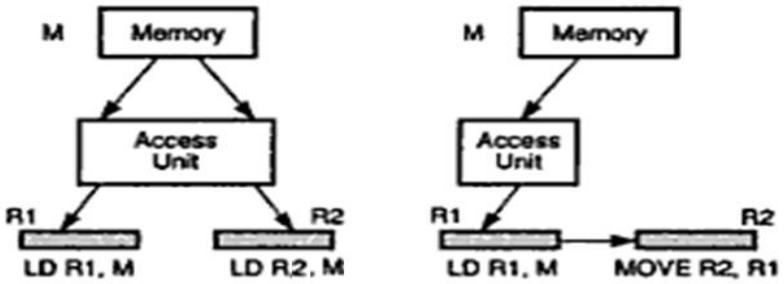
It is replacing unnecessary memory accesses by register-to-register transfers.

- **Memory access is slower** than register-to-register operations.
- **Performance** can be enhanced by **eliminating unnecessary memory accesses**
- This concept can be explored in 3 directions:

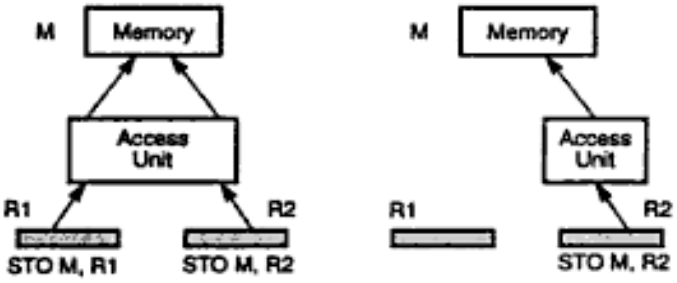
1. Store – Load Forwarding
2. Load – Load Forwarding
3. Store – Store Forwarding



(a) Store-load forwarding



(b) Load-load forwarding



(c) Store-store forwarding due to overlapping

Register Tagging:

It is the use of tagged registers for exploiting concurrent activities among multiple ALUs.

Example: IBM Model 91; Floating Point Execution Unit

The **floating point execution unit** consists of;

- Data registers
- Transfer paths
- Floating Point Adder Unit
- Multiply-Divide Unit
- Reservation stations
- Common Data Bus

Hazard Detection and Resolution:

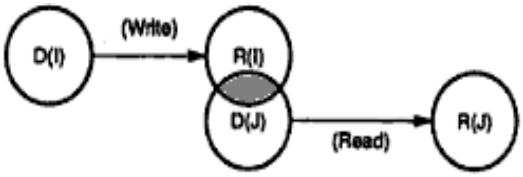
- Hazards are caused by resource usage conflicts among various instructions
- They are triggered by inter-instruction dependencies
- Hazards can be detected in fetch stage by comparing domain and range.
- Once detected, there are two methods:
    1. Generate a warning signal to prevent hazard
    2. Allow incoming instruction through pipe and distribute detection to all pipeline stages.

Terminologies:

- **Resource Objects:** set of working registers, memory locations and special flags
- **Data Objects:** Content of resource objects
- Each **Instruction** can be considered as a mapping from a set of data objects to a set of data objects.
- **Domain D(I) :** set of resource of objects whose data objects **may affect the execution of instruction I.**
- **Range R(I):** set of resource objects whose data objects may be **modified by the execution of instruction I**
- Instruction **reads** from its **domain** and **writes** in its **range**
- Consider execution of instructions I and J, and J appears immediately after I.
- There are 3 types of data dependent hazards:

1. **RAW (Read After Write)**

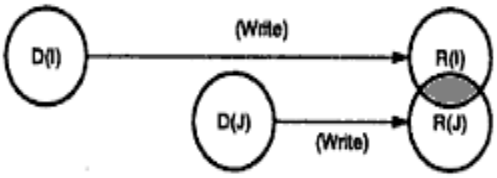
   The necessary condition for this hazard is $R(I) \cap D(J) \neq \phi$



(a) Read-after-Write (RAW) hazard

2. **WAW(Write After Write)**

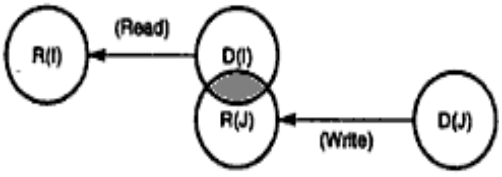
   The necessary condition is $R(I) \cap R(J) \neq \phi$



(b) Write-after-Write (WAW) hazard

3. **WAR (Write After Write)**

   The necessary condition is $D(I) \cap R(J) \neq \phi$



(c) Write-after-Read (WAR) hazard

Job Sequencing and Collision Prevention:

It for the Design of Static Pipeline

Consider reservation table given below at

t=0 0 Sa A Sb Sc 1 2 3 4 5 A A A

Consider reservation table given below at t=0

	0	1	2	3	4	5
Sa	**A**					**A**
Sb		**A**		**A**		
Sc			**A**		**A**	

Consider next initiation made at

t=1 0 1 2 3 4 5 6 Sa A 1 A 2 Sb A 1 A 2 Sc A 1 A 2 7

• The second initiation easily fits in the reservation table

Consider next initiation made at t=1

	0	1	2	3	4	5	6	7
Sa	A_1	A_2				A_1	A_2	
Sb		A_1	A_2	A_1	A_2			
Sc			A_1	A_2	A_1	A_2		

The second initiation easily fits in the reservation table

Now consider the case when first initiation is made at t = 0 and second at

t = 2. 0 1 2 Sa A 1 A 2 Sb Sc A 1 A 2 A 1 3 4 A 1 A A 2 2 A 2 5 6 7 A 1 A 2 A 1 A 2

• Here both markings A 1 and A 2 falls in the same stage time units and is called collision and it must be avoided.

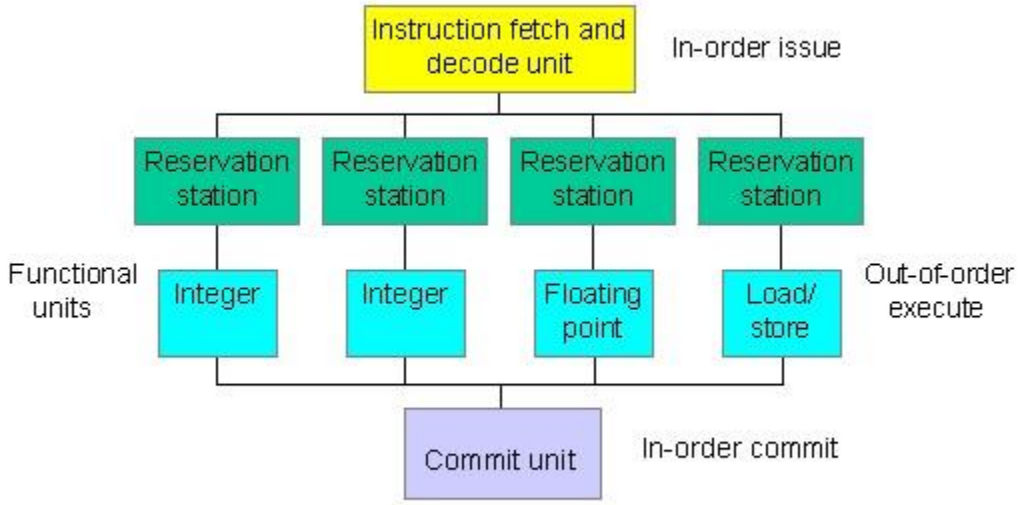
Now consider the case when first initiation is made at t = 0 and second at t = 2.

	0	1	2	3	4	5	6	7
Sa	A_1		A_2			A_1		A_2
Sb		A_1	A_2	A_1A_2		A_2		
Sc			A_1	A_2	A_1A_2		A_2	

Here both markings A1 and A2 falls in the same stage time units and is called **collision** and it must be avoided

Dynamic pipelines:

- Dynamic pipeline can perform more than one function at a time. A pipeline reservation table shows when stages of a pipeline are in use for a particular function.
- Dynamic pipelines have the capability to schedule around stalls. A dynamic pipeline is divided into three units: the instruction fetch and decode unit, five to ten execute or functional units, and a commit unit. Each execute unit has reservation stations, which act as buffers and hold the operands and operations.



- While the functional units have the freedom to execute out of order, the instruction fetch/decode and commit units must operate in-order to maintain simple pipeline behaviour.
- When the instruction is executed and the result is calculated, the commit unit decides when it is safe to store the result.
- If a stall occurs, the processor can schedule other instructions to be executed until the stall is resolved.
- This, coupled with the efficiency of multiple units executing instructions simultaneously, makes a dynamic pipeline an attractive alternative.

Re-configurability:

- Systems with pipeline processing capabilities, processors whose computational subsystems are divided into several distinct stages, each of which may be working with an independent set of data at the same instant of time, are one attractive solution to the demand for faster and more cost effective computers.
- In addition, several of the existing pipelined machines have facilities for a limited amount of restructuring of available resources to provide a system which is more suitable for the efficient execution of a particular problem.
- The amount of restructuring, or reconfiguration, varies from system to system.
- The type of reconfiguration, either a reconfiguration of an actual functional unit or a reconfiguration of the information flow between several functional units, varies as well.
- The reconfigurable aspects of the architecture of several pipelined machines are presented. Performance gains when restructuring is allowed are also examined. Finally, there is a discussion of some possible future reconfigurable pipeline systems.

3.4 Vector Processing:

Vector processing is a central processing unit that can perform the complete vector input in individual instruction. It is a complete unit of hardware resources that implements a sequential set of similar data elements in the memory using individual instruction.

Features of Vector Processing:

There are various features of Vector Processing which are as follows;

- A vector is a structured set of elements. The elements in a vector are scalar quantities. A vector operand includes an ordered set of n elements, where n is known as the length of the vector.
- Each clock period processes two successive pairs of elements. During one single clock period, the dual vector pipes and the dual sets of vector functional units allow the processing of two pairs of elements.

  As the completion of each pair of operations takes place, the results are delivered to appropriate elements of the result register. The operation continues just before the various elements processed are similar to the count particularized by the vector length register.
- In parallel vector processing, more than two results are generated per clock cycle. The parallel vector operations are automatically started under the following two circumstances −
  - When successive vector instructions facilitate different functional units and multiple vector registers.
  - When successive vector instructions use the resulting flow from one vector register as the operand of another operation utilizing a different functional unit. This phase is known as chaining.
- A vector processor implements better with higher vectors because of the foundation delay in a pipeline.
- Vector processing decrease the overhead related to maintenance of the loop-control variables which creates it more efficient than scalar processing.

Vector Processor Characteristics:

- A **vector processor** is a CPU in a computer with parallel processors and the capability for vector processing.
- The main characteristic of a vector processor is that it makes use of the parallel processing capability of the processor where two or more processors operate concurrently.
- This makes it possible for the processors to perform multiple tasks simultaneously or for the task to be split into different subtasks handled by different processors and combined to get the result.

- The vector processor considers all of the elements of the vector as one single element as it traverses through the vector in a single loop.
- Computers with vector processors find many uses that involve computation of massive amounts of data, such as image processing, artificial intelligence, mapping the human genome, space simulations, seismic data, and hurricane predictions.

Multiple Vector Task Dispatching:

- Multiple Vector Task dispatching for an asymmetric or symmetric pipeline processor system is provided where all the processors are dispatched from a single task dispatching queue.
- The workload, i.e. tasks, of the multiprocessor system is distributed to the available processors. Each processor includes a task dispatcher and a signal dispatcher.
- The signal dispatcher runs in a processor whenever a task dispatching element (TDE) is put on the task dispatching queue (TDQ) as a result of the task running in the processor.
- The signal dispatcher examines the TDEs enqueued on the TDQ and determines if any task dispatcher should be invoked, i.e. if any processor is running a lower priority task a task switch should occur. If so, it signals the selected processor to invoke its task dispatcher.
- After completing the task switch, the selected processor must invoke its signal dispatcher to determine if the task it had been performing should now be performed on some other processor in the vector pipeline processor system.

Pipelined Vector Processing Method:

- The term "vector pipeline" was used in the 1970's to describe vector processing at a time when a single vector instruction might (for example) compute the sum of two vectors of floating point numbers using a single pipelined floating point arithmetic unit.
- Pairs of floating point numbers would be brought from memory into the pipelined arithmetic unit to be added together, and once the pipeline filled, you'd get one sum out per cycle and these sums would be streamed back into memory.
- Vector pipelining might mean vectorizing in such a way that vector instructions can be pipelined. Is this a new or coming feature in modern architectures

****

UNIT 4

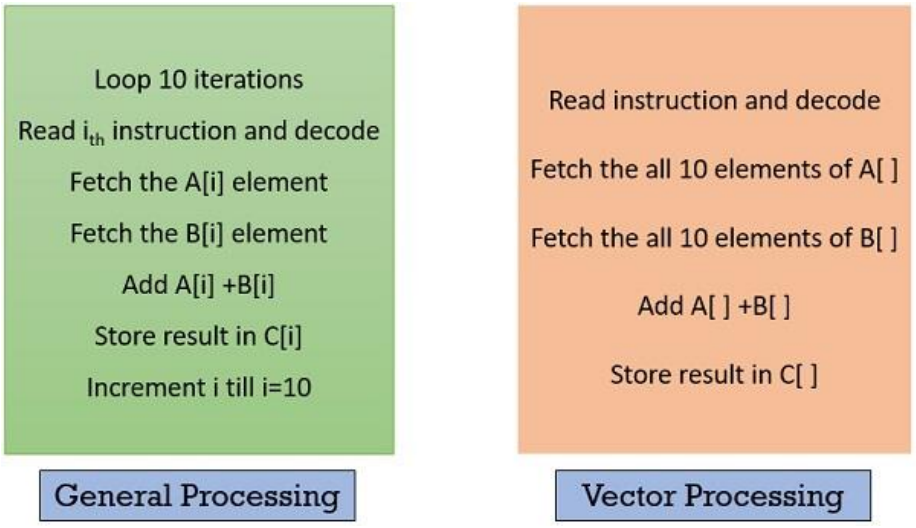
Vectorization and Optimization methods – Parallel Languages for Vector Processing – Design of Vectorizing Compiler – Optimization of Vector Functions – SIMD Array Processors – SIMD Interconnection Networks

VECTORIZATION AND OPTIMIZATION METHODS.

- Vectorization is a special case of **Single Instructions Multiple Data** (SIMD) to denote a single instruction stream capable of operating on multiple data elements in parallel. We can think of vectorization as the unrolling of loops accompanied with SIMD instructions.
- Vectorization is the process of converting an algorithm that performs scalar operations (typically one operation at the time) to vector operations where a single operation can refer to many simultaneous operations.

4.1 **Language Feature in Vector Processing:**

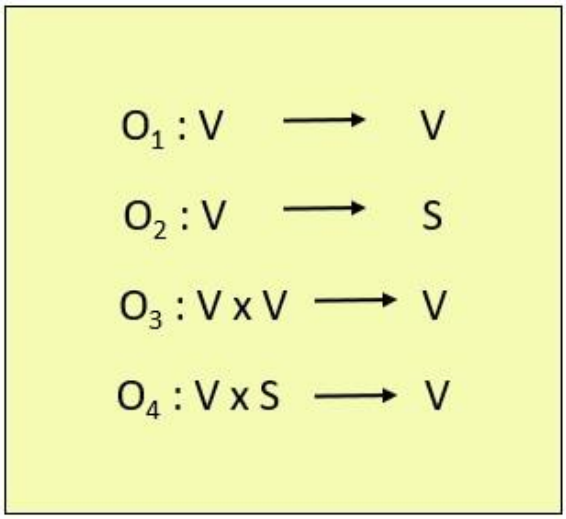
- Vector processing performs the arithmetic operation on the large array of integers or floating-point number. Vector processing operates on all the elements of the array in parallel providing each pass is independent of the other.
- Vector processing avoids the overhead of the loop control mechanism that occurs in general-purpose computers.
- Vector processing operates on the entire array in just one operation i.e. it operates on elements of the array in parallel. But, vector processing is possible only if the operations performed in parallel are independent.
- Look at the figure below, and compare the vector processing with the general computer processing, you will notice the difference. Below, instructions in both the blocks are set to add two arrays and store the result in the third array. Vector processing adds both the array in parallel by avoiding the use of the loop.



Characteristics of Vector Processing:

- Each element of the vector operand is a **scalar quantity** which can either be an integer, floating-point number, logical value or a character. Below we have classified the vector instructions in four types.

- Here, V is representing the vector operands and S represents the scalar operands. In the figure below, O1 and O2 are the unary operations and O3 and O4 are the binary operations.



- Most of the vector instructions are **pipelined** as vector instruction performs the same operation on the different data sets repeatedly. Now, the pipelining has start-up delay, so longer vectors would perform better here.

- The pipelined vector processors can be classified into two types based on from where the operand is being **fetched** for vector processing. The two architectural classifications are Memory-to-Memory and Register-to-Register.

- In **Memory-to-Memory** vector processor the operands for instruction, the intermediate result and the final result all these are retrieved from the **main memory.**

Design of Vectorization Compilers:

- ✓ Single instruction, multiple data (SIMD) computing can be achieved using parallel computers. It can also help you improve code performance while decreasing its size. Vectorization is a technique that really can take advantage of SIMD hardware.

- ✓ This can either be implemented by the programmer or can be carried out by the compiler.As always there are both pros and cons that you should carefully consider before using this optimization yourself.

- ✓ Many compilers come with vectorization optimizations.

- ✓ It can help you increase your code's performance without increasing its size. This is the idea behind vectorizing loops in your code. Vectorization can use fewer lines of code than loop unrolling while achieving the same speeds. Everything has a cost, though, and the cost here is money and power.

- ✓ Compiler optimization is extremely useful, but only if you have one that can work intelligently with your code. TASKING develops compilers, and other tools like static analyzers and standalone debuggers, specifically for the automotive industry. That way you can know that your code is optimized for your unique application.

Optimization of Vector Function:

- Traditionally, loop-based vectorization is the main focus of performance optimization through vectors. Loop-based vectorization can extract data parallelism from nested loops with large iteration spaces by loop transformations and by dependency analysis.

- There are many transformations that may need to be performed to allow the compiler to successfully exploit SIMD parallelism. These transformations include loop interchange, loop peeling, etc. Combining these transformations may generate better results than using one of them exclusively.

- The loop based vectorization techniques can be used to expose parallelism that can be exploited with a basic block vectorization algorithm, so the techniques can be complementary to each other.

- Differentiates between vectorization and concurrentization and vector optimization. Vectorization (concurrentization) is the process of translating serial DO loops into vector code for a vector computer (concurrent code for a multiprocessor computer).

- Vector optimization uses vectorization (concurrentization) but adds some intelligence to decide which loop (s) to run in parallel. Vector optimization tools, such as loop interchanging and generation of multiple versions of a loop, are introduced and some decision mechanisms are discussed.

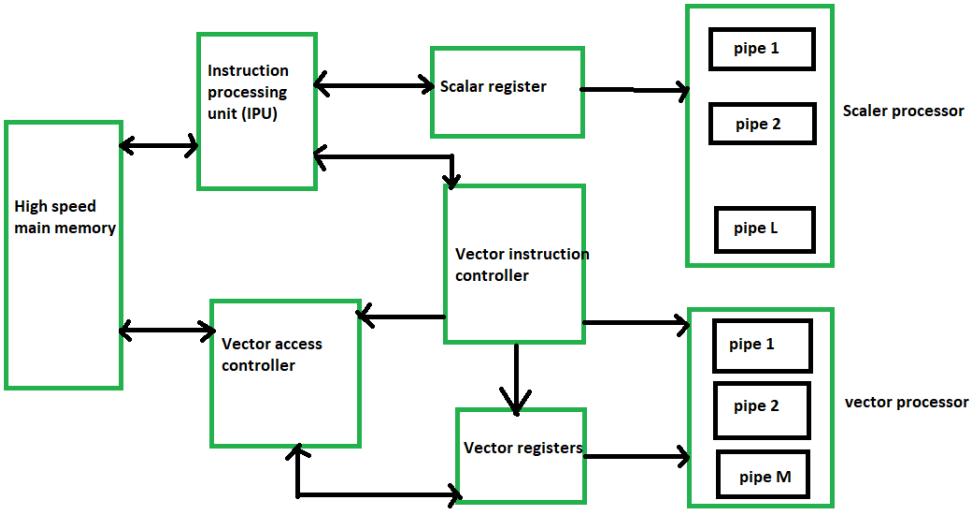
Performance Evaluation of Vector Pipeline Processor:

• A vector processor is an ensemble of hardware resources, including vector registers, functional pipelines, processing elements and register counters for performing register operations.

• Vector processing occurs when arithmetic or logical operations are applied to vectors. It is distinguished from scalar processing which operates on one or one pair of data. The conversion from scalar code to vector code is called vectorization.

• Both pipelined processors and SIMD computers can perform vector operations.

• Vector processing reduces software overhead incurred in the maintenance of looping control, reduces memory access conflicts and above all matches nicely with pipelining and segmentation concept to generate one result per each clock cycle continuously



4.2 **SIMD Array processors.**

- Array Processor performs computations on large array of data. These are two types of Array Processors: Attached Array Processor, and SIMD Array Processor.

- **SIMD array processor:** This is computer with multiple process unit operating in parallel Both types of array processors, manipulate vectors but their internal organization is different

- SIMD is a computer with multiple processing units operating in parallel.

- The processing units are synchronized to perform the same operation under the control of a common control unit. Thus providing a single instruction stream, multiple data stream (SIMD) organization. As shown in figure, SIMD contains a set of identical processing elements (PES) each having a local memory M.

- Each PE includes:
    - ALU
    - Floating point arithmetic unit
    - Working registers

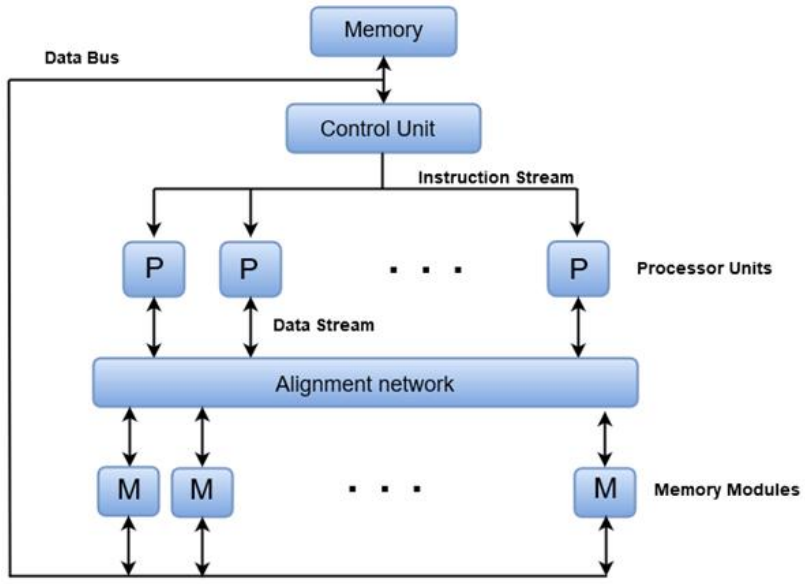
- Master control unit controls the operation in the PEs. The function of master control unit is to decode the instruction and determine how the instruction to be executed. If the instruction is scalar or program control instruction then it is directly executed within the master control unit.

- Main memory is used for storage of the program while each PE uses operands stored in its local memory.

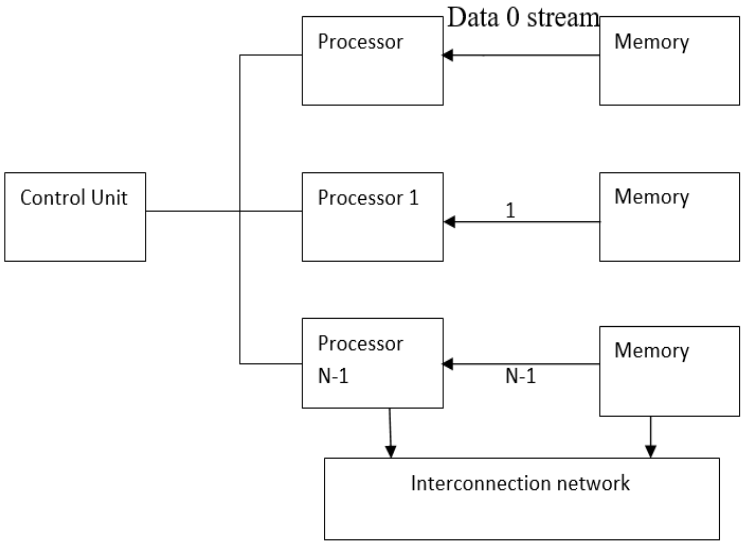
SIMD Computer Organization:

- **SIMD** stands for 'Single Instruction and Multiple Data Stream'. It 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.
- SIMD is mainly dedicated to array processing machines. However, vector processors can also be seen as a part of this group.

SIMD:



- A SIMD system is a multiprocessor machine, capable of executing the same instruction on all the CPUs but operating on the different data stream.



Masking and Data Routing Mechanisms:

Masking vector is used to control the status of PEs during the execution of a vector instruction. Only enabled PEs participate in the execution of a vector instruction.

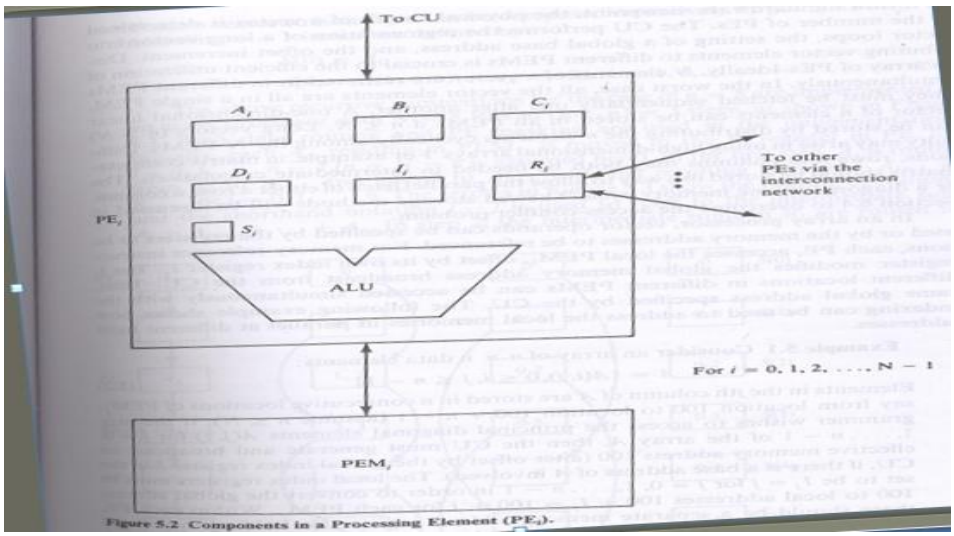


Figure 5.2 Components in a Processing Element (PE_i).

Components in a PE:

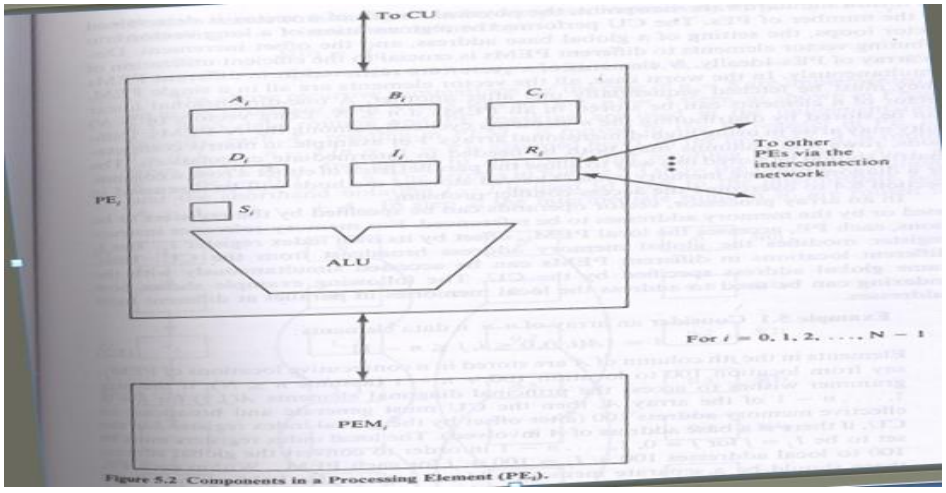


Figure 5.2 Components in a Processing Element (PE_i).

- Here each PE is a processor with its own memory PEM; a set of working registers and flags , namely A, B, C and S; an ALU; a local index register I; an address register D and a data routing register R.
- The R of each PE is connected to the R of other PE via the interconnection n/w.
- When data transfer among PEs occur contents of R registers are being transferred.
- The D is used to hold the m-bit address of the PE.
- Some array processors use 2 routing registers - for i/p and o/p.
- PE is either in active or inactive mode during instruction cycle.

- If a PE is active, it executes the instruction broadcast to it by the CU otherwise it will not.
- Masking schemes are used to specify the status flag S of PE.
- S=1 indicate active PE and 0 for inactive PE.
- In CU there is a global index register I and a masking register M. The M register has N bits. The ith bit of M is denoted as Mi.
- The collection of Si flags for i=0,1,2,---N-1 forms a status register S for all the PEs.
- Bit patterns in registers M and S are exchangeable upon the control of CU when masking is to be set.

Inter –PE communications:

Network design is the fundamental decision in determining appropriate architecture of an interconnection network for an SIMD machine.

Operation mode:

- 2 types of communication: synchronous and asynchronous.
- Synchronous communication needed for establishing communication paths synchronously for either data manipulating function or for data instruction broadcast.
- Asynchronous communication is needed for multiprocessing in which connection requests are issued dynamically.

Control strategy:

- A typical interconnection n/w consists of a no. of switching elements and interconnecting links.
- Interconnection functions are realized by properly setting control of the switching elements.

Switching methodology:

- circuit switching and packet switching.
- circuit switching: physical path is established between source and destination.
- packet switching: data is put in a packet and routed through the interconnection n/w without establishing a physical connection path.

N/w topology:

- N/w is depicted by a graph.
- nodes represent switching points and edges represent communication links.
- Topologies grouped into 2 categories:
- static and dynamic.

Static topology:

- links between 2 processors are passive and dedicated buses cannot be reconfigured for direct connections to other processors. Links in the dynamic category can be reconfigured by setting the n/ws active switching elements.

5.3 **SIMD interconnection Networks :**

- Topological structure of SIMD array processor is characterized by the data-routing n/w used in interconnecting PEs.
- An inter PE communication n/w can be specified by a set of data routing functions.
- Addresses of PEs in an SIMD m/c is S={0,1,2,…N-1}, each routing function f is a bijection (a one to one and onto mapping) from S to S.
- When a **routing function** is executed via the interconnection n/w the PEi copies contents of Ri register into Rf(i) register of PEf(i). Data routing occurs in all active PEs simultaneously.
- Inactive PE may receive data from another PE if a routing function is executed, but it cannot transmit data.
- To pass data between PEs that are not directly connected in the n/w, the data must be passed through intermediate PEs by executing a sequence of routing functions through the interconnection n/w.

Static v/s Dynamic n/w :

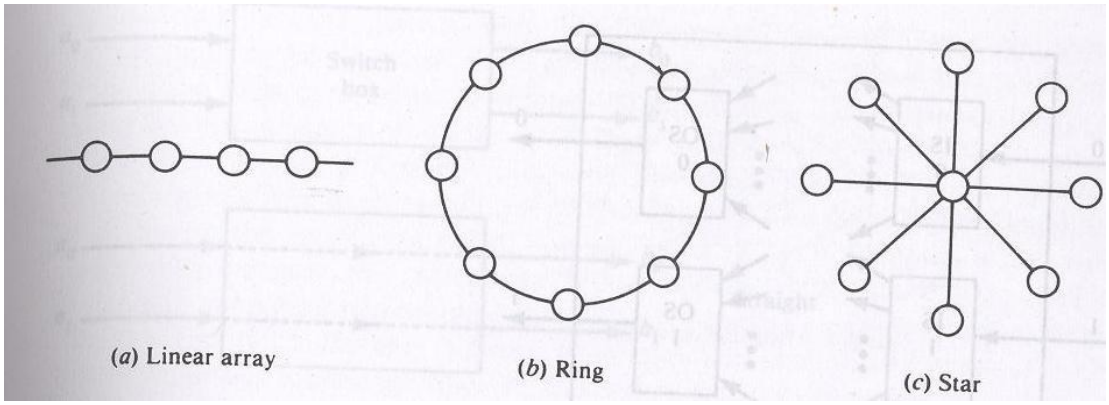
SIMD interconnection n/ws are classified based on n/w topologies: **static n/ws and dynamic n/ws.**

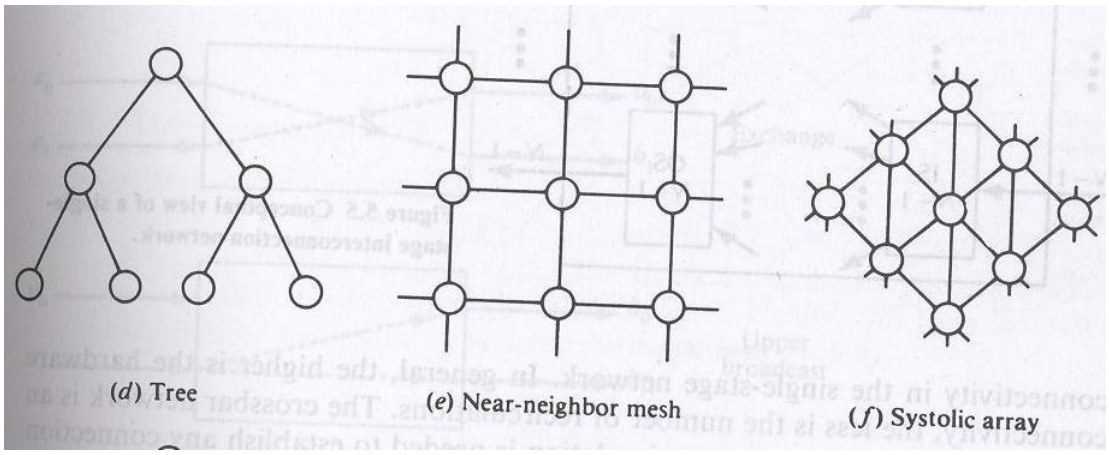
Static n/ws :

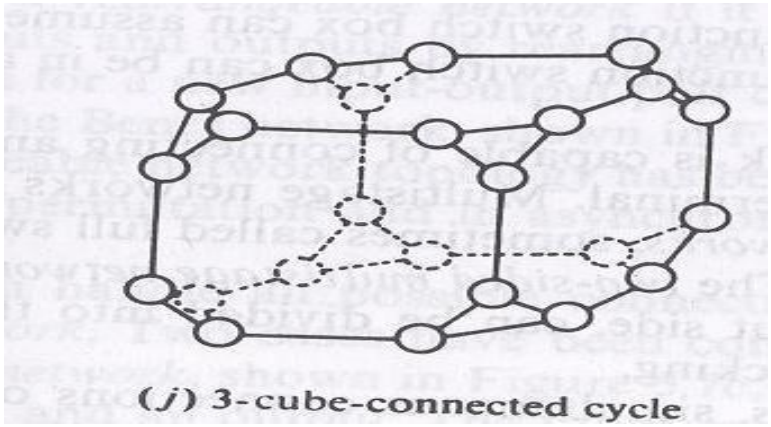
Topologies in the static n/ws can be classified according to the dimensions (1D, 2D, 3D and hypercube) required for layout.

Example:

- 1D: linear array
- 2D: ring, star, tree, mesh, and systolic array.
- 3D: completely connected, chordal ring, 3 cube, 3 cube connected cycle.
- A D dimensional W-wide hypercube contains W nodes in each dimension and there is a connection to a node in each dimension.
- Mesh and 3 cube are actually 2 and 3D hypercubes.

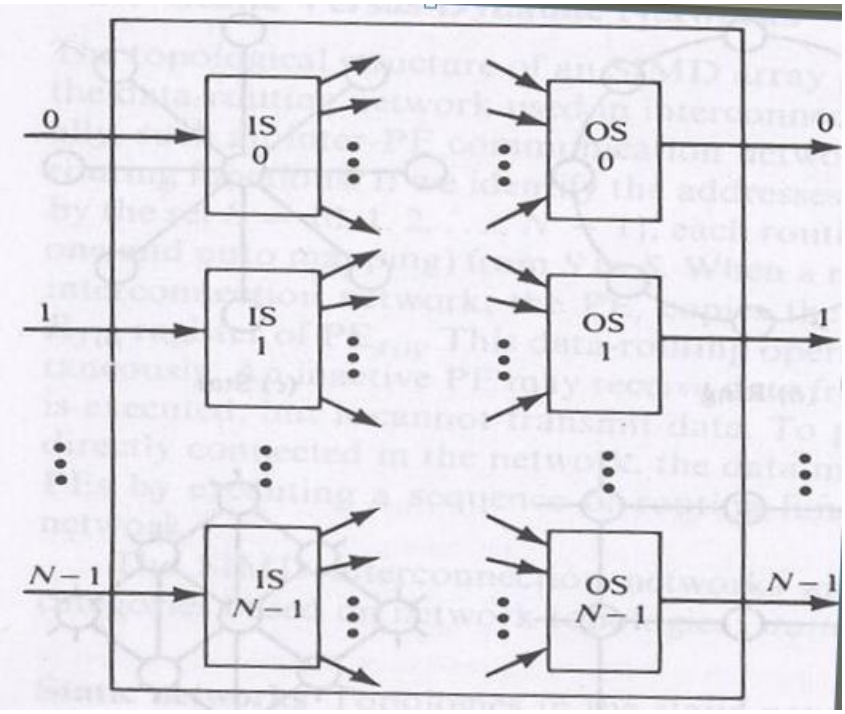






Dynamic n/w:

- 2 classes
- Single stage v/s multistage
- Single stage n/w:
- A single stage n/w is a switching n/w with N i/p selectors (IS) and N o/p selectors (OS).
- Conceptual view of a single stage interconnection n/w



- IS is a 1 to D demultiplexer and OS is an M to1 multiplexer where 1<=D<=N and 1<=M<=N.
- Crossbar switching n/w is a single stage n/w with D=M=N.
- To establish a connecting path, different path control signals will be applied to all IS and OS.

Also called **recirculating n/w.**

- Data items may have to recirculate through the single stage several times before reaching their final destinations.
- The no. of recirculations needed depends on the connectivity in the single stage n/w.

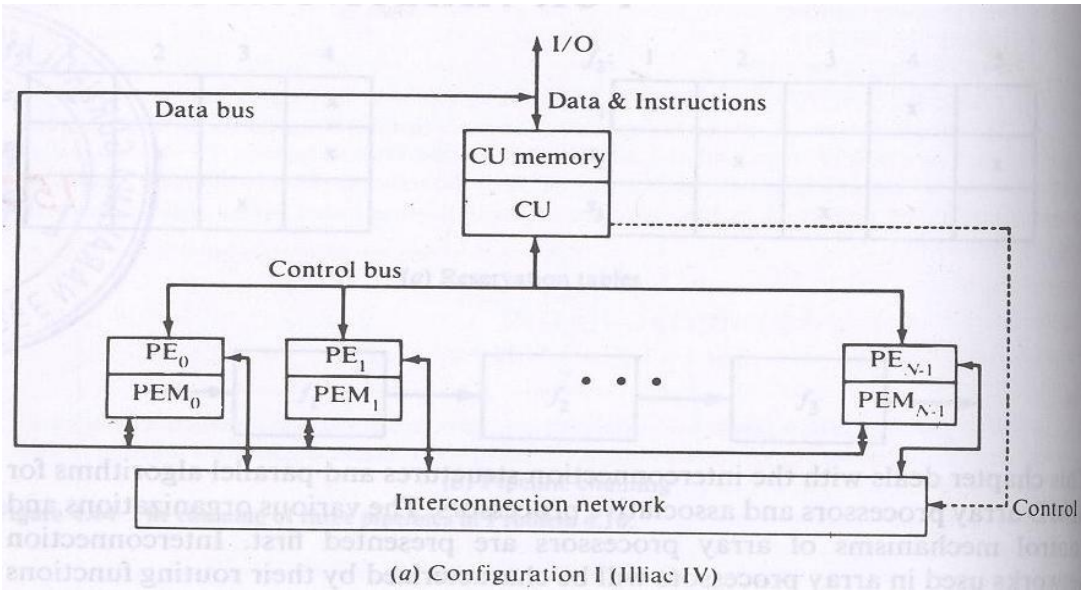
Multistage n/w:

- Many stages of interconnected switches form a multistage SIMD network.
- Multistage n/ws are described by 3 characterizing features: switch box, network topology and control structure.
- **Example:**
  - Banyan
  - Baseline
  - Cube
  - Delta
  - Flip
  - Indirect cube
  - Omega

- Multistage n/w can be **1 sided or 2 sided**.
  - 1 sided n/w, called full switches have i/p-o/p ports on the same side.
  - 2 sided multistage n/w usually have an i/p side and an o/p side, can be divided into 3 classes:
  - blocking, rearrangeable and nonblocking.

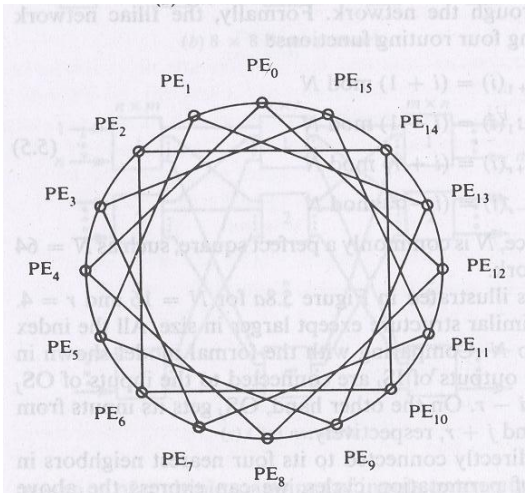
Mesh-connected Illiac n/w:

- Single stage re-circulating n/w is implemented in Illiac-IV array processor with N=64 PEs.
- Each PE_i is allowed to send data to any one of PE_{i+1}, PE_{i-1}, PE_{i+r}, and PE_{i-r} where $r = \sqrt{N}$ in one circulation step through the n/w.



(a) Configuration I (Illiac IV)

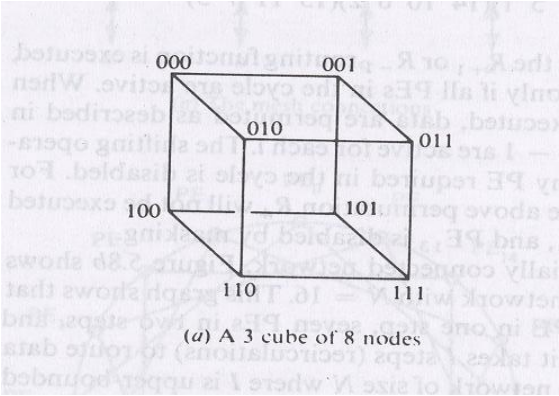
- **Illiac n/w is characterized by the following 4 routing functions**:
  - $R_{+1}(i) = (i+1) \bmod N$
  - $R_{-1}(i) = (i-1) \bmod N$
  - $R_{+r}(i) = (i+r) \bmod N$
  - $R_{-r}(i) = (i-r) \bmod N$
- Where 0<=i<=N-1. N is commonly a perfect square.
- Real Illiac n/w has a similar structure except larger in size.
- The o/ps of ISi are connected to the i/ps of OSj for j=i+1, i-1, i+r, i-r. On the other hand, OSj gets i/ps from ISi for i=j-1, j+1, j-r and j+r.
- Each PEi is directly connected to its 4 nearest neighbours in the mesh n/w.



(*b*) The mesh redrawn

Cube interconnection n/ws :

- The cube can be implemented either as a recirculating n/w or as a multistage n/w for SIMD machines.
- A 3D cube



(*a*) A 3 cube of 8 nodes

- Vertical lines connect vertices (PEs) whose address differ in the most significant bit position. Vertices at both ends of the diagonal lines differ in the middle bit position.
- Horizontal lines differ in the least significant bit position. This unit cube concept can be extended to an n dimensional unit space called an n cube with n bits per vertex.
- A cube n/w for a SIMD m/c with N PEs corresponds to an n cube where $n=\log_2 N$.
- Implementation of a single stage cube network for N=8
- The interconnections of the PEs corresponding to the three routing functions C0, C1, and C2 are shown separately.

Barrel shifter and data manipulator:

- Barrel shifters are also known as plus-minus-2^i (PM2I) networks.
- This type of n/w is based on following routing functions:
- $B_{+i}(j) = (j+2^i) \pmod N$
- $B_{-i}(j) = (j-2^i) \pmod N$
- Where $0<=j<=N-1$, $0<=i<=n-1$ and $n=\log_2 N$.
- The following equivalence is revealed when $r=\sqrt{N} = 2^{n/2}$:
    - $B_{+0} = R_{+1}$
    - $B_{-0} = R_{-1}$
    - $B_{+n/2} = R_{+r}$
    - $B_{-n/2} = R_{-r}$
- This implies Illiac routing functions are a subset of the barrel shifting functions. In addition to adjacent +-1 and fixed distance +-r shiftings, the barrel shifting functions allow either forward or backward shifting of distances which are the integer power of two, ie +-1, +-2,+-4,…$+-2^{n/2}$,..,..,..$+-2^{n-1.}$
- Each PE in a barrel shifter is directly connected to 2(n-1)PEs.
- Connectivity in a barrel shifter is increased from Illiac n/w by having (2n-5). 2^{n-1} more links.
- Illiac n/w has 32 direct links and the same size barrel shifter has 56 links. The two n/ws are identical only when the size is reduced to be no greater than n=2 or N=5.
- The barrel shifter can be implemented as either recirculating single-stage n/w or as a multistage n/w.
- Interconnection patterns in a recirculating barrel shifter for N=8 is given as:
- The barrel shifting functions B+-0, B+-1 and B+-2 are executed by the interconnection patterns shown.
- For a single case barrel shifter of size $N=2^n$, the minimum no. of recirculations B is upper bounded by

$$B <= \frac{\log_2 N}{2}$$

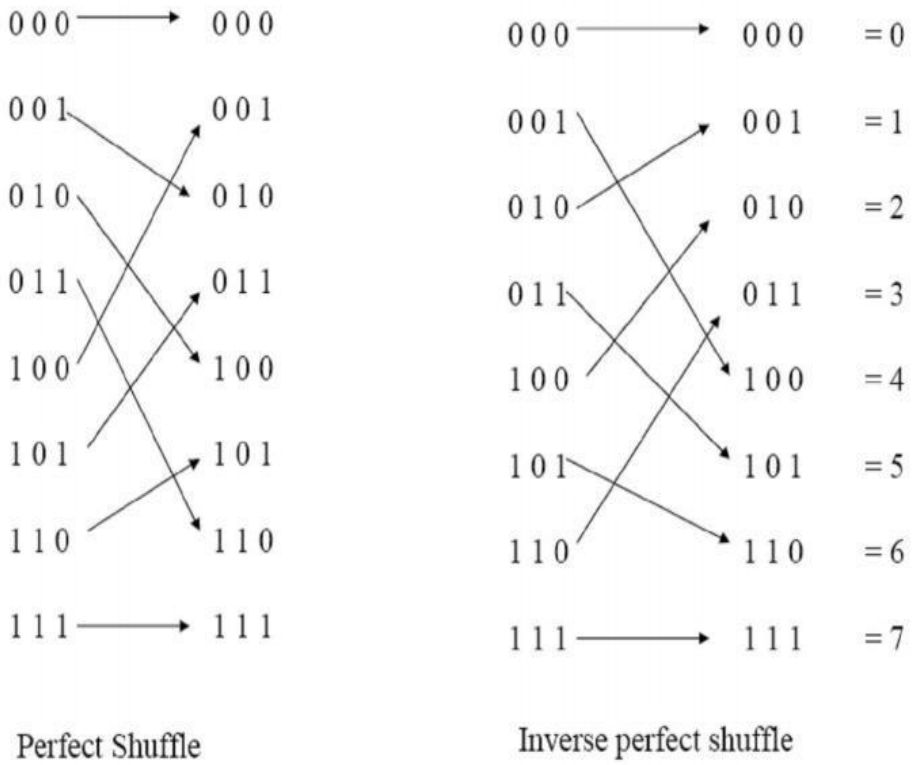
A barrel shifter has been implemented with multiple stages in the form of a data manipulator.

- The data manipulator consists of n stages of N cells is given
- Each cell is essentially a controlled shifter. This n/w is designed for implementing manipulating functions such as permuting, replicating, spacing, masking, and complementing.
- To implement a data manipulating function, proper control lines of the 6 groups (u_1^{2i}, u_2^{2i}, h_1^{2i}, h_2^{2i}, d_1^{2i}, d_2^{2i}) in each column must be properly set through the use of the control register and the associated decoder.
- Schematic logic circuit of a typical cell in a data manipulator is shown:

- For 0<=k<=N-1 and 0<=i<=n-1, the kth cell at stage i has 3 i/ps, 3 sets of o/ps and 3 control signals.
- Individual stage control is used with 3 sets of control signals per stage.
- The control lines ui, hi, and di are connected to the AND gates in each cell of stage i.
- The u^i line controls the backforward barrel shifting (-2i) and the d^i line controls forward barrel shifting (+2i).
- The horizontal line corresponds to no shifting under the control of the h^i signal. The stage i performs the distance 2^i shiftings.
- By passing data through the n stages from left to right, the shifting distance decreases from 2^{n-1} to 2^{n-2} and eventually 2^1 to 2^0 at the o/p end.
- Data manipulation functions that are implementable with the data manipulator

Shuffle Exchange and Omega networks:

- The class of shuffle-exchange networks is based on 2 routing functions shuffle (S) and exchange (E). Let $A = a_{n-1}\ldots a_1a_0$ be a PE address.
- $S(a_{n-1}\ldots a_1a_0) = a_{n-2}\ldots a_1a_0a_{n-1}$
- Where $0<=A<=N-1$ and $n=\log_2 N$.
- The cyclic shifting of the bits in A to the left for one bit position is performed by the S function. This action corresponds to perfectly shuffling a deck of N cards.
- The perfect shuffle cuts the deck into 2 halves from the center and intermixes them evenly.
- The inverse perfect shuffle does the opposite to restore the original ordering.



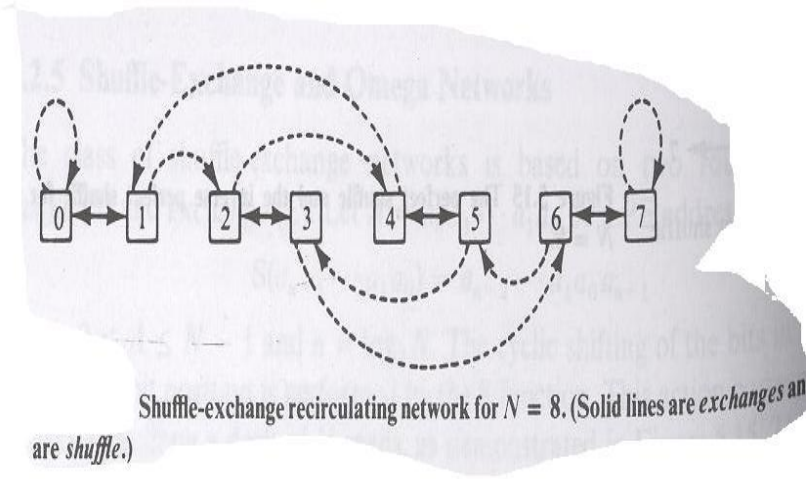
ure 16.13 (a) Perfect Shuffle and Inverse Perfect Shuffle for n = 8

- The exchange routing function E is defined by
- $E(a_{n-1}\ldots a_1 a_0) = a_{n-1}\ldots a_1 a_n$
- The complementing of the least significant digit means the exchange of data between two PEs with adjacent address.
- $E(A)=C_0(A)$, where C_0 is the cube routing function,

$C_i(a_{n-1}\ldots a_1 a_0) = a_{n-1}\ldots a_i + a_i a_{i-1}\ldots a_0$ for i=0,1,2,…n-1.

- These shuffle exchange functions can be implemented as either a recirculating n/w or a multistage n/w.

For N=8, a single stage recirculating shuffle –exchange n/w is shown as:



Shuffle-exchange recirculating network for $N = 8$. (Solid lines are *exchanges* and are *shuffle*.)

- The use of recirculating shuffle-exchange n/w for parallel processing was proposed by Stone.
- A no. of parallel algorithms can be effectively implemented with the use of shuffle and exchange functions. The eg. include fast Fourier transform (FFT), polynomial evaluation, sorting, matrix transposition etc.
- The shuffle exchange functions have been implemented with the multistage Omega n/w by Lawrie.
- Omega n/w for N=8.
- An n X n Omega n/w consists of n identical stages.
- Between 2 adjacent stages is a perfect shuffle interconnection.
- Each stages has N/2 switch boxes under independent box control.
- Each box has 4 functions.
- The switch boxes can be repositioned.
- The ncube network has the same interconnection topology as the repositioned Omega. The 2 n/w differ in 2 aspects.
- The cube n/w uses 2 function switch boxes, whereas the Omega n/w uses 4 function switch boxes. The data flow directions in the 2 n/ws are opposite to each other.

****

UNIT-5

Multiprocessors Architecture and Programming – Functional Structures – Interconnection Networks - Parallel Memory Organizations – Multiprocessor Operating Systems – Language Features to Exploit Parallelism – Multiprocessor Scheduling Strategies.

MULTIPROCESSORS:

• A multiprocessor system is an interconnection of two or more CPU's with memory and input-output equipment.

• Multiprocessors system are classified as multiple instruction stream, multiple data stream systems(MIMD).

• There exists a distinction between multiprocessor and multicomputers that though both support concurrent operations.

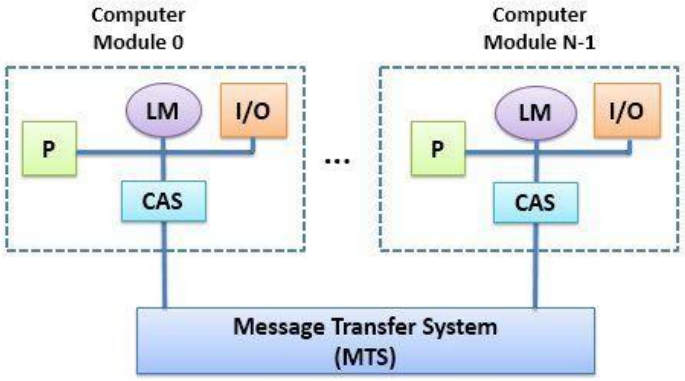
• In multicomputers several autonomous computers are connected through a network and they may or may not communicate but in a multiprocessor system there is a single OS Control that provides interaction between processors and all the components of the system to cooperate in the solution of the problem.

• VLSI circuit technology has reduced the cost of the computers to such a low Level that the concept of applying multiple processors to meet system performance requirements has become an attractive design possibility.

5.1 FUNCTIONAL STRUCTURES.

LOSSELY COUPLED MULTIPROCESSORS:

- Multiprocessor is one which has more than two processors in the system. Now when the **degree of coupling** between these processors is very **low**, the system is called **loosely coupled multiprocessor system**.
- In loosely coupled system each processor has its **own local memory, a set of input-output device**s and a **channel and arbiter switch (CAS)**. We refer to the processor with its local memory and set of input-output devices and CAS as a **computer module**.



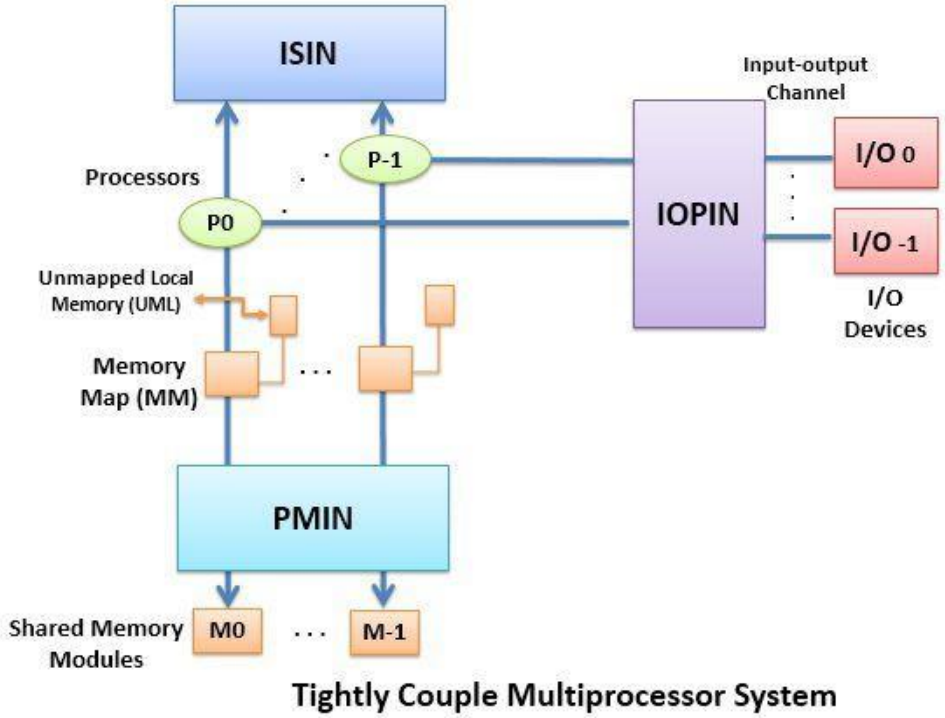
Loosely Couple Multiprocessor System

- Processes that execute on different computer modules communicate with each other by exchanging the **messages** through a physical segment of **message transfer system (MTS)**. The loosely coupled system is also known as **distributed system**. The loosely coupled system is **efficient** when the processes running on different computer module require **minimal interaction**.

TIGHTLLY COUPLED MULTIPROCESSORS:

- The **throughput** of the loosely coupled system may be **too low** for some of the applications that require **fast access time**. In this case, **Tightly coupled microprocessor system** must be used. The tightly coupled system has **processors**, **shared memory modules**, **input-output channels**.
- The above units of the tightly coupled system are connected through the set of three interconnection networks, processor-memory interconnection network (PMIN), I/O-processor interconnection network (IOPIN) and the interrupt-signal interconnection network (ISIN). The use of these three interconnection networks is as follow.

- **PMIN:** It is a switch which **connects** each **processor** to every **memory module**. It can also be designed in a way that a processor can broadcast data to one or more memory module.
- **ISIN:** It allows each **processor** to **direct an interrupt** to any **other processor**.
- **IOPIN**: It allows a **processor** to **communicate** with an **I/O channel** which is connected to input-output devices.



Tightly Couple Multiprocessor System

PROCESSOR CHARACTRISTICS FOR MULTIPROCESSING:

Process recover-ability:

The processes and the processor should be considered as two different entities. In case of a process failure, the interrupted process should be retrieved and executed by another processor. If a processor does not support this feature, the reliability of a multiprocessor is reduced. Most processor maintain the state of the currently executing instruction in the internal registers. In case of a failure the result is not written back to the memory.

Efficient context switching:

The processor should support more than one addressing domain (context) and thus, an efficient context switching for the effective utilization of resources. The context switch time is the total time taken to switch from one process to another process, which includes saving the current state of one process and restoring the state of new process. A special instruction and a number of register sets are used to accomplish the context switching efficiency. The multiprocessor architecture should support such instruction and the register sets.

Large virtual and physical address space:

The processor should support large physical address space to allow programs to access large amount of data if required. It should also support a large virtual address space. The virtual memory space should be segmented to achieve modular sharing. It should also provide the mechanisms for memory protection and checking software reliability .

Effective synchronization primitives:

The processors should provide implementation of some mechanisms that can be used to establish mutual exclusion among concurrently executing cooperating processes. The mechanisms for achieving mutual exclusion need certain operations to be treated and executed as indivisible operations, Semaphore is one of the generally used mechanisms to achieve mutual exclusion.

Inter processor communication mechanism:

The processors that are used to build multiprocessor must have support for inter-processor communication. The mechanism for inter-processor communication must be implemented in the hardware. Such a mechanism becomes especially useful when processors frequently exchange requests for services.

5.2 INTERCONNECTION NETWORKS.

TIME SHARED OR COMMON BUSES:

- A common-bus multiprocessor system consists of a number of processors connected through a common path to a memory unit.
- Disadvantage - Only one processor can communicate with the memory or another processor at any given time. As a consequence, the total overall transfer rate within the system is limited by the speed of the single path .
- A more economical implementation of a dual bus structure is depicted in Fig. below. 4. Part of the local memory may be designed as a cache memory attached to the CPU.

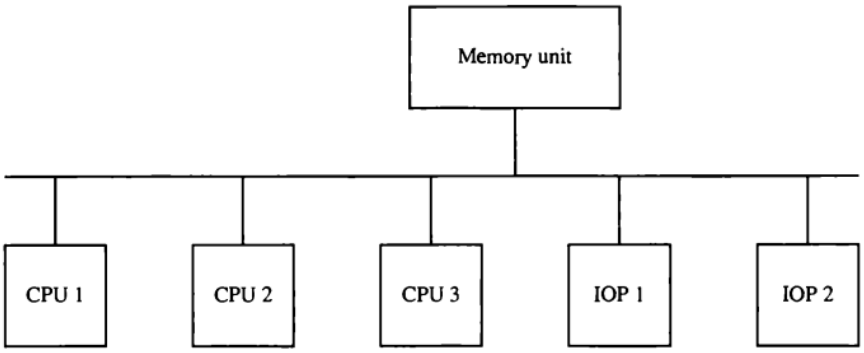


Figure 1 Time-shared common bus organization.

CROSSBAR SWITCH:

- Consists of a number of crosspoints that are placed at intersections between processor buses and memory module paths.
- The small square in each crosspoint is a switch that determines the path from a processor to a memory module.
- Advantage - Supports simultaneous transfers from all memory modules .
- Disadvantage - The hardware required to implement the switch can become quite large and complex.

MULTIPORT MEMORIES:

- A multiport memory system employs separate buses between each memory module and each CPU.
- The module must have internal control logic to determine which port will have access to memory at any given time.
- Memory access conflicts are resolved by assigning fixed priorities to each memory port.
- Advantage - The high transfer rate can be achieved because of the multiple paths.
- Disadvantage - It requires expensive memory control logic and a large number of cables and connections

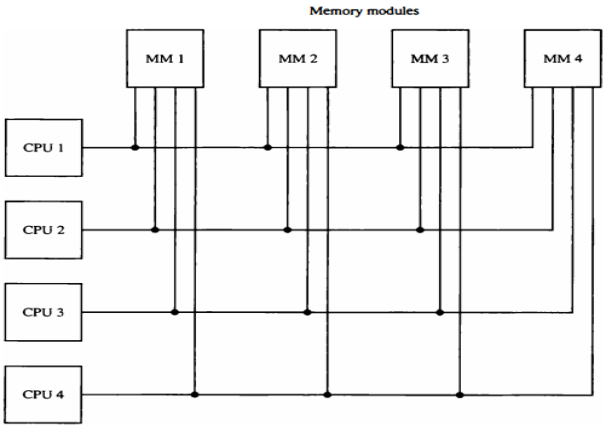
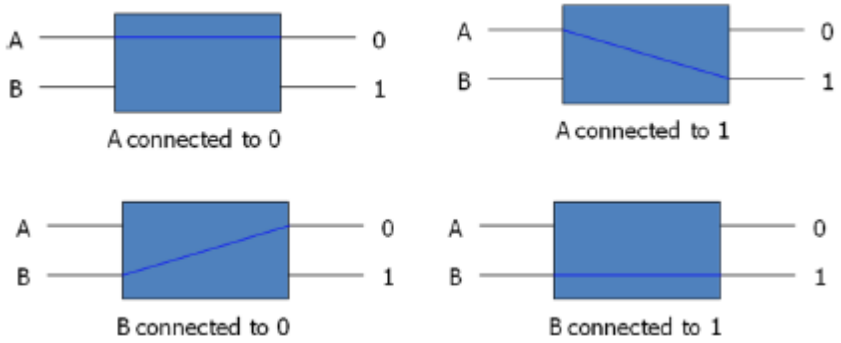


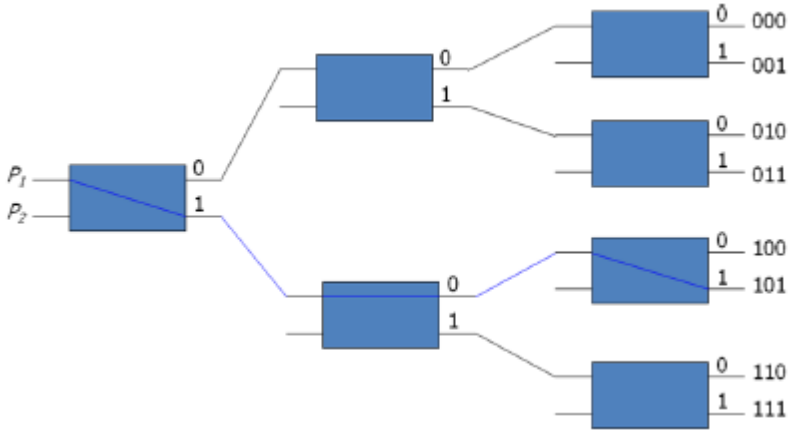
Figure 3 Multiport memory organization.

MULTITAGE INTERCONNECTION NETWORK:

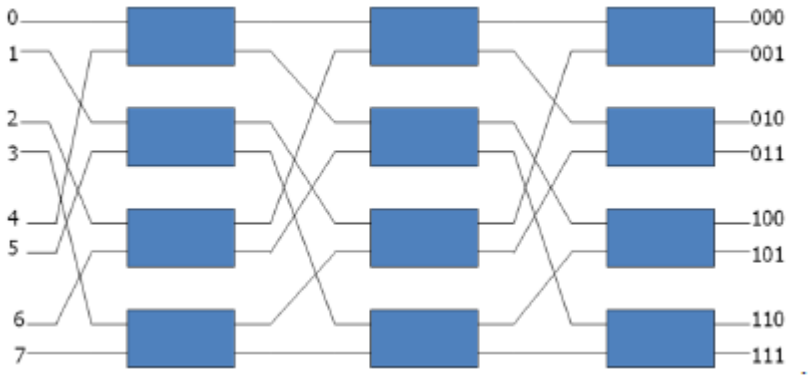
- The basic component of a multistage network is a two-input, two-output interchange switch as shown in Fig. below.



- Using the 2x2 switch as a building block, it is possible to build a multistage network to control the communication between a number of sources and destinations.



- One such topology is the omega switching network shown in Fig. below



PERFORMANCE OF INTERCONNECTION NETWORK:

- A multiprocessor system consists of multiple processing units connected via some interconnection network plus the software needed to make the processing units work together.
- A number of communication styles exist for multiprocessing networks. These can be broadly classified according to the communication model as shared memory (single address space) versus message passing (multiple address spaces).
    - Communication in shared memory systems is performed by writing to and reading from the global memory
    - Communication in message passing systems is accomplished via send and receive commands.
    - In both cases, the interconnection network plays a major role in determining the communication speed. Two schemes are introduced, namely static and dynamic interconnection networks.
        - Static networks form all connections when the system is designed rather than when the connection is needed. In a static network, messages must be routed along established links. (hypercube, mesh, and k-ary n-cube topologies)
        - Dynamic interconnection networks establish connections between two or more nodes on the fly as messages are routed along the links(bus, crossbar, and multistage interconnection).

Analysis and Performance:

Dynamic Networks

- **The Crossbar**;
    - the cost of the crossbar system can be measured in terms of the number of switching elements (cross points) required inside the crossbar. The crossbar possesses a quadratic rate of cost (complexity) given by $O(N^2)$.
    - The delay (latency) within a crossbar switch, measured in terms of the amount of the input to output delay, is constant.The crossbar possesses a constant rate of

delay (latency) given by $O(1)$. It should be noted that the high cost (complexity) of the crossbar network pays off in the form of reduction in the time (latency).

- The crossbar is however a nonblocking network; that is, it allows multiple output connection pattern (permutation) to be achieved.
- A fault-tolerant system can be simply defined as a system that can still function even in the presence of faulty components inside the system. The crossbar can be affected by a single-point failure. Nevertheless, segmenting the crossbar and realizing each segment independently can reduce the effect of a single-point failure in a crossbar.

Multiple Bus

- It consists of M memory modules, N processors, and B buses. A given bus is dedicated to a particular processor for the duration of a bus transaction.
- A processor?memory transfer can use any of the available buses. Given B buses in the system, then up to B requests for memory use can be served simultaneously.
- A multiple bus possesses an $O(B)$ rate of cost (complexity) growth.
- The multiple bus possesses an $O(B \times N)$ rate of delay (latency) growth.
- Multiple bus-multiprocessor organization offers the desirable feature of being highly reliable and fault-tolerant. This is because a single bus failure in a B bus system will leave ($B-1$) distinct fault-free paths between the processors and thememory modules.
- On the other hand, when the number of buses is less than the number of memory modules (or the number of processors), bus contention is expected to increase.

Multistage Interconnection Networks

- Each stage consists of $N/2$, 2×2 SEs.
- The network cost (complexity), measured in terms of the total number of SEs, is $O(N \times log_2 N)$.
- The latency (time) complexity, measured by the number of SEs along the path from input to output, is $O(log_2 N)$.
- Simplicity of message routing inside a MIN is a desirable feature of such networks. There exists a unique path between a given input?output pair.
- MINs are characterized as being 0-fault tolerant; that is, a MIN cannot tolerate the failure of a single component.

Network	Delay (Latency)	Cost (Complexity)	Blocking	Degree of Fault Tolerance
Bus	$O(N)$	$O(1)$	Yes	0
Multiple bus	$O(mN)$	$O(m)$	Yes	$(m-1)$
MINs	$O(\log N)$	$O(N \log N)$	Yes	0
Crossbar	$O(1)$	$O(N^2)$	No	0

Figure 2.10: Performance Comparison of Dynamic Networks.

Static Networks

1. Degree of a node, d, is defined as the number of channels incident on the node.
2. Diameter, D, of a network having N nodes is defined as the longest path, p, of the shortest paths between any two nodes. For example, the diameter of a 4×4 Mesh $D = 6$.
3. A network is said to be symmetric if it is isomorphic to itself with any node labeled as the origin; that is, the network looks the same from any node. Rings and Tori networks are symmetric while linear arrays and mesh networks are not.

- **Completely Connected Networks (CCNs)**;
  - the cost of a completely connected network having N nodes, measured in terms of the number of links in the network, is given by $N(N-1)/2$, that is, $O(N^2)$.
  - The delay (latency) complexity of CCNs, measured in terms of the number of links traversed as messages are routed from any source to any destination, is constant, that is, O(1).
  - The degree of a node in CCN is $N-1$, that is, $O(N)$, while the diameter is $O(1)$.
- **Linear Array Networks (LCNs)**
  - In this network architecture, each node is connected to its two immediate neighboring nodes. Each of the two nodes at the extreme ends of the network is connected only to its single immediate neighbor.
  - The network cost (complexity) measured in terms of the number of nodes of the linear array is $O(N)$.
  - The delay (latency) complexity measured in terms of the average number of nodes that must be traversed to reach from a source node to a destination node is $N/2$, that is, $O(N)$.

- The node degree in the linear array is 2, that is, $O(1)$ and the diameter is $(N-1)$, that is, $O(N)$.

Network	Degree (d)	Diameter (D)	Cost (No. of Links)	Symmetry	Worst Delay
CCNs	$N-1$	1	$N(N-1)/2$	Yes	1
Linear array	2	$N-1$	$N-1$	No	N
Binary tree	3	$2(\lceil \log_2 N \rceil - 1)$	$N-1$	No	$\log_2 N$
n-cube	$\log_2 N$	$\log_2 N$	$nN/2$	Yes	$\log_2 N$
2D-mesh	4	$2(n-1)$	$2(N-n)$	No	$\sqrt{N}$
k-ary n-cube	$2n$	$N\lfloor k/2 \rfloor$	$n \times N$	Yes	$k \times \log_2 N$

Figure 2.11: Performance Characteristics of Static Networks.

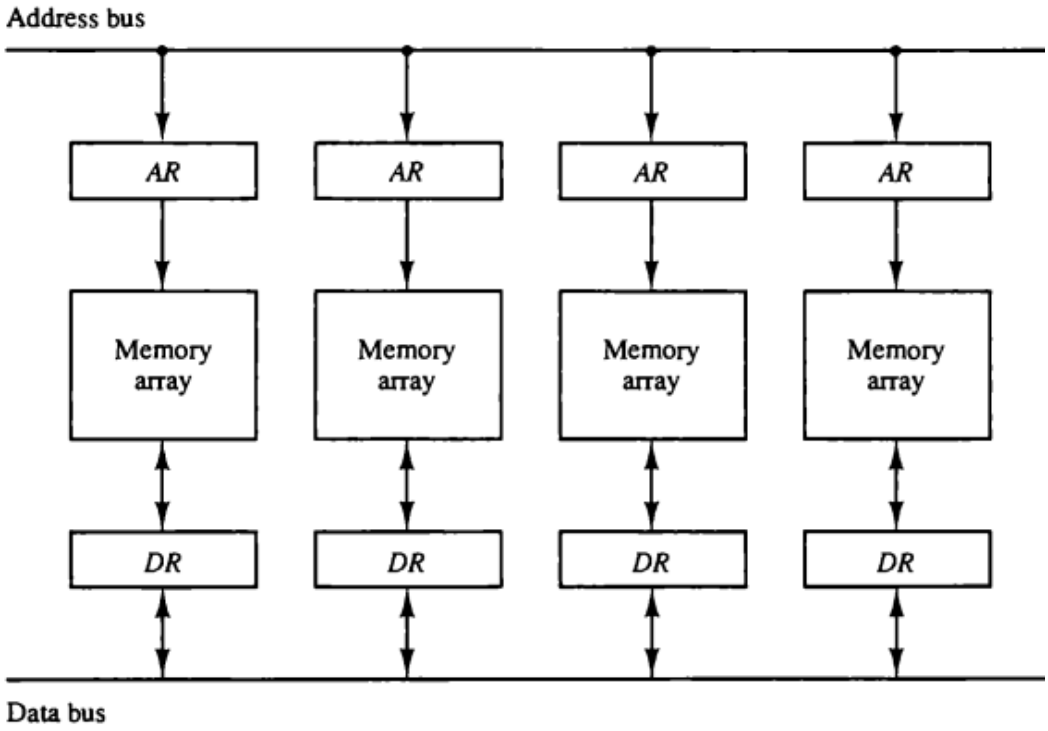
5.3 PARALLEL MEMORY ORGANIZATIONS.

INTERLEAVED MEMORY COFIGURATION:

- Memory interleaving is a concept of dividing the main memory in a number of modules or banks with each module interacting with processor independent of others.
- Each memory modules has its own memory address register and data register. Memory Address Register(MAR) is connected with common unidirectional memory bus and data register is connected with common data bus.
- When CPU requests a data, it sends the address to memory via common memory bus. The address is saved in the memory address register. The data is now retrived from module and kept in data register, from where it is sent to the cpu via common data bus.

To understand this scenario we take an example of a two way interleaved memory.

- In a two way interleaved memory, main memory is divided into two modules in such a way that all the positive addresses are in first module and rest all odd addresses are in second module.
- So, all the positive addresses go to first module and so on. We can also say that, least significant bit of the memory address specifies the address of the module. That is if LSB is 0, first module is selected and if LSB is 1, second module is selected. If it is a four way interleaved memory then least two significant bits of memory address is used to specify the memory module.

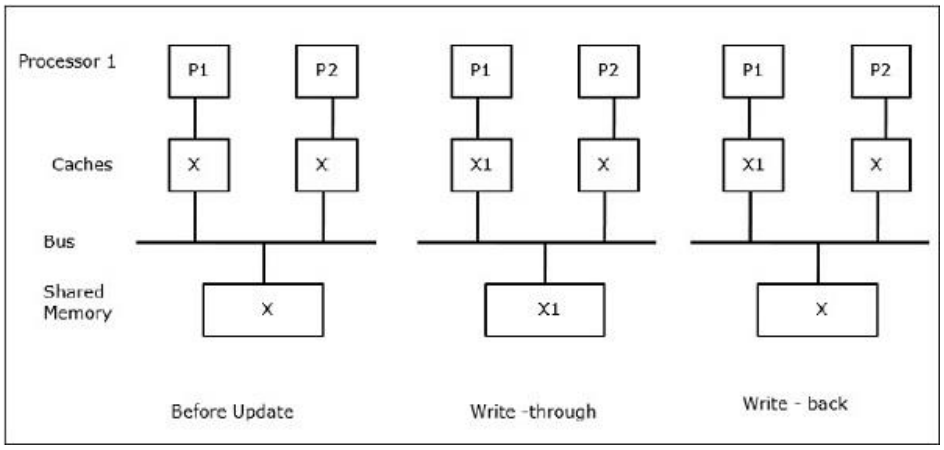


MULTI CACHE PROBLEM AND SOLUTION:

- ✓ A coherence problem may occur in a multicache system as soon as data inconsistency exists in the private caches and the main memory.
- ✓ Without an effective solution to the coherence problem, the effectiveness of a multicache system will be inherently limited.
- ✓ The problem is closely examined in this paper and previous solutions, both centralized approaches and distributed approaches, are analyzed based on the notion of semicritical sections.
- ✓ A state model is then presented which clarifies various coherence mechanisms as well as introduces a new state to enable the multicache system to more efficiently handle the processor writes.
- ✓ Software guidance, for performance and not for integrity, is advocated in a new proposal which in a practical multicache environment explores the benefit of the new state with little cost.

Cache Coherence:

- ✓ In a multiprocessor system, data inconsistency may occur among adjacent levels or within the same level of the memory hierarchy. For example, the cache and the main memory may have inconsistent copies of the same object.
- ✓ As multiple processors operate in parallel, and independently multiple caches may possess different copies of the same memory block, this creates **cache coherence problem**. **Cache coherence schemes** help to avoid this problem by maintaining a uniform state for each cached block of data.

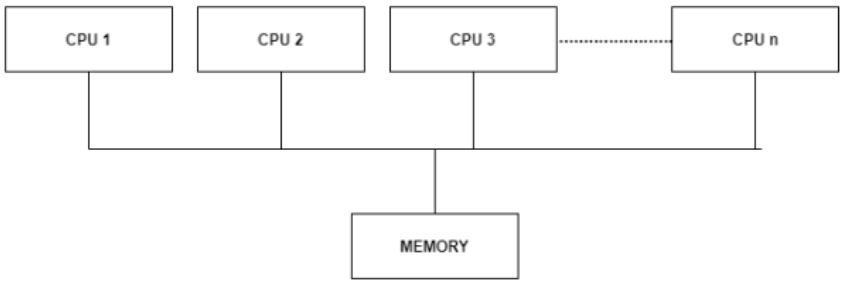


- ✓ Let X be an element of shared data which has been referenced by two processors, P1 and P2. In the beginning, three copies of X are consistent. If the processor P1 writes a new data X1 into the cache, by using **write-through policy**, the same copy will be written immediately into the shared memory. In this case, inconsistency occurs between cache memory and the main memory. When a **write-back policy** is used, the main memory will be updated when the modified data in the cache is replaced or invalidated.

5.4 MULTIPROCESSOR OPERATING SYSTEM.

CLASSIFICATION:

Most computer systems are single processor systems i.e they only have one processor. However, multiprocessor or parallel systems are increasing in importance nowadays. These systems have multiple processors working in parallel that share the computer clock, memory, bus, peripheral devices etc. An image demonstrating the multiprocessor architecture ;



Multiprocessing Architecture

Types of Multiprocessors

There are mainly two types of multiprocessors

Symmetric and Asymmetric multiprocessors.

Symmetric Multiprocessors

In these types of systems, each processor contains a similar copy of the operating system and they all communicate with each other. All the processors are in a peer to peer relationship i.e. no master - slave relationship exists between them.

An example of the symmetric multiprocessing system is the Encore version of Unix for the Multimax Computer.

Asymmetric Multiprocessors

In asymmetric systems, each processor is given a predefined task. There is a master processor that gives instruction to all the other processors. Asymmetric multiprocessor system contains a master slave relationship.

Asymmetric multiprocessor was the only type of multiprocessor available before symmetric multiprocessors were created. Now also, this is the cheaper option.

Advantages of Multiprocessor Systems

There are multiple advantages to multiprocessor systems. Some of these are −

More reliable Systems

In a multiprocessor system, even if one processor fails, the system will not halt. This ability to continue working despite hardware failure is known as graceful degradation. For example: If there are 5 processors in a multiprocessor system and one of them fails, then also 4 processors are still working. So the system only becomes slower and does not ground to a halt.

Enhanced Throughput

If multiple processors are working in tandem, then the throughput of the system increases i.e. number of processes getting executed per unit of time increase. If there are N processors then the throughput increases by an amount just under N.

More Economic Systems

Multiprocessor systems are cheaper than single processor systems in the long run because they share the data storage, peripheral devices, power supplies etc. If there are multiple processes that share data, it is better to schedule them on multiprocessor systems with shared data than have different computer systems with multiple copies of the data.

Disadvantages of Multiprocessor Systems

There are some disadvantages as well to multiprocessor systems. Some of these are:

Increased Expense

Even though multiprocessor systems are cheaper in the long run than using multiple computer systems, still they are quite expensive. It is much cheaper to buy a simple single processor system than a multiprocessor system.

Complicated Operating System Required

There are multiple processors in a multiprocessor system that share peripherals, memory etc. So, it is much more complicated to schedule processes and impart resources to processes.than in single processor systems. Hence, a more complex and complicated operating system is required in multiprocessor systems.

Large Main Memory Required

All the processors in the multiprocessor system share the memory. So a much larger pool of memory is required as compared to single processor systems.

SOFTWARE REQUREMNET FOR MULTIPROCESSOR:

The task of writing basic system software for a multiprocessor system is simplified if the system is organized so that it may be programmed as a single virtual machine. This strategy is considered for tightly coupled real time systems with many processors. Software is written in a high level language supporting parallel execution, without knowledge of the target hardware configuration.

OS requirements:

- Must provide functionality of a multiprogramming OS plus additional features to support multiple processors
- Simultaneous concurrent processes or threads: kernel routines need to be reentrant
- Scheduling done by any processor, can create conflicts.
- Synchronization through locks is required.
- Memory management needs to be coordinated in the different processors
- Much more complex than just multiprogramming OS.

5.5 EXPLOITING CONCURRENCY FOR MULTIPROCESSING.

EXPLOTING PARALLELISM:

Parallelism in computer architectures is the ability for multiple actions to occur simultaneously. Parallelism is often categorized into the three types discussed here.

Data level parallelism (DLP)

- General purpose architectures often support data-level parallelism (DLP), the parallelism found in the application of identical operations across many different data elements [HGLS86]. Some scientific applications and many multimedia applications exhibit large levels of DLP. Vector processors such as Cray Research's Cray-1 and Cray-2 use single instructions to operate on vectors of data elements at a time.

- SIMD instructions allow programmers to specify a single operation to apply to multiple, distinct data elements. Unlike vector processors, generally only a small number of elements that can fit in a single register at one time are operated on simultaneously.

Instruction level parallelism (ILP)

- Many processors today also support instruction-level parallelism (ILP) by taking advantage of resource and data independence between instructions and executing multiple operations at one time. Superscalar processors (or superscalars) are capable of fetching and executing more than one instruction simultaneously and thus exploit some level of ILP.

- Statically-scheduled superscalars may execute several instructions in program order if there are no hazards between the instructions. These superscalars stop executing later instructions when any hazard is first encountered which limits ILP. Another class of architectures, very long instruction word architectures (VLIW), contain multiple operations in a single instruction.

MULTIPROCESSOR SCHEDULING STRATEGIES:

- Multiprocessor scheduling focuses on designing the system's scheduling function, which consists of more than one processor. Multiple CPUs share the load (load sharing) in multiprocessor scheduling so that various processes run simultaneously. In general, multiprocessor scheduling is complex as compared to single processor scheduling. In the multiprocessor scheduling, there are many processors, and they are identical, and we can run any process at any time.

- The multiple CPUs in the system are in close communication, which shares a common bus, memory, and other peripheral devices. So we can say that the system is tightly coupled. These systems are used when we want to process a bulk amount of data, and these systems are mainly used in satellite, weather forecasting, etc.
- There are cases when the processors are identical, i.e., homogenous, in terms of their functionality in multiple-processor scheduling. We can use any processor available to run any process in the queue.

Dimensions of multiple processor management:

- Multiprocessor, scheduling is two dimensional. The scheduler has to decide which process to run and which CPU to run it on. This extra dimension greatly complicates scheduling on multiprocessors.

- Another complicating factor is that in some systems, all the processes are unrelated whereas in others they come in groups. An example of the former situation is a timesharing system in which independent users start up independent processes. The processes are unrelated and each one can be scheduled without regard to the other ones.

DETERMINISTIC SCHEDULING MODELS:

- In this section we examine schedules in which more than one processor can be used to optimize measures of performance. This section is divided into two major parts.
- In the first part-Common Scheduling Environments-the parameters identified in most of the scheduling literature and discussed earlier prevail unless stated otherwise.
- That is, we assume a number of identical processors, a set of tasks with equal or unequal execution times, and a precedence order. Both preemptive and nonpreemptive disciplines are examined.
- In the second part-Special Scheduling Environments- additional constraints are introduced. These constraints include systems with a finite number of resources in each member of a set of resource classes, periodic jobs with specified initiation and completion times, and the presence of intermediate deadlines within a schedule.

****