UNIT IV PARALLELISIM

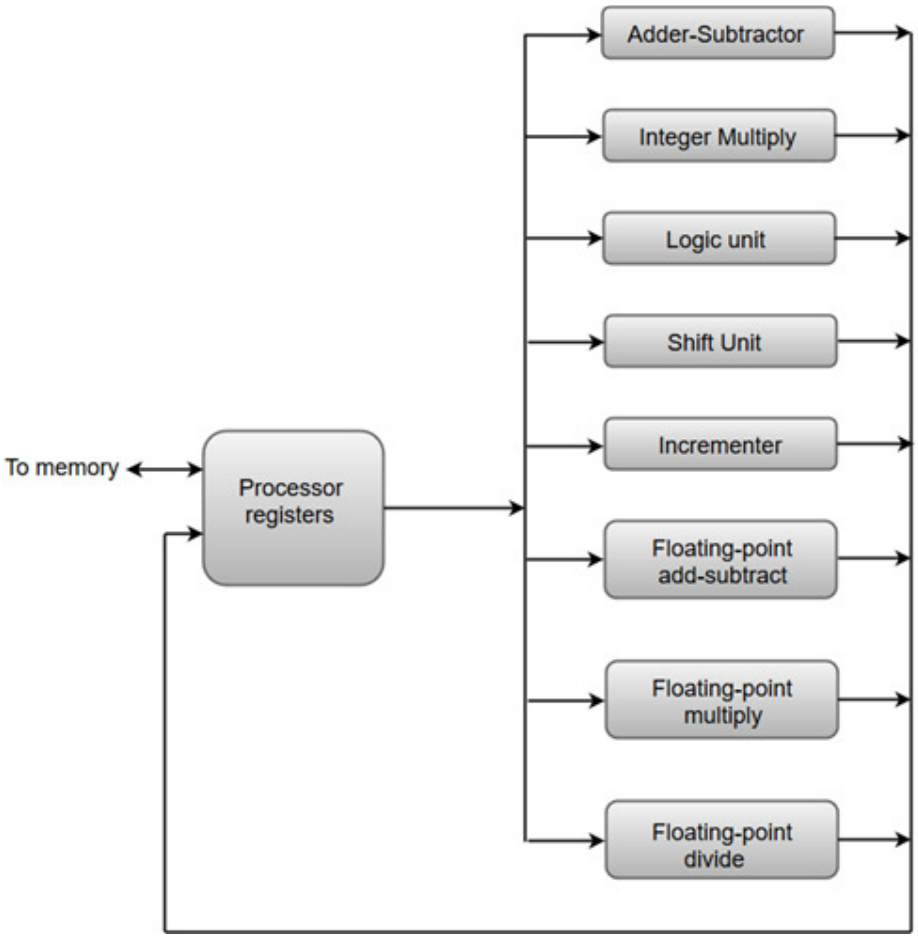
Parallel processing challenges – Flynn‘s classification – SISD, MIMD, SIMD, SPMD, and Vector Architectures - Hardware multithreading – Multi-core processors and other Shared Memory Multiprocessors - Introduction to Graphics Processing Units, Clusters, Warehouse Scale Computers and other Message-Passing Multiprocessors.

Why Parallel Architecture?

- ✓ Parallel computer architecture adds a new dimension in the development of computer system by using more and more number of processors.
- ✓ In principle, performance achieved by utilizing large number of processors is higher than the performance of a single processor at a given point of time.

Parallel Processing

- ✓ Parallel processing can be described as a class of techniques which enables the system to achieve simultaneous data-processing tasks to increase the computational speed of a computer system.
- ✓ A parallel processing system can carry out simultaneous data-processing to achieve faster execution time.
- ✓ For instance, while an instruction is being processed in the ALU component of the CPU, the next instruction can be read from memory.
- ✓ The primary purpose of parallel processing is to enhance the computer processing capability and increase its throughput,
- ✓ A parallel processing system can be achieved by having a multiplicity of functional units that perform identical or different operations simultaneously.
- ✓ The data can be distributed among various multiple functional units.
- ✓ The following diagram shows one possible way of separating the execution unit into eight functional units operating in parallel.
- ✓ The operation performed in each functional unit is indicated in each block if the diagram:



- ✓ The adder and integer multiplier performs the arithmetic operation with integer numbers.
- ✓ The floating-point operations are separated into three circuits operating in parallel.
- ✓ The logic, shift, and increment operations can be performed concurrently on different data.
- ✓ All units are independent of each other, so one number can be shifted while another number is being incremented.

- ✓ Parallel computers can be roughly classified according to the level at which the hardware supports parallelism, with multi-core and multi-processor computers having multiple processing elements within a single machine.
- ✓ In some cases parallelism is transparent to the programmer, such as in bit-level or instruction-level parallelism.
- ✓ But explicitly parallel algorithms, particularly those that use concurrency, are more difficult to write than sequential ones, because concurrency introduces several new classes of potential software bugs, of which race conditions are the most common.
- ✓ Communication and synchronization between the different subtasks are typically some of the greatest obstacles to getting optimal parallel program performance.

Advantages of Parallel Computing over Serial Computing are as follows:

1. It saves time and money as many resources working together will reduce the time and cut potential costs.
2. It can be impractical to solve larger problems on Serial Computing.
3. It can take advantage of non-local resources when the local resources are finite.
4. Serial Computing 'wastes' the potential computing power, thus Parallel Computing makes better work of hardware.

Types of Parallelism:

1. **Bit-level parallelism:** It is the form of parallel computing which is based on the increasing processor's size. It reduces the number of instructions that the system must execute in order to perform a task on large-sized data. *Example:* Consider a scenario where an 8-bit processor must compute the sum of two 16-bit integers. It must first sum up the 8 lower-order bits, then add the 8 higher-order bits, thus requiring two instructions to perform the operation. A 16-bit processor can perform the operation with just one instruction.
2. **Instruction-level parallelism:** A processor can only address less than one instruction for each clock cycle phase. These instructions can be re-ordered and

grouped which are later on executed concurrently without affecting the result of the program. This is called instruction-level parallelism.

3. **Task Parallelism:** Task parallelism employs the decomposition of a task into subtasks and then allocating each of the subtasks for execution. The processors perform execution of sub tasks concurrently.
4. **Data-level parallelism (DLP)** – Instructions from a single stream operate concurrently on several data – Limited by non-regular data manipulation patterns and by memory bandwidth

Architectural Trends

- ✓ When multiple operations are executed in parallel, the number of cycles needed to execute the program is reduced.
- ✓ However, resources are needed to support each of the concurrent activities.
- ✓ Resources are also needed to allocate local storage.
- ✓ The best performance is achieved by an intermediate action plan that uses resources to utilize a degree of parallelism and a degree of locality.
- ✓ Generally, the history of computer architecture has been divided into four generations having following basic technologies −
    - Vacuum tubes
    - Transistors
    - Integrated circuits
    - VLSI
- ✓ Till 1985, the duration was dominated by the growth in bit-level parallelism.
- ✓ 4-bit microprocessors followed by 8-bit, 16-bit, and so on.
- ✓ To reduce the number of cycles needed to perform a full 32-bit operation, the width of the data path was doubled. Later on, 64-bit operations were introduced.
- ✓ The growth in **instruction-level-parallelism** dominated the mid-80s to mid-90s.
- ✓ The RISC approach showed that it was simple to pipeline the steps of instruction processing so that on an average an instruction is executed in almost every cycle.
- ✓ Growth in compiler technology has made instruction pipelines more productive.
- ✓ In mid-80s, microprocessor-based computers consisted of

- An integer processing unit
- A floating-point unit
- A cache controller
- SRAMs for the cache data
- Tag storage

✓ As chip capacity increased, all these components were merged into a single chip.

✓ Thus, a single chip consisted of separate hardware for integer arithmetic, floating point operations, memory operations and branch operations.

✓ Other than pipelining individual instructions, it fetches multiple instructions at a time and sends them in parallel to different functional units whenever possible. This type of instruction level parallelism is called **superscalar execution**.

FLYNN'S CLASSIFICATION

✓ Flynn's taxonomy is a specific classification of parallel computer architectures that are based on the number of concurrent instruction (single or multiple) and data streams (single or multiple) available in the architecture.

✓ The four categories in Flynn's taxonomy are the following:

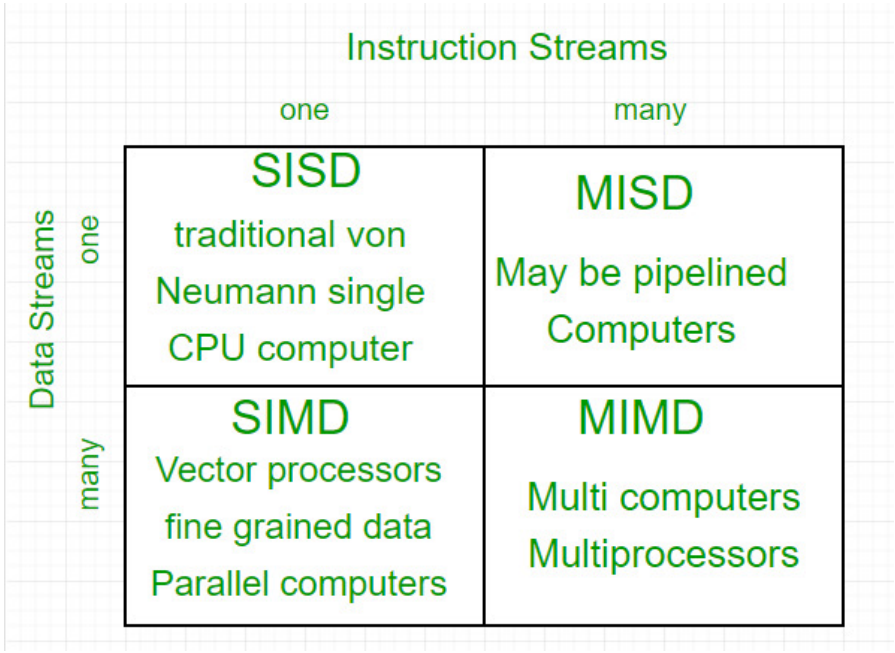
1. (SISD) single instruction, single data
2. (SIMD) single instruction, multiple data
3. (MISD) multiple instruction, single data
4. (MIMD) multiple instruction, multiple data

✓ Instruction stream: is the sequence of instructions asϒ executed by the machine

✓ Data Stream is a sequence of data including input, or partialϒ or temporary result, called by the instruction Stream.

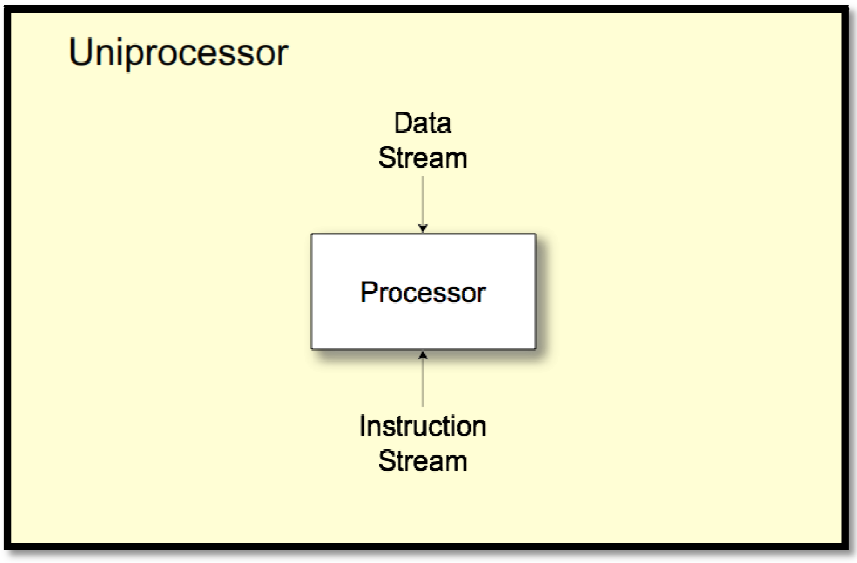
✓ Instructions are decoded by the control unit and then ctrl unit send the instructions to the processing units for execution. •

✓ Data Stream flows between the processors and memory bi directionally.



SISD

An SISD computing system is a uniprocessor machine which is capable of executing a single instruction, operating on a single data stream.

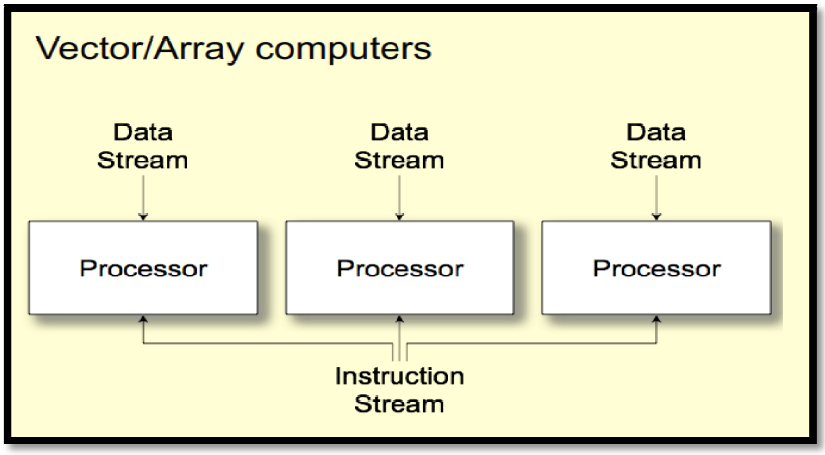


- ✓ In SISD, machine instructions are processed in a sequential manner and computers adopting this model are popularly called sequential computers.
- ✓ Most conventional computers have SISD architecture. All the instructions and data to be processed have to be stored in primary memory.
- ✓ The speed of the processing element in the SISD model is limited(dependent) by the rate at which the computer can transfer information internally.

✓ Dominant representative SISD systems are IBM PC, workstations.

SIMD

- An SIMD system is a multiprocessor machine capable of executing the same instruction on all the CPUs but operating on different data streams



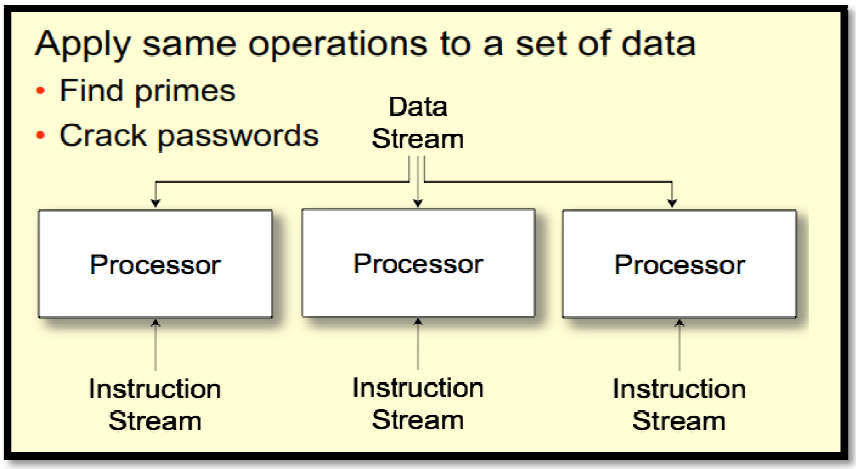
✓ Machines based on an SIMD model are well suited to scientific computing since they involve lots of vector and matrix operations.

✓ So that the information can be passed to all the processing elements (PEs) organized data elements of vectors can be divided into multiple sets(N-sets for N PE systems) and each PE can process one data set.

✓ Dominant representative SIMD systems is Cray's vector processing machine.

MISD

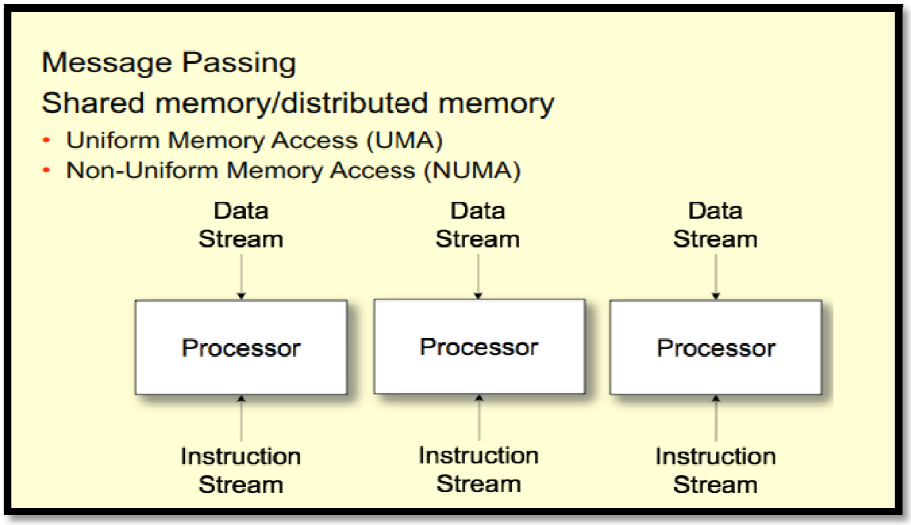
✓ An MISD computing system is a multiprocessor machine capable of executing different instructions on different PEs but all of them operating on the same dataset .



✓ The system performs different operations on the same data set. Machines built using the MISD model are not useful in most of the application, a few machines are built, but none of them are available commercially.

MIMD

✓ An MIMD system is a multiprocessor machine which is capable of executing multiple instructions on multiple data sets.



✓ Each PE in the MIMD model has separate instruction and data streams; therefore machines built using this model are capable to any kind of application.

✓ Unlike SIMD and MISD machines, PEs in MIMD machines work asynchronously.

✓ MIMD machines are broadly categorized into

- **shared-memory MIMD** and
- **distributed-memory MIMD**

based on the way PEs are coupled to the main memory.

In the **shared memory MIMD** model (tightly coupled multiprocessor systems), all the PEs are connected to a single global memory and they all have access to it. The communication between PEs in this model takes place through the shared memory, modification of the data stored in the global memory by one PE is visible to all other PEs. Dominant representative shared memory MIMD systems are Silicon Graphics machines and Sun/IBM's SMP (Symmetric Multi-Processing).

In **Distributed memory MIMD** machines (loosely coupled multiprocessor systems) all PEs have a local memory. The communication between PEs in this model takes place through the interconnection network (the inter process communication channel, or IPC). The network connecting PEs can be configured to tree, mesh or in accordance with the requirement.

VECTOR ARCHITECTURES

- ✓ A multithreaded CPU is not a parallel architecture, strictly speaking; multithreading is obtained through a single CPU, but it allows a programmer to design and develop applications as a set of programs that can virtually execute in parallel: namely, threads.
- ✓ Multithreading is solution to avoid waiting clock cycles as the missing data is fetched: making the CPU manage more peer-threads concurrently; if a thread gets blocked, the CPU can execute instructions of another thread, thus keeping functional units busy.
- ✓ Each thread must have a private Program Counter and a set of private registers, separate from other threads.
- ✓ In a traditional scalar processor, the basic data type is an n-bit word.

- ✓ The architecture often exposes a register file of words, and the instruction set is composed of instructions that operate on individual words.
- ✓ In a vector architecture, there is support of a vector datatype, where a vector is a collection of VL n-bit words (VL is the vector length).
- ✓ There may also be a vector register file, which was a key innovation of the Cray architecture.
- ✓ Previously, vector machines operated on vectors stored in main memory.
- ✓ Figures 1 and 2 illustrate the difference between vector and scalar data types, and the operations that can be performed on them.

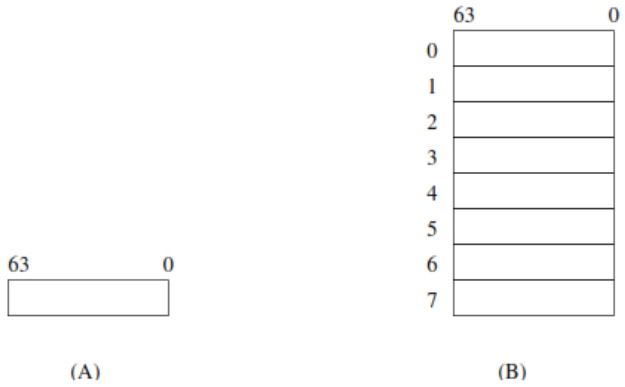


Figure 1: (A): a 64-bit word, and (B): a vector of 8 64-bit words

- ✓ Vector load/store instructions provide the ability to do strided and scatter / gather memory accesses, which take data elements distributed throughout memory and pack them into sequential vectors/streams placed in vector/stream registers.
- ✓ This promotes data locality.
- ✓ It results in less data pollution, since only useful data is loaded from the memory system.
- ✓ It provides latency tolerance because there can be many simultaneous outstanding memory accesses.
- ✓ Vector instructions such as VLD and VST provide this capability.

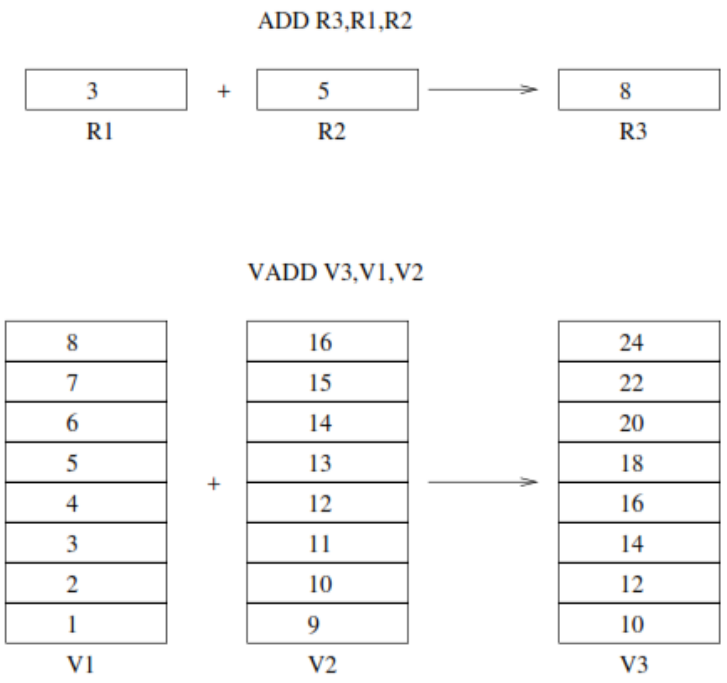


Figure 2: Difference between scalar and vector add instructions

HARDWARE MULTITHREADING

Multithreading

- A mechanism by which the instruction streams is divided into several smaller streams

  (threads) and can be executed in parallel is called multithreading.

Hardware Multithreading

- Increasing utilization of a processor by switching to another thread when one thread is stalled is known as hardware multithreading.

Thread

- A thread includes the program counter, the register state, and the stack. It is a lightweight process; whereas threads commonly share a single address space, processes don't.

Thread Switch

- The act of switching processor control from one thread to another within the same process. It is much less costly than a processor switch.

Process

- A process includes one or more threads, the address space, and the operating system state. Hence, a process switch usually invokes the operating system, but not a thread switch.

Types of Multi-threading

1. Fine-grained Multithreading
2. Coarse-grained Multithreading
3. Simultaneous Multithreading

Coarse-grained Multithreading

A version of hardware multithreading that implies switching between threads only after significant events, such as a last-level cache miss.

- This change relieves the need to have thread switching be extremely fast and is much less likely to slow down the execution of an individual thread, since instructions from other threads will only be issued when a thread encounters a costly stall.

Advantage

- To have very fast thread switching.
- Doesn't slow down thread.

Disadvantage

- It is hard to overcome throughput losses from shorter stalls, due to pipeline start-up costs.
- Since CPU issues instructions from 1 thread, when a stall occurs, the pipeline must be emptied.
- New thread must fill pipeline before instructions can complete.
- Due to this start-up overhead, coarse-grained multithreading is much more useful for reducing the penalty of high-cost stalls, where pipeline refill is negligible compared to the stall time.

Fine-grained Multithreading

- A version of hardware multithreading that implies switching between threads after every instruction resulting in interleaved execution of multiple threads. It switches from one thread to another at each clock cycle.
- This interleaving is often done in a round-robin fashion, skipping any threads that are stalled at that clock cycle.

To make fine-grained multithreading practical, the processor must be able to switch threads on every clock cycle.

Advantage

- Vertical waste is eliminated.
- Pipeline hazards cannot arise.
- Zero switching overhead
- Ability to hide latency within a thread i.e., it can hide the throughput losses that arise from both short and long stalls.
- Instructions from other threads can be executed when one thread stalls.
- High execution efficiency
- Potentially less complex than alternative high performance processors.

Disadvantage

- Clock cycles are wasted if a thread has little operation to execute.
- Needs a lot of threads to execute.
- It is expensive than coarse-grained multithreading.
- It slows down the execution of the individual threads, since a thread that is ready to execute without stalls will be delayed by instructions from other threads.

Simultaneous multithreading (SMT)

- It is a variation on hardware multithreading that uses the resources of a multiple-issue, dynamically scheduled pipelined processor to exploit thread-level parallelism at the same time it exploits instruction level parallelism.
- The key insight that motivates SMT is that multiple-issue processors often have more functional unit parallelism available than most single threads can effectively use.

Since SMT relies on the existing dynamic mechanisms, it does not switch resources every cycle.

- Instead, SMT is always executing instructions from multiple threads, to associate instruction slots and renamed registers with their proper threads.

Advantage

- It is ability to boost utilization by dynamically scheduling functional units among multiple threads.
- It increases hardware design facility.
- It produces better performance and add resources to a fine grained manner.

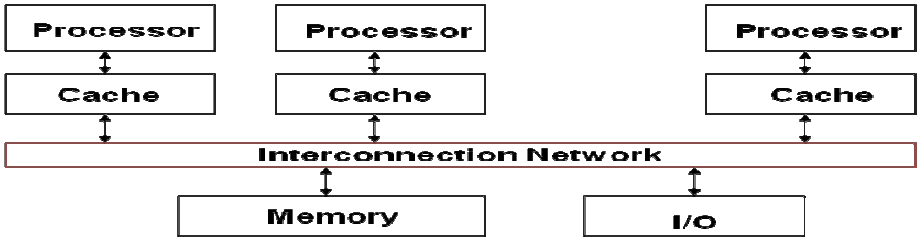
Disadvantage

It cannot improve performance if any of the shared resources are the limiting bottlenecks for the performance.

MULTICORE AND OTHER SHARED MEMORY MULTIPROCESSORS

Multiprocessor: A computer system with at least two processors

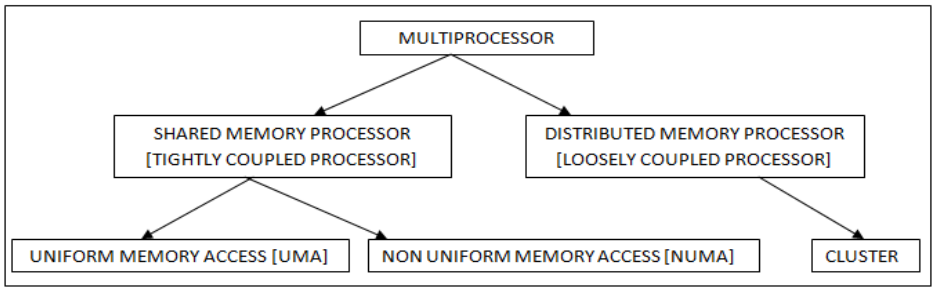
- **Multicore:** More than one processor available within a single chip.



The conventional multiprocessor system used is commonly referred as shared memory multiprocessor system.

- **Shared Memory Multiprocessor (SMP)** is one that offers the programmer a single physical address space across all processors which is nearly always the case for multicore chips.
- Processors communicate through shared variables in memory, with all processors capable of accessing any memory location via loads and stores.

- Systems can still run independent jobs in their own virtual address spaces, even if they all share a physical address space.
- Use of shared data must be coordinated via synchronization primitives (locks) that allow access to data to only one processor at a time



Shared Memory Multiprocessor System.[Tightly coupled processor]

- The conventional multiprocessor system used is commonly referred as shared memory multiprocessor system.
- Single address space shared by all processors. Because every processor communicates through a shared global memory.
- For high speed real time processing, these systems are preferable as their throughput is high as compared to loosely coupled systems
- In tightly coupled system organization, multiple processors share a global main memory, which may have many modules.
- Tightly coupled systems use a common bus, crossbar, or multistage network to connect processors, peripherals, and memories.

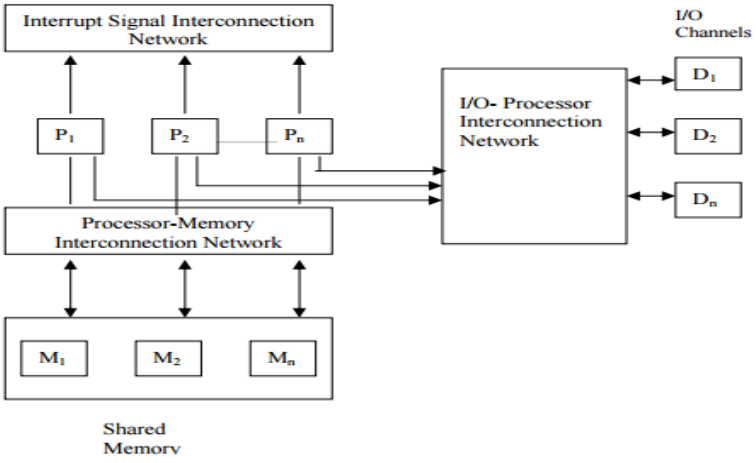
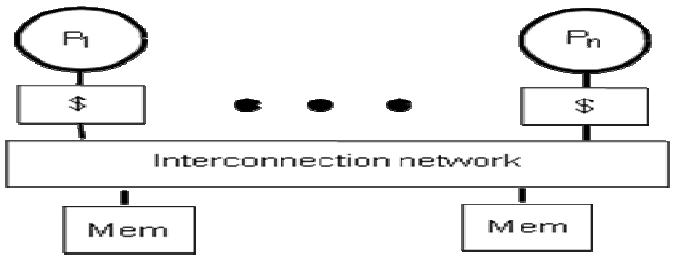


Figure : Tightly coupled system organization

- Two common styles of implementing Shared Memory Multiprocessors (SMP) are,

Uniform memory access (UMA) multiprocessors

- In this model, main memory is uniformly shared by all processors in multiprocessor systems and each processor has equal access time to shared memory.
- This model is used for time-sharing applications in a multi user environment
- Tightly-coupled systems (high degree of resource sharing) suitable for general purpose and time-sharing applications by multiple users



Physical memory uniformly shared by all processors, with equal access time to all words.

- Processors may have local cache memories. Peripherals also shared in some fashion.
- UMA architecture models are of two 20types,

Symmetric:

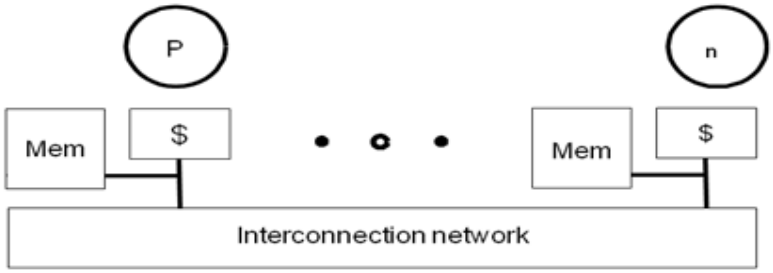
- All processors have equal access to all peripheral devices. All processors are identical.

Asymmetric:

- One processor (master) executes the operating system other processors may be of different types and may be dedicated to special tasks.

Non Uniform Memory Access (NUMA) multiprocessors

- In shared memory multiprocessor systems, local memories can be connected with every processor. The collections of all local memories form the global memory being shared.
- In this way, global memory is distributed to all the processors. In this case, the access to a local memory is uniform for its corresponding processor as it is attached to the local memory.
- But if one reference is to the local memory of some other remote processor, then the access is not uniform.
- It depends on the location of the memory. Thus, all memory words are not accessed uniformly. All local memories form a global address space accessible by all processors
- Programming NUMAs are harder but NUMAs can scale to larger sizes and have lower latency to local memory
- Memory is common to all the processors. Processors easily communicate by means of shared variables.
- These systems differ in how the memory and peripheral resources are shared or distributed
- The access time varies with the location of the memory word.

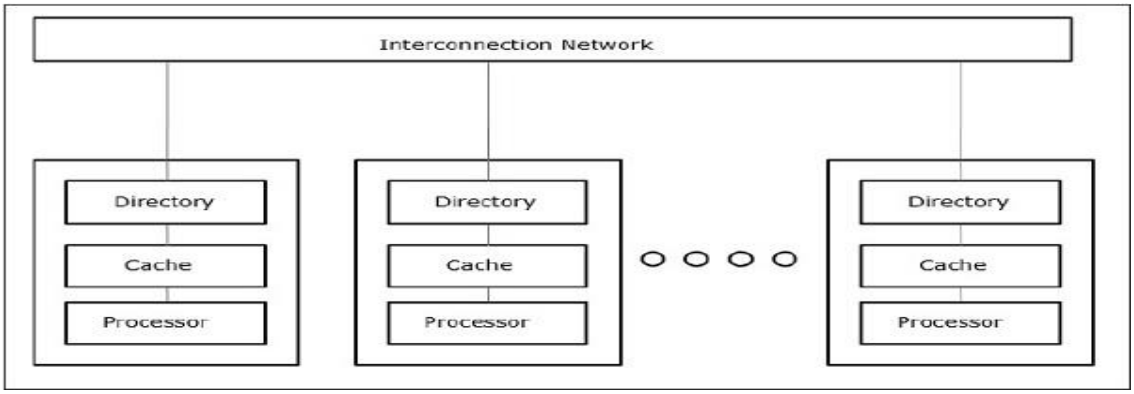


Distributed Memory (NUMA)

- Cache Only Memory Architecture. The COMA model is a special case of the NUMA

model. Here all the distributed memories are converted to cache memories.

- The local memories for the processor at each node are used as cache instead of actual



Distributed Memory [Loosely Coupled Systems]

- These systems do not share the global memory because shared memory concept gives rise to the problem of memory conflicts, which in turn slows down the execution of instructions.
- Therefore, to alleviate this problem, each processor in loosely coupled systems is having a large local memory (LM), which is not shared by any other processor.
- Thus, such systems have multiple processors with their own local memory and a set of I/O devices.
- This set of processor, memory and I/O devices makes a computer system.

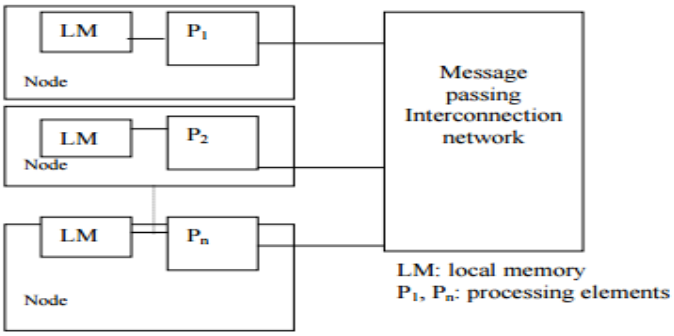


Figure 14: Loosely coupled system organisation

- ✓ Therefore, these systems are also called multi-computer systems.
- ✓ These computer systems are connected together via message passing interconnection network through which processes communicate by passing messages to one another.
- Since every computer system or node in multicomputer systems has a separate memory, they are called distributed multicomputer systems. These are also called loosely coupled systems.

GPU (Graphics Processing Unit)

- ✓ A graphics processing unit (GPU) is a computer chip that performs rapid mathematical calculations, primarily for the purpose of rendering images.
- ✓ In the early days of computing, the central processing unit (CPU) performed these calculations.
- ✓ As more graphics-intensive applications such as AutoCAD were developed, however, their demands put strain on the CPU and degraded performance.
- ✓ GPUs came about as a way to offload those tasks from CPUs and free up processing power.
- ✓ Today, graphics chips are being adapted to share the work of CPUs and train deep neural networks for AI applications.
- ✓ A GPU may be found integrated with a CPU on the same circuit, on a graphics card or in the motherboard of a personal computer or server.
- ✓ NVIDIA, AMD, Intel and ARM are some of the major players in the GPU market.

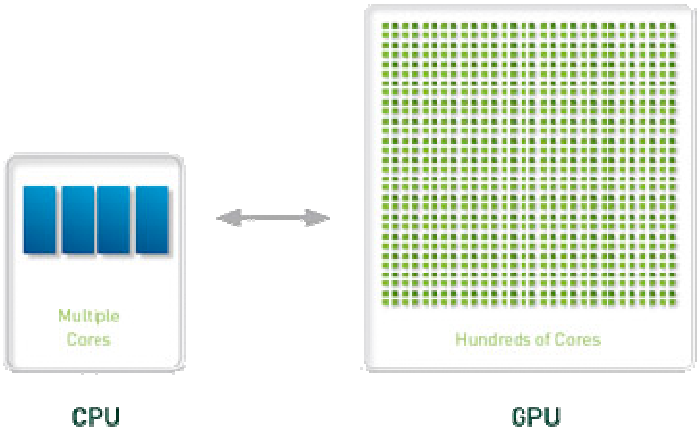
GPU vs. CPU

- ✓ A GPU is able to render images more quickly than a CPU because of its parallel processing architecture, which allows it to perform multiple calculations at the same time.
- ✓ A single CPU does not have this capability, although multicore processors can perform calculations in parallel by combining more than one CPU onto the same chip.

- ✓ A CPU also has a higher clock speed, meaning it can perform an individual calculation faster than a GPU so it is often better equipped to handle basic computing tasks.
- ✓ In general, a GPU is designed for data-parallelism and applying the same operation to multiple data-items (SIMD).
- ✓ A CPU is designed for task-parallelism and doing different operations.

How a GPU works

- ✓ CPU and GPU architectures are also differentiated by the number of cores.
- ✓ The core is essentially the processor within the processor.
- ✓ Most CPUs have between four and eight cores, though some have up to 32 cores.
- ✓ Each core can process its own tasks, or threads.
- ✓ Because some processors have multithreading capability -- in which the core is divided virtually, allowing a single core to process two threads -- the number of threads can be much higher than the number of cores.
- ✓ This can be useful in video editing and transcoding.
- ✓ CPUs can run two threads (independent instructions) per core (the independent processor unit). GPUs can have four to 10 threads per core.

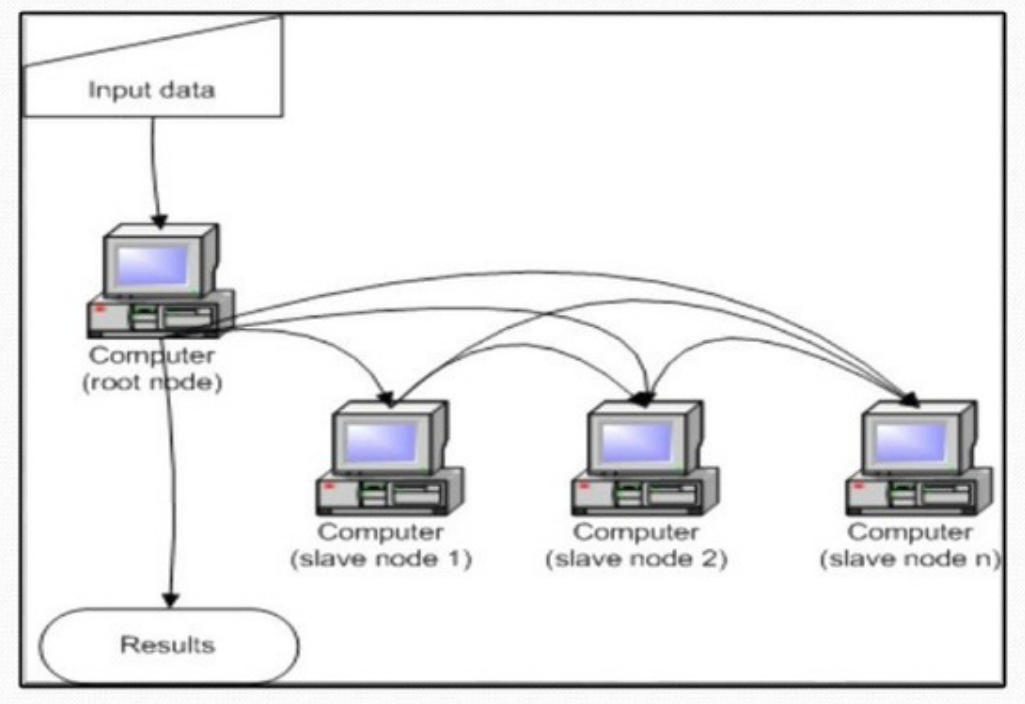


- ✓ GPU computing is the use of a GPU (graphics processing unit) as a co-processor to accelerate CPUs for general-purpose scientific and engineering computing.
- ✓ The GPU accelerates applications running on the CPU by offloading some of the compute-intensive and time consuming portions of the code.
- ✓ The rest of the application still runs on the CPU. From a user's perspective, the application runs faster because it's using the massively parallel processing power of the GPU to boost performance. This is known as "heterogeneous" or "hybrid" computing.
- ✓ A CPU consists of four to eight CPU cores, while the GPU consists of hundreds of smaller cores.
- ✓ Together, they operate to crunch through the data in the application.
- ✓ This massively parallel architecture is what gives the GPU its high compute performance.
- ✓ There are a number of GPU-accelerated applications that provide an easy way to access high-performance computing (HPC).

CLUSTER SYSTEM

- ✓ Clustered systems are similar to parallel systems as they both have multiple CPUs.
- ✓ However a major difference is that clustered systems are created by two or more individual computer systems merged together.
- ✓ Basically, they have independent computer systems with a common storage and the systems work together.

A diagram to better illustrate this is:



The clustered systems are a combination of hardware clusters and software clusters. The hardware clusters help in sharing of high performance disks between the systems. The software clusters makes all the systems work together .

Each node in the clustered systems contains the cluster software. This software monitors the cluster system and makes sure it is working as required. If any one of the nodes in the clustered system fail, then the rest of the nodes take control of its storage and resources and try to restart.

Types of Clustered Systems

- **High performance Cluster**
  - 1000 nodes, high level parallel process
- **Load Balancing Cluster**
  - Balance the work loads
- **Web service Cluster**
  - Web pages & applications
- **Storage Cluster**

- Parallel file systems

- **Database Cluster**
  - Oracle parallel server

WSC

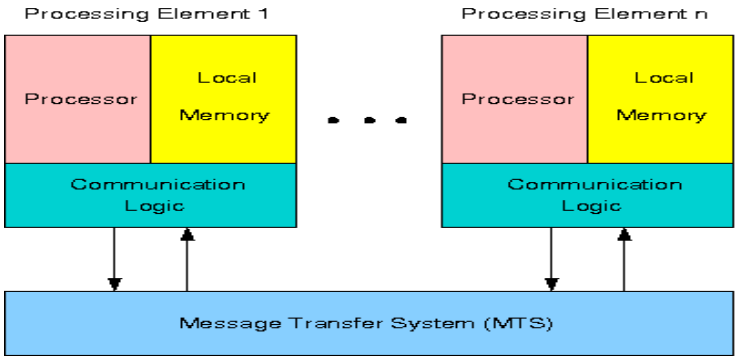
- ✓ Warehouse-scale computers (WSCs) form the foundation of internet services that people use for search, social networking, online maps, video sharing, online shopping, email, cloud computing, etc.
- ✓ The ever increasing popularity of internet services has necessitated the creation of WSCs in order to keep up with the growing demands of the public.
- ✓ Although WSCs may seem to be large datacenters, their architecture and operation are different from datacenters.
- ✓ The WSC is a descendant of the supercomputer. Today's WSCs act as one giant machine.
- ✓ The main parts of a WSC are the building with the electrical and cooling infrastructure, the networking equipment and the servers, about 50000 to 100000 of them.
- ✓ The costs are of the order of $150M to build such an infrastructure . WSCs have many orders of magnitude more users than high performance computing and play a very important role today.

Message Passing Multiprocessor

Communicating between multiple processors by explicitly sending and receiving information.

Send message routine: A routine used by a processor in machines with private memories to pass a message to another processor.

- **Receive message routine:** A routine used by a processor in machines with private memories to accept a message from another processor.



- Distributed memory multicomputer system consists of multiple computers, known as nodes, inter-connected by message passing network.
- Each node acts as an autonomous computer having a processor, a local memory and sometimes I/O devices.

- In this case, all local memories are private and are accessible only to the local processors.
- This is why, the traditional machines are called **no-remote-memory-access (NORMA)** machines.