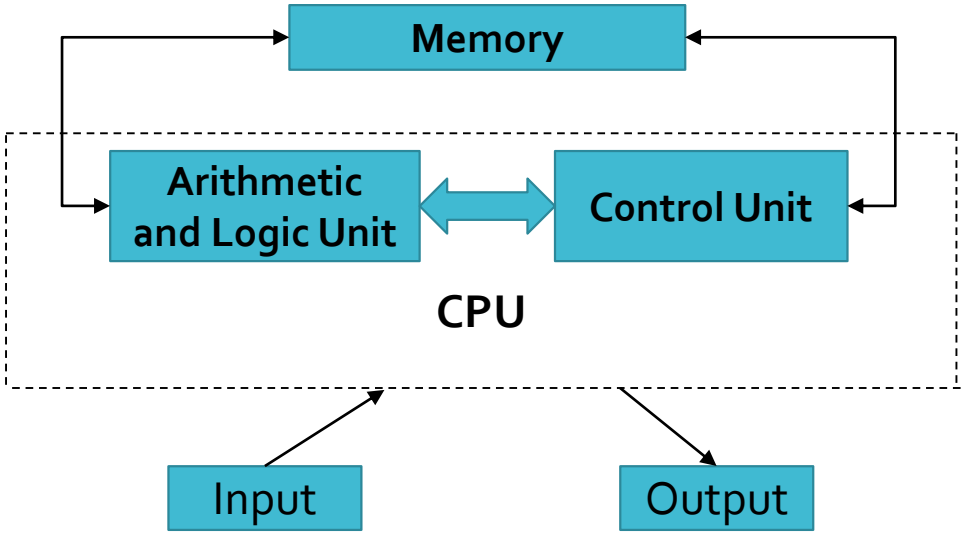
Parallel Processing

Introduction

Von Neumann Architecture

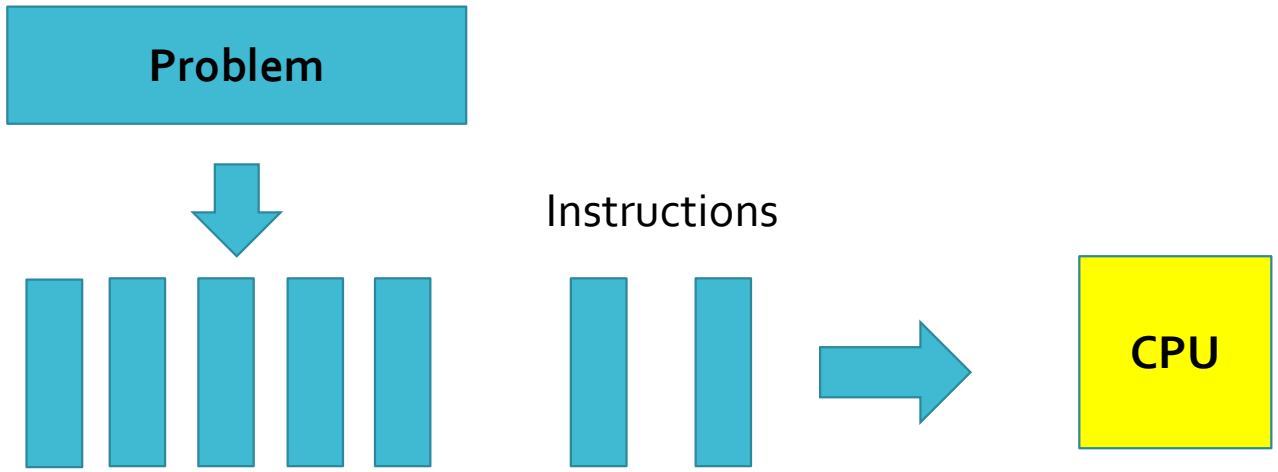
❖ **Von Neumann proposed computer comprised of four basic components:-**

1. Read/write, random access memory is used to store both program instructions and data.
2. Control unit fetches instructions/data from memory, decodes the instructions and then ***sequentially*** coordinates operations to accomplish the programmed task.
3. Arithmetic Unit performs basic arithmetic operations.
4. Input/Output is the interface to the human operator.



Serial Computing

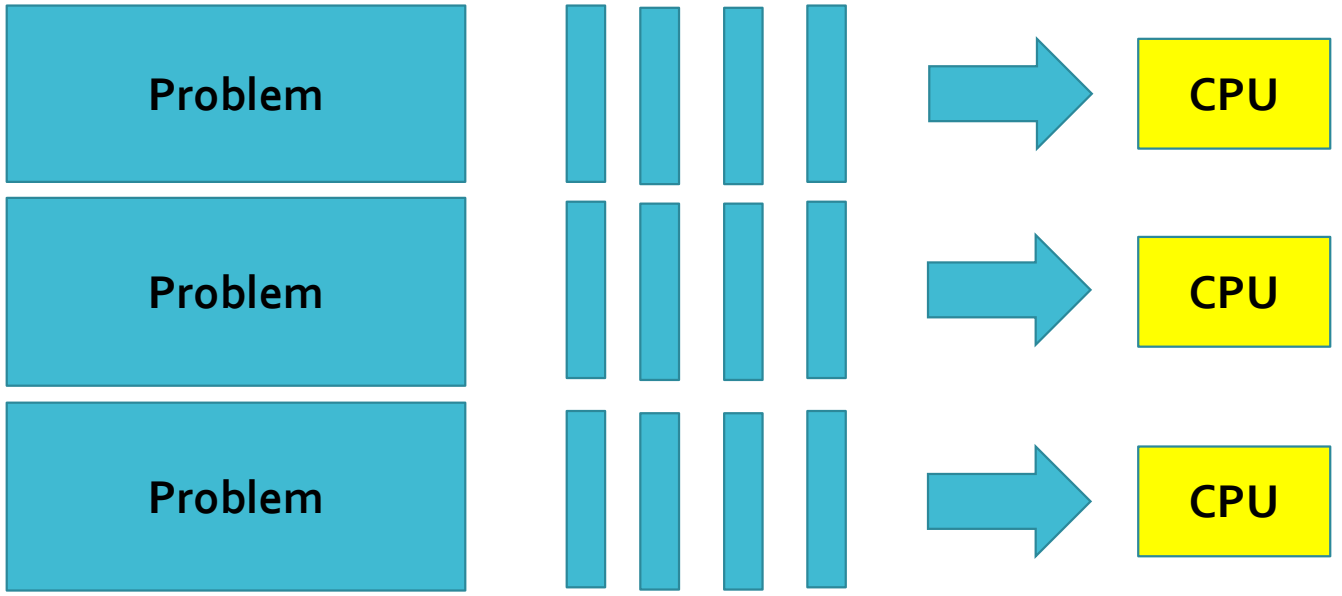
- To be run on a single computer having a single central processing unit (CPU).
- A problem is broken into discrete series of instructions.
- Instructions are executed one after another.
- Only one instruction may execute at any moment in time.



Parallel Computing

- Parallel computing is the simultaneous use of multiple computing resources to solve a computation problem.

1. To be run using multiple CPU's.
2. Problem is broken into discrete parts that can be solved con-currently.
3. Each part is further broken down to a series of instructions.
4. Instructions from each part execute simultaneously on different CPU's.



Parallel Computing Resources

- **The multiple computing resources can include:**

1. A single computer with multiple processors.
2. An arbitrary number of computers connected by a network.
3. A combination of both.

Characteristics of Computational Problem

- **The computational problem usually demonstrates characteristics such as the ability to be:**

1. Broken apart into discrete pieces of work that can be solved simultaneously.
2. Execute multiple program instructions at any moment in time.
3. Solved in less time using multiple computing resources than a single computing resource.

Why Parallel Computing?

- **Saves time and money**
  - Many resources working together will reduce time and cut potential costs.
- **Solve larger problems**
  - Web search engines/database processing millions of instructions per second.
- **Use of non-local resources**
- **Provide Concurrency**
  - Do multiple things at same time.

Where is Parallel Computing Used?

- **Scientific Computing**
  - Numerically intensive simulations
- **Database operations and information systems**
  - Web based business services, web search engines, online transaction processing
  - Client and inventory database management, Data mining, online transaction processing, Management information systems
  - Geographic information systems, seismic data processing
- **Artificial intelligence, Machine learning, Deep learning**
- **Real time systems and control applications**
  - Hardware and robotics control, pattern recognition

Parallel Processors

- Parallel processors are computer systems consisting of multiple processing units connected via some interconnection network plus the software needed to make the processing units together.

Level of Parallel Processing

- **We can have parallel processing at four levels:**
  - Instruction Level Parallelism
  - Loop Level Parallelism
  - Procedure Level Parallelism
  - Program Level Parallelism

Instruction Level parallelism

- Instruction-level parallelism (ILP) is a measure of how many of the instructions in a computer program can be executed simultaneously.
- ILP is the parallel execution of a sequence of instructions belonging to a specific thread of execution of a process.
- There are two approaches to instruction level parallelism: Hardware and Software.

Instruction Level Parallelism

- Hardware level works upon dynamic parallelism, whereas the software level works on static parallelism.
- Dynamic parallelism means the processor decides at run time which instructions to execute in parallel, whereas static parallelism means the compiler decides which instructions to execute in parallel.
- Consider the following instructions:
  - e = a +b
  - f = c + d
  - m = e*f
- If we assume that each operation can be completed in one unit of time then these three instructions can be completed in a total of two units of time, giving an ILP of 3/2.

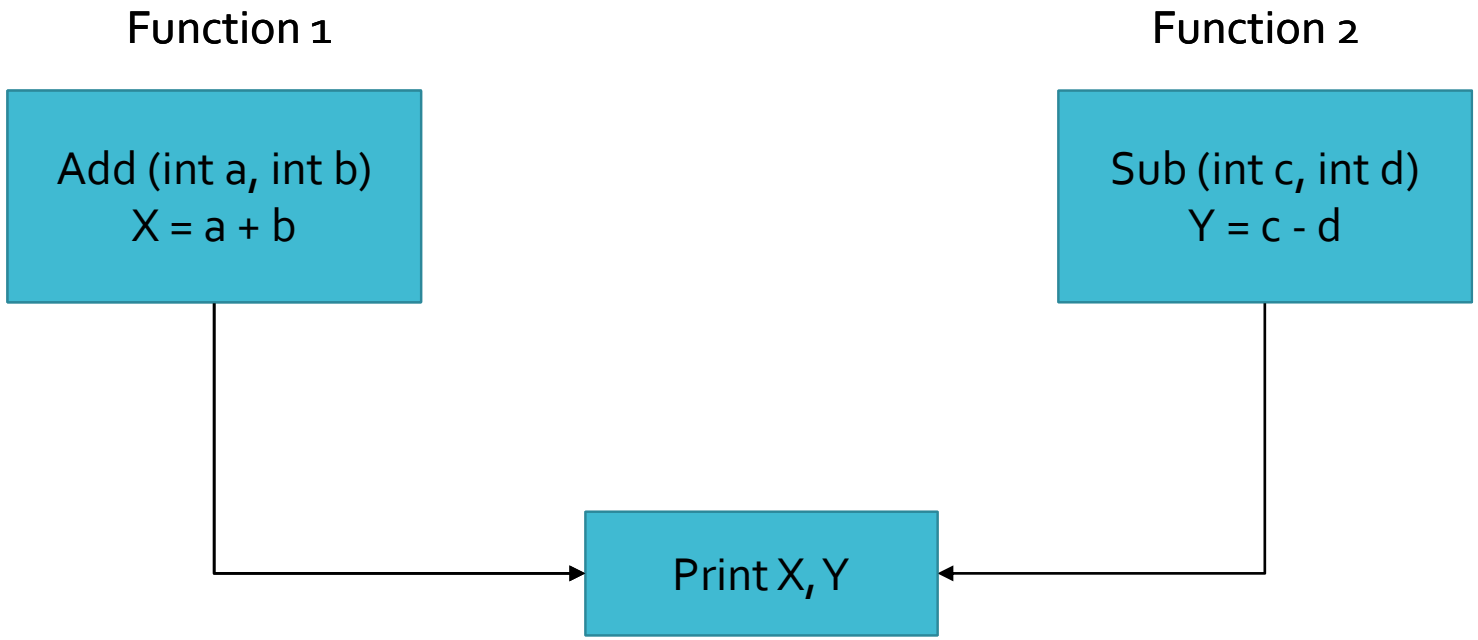
Loop Level Parallelism

- Loop level parallelism is a form of parallelism in software programming that is concerned with extracting parallel tasks from loops.
- Consider the following example:

```
for(i = o; i <= n; i++)
A(i) = B(i) + C(i)
```

Procedure Level Parallelism

- Procedure level parallelism is achieved by the parallel execution of different functions/tasks or procedures.



Program Level Parallelism

- This is the responsibility of the operating system which run processes concurrently.
- Different programs are obviously independent of each other.

Uniprocessor Architecture

- A typical uniprocessor computer consists of three major components:-
  1. Central processing unit (CPU)
     - Set of general purpose registers along with program counter.
     - A special purpose CPU status registers for storing the current state of CPU and program under execution.
     - One arithmetic logic unit.
     - One local cache memory.
  2. Main Memory
  3. Input-Output (I/O) subsystem
- In addition, there is a common synchronous bus architecture for communication between CPU, main memory and I/O sub system.
- Example: **Supermini VAX-11/980 Uniprocessor system**

Uniprocessor Architecture

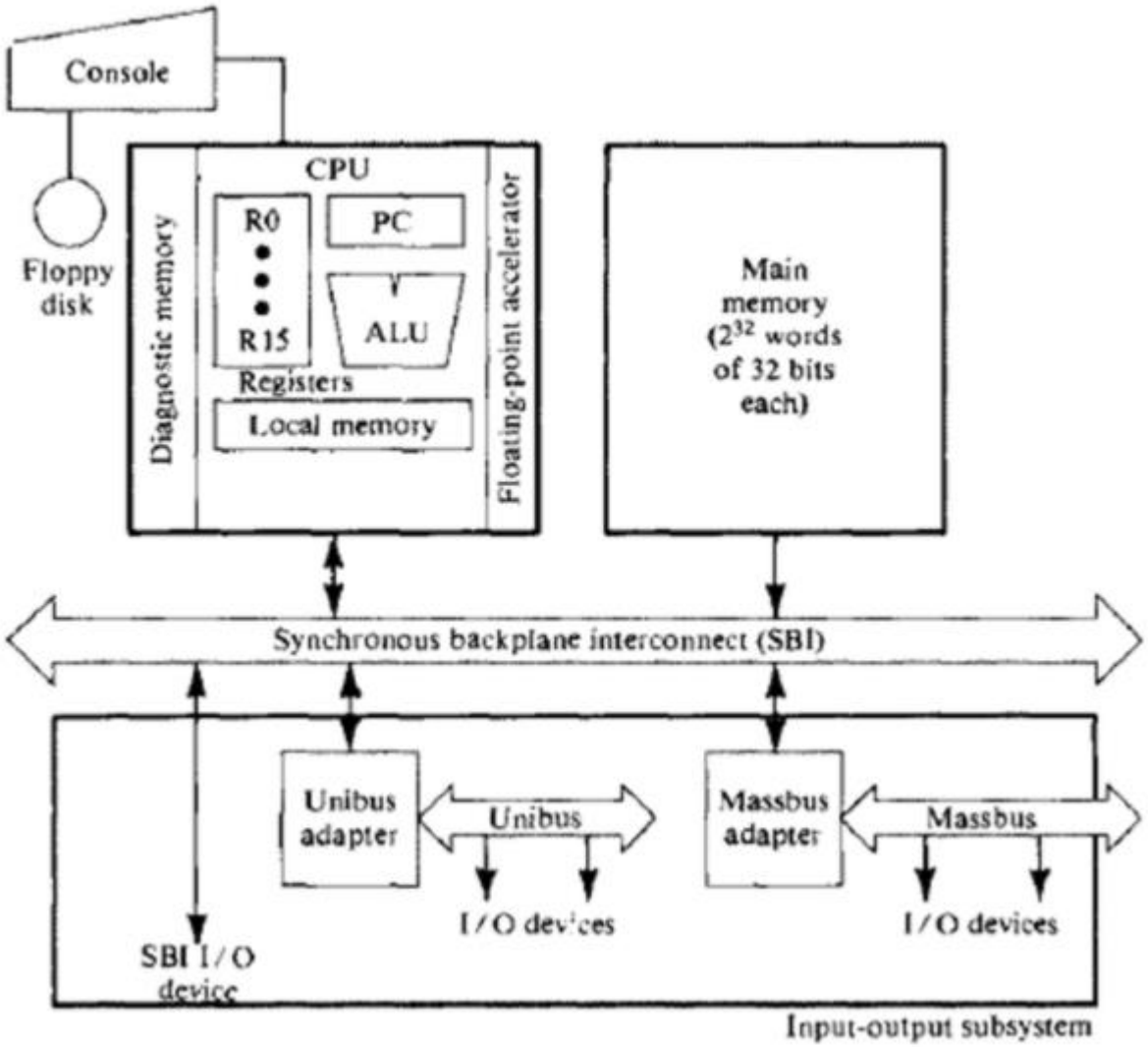


Figure: Architecture of Supermini VAX-11/980 Uniprocessor system

Uniprocessor Architecture

- The CPU contains the master controller of the VAX system.
- There are 16, 32-bit general purpose register one of which is a Program Counter (PC).There is also a special CPU status register containing about the current state of the processor being executed.
- The CPU contains an ALU with an optional Floating-point accelerator, and some local cache memory .
- The CPU can be intervened by the operator through the console connected to floppy disk.
- The CPU, the main memory(2^32 words of 32 bit each) and the I/O subsystem are all connected to a common bus, the synchronous backplane interconnection(SBI).
- Through this bus, all I/O devices can communicate with each other with CPU or with the memory.
- I/O devices can be connected directly to the SBI through the uni bus and its controller or through a mass bus and its controller.

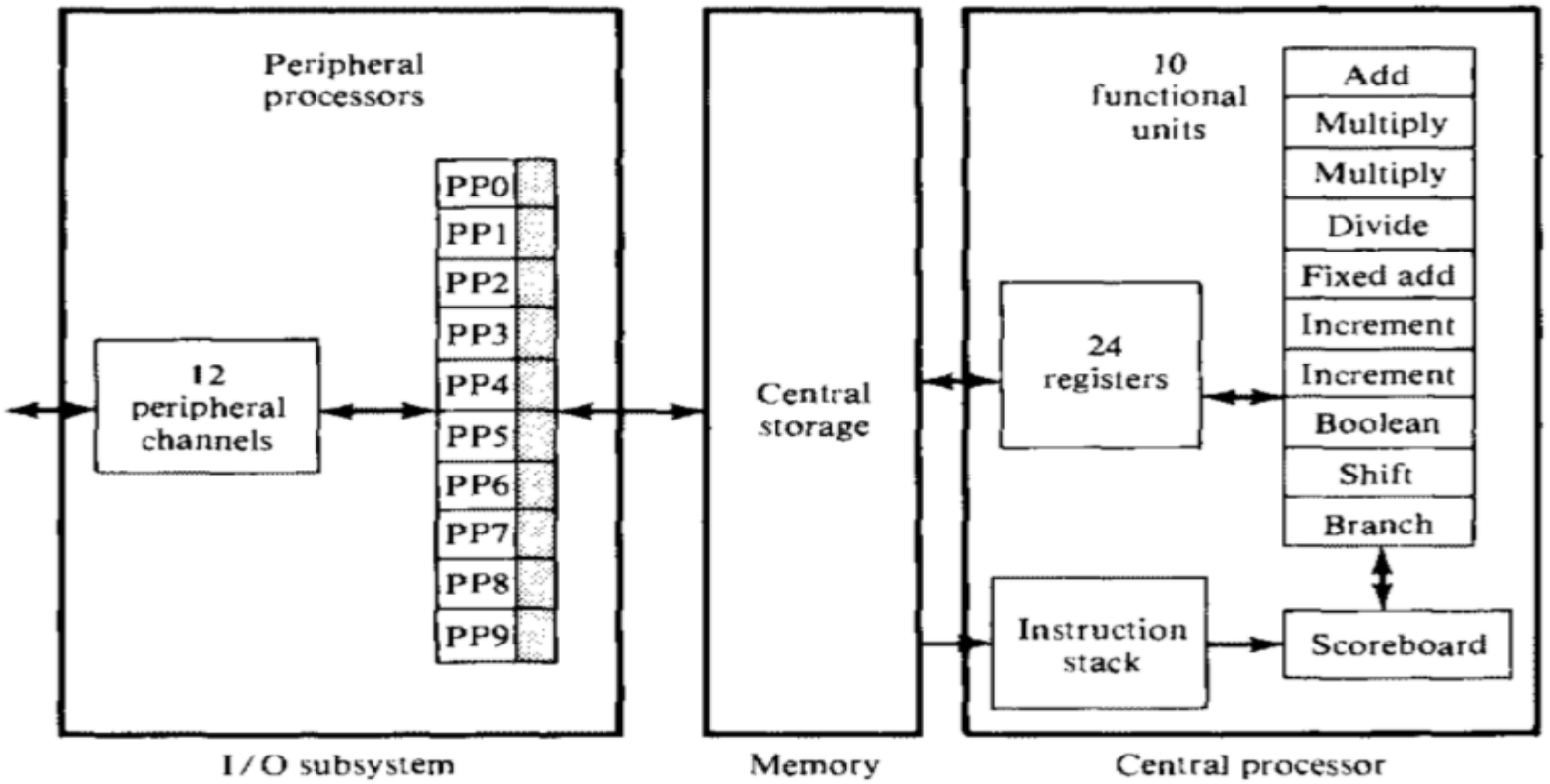
Parallelism in Uniprocessor Architecture

- Parallelism can be achieved in the uniprocessor systems using two main methods:-

1. **Hardware Method**
   - Multiplicity of functional units
   - Parallelism and pipelining within the CPU
   - Overlapped CPU and I/O operations
   - Use of hierarchical memory system
   - Balancing of subsystem bandwidth
2. **Software Method**
   - Multiprogramming
   - Time sharing

Multiplicity of Functional Units

- Many of the functions of the ALU can be distributed to multiple and specialized functional units which can operate in parallel.
- CDC-6600 (designed in 1964) has 10 functional units built into its CPU.
- These 10 units are independent of each other and may operate simultaneously.
- A scoreboard is used to keep track of the availability of the functional units and registers being demanded.



Parallelism and Pipelining within the CPU

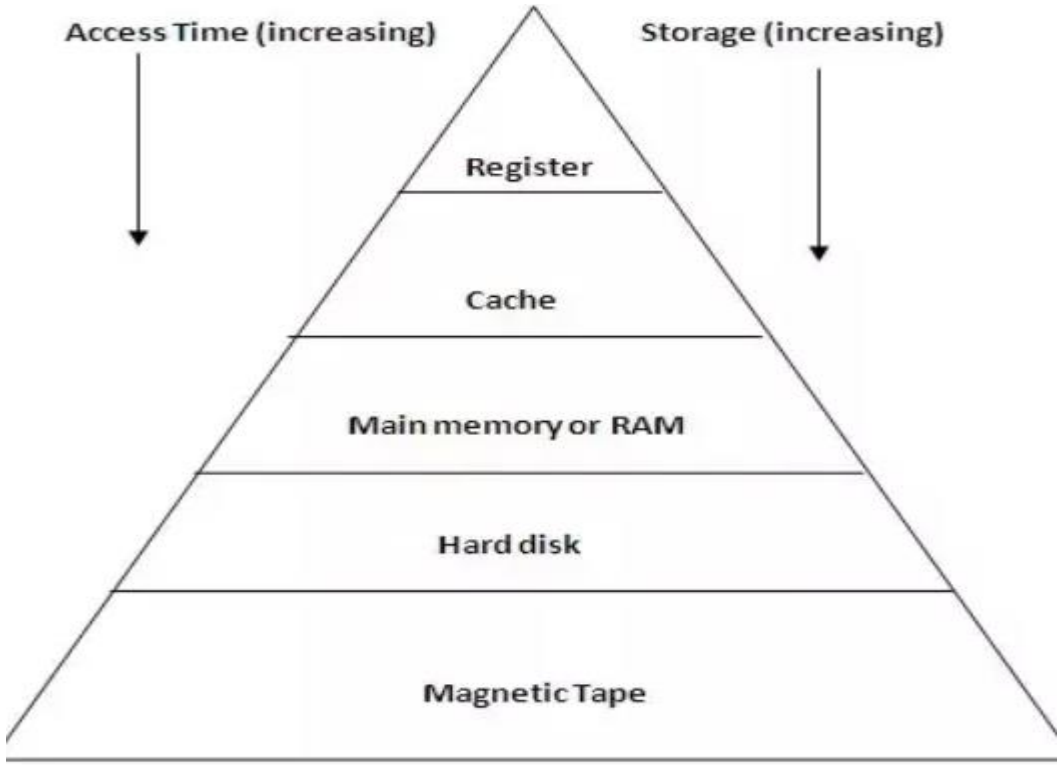
- CPU makes use of parallel adders like carry look ahead adder and carry save adders.
- Various phases of instruction executions arc now pipelined, including instruction fetch, decode, operand fetch, arithmetic logic execution, and store result.
- To facilitate overlapped instruction executions through the pipe, instruction prefetch and data buffering techniques have been developed.
- The instructions are prefetched and related data is buffered so that the instructions can be overlapped through pipes (memory queues).

Overlapped CPU and I/O operations

- I/O operations can be performed simultaneously with the CPU computations by using separate I/O controllers, channels, or I/O processors.
- The direct-memory-access (DMA) channel can be used to provide direct information transfer between the I/O devices and the main memory.

Use of hierarchical memory system

- A hierarchical memory system can be used to close up the speed gap between the CPU and memory.
- Memories are divided into different levels.



Use of hierarchical memory system

- The innermost level is the register files directly addressable by ALU.
- Cache memory can be used to serve as a buffer between the CPU and the main memory.
- Block access of the main memory can be achieved through multiway interleaving across parallel memory modules.
- Virtual memory space can be established with the use of disks and tape units at the outer levels.

Balancing of subsystem bandwidth

- Among 3 subsystem of computer: CPU, Main memory and I/O devices, CPU is the fastest unit and I/O devices are the slowest unit of the computer.
- The performance of this subsystem is measured in terms of bandwidth.
- The bandwidth of a system is defined as the no of operations are performed per unit time. In case of memory, the memory bandwidth is measured by the no of words that can be accessed per unit time.
- **Bandwidth balancing between CPU and Memory**
  - The speed gap between CPU and memory can be closed up by using cache memory between them.
- **Bandwidth balancing between memory and I/O devices**
  - To balance the speed gap of the I/O devices and the memory, I/O channels with different speeds are used between slow I/O devices and main memory.

Software Method for Parallelism

- **Multiprogramming**
  - Usually every process or program consists of either CPU bound instructions or I/O instructions or a combination of both.
  - When a system has many process then, while CPU is busy with some CPU bound process and at the same time a waiting I/O bound process can be allocated I/O resources for it's execution. This is called multiprogramming.
  - Here, processes don't have to wait for each other to complete and hence can execute simultaneously (parallel).

Software Method for Parallelism

- **Time Sharing**
  - In time sharing system, we assign time slices to each and every process and a preemptive strategy is automatically use to preempt a process when it's allocated time span is over.
  - Here, every process is assigned a specific time slot to utilize the CPU resources.
  - The preempted process goes into the waiting state and when given a chance by the scheduler are again assigned the CPU resources.
  - This happens till all the processes are finished.

Parallel Computer Memory Architectures

Shared Memory

General Characteristics

- Shared memory parallel computers have the ability for all processors to access all memory as global address space.
- Multiple processors can operate independently but share the same memory resources.
- Changes in a memory location effected by one processor are visible to all other processors.
- Shared memory machines have been classified as ***UMA*** and ***NUMA***, based upon memory access times.

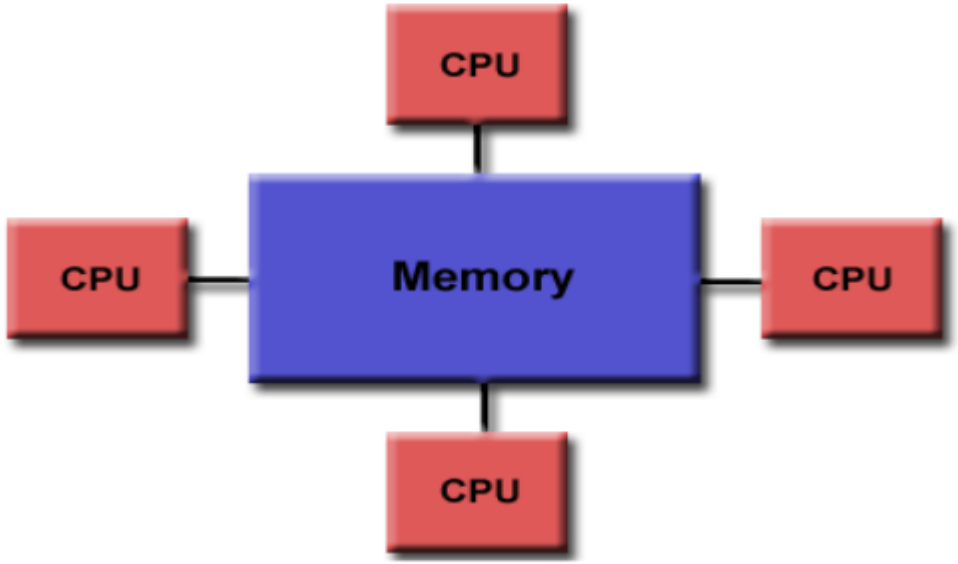
Disadvantages

- Primary disadvantage is the lack of scalability between memory and CPUs. Adding more CPUs can geometrically increases traffic on the shared memory-CPU path, and for cache coherent systems, geometrically increase traffic associated with cache/memory management.

Parallel Computer Memory Architectures

Uniform Memory Access (UMA)

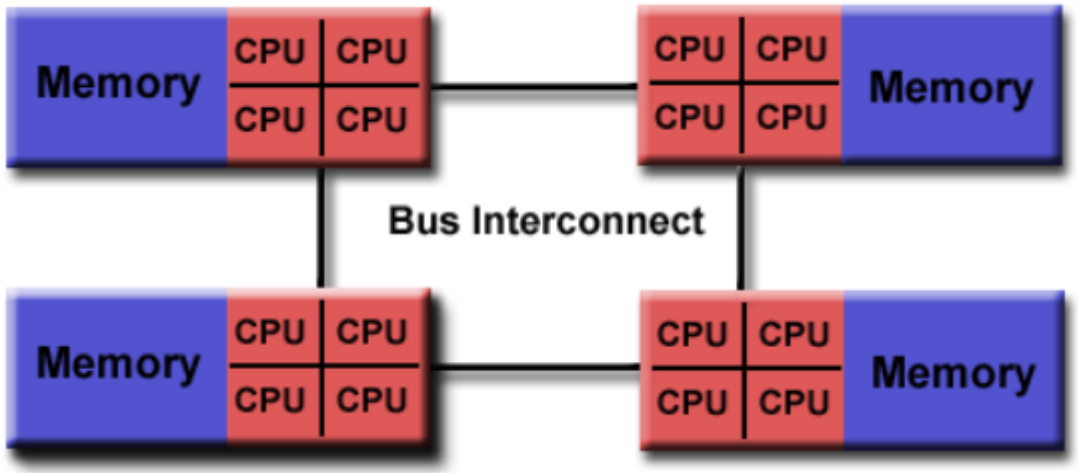
- Most commonly represented today by ***Symmetric Multiprocessor (SMP)*** machines.
- Identical processors.
- Equal access and access times to memory.
- Sometimes called CC-UMA - Cache Coherent UMA. Cache coherent means if one processor updates a location in shared memory, all the other processors know about the update. Cache coherency is accomplished at the hardware level.



Parallel Computer Memory Architectures

Non- Uniform Memory Access (UMA)

- Often made by physically linking two or more SMPs.
- One SMP can directly access memory of another SMP.
- Not all processors have equal access time to all memories.
- Memory access across link is slower.
- If cache coherency is maintained, then may also be called CC-NUMA - Cache Coherent NUMA.



Parallel Computer Memory Architectures

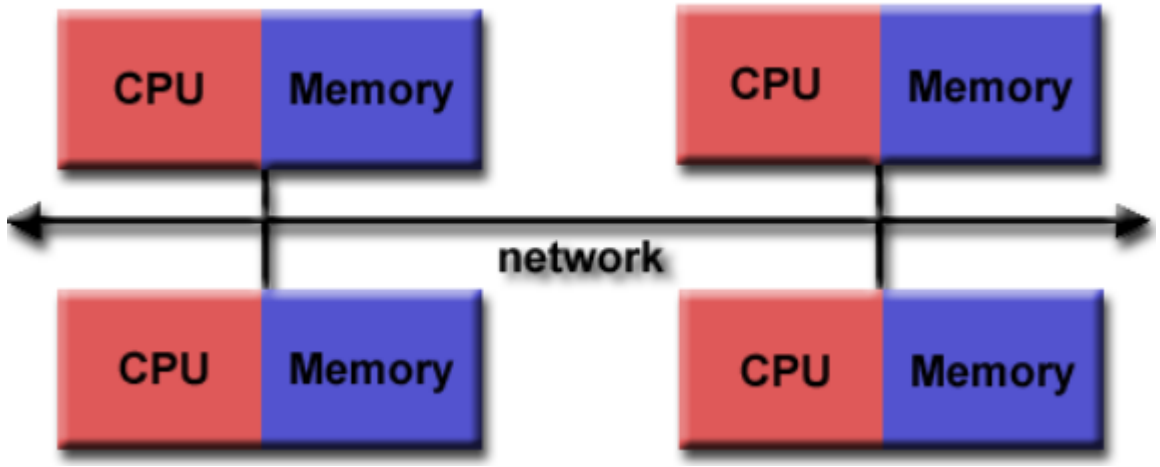
Distributed Memory

General Characteristics

- Distributed memory systems require a communication network to connect inter-processor memory.
- Processors have their own local memory. Memory addresses in one processor do not map to another processor, so there is no concept of global address space across all processors.
- Because each processor has its own local memory, it operates independently. Changes it makes to its local memory have no effect on the memory of other processors. Hence, the concept of cache coherency does not apply.
- When a processor needs access to data in another processor, it is usually the task of the programmer to explicitly define how and when data is communicated. Synchronization between tasks is likewise the programmer's responsibility.

Parallel Computer Memory Architectures

- **Distributed Memory**
  - **Advantages**
    - Memory is scalable with the number of processors. Increase the number of processors and the size of memory increases proportionately.
    - Each processor can rapidly access its own memory without interference.
  - **Disadvantages**
    - The programmer is responsible for many of the details associated with data communication between processors.



Parallel Computer Memory Architectures

Hybrid Shared-Distributed Memory

General Characteristics

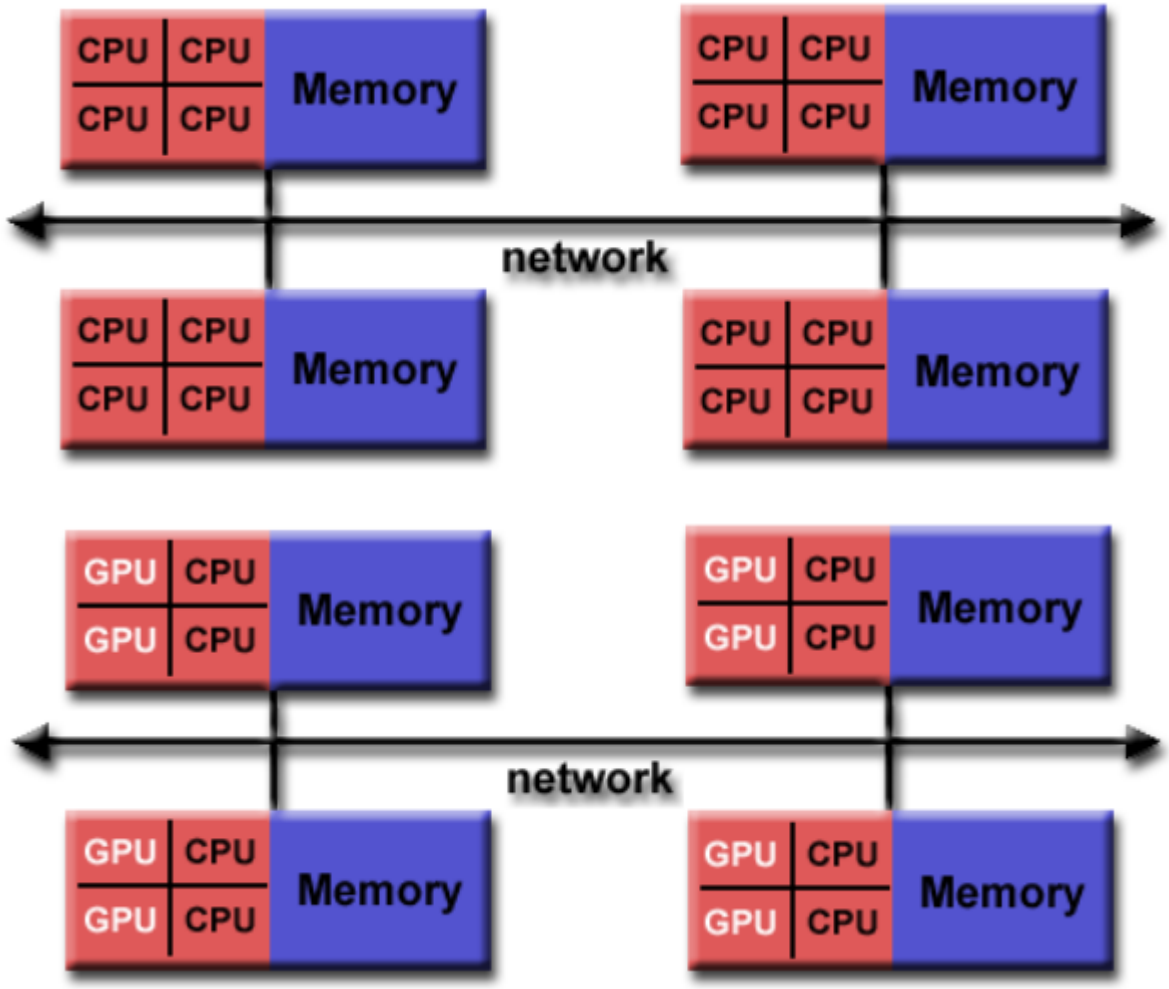
- The shared memory component can be a shared memory machine and/or graphics processing units (GPU).
- The distributed memory component is the networking of multiple shared memory/GPU machines, which know only about their own memory - not the memory on another machine. Therefore, network communications are required to move data from one machine to another.

Advantages and Disadvantages

- Increased scalability is an important advantage.
- Increased programmer complexity is an important disadvantage.

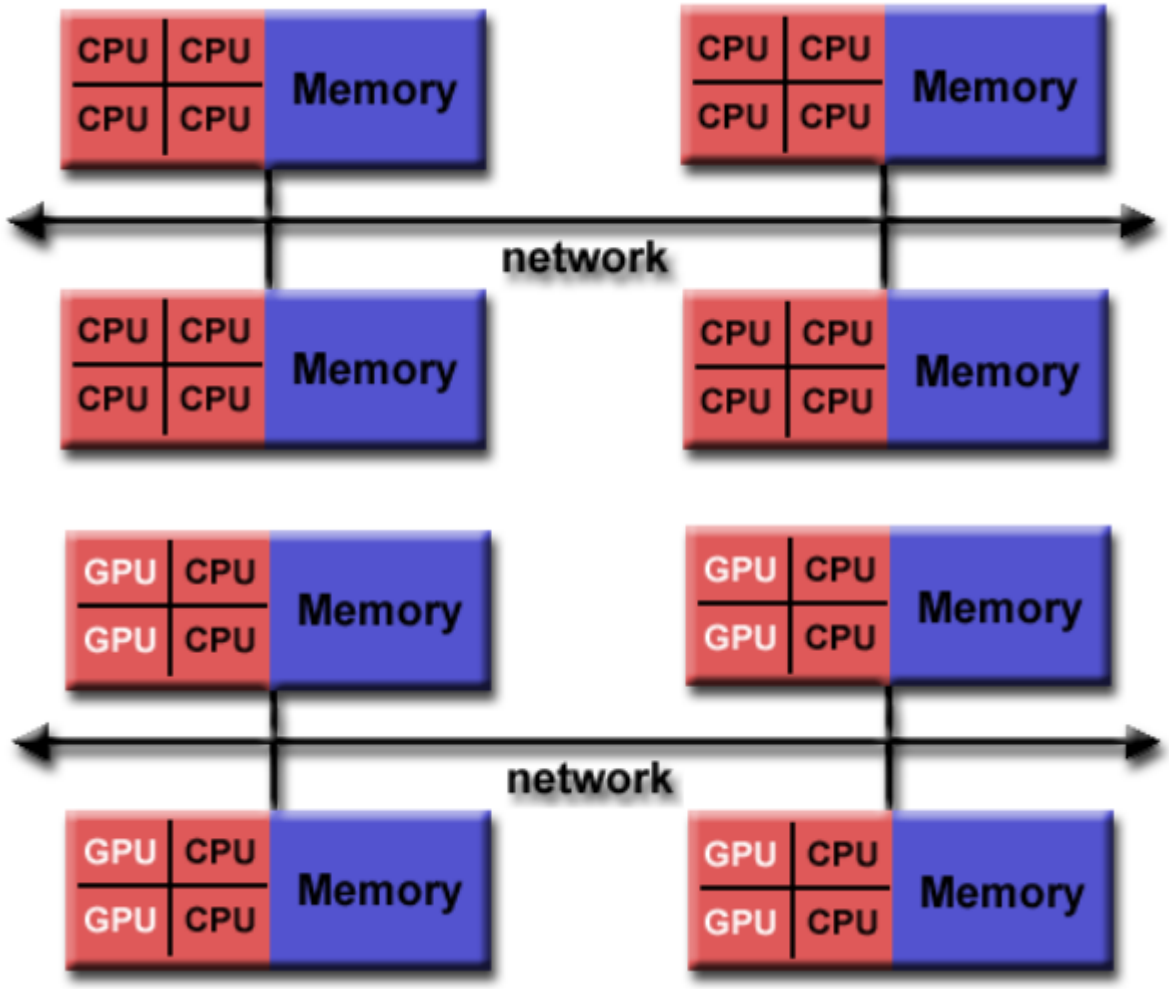
Parallel Computer Memory Architectures

- **Hybrid Shared-Distributed Memory**



Parallel Computer Memory Architectures

- **Hybrid Shared-Distributed Memory**



Parallel Programming Model

- Parallel programming model exists an abstraction above hardware and memory architectures.

- **Shared Memory Model**
  - Multiple processes share common memory space

- **Distributed Memory Model**
  - The user makes call to libraries to explicitly share information between processors.

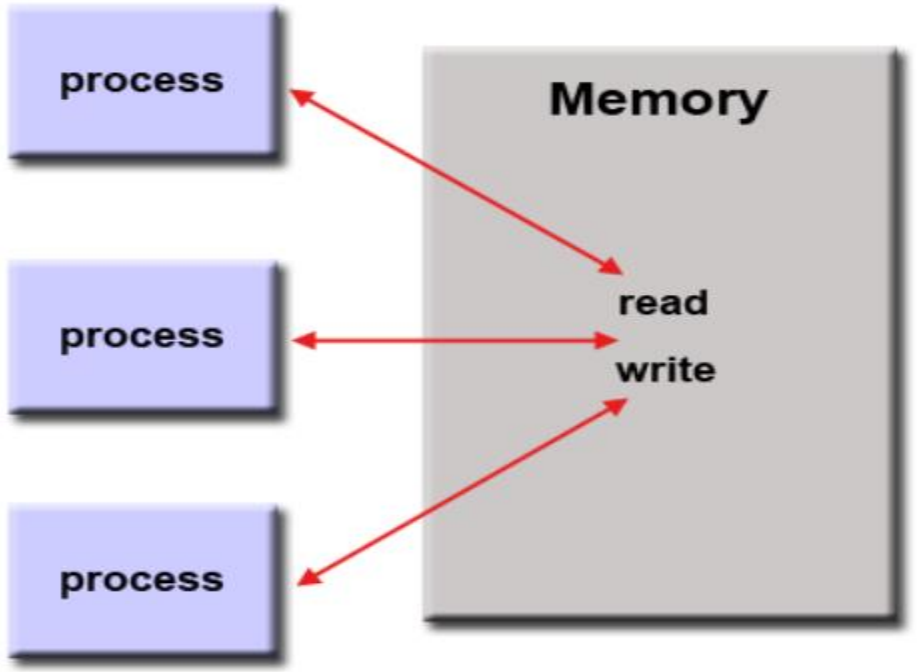
- **Threads Model**
  - A single process having multiple (concurrent) execution paths

- **Data Parallel Model**
  - Data partitioning determines parallelism

Parallel Programming Model

❖Shared Memory Model (Without Thread)

- In this programming model, processes/tasks share a common address space, which they read and write to asynchronously.
- Various mechanisms such as locks / semaphores are used to control access to the shared memory, resolve contentions and to prevent race conditions and deadlocks.
- This is perhaps the simplest parallel programming model.



Parallel Programming Model

❖ **Advantages of shared memory model**

- Data "ownership" is lacking, so there is no need to specify explicitly the communication of data between tasks.
- All processes see and have equal access to shared memory.
- Program development can often be simplified.

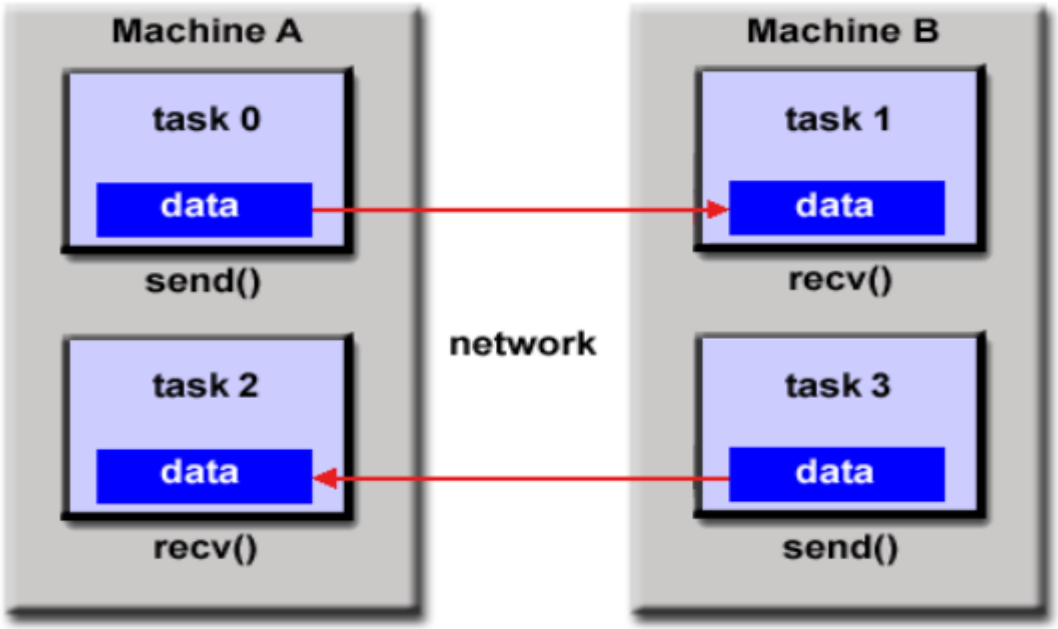
❖ **Disadvantages of shared memory model**

- Different techniques need to use to block the access of the same data among different processors.

Parallel Programming Model

❖ Distributed Memory Model

- A set of tasks that use their own local memory during computation.
- Multiple tasks can reside on the same physical machine and/or across an arbitrary number of machines.
- Tasks exchange data through communications by sending and receiving messages.
- Data transfer usually requires cooperative operations to be performed by each process. For example, a send operation must have a matching receive operation.



Parallel Programming Model

❖ **Advantages of Distributed memory model**

- Data can be stored anywhere.
- Communication between processes are simple.

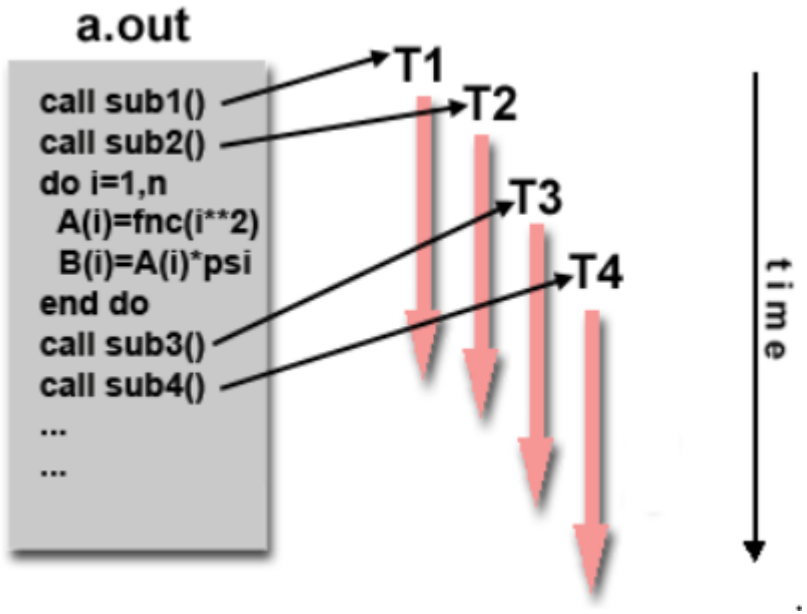
❖ **Disadvantages of Distributed memory model**

- If any dependent process is quit or stop then the other processes will stop working.

Parallel Programming Model

❖ Thread Model

- This programming model is a type of shared memory programming.
- In the threads model of parallel programming, a single "heavy weight" process can have multiple "light weight", concurrent execution paths called threads.
- Each thread has local data, but also, shares the common data.
- Threads communicate with each other through global memory (updating address locations).
- This requires synchronization constructs to ensure that more than one thread is not updating the same global address at any time.



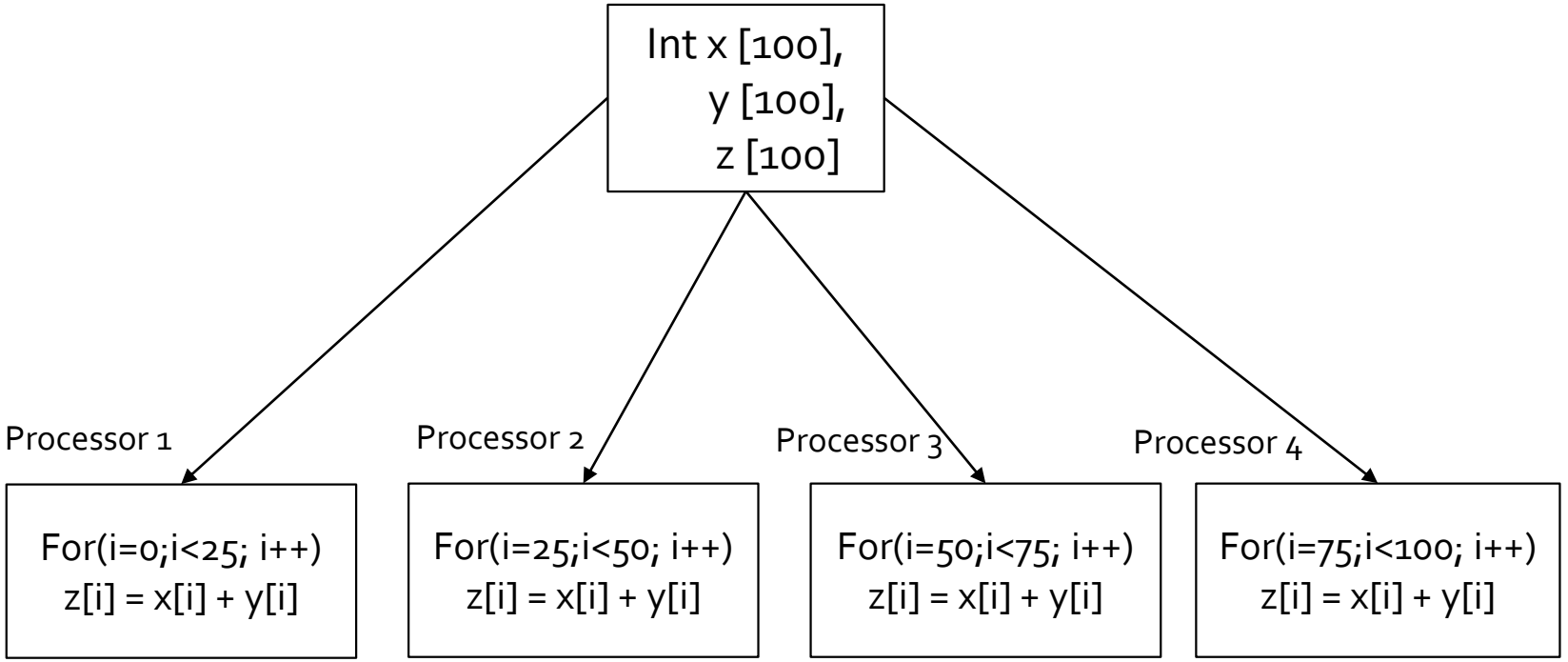
Parallel Programming Model

❖Data Parallel Model

- May also be referred to as the **Partitioned Global Address Space (PGAS)** model.
- The data parallel model demonstrates the following characteristics:
  - Address space is treated globally
  - Most of the parallel work focuses on performing operations on a data set. The data set is typically organized into a common structure, such as an array or cube.
  - A set of tasks work collectively on the same data structure, however, each task works on a different partition of the same data structure.
- On shared memory architectures, all tasks may have access to the data structure through global memory.
- On distributed memory architectures, the global data structure can be split up logically and/or physically across tasks.

Parallel Programming Model

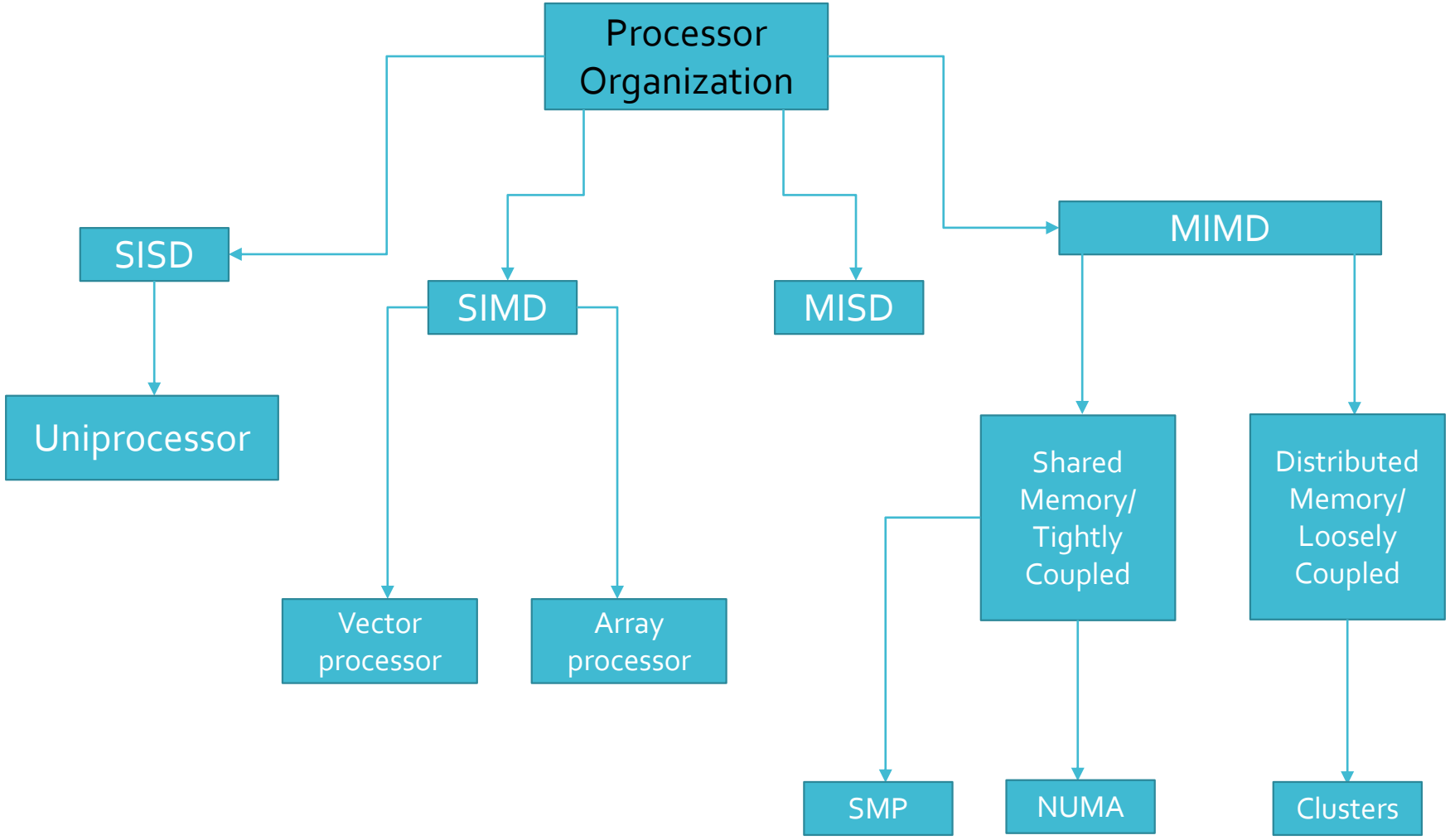
❖ **Example of Data Parallel Model**



Flynn's Taxonomy

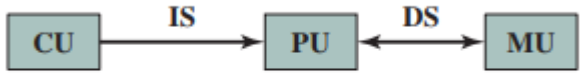
- It is based on the notion of a stream of information. Two types of information flow into a processor.
    1. **Instruction**
    2. **Data**
- The instruction stream is defined as the sequence of instructions executed by the processing unit.
- The data stream is defined as the sequence of data including inputs, outputs, temporary results called by the instruction stream.

Flynn's Taxonomy



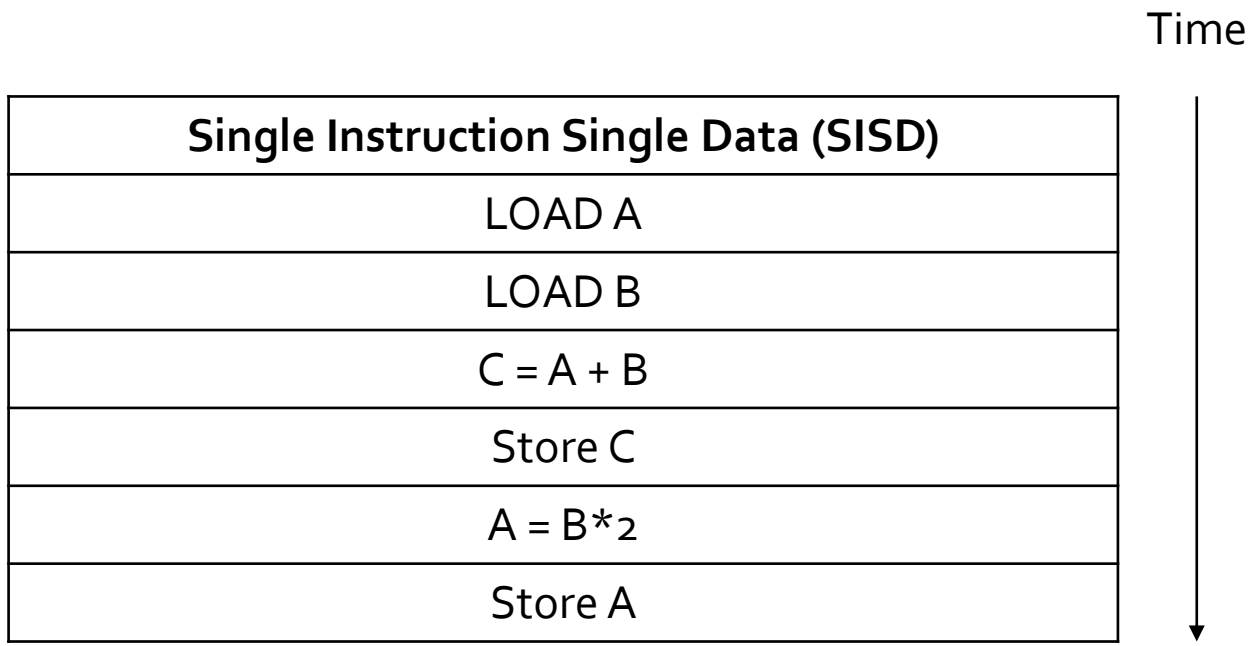
SISD

- SISD means Single Instruction Single Data Stream is a uni-processor machine which is capable of executing a single instruction, operating on a single data stream.
- **Single instruction:** Only one instruction stream is being acted on by the CPU during one clock cycle.
- **Single Data:** Only one data stream is being used as input any one clock cycle.
- Example: CDC- 6600, VAX-11, IBM 7001 are SISD computers.



(a) SISD

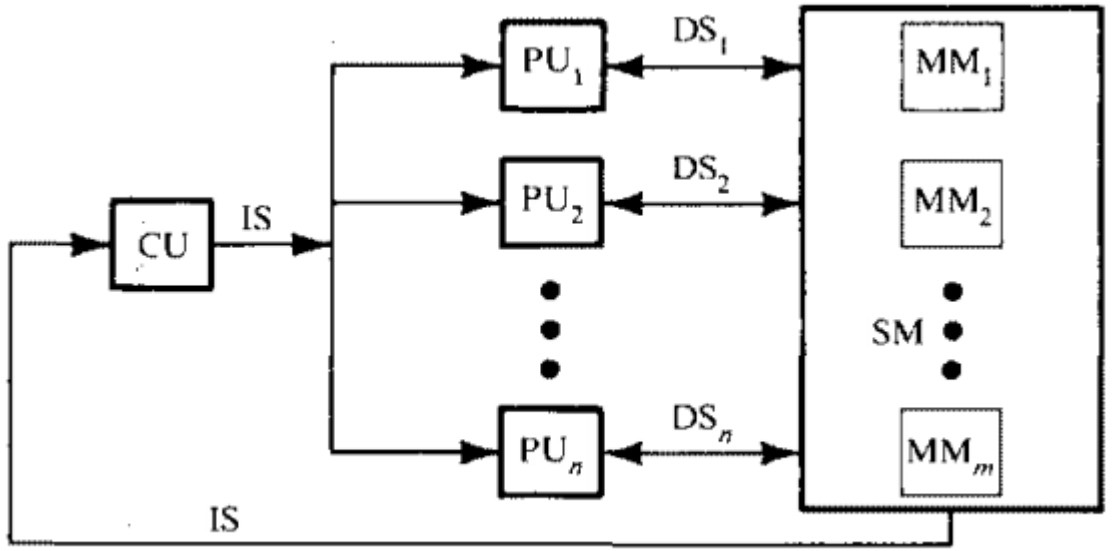
SISD


Time
Single Instruction Single Data (SISD)
LOAD A
LOAD B
C = A + B
Store C
A = B*2
Store A

SIMD

- SIMD means Single Instruction Multiple Data Stream is capable of executing same instructions on all CPU's but operating on different data streams.
- **Single instruction:** All processing units execute the same instruction at any given clock cycle.
- **Multiple Data:** Each processing unit is capable of working on multiple data element.
- SIMD computer has single control unit which issues one instruction at a time but it has multiple ALU's or processing units to carry out on multiple data streams simultaneously.
- **Examples:**
  - **Array processor:** ILLIAC-IV, MPP
  - **Vector processor:** IBM 9000, Cray X-MP, Y-MP, C 90

SIMD



(b) SIMD computer

Distributed memory

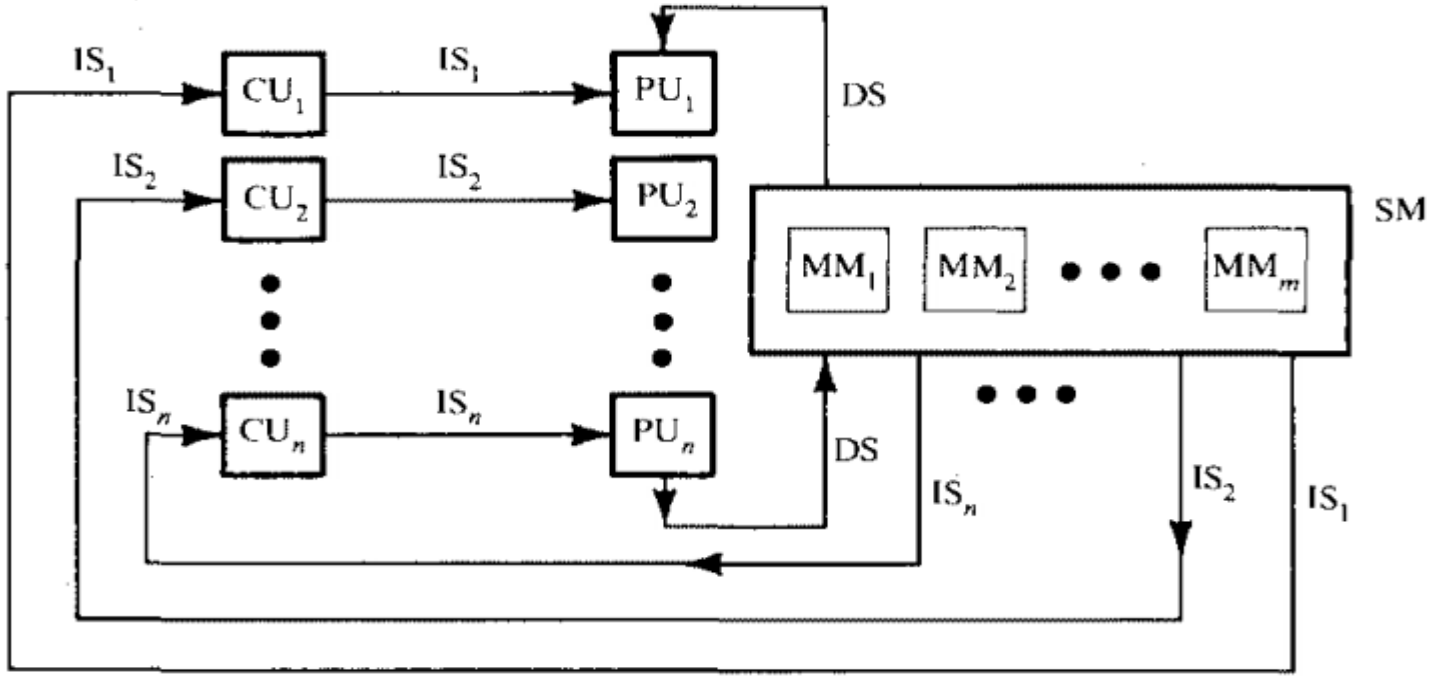
Time

Single Instruction Multiple Data (SIMD)		
LOAD A(1)	LOAD A(2)	LOAD A(n)
LOAD B(1)	LOAD B(2)	LOAD B(n)
C(1) = A(1) + B(1)	C(2) = A(2) + B(2)	C(n) = A(n) + B(n)
Store C(1)	Store C(2)	Store C(n)

MISD

- MISD means Multiple Instruction Single Data Stream is capable of executing different instructions on different PU's but all of them operating on single data stream.
- **Multiple instruction:** Each processing unit may execute different instruction stream.
- **Single Data:** Each processing unit is capable of working on single data element.
- SIMD computer has single control unit which issues one instruction at a time but it has multiple ALU's or processing units to carry out on multiple data streams simultaneously.
- **Examples:** Systolic arrays.

MISD



(c) MISD computer

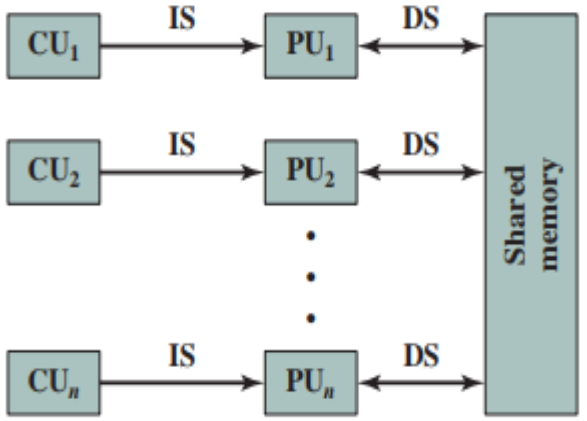
Time

Multiple Instruction Single Data (MISD)		
LOAD A(1)	LOAD A(1)	LOAD A(1)
C(1) = A(1)*2	C(2) = A(1)*2	C(n) = A(1)*n
Store C(1)	Store C(2)	Store C(n)

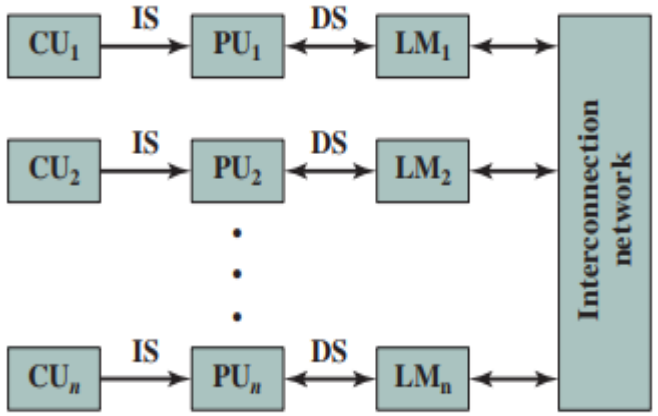
MIMD

- MIMD means Multiple Instruction Multiple Data Stream is capable of executing multiple instructions on multiple data streams.
- **Multiple instruction:** Each processing unit may execute different instruction stream.
- **Multiple Data:** Each processing unit is capable of working on multiple data element.
- MIMD systems are parallel computers capable of processing several program simultaneously.
- Can be categorized as tightly coupled and loosely coupled based on shared and distributed memory.
- **Examples:** IBM-370, Cray-2, Cray X-MP, UNIVAC-1100/80

MIMD



(c) MIMD (with shared memory)



(d) MIMD (with distributed memory)

Time

Multiple Instruction Single Data (MISD)		
LOAD A(1)	LOAD A(2)	LOAD A(n)
C(1) = A(1)*2	C(2) = A(2)*2	C(n) = A(n)*n
Store C(1)	Store C(2)	Store C(n)

Symmetric Multiprocessor (SMP)

- If the processors share a common memory, then each processor accesses programs and data stored in the shared memory, and processors communicate with each other via that memory.
- The most common form of such system is known as a **symmetric multiprocessor (SMP).**

Characteristics of Symmetric Multiprocessor (SMP)

1. There are two or more similar processors of comparable capability.
2. These processors share the same main memory and I/O facilities and are interconnected by a bus or other internal connection scheme, such that memory access time is approximately the same for each processor.
3. All processors share access to I/O devices, either through the same channels or through different channels that provide paths to the same device.
4. All processors can perform the same functions (hence the term *symmetric*).
5. The system is controlled by an integrated operating system that provides interaction between processors and their programs at the job, task, file, and data element levels.

Advantages of Symmetric Multiprocessor (SMP)

- Advantages of SMP over uniprocessor system:-

1. **Performance:** If the work to be done by a computer can be organized so that some portions of the work can be done in parallel, then a system with multiple processors will yield greater performance than one with a single processor of the same type.
2. **Availability:** In a symmetric multiprocessor, because all processors can perform the same functions, the failure of a single processor does not halt the machine. Instead, the system can continue to function at reduced performance.
3. **Incremental growth:** A user can enhance the performance of a system by adding an additional processor.
4. **Scaling:** Vendors can offer a range of products with different price and performance characteristics based on the number of processors configured in the system.

Tightly Coupled Multiprocessor

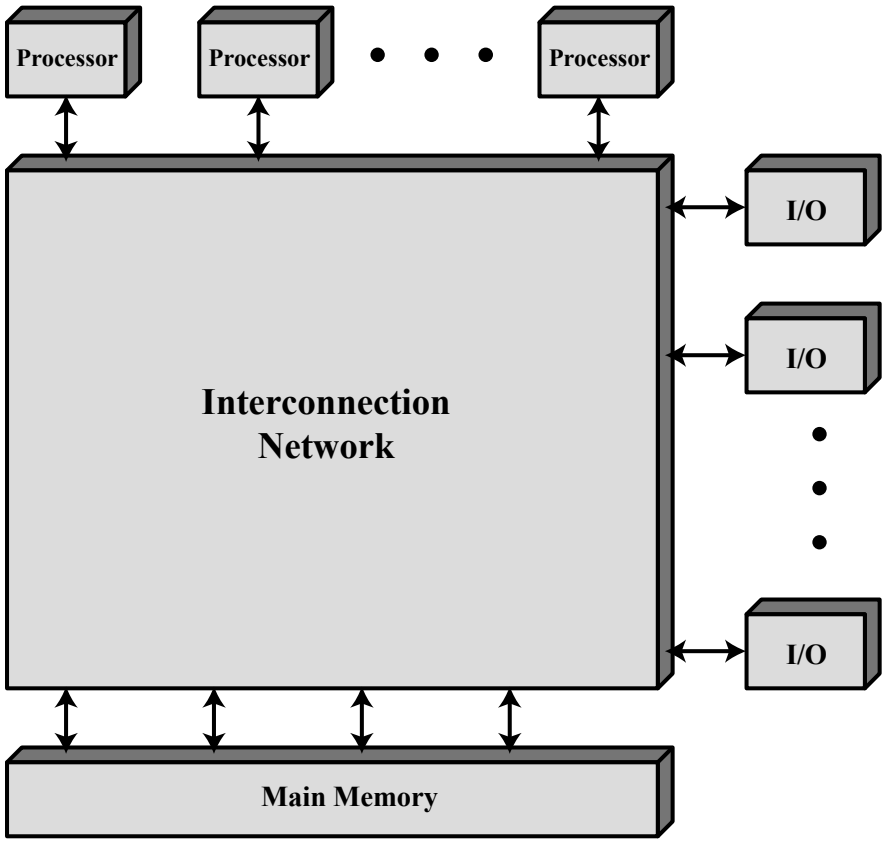


Figure 17.4 Generic Block Diagram of a Tightly Coupled Multiprocessor

Time Shared Bus

- The time-shared bus is the simplest mechanism for constructing a multiprocessor system.
- To facilitate DMA transfers from I/O subsystems to processors, the following features are provided:
- **Features of Time Shared Bus:-**
  - **Addressing:** It must be possible to distinguish modules on the bus to determine the source and destination of data.
  - **Arbitration:** Any I/O module can temporarily function as "master." A mechanism is provided to arbitrate competing requests for bus control, using some sort of priority scheme.
  - **Time-sharing:** When one module is controlling the bus, other modules are locked out and must, if necessary, suspend operation until bus access is achieved.

SMP Organization

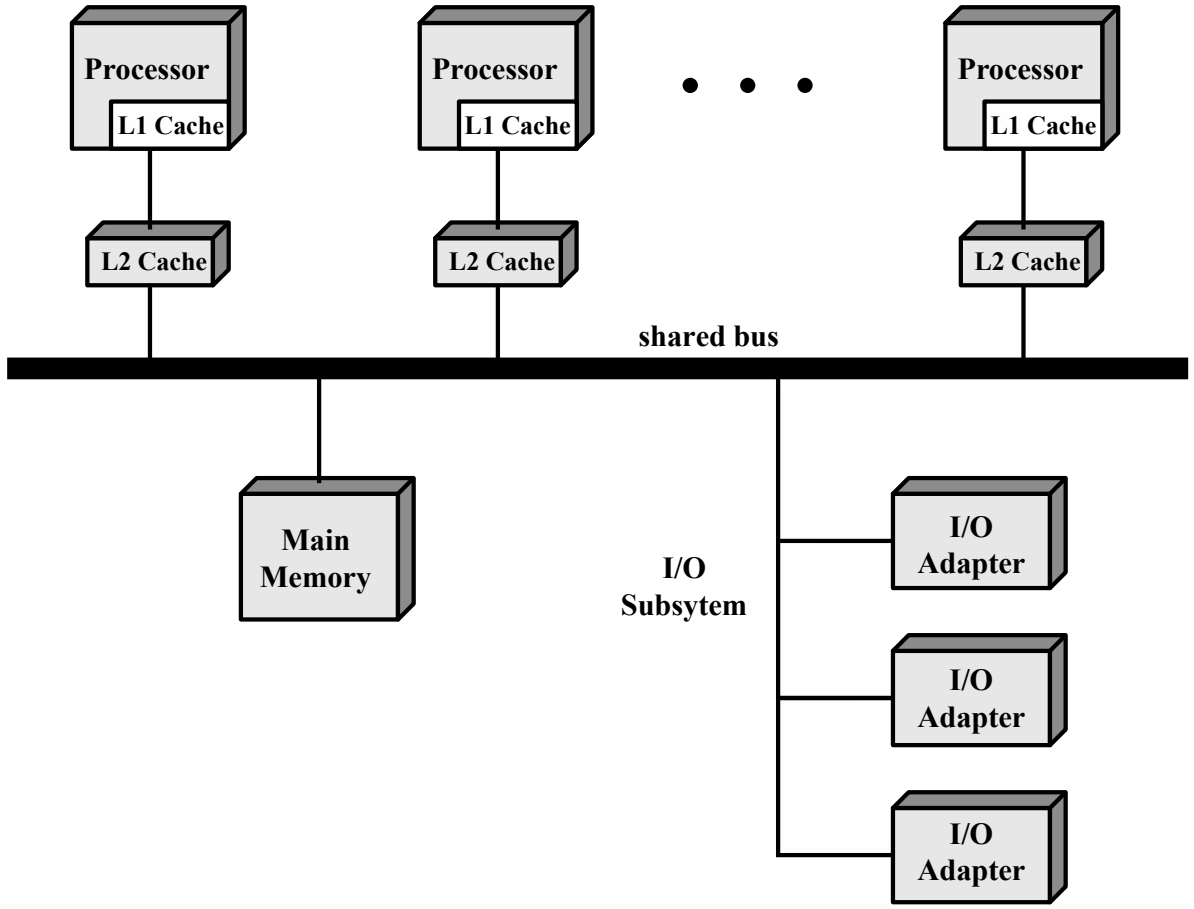


Figure 17.5 Symmetric Multiprocessor Organization

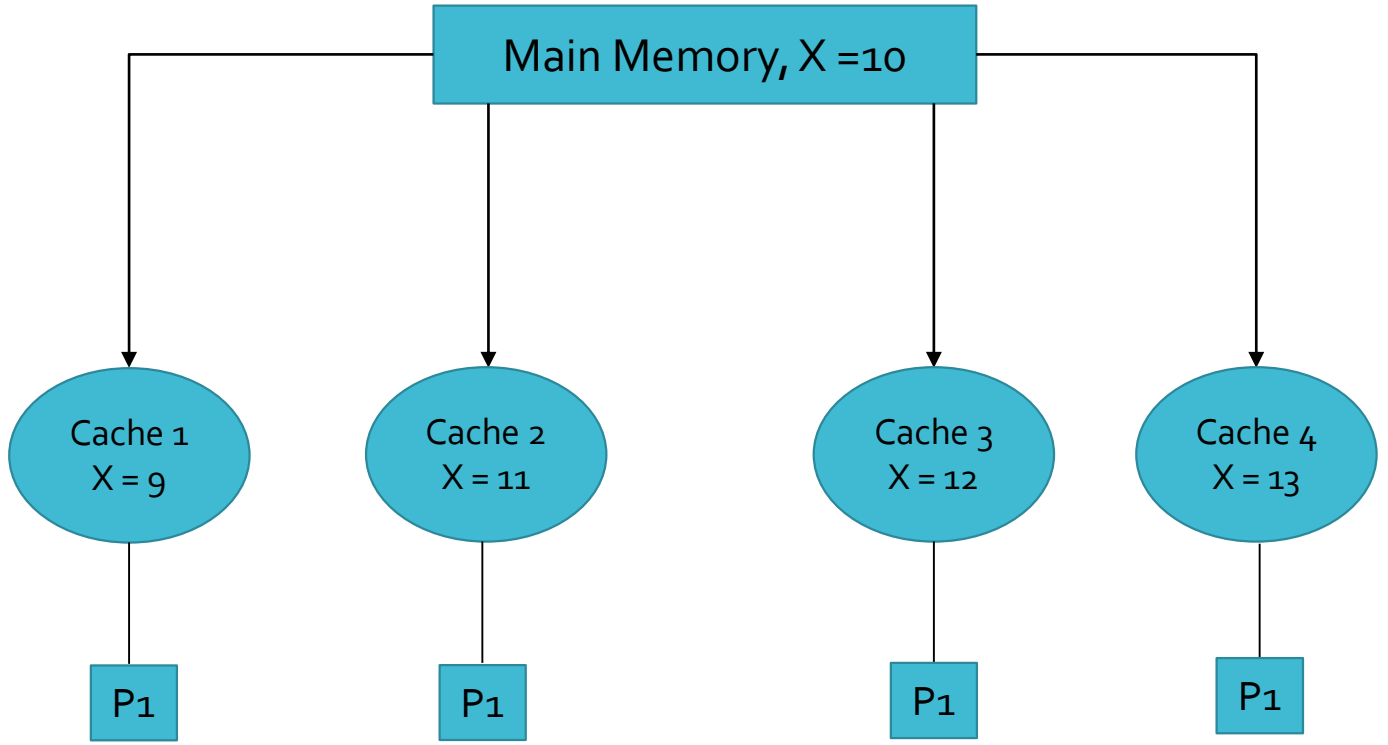
Advantages of Time Shared Bus

- **The bus organization has several attractive features:**
  - **Simplicity:** This is the simplest approach to multiprocessor organization. The physical interface and the addressing, arbitration, and time-sharing logic of each processor remain the same as in a single-processor system.
  - **Flexibility:** It is generally easy to expand the system by attaching more processors to the bus.
  - **Reliability:** The bus is essentially a passive medium, and the failure of any attached device should not cause failure of the whole system.

Disadvantages of Time Shared Bus

- The main drawback to the bus organization is performance.
- All memory references pass through the common bus.
- Thus, the bus cycle time limits the speed of the system.
- To improve performance, it is desirable to equip each processor with a cache memory. This also introduce a new problem called cache coherence problem.
- This should reduce the number of bus accesses dramatically.

Cache Coherence Problem



**None of the processor perform the write operation in the shared memory so, finally in the shared memory X = 10 and in the local caches, X = 9, 11, 12, 13 therefore inconsistent view of memory. This is called cache coherence problem.

Multiprocessor OS Design Considerations

- **Simultaneous concurrent processes**
  - OS routines need to be reentrant to allow several processors to execute the same IS code simultaneously
  - OS tables and management structures must be managed properly to avoid deadlock or invalid operations
- **Scheduling**
  - Any processor may perform scheduling so conflicts must be avoided
  - Scheduler must assign ready processes to available processors
- **Synchronization**
  - With multiple active processes having potential access to shared address spaces or I/O resources, care must be taken to provide effective synchronization
  - Synchronization is a facility that enforces mutual exclusion and event ordering
- **Memory management**
  - In addition to dealing with all of the issues found on uniprocessor machines, the OS needs to exploit the available hardware parallelism to achieve the best performance
  - Paging mechanisms on different processors must be coordinated to enforce consistency when several processors share a page or segment and to decide on page replacement
- **Reliability and fault tolerance**
  - OS should provide graceful degradation in the face of processor failure
  - Scheduler and other portions of the operating system must recognize the loss of a processor and restructure accordingly

Cache Coherence Problem

- Multiple copies of the same data can exist in different caches simultaneously, and if processors are allowed to update their own copies freely, an inconsistent view of memory can result.

- Cache coherence refers to the problem of keeping the data in these caches consistent. The main problem is dealing with writes by a processor. **There are two general strategies for dealing with writes to a cache:**

    1. **Write-through** - all data written to the cache is also written to memory at the same time. It also responsible for cache coherence problem.
    2. **Write-back** - when data is written to a cache, a dirty bit is set for the affected block. The modified block is written to memory only when the block is replaced. It is clear that a write-back policy can result in inconsistency.

Solution to Cache Coherence Problem

❖ **There are two solutions of cache coherence problem:-**

1. **Software Solution**
   - In software approach, the detecting of potential cache coherence problem is transferred from run time to compile time.
   - In compile time; software approaches generally make conservative decisions. Leading to inefficient cache utilization.
   - Compiler-based cache coherence mechanism perform an analysis on the code to determine which data items may become unsafe for caching, and they mark those items accordingly. So, there are some more cacheable items, and the operating system or hardware does not cache those items.
   - The simplest approach is to prevent any shared data variables from being cached.
   - More efficient approaches analyze the code to determine safe periods for shared variables. The compiler then inserts instructions into the generated code to enforce cache coherence during the critical periods.
2. **Hardware Solution**

Solution to Cache Coherence Problem

❖ **Hardware Solution**

- Hardware solution provide dynamic recognition at run time of potential inconsistency conditions. Because the problem is only dealt with when it actually arises, there is more effective use of caches, leading to improved performances over a software approach.
- Hardware schemes can be divided into two categories:
  1. **directory protocol**
  2. **snoopy protocols**

Solution to Cache Coherence Problem

❖ Directory Based Protocol

- Directory protocols collect and maintain information about where copies of lines reside. Typically, there is centralized controller that is part of the main memory controller, and a directory that is stored in main memory.
- The directory contains global state information about the contents of the various local caches.
- When an individual cache controller makes a request, the centralized controller checks and issues necessary commands for data transfer between memory and caches or between caches themselves.
- It is also responsible for keeping the state information up to date, therefore, every local action that can effect the global state of a line must be reported to the central controller.
- The controller maintains information about which processors have a copy of which lines.
- Before a processor can write to a local copy of a line, it must request exclusive access to the line from the controller.

Solution to Cache Coherence Problem

- **Directory Based Protocol**
  - Before granting thus exclusive access, the controller sends a message to all processors with a cached copy of this time, forcing each processors to invalidate its copy.
  - After receiving acknowledgement back from each such processor, the controller grants exclusive access to the requesting processor.
  - When another processor tries to read a line that is exclusively granted to another processors, it will send a miss notification to the controller.
  - The controller then issues a command to the processor holding that line that requires the processors to do a write back to main memory.
  - Directory schemes suffer from the drawbacks of a central bottleneck and the overhead of communication between the various cache controllers and the central controller.

Solution to Cache Coherence Problem

Disadvantages of Directory Based Protocol

- Directory schemes suffer from the drawbacks of a central bottleneck and the overhead of communication between the various cache controllers and the central controller.

Applications

- It is used in scalable systems such as developing CC-NUMA (Cache Coherence Non-Uniform Memory Access) architecture.

Solution to Cache Coherence Problem

- **Snoopy Bus Protocol**
  - Snoopy protocols distribute the responsibility for maintaining cache coherence among all of the cache controllers in a multiprocessor system.
  - A cache must recognize when a line that it holds is shared with other caches.
  - When an update action is performed on a shared cache line, it must be announced to all other caches by a broadcast mechanism.
  - Each cache controller is able to "snoop" on the network to observed these broadcasted notification and react accordingly.
  - Snoopy protocols are ideally suited to a bus-based multiprocessor, because the shared bus provides a simple means for broadcasting and snooping.

Solution to Cache Coherence Problem

❖ **Snoopy Bus Protocol**

- Two basic approaches to the snoopy protocol have been explored:-

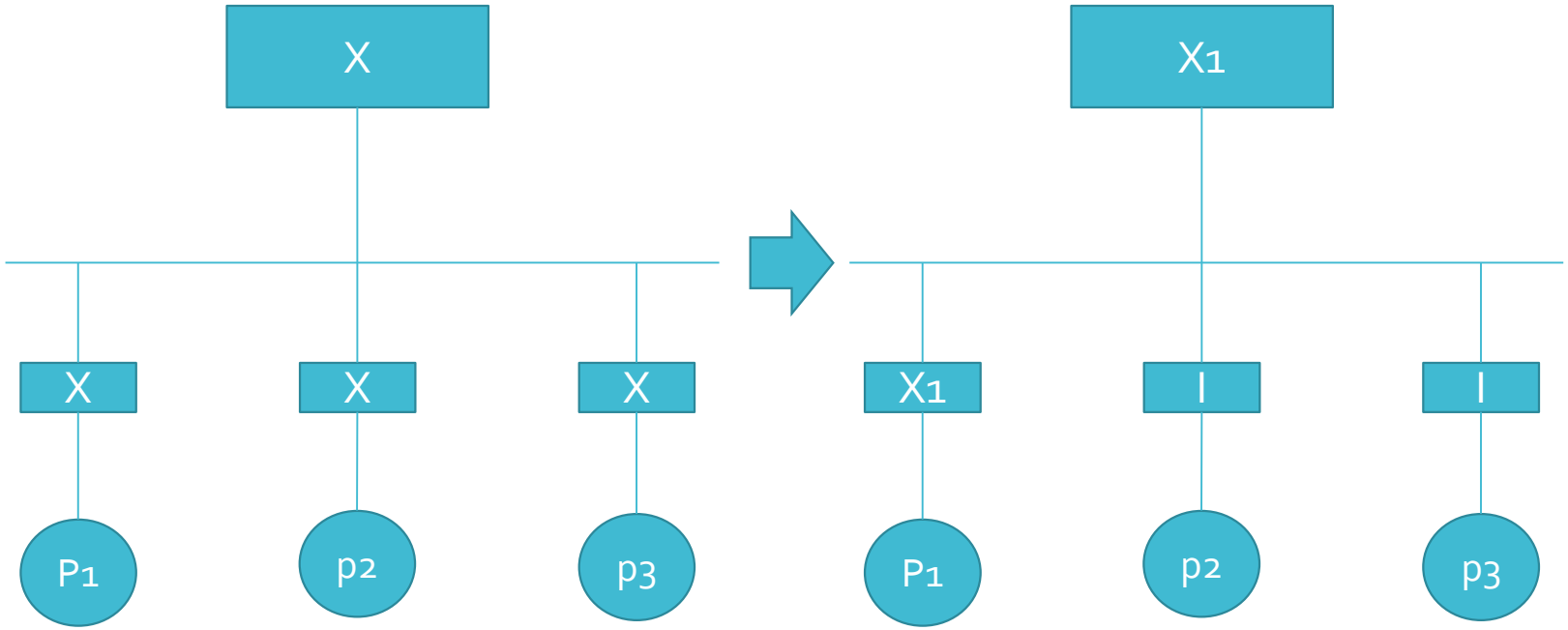
1. **Write invalidates**

➢ When a local cache copy is modified, it invalidate all other remote copies of the caches (invalidated items are also called 'dirty').

2. **Write- update (write-broadcast)**

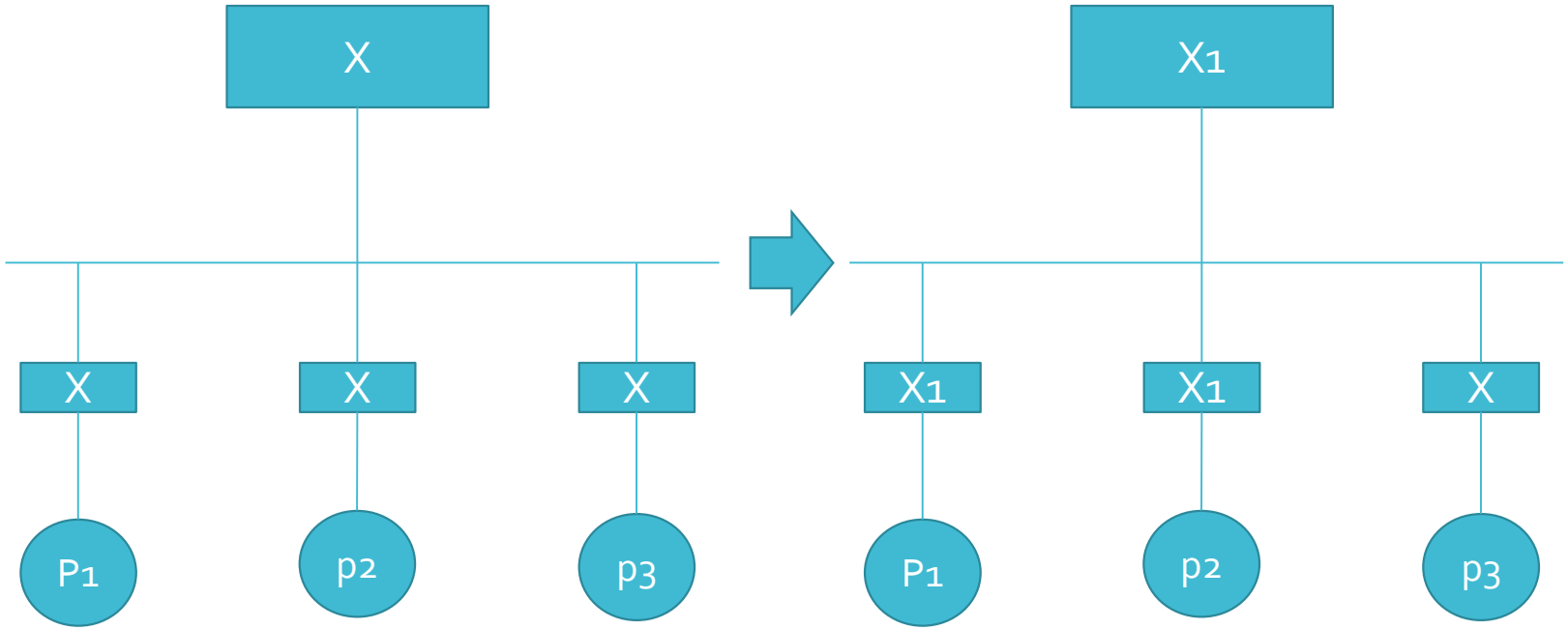
➢ When a local cache copy is modified it broadcasts the modified value of the data object to all other caches at the time of modification.

Solution to Cache Coherence Problem



Write Invalidate (P1 updates it cache from X to X1)

Solution to Cache Coherence Problem



Write Broadcast(P1 updates it cache from X to X1)

Solution to Cache Coherence Problem

❖ **MESI Protocol**

- The write-invalidate approach is the most widely used in commercial multiprocessor systems. It marks the state of every cache line (using two extra bits in the cache tag) as modified, exclusive, shared, or invalid. For this reason, the write-invalidate protocol is called **MESI**.

• **Modified:** The line in the cache has been modified (different from main memory) and is available only in this cache.

• **Exclusive:** The line in the cache is the same as that in main memory and is not present in any other cache.

• **Shared:** The line in the cache is the same as that in main memory and may be present in another cache.

• **Invalid:** The line in the cache does not contain valid data.

Clusters

- Cluster is a group of interconnected, whole computers working together as a unified computing resource that can create the illusion of being one machine.
- The term *whole computer* means a system that can run on its own, apart from the cluster; in the literature, each computer in a cluster is typically referred to as a *node.*
- Clustering is an alternative to symmetric multiprocessing as an approach to providing high performance and high availability and is particularly attractive for server applications.

Benefits of Clusters

- **Absolute scalability:** It is possible to create large clusters that far surpass the power of even the largest standalone machines. A cluster can have tens, hundreds, or even thousands of machines, each of which is a multiprocessor.

- **Incremental scalability:** A cluster is configured in such a way that it is possible to add new systems to the cluster in small increments. Thus, a user can start out with a modest system and expand it as needs grow, without having to go through a major upgrade in which an existing small system is replaced with a larger system.

- **High availability:** Because each node in a cluster is a standalone computer, the failure of one node does not mean loss of service. In many products, fault tolerance is handled automatically in software.

- **Superior price/performance:** By using commodity building blocks, it is possible to put together a cluster with equal or greater computing power than a single large machine, at much lower cost.

Difference Between Clusters and SMP

SMP

- Easy to manage
- Easy to configure
- It requires less physical space
- Low power consumption
- No of processors limit

Clusters

- Relatively difficult to manage and configure
- High power consumption
- It is superior in terms of
  - Scalability
  - Reliability

Clustering Methods

- **Passive Standby**
  - One computer handle all of the processing load while the other computer remains inactive, standing by to take over in the event of a failure of the primary.
  - To coordinate the machines, the active, or primary, system periodically sends a “heartbeat” message to the standby machine.
  - When the message stops arriving, the standby assumes that the primary server has failed and puts itself into operation.
  - **Benefits**
    - It increases availability
  - **Limitations**
    - High cost
    - Performance is low

Clustering Methods

- **Active Secondary**
  - The secondary server is also available for processing long with the primary server.
  - **Benefits**
    - Performance relative to cost is increased.
  - **Limitations**
    - Increased complexity.

Classification of Clusters

- **Separate Servers**
  - Each computer is a separate server with it's own disk and no disks are shared between systems.
  - Data is continuously copied from the primary server to the secondary server.
  - Benefits
    - It provides high performance and availability.
  - Limitations
    - High network and server overhead due to continuous copying operations.

Classification of Clusters

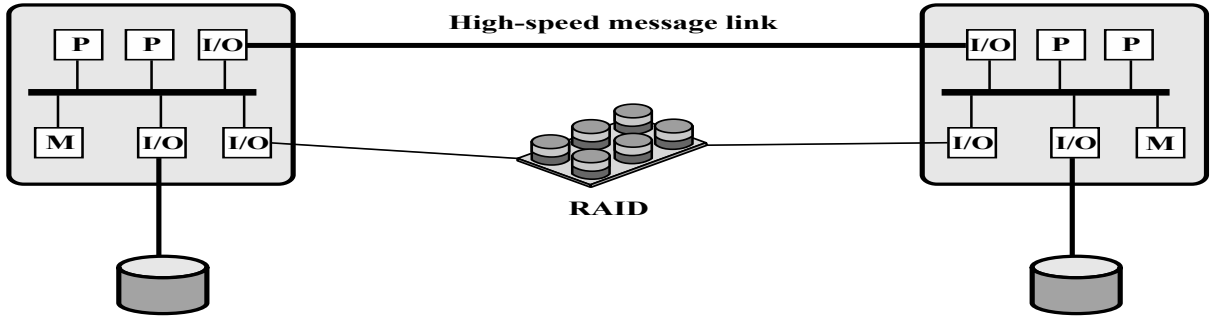
- **Servers connected to disks**
  - To reduce the communications overhead, most clusters now consist of servers connected to common disks.
  - In one variation on this approach, called **shared nothing**, the common disks are partitioned into volumes, and each volume is owned by a single computer.
  - If that computer fails, the cluster must be reconfigured so that some other computer has ownership of the volumes of the failed computer.

Classification of Clusters

- **Servers Share disks**
  - Multiple computers share the same disks at the same time (called the **shared disk** approach), so that each computer has access to all of the volumes on all of the disks.
  - This approach requires the use of some type of locking facility to ensure that data can only be accessed by one computer at a time.

Classification of Clusters

(a) Standby server with no shared disk



(b) Shared disk

Figure: Cluster Organization

Non-Uniform Memory Access (UMA)

- **Uniform Memory Access (UMA)**
  - All processors have access to all parts of main memory using loads and stores. The memory access time of a processor to all regions of memory is the same. The access times experienced by different processors are the same. **Example: SMP**
- **Non-Uniform Access (NUMA)**
  - All processors have access to all parts of main memory using loads and stores. The memory access time of a processor differs depending on which region of main memory is accessed. **Example: Cluster**
- **Cache Coherence NUMA (CC-NUMA)**
  - A NUMA system in which cache coherence is maintained among the caches of the various processors.

Non-Uniform Memory Access (UMA)

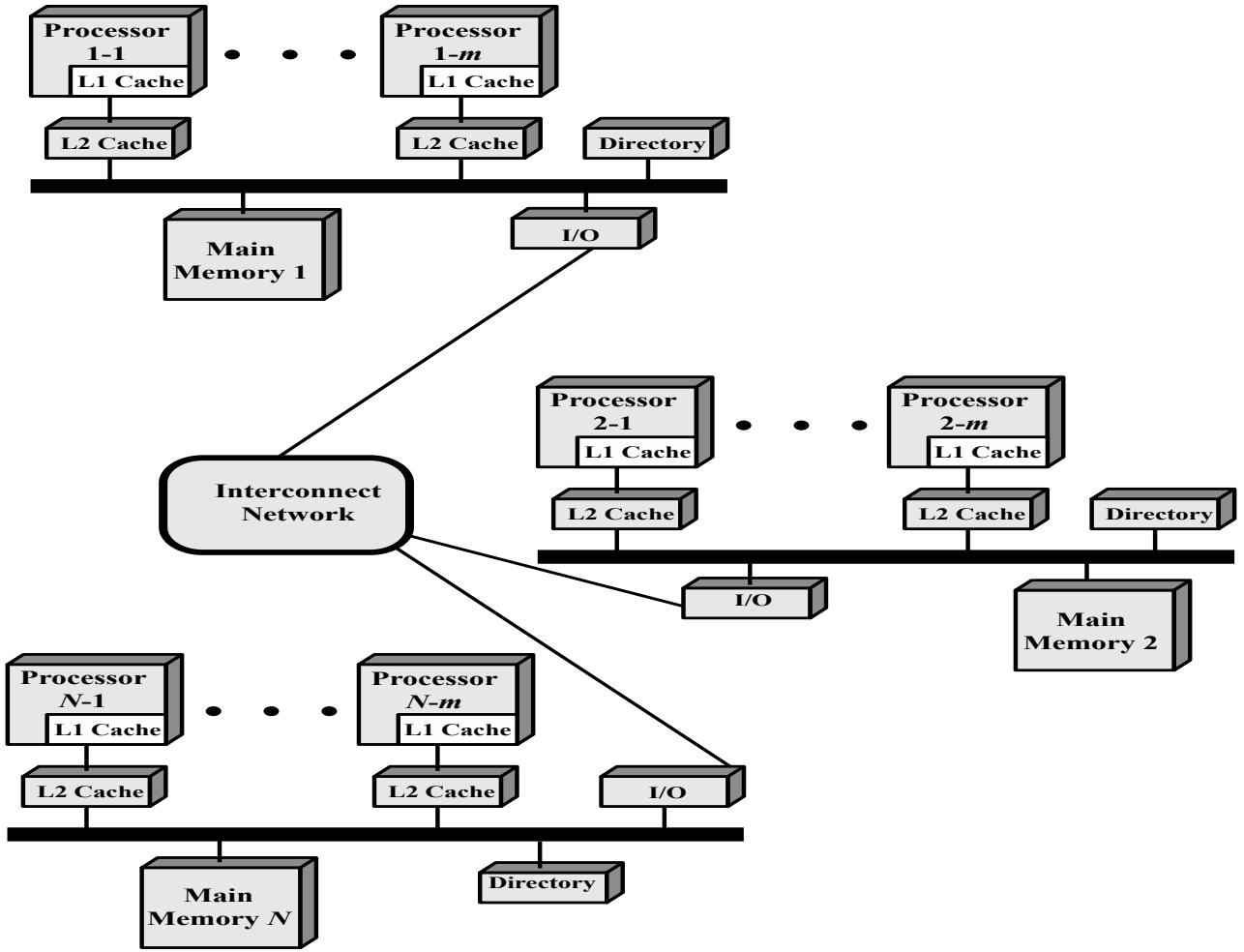


Figure: CC-NUMA

CC-NUMA

- **Suppose that processor 3 on node 2 (P2-3) requests a memory location 798, which is in the memory of node 1. The following sequence occurs:**
  1. P2-3 issues a read request on the snoopy bus of node 2 for location 798.
  2. The directory on node 2 sees the request and recognizes that the location is in node 1.
  3. Node 2's directory sends a request to node 1, which is picked up by node 1's directory.
  4. Node 1's directory, acting as a surrogate of P2-3, requests the contents of 798, as if it were a processor.
  5. Node 1's main memory responds by putting the requested data on the bus.
  6. Node 1's directory picks up the data from the bus.
  7. The value is transferred back to node 2's directory.
  8. Node 2's directory places the data back on node 2's bus, acting as a surrogate for the memory that originally held it.
  9. The value is picked up and placed in P2-3's cache and delivered to P2-3.

Advantages of CC-NUMA

- **Main advantage** of a CC-NUMA system is that it can deliver effective performance at higher levels of parallelism than SMP without requiring major software changes.