Name of Faculty: Dr Jaikaran Singh

Designation : Professor

Department : ECE

Subject: LNCTS CS-4th Sem. CSO

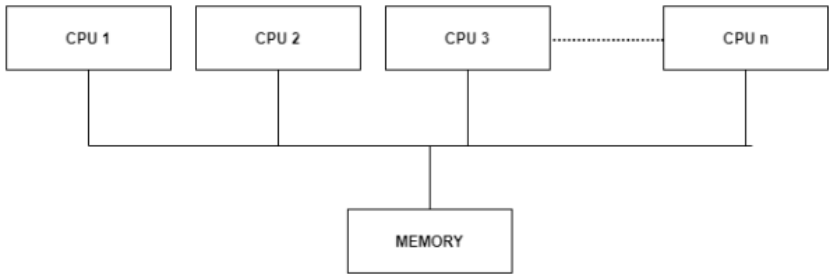
Unit : 5

Topic : Notes for Unit -V

Contents : Multiprocessors: Characteristics of Multiprocessor, Structure of Multiprocessor- Interprocessor Arbitration, Inter-Processor Communication and Synchronization. Memory in Multiprocessor System, Concept of Pipelining, Vector Processing, Array Processing, RISC And CISC, Study of Multicore Processor –Intel, AMD.

Multiprocessor Systems

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 is:



Multiprocessing Architecture

Types of Multiprocessors

There are mainly two types of multiprocessors i.e. symmetric and asymmetric multiprocessors. Details about them are as follows:

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.

Characteristics of multiprocessors

1. A multiprocessor system is an interconnection of two or more CPUs with memory and input-output equipment.
2. The term "processor" in multiprocessor can mean either a central processing unit (CPU) or an input-output processor (IOP).
3. Multiprocessors are classified as *multiple instruction stream, multiple data stream* (MIMD) systems
4. The similarity and distinction between multiprocessor and multicomputer are ``
    - Similarity
        - Both support concurrent operations
    - Distinction
        - The network consists of several autonomous computers that may or may not communicate with each other.
        - A multiprocessor system is controlled by one operating system that provides interaction between processors and all the components of the system cooperate in the solution of a problem.
5. Multiprocessing improves the reliability of the system.
6. The benefit derived from a multiprocessor organization is an improved system performance.
    - Multiple independent jobs can be made to operate in parallel.
    - A single job can be partitioned into multiple parallel tasks.
7. Multiprocessing can improve performance by decomposing a program into parallel executable tasks.
    - The user can explicitly declare that certain tasks of the program be executed in parallel.
        - This must be done prior to loading the program by specifying the parallel executable segments.
    - The other is to provide a compiler with multiprocessor software that can automatically detect parallelism in a user's program.
8. Multiprocessor are classified by the way their memory is organized.
    - A multiprocessor system with *common shared memory* is classified as a *shared-memory* or *tightly coupled multiprocessor*.

- Tolerate a *higher degree* of interaction between tasks.

- Each processor element with its own *private local memory* is classified as a *distributed-memory* or *loosely coupled system.*
  - Are most efficient when the interaction between tasks is *minimal*

Interconnection Structures

1. The components that form a multiprocessor system are CPUs, IOPs connected to input-output devices, and a memory unit.
2. The interconnection between the components can have different physical configurations, depending on the number of transfer paths that are available
   - o Between the processors and memory in a shared memory system
   - o Among the processing elements in a loosely coupled system
3. There are several physical forms available for establishing an interconnection network.
   - o Time-shared common bus
   - o Multiport memory
   - o Crossbar switch
   - o Multistage switching network
   - o Hypercube system

Time Shared Common Bus

1. A common-bus multiprocessor system consists of a number of processors connected through a common path to a memory unit.
2. *Disadv.*:
   - o Only one processor can communicate with the memory or another processor at any given time.
   - o As a consequence, the total overall transfer rate within the system is limited by the speed of the single path
3. 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.

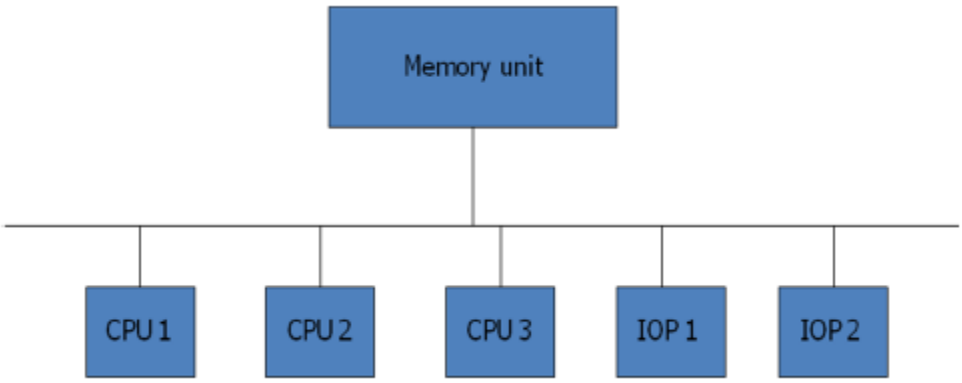


Fig: Time shared common bus organization

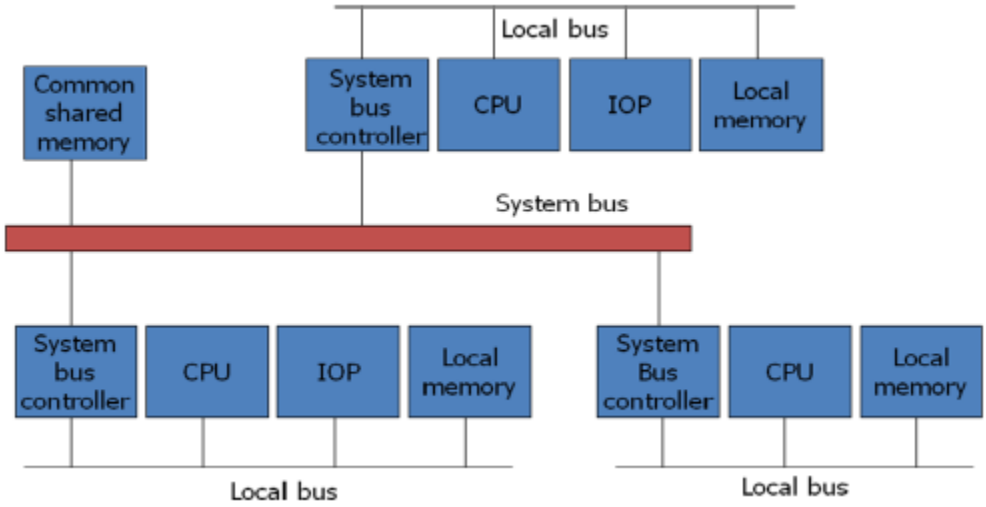


Fig: System bus structure for multiprocessorsa

Multiport Memory

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

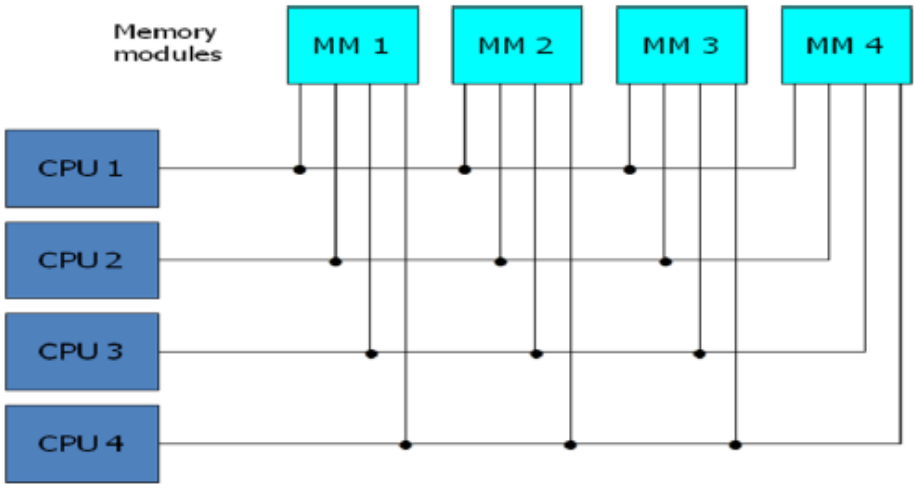


Fig: Multiport memory organization

Crossbar Switch

1. Consists of a number of *crosspoints* that are placed at intersections between processor buses and memory module paths.
2. The small square in each crosspoint is a *switch* that determines the path from a processor to a memory module.
3. Adv.:
   - o Supports simultaneous transfers from all memory modules
4. Disadv.:
   - o The hardware required to implement the switch can become quite large and complex.
5. Below fig. shows the functional design of a crossbar switch connected to one memory module.

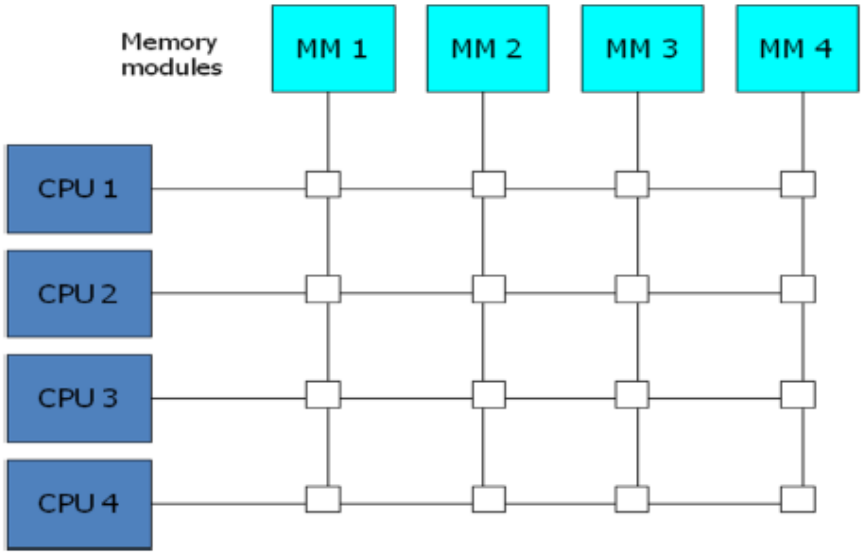


Fig: Crossbar switch

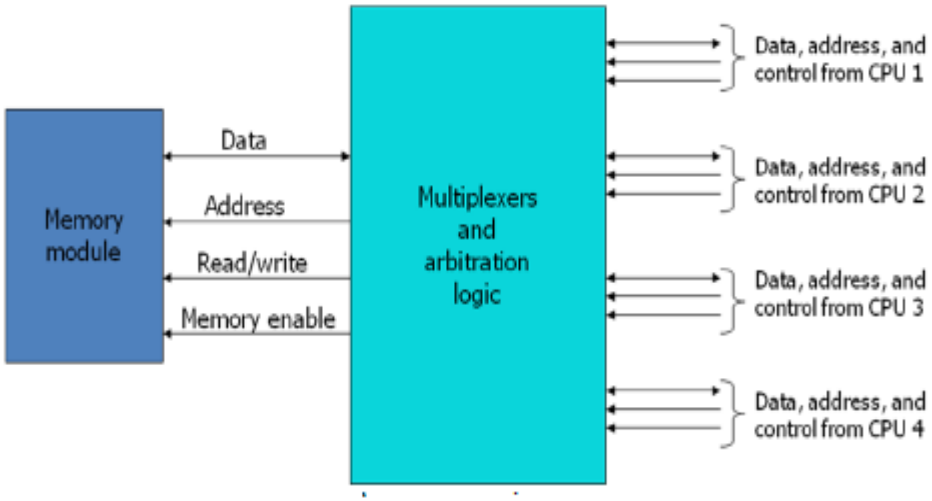
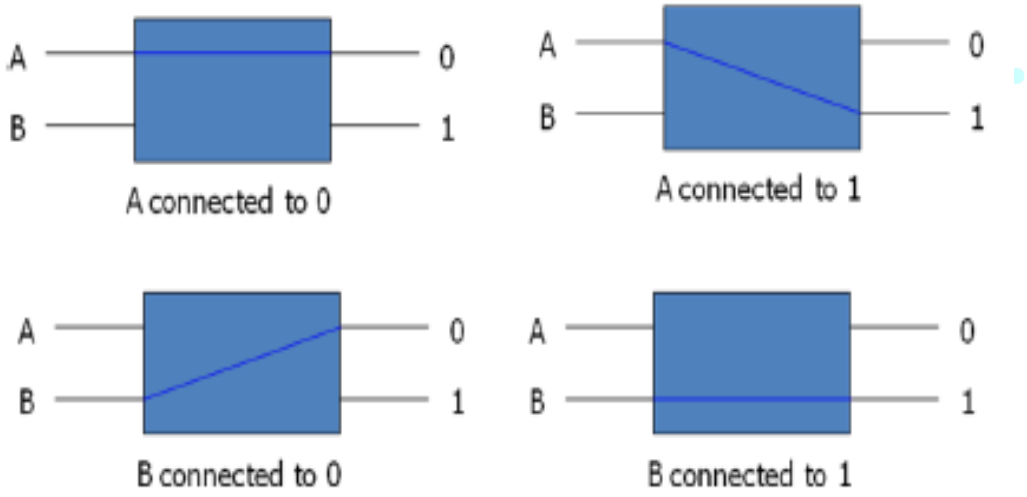


Fig: Block diagram of crossbar switch

Multistage Switching Network

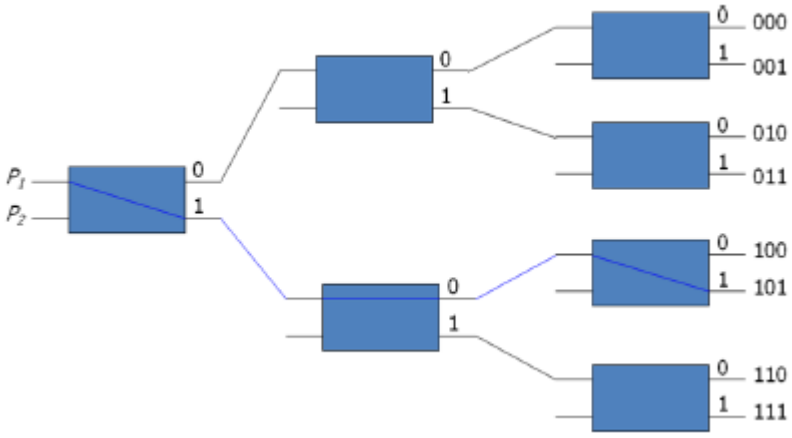
1. The basic component of a multistage network is a two-input, two-output interchange switch as shown in Fig. below.



2. 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.

o To see how this is done, consider the binary tree shown in Fig. below.

o Certain request patterns cannot be satisfied simultaneously. i.e., if P1 □ 000~011, then P2 □ 100~111



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

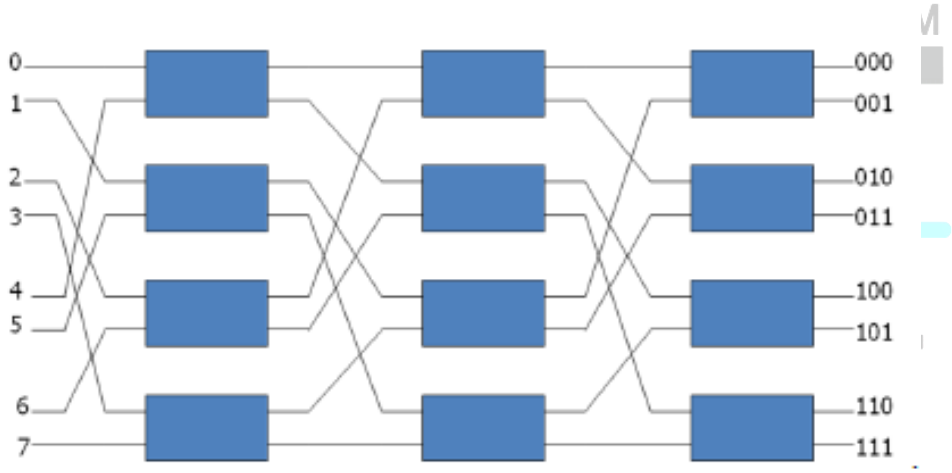


Fig: 8 x 8 Omega Switching Network

4. Some request patterns cannot be connected simultaneously. i.e., any two sources cannot be connected simultaneously to destination 000 and 001
5. In a tightly coupled multiprocessor system, the source is a processor and the destination is a memory module.
6. Set up the path □ transfer the address into memory □ transfer the data

7. In a loosely coupled multiprocessor system, both the source and destination are processing elements.

Hypercube System

1. The hypercube or binary n-cube multiprocessor structure is a loosely coupled system composed of N=2n processors interconnected in an n-dimensional binary cube.
   - o Each processor forms a node of the cube, in effect it contains not only a CPU but also local memory and I/O interface.
   - o Each processor address differs from that of each of its n neighbors by exactly one bit position.
2. Fig. below shows the hypercube structure for n=1, 2, and 3.
3. Routing messages through an *n*-cube structure may take from one to *n* links from a source node to a destination node.
   - o A routing procedure can be developed by computing the exclusive-OR of the source node address with the destination node address.
   - o The message is then sent along any one of the axes that the resulting binary value will have 1 bits corresponding to the axes on which the two nodes differ.
4. A representative of the hypercube architecture is the Intel iPSC computer complex.
   - o It consists of 128(*n*=7) microcomputers, each node consists of a CPU, a floating-point processor, local memory, and serial communication interface units.

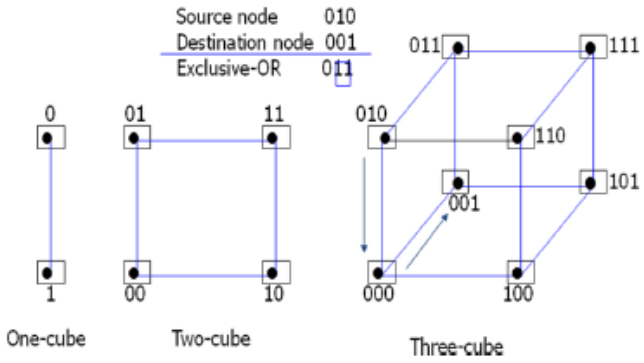


Fig: Hypercube structures for n=1,2,3

Inter processor Communication and Synchronization

1. The various processors in a multiprocessor system must be provided with a facility for *communicating* with each other.
    - o A communication path can be established through *a portion of memory* or *a common input-output channels*.
2. The sending processor structures a request, a message, or a procedure, and places it in the memory mailbox.
    - o *Status bits* residing in common memory
    - o The receiving processor can check the mailbox *periodically*.
    - o The response time of this procedure can be time consuming.
3. A more efficient procedure is for the sending processor to alert the receiving processor directly by means of an *interrupt signal*.
4. In addition to shared memory, a multiprocessor system may have other shared resources. e.g., a magnetic disk storage unit.
5. To prevent conflicting use of shared resources by several processors there must be a provision for assigning resources to processors. i.e., operating system.
6. There are three organizations that have been used in the design of operating system for multiprocessors: *master-slave configuration, separate operating system*, and *distributed operating system*.
7. In a master-slave mode, one processor, master, always executes the operating system functions.
8. In the separate operating system organization, each processor can execute the operating system routines it needs. This organization is more suitable for *loosely coupled systems*.
9. In the distributed operating system organization, the operating system routines are distributed among the available processors. However, each particular operating system function is assigned to only one processor at a time. It is also referred to as a *floating operating system*.

Loosely Coupled System

1. There is *no shared memory* for passing information.
2. The communication between processors is by means of message passing through *I/O channels*.
3. The communication is initiated by one processor calling a *procedure* that resides in the memory of the processor with which it wishes to communicate.

4. The communication efficiency of the interprocessor network depends on the *communication routing protocol, processor speed, data link speed, and the topology of the network.*

Interprocess Synchronization

1. The instruction set of a multiprocessor contains basic instructions that are used to implement communication and synchronization between cooperating processes.
    - o Communication refers to the exchange of data between different processes.
    - o Synchronization refers to the special case where the data used to communicate between processors is control information.

2. Synchronization is needed to enforce the *correct sequence of processes* and to ensure *mutually exclusive access* to shared writable data.
3. Multiprocessor systems usually include various mechanisms to deal with the synchronization of resources.
    - o Low-level primitives are implemented directly by the hardware.
    - o These primitives are the basic mechanisms that enforce mutual exclusion for more complex mechanisms implemented in software.
    - o A number of hardware mechanisms for mutual exclusion have been developed.
4. A binary semaphore

Mutual Exclusion with Semaphore

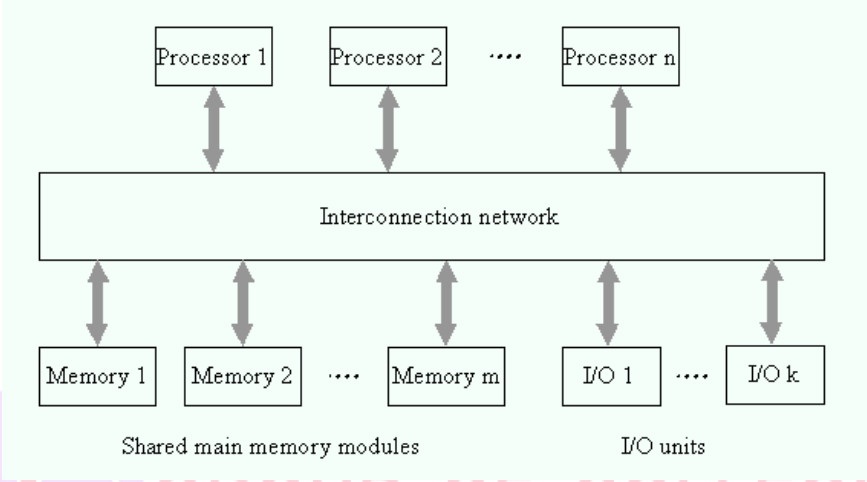
1. A properly functioning multiprocessor system must provide a mechanism that will guarantee orderly access to shared memory and other shared resources.
    - o Mutual exclusion: This is necessary to protect data from being changed simultaneously by two or more processors.
    - o Critical section: is a program sequence that must complete execution before another processor accesses the same shared resource.
2. A *binary variable* called a *semaphore* is often used to indicate whether or not a processor is executing a critical section.
3. Testing and setting the semaphore is itself a critical operation and must be performed as a single indivisible operation.
4. A semaphore can be initialized by means of a *test and set instruction* in conjunction with a hardware *lock* mechanism.

5. The instruction TSL SEM will be executed in two memory cycles (the first to read and the second to write) as follows: R □ M[*SEM*], M[*SEM*] □ 1
6. Note that the lock signal must be active during the execution of the test-and-set instruction.

Memory in multiprocessor Systems

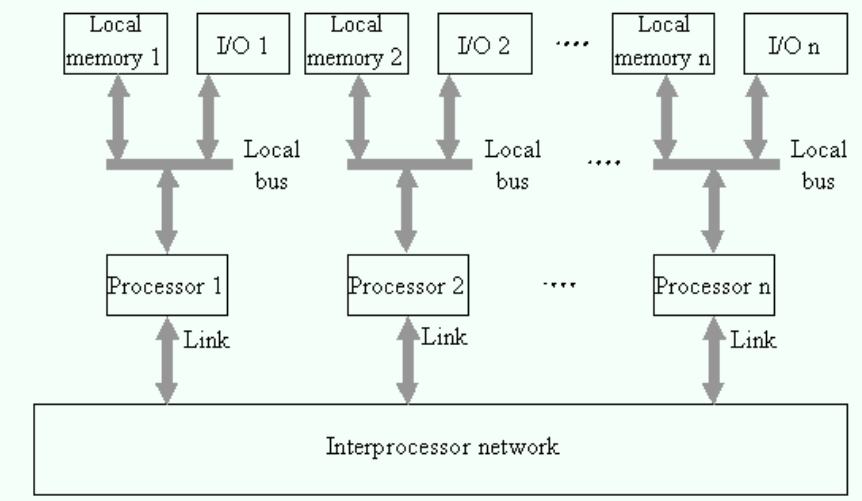
Data and code in a parallel program are stored in the main memory accessible for processors of the executive system.

Regarding the way in which the main memory is used by processors in a multiprocessor system, we divide parallel systems onto **shared memory system** and **distributed memory systems.** See the figures below.



A multiprocessor system with shared memory (tightly coupled system)

In a shared memory system, all processors can access all the main memory address space. Fragments of the address space are usually located in separate memory modules, which are supplied with separate address decoders. Communication between processors (program code fragments) is done by means of shared variables access in the main memory. It is called **communication through shared variables**. Fetching instructions for execution in processors is also done from a shared memory. The efficiency of accesing memory modules depends on the structure and properties of the interconnection network. This network is a factor, which limitates the memory access throughput for a larger number of processors. It sets a limit on the number of processors in such systems, with which good efficiency of a parallel system is achieved. Multiprocessor systems with shared memory are called **tightly coupled systems** or **multiprocessors**. Due to symmetric access of all processors to all memory modules, the computations in such systems are called **Symmetric Multiprocessing - SMP**.



A multiprocessor system with a distributed memory (loosely coupled system)

In a distributed memory multiprocessor system, each processor has its local memory with the address space available only for this processor. Processors can exchange data through the interconnection network by means of communication through the **message passing**.

The instructions "send message" and "receive message" are used in programs for this purpose. The communication instructions send or receive messages with the use of identifiers of special elements (variables) are called **communication channels**.

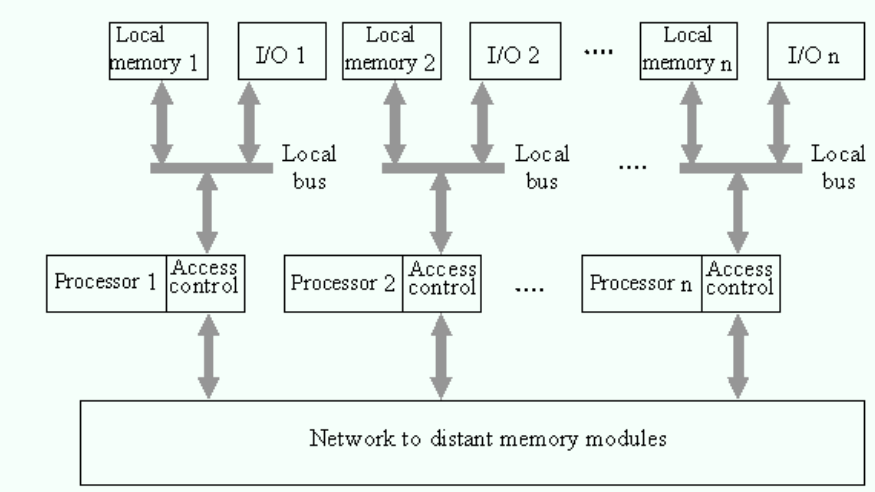
The channels represent the use of connections that exist permanently (or are created in the interconnection network) between processors. There exist processors that are specvially adapted for sending and receiving messages by the existance of communication links. Communication links can be serial or parallel. The number of communication links in such processors is from 4 to 6 (ex. transputer - 4 serial links, SHARC - a DSP (Data Signal Processor) from Analog Devices - 6 parallel links). Each link is supervised by an independent processor controller that organizes external data transmissions over the link. When a message is sent, it is fetched from the processor main memory. A message received from a link is next sent to the main memory. Multiprocessor systems that have distributed memory are called in the literature **loosely coupled systems**. In such systems, it is possible to organize many inter-processor connections at the same time. It provides high communication efficiency and, as a consequence, high efficiency of parallel computations in processors (due to distribution of memory accesses), which gives rise to calling computations in such systems the **Massively Parallel Processing - MPP.**

Communication by message passing in such systems can be executed according to the **synchronous or asynchronous communication model.**

In the synchronous communication model, the partner processes (programs) - the sending and the receiving one, get synchronized on communication instructions in a given channel. It means that the sending process can start transmitting data only if the receiving process in another processor has reached execution of the receive instruction in the same channel as the sending one. Since the communication is performed with the use of send and receive instructions in both processors simultaneously, there is no need of buffering of messages, and so, they are sent as if they were sent directly from the main memory of one processor to the memory of the other one. All this is done under supervision of link controllers in both processors.

With the asynchronous communication model, the sending and receiving processes (programs) do not synchronize communication execution in the involved channels. A message is sent to a channel at any time and it is directed to the buffer for messages in a given channel in the controller at the other side of the interconnection between the processors. The receiving process reads the message from the buffer of the given channel at any convenient time.

The third type of multiprocessor systems are systems with the **distributed shared** memory called also the **virtual shared memory**. In such systems, which currently show strong development, each processor has a local main memory. However, each memory is placed in a common address space of the entire system. It means that each processor can have access to the local memory of any other processor. In this type of the system, communication between processors is done by accessing sahred variables. It involves execution of a simple read or write instruction converning the shared variables in the memory of another processor. In each processor, a memory interface unit egzamines addresses used in current processor memory access instructions. As a result, it directs instruction execution to the local main memory bus or it sends the address together with the operation code to the local memory interface of another processor. Sending the address and later the data is performed through the network that connects all processors (their local main memory interface units).



A multiprocessor system with the distributed shared memory

This type of the system is called in the literature the system with the Scalable Parallel Processing - SPP, since with the increase of the number of processors in the system, used to perform a given parallel program, the parallel program speedup in the system increases proportionally to the increase of the number of processors. Such a property is called **parallel system scalability**.

Pipelining in Computer Architecture

Pipelining organizes the execution of the **multiple instructions simultaneously**. Pipelining improves the **throughput** of the system. In pipelining the instruction is divided into the subtasks. Each subtask performs the dedicated task.

The instruction is divided into 5 subtasks: **instruction fetch**, **instruction decode**, **operand fetch**, **instruction execution** and **operand store**. The instruction fetch subtask will only perform the instruction fetching operation, instruction decode subtask will only be decoding the fetched instruction and so on the other subtasks will do.

In this section, we will discuss the types of pipelining, pipelining hazards, its advantage. So let us start.

1. Introduction
2. Types of Pipelining
3. Pipelining Hazards
4. Advantages
5. Key Takeaways

Introduction

Have you ever visited an industrial plant and see the assembly lines over there? How a product passes through the assembly line and while passing it is worked on, at different phases simultaneously. For **example**, take a car manufacturing plant. At the first stage, the automobile chassis is prepared, in the next stage workers add body to the chassis, further, the engine is installed, then painting work is done and so on.

The group of workers after working on the chassis of the first car don't sit idle. They start working on the chassis of the next car. And the next group take the chassis of the car and add body to it. The same thing is repeated at every stage, after finishing the work on the current car body they take on next car body which is the output of the previous stage.

Here, though the first car is completed in several hours or days, due to the assembly line arrangement it becomes possible to have a new car at the end of an assembly line in every clock cycle.

Similarly, the concept of **pipelining** works. The output of the first pipeline becomes the input for the next pipeline. It is like a set of data processing unit connected in series to utilize processor up to its maximum.

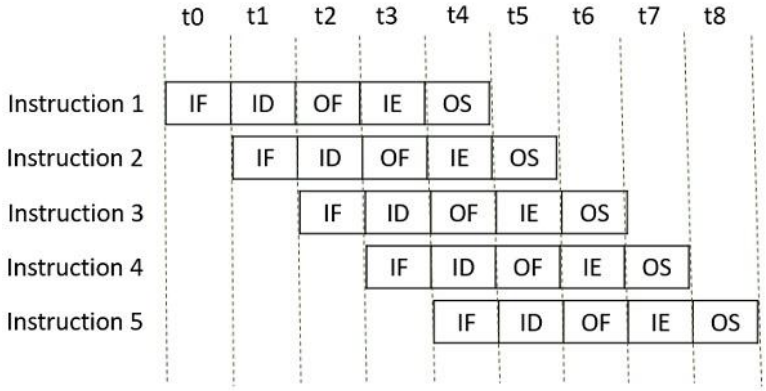
An instruction in a process is divided into 5 subtasks likely,

Instruction Fetch	Instruction Decode	Operand Fetch	Instruction Execute	Operand Store

1. In the first subtask, the instruction is fetched.
2. The fetched instruction is decoded in the second stage.
3. In the third stage, the operands of the instruction are fetched.
4. In the fourth, arithmetic and logical operation are performed on the operands to execute the instruction.
5. In the fifth stage, the result is stored in memory.

Now, understanding the division of the instruction into subtasks. Let us understand, how the n number of instructions in a process, are pipelined.

Look at the figure below the 5 instructions are pipelined. The first instruction gets completed in 5 clock cycle. After the completion of first instruction, in every new clock cycle, a new instruction completes its execution.



Pipelining of 5 Instructions

Observe that when the Instruction fetch operation of the first instruction is completed in the next clock cycle the instruction fetch of second instruction gets started. This way the hardware never sits idle it is always busy in performing some or other operation. But, no two instructions can **execute** their **same stage** at the **same clock cycle**.

Types of Pipelining

In 1977 Handler and Ramamoorthy classified pipeline processors depending on their functionality.

1. Arithmetic Pipelining

It is designed to perform high-speed floating-point addition, multiplication and division. Here, the multiple arithmetic logic units are built in the system to perform the parallel arithmetic computation in various data format. Examples of the arithmetic pipelined processor are Star-100, TI-ASC, Cray-1, Cyber-205.

2. Instruction Pipelining

Here, the number of instruction are pipelined and the execution of current instruction is overlapped by the execution of the subsequent instruction. It is also called **instruction lookahead**.

3. Processor Pipelining

Here, the processors are pipelined to process the **same data stream**. The data stream is processed by the first processor and the result is stored in the memory block. The result in the memory block is accessed by the second processor. The second processor reprocesses the result obtained by the first processor and the passes the refined result to the third processor and so on.

4. Unifunction Vs. Multifunction Pipelining

The pipeline performing the precise function every time is unifunctional pipeline. On the other hand, the pipeline performing multiple functions at a different time or multiple functions at the same time is multifunction pipeline.

5. Static vs Dynamic Pipelining

The static pipeline performs a fixed-function each time. The static pipeline is unifunctional. The static pipeline executes the same type of instructions continuously. Frequent change in the type of instruction may vary the performance of the pipelining.

Dynamic pipeline performs several functions simultaneously. It is a multifunction pipelining.

6. Scalar vs Vector Pipelining

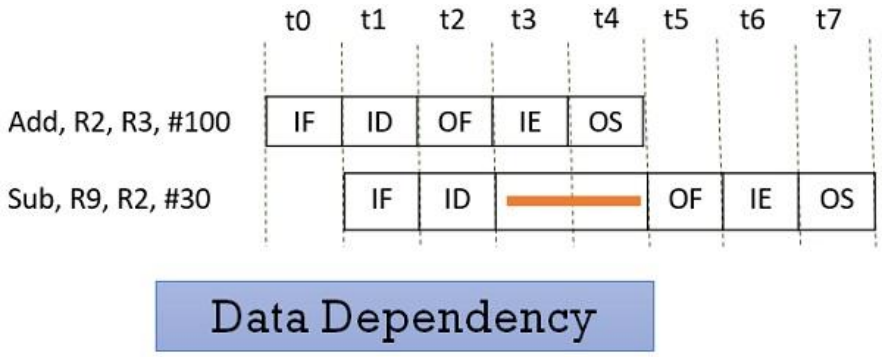
Scalar pipelining processes the instructions with scalar operands. The vector pipeline processes the instruction with vector operands.

Pipelining Hazards

Whenever a pipeline has to stall due to some reason it is called pipeline hazards. Below we have discussed four pipelining hazards.

1. Data Dependency

Consider the following two instructions and their pipeline execution:



In the figure above, you can see that result of the **Add** instruction is stored in the register **R2** and we know that the final result is stored at the end of the execution of the instruction which will happen at the clock cycle **t4**.

But the **Sub** instruction need the value of the register **R2** at the cycle **t3**. So the Sub instruction has to **stall** two clock cycles. If it doesn't stall it will generate an incorrect result. Thus depending of one instruction on other instruction for data is **data dependency**.

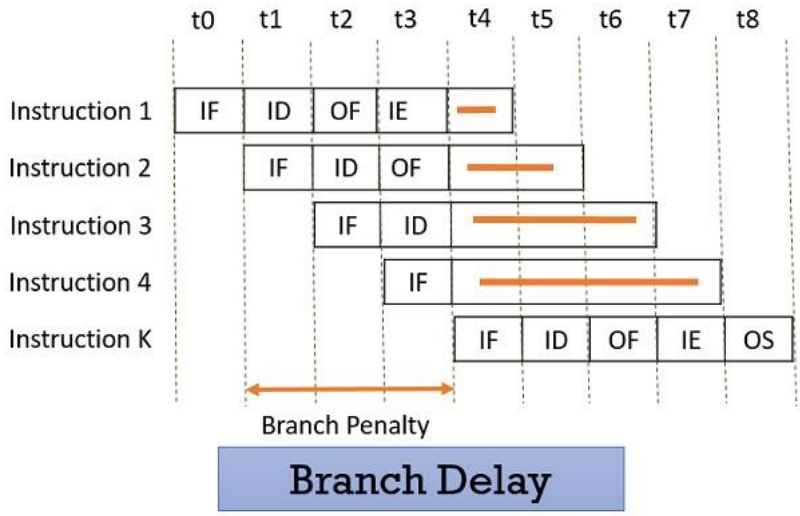
2. Memory Delay

When an instruction or data is required, it is first searched in the cache memory if not found then it is a **cache miss**. The data is further searched in the memory which may take ten or more cycles. So, for that number of cycle the pipeline has to stall and this is a **memory delay** hazard. The cache miss, also results in the delay of all the subsequent instructions.

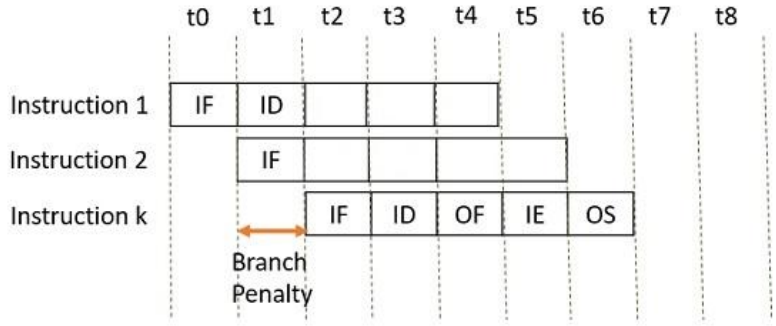
3. Branch Delay

Suppose the four instructions are pipelined I_1, I_2, I_3, I_4 in a sequence. The instruction I_1 is a branch instruction and its target instruction is I_k. Now, processing starts and instruction I_1 is fetched, decoded and the target address is computed at the 4^{th} stage in cycle t3.

But till then the instructions I_2, I_3, I_4 are fetched in cycle 1, 2 & 3 before the target branch address is computed. As I_1 is found to be a branch instruction, the instructions I_2, I_3, I_4 has to be discarded because the instruction I_k has to be processed next to I_1. So, this delay of three cycles 1, 2, 3 is a **branch delay**.



Prefetching the target branch address will reduce the branch delay. Like if the target branch is identified at the decode stage then the branch delay will reduce to 1 clock cycle.



Branch Delay when Target Branch is Determined at Decode stage

4. Resource Limitation

If the two instructions request for accessing the same resource in the same clock cycle, then one of the instruction has to stall and let the other instruction to use the resource. This stalling is due to **resource limitation**. However, it can be prevented by adding more hardware.

Advantages

1. Pipelining improves the throughput of the system.
2. In every clock cycle, a new instruction finishes its execution.
3. Allow multiple instructions to be executed concurrently.

Key Takeaways

- Pipelining divides the instruction in 5 stages instruction fetch, instruction decode, operand fetch, instruction execution and operand store.
- The pipeline allows the execution of multiple instructions concurrently with the limitation that no two instructions would be executed at the **same stage** in the **same clock cycle**.
- All the stages must process at equal speed else the slowest stage would become the bottleneck.
- Whenever a pipeline has to stall for any reason it is a pipeline hazard.

This is all about pipelining. So, basically the pipelining is used to improve the performance of the system by improving its efficiency.

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.

In this section, we will have a brief introduction on vector processing, its characteristics, about vector instructions and how the performance of the vector processing can be enhanced? So lets us start.

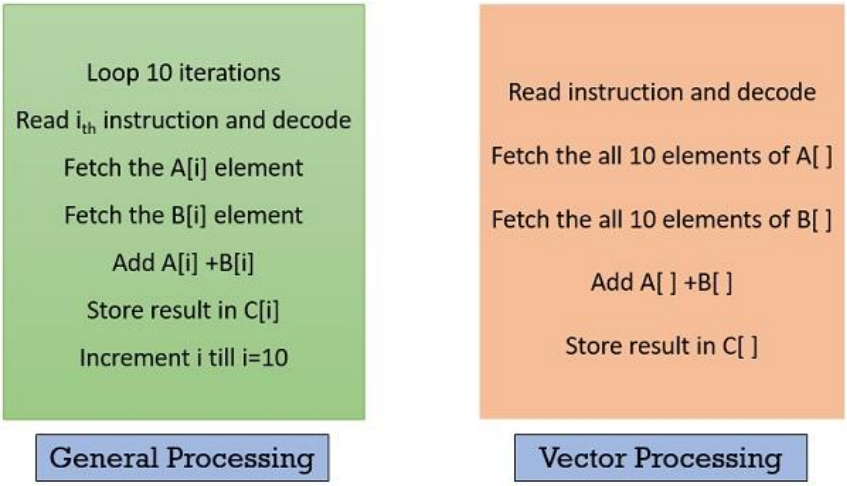
Content: Vector Processing in Computer Architecture

Introduction

We need computers that can solve mathematical problems for us which include, arithmetic operations on the large arrays of integers or floating-point numbers quickly. The general-purpose computer would use loops to operate on an array of integers or floating-point numbers. But, for large array using loop would cause overhead to the processor.

To avoid the overhead of processing loops and fasten the computation, some kind of parallelism must be introduced. **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.



Operating on multiple data in just one instruction is also called **Single Instruction Multiple Data** (SIMD) or they are also termed as **Vector instructions**. Now, the data for vector instruction are stored in **vector registers**.

Each vector register is capable of storing several data elements at a time. These several data elements in a vector register is termed as a **vector operand**. So, if there are n number of elements in a vector operand then n is the **length of the vector**.

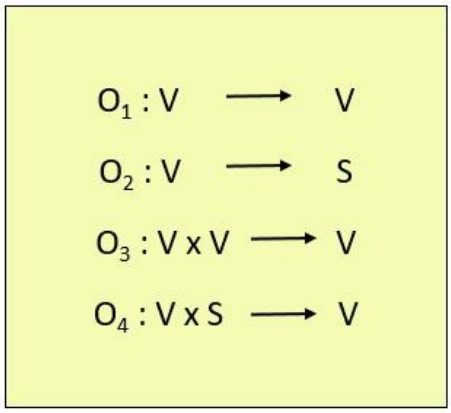
Supercomputers were evolved to deal with billions of floating-point operations/second. Supercomputer optimizes numerical computations (vector computations).

But, along with vector processing supercomputers are also capable of doing scalar processing. Later, **Array processor** was introduced which particularly deals with vector processing, they do not indulge in scalar processing.

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**. TI-ASC, CDC STAR-100, and Cyber-205 use memory-to-memory format for vector instructions.

In **Register-to-Register** vector processor the source operands for instruction, the intermediate result, and the final result all are retrieved from **vector or scalar registers**. Cray-1 and Fujitsu VP-200 use register-to-register format for vector instructions.

Vector Instruction

A vector instruction has the following fields:

1. Operation Code

Operation code indicates the **operation that** has to be performed in the given instruction. It decides the functional unit for the specified operation or reconfigures the multifunction unit.

2. Base Address

Base address field refers to the **memory location** from where the operands are to be fetched or to where the result has to be stored. The base address is found in the memory reference instructions. In the vector instruction, the operand and the result both are stored in the vector registers. Here, the **base address** refers to the designated **vector register**.

3. Address Increment

A vector operand has several data elements and address increment specifies the **address of the next element in the operand**. Some computer stores the data element consecutively in main memory for which the increment is always 1. But, some computers that do not store the data elements consecutively requires the variable address increment.

4. Address Offset

Address Offset is always specified related to the base address. The effective **memory address** is calculated using the address offset.

5. Vector Length

Vector length specifies the **number of elements in a vector operand**. It identifies the **termination** of a vector instruction.

Improving Performance

In vector processing, we come across two overheads setup time and flushing time. When the vector processing is pipelined, the time required to *route the vector operands* to the *functional unit* is called **Set up time**. **Flushing time** is a *time duration* that a vector instruction takes right from its *decoding* until its *first result is out* from the pipeline.

The vector length also affects the efficiency of processing as the longer vector length would cause overhead of subdividing the long vector for processing.

For obtaining the better performance the optimized object code must be produced in order to utilize pipeline resources to its maximum.

1. Improving the vector instruction

We can improve the vector instruction by reducing the memory access, and maximize resource utilization.

2. Integrate the scalar instruction

The scalar instruction of the same type must be integrated as a batch. As it will reduce the overhead of reconfiguring the pipeline again and again.

3. Algorithm

Choose the algorithm that would work faster for vector pipelined processing.

4. Vectorizing Compiler

A vectorizing compiler must regenerate the parallelism by using the higher-level programming language. In advance programming, the four-stage are identified in the development of the parallelism. Those are

Parallel Algorithm(A)

High-level Language(L)

Efficient object code(O)

Target machine code (M)

You can see a parameter in the parenthesis at each stage which denotes the degree of parallelism. In the ideal situation, the parameters are expected in the order A≥L≥O≥M.

Summary

- Computers having vector instruction are vector processors.

- Vector processor have the vector instructions which operates on the large array of integer or floating-point numbers or logical values or characters, all elements in parallel. It is called **vectorization**.
- Vectorization is possible only if the operation performed in parallel are **independent** of each other.
- Operands of vector instruction are stored in the **vector register**. A vector register stores several data elements at a time which is called **vector operand**.
- A vector operand has several **scalar data elements**.
- A vector instruction needs to perform the same operation on the different data set. Hence, vector processors have a **pipelined** structure.
- Vector processing ignores the overhead caused due to the loops while operating on an array.

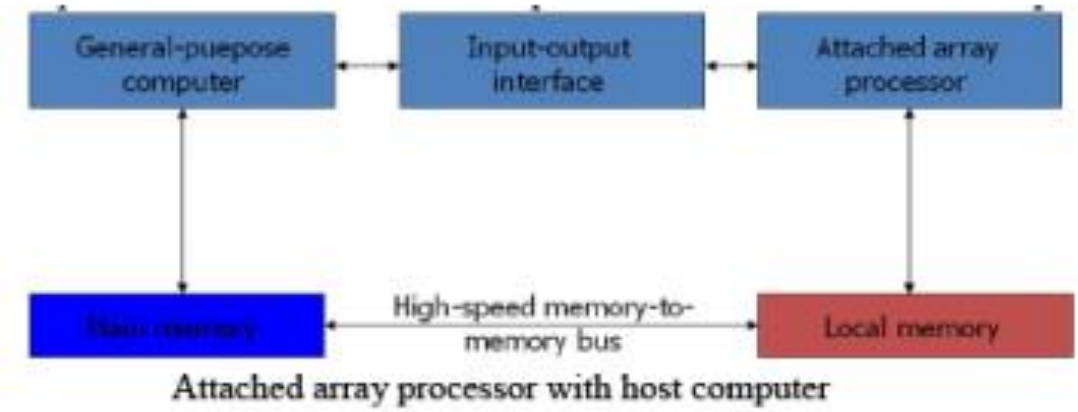
So, this is how vector processing allows parallel operation on the large arrays and fasten the processing speed.

Array Processing

- An array processor is a processor that performs computations on large arrays of data.
- The term is used to refer to two different types of processors.
  - **Attached array processor:**
    - Is an auxiliary processor.
    - It is intended to improve the performance of the host computer in specific numerical computation tasks.
  - **SIMD array processor:**
    - Has a single-instruction multiple-data organization.
    - It manipulates vector instructions by means of multiple functional units responding to a common instruction.

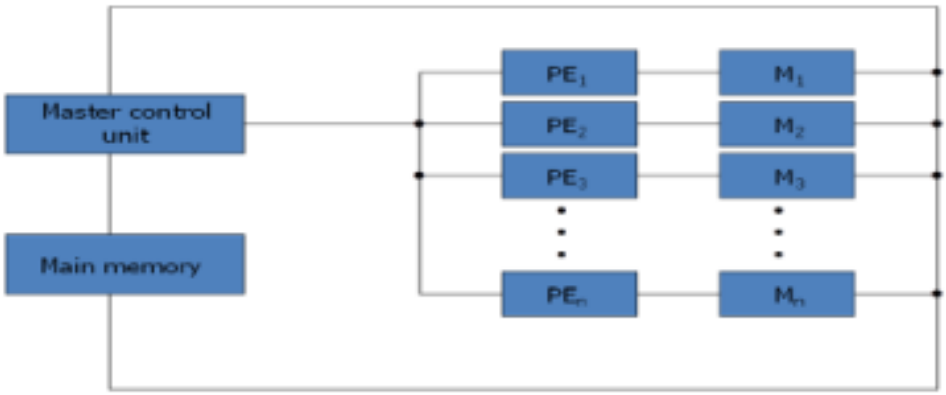
Attached Array Processor

- Its purpose is to enhance the performance of the computer by providing vector processing for complex scientific applications.
  - Parallel processing with multiple functional units
- Fig. shows the interconnection of an attached array processor to a host computer.
- For example, when attached to a VAX 11 computer, the FSP-164/MAX from Floating-Point Systems increases the computing power of the VAX to 100megaflops.
- The objective of the attached array processor is to provide vector manipulation capabilities to a conventional computer at a fraction of the cost of supercomputer.



Attached array processor with host computer

- Masking schemes are used to control the status of each PE during the execution of vector instructions.
  - Each PE has a flag that is set when the PE is active and reset when the PE is inactive.
- Masking schemes are used to control the status of each PE during the execution of vector instructions.
  - Each PE has a flag that is set when the PE is active and reset when the PE is inactive.
- For example, the ILLIAC IV computer developed at the University of Illinois and manufactured by the Burroughs Corp.
  - Are highly specialized computers.
  - They are suited primarily for numerical problems that can be expressed in vector or matrix form.



SIMD array processor organization

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.

In this section, we will have a brief introduction on vector processing, its characteristics, about vector instructions and how the performance of the vector processing can be enhanced? So lets us start.

Content: Vector Processing in Computer Architecture

6. Introduction
7. Characteristics
8. Vector Instruction
9. Improving Performance
10. Key Takeaways

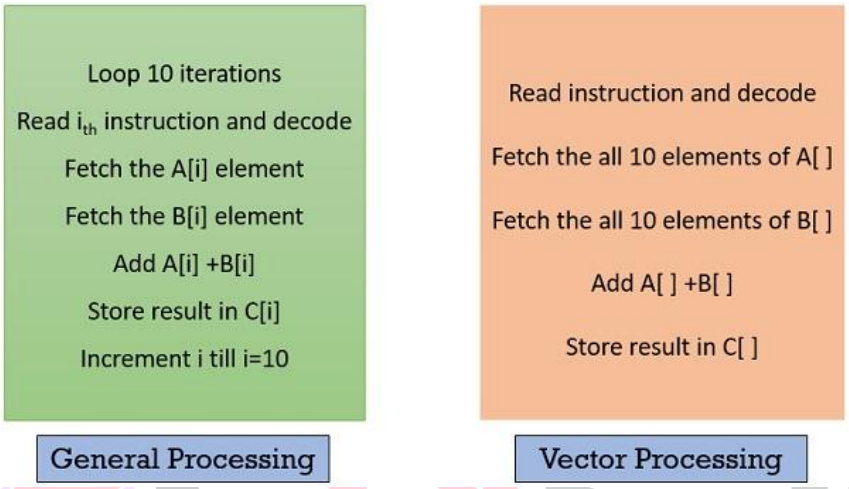
Introduction

We need computers that can solve mathematical problems for us which include, arithmetic operations on the large arrays of integers or floating-point numbers quickly. The general-purpose

computer would use loops to operate on an array of integers or floating-point numbers. But, for large array using loop would cause overhead to the processor.

To avoid the overhead of processing loops and fasten the computation, some kind of parallelism must be introduced. **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.



Operating on multiple data in just one instruction is also called **Single Instruction Multiple Data** (SIMD) or they are also termed as **Vector instructions**. Now, the data for vector instruction are stored in **vector registers**.

Each vector register is capable of storing several data elements at a time. These several data elements in a vector register is termed as a **vector operand**. So, if there are n number of elements in a vector operand then n is the **length of the vector**.

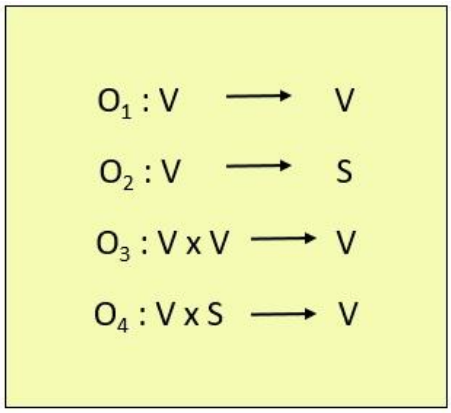
Supercomputers were evolved to deal with billions of floating-point operations/second. Supercomputer optimizes numerical computations (vector computations).

But, along with vector processing supercomputers are also capable of doing scalar processing. Later, **Array processor** was introduced which particularly deals with vector processing, they do not indulge in scalar processing.

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**. TI-ASC, CDC STAR-100, and Cyber-205 use memory-to-memory format for vector instructions.

In **Register-to-Register** vector processor the source operands for instruction, the intermediate result, and the final result all are retrieved from **vector or scalar registers**. Cray-1 and Fujitsu VP-200 use register-to-register format for vector instructions.

Vector Instruction

A vector instruction has the following fields:

1. Operation Code

Operation code indicates the **operation that** has to be performed in the given instruction. It decides the functional unit for the specified operation or reconfigures the multifunction unit.

2. Base Address

Base address field refers to the **memory location** from where the operands are to be fetched or to where the result has to be stored. The base address is found in the memory reference instructions. In the vector instruction, the operand and the result both are stored in the vector registers. Here, the **base address** refers to the designated **vector register**.

3. Address Increment

A vector operand has several data elements and address increment specifies the **address of the next element in the operand**. Some computer stores the data element consecutively in main memory for which the increment is always 1. But, some computers that do not store the data elements consecutively requires the variable address increment.

4. Address Offset

Address Offset is always specified related to the base address. The effective **memory address** is calculated using the address offset.

5. Vector Length

Vector length specifies the **number of elements in a vector operand**. It identifies the **termination** of a vector instruction.

Improving Performance

In vector processing, we come across two overheads setup time and flushing time. When the vector processing is pipelined, the time required to *route the vector operands* to the *functional unit* is called **Set up time**. **Flushing time** is a *time duration* that a vector instruction takes right from its *decoding* until its *first result is out* from the pipeline.

The vector length also affects the efficiency of processing as the longer vector length would cause overhead of subdividing the long vector for processing.

For obtaining the better performance the optimized object code must be produced in order to utilize pipeline resources to its maximum.

1. Improving the vector instruction

We can improve the vector instruction by reducing the memory access, and maximize resource utilization.

2. Integrate the scalar instruction

The scalar instruction of the same type must be integrated as a batch. As it will reduce the overhead of reconfiguring the pipeline again and again.

3. Algorithm

Choose the algorithm that would work faster for vector pipelined processing.

4. Vectorizing Compiler

A vectorizing compiler must regenerate the parallelism by using the higher-level programming language. In advance programming, the four-stage are identified in the development of the parallelism. Those are

Parallel Algorithm(A)

High-level Language(L)

Efficient object code(O)

Target machine code (M)

You can see a parameter in the parenthesis at each stage which denotes the degree of parallelism. In the ideal situation, the parameters are expected in the order A≥L≥O≥M.

Summary

- Computers having vector instruction are vector processors.
- Vector processor have the vector instructions which operates on the large array of integer or floating-point numbers or logical values or characters, all elements in parallel. It is called **vectorization**.

- Vectorization is possible only if the operation performed in parallel are **independent** of each other.
- Operands of vector instruction are stored in the **vector register**. A vector register stores several data elements at a time which is called **vector operand**.
- A vector operand has several **scalar data elements**.
- A vector instruction needs to perform the same operation on the different data set. Hence, vector processors have a **pipelined** structure.
- Vector processing ignores the overhead caused due to the loops while operating on an array.

So, this is how vector processing allows parallel operation on the large arrays and fasten the processing speed.

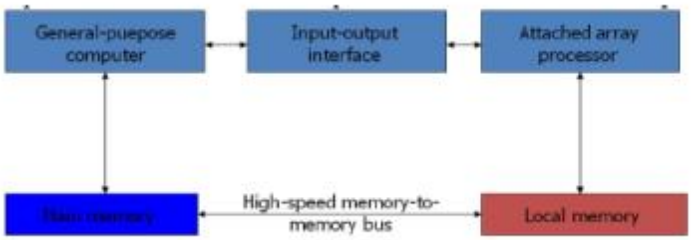
Array Processing

- An array processor is a processor that performs computations on large arrays of data.
- The term is used to refer to two different types of processors.
  - **Attached array processor:**
    - Is an auxiliary processor.
    - It is intended to improve the performance of the host computer in specific numerical computation tasks.
  - **SIMD array processor:**
    - Has a single-instruction multiple-data organization.
    - It manipulates vector instructions by means of multiple functional units responding to a common instruction.

Attached Array Processor

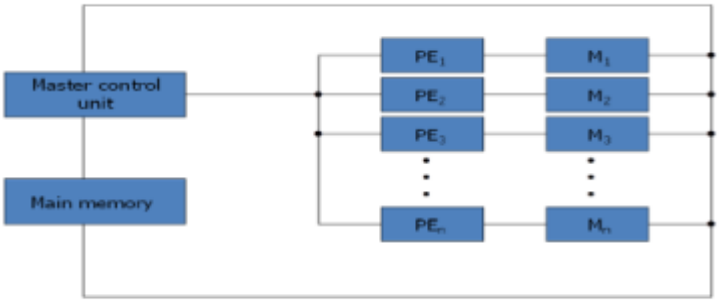
- Its purpose is to enhance the performance of the computer by providing vector processing for complex scientific applications.
  - Parallel processing with multiple functional units

- Fig. shows the interconnection of an attached array processor to a host computer.
- For example, when attached to a VAX 11 computer, the FSP-164/MAX from Floating-Point Systems increases the computing power of the VAX to 100megaflops.
- The objective of the attached array processor is to provide vector manipulation capabilities to a conventional computer at a fraction of the cost of supercomputer.



Attached array processor with host computer

- Masking schemes are used to control the status of each PE during the execution of vector instructions.
  - Each PE has a flag that is set when the PE is active and reset when the PE is inactive.
- Masking schemes are used to control the status of each PE during the execution of vector instructions.
  - Each PE has a flag that is set when the PE is active and reset when the PE is inactive.
- For example, the ILLIAC IV computer developed at the University of Illinois and manufactured by the Burroughs Corp.
  - Are highly specialized computers.
  - They are suited primarily for numerical problems that can be expressed in vector or matrix form.



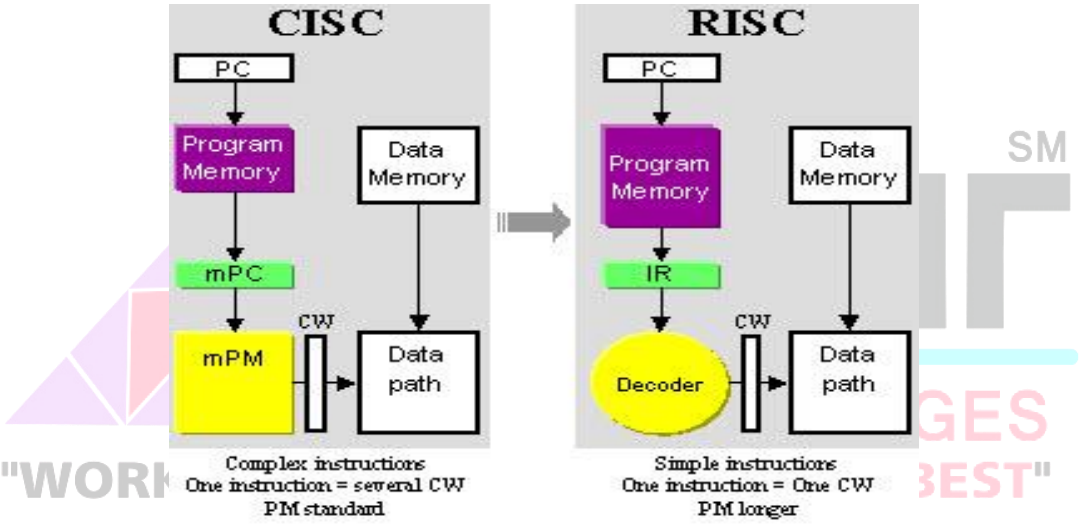
SIMD array processor organization

RISC and CISC

Central Processing Unit Architecture operates the capacity to work from "Instruction Set Architecture" to where it was designed. The architectural designs of CPU are RISC (Reduced instruction set computing) and CISC (Complex instruction set computing). CISC has the ability to execute addressing modes or multi-step operations within one instruction set. It is the design of the CPU where one instruction performs many low-level operations. For example, memory storage, an arithmetic operation and loading from memory. RISC is a CPU design strategy based on the insight that simplified instruction set gives higher performance when combined with a microprocessor architecture which has the ability to execute the instructions by using some microprocessor cycles per instruction.

This article discusses about the RISC and CISC architecture with suitable diagrams.

- Hardware of the Intel is termed as Complex Instruction Set Computer (CISC)
- Apple hardware is Reduced Instruction Set Computer (RISC).
-

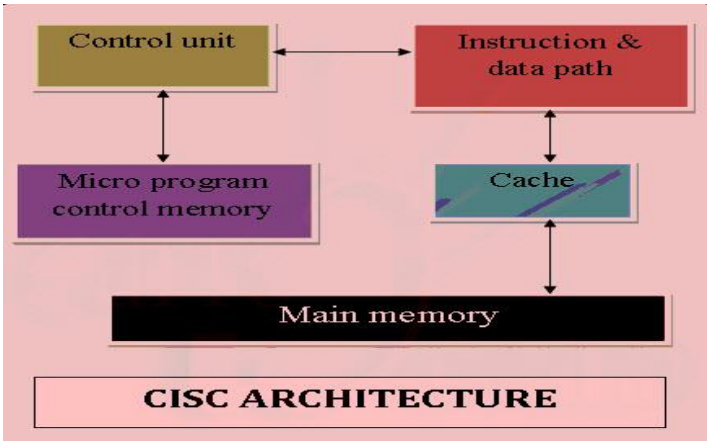


What is RISC and CISC Architecture

Hardware designers invent numerous technologies & tools to implement the desired architecture in order to fulfill these needs. Hardware architecture may be implemented to be either hardware specific or software specific, but according to the application both are used in the required quantity. As far as the processor hardware is concerned, there are 2 types of concepts to implement the processor hardware architecture. First one is RISC and other is CISC.

CISC Architecture

The CISC approach attempts to minimize the number of instructions per program, sacrificing the number of cycles per instruction. Computers based on the CISC architecture are designed to decrease the memory cost. Because, the large programs need more storage, thus increasing the memory cost and large memory becomes more expensive. To solve these problems, the number of instructions per program can be reduced by embedding the number of operations in a single instruction, thereby making the instructions more complex.



- MUL loads two values from the memory into separate registers in CISC.
- CISC uses minimum possible instructions by implementing hardware and executes operations.
- Instruction Set Architecture is a medium to permit communication between the programmer and the hardware. Data execution part, copying of data, deleting or editing is the user commands used in the microprocessor and with this microprocessor the Instruction set architecture is operated.
- The main keywords used in the above Instruction Set Architecture are as below

Instruction Set: Group of instructions given to execute the program and they direct the computer by manipulating the data. Instructions are in the form – Opcode (operational code) and Operand. Where, opcode is the instruction applied to load and store data, etc. The operand is a memory register where instruction applied.

Addressing Modes: Addressing modes are the manner in the data is accessed. Depending upon the type of instruction applied, addressing modes are of various types such as direct mode where straight data is accessed or indirect mode where the location of the data is accessed. Processors having identical ISA may be very different in organization. Processors with identical ISA and nearly identical organization is still not nearly identical.

CPU performance is given by the fundamental law

$$CPU\ Time = \frac{Seconds}{Program} = \frac{Instructions}{Program} X \frac{Cycles}{Instructions} X \frac{Seconds}{Cycle}$$

Thus, CPU performance is dependent upon Instruction Count, CPI (Cycles per instruction) and Clock cycle time. And all three are affected by the instruction set architecture.

	Instruction Count	CPI	Clock
Program	X		
Compiler	X	X	
Instruction Set Architecture	X	X	X
Microarchitecture		X	X
Physical Design			X

Instruction Count of the CPU

This underlines the importance of the instruction set architecture. There are two prevalent instruction set architectures

Examples of CISC PROCESSORS

IBM 370/168 – It was introduced in the year 1970. CISC design is a 32 bit processor and four 64-bit floating point registers.

VAX 11/780 – CISC design is a 32-bit processor and it supports many numbers of addressing modes and machine instructions which is from Digital Equipment Corporation.

Intel 80486 – It was launched in the year 1989 and it is a CISC processor, which has instructions varying lengths from 1 to 11 and it will have 235 instructions.

Characteristics of CISC Architecture

- Instruction-decoding logic will be Complex.
- One instruction is required to support multiple addressing modes.
- Less chip space is enough for general purpose registers for the instructions that are 0operated directly on memory.
- Various CISC designs are set up two special registers for the stack pointer, handling interrupts, etc.
- MUL is referred to as a "complex instruction" and requires the programmer for storing functions.

Advantages of CISC architecture

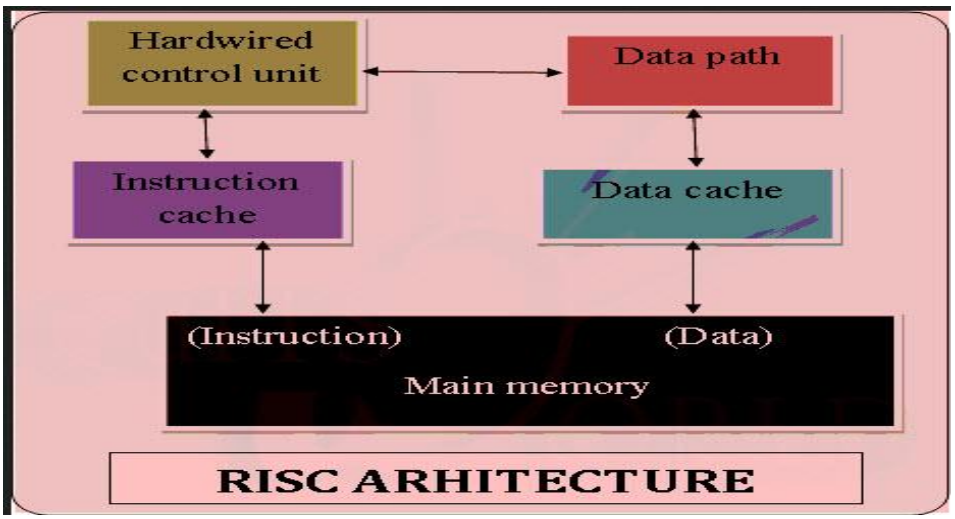
- Microprogramming is easy assembly language to implement, and less expensive than hard wiring a control unit.
- The ease of microcoding new instructions allowed designers to make CISC machines upwardly compatible:
- As each instruction became more accomplished, fewer instructions could be used to implement a given task.

Disadvantages of CISC architecture

- The performance of the machine slows down due to the amount of clock time taken by different instructions will be dissimilar
- Only 20% of the existing instructions is used in a typical programming event, even though there are various specialized instructions in reality which are not even used frequently.
- The conditional codes are set by the CISC instructions as a side effect of each instruction which takes time for this setting – and, as the subsequent instruction changes the condition code bits – so, the compiler has to examine the condition code bits before this happens.

RISC Architecture

RISC (Reduced Instruction Set Computer) is used in portable devices due to its power efficiency. For Example, Apple iPod and Nintendo DS. RISC is a type of microprocessor architecture that uses highly-optimized set of instructions. RISC does the opposite, reducing the cycles per instruction at the cost of the number of instructions per program Pipelining is one of the unique feature of RISC. It is performed by overlapping the execution of several instructions in a pipeline fashion. It has a high performance advantage over CISC.



RISC processors take simple instructions and are executed within a clock cycle

RISC Architecture Characteristics

- Simple Instructions are used in RISC architecture.
- RISC helps and supports few simple data types and synthesize complex data types.
- RISC utilizes simple addressing modes and fixed length instructions for pipelining.
- RISC permits any register to use in any context.
- One Cycle Execution Time
- The amount of work that a computer can perform is reduced by separating "LOAD" and "STORE" instructions.
- RISC contains Large Number of Registers in order to prevent various number of interactions with memory.
- In RISC, Pipelining is easy as the execution of all instructions will be done in a uniform interval of time i.e. one click.
- In RISC, more RAM is required to store assembly level instructions.
- Reduced instructions need a less number of transistors in RISC.
- RISC uses Harvard memory model means it is Harvard Architecture.
- A compiler is used to perform the conversion operation means to convert a high-level language statement into the code of its form.

The Advantages of RISC architecture

- RISC(Reduced instruction set computing)architecture has a set of instructions, so high-level language compilers can produce more efficient code
- It allows freedom of using the space on microprocessors because of its simplicity.
- Many RISC processors use the registers for passing arguments and holding the local variables.

- RISC functions use only a few parameters, and the RISC processors cannot use the call instructions, and therefore, use a fixed length instruction which is easy to pipeline.
- The speed of the operation can be maximized and the execution time can be minimized. Very less number of instructional formats, a few numbers of instructions and a few addressing modes are needed.

The Disadvantages of RISC architecture

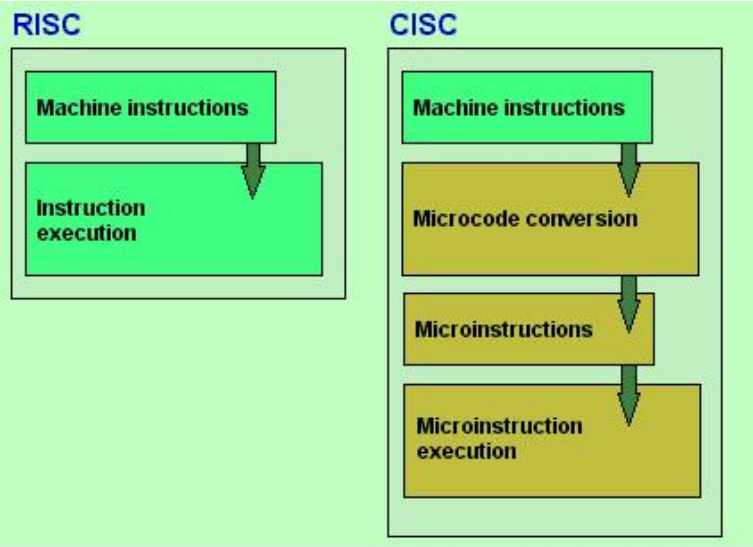
- Mostly, the performance of the RISC processors depends on the programmer or compiler as the knowledge of the compiler plays a vital role while changing the CISC code to a RISC code
- While rearranging the CISC code to a RISC code, termed as a code expansion, will increase the size. And, the quality of this code expansion will again depend on the compiler, and also on the machine's instruction set.
- The first level cache of the RISC processors is also a disadvantage of the RISC, in which these processors have large memory caches on the chip itself. For feeding the instructions, they require very fast memory systems.

RISC & CISC Comparison

CISC	RISC
It is prominent on Hardware	It is prominent on the Software
It has high cycles per second	It has low cycles per second
It has transistors used for storing Instructions which are complex	More transistors are used for storing memory
LOAD and STORE memory-to-memory is induced in instructions	LOAD and STORE register-register are independent
It has multi-clock	It has a single - clock

Comparison between CISC & RISC

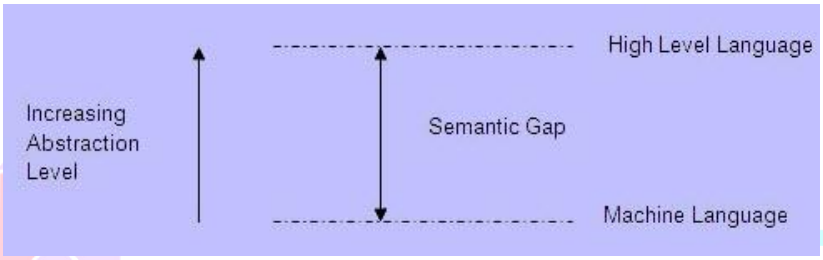
MUL instruction is divided into three instructions
"LOAD" – moves data from the memory bank to a register
"PROD" – finds product of two operands located within the registers
"STORE" – moves data from a register to the memory banks
The main difference between RISC and CISC is the number of instructions and its complexity.



RISC Vs CISC

SEMANTIC GAP

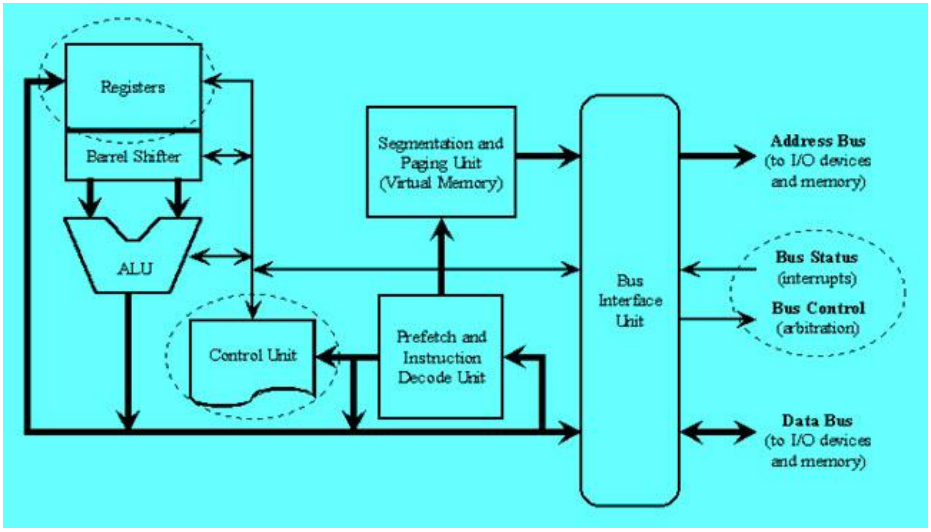
Both RISC and CISC architectures have been developed as an attempt to cover the semantic gap.



Semantic Gap

With an objective of improving efficiency of software development, several powerful programming languages have come up, viz., Ada, C, C++, Java, etc. They provide a high level of abstraction, conciseness and power. By this evolution the semantic gap grows. To enable efficient compilation of high level language programs, CISC and RISC designs are the two options.

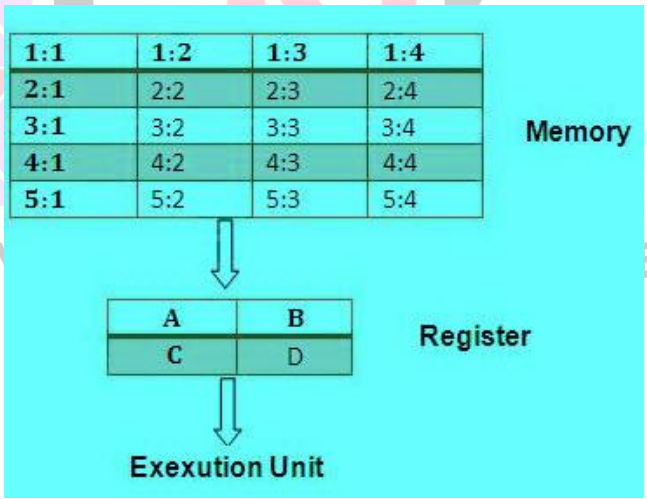
CISC designs involve very complex architectures, including a large number of instructions and addressing modes, whereas RISC designs involve simplified instruction set and adapt it to the real requirements of user programs.



CISC and RISC Design

Multiplication of two Numbers in Memory

If the main memory is divided into areas that are numbered from row1:column 1 to row 5 :column 4. The data is loaded into one of four registers (A, B, C, or D). To find multiplication of two numbers- One stored in location 1:3 and other stored in location 4:2 and store back result in 1:3.



Multiplication of Two Numbers

Multi Core Processor

Single-Core Processor

A single-core processor machine as shown in Figure 1, consists of one processor, two or more levels of cache memory, main memory, hard disk, and Input/Output (I/O) devices. Levels of cache relates to the size and distance from the processor as shown in Figure 2 which displays the memory hierarchy, for example accessing data from Level 1 (L1) cache is faster than accessing it from L2 cache, and so on. Consequently, the use of cache memory reduces the Memory Access Time (MAT) resulting in a better performance.

According to Moor's law that was stated in 1965, the number of transistors on a chip will roughly double each year, then he refined the period in 1975 to be two years. Moore's law is often quoted as Dave House's revision that computer performance will double every 18 months. The problem of adding more transistors on a chip in the amount of generated heat that exceeds the advancements rate of the cooling techniques which is known as “the power wall” problem.

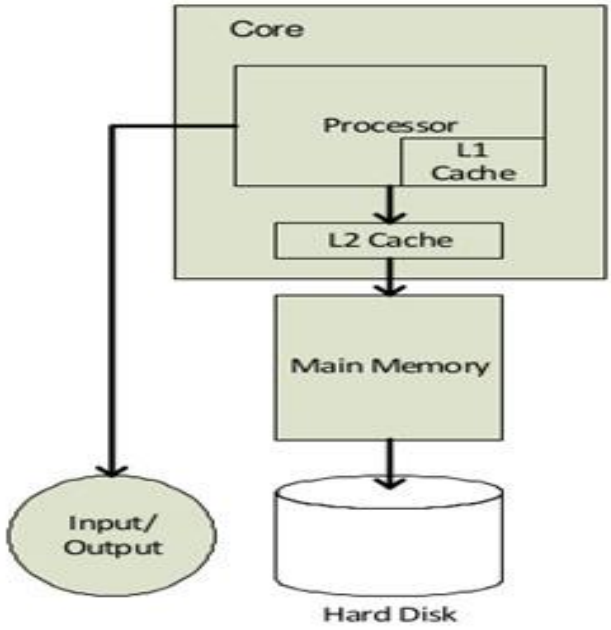


Figure 1. Single-core processor machine

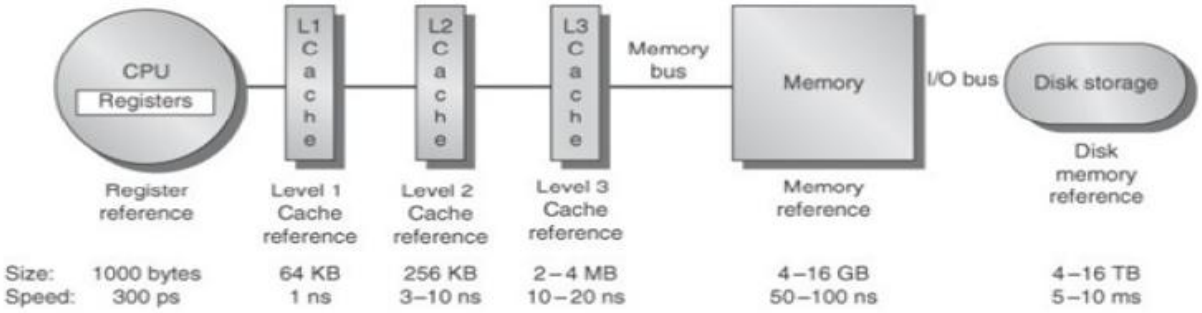


Figure 2. Memory hierarchy

Multi-core Processor

A multi-core processor is an integrated circuit (IC) to which two or more processors have been attached for enhanced performance, reduced power consumption, and more efficient simultaneous processing of multiple tasks, it is a growing industry trend as single-core processors rapidly reach the physical limits of possible complexity and speed. A basic block diagram of a generic multi-core processor is shown in Figure 3.

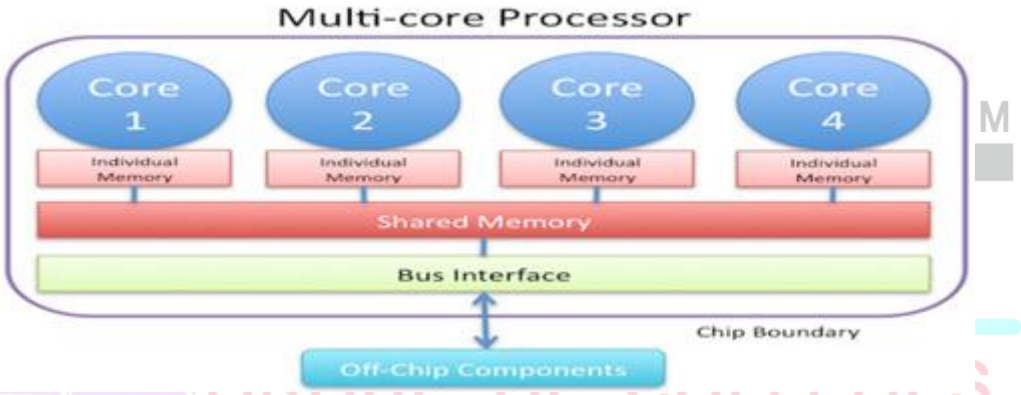


Figure 3. Block diagram for a general Multi-Core processor

The high performance demand of users also motivated the shift from single-core to multi-core processors. A comparison between a single-core and a multi-core processors that occupies the same die area is shown in Figure 4.

	Single core processor (45nm)	Multi-core processor (45nm)
Vdd	1.0V	1.0V
I/O pins(total)	1280 (ITRS)	3000 (Estimated)
Operating frequency	7.8GHz	4GHz
Chip-package data rate	7.8 Gb/s	4Gb/s
Bandwidth	125GByte/s	1 TeraByte/s
Power	429.78W	107.39W
Total number of pins on chip	3840	9000(Estimated)
Number of pins on the package	2480	4500(Estimated)

Figure 4. A comparison between single-core vs. multi-core processors

Issues in Developing Multi-Core Processors Machines

The first issue is the communication between core processors and the main memory in a multicore processors' environment. This is done either by the use of a single communication bus "shared memory model" or an interconnection network "distributed memory model" as shown in Figure 5. The single bus approach has an upper limit of 32 cores, after that the bus will be overfilled with transactions that lower the performance of the system.

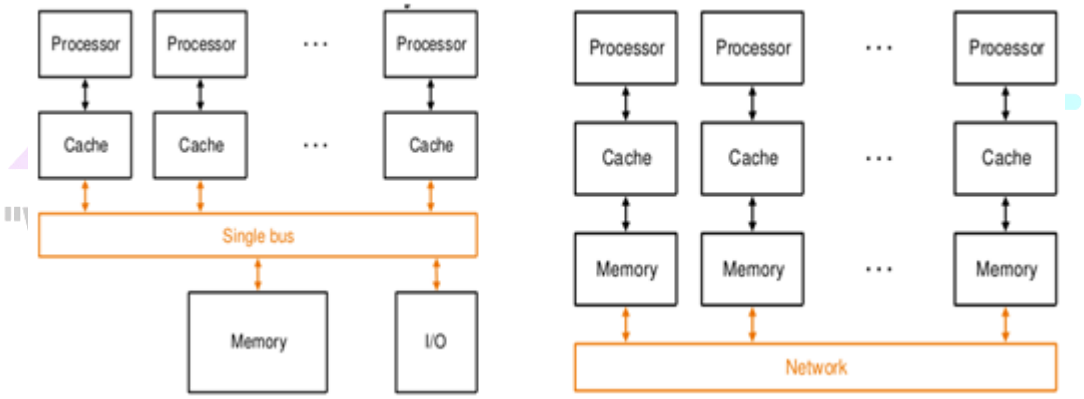


Figure 5. Shared memory approach (right) vs. distributed memory model (left)

Since every core processor has its own memory in the distributed memory model, a copy of the data might not be always the most updated version, which will result in a cache coherence problem. For example, if we have a dual core processor, each core will get a portion of the memory, if the first core writes a new value for a parameter and the second core had to read the value of this parameter it will read its own value unless there is a coherence policy. Rreading a non consistent value of this parameter may result in a program crash. There are two schemes that forces cache coherence, the snooping protocol and a directory protocol. The snooping protocol is

designed only for a bus based system, it uses a number of states to determine whether or not there is a need to update the cache entries or not and also if it has control over writing to the block. However, the directory protocol has the scalability to work on any arbitrary network. A directory is used to hold information about which of the memory locations are being used exclusively by one core, and which are shared among multiple cores.

The second issue that rise in order to fully utilize the multi-core processor technology is parallelism, programs should have the characteristic of being executed in a parallel order. There are three types of parallelism: Instruction level parallelism, thread level parallelism, and data level parallelism. In the case of Instruction level parallelism, the execution of the instructions could be done in a parallel way as well as in a sequential way. In the case of thread level parallelism, multiple threads of the same task are presented to the processor to be executed simultaneously as shown in Figure 6. In the case of data level parallelism, common data is being shared among executing processes through memory coherence, which will improve performance by reducing the time required to load and access memory. However, according to Amdahl's law the performance of Parallel applications in a multi-core environment is limited by its non-parallel part that form bottlenecks. So, for an optimal use of the multi-core processors the non-parallel part has to be optimized by either parallelizing the non-parallel part or by making them faster using more efficient algorithms.

Starvation is a problem that could occur if the program isn't designed in a parallel way, this is because one or more cores might starve for data. For example, if a single-threaded application is to be run in a multi-core processor machine. The thread will run in one of the cores while the other cores remains idle. Another example could be seen in a multi-core processor machine with a shared cache such as the Intel Core 2 Duo's shared L2 cache, unless a proper replacement policy was placed one core may starve for cache usage and keeps making a costly calls out to main memory. The replacement policy should have a mechanism for ejecting cache entries that other cores have recently loaded. In the case where the number of cores is large, applying this replacement policy becomes difficult to reduce the amount of ejected cache space without increasing cache misses.

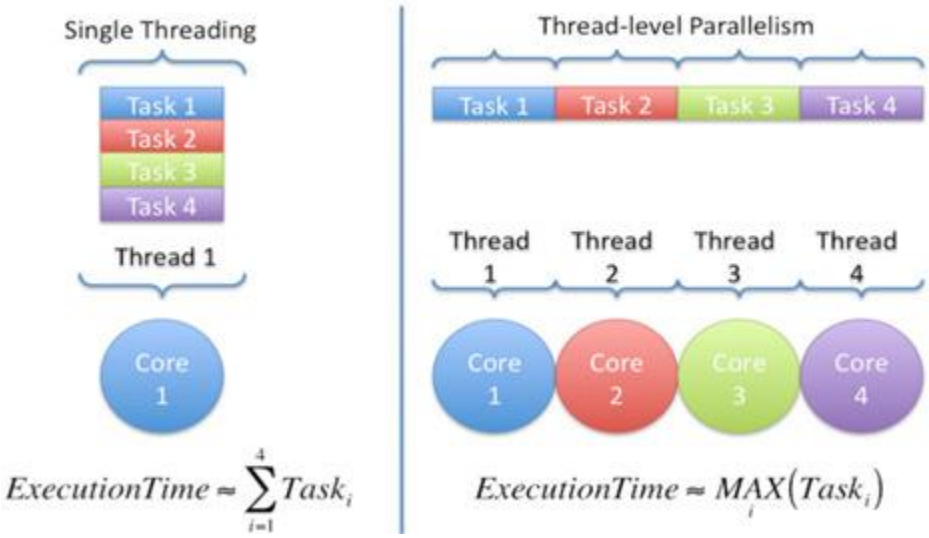


Figure 6. Thread level parallelism

The third issue that rise in the development of multi-core processors is power dissipation. If we allocate two cores on the same chip size a large amount of heat will be generated unless there is a power control mechanism that shuts down the unused core or limits its power consumption.

Finally, the fourth issue that rise in the development of multi-core processors is whether to use homogeneous or heterogeneous cores. Homogeneous cores are all exactly the same, they run on an equivalent frequencies, have the same cache sizes and functionalities. However, eterogeneous cores are different in their frequencies, memory models and functionalities. The choice will be based on making a trade-off between processor complexity and customization. The production of homogeneous cores are easier since all cores contains the same hardware and use the same instruction set. While in the case of heterogeneous cores, each core could have a specific function and run its own specialized instruction set. For example, the CELL processor has heterogeneous cores, one Power Processing Element (PPE) and eight Synergistic Processing Elements (SPE).

The PPE core is used as a large centralized processing unit, while the other PPEs are used for different functionalities such as graphics, communications, enhanced mathematics, audio, and so on. The heterogeneous model is more complex, but may have efficiency, power, and thermal benefits that outweigh its complexity.

4. PERFORMANCE MEASUREMENT OF A PROCESSOR

There are many metrics which could be used in measuring processors performance such as: throughput which is the average rate of how many processes were executed successfully; response time which is the time between the time between the request time and time that the

system starts working on this request; execution time which is the time needed to finish the request, energy consumption, and the memory bandwidth which is the rate of data sustained from
the CPU core to the RAM. All these metrics could be classified into three main categories: higher is better, lower is better, and nominal is better. For example, higher throughput is better, lower execution time is better, and nominal power consumption is better .

COMPARING TWO MULTI-CORE PROCESSORS MACHINES

In this section a comparison between two multi-core processors machines will be made, the Intel core i7-4960X processor (Ivy Bridge E) and the AMD Phenom II X6 processor.

INTEL CORE I7-4960X PROCESSOR (IVY BRIDGE E)

Intel uses a naming scheme for its Core processors, the three brands are, Core i3, Core i5 and Core i7, these brand names aren't related to specific technical features such as the number of cores. Instead, they correspond to three, four and five stars in Intel's Intel Processor Rating from low-level (Core i3), through mid-range (Core i5) to high-end performance (Core i7), following on from the entry-level Celeron (one star) and Pentium (two stars) processors. The Intel brand name of Intel Core i7 applies to several families of desktop and laptop 64-bit x86-64 processors using the Nehalem, Westmere, Sandy Bridge, Ivy Bridge and Haswell micro architectures. Both the regular Core i7 and the Extreme Edition are advertised as five stars in the Intel Processor Rating. A comparison of the Core i7 family can be found in. One of Intel's latest Core i7 models in the market is the Core i7-4960X processor (Ivy Bridge E), its die details' diagram can be found in , and its technical specifications are :

- 6 cores
- 12 threads
- Clock speed of 3.6 GHz
- Max turbo frequency is 4 GHz
- Level 1 cache (32 KB (code) / 32 KB (data)) per core
- Level 2 cache 256 KB
- Level 3 cache 15360 KB shared for all cores
- Instruction set of 64 bits
- Thermal Design Power (TDP) of 130W
- Max Temperature is 66.8 C
- Introduction date (September 10, 2013)
- Die size is 22nm
- supports 4 memory channels
- supports DDR3-1066, DDR3-1333, DDR3-1600, DDR3-1866

5.2. AMD PHENOM II X6 1100T

Phenom II is a family of AMD's multi-core 45nm processors using the AMD K10 micro architecture, succeeding the original Phenom. Advanced Micro Devices released the Socket. AM2+ version of Phenom II in December 2008, while Socket AM3 versions with DDR3 support, along with an initial batch of triple- and quad-core processors were released on February 9, 2009. Dual-processor systems require Socket F+ for the Quad FX platform. The second-generation of Phenom II X6 was released on April 27, 2010.

One of the latest AMD Phenom processors in the market is AMD Phenom II X6 1100T, its technical specifications are:

- 6 cores
- 6 threads
- Clock speed of 3.3 GHz
- Max turbo frequency is 3.7 GHz
- Level 1 cache (64 KB (code) / 64 KB (data)) per core
- Level 2 cache 512 KB per core
- Level 3 cache 6144 KB shared for all cores
- Instruction set of 64 bits
- Thermal Design power of 125W
- Max Temperature is 62 C
- Die size 45nm
- supports 2 memory channels
- supports DDR2-1066, DDR3-1333

Question Bank

1 (a) Write comparison of RISC and CISC Processor

(b) Explain vector Processing and array processing

2 (a) Give Definition of vector Processing and enlist the application

(b) Define Multiprocessor. Explain clearly the characteristics of multiprocessor.

3 (a) With the help of suitable diagrams, explain simplex, half duplex and full duplex transmission.

(b) What is pipelining? Draw a space time diagram for a six segment pipeline showing the time it takes to process eight tasks.

4 (a) What is parallel Processing? What its advantage

(b) What are the advantages of pipelining?

5 (a) Why does pipeline improve performance?

(b) What are the pipeline hazards? How do they affect the speedup?

6 (a) Explain SIMD array processing along with the architectural diagram.

(b) What is Inter connection Network?

7 (a) Write down different characteristics on multiprocessor.

(b) What is cache coherence and why is it important in shared memory multiprocessor system?

8 (a) Draw a four segment pipeline.

(b) Differentiate Instruction and arithmetic pipeline

9 (a) Explain the operation of a multiprocessor system with multiport memory.

(b) Discuss the factors which affect the performance of pipelining processor based system

10. Write short notes

1. Structure of multiprocessor
2. AMD Processor
3. Intel Processor
4. superscalar processors.
5. Interprocessor