Parul® University
Vadodara, Gujarat

NAAC GRADE A++

HIGH PERFORMANCE COMPUTING

Study Guide

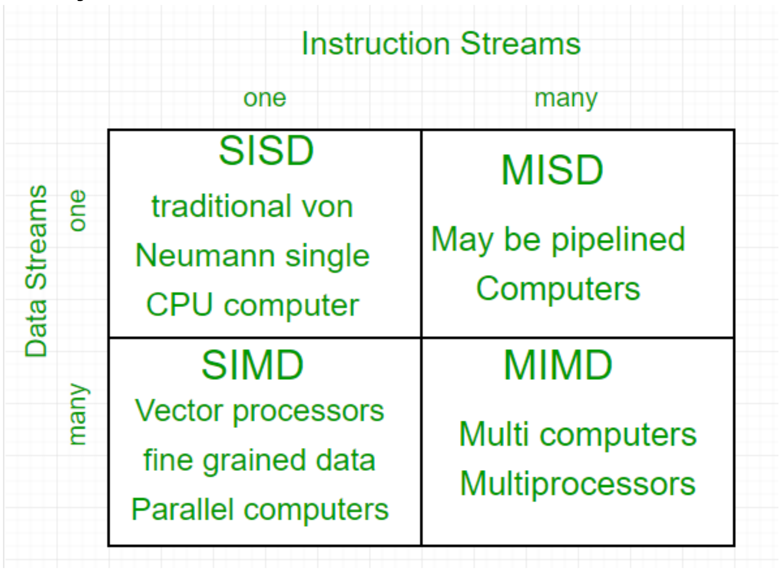
DEEPIKA PANDEY
AI&DS
Parul University,PIET

Parul® University Vadodara, Gujarat NAAC GRADE A++

INDEX

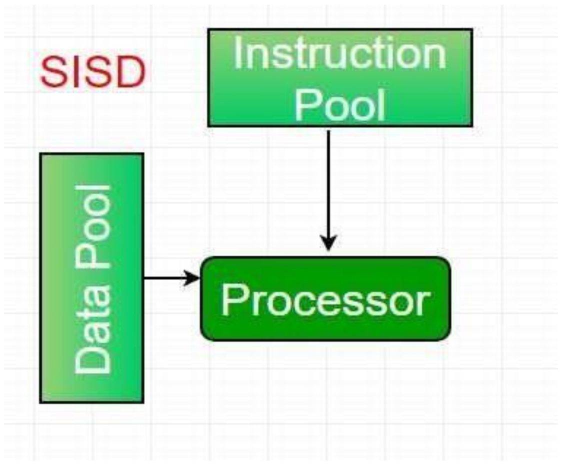
- Moore’s Law:
  Moore’s Law is the observation that the number of transistors on an integrated circuit will double every two years with minimal rise in cost.

- Flynn’s Taxonomy:
  Flynn's taxonomy is a classification of computer architectures, proposed by Michael J. Flynn in 1966 and extended in 1972.



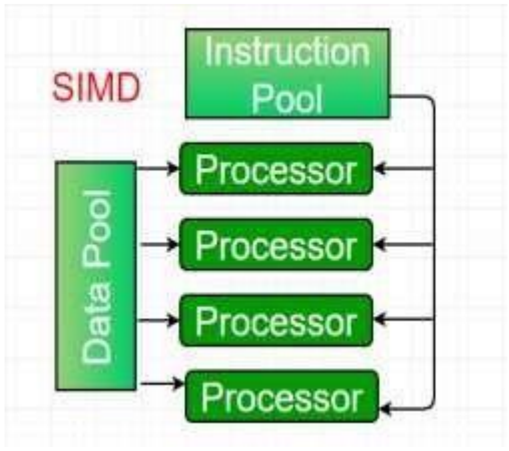
Flynn’s classification –

1. **Single-instruction, single-data (SISD) systems –**
   An SISD computing system is a uniprocessor machine which is capable of executing a single instruction, operating on a single data stream. In SISD, machine instructions are processed in a sequential manner and computers adopting this model are popularly called sequential computers. Most conventional computers have SISD architecture. All the instructions and data to be processed have to be stored in primary memory.

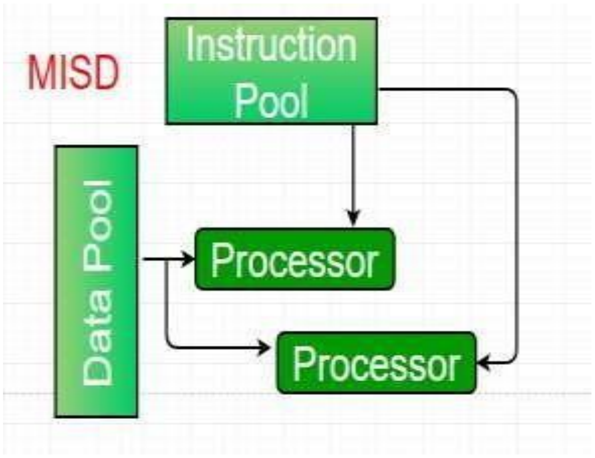


2. **Single-instruction, multiple-data (SIMD) systems** – An SIMD system is a multiprocessor machine capable of executing the same instruction on all the CPUs but operating on different data streams.

Machines based on an SIMD model are well suited to scientific computing since they involve lots of vector and matrix operations. So that the information can be passed to all the processing elements (PEs) organized data elements of vectors can be divided into multiple sets(N-sets for N PE systems) and each PE can process one data set.



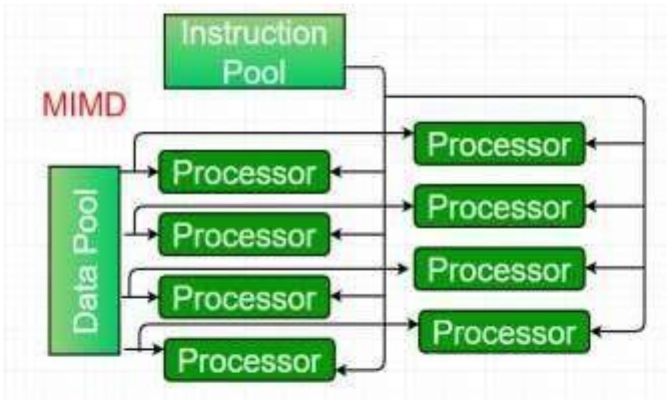
1. **Multiple-instruction, single-data (MISD) systems** – An MISD computing system is a multiprocessor machine capable of executing different instructions on different PEs but all of them operating on the same dataset.



2. **Multiple-instruction, multiple-data (MIMD) systems –**

An MIMD system is a multiprocessor machine which is capable of executing multiple instructions on multiple data sets. Each PE in the MIMD model has separate instruction and data streams; therefore machines built using this model are capable to any kind of application. Unlike SIMD and

MISD machines, PEs in MIMD machines work asynchronously.



- Throughput: Throughput is a measure of how many units of information a system can process in a given amount of time.
- Latency: Memory latency is the time between initiating a request for a byte or word in memory until it is retrieved by a processor.

- Types of Parallelism:

What is Implicit Parallelism?

Implicit Parallelism is defined as a parallelism technique where parallelism is automatically exploited by the compiler or interpreter. The objective of implicit parallelism is the parallel execution of code in the runtime environment. In implicit parallelism, parallelism is being carried out without explicitly mentioning how the computations are parallelized. The compiler assigns the resources to target machines for performing parallel operations. Implicit parallelism requires less programming effort and has applications in shared memory multiprocessors.

Examples of Implicit Parallelism

- **Pipelining:** Pipelining is an example of implicit parallelism that is being used by the processors to execute multiple instructions simultaneously. In the process of pipelining each stage of execution is performed in parallel with other instructions.
- **Multithreading:** Multithreading is defined as a process of

executing multiple threads in a single process. Every process has its own instructions and can be executed independently of the other threads. Multithreading is commonly used in parallel processing applications.

- **Vectorization:** Vectorization is the technique used for optimizing the code in order to execute it on the processors that supports Single Instruction and Multiple Data Instructions. It allows performing the same operation on multiple data elements at the same time.

- **Out-of-Order:** Out-of-Order is a technique being used by the processors for the execution of instructions that maximizes parallelism. In this technique, instructions are executed as soon as the required operands are available instead of the order in which the operands appear in the program.

What is Explicit Parallelism?

Explicit Parallelism is defined as a parallelism technique where concurrent operations are executed parallel with the help of primitives that are known as special purpose directives or function calls. In explicit parallelism. In Explicit parallelism, the compiler does not detect the parallelism for the allocation of resources. The programming efforts required in explicit parallelism are more as compared to implicit parallelism as the execution of concurrent tasks takes place manually depending upon the source code developed by the programmer. The resources are being utilized more efficiently in explicit parallelism and have applications in loosely coupled multiprocessors.

Examples of Explicit Parallelism

- **Parallel loops:** Parallel loops are defined as a construct that allows a loop to be executed in parallel. The sequence and condition of the loops are specified according to which they are executed.

- **Message passing:** Message passing is a technique that is used to enable communication that takes place between processes and the threads that are running on different processors.

- **Data parallelism:** Data parallelism is defined as a technique that is used to divide the present large datasets into smaller subsets. These divided subsets can then be processed in parallel.

- **Implicit Parallelism vs Explicit Parallelism**

Parameter	**Implicit Parallelism**	**Explicit Parallelism**
Definition	Implicit Parallelism is defined as characteristics of parallel programming that	Explicit Parallelism is defined as characteristics of parallel programming that execute the

Parameter	Implicit Parallelism	Explicit Parallelism
	automatically allow the compiler or interpreter to exploit the parallelism.	concurrent computations with the help of primitives that are in the form of special purpose directives.
Programming Languages used	Implicit Parallelism makes use of conventional programming languages such as C, C++, and Fortran for writing the source code.	Explicit Parallelism requires more programming efforts and makes use of programming languages such as C, C++, Fortran, and Pascal.
Compilation of source code	In Implicit Parallelism, the source program is coded sequentially and then translated into parallel object code by a parallelizing compiler.	In Explicit parallelism, parallelism is explicitly specified in the source code itself.
Resource Allocation	In Implicit Parallelism, parallelism is detected by the **Applications**	In explicit parallelism, as parallelism is specified compiler and then assigns the resources to the target machine code.
Programming efforts		Implicit parallelism requires fewer programming efforts by the programmers as compared to explicit parallelism.
Resource utilization		In Implicit parallelism, resource utilization is less efficient because resource allocation is done by the compiler according to the need.
Scalability		Implicit parallelism is less scalable due to system control.
		Implicit Parallelism is used in shared memory multiprocessors.

explicitly, there is no need for the compiler to detect parallelism, and resources are allocated explicitly. Explicit parallelism requires more programming efforts by the programmers as compared to implicit parallelism. In Explicit parallelism, resource utilization is more efficient because resources are	allocated explicitly and make used of resources more efficiently. Explicit parallelism is more scalable as it has programmer control. Explicit parallelism is used in loosely coupled multiprocessors.

- Levels of Parallelism

  - **Bit-level parallelism –**

    Example: Consider a scenario where an 8-bit processor must compute the sum of two 16-bit integers. It must first sum up the 8 lower-order bits, then add the 8 higher-order bits, thus requiring two instructions to perform the operation. A 16-bit processor can perform the operation with just one instruction.

  - **Instruction-level parallelism –**

    A processor can only address less than one instruction for each clock cycle phase. These instructions can be re-ordered and grouped which are later on executed concurrently without affecting the result of the program. This is called instruction-level parallelism.

  - **Task Parallelism –**

    Task parallelism employs the decomposition of a task into subtasks and then allocating each of the subtasks for execution. The processors perform the execution of sub-tasks concurrently.

  - **Data-level parallelism (DLP) –**

    Instructions from a single stream operate concurrently on several data Limited by non-regular data manipulation patterns and by memory bandwidth.

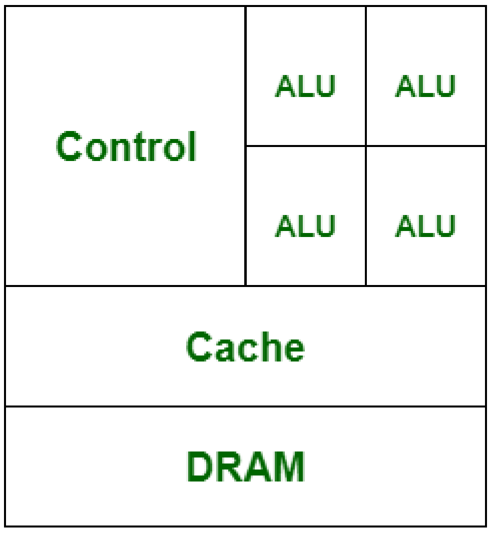
- CPU Vs GPU:

  Central Processing Unit (CPU): CPU is known as brain for every ingrained system. CPU comprises the arithmetic logic unit (ALU) accustomed quickly to store the information and perform calculations and Control Unit (CU) for performing instruction sequencing as well as branching. CPU interacts with more computer components such as memory, input and output for performing instruction.

Graphics Processing Unit (GPU): GPU is used to provide the images in computer games. GPU is faster than CPU’s speed and it emphasis on high throughput. It’s generally incorporated with electronic equipment for sharing RAM with electronic equipment that is nice for the foremost computing task. It contains more ALU units than CPU.

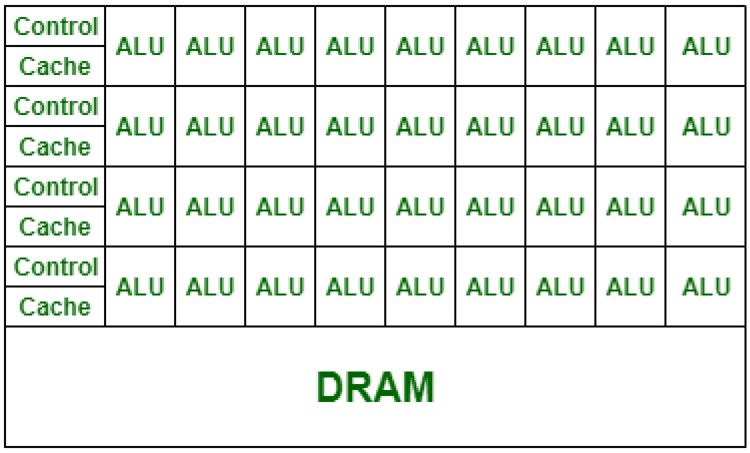
The basic difference between CPU and GPU is that CPU emphasis on low latency. Whereas, GPU emphasis on high throughput. Let's see the difference between CPU and GPU:

Central Processing Unit (CPU): CPU is known as brain for every ingrained system. CPU comprises the arithmetic logic unit (ALU) accustomed quickly to store the information and perform calculations and Control Unit (CU) for performing instruction sequencing as well as branching. CPU interacts with more computer components such as memory, input and output for performing instruction.



CPU

Graphics Processing Unit (GPU): GPU is used to provide the images in computer games. GPU is faster than CPU's speed and it emphasis on high throughput. It's generally incorporated with electronic equipment for sharing RAM with electronic equipment that is nice for the foremost computing task. It contains more ALU units than CPU.



GPU

The basic difference between CPU and GPU is that CPU emphasis on low latency. Whereas, GPU emphasis on high throughput. Let's see the difference between CPU and GPU:

S.NO	CPU	GPU
1.	CPU stands for Central Processing Unit.	While GPU stands for Graphics Processing Unit.
2.	CPU consumes or needs more memory than GPU.	While it consumes or requires less memory than CPU.
3.	The speed of CPU is less than GPU's speed.	While GPU is faster than CPU's speed.
4.	CPU contain minute powerful cores.	While it contains more weak cores.
5.	CPU is suitable for serial instruction processing.	While GPU is not suitable for serial instruction processing.
6.	CPU is not suitable for parallel instruction processing.	While GPU is suitable for parallel instruction processing.
7.	CPU emphasis on low latency.	While GPU emphasis on high throughput.

• Real life applications of HPC:

High Performance Computing (HPC) is a term used to describe the use of supercomputers and parallel processing strategies to carry out difficult calculations and data analysis activities. From scientific research to engineering and industrial design, HPC is employed in a wide range of disciplines and applications. Here are a few of the most significant HPC use cases and applications:

1. **Scientific research:** HPC is widely utilized in this sector, especially in areas like physics, chemistry, and astronomy. With standard computer techniques, it would be hard to model complex physical events, examine massive data sets, or carry out sophisticated calculations.
2. **Weather forecasting:** The task of forecasting the weather is difficult and

data-intensive, requiring sophisticated algorithms and a lot of computational power. Simulated weather models are executed on HPC computers to predict weather patterns.

3. **Healthcare:** HPC is being used more and more in the medical field for activities like medication discovery, genome sequencing, and image analysis. Large volumes of medical data can be processed by HPC systems rapidly and accurately, improving patient diagnosis and care.
4. **Energy and environmental studies:** HPC is employed to simulate and model complex systems, such as climate change and renewable energy sources, in the energy and environmental sciences. Researchers can use HPC systems to streamline energy systems, cut carbon emissions, and increase the resilience of our energy infrastructure.
5. **Engineering and Design:** HPC is used in engineering and design to model and evaluate complex systems, like those found in vehicles, buildings, and aeroplanes. Virtual simulations performed by HPC systems can assist engineers in identifying potential problems and improving designs before they are built.

- Memory Architectures:

  Uniform Memory Access (UMA):
  In UMA, where Single memory controller is used. Uniform Memory Access is slower than non-uniform Memory Access. In Uniform Memory Access, bandwidth is restricted or limited rather than non-uniform memory access. There are 3 types of buses used in uniform Memory Access which are: Single, Multiple and Crossbar. It is applicable for general purpose applications and time-sharing applications.

Advantages of UMA:

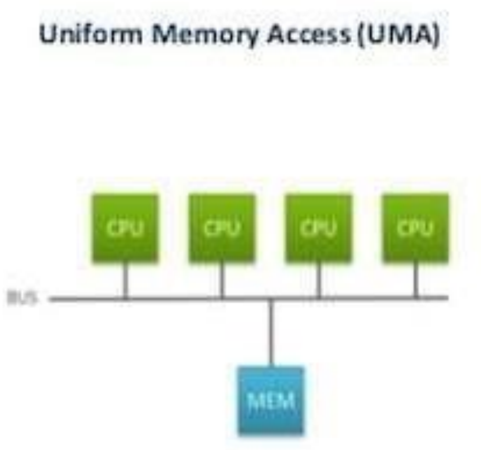
- Easy to Implement: UMA architecture is relatively easy to implement, as all processors or cores have equal access to the memory pool. This makes it an ideal choice for small-scale systems, such as desktop computers or low-end servers.
- Low Latency: Since all memory locations have equal access times, UMA provides low latency, which ensures that processors or cores can access memory quickly and efficiently. This makes UMA ideal for high-performance computing applications that require fast memory access.
- Low Cost: UMA architecture is relatively inexpensive to implement, as it requires only a single shared memory bus to connect all processors or cores

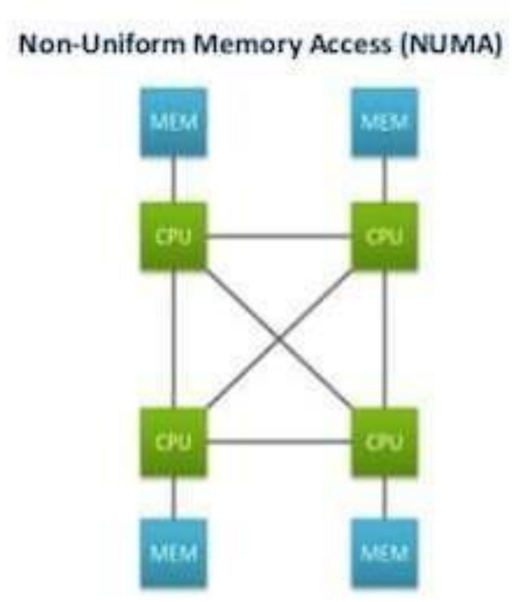
to the memory pool. This makes it an ideal choice for low-cost computing systems.

Non-uniform Memory Access (NUMA):
In NUMA, where different memory controller is used. Non-uniform Memory Access is faster than uniform Memory Access. Non-uniform Memory Access is applicable for real-time applications and time-critical applications.

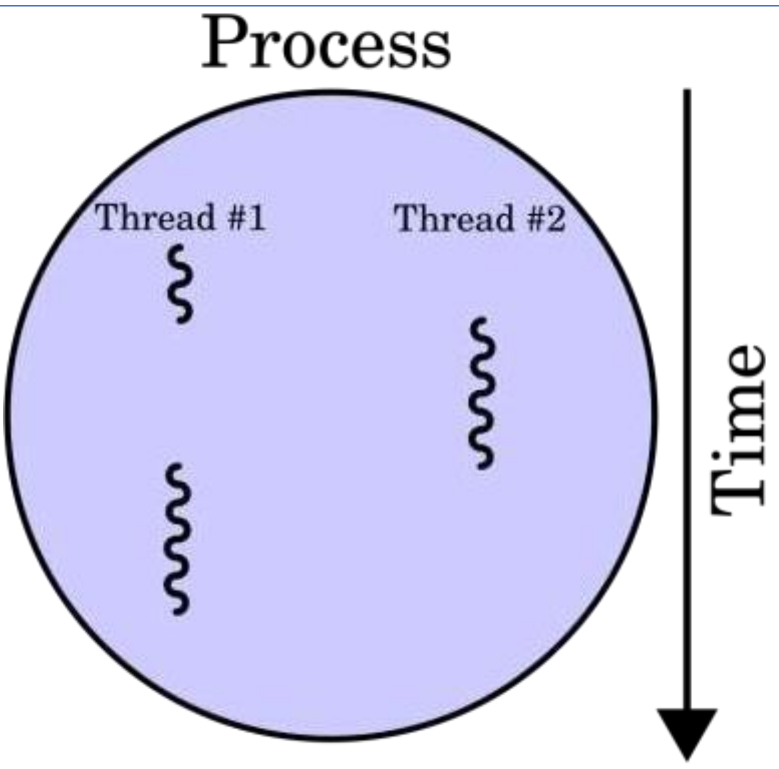
Advantages of a NUMA :

- Improved performance: By providing each processor with its own local memory, NUMA can reduce memory access times and improve overall system performance.
- Scalability: NUMA systems are highly scalable and can handle large workloads by adding additional processors and memory nodes.
- Reduced memory contention: NUMA can help reduce memory contention by allowing each processor to access its own local memory, reducing the need for multiple processors to access the same memory location.

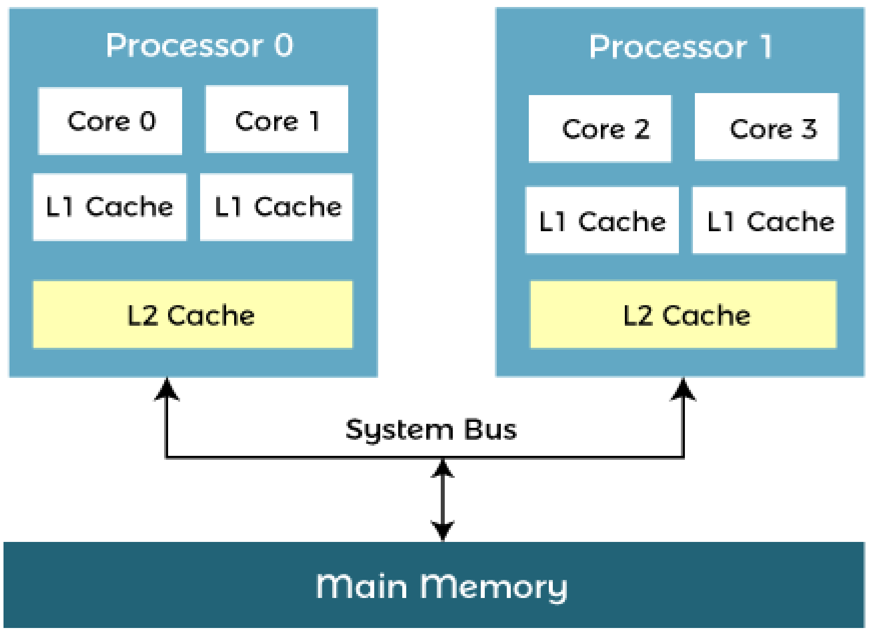




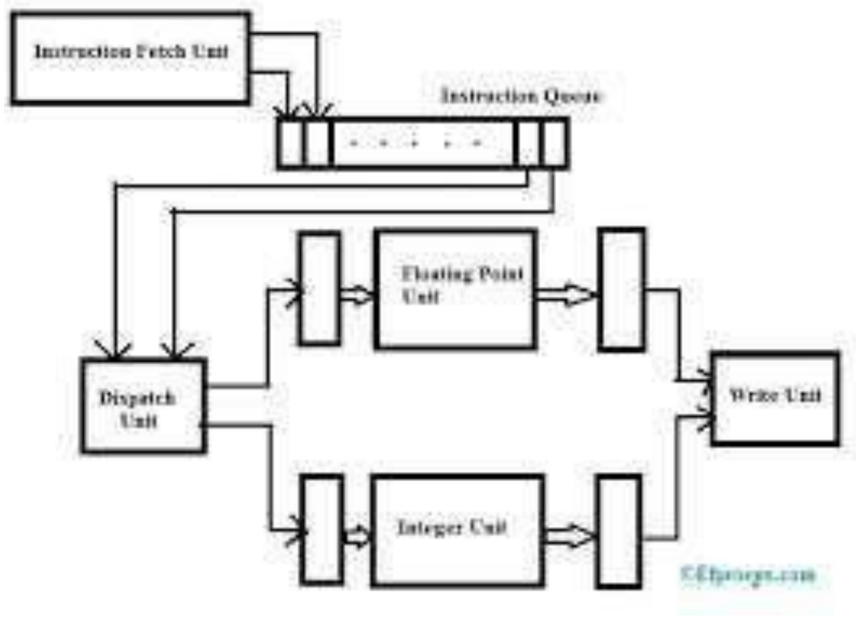
- Multithreading Architecture:



- Multicore architecture:



- N-wide Super Scalar Architecture:

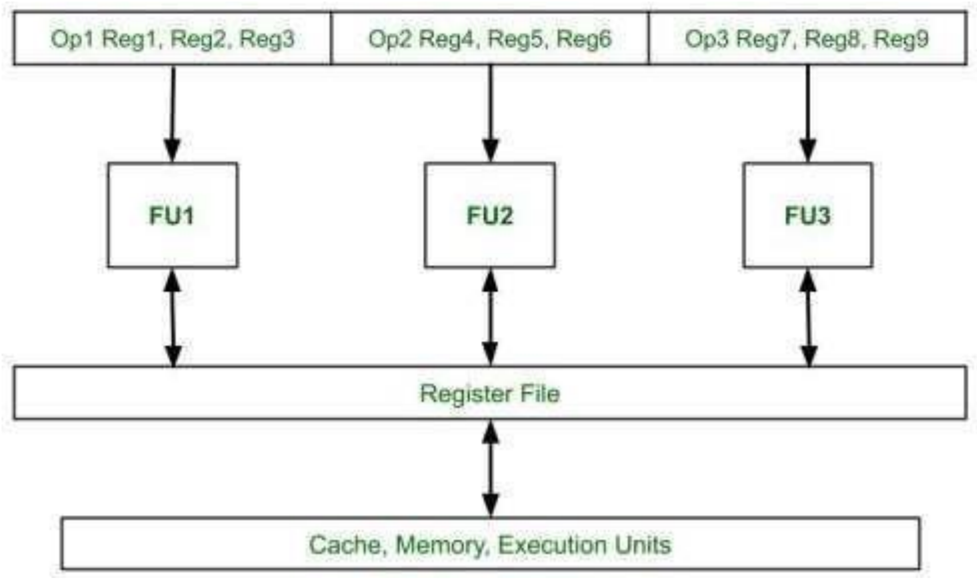


Superscalar processor architecture mainly includes parallel execution units where these units can implement instructions simultaneously. So first, this parallel architecture was implemented within a RISC processor that utilizes simple & short instructions to execute calculations. So due to their superscalar abilities, normally RISC processors have performed better as compared to CISC processors which run at the same megahertz. But, most CISC processors now like the Intel Pentium comprise some RISC architecture also, which allows them to perform instructions in parallel.

The superscalar processor is equipped with several processing units for handling various instructions in parallel in every processing stage. By using the above architecture, a number of instructions start execution within a similar clock cycle. These processors are capable of obtaining an instruction execution output of the above one instruction for each cycle.

In the above architecture diagram, a processor is used with two execution units where one is used for integer & other one is used for the operations of floating point. The instruction fetch unit (IFU) is capable of instructions reading at a time & stores them within the instruction queue. In every cycle, the dispatch unit fetches & decodes up to 2 instructions from the queue front. If there is a single integer, single floating point instruction & no hazards, then both instructions are dispatched within a similar clock cycle.

VLIW Architecture:



The limitations of the Superscalar processor are prominent as the difficulty of scheduling instruction becomes complex. The intrinsic parallelism in the instruction stream, complexity, cost, and the branch instruction issue get resolved by a higher instruction set architecture called the **Very Long Instruction Word (VLIW)** or **VLIW Machines**. VLIW uses Instruction Level Parallelism, i.e. it has programs to control the parallel execution of the instructions.

In other architectures, the performance of the processor is improved by using either of the following methods: pipelining (break the instruction into subparts), superscalar processor (independently execute the instructions in different parts of the processor), out-of-order-execution (execute orders differently to the program) but each of these methods add to the complexity of the hardware very much. VLIW Architecture deals with it by depending on the compiler. The programs decide the parallel flow of the instructions and to resolve conflicts. This increases compiler complexity but decreases hardware complexity by a lot.

Features:

- The processors in this architecture have multiple functional units, fetch from the Instruction cache that have the Very Long Instruction Word.
- Multiple independent operations are grouped together in a single VLIW Instruction. They are initialized in the same clock cycle.
- Each operation is assigned an independent functional unit.
- All the functional units share a common register file.

- Instruction words are typically of the length 64-1024 bits depending on the number of execution unit and the code length required to control each unit.
- Instruction scheduling and parallel dispatch of the word is done statically by the compiler.
- The compiler checks for dependencies before scheduling parallel execution of the instructions.

Parul® University
Vadodara, Gujarat
NAAC A++
RA

https://paruluniversity.ac.in/