MULTITHREADING

Thread

A thread in an operating system (OS) is a ***lightweight process that shares the same address space as the parent process but has its own stack and register state.***

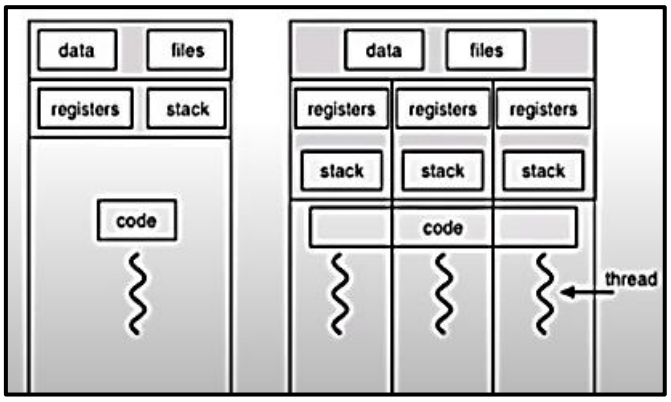
A process can have multiple threads all those multiple threads in the process help the process to preform multiple executions which improves the working,

There ***must be at least one thread in a process.***

Three states of threads : ***Running, Ready, Blocked***

Benefits

- It ***shares common data*** and ***inter process communication.***
- ***Speed improvement*** when multiple threads cooperate to complete a single job.
- ***Context switching becomes fast*** when working with threads.



Types of threads

1. **User level threads**
2. **Kernel level threads**

1. User-level threads:

The operating system does not directly support user level threads. Instead, ***threads are managed by a user-level thread library,*** which is part of the application.

The ***library manages the threads and schedules them on available processors.*** The advantages of user-level threads include ***greater flexibility and portability,*** as the application has more control over thread management.

However, the ***disadvantage is that user-level threads are not as efficient as kernel-level threads,*** as they ***rely on the application to manage thread scheduling.***

2. Kernel-level threads:

The operating system ***directly supports threads*** as part of the kernel. Each thread is a separate entity that can be ***scheduled and executed independently by the operating system.***

The advantages of kernel-level threads ***include better performance and scalability,*** as the operating system can schedule threads more efficiently.

MULTITHREADING

Multithreading is the ***ability of a program to manage multiple threads of execution simultaneously.***

Multithreading can be used to improve the performance of programs in a number of ways.

For example, take advantage of multiple processors to speed up computationally intensive tasks.

Here are some examples of how multithreading can be used:

A ***web browser*** can use multiple threads ***to load different parts of a web page*** simultaneously. This can make the web page load faster for the user.

challenges of multithreading:

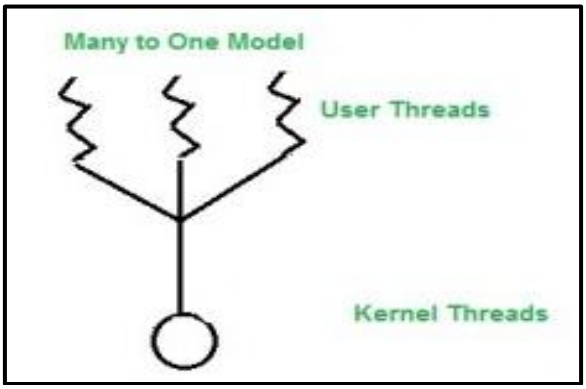
1. **Race conditions:** A race condition occurs when two or more threads are trying to access same resource at the same time.
2. **Deadlocks:** A deadlock occurs when two or more threads are waiting for each other to finish executing before they can continue.
3. **Increased complexity:** Multithreaded programs are generally more complex than single-threaded programs.

Multithreading Models

Many operating systems support kernel thread and user thread in a combined way. Example of such system is ***Solaris.***

1. Many-to-One Model (User-Level Threads):

- **Overview**: In the Many-to-One model, ***multiple user-level threads are managed by a single kernel-level (or system-level) thread.*** This means that the ***operating system sees only one thread per process, even if there are multiple threads within that process.***

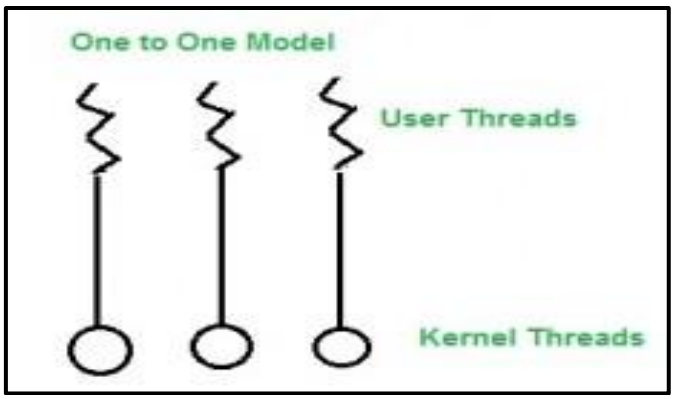


- **Advantages**:
  - **Lightweight**: User-level threads are ***lightweight because they are managed entirely by the application*** without kernel involvement.
  - **Portability:** This model can be implemented on systems that do not support multithreading natively.

- **Disadvantages**:
  - **Limited Concurrency:** Since all threads share a single kernel thread, if one user-level thread blocks (e.g., due to I/O), it blocks the entire process.
  - **Lack of Parallelism:** True parallelism is not achieved because only one kernel-level thread is handling all user-level threads.

2. One-to-One Model (Kernel-Level Threads):

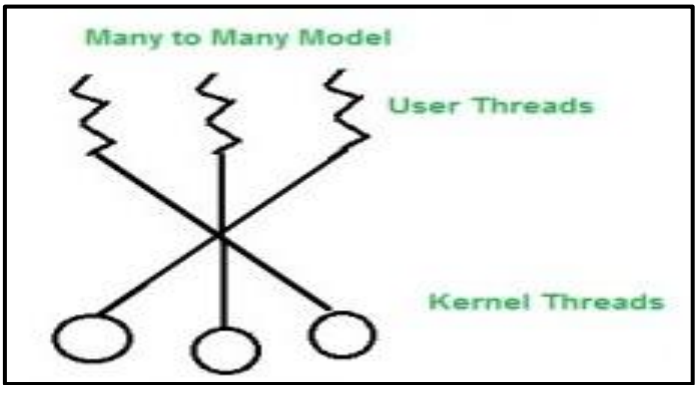
- **Overview**: In the One-to-One model, each user-level thread corresponds directly to a kernel-level thread managed by the operating system. ***This means that multiple threads in a process are truly concurrent and can run in parallel on multi-core processors.***



- **Advantages**:
  - **Concurrency**: ***Multiple threads can execute concurrently,*** making full use of multi-core processors.
  - **Fault Tolerant**: If one thread blocks, other threads in the same process can continue to run.
- **Disadvantages**:
  - **Overhead:** ***Creating and managing kernel-level threads can be more resource-intensive than user-level threads.***
  - **Limited Scalability:** The number of ***kernel threads may be limited*** by the operating system.

3. Many-to-Many Model (Hybrid or Two-Level Model):

- **Overview**: The Many-to-Many model combines elements of both the Many-to-One and One-to-One models. It allows ***multiple user-level threads to be mapped to an equal or smaller number of kernel-level threads.*** This model aims to provide concurrency while ***maintaining control over the number of kernel threads.***



- **Advantages**:
  - Many-to-Many allows for concurrency while keeping the number of kernel threads under control.
  - The system can ***dynamically map user threads to available kernel threads based on workload.***
- **Disadvantages**:
  - Mapping between user-level and kernel-level threads can be complex.
  - If not managed carefully, it can lead to ***inefficient resource allocation.***

DIFFERENCE TABLE OF MULTITASKING AND MULTITHREADING

Aspect	Multitasking	Multithreading
Definition	Running multiple processes or tasks concurrently	Running multiple threads (smaller units) within a single process concurrently
Resource Usage	Requires ***separate processes*** with their ***own memory and resources***	Uses a single process's memory and resources for multiple threads
Communication	Processes often communicate via ***inter-process communication (IPC) mechanisms***	Threads within the same process can ***communicate easily by sharing memory***
Overhead	Typically, has ***higher overhead due to separate processes,*** leading to potential ***slower context switching***	Lower overhead because threads share the same address space and resources
Isolation	Provides ***strong isolation between processes;*** a failure in one process doesn't affect others	Threads are closely connected and can potentially interfere with each other, if not managed properly
Synchronization	Requires ***more complex synchronization mechanisms*** (e.g., ***semaphores, message queues***) to coordinate between processes	Easier synchronization ***using built-in thread synchronization mechanisms*** (***e.g., locks, mutexes***)
Scalability	Less scalable due to higher overhead when creating new processes	***More scalable*** as creating new threads within a process is typically ***faster and more efficient***
Fault Tolerance	Better fault tolerance as a failure in one process usually doesn't impact others	More susceptible to failures in one thread affecting the entire process
Examples	Running multiple applications concurrently on a computer (e.g., word processing, web browsing)	***A web server*** handling multiple client connections concurrently or parallel processing in applications

LATENCY HIDING TECHNIQUES

Latency refers to the ***delay or the amount of time*** taken for something to happen. In computing, it often refers to the ***delay in accessing or retrieving data from memory or storage,*** which can impact how fast a computer performs tasks. ***Lower latency is better*** because it means things happen more quickly.

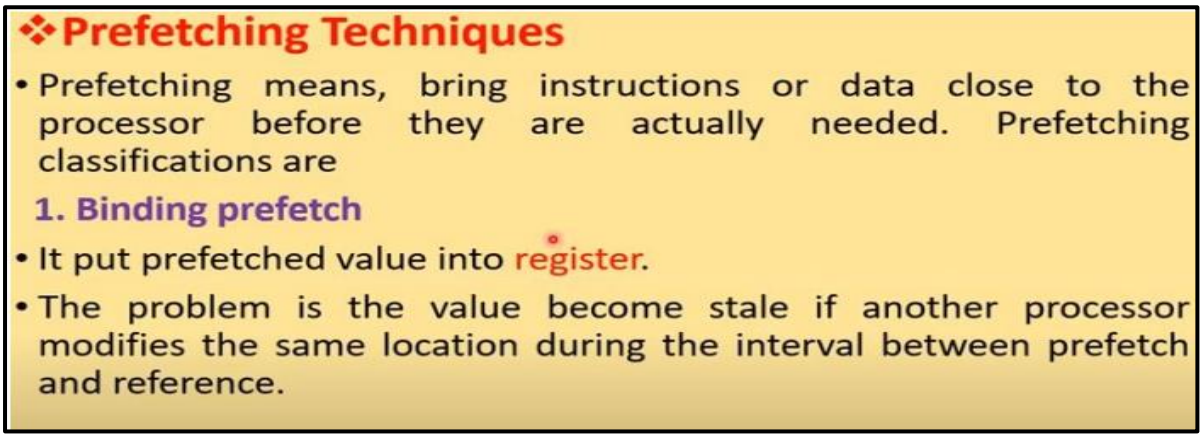
Latency hiding is a technique for improving the performance of computer systems by ***reducing the impact of latency***

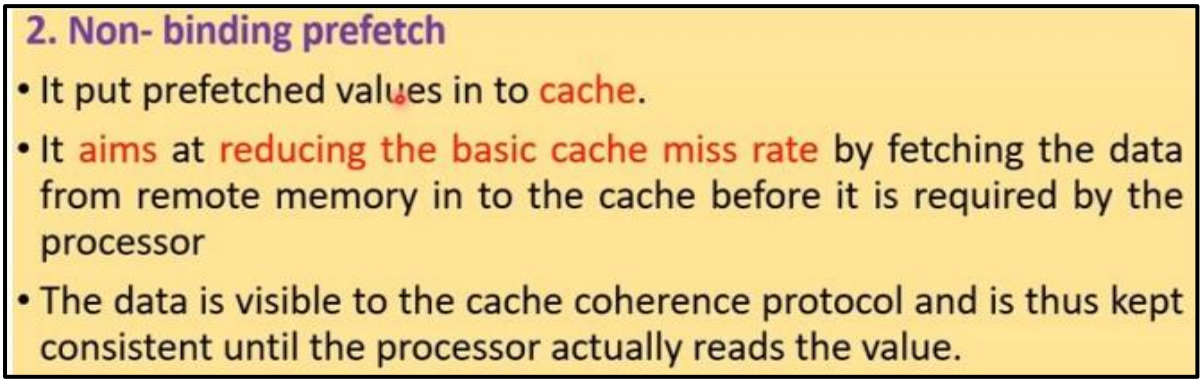
Latency hiding can be accomplished by four approaches:

1. Prefetching Techniques:

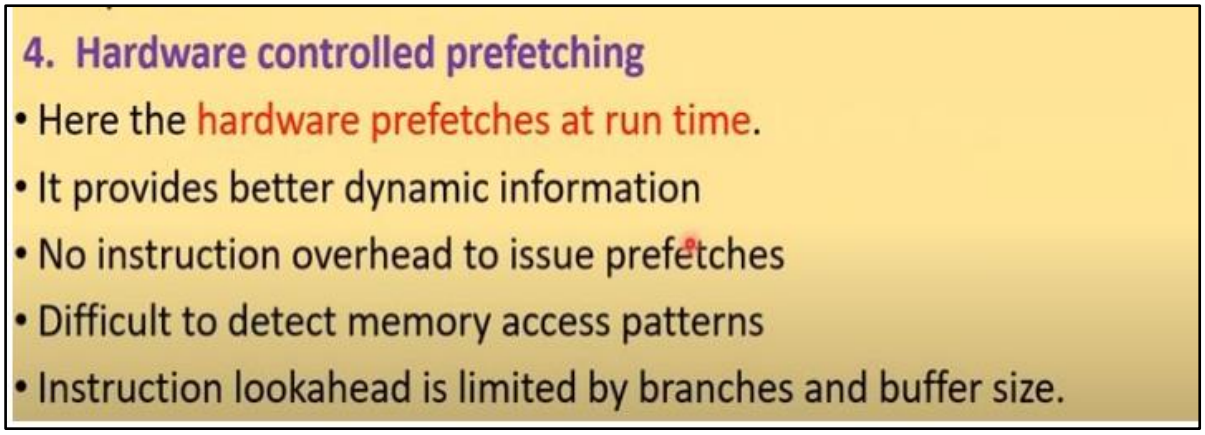
- **What It Is**: Imagine you're reading a book, and you have a friend who predicts what page you'll read next. Your friend fetches that page and keeps it ready for you. This way, you don't have to wait when you turn the page.
- **In Computers**: Prefetching is like having a ***smart computer that guesses what data you'll need from memory and brings it in advance,*** so it's ready when your program asks for it. This saves time waiting for data to arrive.

Prefetching technique involve its own prefetch techniques as follows.



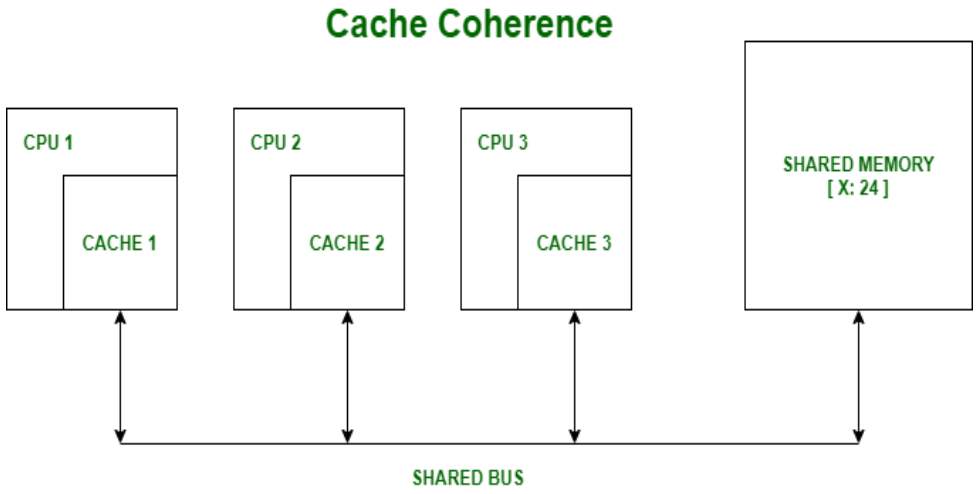


3. Software-controlled prefetching is often done by inserting specific prefetching instructions directly into a program's code. These prefetch instructions tell the computer to fetch data from memory in advance, based on predictions made by the programmer or the compiler.



2. Coherent Caches:

- **What It Is**: Think of a cache like a small storage area near your desk where you keep your most-used books. When you need a book, you check the cache first because it's faster than going to the library.
- **In Computers**: Coherent caches are like having multiple small storage areas (caches) that store copies of important data. These ***caches are synchronized so that when one gets updated, the others also get the same update. T***his helps keep data consistent and speeds up access.



3. Relaxed Memory Consistency Models:

- Relaxed memory consistency models allow computers to be more flexible in ***how they access and update memory.*** This flexibility can ***improve performance*** but requires careful programming to ensure correctness.

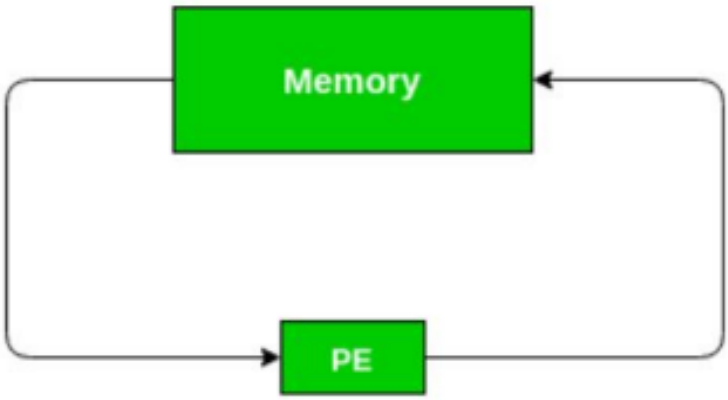
4. Multiple Contexts:

- Multiple contexts mean the ***computer can work on several tasks simultaneously.*** It's like a multitasking computer that can switch between different jobs, making the ***best use of its time and resources.***

SYSTOLIC ARRAY IN MULTI-THREADING

Systolic Array:

- A systolic array is a ***homogeneous network*** of data processing units (DPUs) called ***cells or nodes.***
- Each computing unit (node or DPU) performs its own calculation based on the ***data it receives from the units that come before it.*** After completing its computation, the unit stores the result and then passes it along to the next unit in the sequence.
- Systolic arrays consist of a large number of identical simple processors or processing elements (PEs) arranged in a well-organized structure, such as a ***linear or two-dimensional array.***
- Each processing element is connected to other PEs and has ***limited private storage.***



Characteristics:

1. **Parallel Computing:** Systolic arrays enable parallel computing by performing many processes simultaneously. The ***non-centralized structure of the arrays*** allows for parallel execution.
2. **Pipelinability:** Systolic arrays ***exhibit high-speed pipelining capabilities,*** achieving a ***linear rate of pipelinability.***
3. **Synchronous Evaluation:** Computation in a systolic array is timed by a ***global clock,*** and data is passed through the network accordingly. The ***global clock synchronizes the array,*** and ***fixed-length clock cycles*** are used.
4. **Advantages of Systolic Arrays:**
    - ***High degree of parallelism,*** resulting in ***high throughput.***
    - ***Compact, robust,*** and ***efficient.***
    - Simple and ***regular data control flow.***
    - All operand data and partial results are stored within the processor array, ***eliminating the need for external buses, main memory, or internal caches*** during each operation.

Applications:

- Data sorting tasks
- Dynamic programming algorithms
- Artificial intelligence
- Image processing
- Pattern recognition