COMPUTER ORGANIZATION AND ARCHITECTURE

COA LAB 2023

ASSIGNMENT -2

Name: GUNDA ANISH

Roll No: 21CS01002

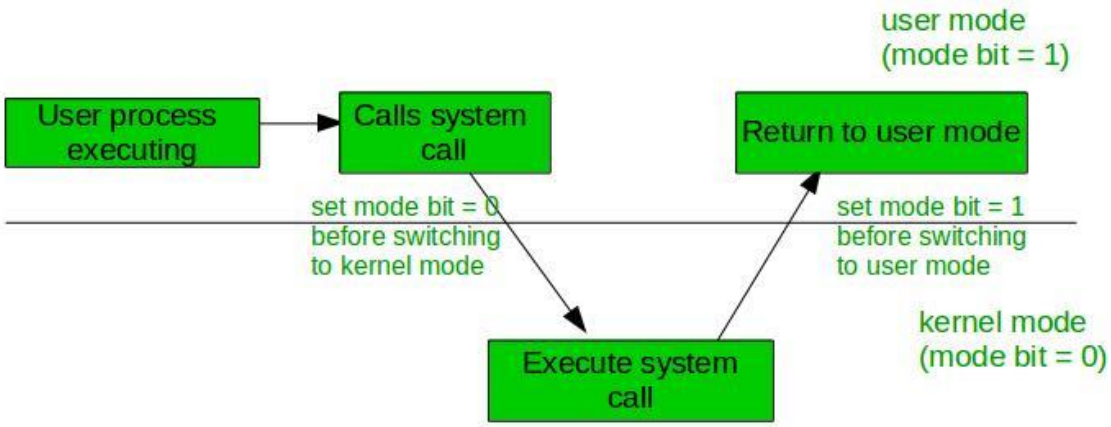
Report on Key topics:

- Kernel
- Thread
- Process
- SIMD
- GPU Memory Hierarchy – SRAM/DRAM, Shared Memory, Constant Memory
- Scheduler
- Warp
- Thread Block

Kernel:

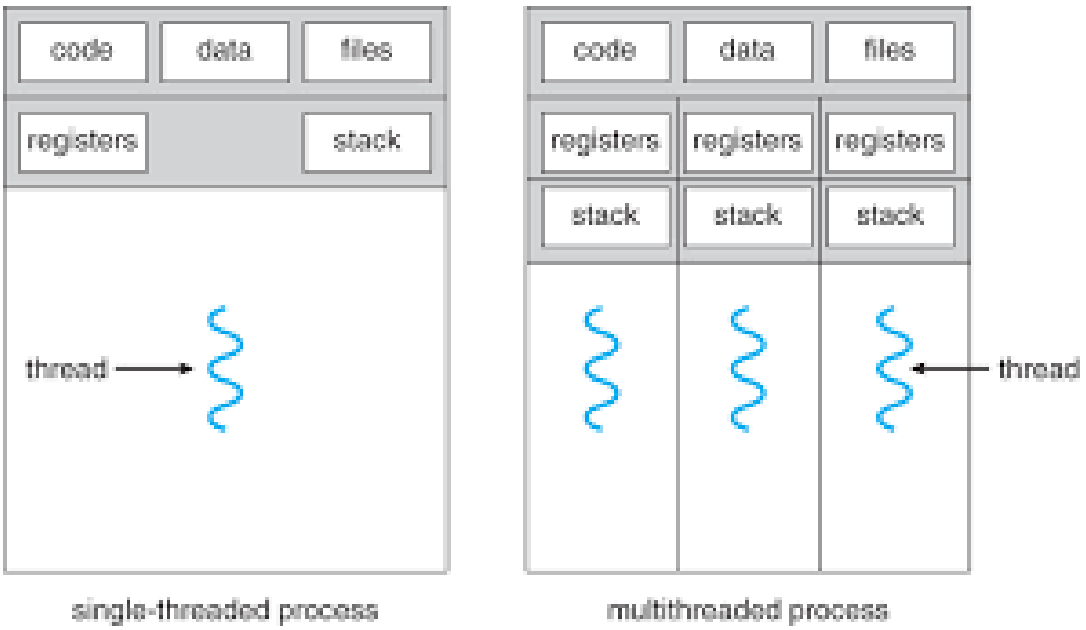
- It's a core component of operating system that manages operations of computer and hardware.
- It acts as a bridge between applications and data processing performed at hardware level using *system calls*.
- Kernel load first into memory when OS is loaded and remains in memory until operating system shuts down again.

- **Tasks of a kernel:**
  - It has a process table which keeps track of all active processes.
  - Its objective is to decide state of incoming processes, control disk, memory and task management.

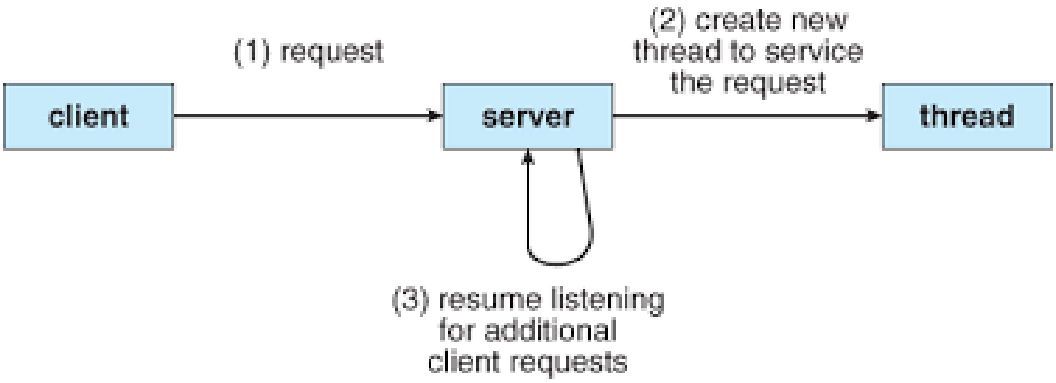


Thread:

- A thread refers to a single sequential flow of activities being executed in a process.
- It also refers to an execution unit in the process that has its own stack, register set and a program counter.
- They aren't allowed to exist outside a process.
- It shares with other threads belonging to the same process its code section, data section, and other OS resources, such as open files.
- A traditional (heavyweight) process has a single thread of control.
- If a process has multiple threads of control, it can perform more than one task at a time.



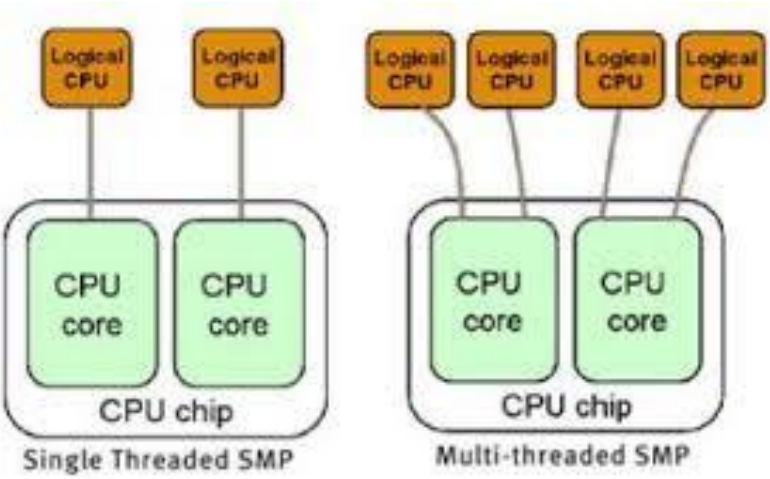
- **Why do we need multi-threaded processes ?**
  - **Example - 1:**
    - Consider a case of word document:
    - If we want to enter keystrokes, have a spell checker running at same time → *It would not be possible with single thread.*
    - It is because at any point of time single thread is being executed and it can't perform spell check unless keystrokes are entered or vice-versa.
    -
  - **Example - 2**



- If server runs as a single-threaded process that accepts requests, then each time the server receives a request then it should start creating a new process.
- In case of multi-threaded process instead of creating new process, server will create a new thread and resumes listening to requests.

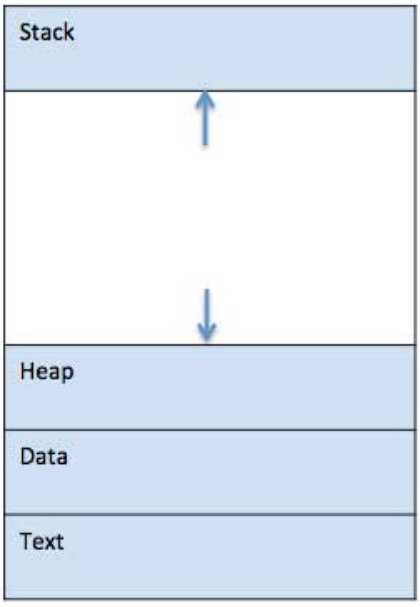
➢ **Benefits of Multi-threaded processes :**

- Responsiveness:
  - It allows the program to continue running even if a part of it is blocked or performing a lengthy operation.
- Resource Sharing:
  - Processes may only share resources through techniques such as shared memory or message passing.
  - Whereas in case of multiple-threads, the code and data are at same address space for all threads belonging to a process.
- Economy:
  - As threads share the resources of the process, context-switching in case of threads is more economical in time as we get rid of overhead during switching processes.
- Scalability:
  - A single-thread process can run only on one processor irrespective of how many are available.
  - In case of multi-threaded processes, multi-cores increase parallelism.

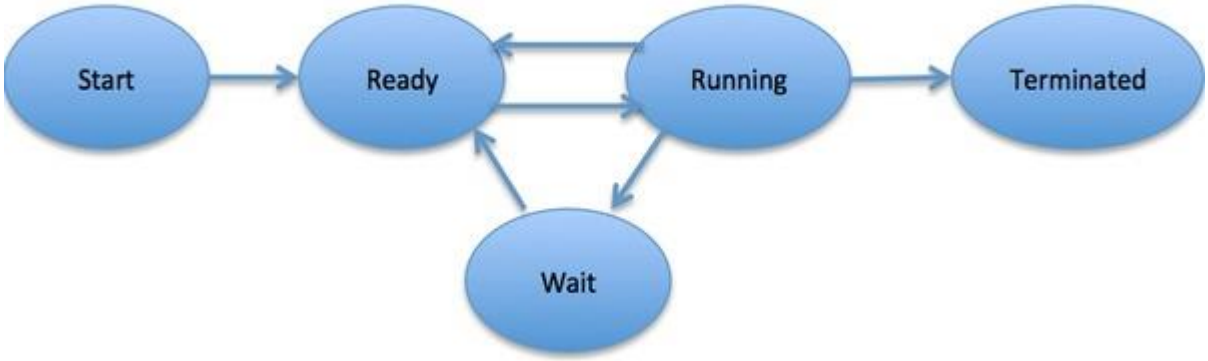


Process:

- A program is a set of instructions that perform a specific task when executed by a computer.
- A process is nothing but a program in execution.
- A program is a passive entity when not in execution and is stored in the secondary storage.
- When the program is loaded into main memory and starts executing then it becomes an active entity and its known as a process.



- **Life cycle of a process:**

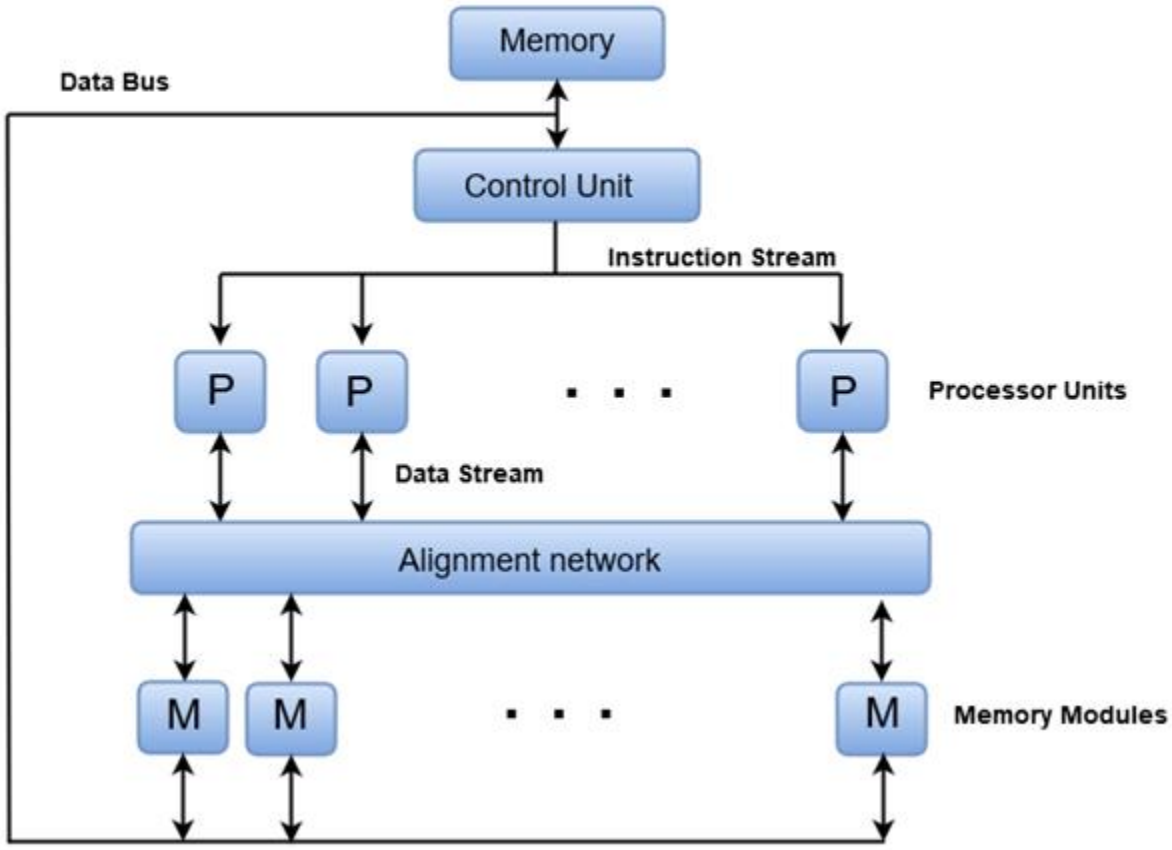


  - Start:
    - This is the initial state when the process is started/created.
  - Ready:
    - This is the state when the process is waiting to be assigned to a processor.
    - A process can come to this state from *Start* state or *Running* state when the processor receives an interrupt to execute another processor or from *wait* state.
  - Running:
    - Once the process has been assigned to the processor by OS scheduler, its state turn into running and the processor executes its instructions.
  - Waiting:
    - Process moves into waiting state if it's waiting for a resource such as I/O resource or waiting for a file to open.
  - Terminated:
    - Once the process finishes execution or gets terminated by the OS, it moves to the terminated state and waits to be removed from the main memory.

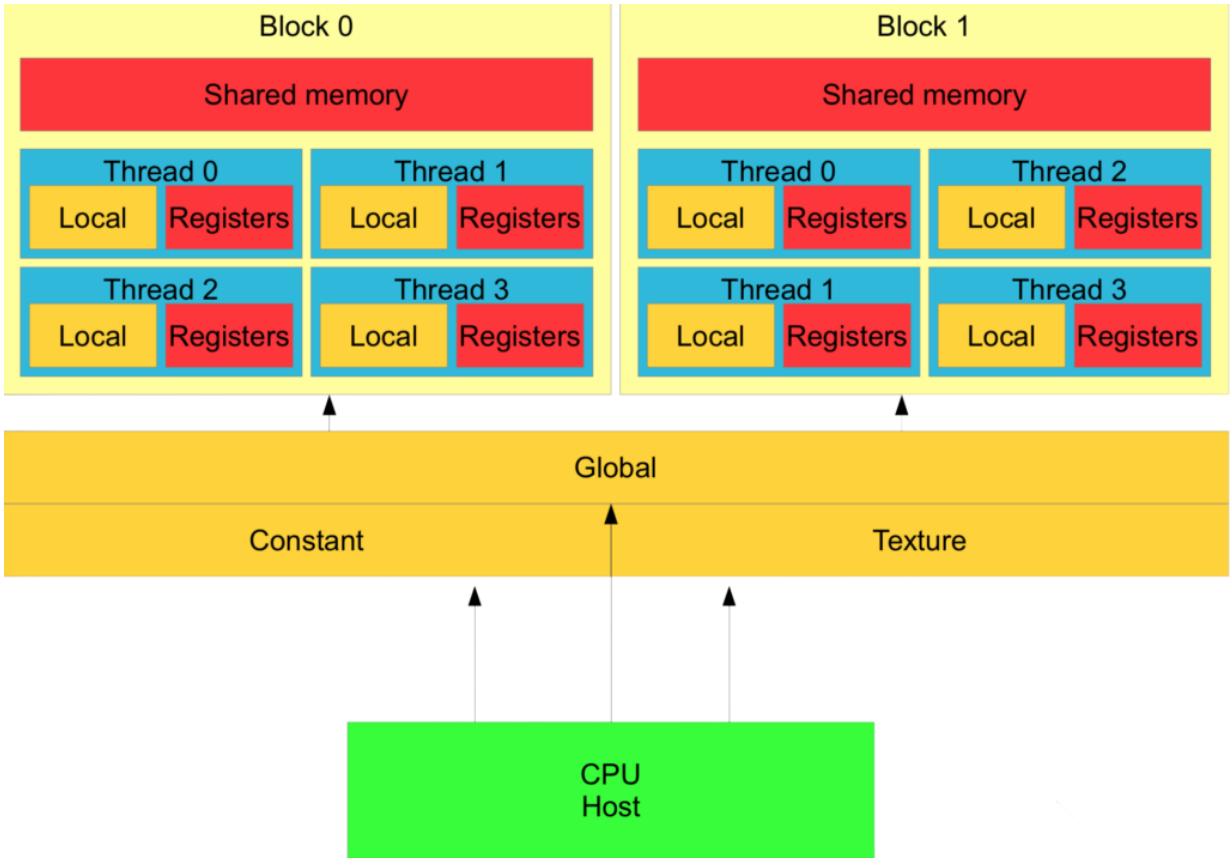
SIMD:

- SIMD stands for *single-instruction-multiple-data* stream.
- It represents an organization that includes many processing units under the supervision of a common control unit.
- All processors receive the same instruction but operate on different data.

SIMD:



GPU ItMemory Hierarchy:



SRAM vs DRAM:

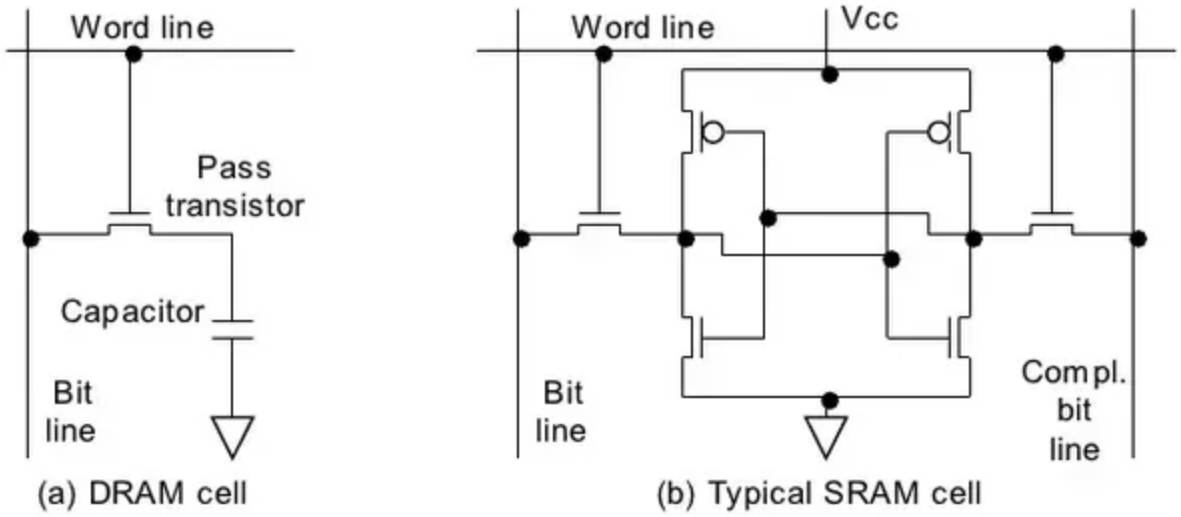
SRAM:

- SRAM stands for *Static Random-Access Memory.*
- It is a volatile memory, which mean that the data is lost when the power is off.
- It stores each bit using latching circuitry.
- It is quicker and more costly than DRAM.
- It must be updated on a regular basis.
- It is often used for a CPU's cache and internal registers.
- SRAM is named static because it does not require any changes or actions, such as refreshing, to keep data intact.

DRAM:

- DRAM stands for *Dynamics Random-Access Memory.*

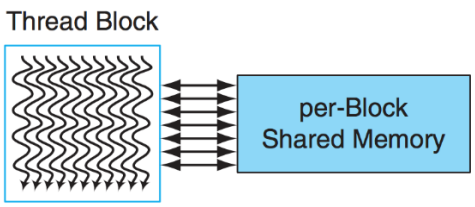
- It is a form of RAM where each bit of data is stored in its own capacitor within an integrated circuit.
- DRAM is dynamic because it requires regular modification or activity, such as refreshing, to keep the data intact.
- It is employed in the implementation of main memory.
- Capacitors and a few transistors are used to make DRAM where bit 1 is represented by fully charged capacitor and a bit 0 by discharged capacitor.



Shared Memory vs. Constant Memory:

Shared Memory:

- Shared memory is very fast.
- It's used to enable faster communication between threads.
- It only exists for the lifetime of the block.
- It's fastest when all threads read data from different bank.
- Bank conflicts occur only within a warp but not between different warps.

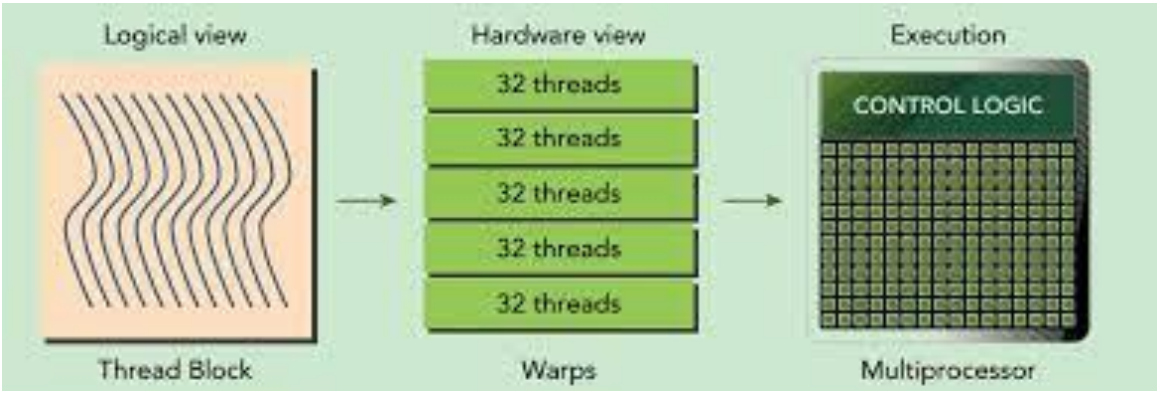


Constant Memory:

- The constant memory can be written into and read by the host.
- It is used for storing data that will not change over the course of kernel execution.
- It supports low latency, read-only access by the device when all threads simultaneously read data from the same location.
- All threads have access to constant memory but they can only read data from it but not write into it.

Warp:

- A warp is a set of 32 threads within a thread block such that all threads in a warp execute the same instruction.
- The threads of a thread block execute concurrently on one SM(stream multiprocessor), and multiple thread blocks can execute concurrently on one SM.
- As thread blocks terminate, new thread blocks are launched on the vacated SMs.



Scheduler:

- Process scheduling is the activity of the process manager that handles the removal of the running process from the CPU and the selection of another process on the basis of a particular strategy.

- Schedulers are special system software that handles process scheduling in various ways.
- Their task is to select the jobs to be submitted to the system and to decide which process to run.
- Types of Schedulers:
  - Long-term scheduler
  - Short-term scheduler
  - Medium-term scheduler
- **Long-term Scheduler:**
  - It's also called as a job scheduler.
  - It determines which programs are admitted to the system for processing.
  - It selects processes from the queue and loads them into the memory for execution.
  - Process loads into memory for execution.
  - Time-sharing OS has no long-term schedulers.
- **Short-term Scheduler:**
  - It's also called a CPU scheduler.
  - CPU scheduler selects processes from the ready queue and allocates CPU to one of them.
  - These make decisions about which processes to execute next.
- **Medium-term Scheduler:**
  - This type of scheduling is a part of swapping.
  - It removes the processes from the memory.
  - This is in-charge of handling the swapped-out processes.
  - A suspended process cannot make any progress towards completion and in such cases the process is moved from main memory to secondary memory, this is know as swapping out.

Thread Block:

- A thread block is a programming abstraction representing a group of threads that can be executed serially or in parallel.
- For better process and data mapping, threads are grouped into thread blocks.
- The number of threads varies with available shared memory.
- The threads in same thread block run on same stream processor.
- Threads in the same block can communicate via shared memory.
- Multiple blocks are combined to form grids.
- All blocks in the same grid contain same number of threads.