**CPU SCHEDULING**

1. **BASIC CONCETS**

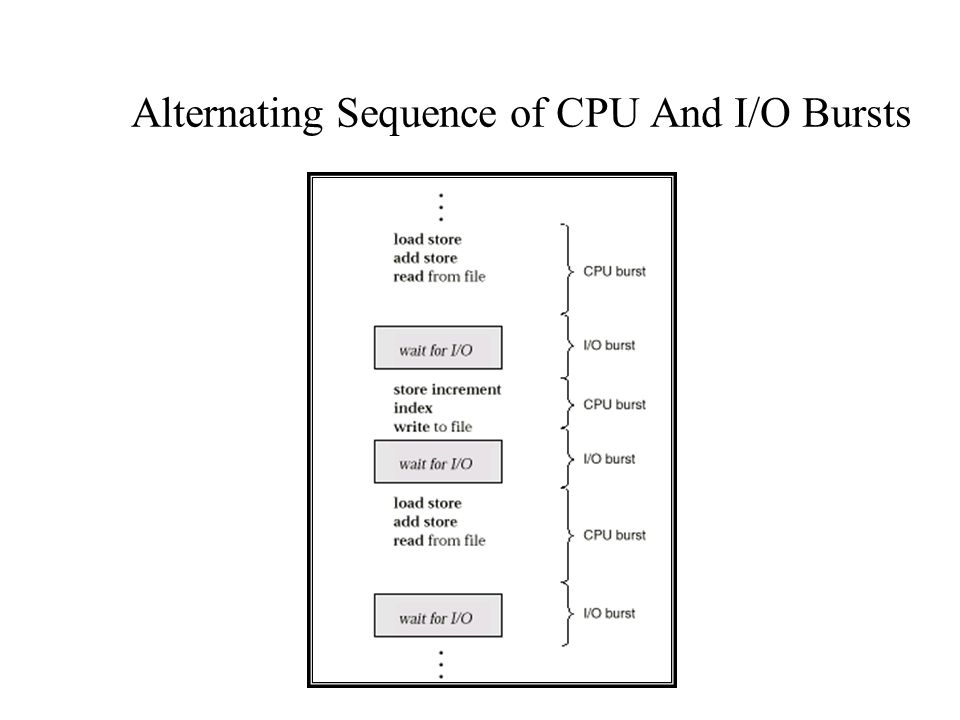
The objective of multiprogramming is to have some process running at all times to maximize CPU utilization.

Scheduling: deciding which process/ thread should occupy a resource (CPU, disk, etc).

* CPU- I/O BURST CYCLE

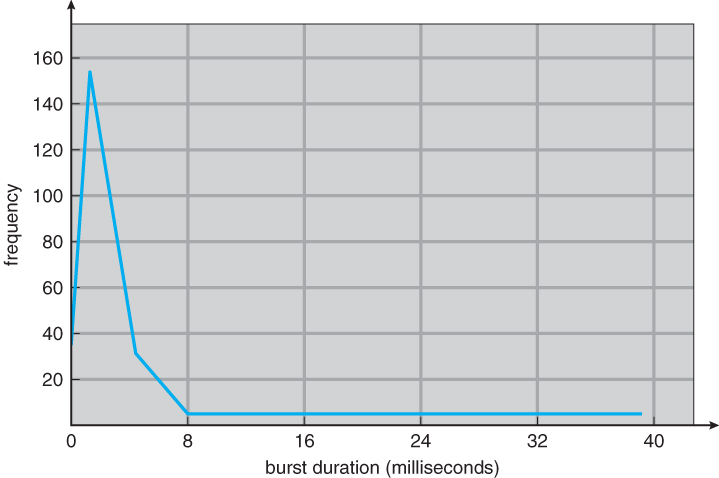
The success of CPU scheduling depends on an observed property of processes.

Process execution consists of a cycle of CPU execution and I/O wait. Processes alternate between these two states. Process execution begins with a CPU burst. This is followed by an I/O burst, which is followed by another CPU burst, then another I/O burst and so on. Eventually, the final CPU burst ends with a system request to terminate execution.



The curve is generally characterised as exponential or hyper exponential, with a large number of short CPU bursts and a small number of long CPU bursts. A I/O bound program typically has short CPU bursts while a CPU bound program might have a few long CPU bursts.

**HISTOGRAM OF CPU BURST DURATION**



* CPU SCHEDULER

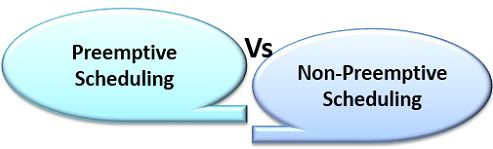
Whenever CPU become idle, O.S. must select one of the processes in the ready queue to be executed. The selection process is carried out by short term scheduler (CPU Scheduler). The scheduler selects a process from processes in memory that are ready to execute (from ready queue) and allocate CPU to that process. The ready queue is not necessarily a FIFO (first in first out) queue.

CPU scheduling decisions may take place under following circumstances.

1. When a process switches from running state to waiting state.
2. When a process switches from running state to ready state.
3. When a process switches from waiting state to ready state.
4. When a process terminates.

In 1 and 4 circumstance: no choice for scheduling (non-preemptive or cooperative).

In 2 and 3 circumstance: choice for scheduling (preemptive).



* PREEMPTIVE

In preemptive scheduling the currently running process may be interrupted and moved to the ready state by OS (forcefully).

* NON-PREEMPTIVE

In non-preemptive scheduling, the running process can only lose the processor voluntarily by terminating or by requesting an I/O. once CPU given to a process it cannot be pre-empted until the process completes its CPU burst.

**Question:** Why preemptive scheduling incur a cost associated with excess to shared data.

**Answer:** Consider the case of two processes that share data. While one is updating data, it is pre-empted so that second process can run. The second process then tries to read the data which is in an inconsistent state. In such situation we need new mechanism to co-ordinate access to shared data.

* DISPATCHER

It is an important component involved in CPU scheduling. The OS code that takes the CPU away from the current process and hands it over to the newly scheduled process is known as the dispatcher.

It is a module that gives control of CPU to process selected by Short Term Scheduler. This function involves:

1. Switch Context
2. Switching to user mode
3. Jumping to proper location in user program to restart that program.

**Dispatch Latency:** The time it takes for dispatcher to stop one process and start another running is known as dispatch latency.

1. **SCHEDULING CRETERIA/ OPTIMIZATION CRITERIA**

* CPU UTILIZATION

We want to keep CPU as busy as possible. CPU utilization can range from 0 to 100 percent. In real time system, it should range from 40% (light system) to 90% (heavy system).

* THROUGHPUT

Number of processes that are completed per unit time is called throughput. For long process this rate may be one process per hour, for short transaction it may be ten processes per second.

* TURNAROUND TIME

How long it takes to execute that process. The interval from time of submission of a process to time of completion is turnaround time. It is the sum of the periods spent waiting to get into memory, waiting in the ready queue, executing on the CPU and doing I/O.

Turnaround time is generally limited by the speed of output device. An accurate illustration should involve many processes, each being a sequence of several hundred CPU bursts and I/O burst (ideal case) for simplicity we consider only one CPU burst (in millisecond) per second.

* WAITING TIME

It affects the amount of time that a process spends waiting in ready queue. Waiting time is sum of periods spent waiting in ready queue.

* RESPONSE TIME

The measure called response time is time process takes to start responding and not time it takes to output the response.