1. source code -> preprocessor -> assembler -> linker -> executable object file
2. processor : in cpu, execute instructions
3. register > L1 cache > L2 cache > Main Memory(DRAM) > secondary storage.
4. process and context switch
5. operating system : sits between hardware and application software
    a. Resource allocation
    b. Protection
    c. Reclamation
    d. Virtualization

   Types of Operating System:
    a. Monolithic
    b. Layered
    c. Microkernel
6. Bootstrap Program : stored in ROM or EPROM, loded during power on
9. Process and Thread: A process is an execution of a program but a thread is a single execution sequence within the process. A process can contain multiple threads. A thread is sometimes called a **lightweight process**. So like multiple process can run In parallel similarly multiple threads can run in parallel and reduce the total execution time of the process.

Difference between Process and Thread:

S.NO	PROCESS	THREAD
1.	Process means any program is in execution.	Thread means segment of a process.
2.	Process takes more time to terminate.	Thread takes less time to terminate.
3.	It takes more time for creation.	It takes less time for creation.

4.	It also takes more time for context switching.	It takes less time for context switching.
5.	Process is less efficient in term of communication.	Thread is more efficient in term of communication.
6.	Process consume more resources.	Thread consume less resources.
7.	Process is isolated.	Threads share memory.
8.	Process is called heavy weight process.	Thread is called light weight process.
9.	Process switching uses interface in operating system.	Thread switching does not require to call a operating system and cause an interrupt to the kernel.

10.	If one server process is blocked no other server process can execute until the first process unblocked.	Second thread in the same task could run, while one server thread is blocked.
11.	Process has its own Process Control Block, Stack and Address Space.	Thread has Parents' PCB, its won Thread Control Block and Stack and common Address space.

10.CPU scheduling:

a. Preemptive scheduling
b. Nonpreemptive scheduling

A. FCFS (First come first serve) Scheduling:

- Preemptive FCFS
- Non Preemptive FCFS

B. Shortest Job First

- Preemptive SJF
- Non Preemptive SJF
- Approximate SJF

C. Priority Scheduling

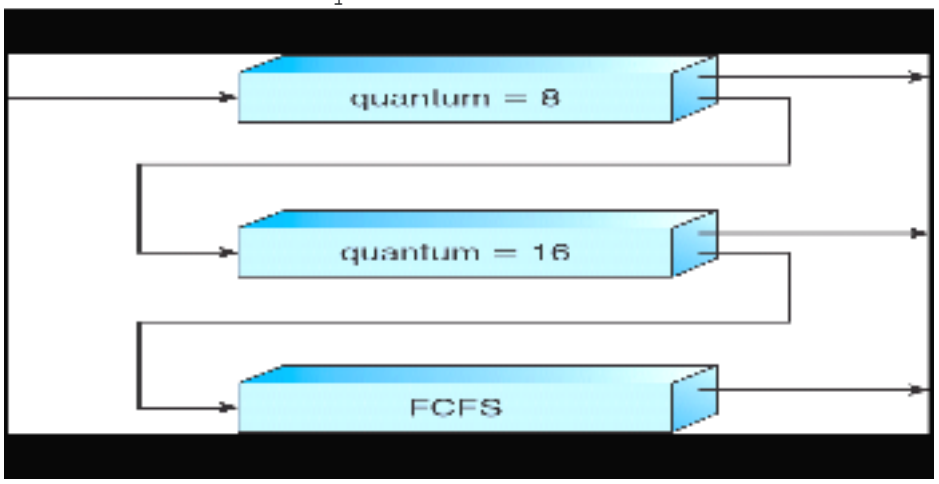
- Preemptive PS
- Non Preemptive PS

Problem : Starvation : low priority processes may never execute
Solution : Aging : as time progresses, increase the priority of the process

D. Round Robin (RR):Each process gets a small unit of CPU time (time quantum), usually 10-100 milliseconds. After this time has elapsed, the process is preempted and added to the end of the ready queue.

- Time slice too large
  - FIFO behavior
  - Poor response time
- Time slice too small
  - Too many context switches (overheads)
  - Inefficient CPU utilization

11. Multilevel Feedback Queue :
- Favors I/O bound processes to achieve good device utilization
- Separates processes into categories based on their CPU eed
- Each time a process leaves the queue it is stamped with the identity of the lowest level queue in which it last resided.
- When the process reenters the fray for CPU, it is sent directly to the queue where it last completed its execution.
- (the past behavior is a good indicator of the future behavior)
- If the process changes its characteristics from CPU bound to I/O bound
    1) the process is stamped with the time spent last time on the CPU.
    2) if a process voluntarily relinquishes the CPU, it may be moved up to the next level queue.



12. Lottery Scheduling: Flexible Proportional-Share Resource Management
- Lottery scheduling is a randomized resource allocation mechanism.
- Resource rights are represented by lottery tickets.
- Each allocation is determined by holding a lottery; the resource is granted to the client with the winning ticket.
- This effectively allocates resources to competing clients in proportion to the number of tickets that they hold.

- Scheduling by lottery is probabilistically fair. The expected allocation of resources to clients is proportional to the number of tickets that they hold.
- Since the scheduling algorithm is randomized, the actual allocated proportions are not guaranteed to match the expected proportions exactly.
- However, the disparity between them decreases as the number of allocations increases.

14. Process Synchronization:
- Requirement
- Solution : Mutex lock
                : critical section problem
                : Spin Around condition
                : Wait on condition, Signal, Broadcast
                : Bounded Buffer Problem : Github

: Readers and Writers Problem : Georgia Tech
: Dining-Philosophers Problem

- Problem in Mutex Lock
  : Deadlock
    a) Mutual Exclusion
    b) Hold and Wait
    c) No Preemption
    d) Circular wait
  : Deadlock Detection:
    a) Resource-Allocation Graph
    b) Wait on graph

  : How to avoid it
    a) linearisation of resources before lock
    b) The Ostrich Algorithm