Process Management Basics

Foundation Course on Operating Systems for GATE 2022

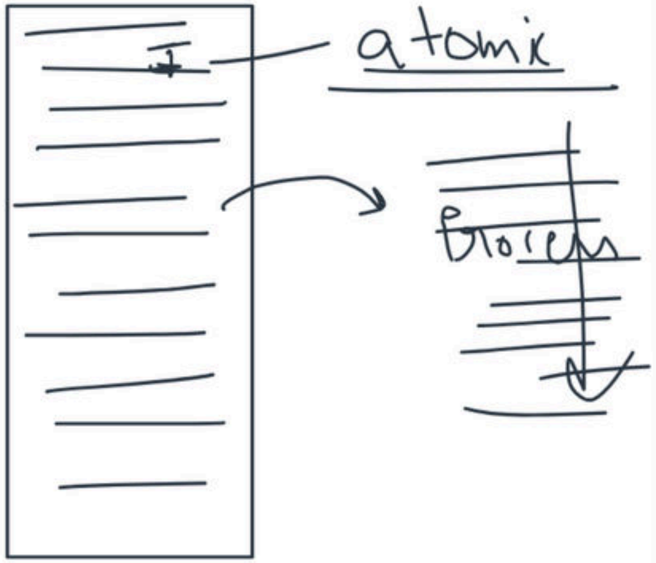
Sanchit Jain • Lesson 3 • July 23, 2021

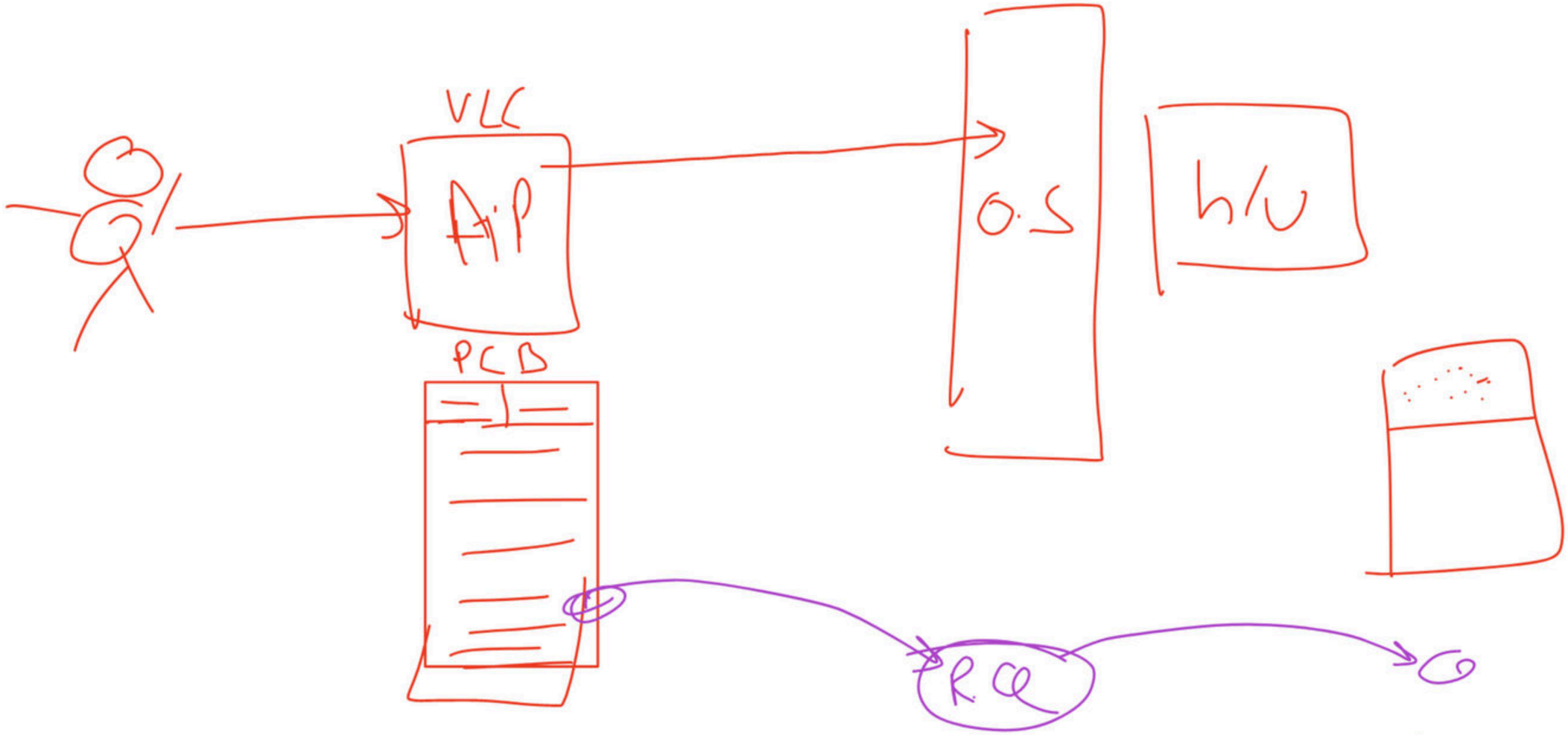
Process

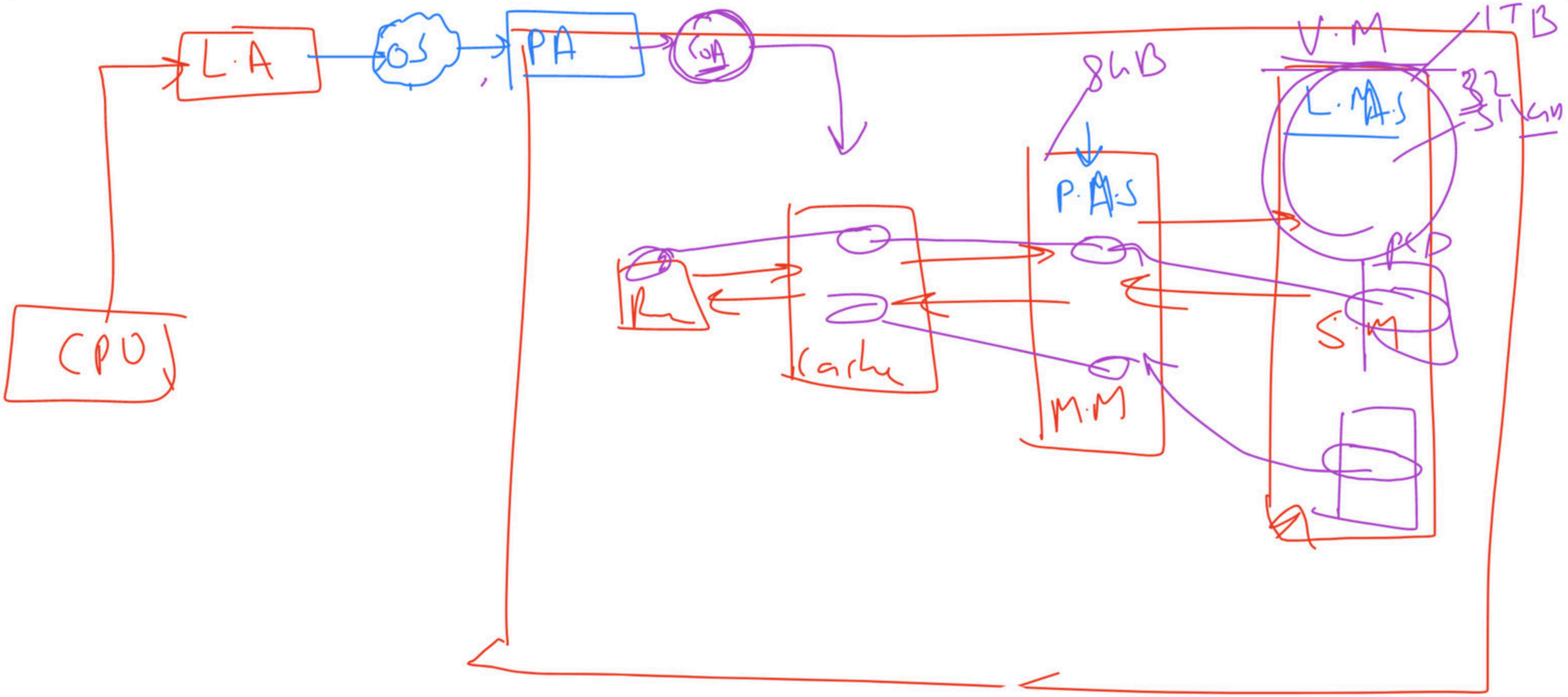
CPU Scheduling (1-2)
Process synchronization (1-2)
D.L (1-2)

3-4

- In general, a process is a program in execution.
- A Program is not a Process by default. A program is a passive entity, i.e. a file containing a list of instructions stored on disk (secondary memory) (often called an executable file).
- A program becomes a Process when an executable file is loaded into main memory and when it's PCB is created.
- A process on the other hand is an Active Entity, which require resources like main memory, CPU time, registers, system buses etc.

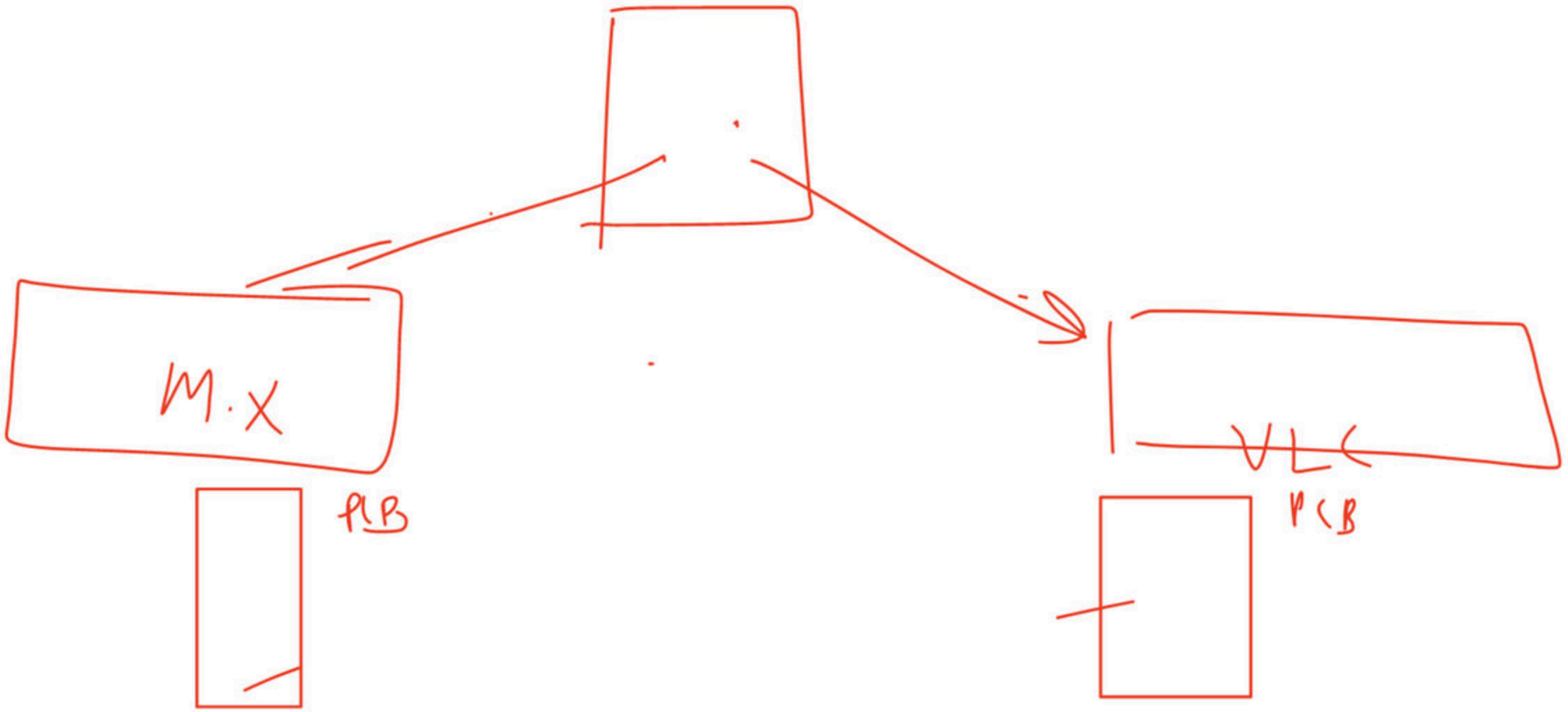



VLC
A.P
O.S
h/w
PCB
R.Q

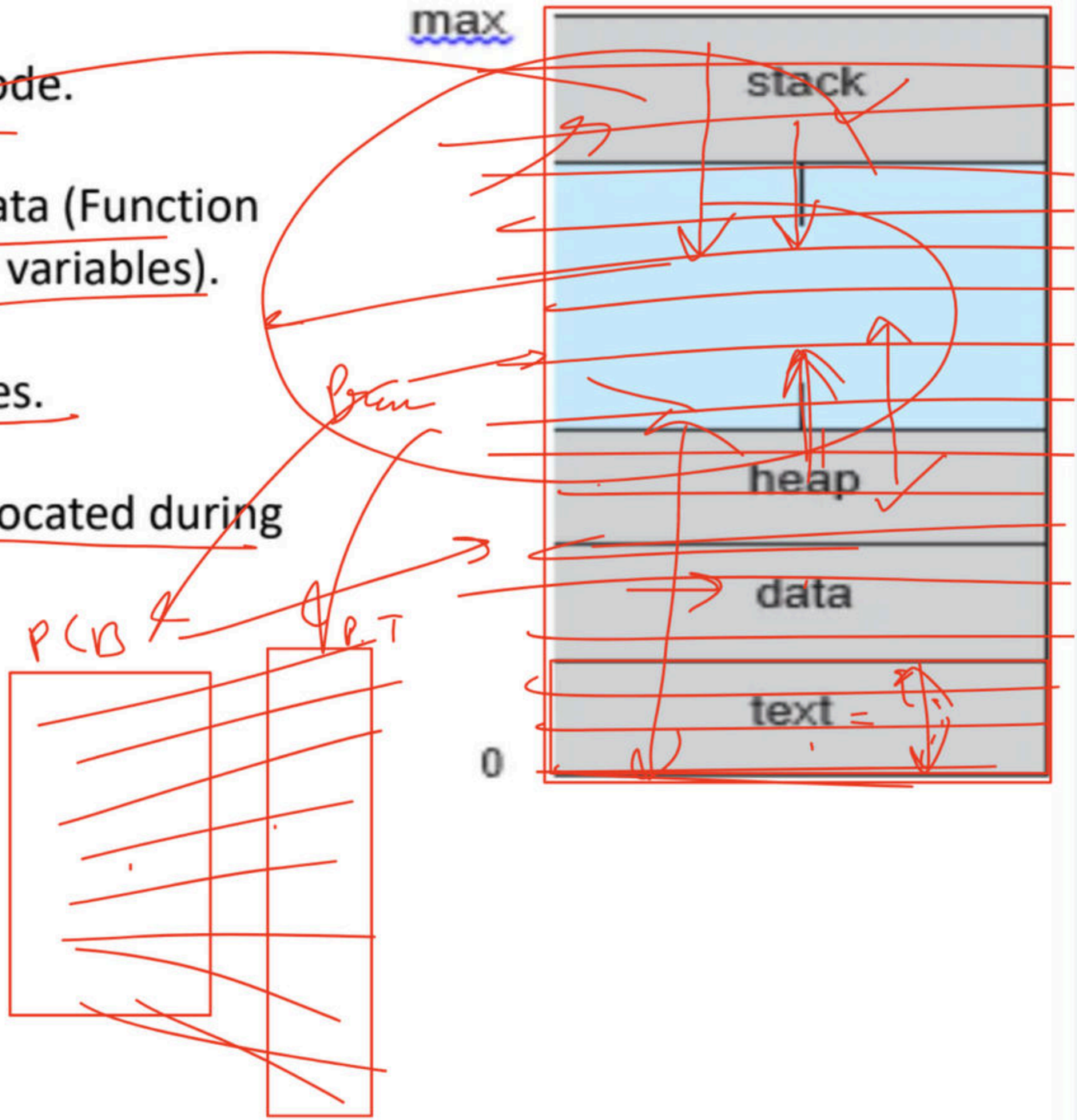

L.A
OS
PA
CPU
Cache
M.M
P.A.S
8KB
V.M
L.A.S
1TB
32
31
PD
S.M

- Even if two processes may be associated with same program, they will be considered as two separate execution sequences and are totally different process.

- For instance, if a user has invoked many copies of web browser program, each copy will be treated as separate process. even though the text section is same but the data, heap and stack sections can vary.



- A Process consists of following sections:
  - **Text section:** also known as Program Code.
  - **Stack:** which contains the temporary data (Function Parameters, return addresses and local variables).
  - **Data Section:** containing global variables.
  - **Heap:** which is memory dynamically allocated during process runtime.



Q Process is a: **(ISRO 2009)**

a) A program in high level language kept on disk

b) Contents of main memory

c) A program in execution

d) A job in secondary memory

Break

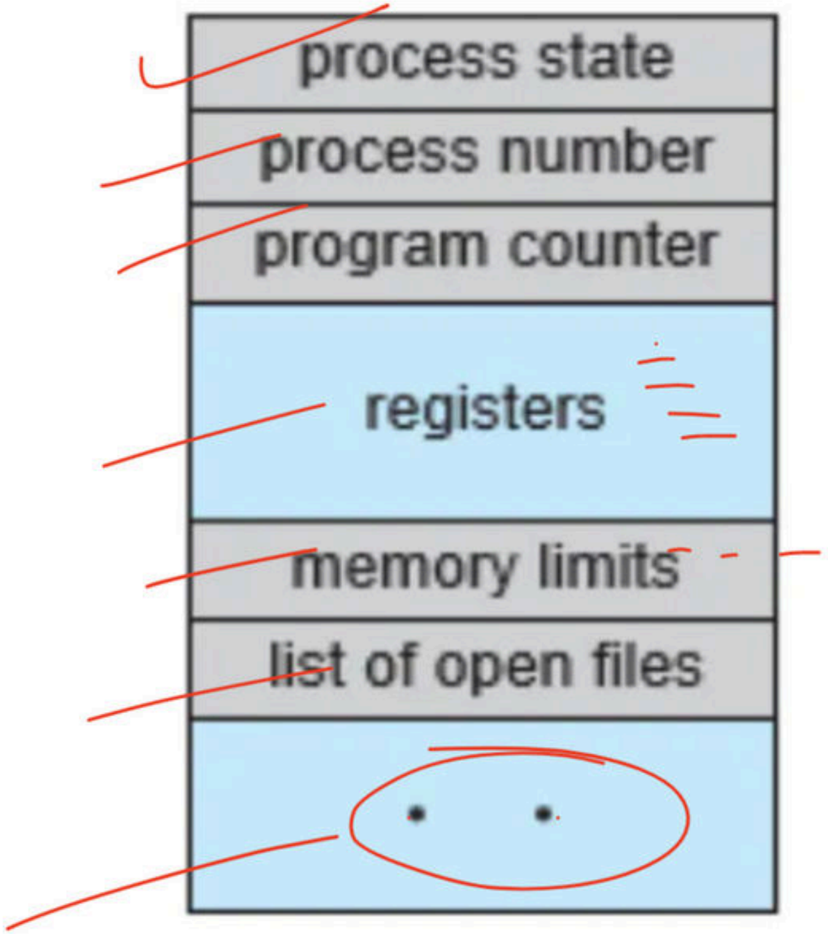
Process Control Block (PCB)

- Each process is represented in the operating system by a process control block (PCB) — also called a task control block.

- PCB simply serves as the repository for any information that may vary from process to process. It contains many pieces of information associated with a specific process, including these:
  - **Process state:** The state may be new, ready, running, waiting, halted, and so on.

  - **Program counter:** The counter indicates the address of the next instruction to be executed for this process.

  - **CPU registers:** The registers vary in number and type, depending on the computer architecture. They include accumulators, index registers, stack pointers, and general-purpose registers, plus any condition-code information. Along with the program counter, this state information must be saved when an interrupt occurs, to allow the process to be continued correctly afterward.

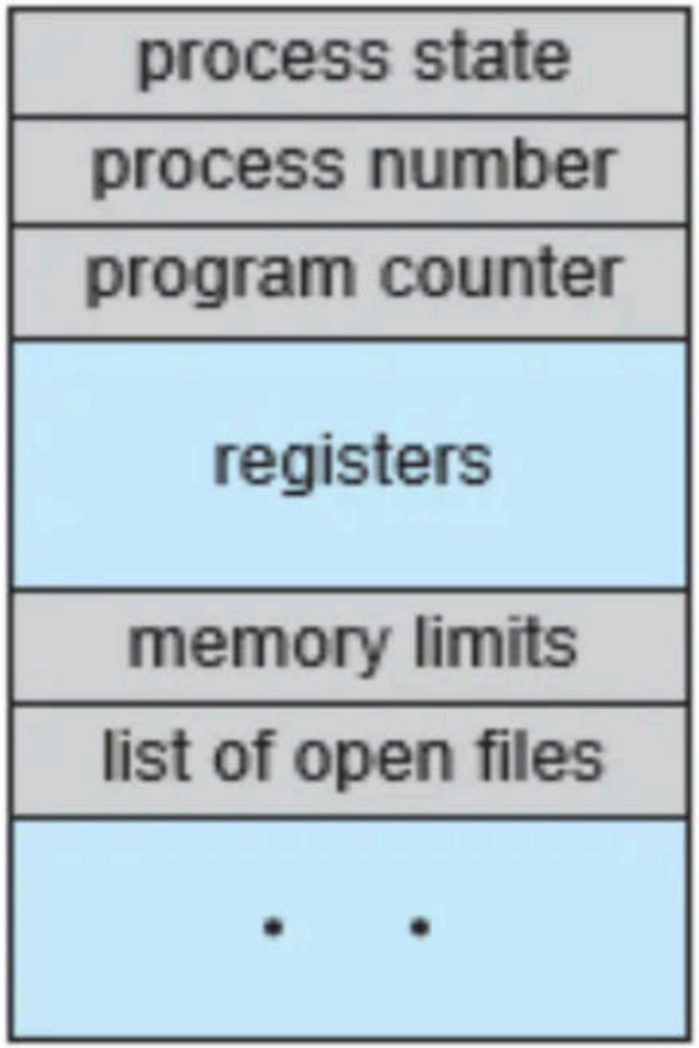


- **<u>CPU-scheduling information</u>:** This information includes a process priority, pointers to scheduling queues, and any other scheduling parameters.

- **<u>Memory-management information</u>:** This information may include such items as the value of the base and limit registers and the page tables, or the segment tables, depending on the memory system used by the operating system.

- **<u>Accounting information</u>:** This information includes the amount of CPU and real time used, time limits, account numbers, job or process numbers, and so on.

- **<u>I/O status information</u>:** This information includes the list of I/O devices allocated to the process, a list of open files, and so on.

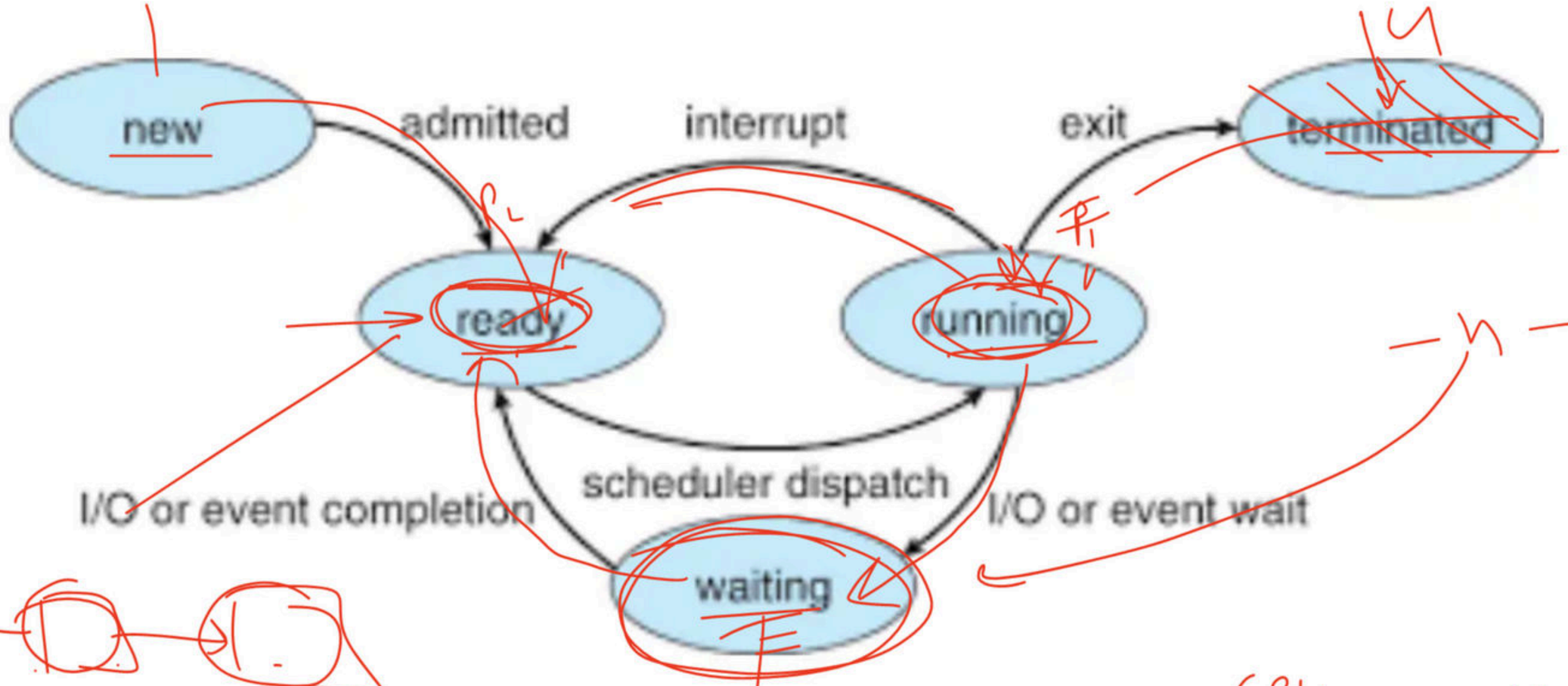


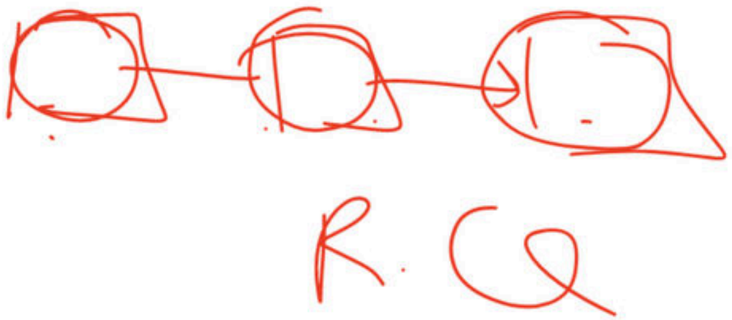
Break

The Human Life Cycle

Process States

- A Process changes states as it executes. The state of a process is defined in parts by the current activity of that process. A process may be in one of the following states:
  - **New:** The process is being created.
  - **Running:** Instructions are being executed.
  - **Waiting (Blocked):** The process is waiting for some event to occur (such as an I/O completion or reception of a signal).
  - **Ready:** The process is waiting to be assigned to a processor.
  - **Terminated:** The process has finished execution.

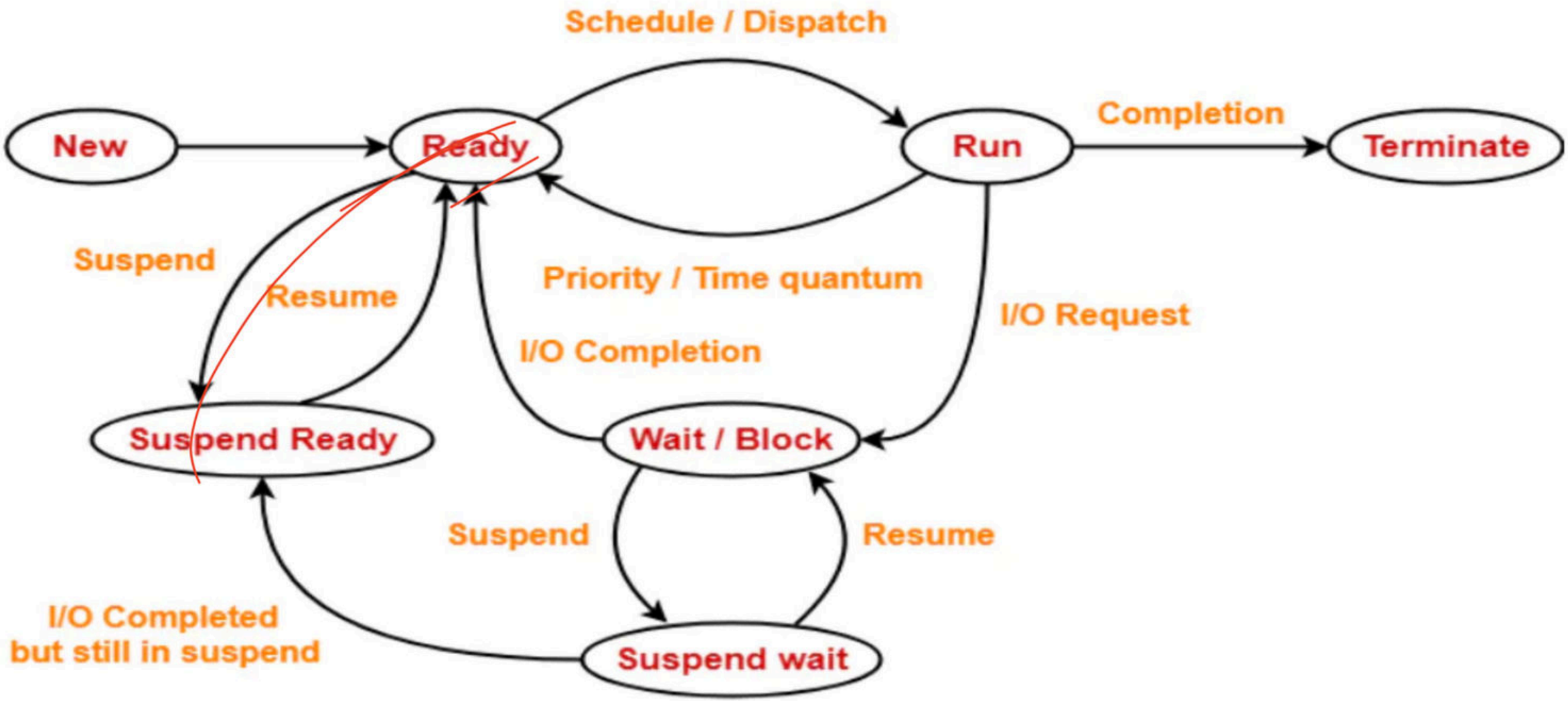

new
admitted
interrupt
exit
terminated
ready
running
scheduler dispatch
I/O or event completion
I/O or event wait
waiting


R.Q

I/O

CPU

I/O



Process State Diagram

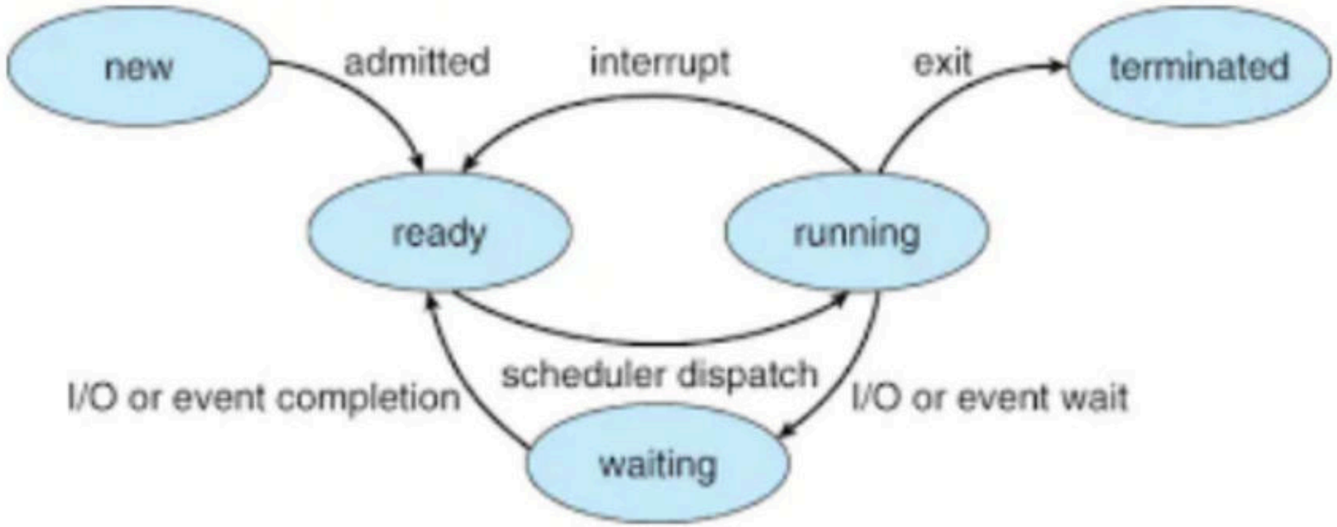
Break

Q How many states can a process be in? **(NET-DEC-2007)**

a) 2 **b)** 3 **c)** 4 **d)** 5

Q What is the minimum and maximum number of processes that can be in the ready, run, and blocked states, if total number of process is n?

	Min	Max
Ready		
Run		
Block		



Q On a system with P CPUs and n processes, what is the minimum and maximum number of processes that can be in the ready, run, and blocked states?

	Min	Max
Ready		
Run		
Block		

Q The state of a process after it encounters an I/O instruction is **(ISRO 2013)**

a) ready **b)** blocked **c)** idle **d)** running

Q There are three processes in the ready queue. When the currently running process requests for I/O how many process switches take place? **(ISRO 2011)**

a) 1 **b)** 2 **c)** 3 **d)** 4

Q What is the ready state of a process?

a) when process is scheduled to run after some execution

b) when process is using the CPU

c) when process is unable to run until some task has been completed

d) none of the mentioned

Q A task in a blocked state

a) is executable

b) is running

c) must still be placed in the run queues

d) is waiting for some temporarily unavailable resources

Break

Q Which combination of the following features will suffice to characterize an OS as a multi-programmed OS? **(GATE-2002) (2 Marks)**

(a) More than one program may be loaded into main memory at the same time for execution.

(b) If a program waits for certain events such as I/O, another program is immediately scheduled for execution.

(c) If the execution of program terminates, another program is immediately scheduled for execution.

(A) a **(B)** a and b **(C)** a and c **(D)** a, b and c

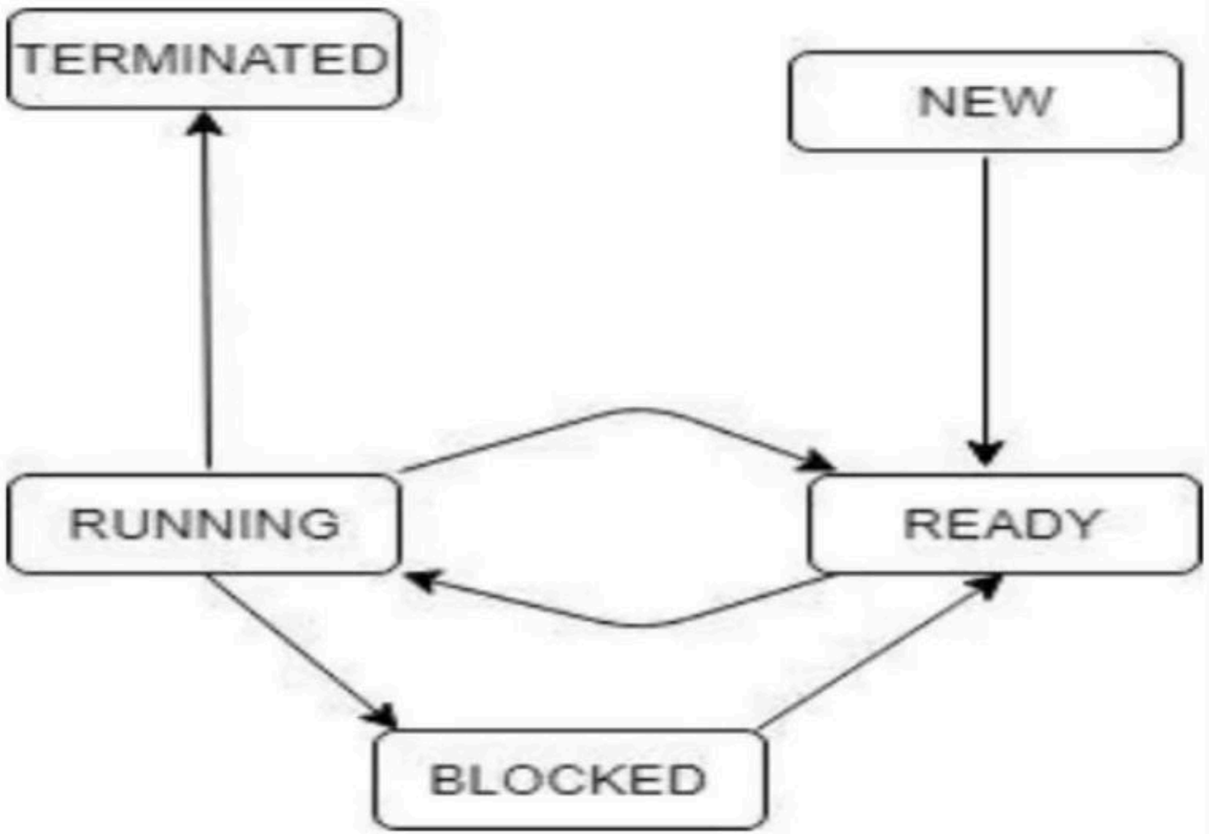
Q The process state transition diagram in below figure is representative of Untitled Diagram **(GATE-1996) (1 Marks)**

a) a batch operating system

b) an operating system with a preemptive schedular

c) an operating system with a non-preemptive schedular

d) a uni-programmed operating system



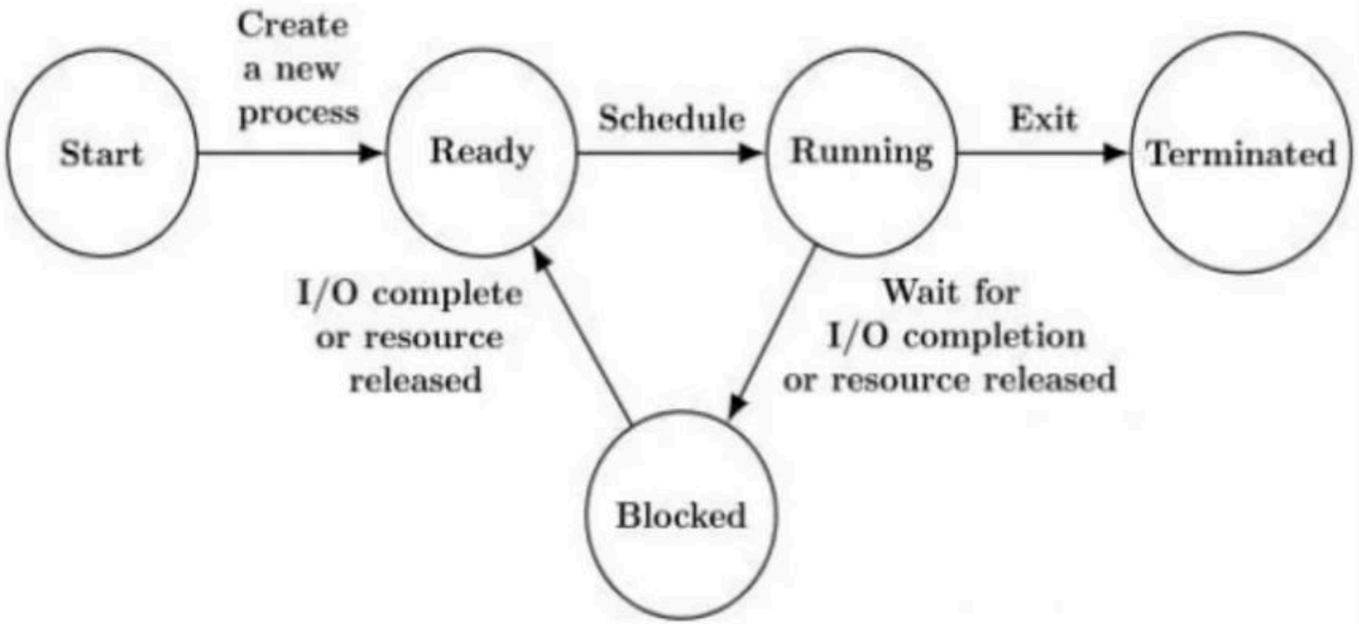
Q The process state transition diagram of an operating system is as given below. Which of the following must be FALSE about the above operating system? **(GATE-2006) (1 Marks)**

a) It is a multiprogram operating system

b) It uses preemptive scheduling

c) It uses non-preemptive scheduling

d) It is a multi-user operating system



Q In the following process state transition diagram for a uniprocessor system, assume that there are always some processes in the ready state:

Now consider the following statements:

I) If a process makes a transition D, it would result in another process making transition A immediately.

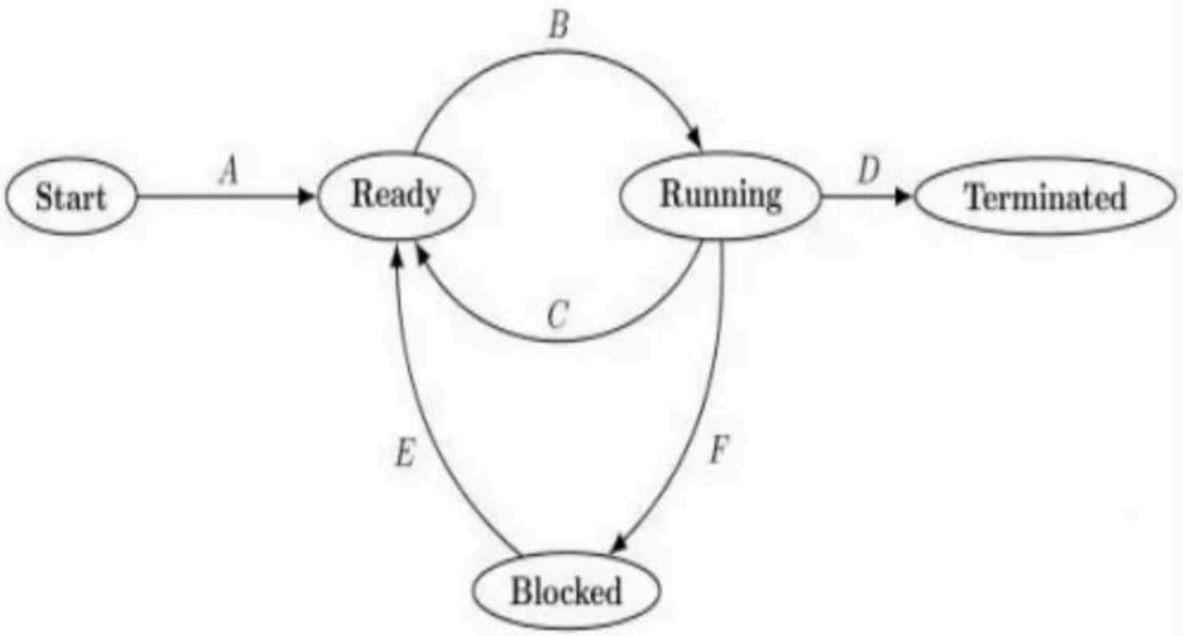
II) A process P2 in blocked state can make transition E while another process P1 is in running state.

III) The OS uses preemptive scheduling.

IV) The OS uses non-preemptive scheduling.

Which of the above statements are TRUE? **(GATE-2009) (2 Marks)**

a) I and II **b)** I and III **c)** II and III **d)** II and IV



Q A computer handles several interrupt sources of which the following are relevant for this question.

- Interrupt from CPU temperature sensor (raises interrupt if CPU temperature is too high)
- Interrupt from Mouse (raises interrupt if the mouse is moved or a button is pressed)
- Interrupt from Keyboard (raises interrupt when a key is pressed or released)
- Interrupt from Hard Disk (raises interrupt when a disk read is completed)

Which one of these will be handled at the HIGHEST priority? **(GATE - 2011) (1 Marks)**

(A) Interrupt from Hard Disk

(B) Interrupt from Mouse

(C) Interrupt from Keyboard

(D) Interrupt from CPU temperature sensor

Q Match the following: **(NET-SEP-2013)**

List – I	List - II
Process state	Reason for transition
a) Ready → Running	i. Request made by the process is satisfied or an event for which it was waiting occurs.
b) Blocked → Ready	ii. Process wishes to wait for some action by another process.
c) Running → Blocked	iii. The process is dispatched.
d) Running → Ready	iv. The process is pre-empted.

	a	b	c	d
a)	iii	i	ii	iv
b)	iv	i	iii	ii
c)	iv	iii	i	ii
d)	Iii	iii	ii	i

Q Which is the correct definition of a valid process transition in an operating system? **(NET-JUNE-2005)**

a) Wake up : Ready → Running

b) Dispatch : Ready→ Running

c) Block: Ready → Running

d) Timer run out: Ready → Blocked

Break

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

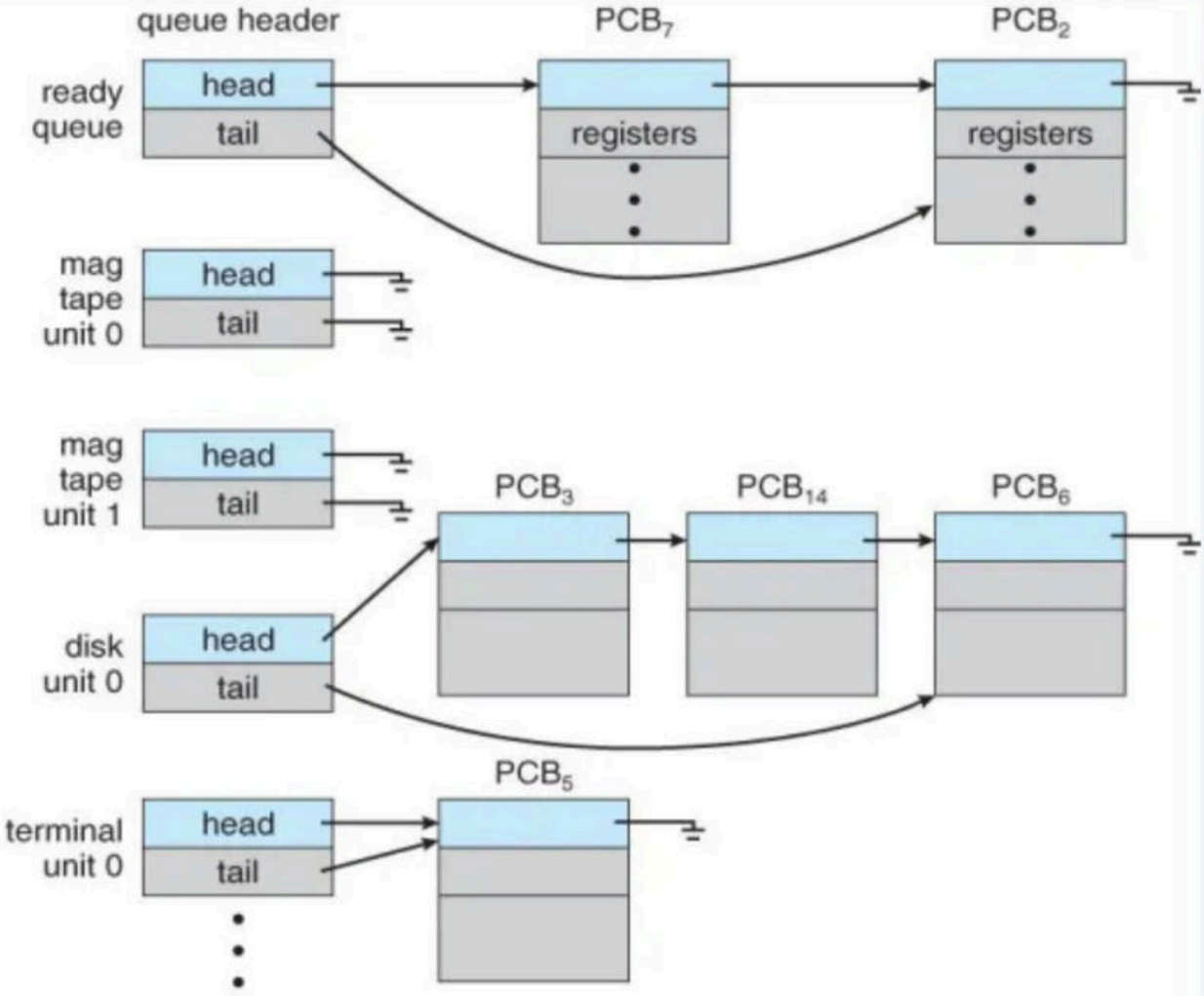
- The objective of time sharing is to switch the CPU among processes so frequently that users can interact with each program while it is running.

- To meet these objectives, the process scheduler selects an available process (possibly from a set of several available processes) for execution on the CPU. Note: For a single-processor system, there will never be more than one running process.

- Scheduling Queues - As processes enter the system, they are put into a job queue, which consists of all processes in the system.

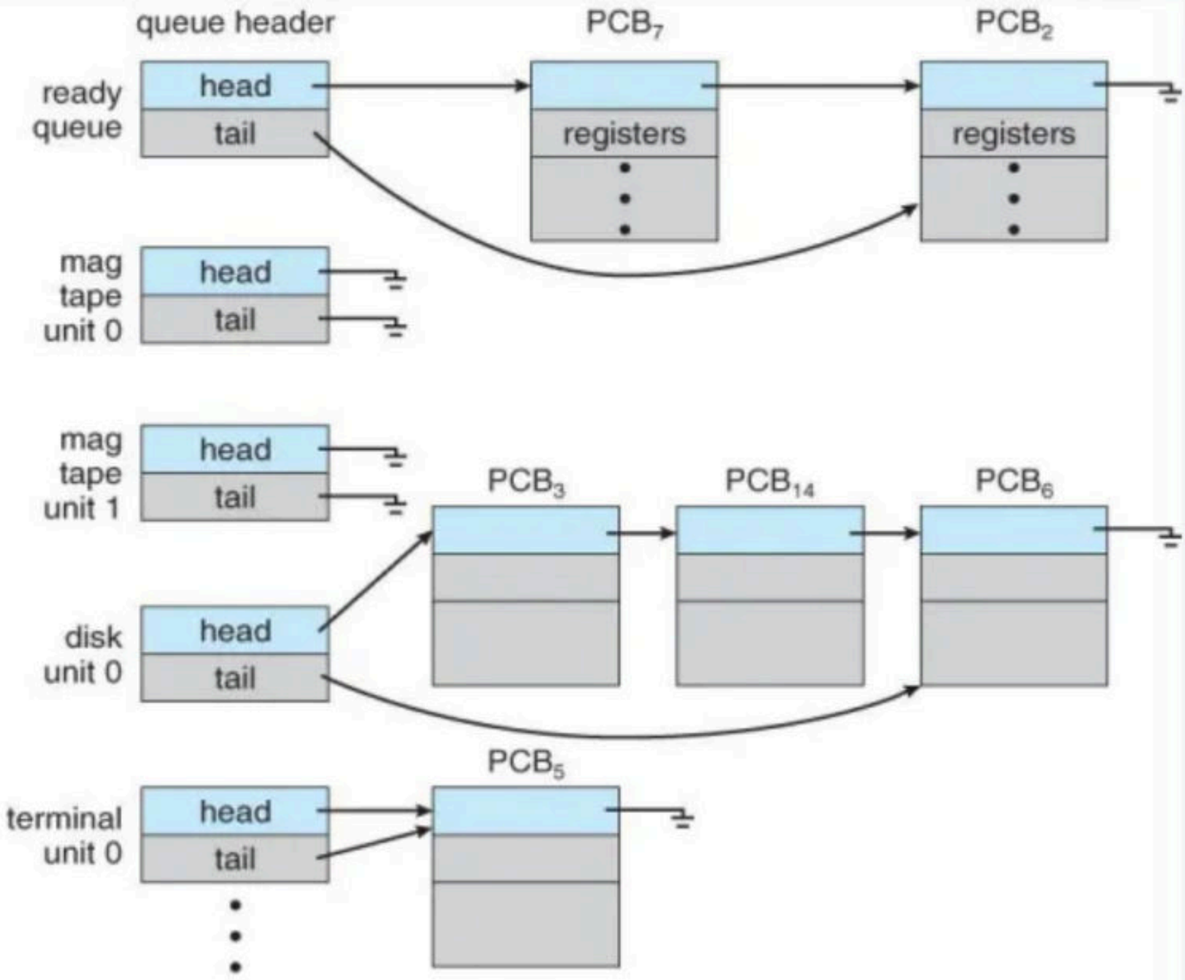
- The processes that are residing in main memory and are ready to execute are kept on a list called the ready queue. This queue is generally stored as a linked list.

- A ready-queue header contains pointers to the first and final PCBs in the list. Each PCB includes a pointer field that points to the next PCB in the ready queue.

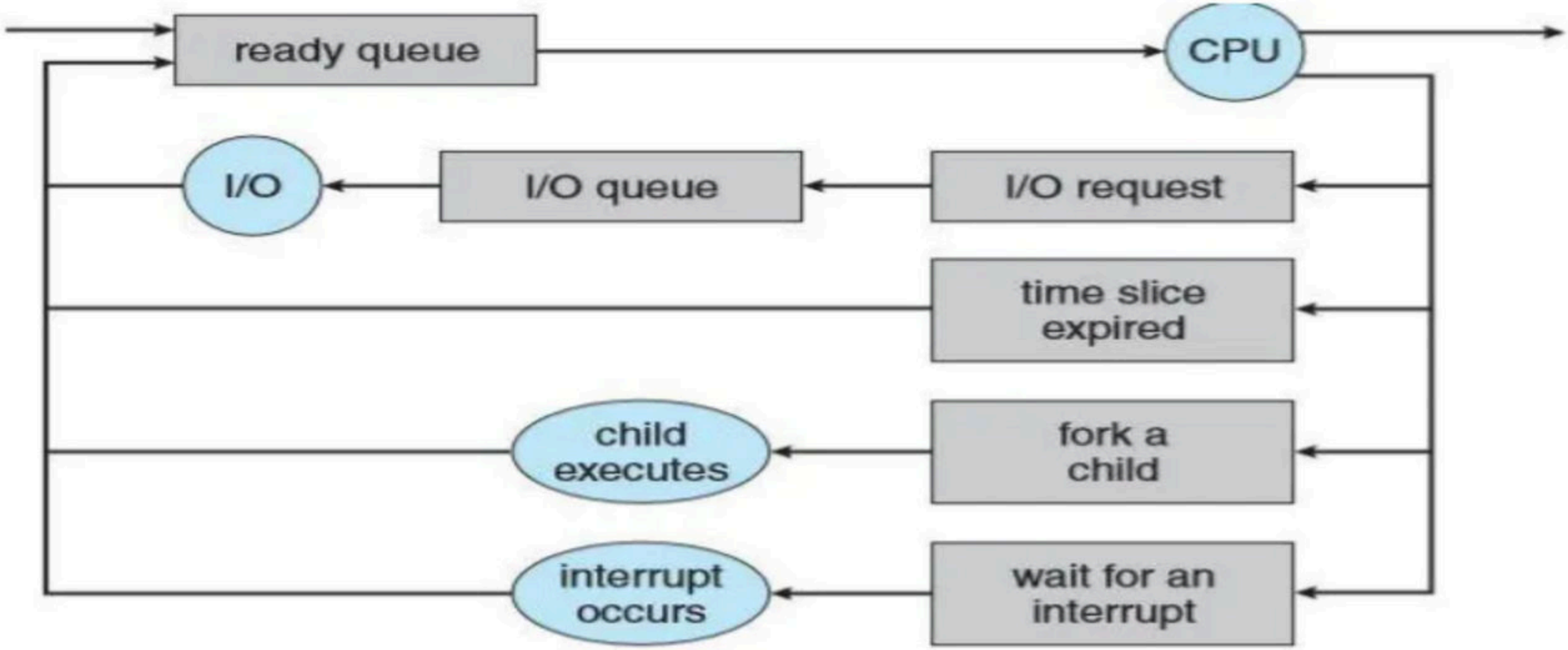


- The system also includes other queues. When a process is allocated the CPU, it executes for a while and eventually quits, is interrupted, or waits for the occurrence of a particular event, such as the completion of an I/O request.

- Suppose the process makes an I/O request to a shared device, such as a disk. Since there are many processes in the system, the disk may be busy with the I/O request of some other process. The process therefore may have to wait for the disk. The list of processes waiting for a particular I/O device is called a device queue. Each device has its own device queue.

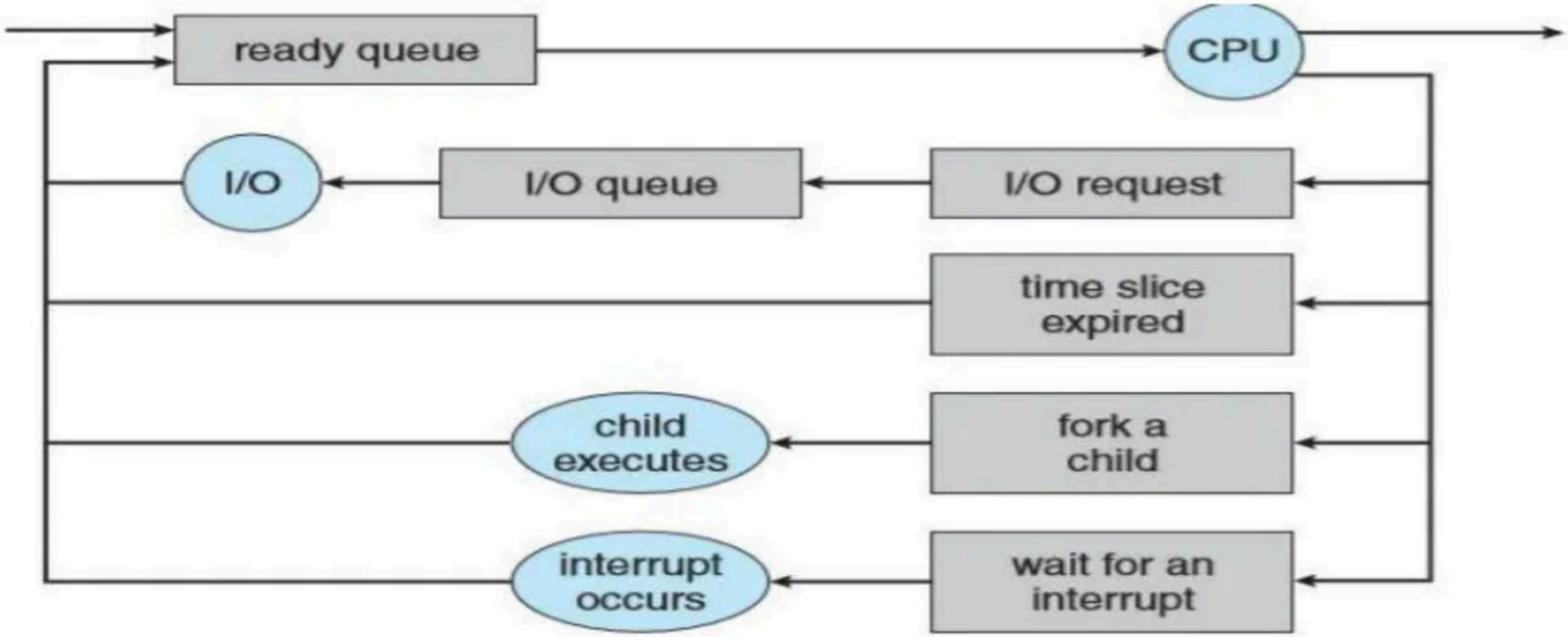


- Each rectangular box represents a queue. Two types of queues are present: the ready queue and a set of device queues. The circles represent the resources that serve the queues. A new process is initially put in the ready queue. It waits there until it is selected for execution, or dispatched.



Queueing-diagram representation of process scheduling.

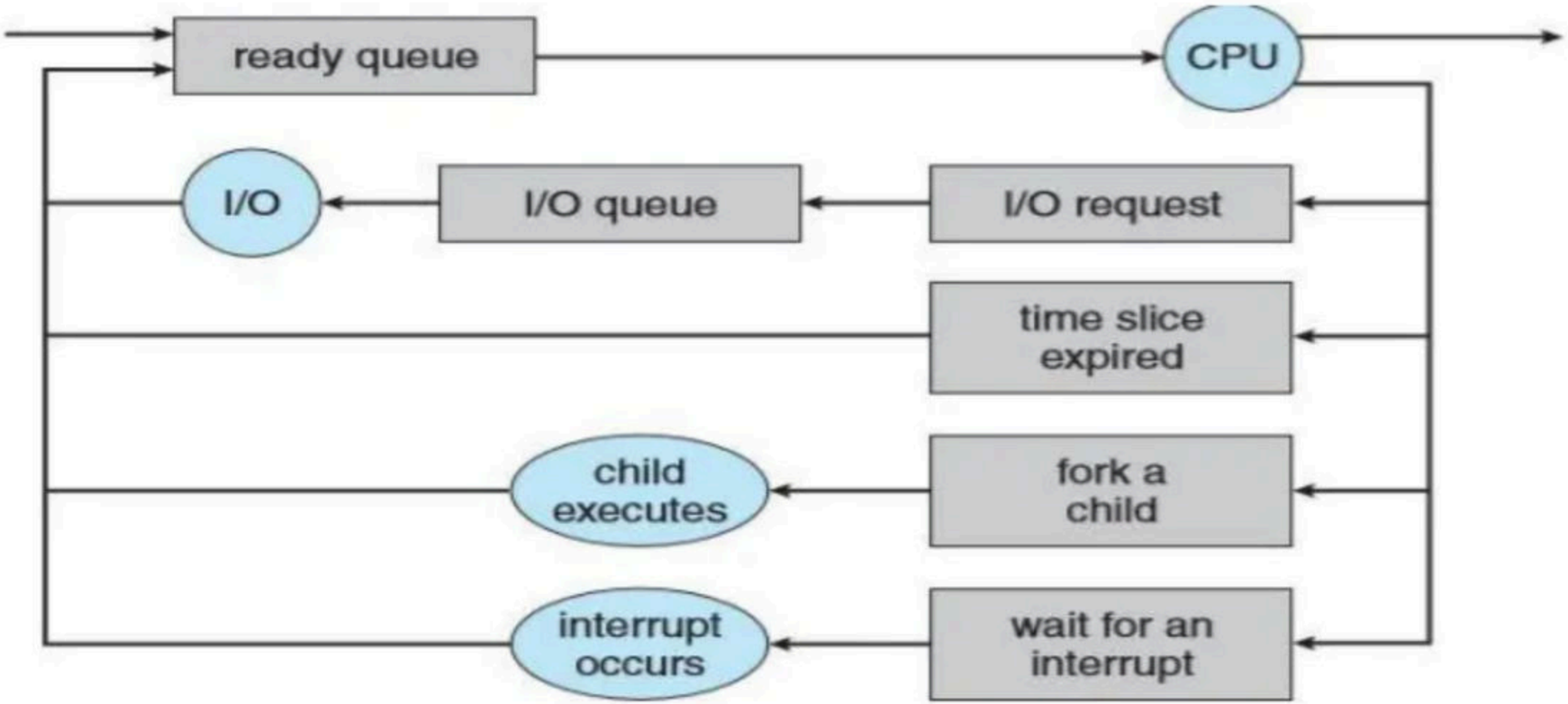
- Once the process is allocated the CPU and is executing, one of several events could occur:
  - The process could issue an I/O request and then be placed in an I/O queue.
  - The process could create a new child process and wait for the child's termination.
  - Time slice expired, the process could be removed forcibly from the CPU, as a result of an interrupt, and be put back in the ready queue.



Queueing-diagram representation of process scheduling.

- In the first two cases the process eventually switches from the waiting state to the ready state and is then put back in the ready queue.

- A process continues this cycle until it terminates, at which time it is removed from all queues and has its PCB and resources deallocated.



Queueing-diagram representation of process scheduling.

Break

Schedulers

- **Schedulers:** A process migrates among the various scheduling queues throughout its lifetime. The operating system must select, for scheduling purposes, processes from these queues in some fashion. The selection process is carried out by the appropriate scheduler.

- **Types of Schedulers**
  - **Long Term Schedulers (LTS)/Spooler:** In multiprogramming os, more processes are submitted than can be executed immediately. These processes are spooled to a mass-storage device (typically a disk), where they are kept for later execution. The long-term scheduler, or job scheduler, selects processes from this pool and loads them into memory for execution.

  - **Short Term Scheduler (STS):** The short-term scheduler, or CPU scheduler, selects from among the processes that are ready to execute and allocates the CPU to one of them.

- **Difference between LTS and STS** - The primary distinction between these two schedulers lies in frequency of execution.

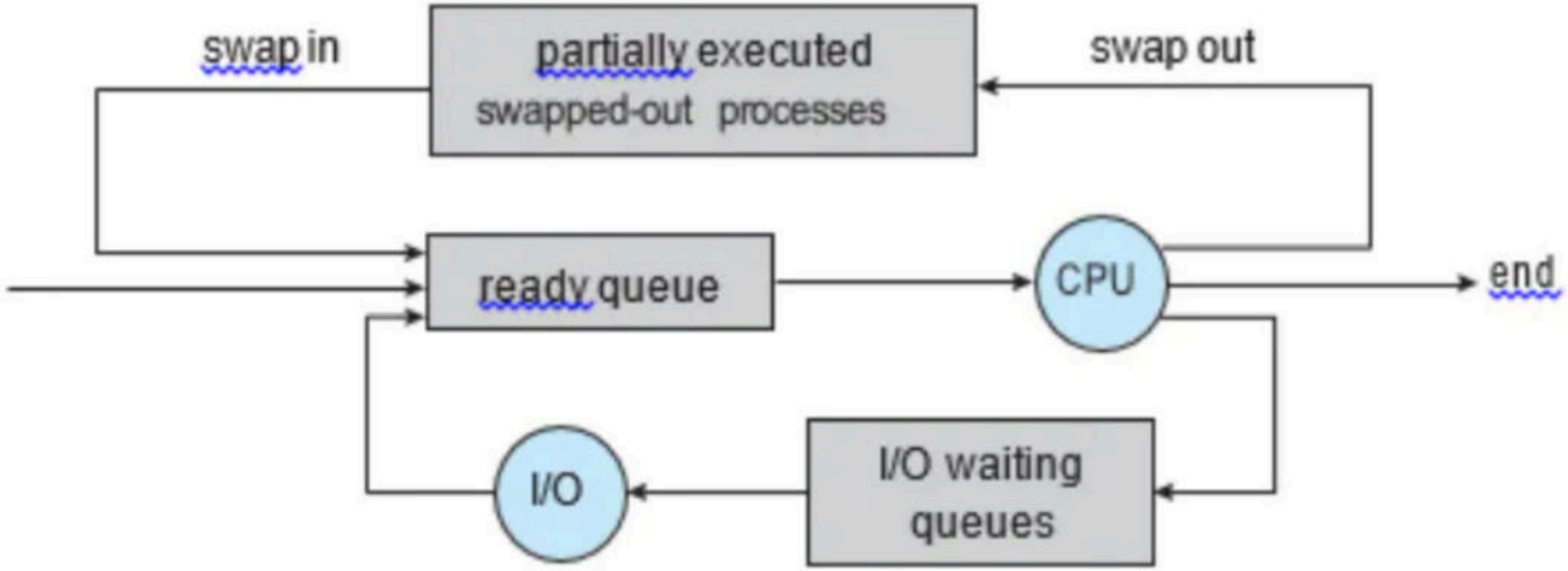
- The short-term scheduler must select a new process for the CPU frequently. A process may execute for only a few milliseconds before waiting for an I/O request. Often, the short-term scheduler executes at least once every 100 milliseconds.

- Because of the short time between executions, the short-term scheduler must be fast. The long-term scheduler on the other hand executes much less frequently; minutes may separate the creation of one new process and the next.

Degree of Multiprogramming - The number of processes in memory is known as Degree of Multiprogramming.

- The long-term scheduler controls the degree of multiprogramming as it is responsible for bringing in the processes to main memory.

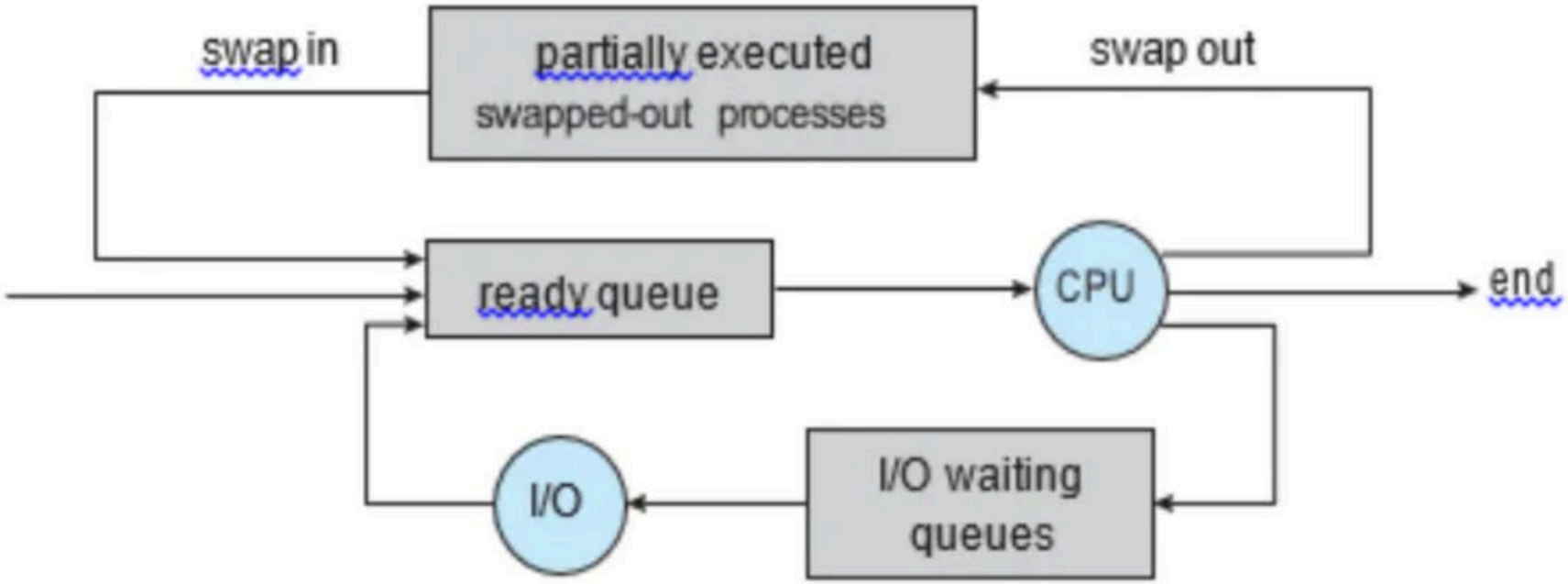
- If the degree of multiprogramming is stable, then the average rate of process creation must be equal to the average departure rate of processes leaving the system. So, this means the long-term scheduler may need to be invoked only when a process leaves the system. Because of the longer interval between executions, the long-term scheduler can afford to take more time to decide which process should be selected for execution.



Medium-term scheduler: The key idea behind a medium-term scheduler is that sometimes it can be advantageous to remove a process from memory (and from active contention for the CPU) and thus reduce the degree of multiprogramming.

Later, the process can be reintroduced into memory, and its execution can be continued where it left off. This scheme is called swapping.

The process is swapped out, and is later swapped in, by the medium-term scheduler. Swapping may be necessary to improve the process mix or because a change in memory requirements has overcommitted available memory, requiring memory to be freed up.



- **Dispatcher** - The dispatcher is the module that gives control of the CPU to the process selected by the short-term scheduler.

- This function involves the following: Switching context, switching to user mode, jumping to the proper location in the user program to restart that program.

- The dispatcher should be as fast as possible, since it is invoked during every process switch. The time it takes for the dispatcher to stop one process and start another running is known as the dispatch latency.

Q A software to create a Job Queue is called..... **(NET-DEC-2006)**

a) Linkage editor

b) Interpreter

c) Driver

d) Spooler

Q A special software that is used to create a job queue is called **(NET-JUNE-2011)**

(A) Drive

(B) Spooler

(C) Interpreter

(D) Linkage editor

Q Which module gives control of the CPU to the process selected by the short - term scheduler? **(NET-NOV-2017)**

a) Dispatcher

b) Interrupt

c) Scheduler

d) Threading

Break

CPU Bound and I/O Bound Processes

- A process execution consists of a cycle of CPU execution or wait and i/o execution or wait.
- Normally a process alternates between two states.
- Process execution begin with the CPU burst that may be followed by a i/o burst, then another CPU and i/o burst and so on.
- Eventually in the last will end up on CPU burst. So, process keep switching between the CPU and i/o during execution.

- **<u>I/O Bound Processes:</u>** An I/O-bound process is one that spends more of its time doing I/O than it spends doing computations.

- **<u>CPU Bound Processes</u>:** A CPU-bound process, generates I/O requests infrequently, using more of its time doing computations.

- It is important that the long-term scheduler select a good process mix of I/O-bound and CPU-bound processes.

- If all processes are I/O bound, the ready queue will almost always be empty, and the short-term scheduler will have little to do.

- Similarly, if all processes are CPU bound, the I/O waiting queue will almost always be empty, devices will go unused, and again the system will be unbalanced.

- So, to have the best system performance LTS needs to select a good combination of I/O and CPU Bound processes.

Break

Context Switch

- When an interrupt occurs, the system needs to save the current context of the process running on the CPU so that it can restore that context when its processing is done.

- Switching the CPU to another process requires performing a state save of the current process and a state restore of a different process. This task is known as a context switch. When a context switch occurs, the kernel saves the context of the old process in its PCB and loads the saved context of the new process scheduled to run. Context-switch time is pure overhead, because the system does no useful work while switching.

- Switching speed varies from machine to machine, depending on the memory speed, the number of registers that must be copied, and the existence of special instructions (such as a single instruction to load or store all registers). A typical speed is a few milliseconds.

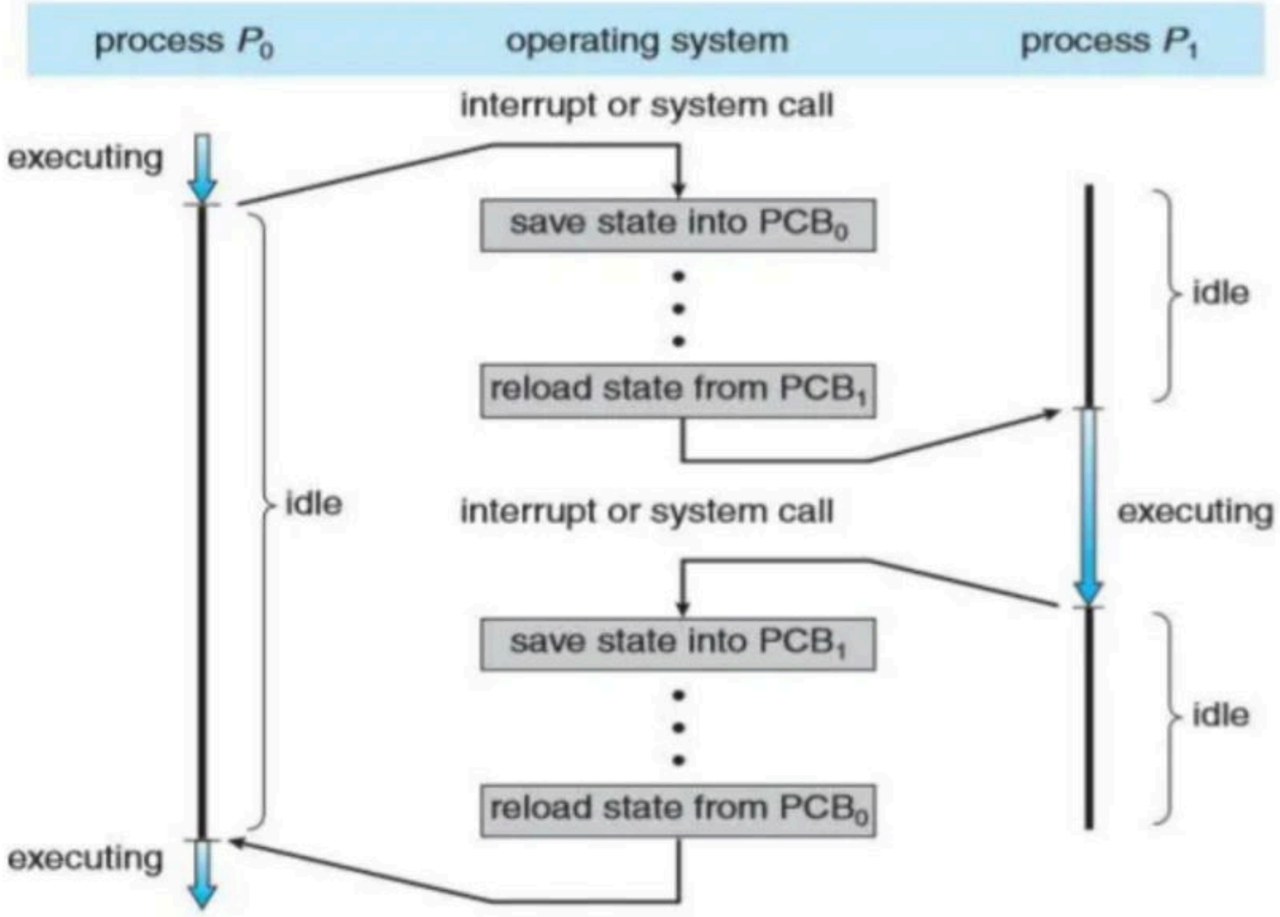


Diagram showing CPU switch from process to process.

Q Which of the following does not interrupt a running process? **(GATE-2001) (2 Marks)**

(A) A device

(B) Timer

(C) Scheduler process

(D) Power failure

Q Which is the correct definition of a valid process transition in an operating system?

a) Wake up: ready → running

b) Dispatch: ready → running

c) Block: ready → running

d) Timer runout: ready → running

Q A process stack does not contain

a) function parameters

b) local variables

c) return addresses

d) PID of child process

Q Loading operating system from secondary memory to primary memory is called..... **(NET-DEC-2006)**

(A) Compiling

(B) Booting

(C) Refreshing

(D) Reassembling