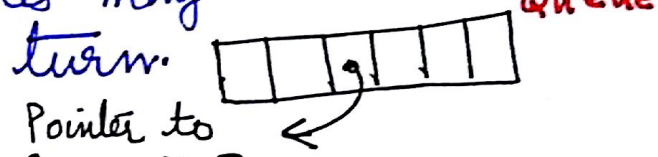
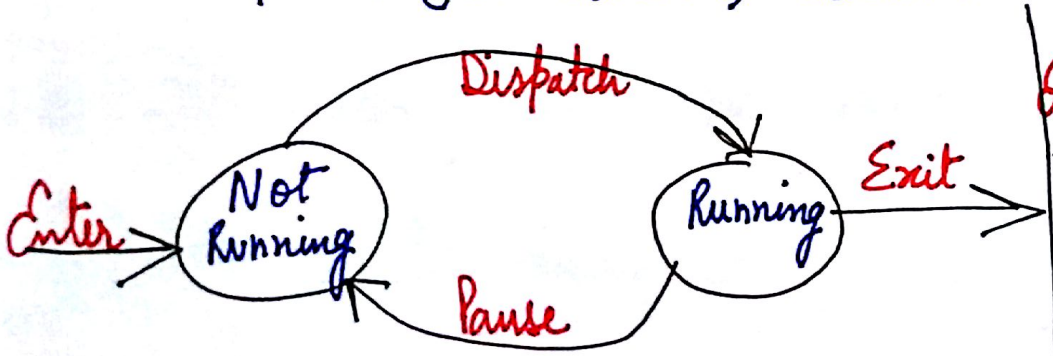
Process Modeling: In order to execute any program, a process has to be created.
List of instruction sequence is called **trace**. (interleaved traces)
Dispatcher :- Switch processor from one processor to another.

Example: 3 process with a dispatcher. (OS time out 6-insts.)

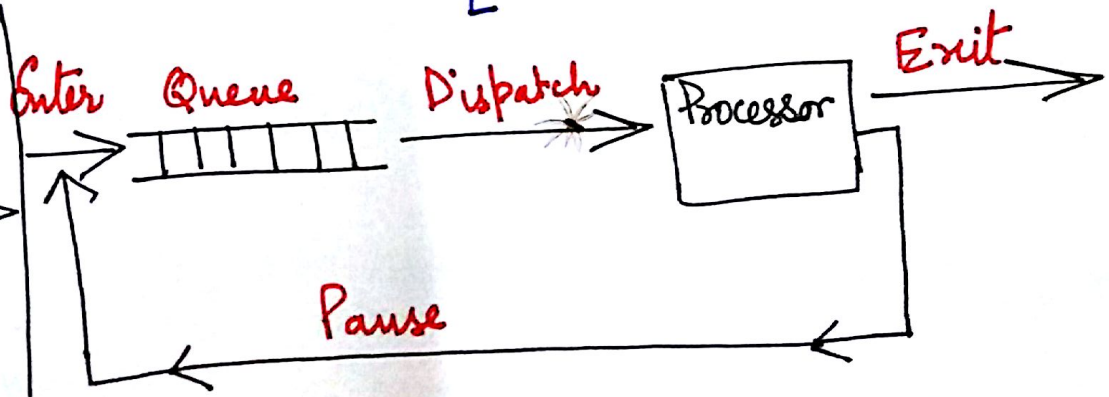
⇒ **Two state process modeling:** At any time process is either running or not running.

*) A new process is created with PCB indicating process state as not running.
*) Due to interrupt or OS time out these states may be switched.
*) Processes not running waits in a queue for their turn.
*) Time to time processor got interrupted and dispatcher (OS module) Select some process to run. [State transitions occur]

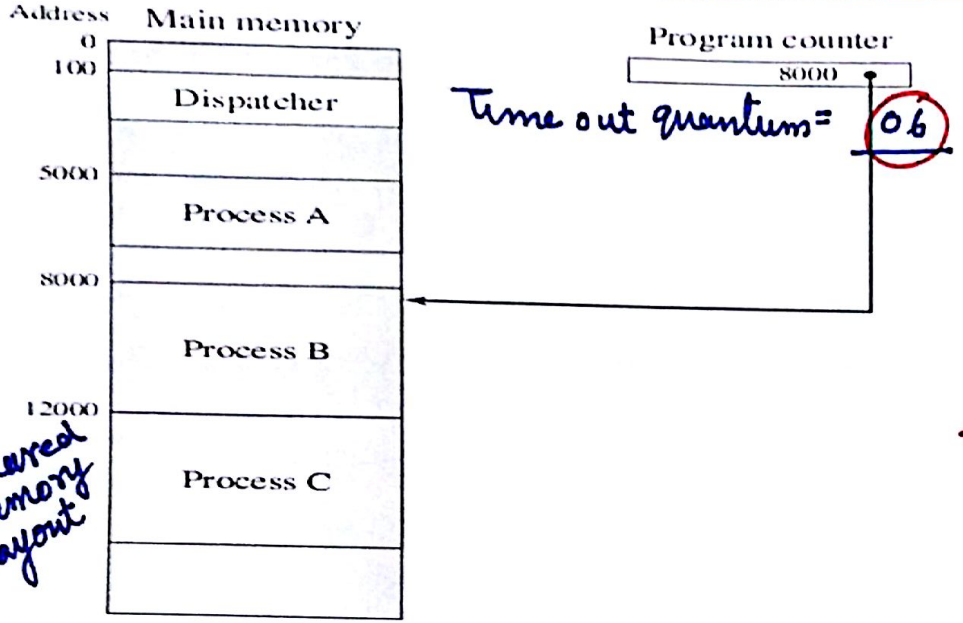




(A) State Transition



(B) Queuing diagram.



Shared Memory Layout

Time out quantum = 06

(a) Trace of process A	(b) Trace of process B	(c) Trace of process C
5000	8000	12000
5001	8001	12001
5002	8002	12002
5003	8003	12003
5004	I/O invoked	12004
5005		12005
5006		12006
5007		12007
5008		12008
5009		12009
5010		12010
5011		12011

5000 = Starting address of program of process A
8000 = Starting address of program of process B
12000 = Starting address of program of process C

28 instructions

1a

1	5000	27	12004
2	5001	28	12005
3	5002	----------Time-out	
4	5003	29	100
5	5004	30	101
6	5005	31	102
----------Time-out		32	103
7	100	33	104
8	101	34	105
9	102	35	5006
10	103	36	5007
11	104	37	5008
12	105	38	5009
13	8000	39	5010
14	8001	40	5011
15	8002	----------Time-out	
16	8003	41	100
----------I/O request		42	101
17	100	43	102
18	101	44	103
19	102	45	104
20	103	46	105
21	104	47	12006
22	105	48	12007
23	12000	49	12008
24	12001	50	12009
25	12002	51	12010
26	12003	52	12011
		----------Time-out	

Dispatcher

100 = Starting address of dispatcher program

52 instructions

Shaded areas indicate execution of dispatcher process;
first and third columns count instruction cycles;
second and fourth columns show address of instruction being executed

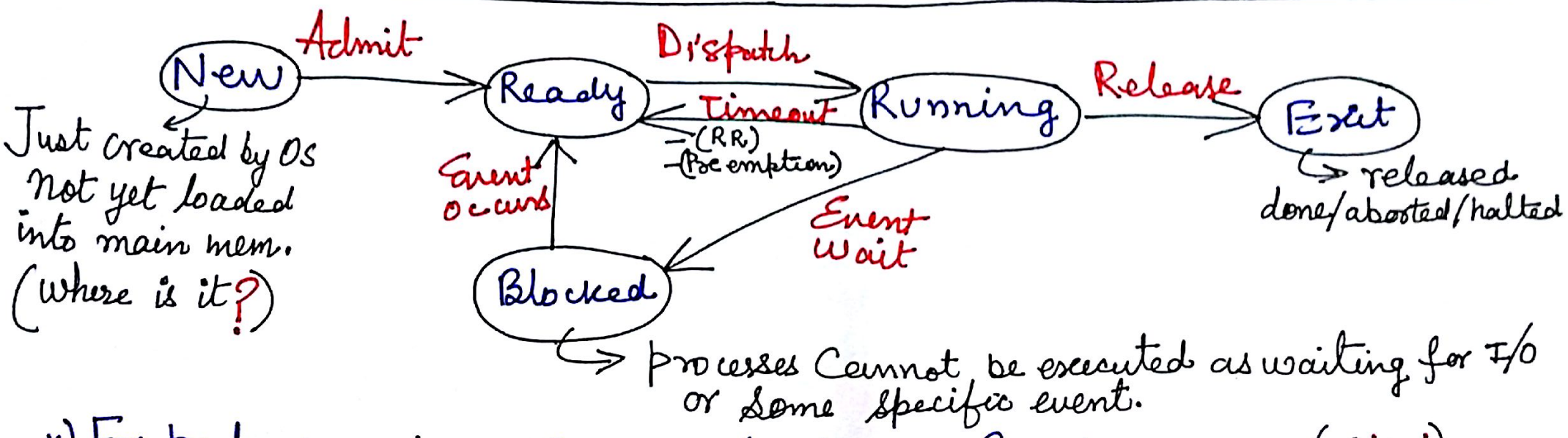
Moving towards five State model :- When queuing is effective. (every one is present) (FIFO) (2)

(*) Queue + Round Robin [Processer Strategy] each process run for a time quantum and then returns to queue. (unless blocked)

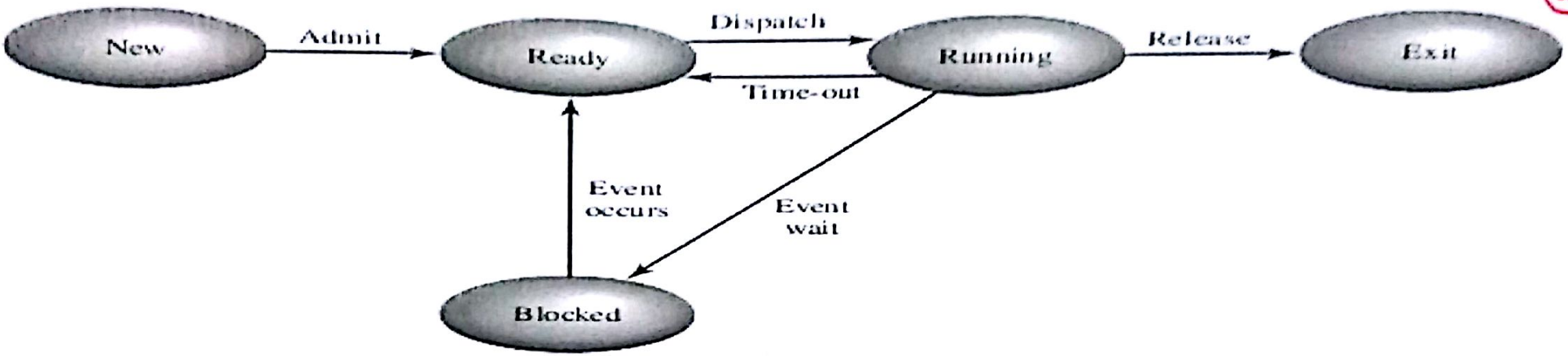
But the real problem is :

→ All NOT RUNNING processes are NOT ready to execute (may be waiting for I/O)

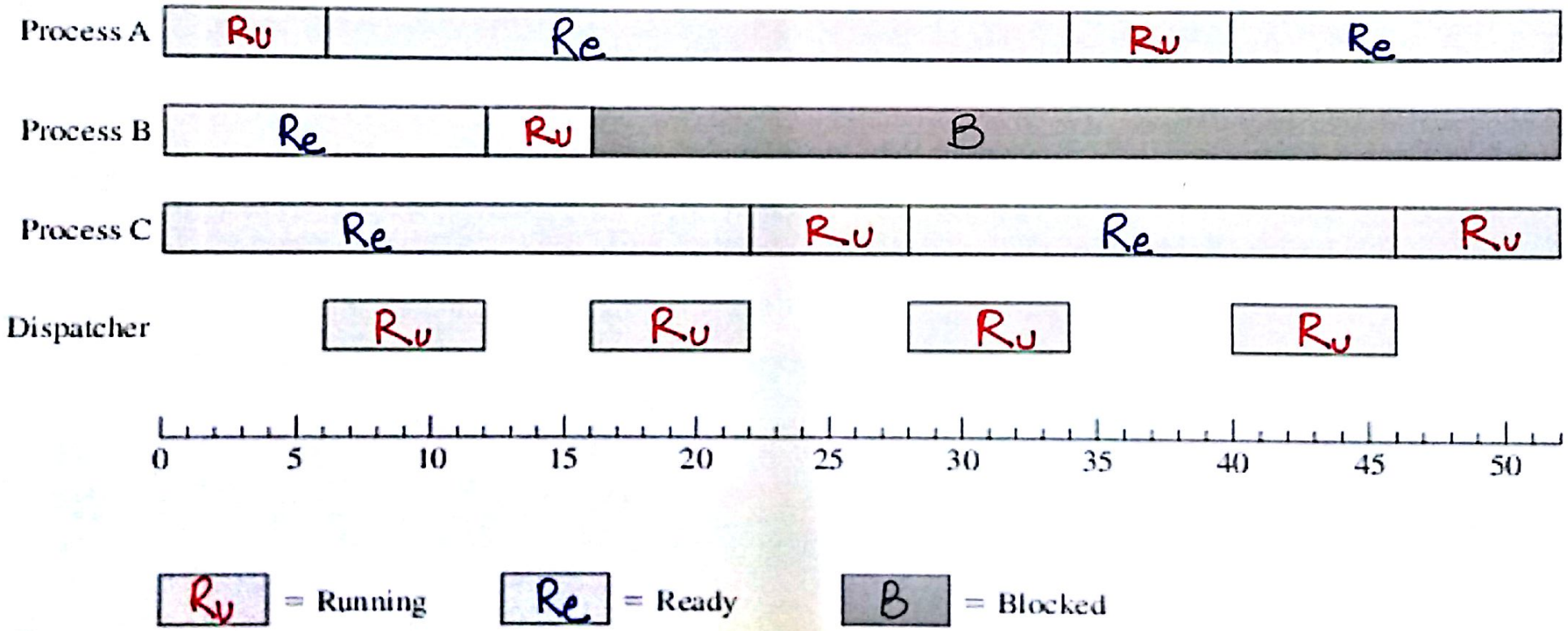
→ Dispatcher require (oldest + Ready). Hence Splitting of NOT RUNNING State is required. (1) Ready (2) Blocked (Both require separate Queue)



*) For performance issue OS may limit the Ready queue (New)



Five-State Process Model



Process States for the Trace of Figure 3.4

Refi 1: If an event occur some block processes must be moved from Blocked to Ready State. (Require full scan of blocked queue)

Sol Create individual queues for each event. (Multiple queuing [queue management])

Refine-02

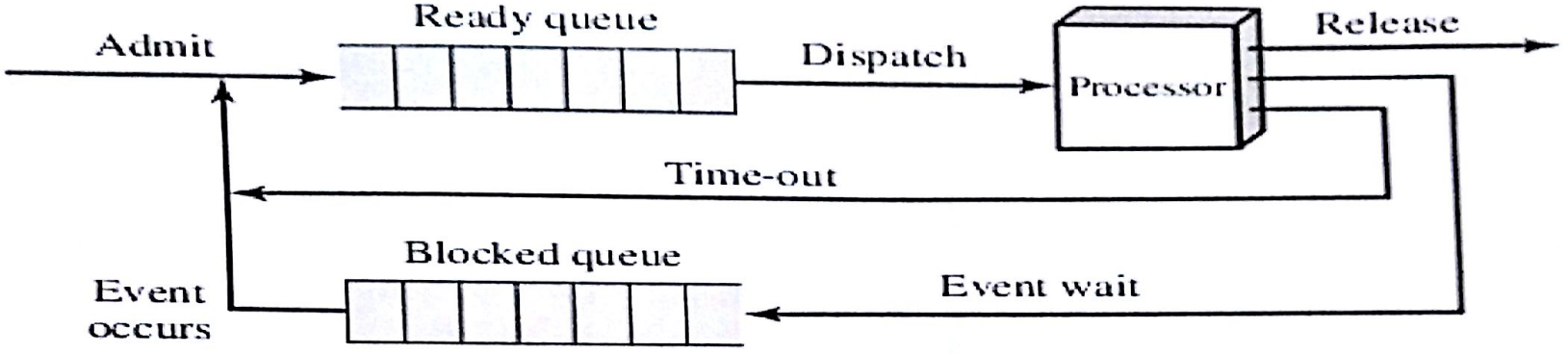
* If no priority → FIFO ; Else preemption.
With priority → individual priority queues.
Highest priority & longest waiting

Requirement of swapping:

→ 3 Principle states (Ready, Running and blocking)
→ All queued processes are in main memory.
→ Multiprograming is done as I/O are very slow hence CPU is idle mostly.
→ CPU is so fast that even after multiprograming CPU is idle.

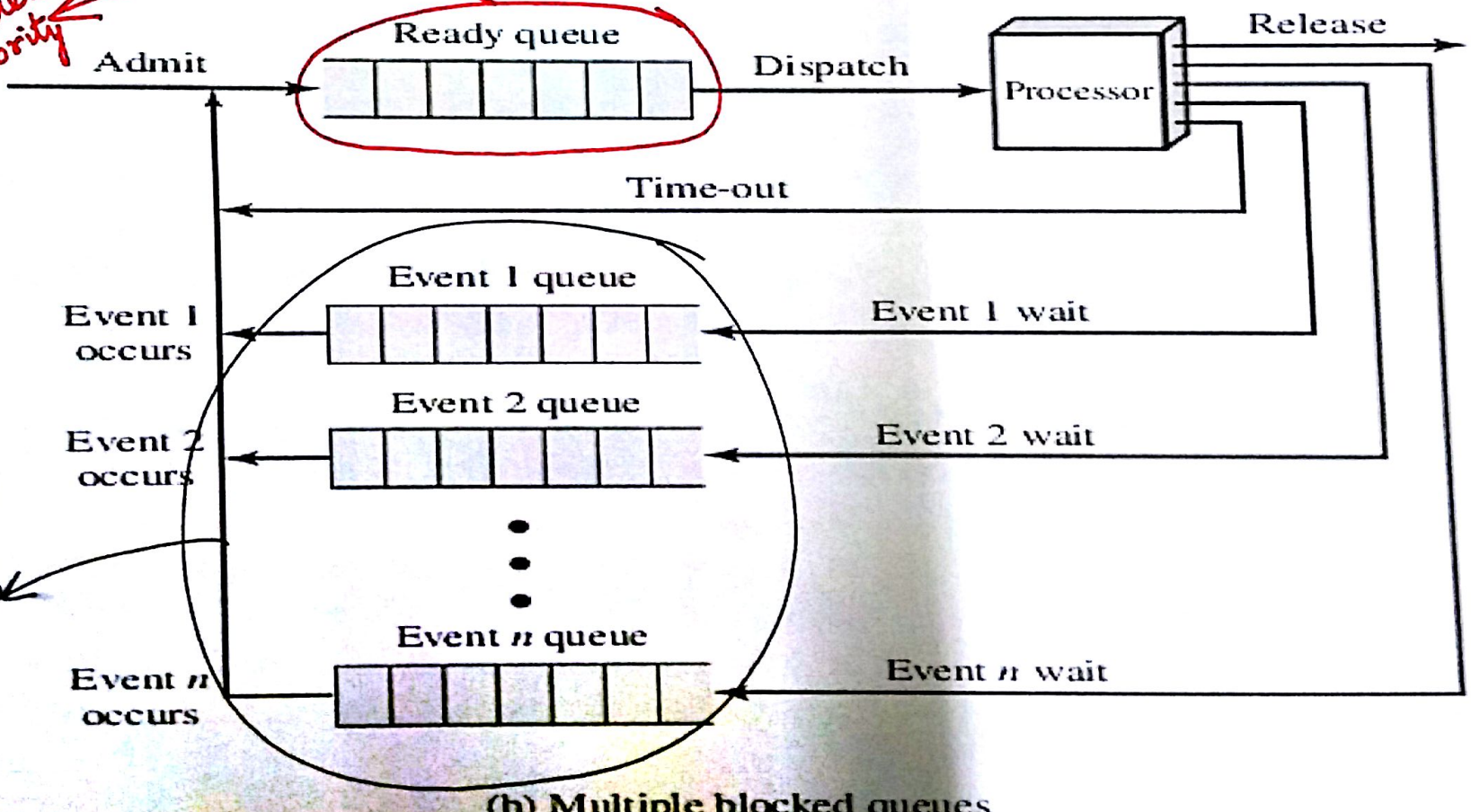
Degree of multiprograming depends upon the main memory size. (Limited).
→ CPU utilization can be improved by increasing degree of multiprograming which is limited due to main memory size. → Can be extended.

OR - Swapping: Blocked Processes swapped out to disk and some other processes is swapped in. (Suspended)



(a) Single blocked queue

Even this can also be managed as individual queues of various priority



Individual queues for each event

(b) Multiple blocked queues

Queueing Model for Figure 3.6

(*) -No process in main memory is in Ready queue. (Ready state) (This doesn't mean that main memory do not hold any process. All are blocked.)

- Some processes in Blocked queue are swaped out onto disk (Suspended queue) (Blocked state) (Suspended state)

- OS may swap in some process from Suspend Queue (ready) Existing process time being not in main mem. (Suspended) or may schedule or swap in new process.

- Swapping is a disk I/O itself (usually better than tape or print I/O)

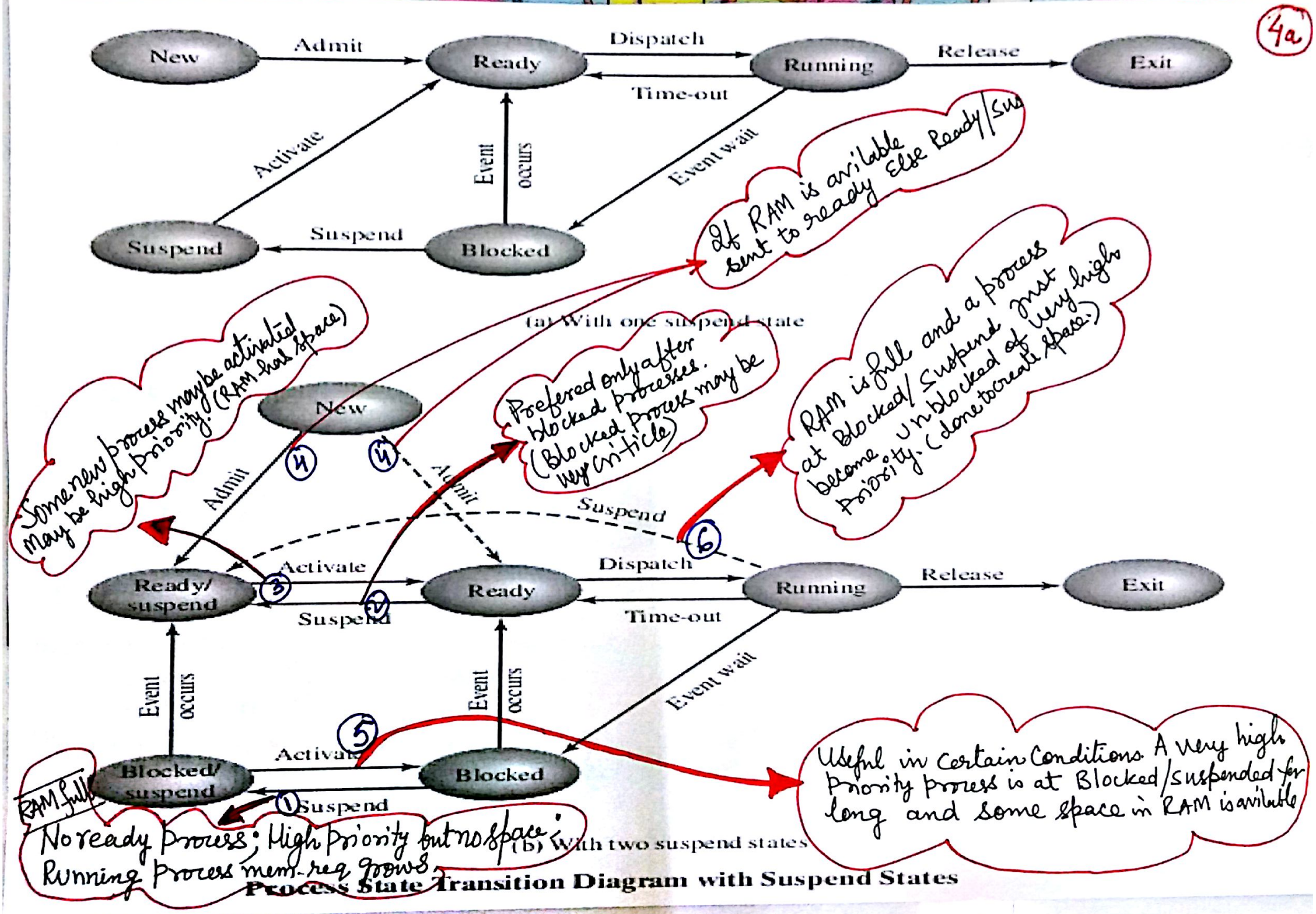
- Swapping in from Suspended queue is useful (Introduction of Suspend state) as the process was blocked (It contains all blocked process) But during this time lapse some may be got ready as their event has occured. (Suspended but Ready)

- Hence in Suspended queue we have two types of processes.

Suspended Queue

Again State Splitting

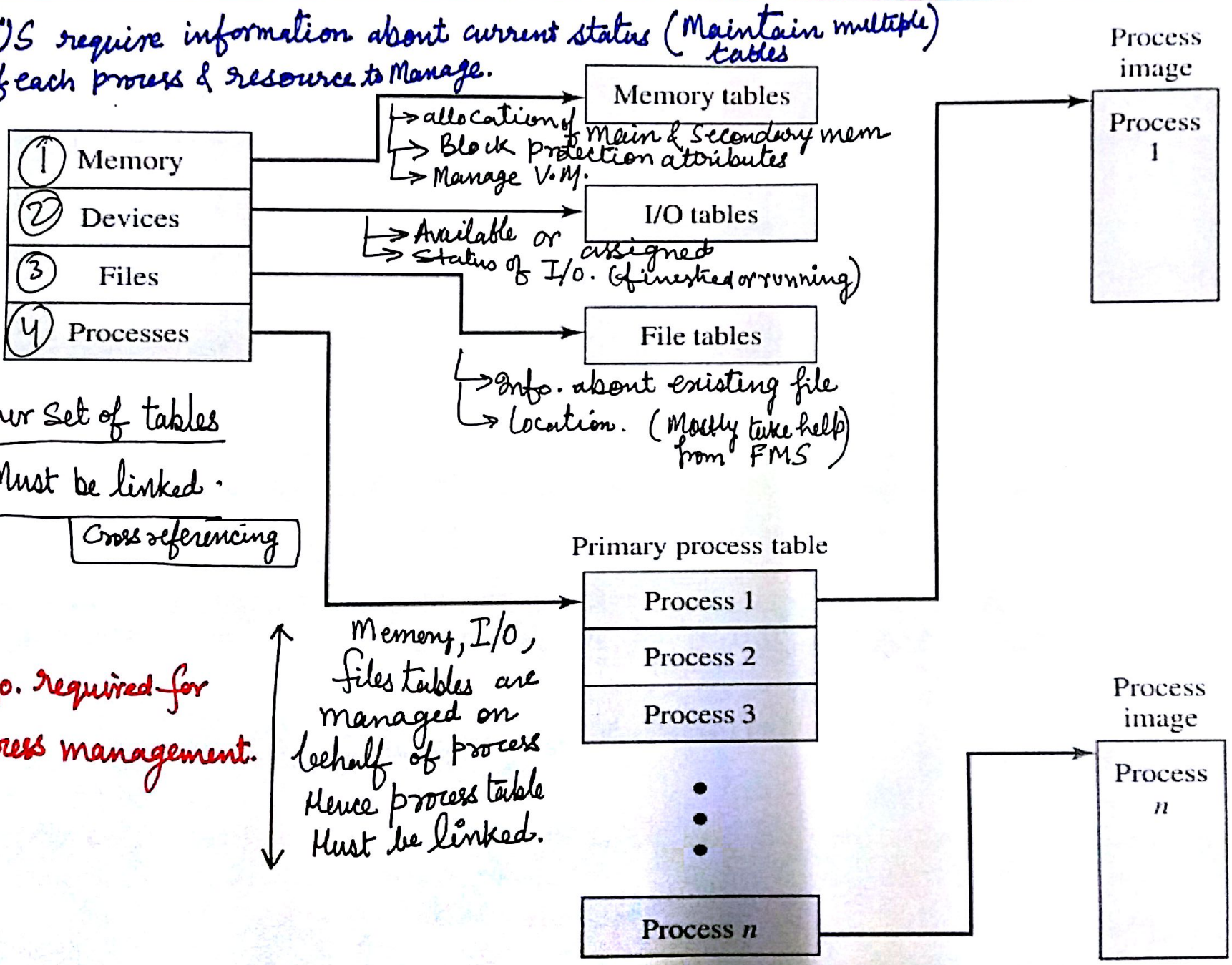
(*) Virtual memory helps —? Generalization of Swapping.



Process State Transition Diagram with Suspend States

4a

OS require information about current status (Maintain multiple tables) of each process & resource to Manage.



General Structure of Operating System Control Tables

(*) Process Control Structure :

(a) Process location : Physical process manifestation.
↳ Program + data + Stack + attributes
PCB
Procedure Calls

OS is required to know the location of any process in Main & secondary Mem. (only some portion is in MM rest in SM)

(b) Process Attributes : 3 group of information is required. a) Process Identifn
b) Process state info
c) Process control info.

① Identification : ⊙ Unique PID for reference and cross referencing. index to process table
⊙ Interprocess Communication ⊙ Parent child management.
⊙ Process is also assigned a user.

② State info : ⊙ Processor reg. Content (PSW) ⊙ Control Status reg PC, Condition code, Status of interrupt.
⊙ Stack Pointers

③ Control info : ⊙ State ⊙ Priority ⊙ Scheduling ⊙ Waiting event
⊙ IPC based flags ⊙ Memory managmt pointer (seg/Page)

5a

Process Identification

Identifiers

Numeric identifiers that may be stored with the process control block include

- Identifier of this process
- Identifier of the process that created this process (parent process)
- User identifier

Processor State Information

User-Visible Registers

A user-visible register is one that may be referenced by means of the machine language that the processor executes while in user mode. Typically, there are from 8 to 32 of these registers, although some RISC implementations have over 100.

Control and Status Registers

These are a variety of processor registers that are employed to control the operation of the processor. These include

- **Program counter:** Contains the address of the next instruction to be fetched
- **Condition codes:** Result of the most recent arithmetic or logical operation (e.g., sign, zero, carry, equal, overflow)
- **Status information:** Includes interrupt enabled/disabled flags, execution mode

Stack Pointers

Each process has one or more last-in-first-out (LIFO) system stacks associated with it. A stack is used to store parameters and calling addresses for procedure and system calls. The stack pointer points to the top of the stack.

Process Control Information

Scheduling and State Information

This is information that is needed by the operating system to perform its scheduling function. Typical items of information:

- **Process state:** Defines the readiness of the process to be scheduled for execution (e.g., running, ready, waiting, halted).
- **Priority:** One or more fields may be used to describe the scheduling priority of the process. In some systems, several values are required (e.g., default, current, highest-allowable).
- **Scheduling-related information:** This will depend on the scheduling algorithm used. Examples are the amount of time that the process has been waiting and the amount of time that the process executed the last time it was running.
- **Event:** Identity of event the process is awaiting before it can be resumed.

Data Structuring

A process may be linked to other process in a queue, ring, or some other structure. For example, all processes in a waiting state for a particular priority level may be linked in a queue. A process may exhibit a parent–child (creator–created) relationship with another process. The process control block may contain pointers to other processes to support these structures.

Interprocess Communication

Various flags, signals, and messages may be associated with communication between two independent processes. Some or all of this information may be maintained in the process control block.

Process Privileges

Processes are granted privileges in terms of the memory that may be accessed and the types of instructions that may be executed. In addition, privileges may apply to the use of system utilities and services.

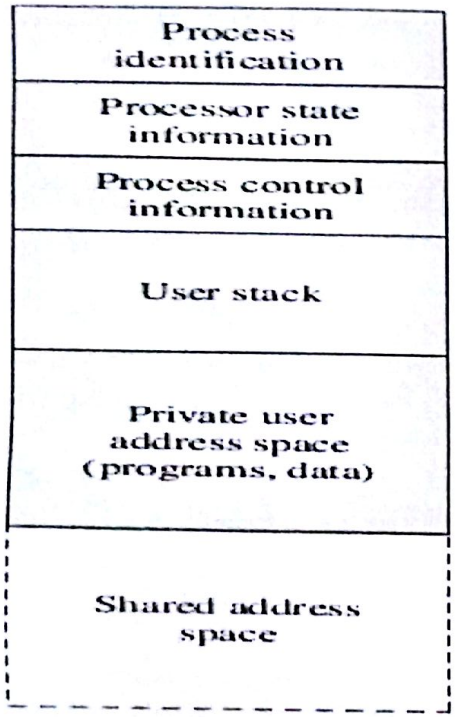
Memory Management

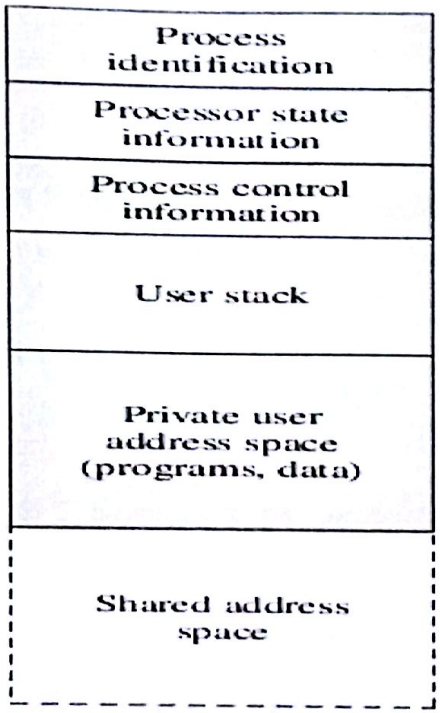
This section may include pointers to segment and/or page tables that describe the virtual memory assigned to this process.

Resource Ownership and Utilization

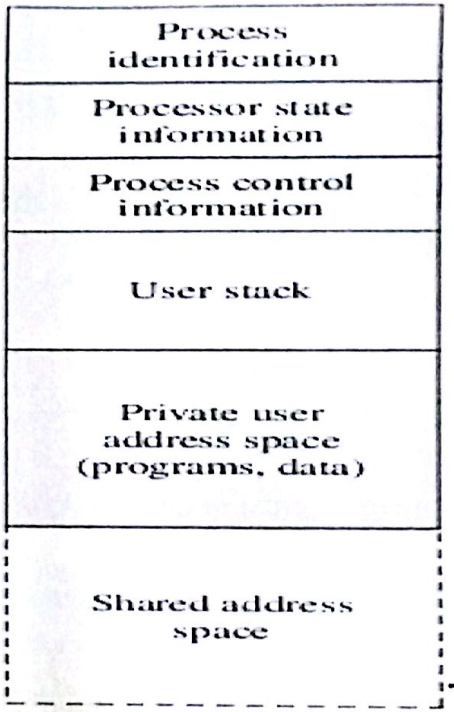
Resources controlled by the process may be indicated, such as opened files. A history of utilization of the processor or other resources may also be included; this information may be needed by the scheduler.

5c

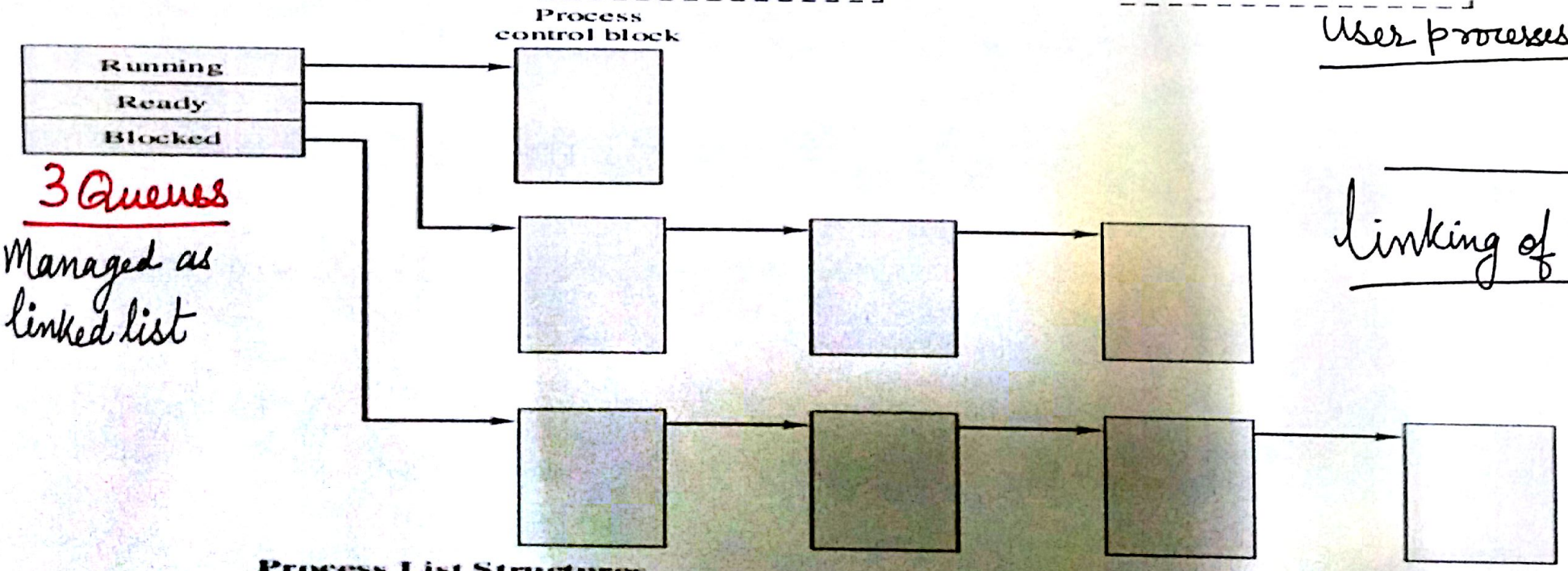




This may not be Contigous depends on MMU. Paged or Segmented ...



User processes in VM



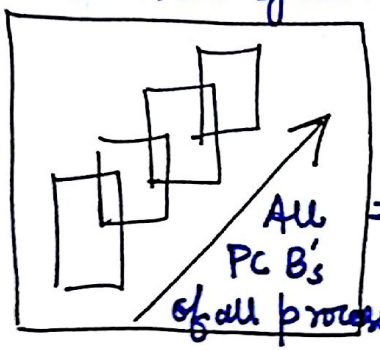
3 Queues

Managed as linked list

linking of PCB

Process List Structures

* PCB is most important data structure for an OS.
They are read and modified by various OS modules like
  - Scheduling
  - resource allocation
  - interrupt processing
  - Performance Monitoring and analysis.



⇒ State of OS

Since everyone is accessing (R/W) PCB directly via PID there is a issue of protection and security.

⇐ Any routine can damage the PCB. (Performance Vs security)

⇐ Hence handler routines are used that performs reading and writing in these blocks

handlers are safe?

(i) Process Control Modes :
- Processor support 2 modes
  - Less privileged (User mode)
  - More Privileged (Kernel mode) (does Process, Memory and I/O Management)

* It is done to protect key operating System tables from user progm.
* In Kernel mode Software has complete control.
* A bit in PSW is set for mode of execution.
* User calls ← OS service or Interrupt is trigger ⇒ Mode is set to Kernel and while returning reset to User mode.

(ii) Process Creation : (a) Assign unique PID to every process.
(b) Allocate Space for the process in (Primary / Secondary).
(c) Initialize the process Control block.
(d) Fix up appropriate linkages. (in various queues)
(e) Create / expand other data structure. (Say a file of all process accounting or performance analysis file)

When and how to switch a running process ⟹ when Switching Mode and State Switching.

OS Can Switch a process only when it has gained its control [Interrupt (I/O), Trap (Error), Supervisor call (System call)]

① Interrupt : Does some basic book keeping and Branches to OS routine.
⊙ Clock interrupt ⊙ I/O interrupt (I/O event occur) (trigger process State switching) ⊙ Memory fault (address not in main memory)

① Trap : Checks and handle error. If fatal ⟶ State = Exit
- may also try to recover and process switching.

⊙ Supervisor Call : Running process request an I/O say file open. ⟶ Transfer Control to OS.

(iii) Mode switching: During instruction cycle processor checks for pending interrupts and waits for them. (Switch process if req)
↳ leads to state switch.
Else → Sets PC to start address of interrupt handler → Switches from user to Kernel mode.
(Process context is saved.)

*) Of interrupt followed by process switch
⇒ OS gets the control.

(iv) State switching: Mode switching and state switching both are different.

(a) Save the content – PC & other Reg.
(b) Update PCB's of current & old blocks.
(c) Link PCB to appropriate queues.
(d) Process selection for execution.
(e) Update its PCB also update various OS related Key files.
(f) Restore new process content and proceed with its execution.