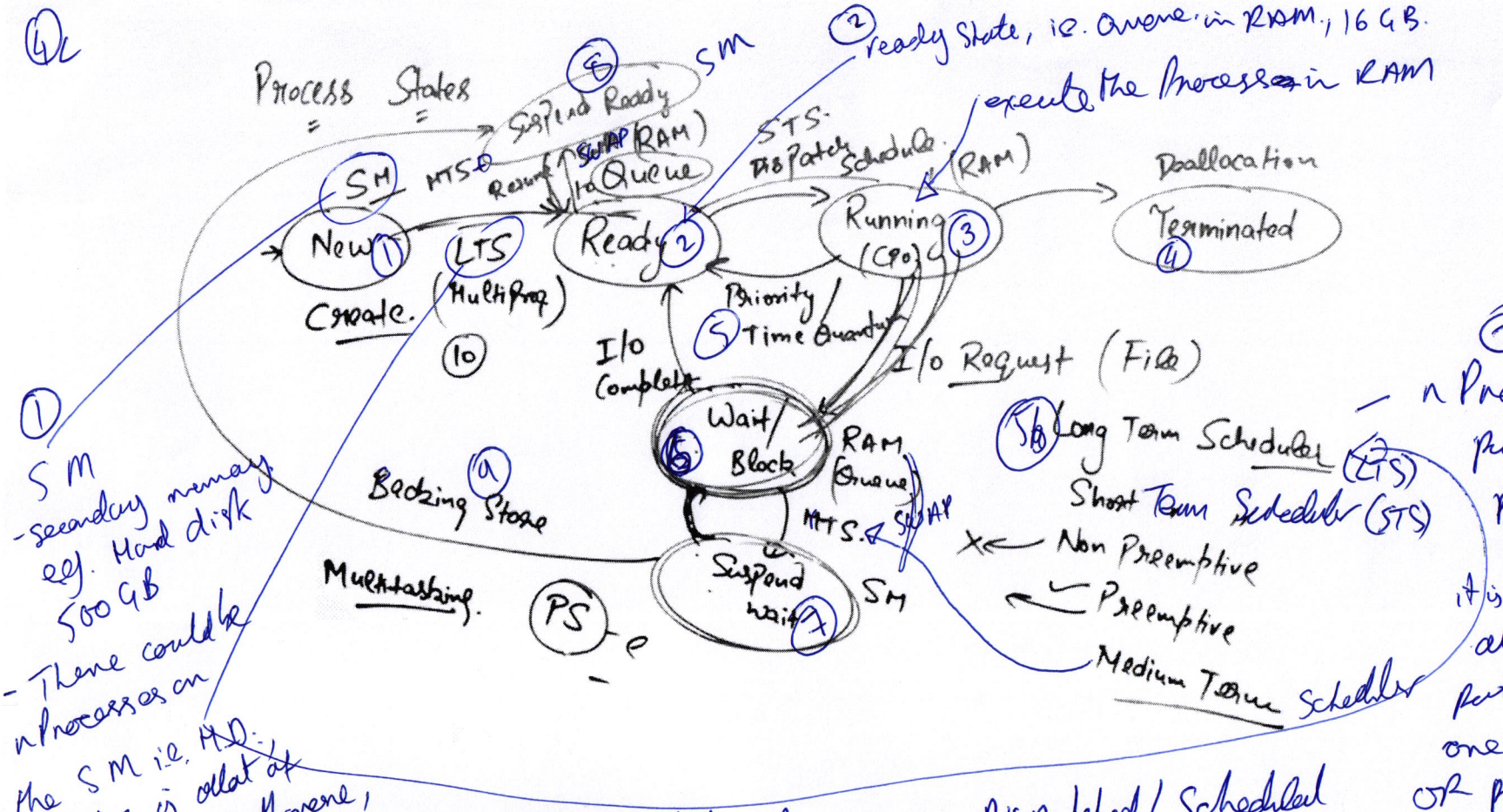


- This is for illustration purposes, our OS doesn't known this Process State diagram.

(1) SM
- secondary memory eg. Hard disk 500 GB
- There could be n Processes on the SM i.e. HDD as there is allot of space available. However, only x Process can be put into RAM, due to memory limitation.

(2) Ready State, ie. queue in RAM, 16 GB.
- n Process, only x are put in Ready State, who puts these in ~~the Scheduler~~ Processes in Ready State? it is Long Term Scheduler, As all n Processes cannot be put into Ready State at one Time, they are scheduled OR protised by LTS.
- The LTS, ~~puts~~ schedules the ~~tasks~~ Process ~~log in~~ ~~the~~ using 2 methods;
  1) Multiprogramming
  2) Multitasking

(3) Running State - Process are Dispatched/ Scheduled to the CPU
- How many Processes u can schedule at One Time, depends on Number of ~~Processes~~ (CPU) available, if only 1 then only 1 Process at one time, however currently you have 8 CPU's on i7 Intel. (multi-processor)

(4) Deallocation of Process from RAM of the Process completed, to make available for other Processes pending in Ready State

4

(5a) High Propenity Process interrupt - this will cause the current process to be passed back to ready queue/state. This is known as multi-tasking.

- High Priority process only 1 task can be completed by CPU, therefore the other process is moved back to RAM (Ready State)

- Time Quantum - 10 sec for each Process, another form of multi-tasking eg. Round Robin

- Who helps in this is STS (Short-term scheduler) which schedules tasks in multi-tasking

- Non-Preemptive → ⊳ arrow not available
- Preemptive

(6) If there is I/O Request for a File from SM (ie. Hard disk), the current process will go into wait state, whilst the CPU will start to process a new Process, Task.

One other reason it is asked to wait as the CPU speed is MIPS whilst SM is slower.

Once I/O request is complete the Process is moved to Ready State, NOT Running state as their is already a process currently running in the CPU.

Upto here all States are defined as Primary States for the Process LifeCycle.

(7) If processes that are in scheduled to the CPU, they require I/O request, and in the wait state is full with pending processes and it cannot store any new processes in the wait state, because of limited memory in RAM.

That process will be SWAPed into SM, ie. into "Suspend Wait State" and there it will carry on with its I/O request. Once complete it will move back to Wait State if room is available.

3e

Multiprogrammed OS — Non Preemptive — ~~but~~ as much processes into RAM.

↳ to avoid IDLENESS

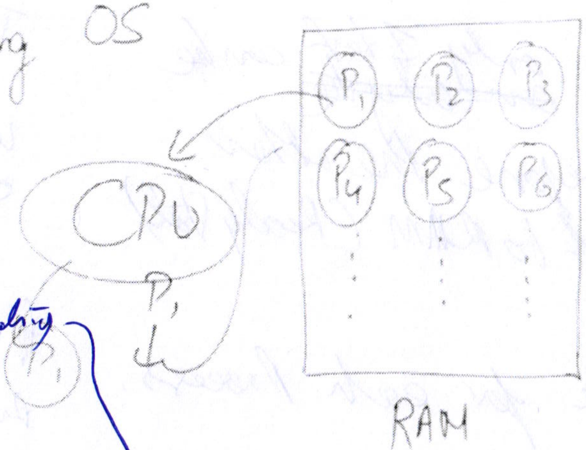
P_1 → CPU — will complete execution before it starts P_2 process.

— further if P_1 asks for an I/O interrupt e.g. keyboard input then in the meanwhile P_2 process will start whilst P_1 I/O is completed

Multitasking / Time Sharing OS

Preemptive / ↓ Responsiveness / also NON-Ideal

Benefit – as u are allocating same time to ~~all~~ all process, so it give response to all Process.



will allocate a certain amount of time for a Process, if the P_1 is completed within that time, Good. Else this process will be re-scheduled again in the future, like a Round Robin Approach. It will process P_2 for the same time, then P_3 or so on., then will come back to P_1

Zumairah

I will thm

6/A $6+4=10$

(7).... Who manages these processes scheduling, it is the MTS (Medium Term Scheduler)

RAM → SM
SWAP out

(8) If High VIP priority Process is scheduled and Ready State is Full. Then a processes is moved from Ready (RAM) → Suspend Ready (SM) State to make Space for the High VIP Process.

Again this is managed by MTS.

(9) Backing Store: if Process in Suspend State is waiting to go to Wait State, but has been persistently full then that process is moved to "Suspend Ready" State so it gets priority to move into Ready State asap., in order to Resume that process.

These are Additional states as in VM. virtual memory.

In Unix u can see the relevant States of the processes using the PS state