A08

PAGE NO. DATE 04/12

↳ An operating sysm is a progm that manages the computer hardware and provides a basis for application progms, it acts as an intermediatery b/w computer hardware and computer user.

→ Goals: ① convinience ② efficiency of task.

→ Objectives: ③ ability to evolve

→ Functions of OS: ① Memory managment
② Device managment
③ Processor managment
④ Security
⑤ Error detection
⑥ Co-ordination b/w the software & user.
⑦ Job Scheduling
⑧ File managment.

→ ① Job scheduling (② CPU scheduling):
If several jobs are ready to be brought to main memory and if there is not enough space in MM then the system must choose among them. This task is called Job scheduling.

→ CPU scheduling: If several jobs are ready to run at the same time the sysm must choose among them.

⇒ User View / System View:

AOS

7-state process model

PAGE No. DATE 8/12/22

→ while creating a process, it changes it states.

5-state process model.

① New state: a process is being created.

② Ready: process is loaded in the main memory and waiting to be assigned to a processor

③ Running: Instructions of the process are being executed.

④ Terminated: process finished its execution.

⑤ Waiting: when a process is waiting for some event or i/p, o/p ~~device~~ to be completed then process is called in waiting state.

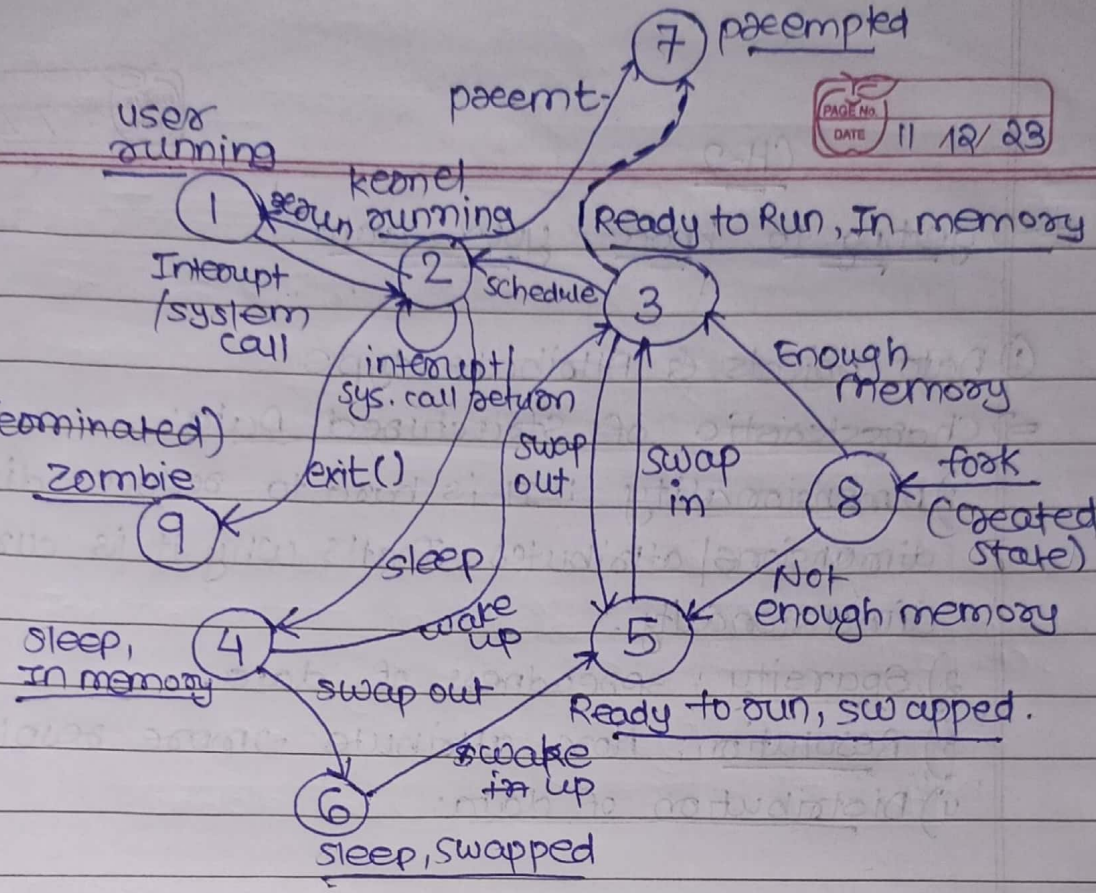
⑥ suspended-wait: when the i/p-o/p device is not available, the process gets suspended and goes to the suspended-wait state. And when i/p o/p device is available, the process gets resume and goes to waiting state.

⑦ suspended-ready: when the enough space is not available in MM, it goes to susP-ready state. And whenever the memory gets availab the process will get resumed and go to the Ready state again.

⇒ Process: It is a program in execution and a program is a piece of code. A process includes process stack and data section.

↓ temp^{s} var, local var^{s}, return values

↓ global var^{s}.



⇒ 9-state system:

① user running: process is running in user mode.

② kernel running: process is " " kernel " .

③ Ready to Run, In memory: process is not executing but ready to run as soon as the kernel schedule it.

④ Sleep, In memory: process is sleeping but it is in MM.

⑤ Ready to run, Swapped: A process is ready to run but the swapper must swap the process into MM before the kernel can schedule it for execution.

⑥ Sleep, Swapped: The process is sleeping and swapper has swapped the process to Secondary Memory. to ~~wai~~ make space for the execution of other processes in MM.

⑦ preempted: Kernel preempts a running process for allocation of another process. While the 1st process is moving from kernel mode to user mode.

⑧ fork: created state: A new process is created and

it is not running.

⑨ Zombie: A process has been executed and exit() system call has been enabled.

PAGE No. DATE 12/12/23

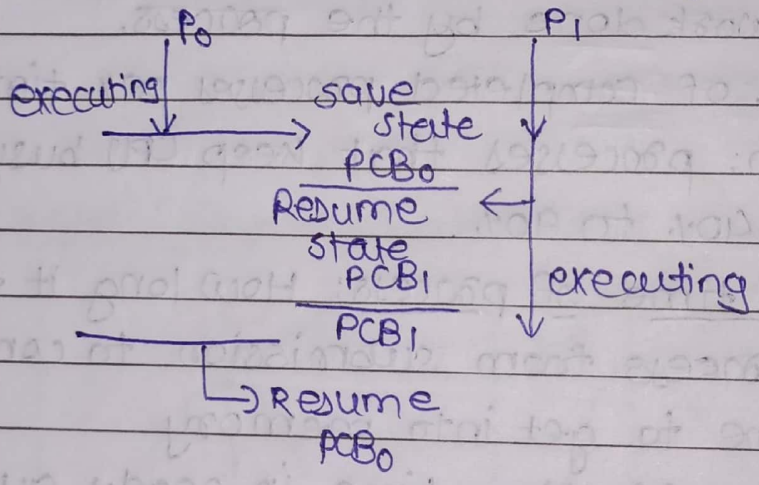
AOS

process control block:

process ID
process state
Prog^m counter(PC)
Registers
memory limits
list of open files
....

⇒ CPU scheduling (priority, Queues, Assignment time).
→ Memory managment (Page tables, Segment tables)
⇒ Accounting information (CPU utilization, time limits, etc.)
→ I/o status information

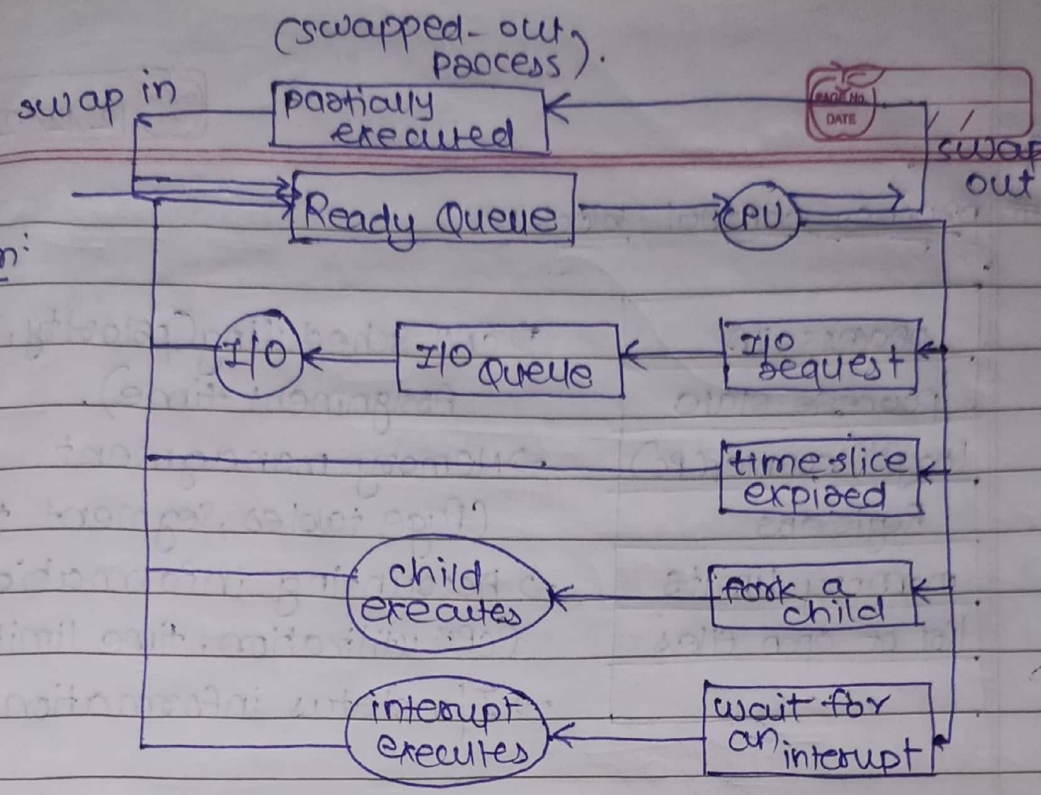
→ switching b/w processes: (diagram).



⇒ Multi programming & time sharing:

→ Job queue: It will contain all the processes in the system.
→ Ready queue: processes which are ready and waiting for the exec^n.
→ Device queue: processes which are waiting for I/o devices.

Queuing diagram:



→ Throughput: work done by the process.
(No. of completed processes per time unit).

→ CPU Utilization: processes that keep CPU busy. It varies from 40% to 90%.

→ Turn around time of process: How long it takes to execute a process from submission to completion.

→ (Waiting time to get into memory
+ Waiting time in ready queue
+ execution time in CPU
+ I/O etc) = Turn-around time.

→ Waiting time: Sum of periods spent in the ready queue and time it takes for the I/O.

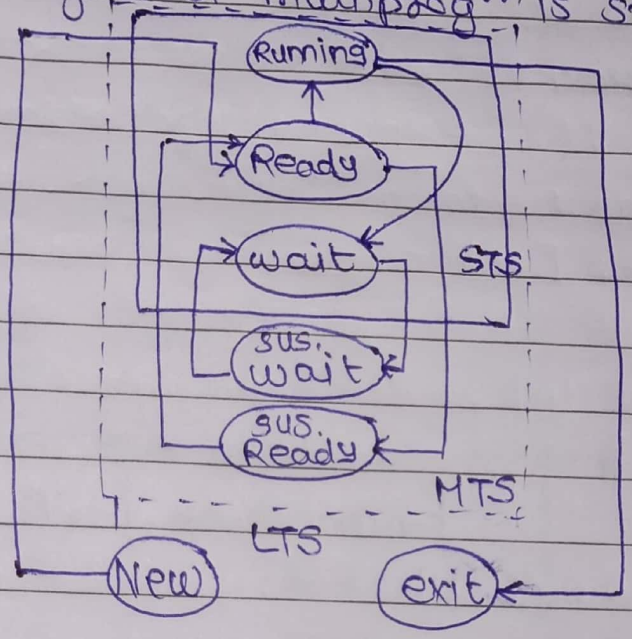
→ Response time: Time from submission of procs to first response produced.

AOS

PAGE No.
DATE 14/12/23

→ Schedulers → long-term → maintains degree of multiprogramming. (control)
↓ decides the processes brought to the ready state.
→ short-term
→ medium-term
most preferred.

→ [Avg. rate of process creation = Avg. departure rate of process leaving the system]
means, degree of multiprogm is stable.



⇒ Levels of Scheduler.

* First Come First Serves: (FCFS) ⇒ (Non-preemptive).

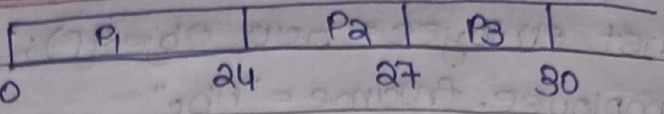
eg.

process	burst time	Arrival time
P_1	24	2
P_2	3	1
P_3	3	0

* ⇒ [AWT (Avg. waiting time) = ~~completion~~ Turn Around time − Burst time]

* ⇒ [ATT (Avg. turnaround time) = completion time − arrival time]

→ Gantt chart:



→ ~~ATAT~~ $_{P_1}$ = ~~24 - 0~~
(~~Avg turn around time~~)
= ~~24 sec~~

without considering arrival time

~~ATAT~~ $_{P_2}$ =

⇒ Waiting time $_{P_1}$ = 0

WT_{P_2} = 24

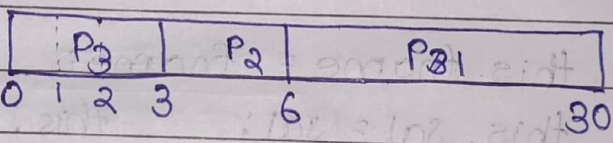
WT_{P_3} = 27

Avg. WT = $\frac{24+27+0}{3}$

= 17

⇒ considering arrival time :-

Gantt chart:



$WT_{P_3} = 0$, ~~$WT_{P_2} = 3$~~, ~~$WT_{P_1} = 6$~~

~~Avg. WT = $\frac{0+3+6}{3}$ = 3.~~

$ATT_{P_3} = 3-0 = 3$

$ATT_{P_2} = 6-1 = 5$

$ATT_{P_1} = 30-2 = 28$.

$WT_{P_3} = 0$, $WT_{P_2} = 3-1 = 2$

$WT_{P_1} = 6-2 = 4$

Avg. WT = $\frac{4+2}{3} = 2$.

→ convoy effect: huge difference b/w avg. waiting time considering and without considering the Arrival time.

→ whenever larger process is executing, shorter process needs to wait, which degrade the CPU ....

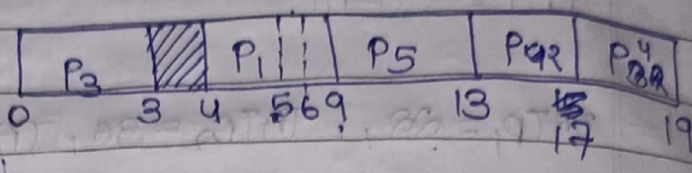
PAGE NO. DATE 15/12/23

AOS

eg.

	A.T.	B.T.
P_1	4	5
P_2	6	4
P_3	0	3
P_4	6	2
P_5	5	4

=> Gantt chart:



Note: -> when two process have same A.T. then give priority to lower process id

=> $ATT_{P_1} = 9-4 = 5$ $\quad ATT_{P_4} = 19-6 = 13$ $\quad$ Avg. ATT. $= \frac{40}{5} = 8$

$ATT_{P_2} = 17-6 = 11$ $\quad ATT_{P_5} = 13-5 = 8$

$ATT_{P_3} = 3-0 = 3$

-> $WT_{P_3} = 0$ $\quad WT_{P_2} = 13-6 = 7$

$WT_{P_1} = 0$ $\quad WT_{P_4} = 17-6 = 11$

$WT_{P_5} = 9-5 = 4$

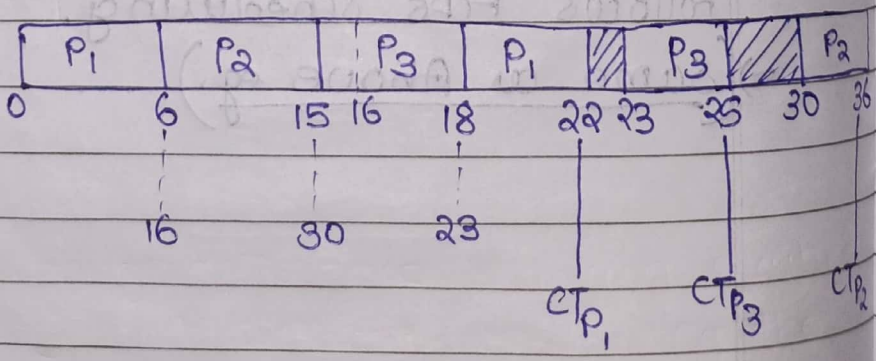
$AWT = \frac{11+7+4}{5} \cong [4]$ $\quad (RT = ST - AT)$

start time ↑ response time ↑

eg.

	AT.	CPU_BT.	IO_BT.	CPU_BT	ST	CT	TAT	WT	RT
P_1	0	6	10	4	0	22	22	2	0
P_2	0	9	15	6	6	36	36	6	6
P_3	0	3	5	2	15	25	25	15	15

=> Gantt chart:



→ WT = TAT − CPU_BT − IO_BT

TAT = CT − AT

$TAT_{P_1} = 29$, $TAT_{P_2} = 36$, $TAT_{P_3} = 25$

[ATAT = 27.667]

[AWT = 7.667]

* ⇒ CPU Utilization = $\frac{\text{total time} - \text{idle time}}{\text{total time}}$

$= \frac{36-6}{36} = \frac{30}{36} = 5/6 \times 100$

$= 83.33\%$

* ⇒ throug-put = $\frac{\text{process no.}}{\text{Max(CT)} - \text{Min(AT)}}$ (total time)

$= \frac{3}{36} = 8.33\% = 0.0833$

eg. 3 processes P_1, P_2, P_3 with process time (BT) 20, 30, 10 resp^{c}. Each process uses the first 30% of its process time in CPU, then 50% in IO and last 20% in CPU. Find AWT, ATAT, RT if the system follows FCFS scheduling.

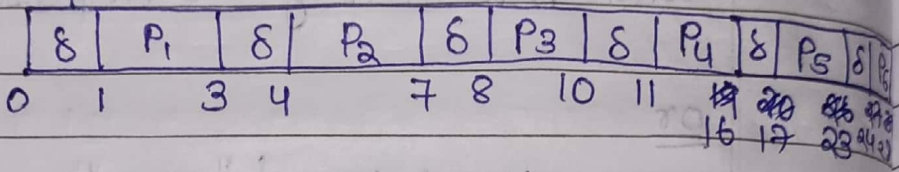
⇒ (Same as Above eg.)

eg.

	AT.	BT.
P_1	0	2
P_2	1	3
P_3	2	2
P_4	3	5
P_5	4	6
P_6	5	4

FCFS
=) with 1 unit CPU overhead time. whenever process is switched.

→ Gantt chart:



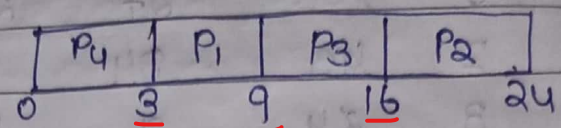
SJF

PAGE No. DATE 19/12/23

AOS

eg. =>

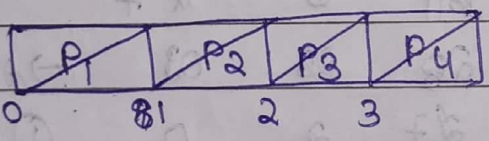
	BT	WT	TAT (non preemptive)
P_1	6	3	6
P_2	8	16	
P_3	7	9	
P_4	3	0	

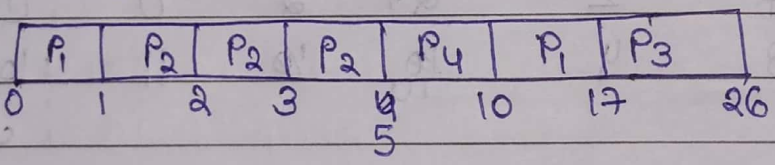


$$AWT = \frac{3+16+9}{4} = 7$$

eg * preemptive SJF: (SRTF - Shortest Remaining time First).

	AT	BT	CT	TAT	WT
P_1	0	8-1=7	17	17	9
P_2	1	4-1=3	5	4	0
P_3	2	9	26	24	15
P_4	3	5	10	7	2





WT = TAT − Total time of execution ~~AT~~.

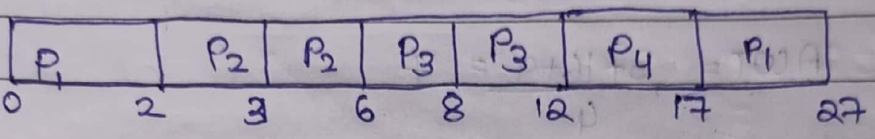
TAT = CT − AT

$$AWT = \frac{9+15+2}{4} = \frac{26}{4} = 6.5$$

$$ATAT = \frac{21+24+7}{4} = \frac{52}{4} = 13.$$

eg.

SJF	AT	BT	TAT	CT	WT
P_1	0	10	27	27	15
P_2	2	3	4	6	2
P_3	3	4	9	12	3
P_4	8	5	9	17	4



$$AWT = \frac{15+3+4}{4} = \frac{22}{4} = 5.5$$

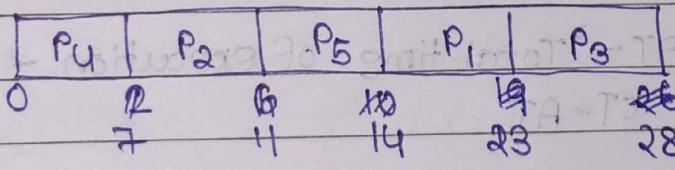
$$ATAT = \frac{27+4+18}{4} = \frac{49}{4} = 12.75$$

Priority Scheduling: (lesser the no. → higher the priority) ⇒ if 2 proc^s have same priority then FCFS.

eg.

	B.T.	Priority	CT	TAT	WT
P_1	9	5	23	19	14
P_2	4	3	11	6	2
P_3	5	7	28	23	18
P_4	7	2	7	7	0
P_5	3	4	14	11	8

→ disadvantages:
① process with lower priority will starve.



$$AWT = \frac{14+7+23+11}{5} = \frac{55}{5} = 11$$

$$ATAT = \frac{23+11+7+14+28}{5} = \frac{83}{5} = 16.6$$

AOS

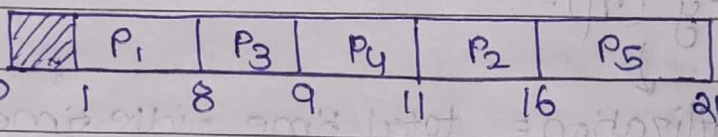
18/12/23

* Shortest Job First (SJF): [used in batch system when used for Benchmarking. LTS is used.]
→ Criteria: Burst Time(BT)
→ Non-preemption mode.

eg.

ID	AT	BT	CT	TAT	WT
P_1	1	7	8	7	0
P_2	2	5	16	14	9
P_3	3	1	9	6	5
P_4	4	2	11	7	5
P_5	5	8	24	19	11

⇒



At 8th second, there will be all processes arrived from them, process with less/smallest BT. will be executed first.

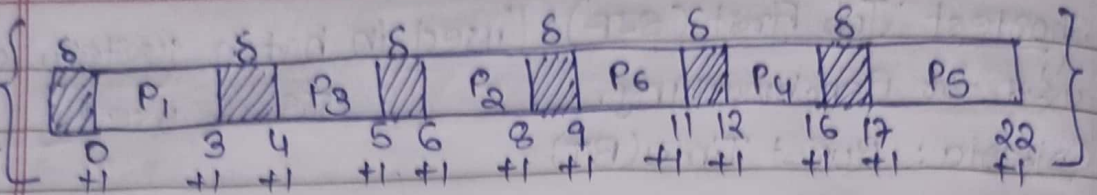
ATAT = 10.6, AWT = 6.

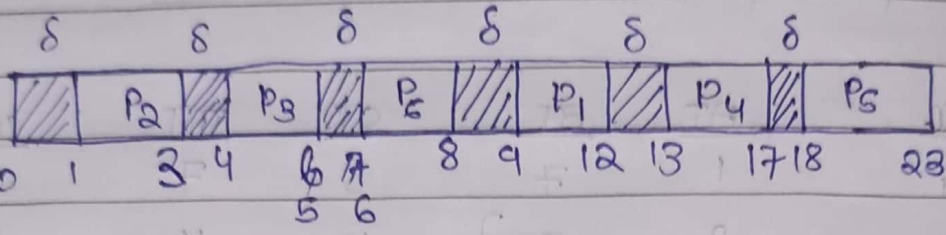
⇒ Advantages: ① Max^m throughput. ② Min^m W.T.
⇒ disadvantages: ① computing B.T. is nearly Impossible.
② Longest Jobs will starve.

eg.

ID	AT	BT	CT	TAT	WT
P_1	0	3	12	12	9
P_2	1	2	3	2	0
P_3	2	1	5	3	2
P_4	3	4	17	14	10
P_5	4	5	23	19	14
P_6	5	2	8	3	1

with 1 sec overhead of CPU.





AWT = 6

ATAT = 8.83

efficiency (η) =

↓

CPU utilization = $\frac{\text{total time} - \text{idle time}}{\text{total time}}$

$= \frac{23-6}{23} = \frac{17}{23} \times 100$

$= [73.9\ \%]$

throughput = $\frac{\text{total process no.}}{\max(CT) - \min(AT)} = \frac{6}{23-0} = \frac{6}{23}$

$\left[\approx \frac{\cancel{23}}{26}\ \%\right]$

* ⇒ Static Methods: (to calculate BT):
  - → based on Process type
  - → based on Process size.

⇒ Dynamic Methods:

→ based on past experience with the process and predict the future of the process.

① simple averaging ② exponential averaging.

→ simple averaging: It is avg. of all the processes executed till now. and based on that we'll predict.

eg. 100 ms, 250 ms, 200 ms, 10 ms.

$$avg = \frac{100+250+200+10}{4} = 140.\ ms.$$

$$\left[T_{n+1} = \frac{1}{n}\sum_{i=1}^{n} t_i\right]$$

→ exponential averaging:

$$*\left[T_{n+1} = \alpha t_n + (1-\alpha) T_n\right]^*$$

; T_{n+1} = predicted BT. for P_{n+1} process.

t_n = actual BT. of P_n process.

T_n = predicted BT. of P_n process.

α = smoothning factor.

Q. Calculate the predicted BT. using exponential avg^{ng}. for 5th process, if the predicted BT. for 1st process is 10 and actual BT. of first four processes are 4, 8, 6, 7 (t_1 t_2 t_3 t_4) resp^{c}. ($\alpha = 0.5$)

=)

$$T_2 = 1/2(4) + 1/2(10) = 7$$

$$T_3 = \frac{1}{2}(8) + 1/2(7) = 7.5$$

$$T_4 = \frac{1}{2}(6) + \frac{1}{2}(7.5) = 3 + 3.75 = 6.75$$

$$T_5 = \frac{1}{2}(7) + \frac{1}{2}(6.75) = 3.5 + 3.375$$

$$= [6.875]$$ Ans.