UNIT- 3

* Clock Driven Scheduling :-

It is a periodic task scheduling method, primarily used for hard real-time systems, where all properties of all jobs are known at design time; such that offline scheduling techniques can be used.

When scheduling is clock driven decisions about what jobs executes when, are made at specific time instants. These instants are chosen before the system begins execution.

Two important clock driven schedulers are :

1) Table- Driven
2) Cyclic Schedulers

* General Structure of Cyclic Schedule :-

It describes the frames size, major cycles and its constraints.

1) Frame & Major Cycles :-

The scheduling decision time partition the time line into intervals called frames. Every frame has length f, where f is the frame size.

Every scheduling decisions are made only at the beginning of frame, there is no preemption within each frame. The first job of every task is released at the beginning of some frame.

2) Frame Size Constraints :-

These constraints help to decide the frame size.

(i) Minimum Context Switching:

We want the frame to be sufficiently long so that every job can start and complete its execution within a frame. In this way, no job will be preempted.

$$f \geq \max(e_i)$$

(ii) Frame Size:

To keep the length of the cyclic schedule as short as possible, the frame size f should be chosen so that it divides H (length of hyper period). This condition met, when f divides the P_i of atleast one task T_i.

$$\left\lfloor \frac{P_i}{f} \right\rfloor - \frac{P_i}{f} = 0 \qquad \frac{P}{f} = \text{integer}$$

To make it possible for the scheduler to determine whether every job completes by its deadline, the frame size should be sufficiently small and there is atleast one frame between the release time and deadline of every job.

$$2f - GCD(P_i, f) \leq D_i$$

(iii) Job Slices:

Sometimes, the given parameters of some task cannot meet all three frame size constraints simultaneously. In this situation, partition the task that has a large execution time into slices with smaller execution times.

EX A cyclic scheduler is to be used to run the following set of periodic tasks on a uniprocessor, find out the frame size.

$T_1 = (1, 4)$; $T_3 = (1, 20)$ $\quad T_i = (e_i, P_i)$

$T_2 = (1, 5)$; $T_4 = (2, 20)$ $\quad$ & $d_i = P_i$

Soln Constraint 1: $F \geq \max(e_i)$

$F \geq 2$

Constraint 2: $H = LCM(4, 5, 20) = 20$

F can be 2, 4, 5, 10, 20

$\frac{P_i}{f}$ = integer for atleast one

Constraint 3: $2f - GCD(P_i, f) \leq D_i$

For $f = 2$

$T_1 \Rightarrow 2 \times 2 - GCD(4, 2) \leq 4$

$4 - 2 = 2 \leq 4$ True

$T_2 \Rightarrow 2 \times 2 - GCD(5, 2) = 3 \leq 5$ True

$T3 \Rightarrow 2 \times 2 - GCD(20, 2) = 2 \leq 20$ True

$T4 \Rightarrow 2 \times 2 - GCD(20, 2) = 2 \leq 20$ True

True for all tasks

So frame size (f) = 2 Ans.

EX Consider the following set of periodic real-time tasks to be scheduled by a cyclic scheduler. Determine the frame size for the task set.

$T_1 = (1, 4)$

$T_2 = (2, 5, 7)$

$T_3 = (5, 20)$

$T_i = (C_i, P_i, d_i)$

& $d_i = P_i$ for T_1 & T_3

Soln

Constraint 1: $f \geq \max(C_i)$

$f \geq 5$

Constraint 2: $\frac{P_i}{f}$ = integer for atleast one task

f can be 5, 10, 20.

Constraint 3:

$$2f - GCD(P_i, f) \leq D_i$$

for $f = 5$

$T_1 \Rightarrow 2 \times 5 - GCD(4, 5) \leq 4$

$10 - 1 = 9 \leq 4$ false

So we partition the task which has higher execution time.

So split T_3 into two or three tasks.

$T_{3,1} = (3, 20)$

$T_{3,2} = (2, 20)$

① $f \geq 3$

② f can be 4, 5, 10, 20

③ for $f = 4$

$T_1 \Rightarrow 8 - 4 = 4 \leq 4$ True

$T_2 \Rightarrow 8 - 1 = 7 \leq 7$ True

$T_{3,1} \Rightarrow 8 - 4 = 4 \leq 4$ True

$T_{3,2} \Rightarrow 8 - 4 = 4 \leq 4$ True

So frame size (f) = 4 Ans.

* Fixed-Priority Scheduling Algorithm :-

It assigns the same priority to all the jobs in each task. Rate monotonic algorithm and deadline monotonic algorithm are based on fixed priority scheduling algorithm.

1) Rate-Monotonic Algorithm (RMA) :-

In this algorithm priorities are assigned to task according to their periods. The task which has shorter period has higher priority.

Period ↓ ⇒ Priority ↑

EX $T_i = (P_i, e_i)$

$T_1 = (4, 1)$; $T2 = (5, 2)$ & $T3 = (20, 5)$

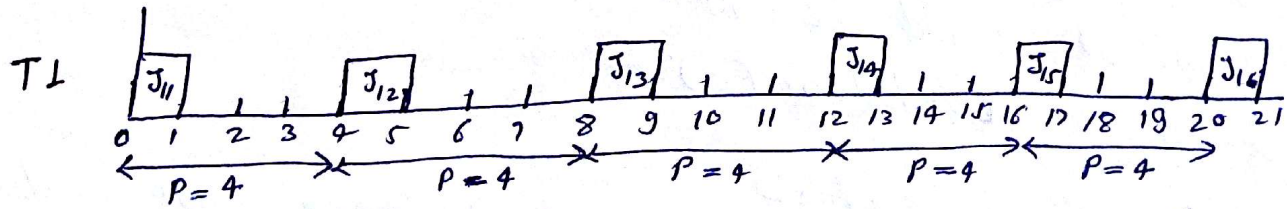
Schedule all tasks on uni-processor using RMA.

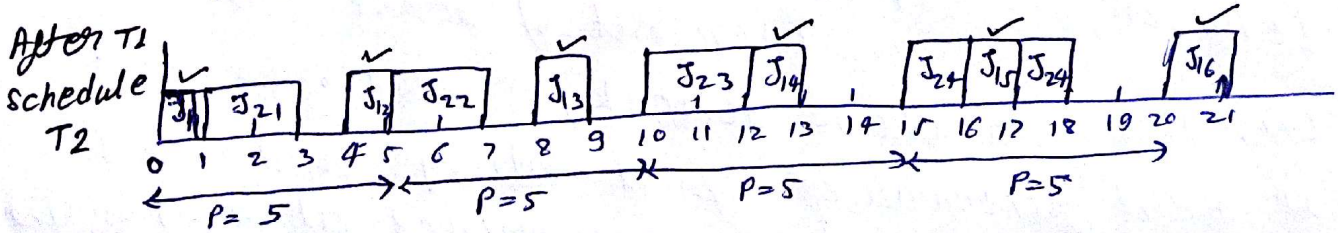
Sol^n

$\because$ Period ↓ ⇒ Priority ↑

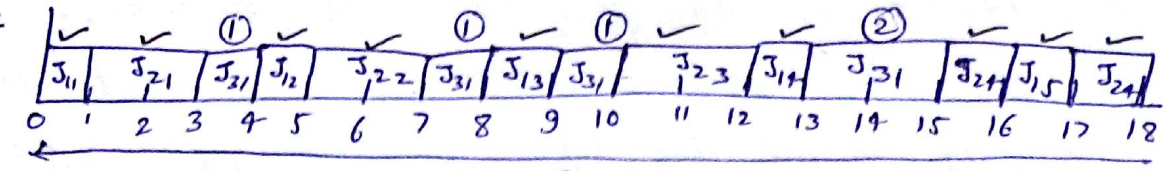
$\therefore$ $T_1 > T_2 > T_3$ (Priority order)

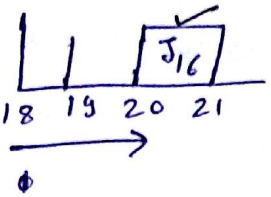
Task T_1 has highest priority ; so it will schedule first and it can preempt all other task. After T_1, task T_2 and T_3 schedule according to their priority. All tasks release at time 0.





After T1 & T2 schedule T3





* Schedulability Test for RMA :-

1) Necessary Condition :-

A set of periodic real time tasks would not be RMA schedulable unless they satisfy the necessary condition. This condition is necessary but not sufficient.

$$\boxed{\sum_{i=1}^{n} \frac{C_i}{P_i} \leq 1}$$, where $\frac{C_i}{P_i} = U_i$ (utilization)

$C_i \rightarrow$ execution time, $P_i \rightarrow$ period, $n \rightarrow$ no. of tasks.

2) Sufficient condition :-

a) Liu & Layland :- It was obtained by Liu and Layland in 1973.

$$\sum_{i=1}^{n} U_i \leq n\left(2^{1/n} - 1\right)$$

If set of task satisfies this sufficient condition, then it would be RMA schedulable and not required to check condition (b).

b) Lehoczky's Test :- when set of tasks fails condition (a), then we check lehoczky's test. So task set is RMA schedulable if it satisfy condition 1 and in condition 2 either (a) or (b).

$$c_i + \sum_{k=1}^{j-1} \left\lceil \frac{d_j}{d_k} \right\rceil \times c_k \leq d_i$$

Here $k = 1$ to $i-1$ are those task which has higher precedence than i.

EX $T = (c, p)$

$T_1 = (20, 100)$; $T_2 = (30, 150)$; $T_3 = (60, 200)$

Check it satisfy the schedulability of RMA.

Soln 1) $\sum_{j=1}^{n} u_j \leq 1$

$$\Sigma u_i = \frac{20}{100} + \frac{30}{150} + \frac{60}{200}$$

$$\Sigma u_i = 0.7 \leq 1$$

So 1^{st} condition satisfy.

2) (a) $\sum_{j=1}^{n} u_j \leq n\left(2^{1/n} - 1\right)$

$$0.7 \leq 3\left(2^{1/3} - 1\right)$$

$$0.7 \leq 0.78$$

II^{nd} condition satisfy; so don't require to check condition 2 (b).

So all these tasks are schedulable under RMA.

Note If d_i is not given, then by default $d_i = P_i$

EX $T_1 = (20, 100)$

$T_2 = (30, 150)$

$T_3 = (90, 200)$

$T_i = (c_i, P_i)$ & $d_i = P_i$

check it satisfy the schedulability of RMA.

Sol^n 1) $\sum_{j=1}^{n} u_j \leq 1$

$$\sum_{j=1}^{n} u_j = \frac{20}{100} + \frac{30}{150} + \frac{90}{200}$$

$$\sum_{j=1}^{n} u_j = 0.85 \leq 1 \quad (I^{st} \text{ condition satisfy}).$$

2) (a) $\sum_{j=1}^{n} u_j \leq n\left(2^{1/n} - 1\right)$

$$0.85 \leq 3\left(2^{1/3} - 1\right)$$

$$0.85 \not\leq 0.78 \quad (\text{Not satisfy})$$

So check condition 2 (b).

(b) $c_j + \sum_{k=1}^{j-1}\left\{\left[\frac{d_j}{d_k}\right] \times c_k\right\} \leq d_j$

For $T1$ $\Rightarrow$ $j=1$ $\quad c_1 \leq d_1$

$$20 \leq 100$$

For $T2$ $\Rightarrow$ $j=2$ $\quad c_2 + \left(\frac{d_2}{d_1} \times c_1\right) \leq d_2$

$$30 + 30 \leq 150$$

$$60 \leq 150$$

For $T3$ $\Rightarrow$ $j=3$ $\quad c_3 + \left(\frac{d_3}{d_2} \times c_2 + \frac{d_3}{d_1} \times c_1\right) \leq d_3$

$$90 + (40 + 40) \leq 200$$

$$170 \leq 200$$

condition 2 (b) satisfy.

So all these tasks are ~~satisfy~~ schedulable under RMA.

2) Deadline - Monotonic Algorithm (DMA) :-

It is also a fixed-Priority algorithm. In this algorithm priorities are assigned according to their relative deadlines. Shorter the relative deadline, task has higher Priority.

Relative Deadline ↓ ⇒ Priority ↑

EX $T_i = (\phi_i, P_i, e_i, d_i)$

$T_1 = (50, 50, 25, 100)$

$T_2 = (0, 62.5, 10, 20)$

$T_3 = (0, 125, 25, 50)$

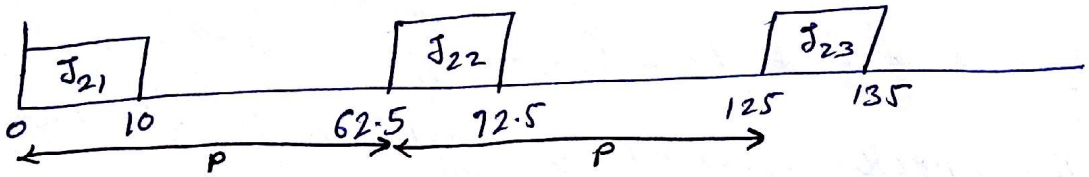
Schedule tasks on uniprocessor using DMA.

Sol" Relative deadline ↓ ⇒ Priority ↑

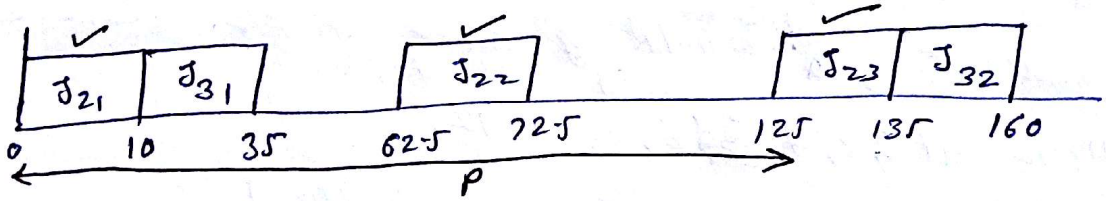
So $T2 > T3 > T1$ (Priority Order)

First schedule task T2 then T3 and T1.. Here higher priority task can preempt lower priority task.

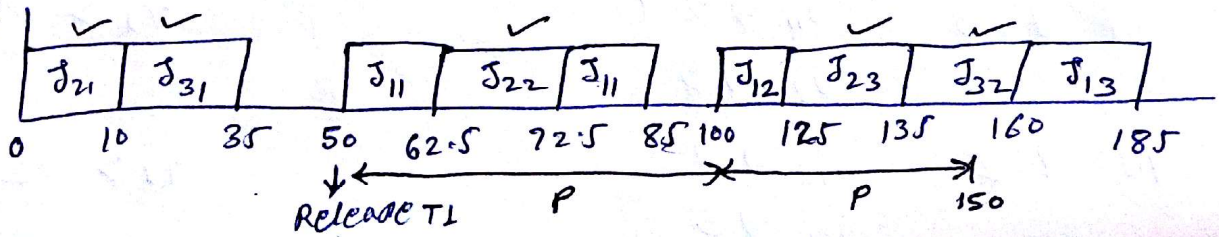
Schedule T2



Schedule T3



Schedule T1



✳ Dynamic Priority Scheduling Algorithm :-

It assigns different priority to the individual job in each task. The priority of the task with respect to other tasks changes as job are released and completed. Dynamic scheduling algorithm are -

1) Earlist Deadline First (EDF)
2) Least Slack Time (LST)

1) Earlist Deadline First (EDF) :-

In this algorithm priorities are assigned according to their absolute deadline. A job with lower absolute deadline has higher priority.

Absolute Deadline ↓ ⇒ Priority ↑

where Absolute Deadline = Release Time + Relative Deadline

In periodic task, a job release after each period.

EX $T_i = (P_i, e_i)$ & $d_i = P_i$

$T_1 = (2, 0.9)$ & $T_2 = (5, 2.3)$

Soln We check priority of each task at the position multiple of ~~periods of each tasks. Cross all tasks~~ period of each task.

Here at 0, 2, 4, 5, 6, 8, 10, 12, -----

At $t = 0$ $J_{11} \Rightarrow 2$ (Absolute deadline) $T_1 > T_2$

$J_{21} \Rightarrow 5$

At $t = 2$ $J_{12} \Rightarrow 4$ $T_1 > T_2$

$J_{21} \Rightarrow 5$

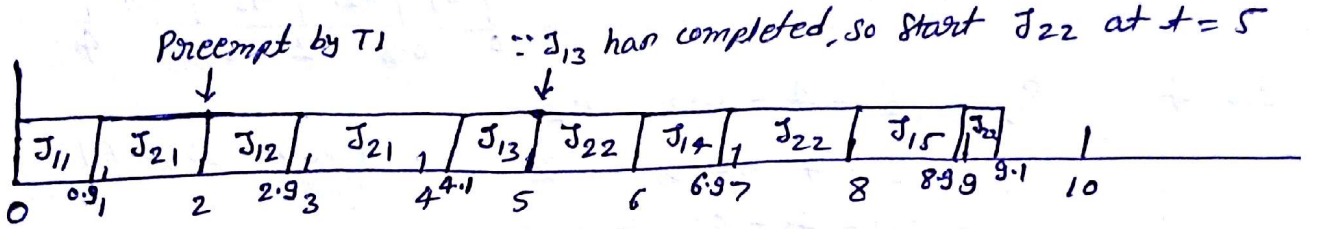
At $t=4$ — $J_{13} = 6$, $J_{21} = 5$ — $T2 > T1$

~~At $t=5$~~ — ~~$J_{13} = 6$~~, ~~$J_{22} = 10$~~ — ~~$T1 > T2$~~

At $t=5$ — $J_{13} = 6$, $J_{22} = 10$ — $T1 > T2$

At $t=6$ — $J_{14} = 8$, $J_{22} = 10$ — $T1 > T2$

At $t=8$ — $J_{15} = 10$, $J_{22} = 10$ — $T1 = T2$ (Any Job can schedule)



2) Least Slack Time (LST) :-

In this algorithm priorities are assign according to slack time. At time t, the slack time of each task job is —

At time $t \Rightarrow \boxed{S_j = d_j - t_0 - x_j}$

where d_j is the absolute deadline of job task i
x_j is the remaining execution time of job task i

Task which has lower slack time, assign higher priority.

$$\boxed{\text{Slack Time} \downarrow \Rightarrow \text{Priority} \uparrow}$$

EX $T_j = (P_j, e_j)$ & $d_i = P_i$

$T_1 = (2, 0.8)$.

$T_2 = (5, 1.5)$

$T_3 = (5.1, 1.5)$

Schedule task using LST.

Check Priority at $t = 0, 2, 4, 5, 6, 8, 10, 10.2, \ldots$

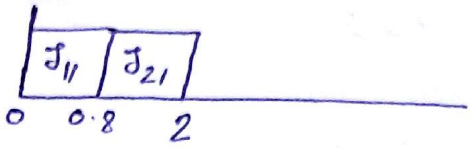
At $t = 0$

$S_1 = 2 - 0 - 0.8 = 1.2$

$S_2 = 5 - 0 - 1.5 = 3.5$

$S_3 = 5.1 - 0 - 1.5 = 3.6$

$T1 > T2 > T3$



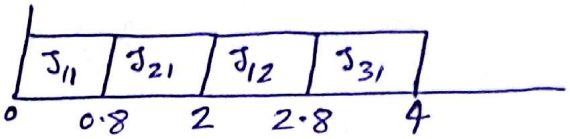
At $t = 2$

$S_1 = 4 - 2 - 0.8 = 1.2$

$S_2 = 5 - 2 - 0.3 = 2.7$

$S_3 = 5.1 - 2 - 1.5 = 1.6$

$T1 > T3 > T2$



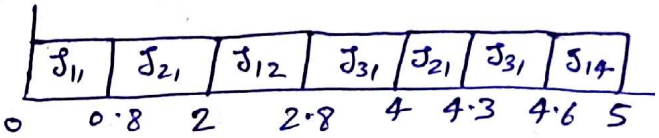
At $t = 4$

$S_1 = 6 - 4 - 0.8 = 1.2$

$S_2 = 5 - 4 - 0.3 = 0.7$

$S_3 = 5.1 - 4 - 0.3 = 0.8$

$T2 > T3 > T1$



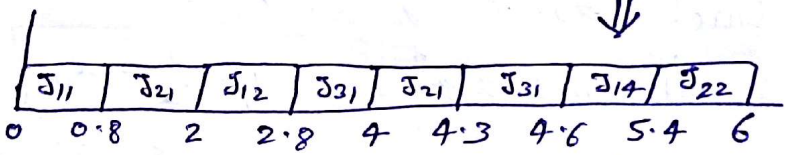
At $t = 5$

$S_1 = 6 - 5 - 0.4 = 0.6$

$S_2 = 10 - 5 - 1.5 = 3.5$

$S_3 = 5.1 - 5 - 0 =$ Task completed (0 execution time)

$T1 > T2$



At $t = 5.1$

$S_1 = 6 - 5.1 - 0.3 = 0.6$

$S_2 = 10 - 5.1 - 1.5 = 3.4$

$S_3 = 10.2 - 5.1 - 1.5 = 3.6$

$T1 > T2 > T3$

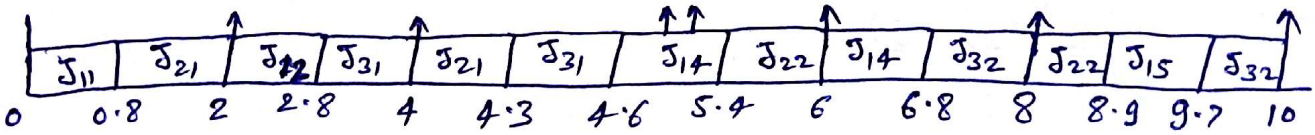
At $t = 6$

$S_1 = 8 - 6 - 0.8 = 1.2$

$S_2 = 10 - 6 - 0.9 = 3.1$

$S_3 = 10.2 - 6 - 1.5 = 2.7$

$T1 > T3 > T2$



At $t = 8$

$S_1 = 10 - 8 - 0.8 = 1.2$

$S_2 = 10 - 8 - 0.9 = 1.1$

$S_3 = 10.2 - 8 - 0.3 = 1.9$

$T2 > T1 > T3$
