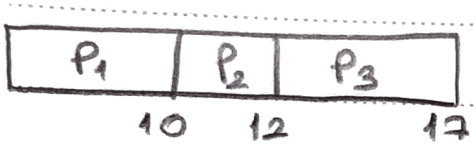
Date No

First Come First Served (Nonpreemptive)

Process	Next Burst Time
P_1	10
P_2	2
P_3	5



$WT(P_1) = 0$

$WT(P_2) = 10$

$WT(P_3) = 12$

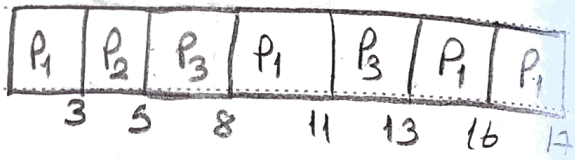
$\frac{0+10+12}{3} = 22/3$

Number of contex switch = 2

Round Robin

Process	Next Burst Time
P_1	~~10~~ ~~7~~ 4
P_2	~~2~~ 0
P_3	~~5~~ ~~2~~ 0

Time quantum = 3



$WT(P_1) = 13 - 6 = 7$

$WT(P_2) = 3$

$WT(P_3) = 11 - 3 = 8$

$\frac{7+3+8}{3}$

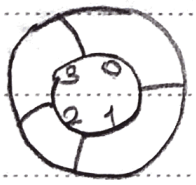
Number of context switch = 6

---

for every schedule Gantt Chart, AV(WT), Number of context switch

P = Producer → It produces an item and it inserts into a buffer.

C = Consumer → It extracts an item from a buffer.



(Circular) Buffer

Buffer size = 4

```
in = 0   Producer ( )
    while (true)
    // produce an item
    if the buffer is not full =
    Buffer [in] = new item
    in = in + 1 mod (buffer size)
    counter = counter + 1
    end
end
```

yoofun

Date No

Counter = It accounts the number of item stored into the buffer.

Buffer is empty → Counter = 0

out = 0 Consumer()

```
while(true)
    if(counter > 0)
    item = buffer[out]
    out = out + 1 mod (buffersize)
    counter = counter - 1
end
end
```

Simulation (Nonpreemptive)

	in	out	counter	
	0	0	0	→ sonraki haller
P	1		1	
P	2		2	
P	3		3	
P	0		4	
P			wait	
C		1	3	
C		2	2	
P	1		3	
P	2		4	
P			wait	

yoofun®

Date No

Producer

$$\equiv \text{counter}++ = \begin{cases} reg = counter \\ reg = reg + 1 \\ counter = reg \end{cases}$$

Consumer

$$\equiv \text{counter}-- = \begin{cases} reg = counter \\ reg = reg - 1 \\ counter = reg \end{cases}$$

Critical Section = The part of the program changing using shared variable

Semaphore $S=1 \equiv$ green light $\equiv$ C.S. free

$S=0 \equiv$ red light $\equiv$ C.S. busy

Semaphore with Passive Waiting

S.valve = 1

S.L // qveue associated to S

Semaphore with Active Waiting

```
Int s //s=1 ≡ c.s free
        s=0 ≡ c.s busy

wait(s){
    while(s<=0){
        //wait do nothing
    }
    s--
}
signal(s){
    s++
}
```

yoofun

Date No

Producer

$$\equiv \text{counter}++ = \begin{cases} reg = counter \\ reg = reg + 1 \\ counter = reg \end{cases}$$

Consumer

$$\equiv \text{counter}-- = \begin{cases} reg = counter \\ reg = reg - 1 \\ counter = reg \end{cases}$$

Critical Section = The part of the program changing using shared variable.

Semaphore $S=1 \equiv$ green light $\equiv$ C.S. free

$S=0 \equiv$ red light $\equiv$ C.S. busy

Semaphore with Passive Waiting

S.valve = 1

S.L // qveue associated to S

Semaphore with Active Waiting

```
Int s //S=1 ≡ c.s free
      S=0 ≡ c.s busy

wait(s){
    while(s<=0){
        //wait do nothing
    }
    s--
}
signal(s){
    s++
}
```

yoofun

$S_1 = 1 \quad S_2 = 1$

t_0: wait(S_1) $\quad t_1$: wait(S_2)

t_2: wait(S_2) $\quad t_3$: wait(S_1)

Date No

Process P_i Process P_j

Hp: P_i and P_j need 2 resources R_1

Example: S_1 = the semaphore associated to a file

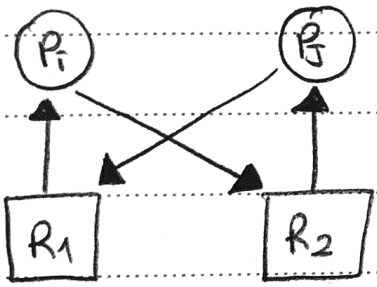
S_2 = the semaphore associated to a printer R_2

t_0 P_i wait(S_1) = P_i get access to R_1 $S_1 = 0$

t_1 P_j wait(S_2) = P_j get access to R_2 $S_2 = 0$

t_2 P_i wait(S_2) = P_i active waiting because $S_2 = 0$

t_3 P_j wait(S_1) = P_j active waiting because $S_1 = 0$



Circular Wait

$P_i \rightarrow R_2 \rightarrow P_j \rightarrow R_1 \rightarrow P_i$

Ex $R = \{ R_1, R_2, R_3, R_4 \}$

$P = \{ P_1, P_2, P_3 \}$

$E = \{ R_2 \rightarrow P_1, R_2 \rightarrow P_2, P_1 \rightarrow R_1, R_1 \rightarrow P_2, P_2 \rightarrow R_3, R_3 \rightarrow P_3 \}$

Deadlock? No the system is not in deadlock

Exit sequence: P_3 uses R_3 and release it

R_3 available

P_2 can get R_3. It has all resources, it can run, use them and return them

R_1 available

P_1 can get R_1. It has both R_1 and R_2. It can run, use the R and return them.

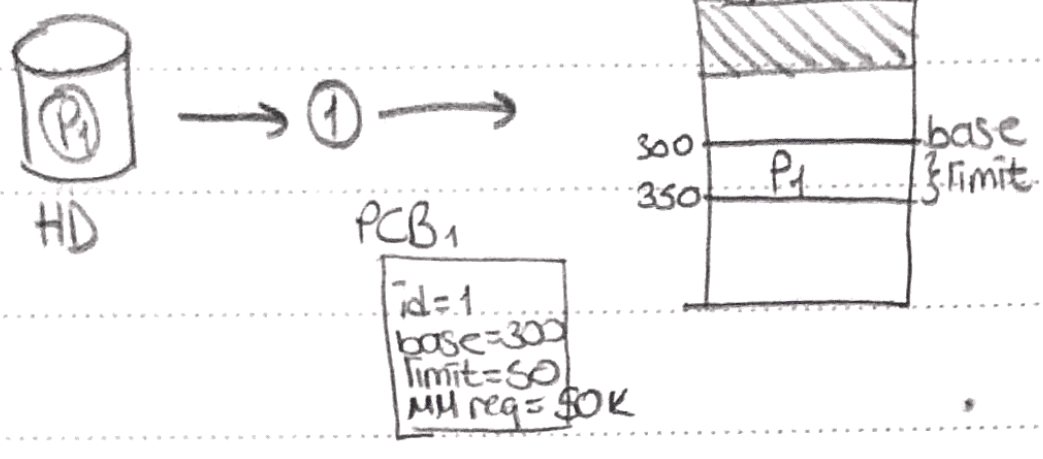
yoofun®

deadlock varsa ayde göster.

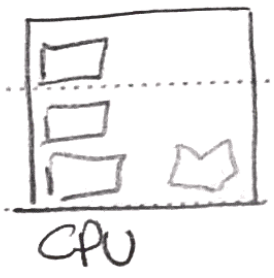
Date No

t_0 = P_1 is in NewState

t_1 = P_1 is in the Ready State



t_3 = P_1 is in the Running Sta



t_4 = P_1 waiting

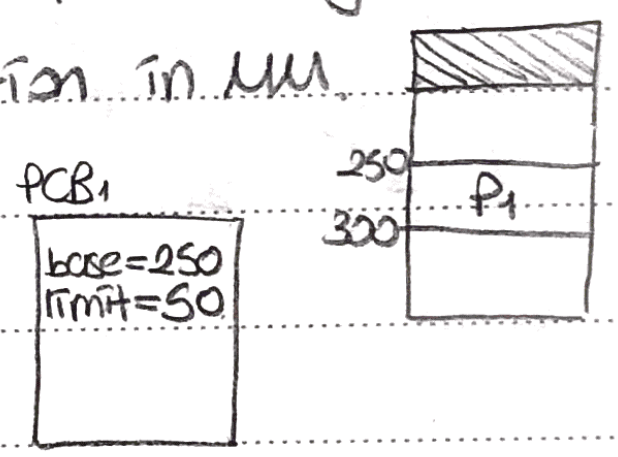
t_5 = Middle Term Scheduler
swap out P_1

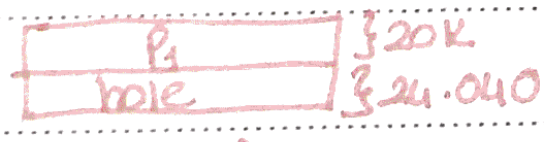
t_6 = The P that was into the C.S left (signal(S)) and moves P_1 into the Ready Suspended State

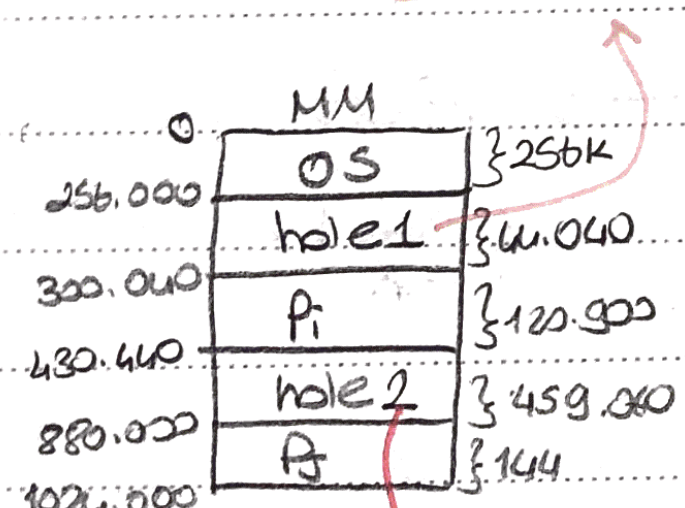
t_8 = The middle term scheduler swap in P_1

The new state of P_1 is "Ready"

P_1 has a new partition in MM





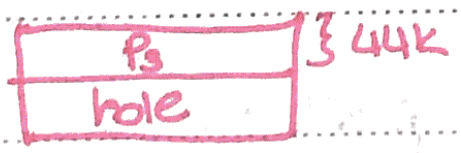


state field = free/allocated

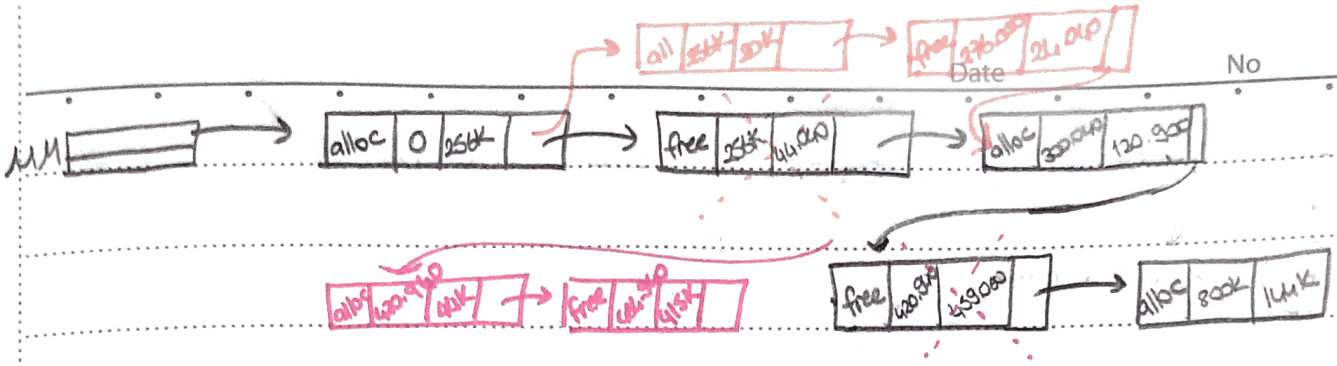
relocation = It stores the base odd un of part

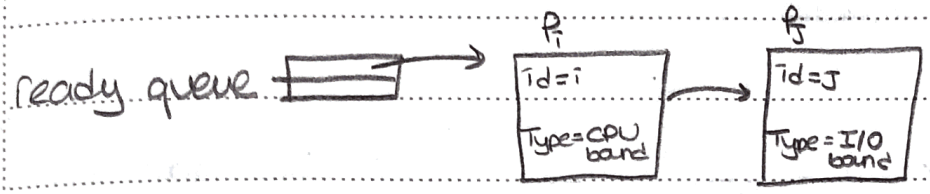
wide = To store the wide of the part

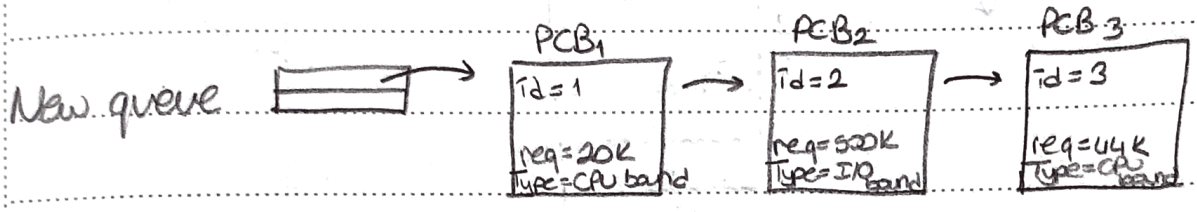
next field = pointer to the next element of the list.



yoofun







Long Term Scheduler = Best fit Algorithm → The long term scheduler assigns the nearby hole
↳ save time if we avoid to split the hole into 2.

Worse fit Algorithm → The long term scheduler assigns the biggest hole.

First fit Algorithm → The long term schedule assigns the 1st hole to P_i

quick → diğerleri bütün listeye bakmak zorunda

★ by assigning the biggest hole, the splitting the partition results in the biggest left over.

Apply best fit alg. to allocate P_3

↳ Best fit alg. assigns hole1 to P_3, little difference

↳ No division of the partition

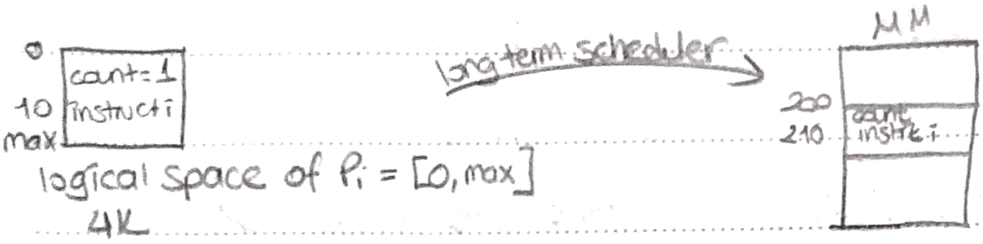
↓

The long term scheduler must look at the complete list.

Use the worst fit alg. to allocate P_3

P_2 → There is not a hole big enough → P_2 cannot be moved into the ready state

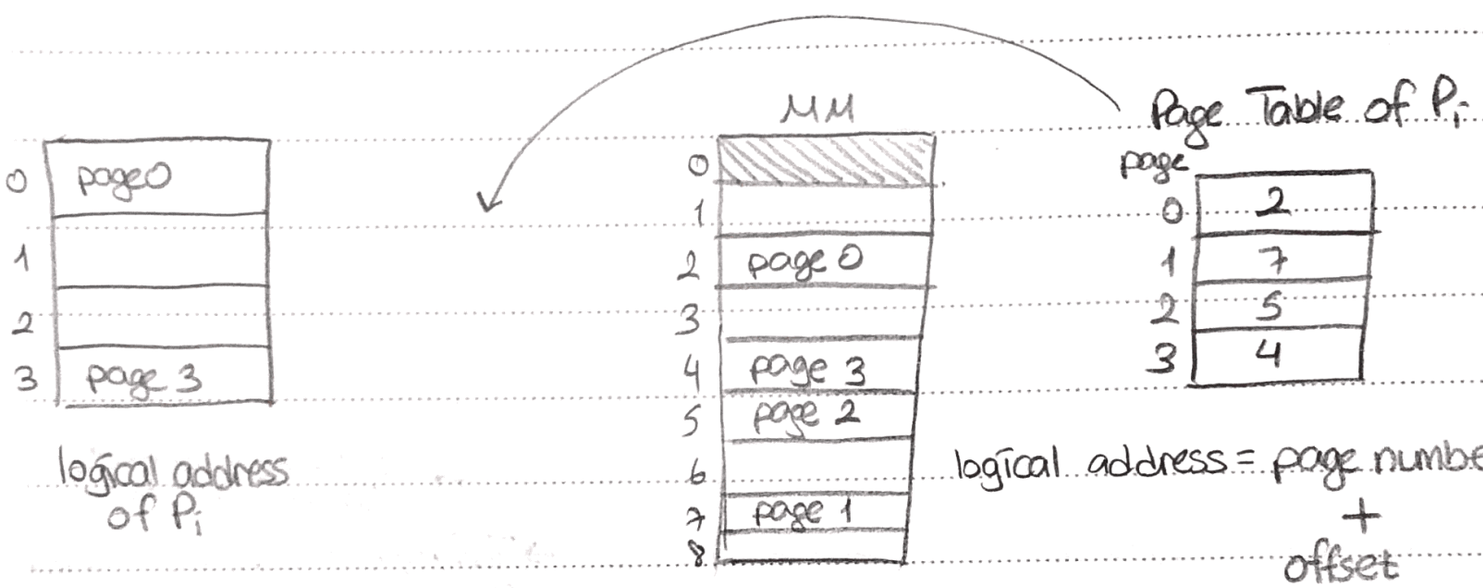
Prob → External fragmentation



logical space of P_i = [0, max]

4K

base or relocation = 200



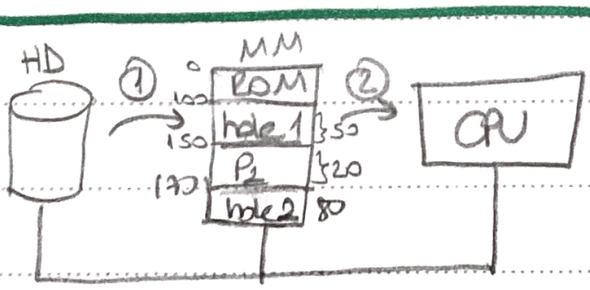
logical address of P_i

logical address = page number + offset

size(page) = size(frame)

MM = {frame}

physical address = frame number + offset



TELEMP

① Long Term Scheduler

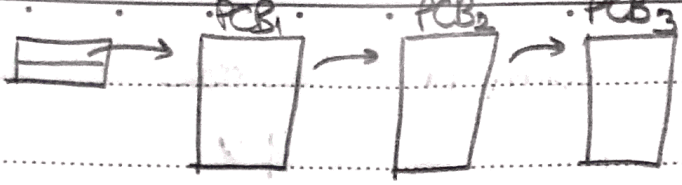
② Short Term Scheduler

Middle Term Scheduler = swap in / swap out

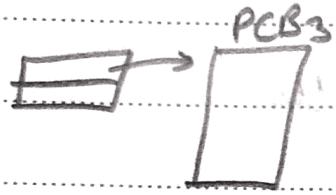
The aim of middle and long term scheduler is to have in MM a good mixture of CPU band / I/O band

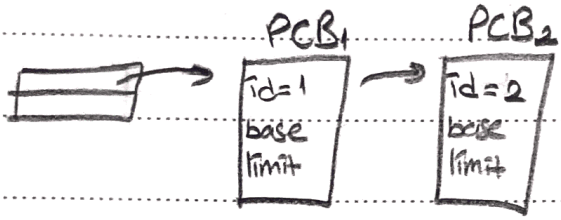
yoofun

Date No

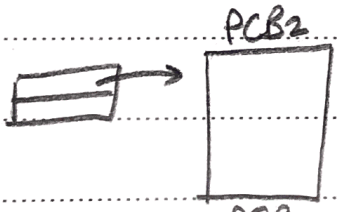
t_0: input queue 

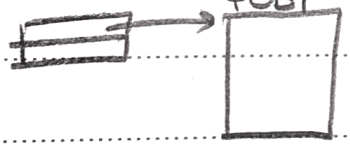
t_1: Assume that the LTS assignes a partition to P_1, P_2. Show the input queue and ready queue

input queue: 

ready queue: 

t_2: The STS moves P_1 into the running state and P_1 asks for a resource (file) and moves into the waiting state. Show the interesting queue.

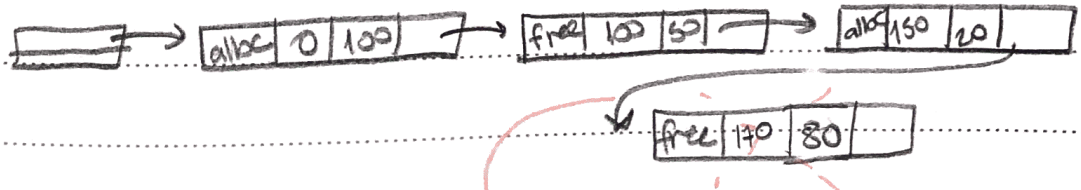
ready queue: 

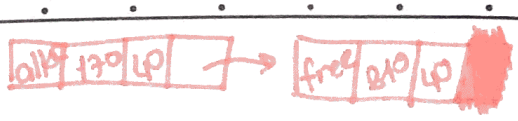
file waiting: 

t_3: The MTS swap out P_1. Which is the new state of P_1?

waiting suspended

Show the list describing MM

MM 



yoofun®

t_4: The file P_1 was waiting for I/S available and the OS assigned to P_1. Which is the new state of P_1?

Ready Suspended

Simulate worst fit algorithm to assign a new partition to P_1

P_1 is assigned to hole 2

Show the new list describing MM

yoofun®