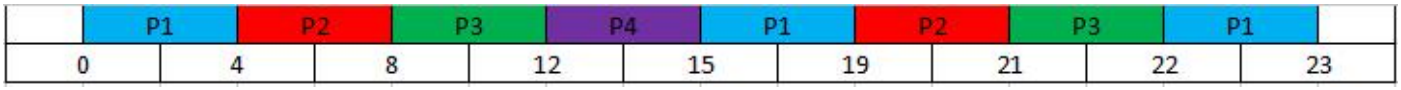
Jawaban No 1



Waiting Time					
P1	(0 - 0)	(15 - 4)	(22 - 19)	=	14
P2	(4 - 1)	(19 - 8)		=	14
P3	(8 - 2)	(21 - 12)		=	15
P4	(12 - 3)			=	9

Turn Around Time			
P1	(23 - 0)	=	23
P2	(21 - 1)	=	20
P3	(22 - 2)	=	20
P4	(15 - 3)	=	12

Pembuktian			
P1	(23 - 14)	=	9
P2	(20 - 14)	=	6
P3	(20 - 15)	=	5
P4	(12 - 9)	=	3

Rata-Rata Waiting Time		
14 + 14 + 15 + 9	=	13
4		

Rata-Rata Turn Around Time		
23 + 20 + 20 + 12	=	18,75
4		

Jawaban No 2

Terdapat beberapa Tantangan yang harus diselesaikan :

1. Mutual Exclusion = Merupakan kondisi dimana hanya ada satu proses yang mengakses suatu sumber daya. Bagian dari code program yang sedang mengakses sumber daya yang di pakai bersama dinamakan critical section.

2. Deadlock = keadaan dimana sistem seperti terhenti dikarenakan setiap proses memiliki sumber daya yang tidak bisa dibagi dan menunggu untuk mendapatkan sumber daya yang sedang dimiliki oleh proses lain.

3. Starvation = Proses yang kekurangan resource (karena terjadi deadlock) tidak akan pernah mendapat resource yang dibutuhkan sehingga mengalami starving (kelaparan).

4. Sinkronisasi = menjaga agar data tersebut tetap konsisten dan mengatur urutan jalannya proses-proses sehingga dapat berjalan dengan lancar dan terhindar dari deadlock atau starvation.

Jawaban No 3

Race Condition adalah suatu kondisi dimana dua atau lebih proses mengakses sumber daya / resource secara konkuren, hasil akhir dari data tersebut tergantung dari proses mana yang terakhir selesai dieksekusi.

Untuk **mencegah race condition**, proses – proses yang berjalan secara konkuren harus disinkronisasi.

Jawaban No 4

1. Algoritma Turn menerapkan sistem bergilir kepada kedua proses yang ingin mengeksekusi Critical Section, sehingga kedua proses tersebut harus bergantian menggunakan Critical Section. Algoritma ini menggunakan variabel bernama TURN, nilai turn menentukan proses mana yang boleh memasuki Critical Section. Hanya proses yang mempunyai ID yang sama dengan ID giliran (Turn) yang boleh masuk ke Critical Section.

2. Algoritma Flag mengantisipasi masalah yang muncul pada Algoritma Turn dengan mengubah penggunaan variabel turn dengan variabel flag. Variabel flag menyimpan kondisi proses mana yang boleh masuk Critical Section. Setiap proses mengecek status proses yg lain. Jika proses lain sedang berada di Critical Section-nya, maka dia akan menunggu sampai proses lain tersebut keluar dari Critical Section-nya. Proses yang membutuhkan akses ke Critical Section akan memberikan nilai flag-nya true. Sedangkan proses yang tidak membutuhkan Critical Section akan men-setting nilai flag-nya bernilai false.

3. Algoritma Turn-Flag Merupakan penggabungan antara Algoritma Turn dan Algoritma Flag. Sama seperti pada Algoritma Turn dan Flag, variabel turn menunjukkan giliran proses mana yang diperbolehkan memasuki Critical Section dan variable flag menunjukkan informasi proses kebutuhan akses ke Critical Section.

4. Algoritma Tukang Roti adalah solusi untuk masalah Critical Section pada sejumlah n proses. Algoritma ini juga dikenal sebagai Lamport's Baker Algorithm. Ide algoritma ini adalah dengan menggunakan prinsip penjadwalan seperti yang ada di tempat penjualan roti. Para pelanggan yang ingin membeli roti sebelumnya harus mengambil nomor urut terlebih dahulu dan urutan orang yang boleh membeli ditentukan oleh nomor urut yang dimiliki masing-masing pelanggan tersebut.

Jawaban No 5

1	2	3	1	0	1	3	4	3	2	1	4	0	4	3	1	2	0
*1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	*2	2
	*2	2	2	2	2	2	2	2	2	2	2	*0	0	0	0	0	0
		*3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
				*0	0	0	*4	4	4	4	4	4	4	4	4	4	4
Page Foult = 7																	

Jawaban No 6

1. CPU akan memberikan alamat kedalam memori
2. Alamat akan diperiksa apakah sesuai dengan base register atau tidak
- Jika Ya, maka proses dilanjutkan ke langkah 3
- JIka Tidak, maka proses dihentikan
3. Alamat hanya boleh mengakses dari base register sampai limit register
- Jika Terpenuhi, maka alamat akan masuk kedalam memori
- Jika Tidak Terpenuhi, maka proses dihentikan

Jawaban No 7

Diketahui :
- memory access (ma) = 300 ns
- page fault time = 30 ms
- probabilitas (p) = Terdapat 100000 ma terdapat 100 page fault

Ditanya :
- Effective access time

Dijawab :
- EAT = ((1-p) x 300 ns) + (p x 30 ms)

- EAT = ((1-p) x 300 ns) + (p x 30000000 ns)

- EAT = (300 - 300p) + (30000000p)

- EAT = 300 (- 300p + 30000000p)

- EAT = 300 + 29999700p nanosecond

- EAT = 300 + 29999700(100/100000) nanosecond

- EAT = 300 + 29999,7 nanosecond

- EAT = 30299,700 nanosecond

- EAT = 30,3 microsecond