CSPT0524IT – W4D1 - PRATICA

Report di Svolgimento dell'Esercizio Pratico W4D1 + Esercizio Facoltativo

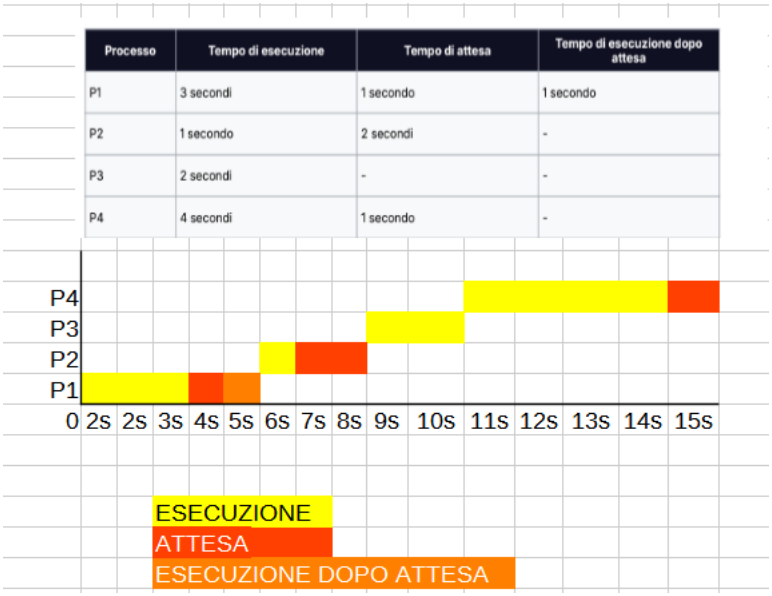
Traccia:
Si considerino 4 processi (P1, P2, P3, P4) con i tempi di esecuzione e di attesa input/output dati in tabella.
I processi arrivano alle CPU in ordine P1, P2, P3, P4.
Individuare il modo più efficace per la gestione e l'esecuzione dei processi, **tra i metodi già visti a lezione**.
Abbozzare un diagramma che abbia sulle ascisse il tempo passato da un instante «0» e sulle ordinate il nome del Processo.

Processo	Tempo di esecuzione	Tempo di attesa	Tempo di esecuzione dopo attesa
P1	3 secondi	1 secondo	1 secondo
P2	1 secondo	2 secondi	-
P3	2 secondi	-	-
P4	4 secondi	1 secondo	-

-In questo esercizio ho creato tre schemi per l'esecuzione dei processi in mono-tasking, multi-tasking e timesharing per mostrare la sequenza dei processi in termini tempo, successivamente ho risolto e continuato l'esercizio facoltativo.

-Il modo piu efficiente è il sistema Multi-Tasking con algoritmo Round Robin che mette in atto il Time-Sharing.

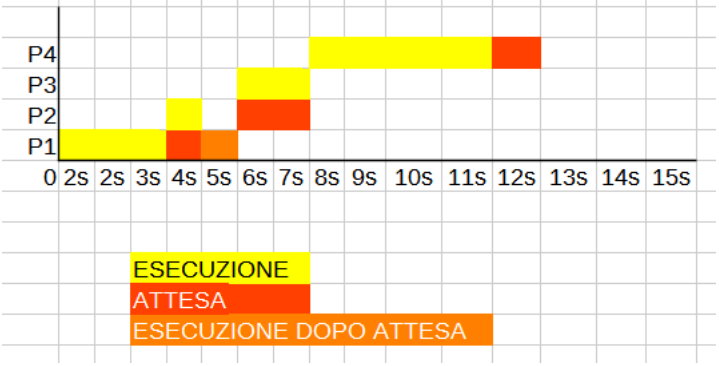
1. Grafico Mono-Tasking



- Il sistema mono-tasking consente di eseguire un processo alla volta; se il **processo1** va in esecuzione deve terminare, altrimenti non può andare in esecuzione il **processo2** che a sua volta se non termina non può andare in esecuzione il processo 3 e così via...

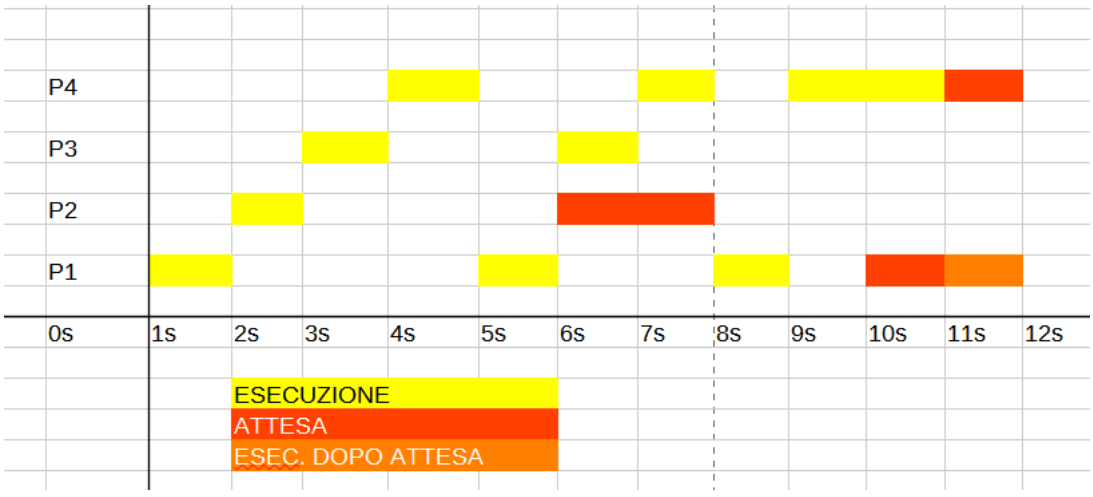
2. Grafico Multi-Tasking

Processo	Tempo di esecuzione	Tempo di attesa	Tempo di esecuzione dopo attesa
P1	3 secondi	1 secondo	1 secondo
P2	1 secondo	2 secondi	-
P3	2 secondi	-	-
P4	4 secondi	1 secondo	-



- Il sistema multi-tasking consente di avviare piu processi ma, solo se **processo1** viene messo in attesa può andare in esecuzione **processo2**.
Quando il **processo1** torna in esecuzione, il processo2 viene messo in attesa fino a termine del processo1 per poi schedulare il **processo2** e così via...

3. Grafico Multi-Tasking TimeSharing (Round Robin)



- L'algoritmo Round Robin mette in pratica il Timesharing assegnando una parte di cpu per ogni processo a un periodo di tempo (quanto), che in questo caso è di **1 secondo**.
Se il **processo1** non termina entro il periodo di tempo assegnato di **1sec (quanto)**, la cpu interrompe il **processo1** per concedere al **processo2** il suo periodo di tempo (quanto) e così via...

FACOLTATIVO:

	t_0	T_x
P1	0	14
P2	30	16
P3	6	40
P4	46	26
P5	22	28

time slice	Inizio	Fine	Processo	Resto	Attesa
1	0	12	P1	2	(P3 t6)
2	12	24	P3	28	(P1 resto 2) + (P5 t22)
3	24	26	P1 → FINE	0	(P5 t22) + (P3 resto28)
4	26	38	P5	16	(P3 resto28) + (P2 t30) + (P5 resto16)
5	38	50	P3	16	(P2 t30) + (P5 resto16) + (P3 resto16) +(P4 t46)
6	50	62	P2	4	(P5 resto16) + (P3 resto16) + (P4 t46) + (P2 resto4)
7	62	74	P5	4	(P3 resto16) + (P4 t46) + (P2 resto4) + (P5 resto4)
8	74	86	P3	4	(P4 t46) + (P2 resto4) + (P5 resto4) + (P3 resto4)
9	86	98	P4	14	(P2 resto4) + (P5 resto4) + (P3 resto4) + (P4 resto14)
10	98	102	P2 → FINE	0	(P5 resto4) + (P3 resto4) + (P4 resto14)
11	102	106	P5 → FINE	0	(P3 resto4) + (P4 resto14)
12	106	110	P3 → FINE	0	(P4 resto14)
13	110	122	P4	2	(P4 resto2)
14	122	124	P4 → Fine	0	

- Ho aggiunto la colonna dei resti e la colonna dei processi in attesa e quindi da completare.

NOTA:
Legenda per comprendere la lista di attesa dei processi che ho scritto
(P3 t6) = Processo 3 con Tempo di avvio di 6millisecondi.
(P1 resto2) = Processo 1 con un resto di 2ms per terminare.

Francesco Rinaldi