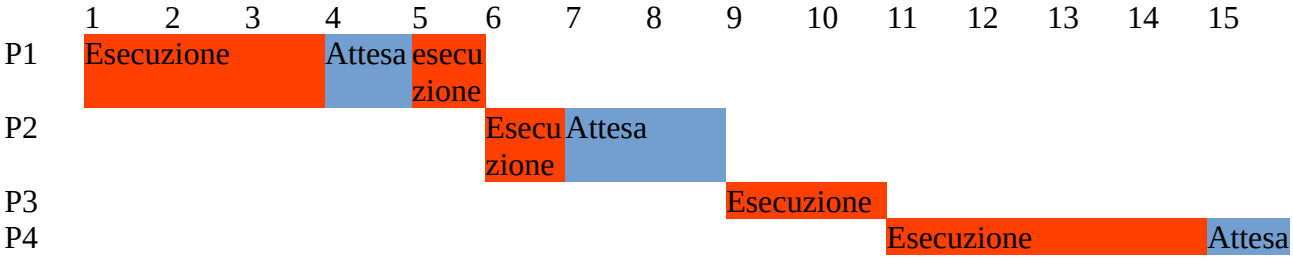
Consegna week 4 day 1 – Daniele Rufo

L’esercizio di oggi verte sui meccanismi di pianificazione dell’utilizzo della CPU (o processore). In ottica di ottimizzazione della gestione dei processi, abbiamo visto come lo scheduler si sia evoluto nel tempo per passare da approccio mono-tasking ad approcci multi-tasking. 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.

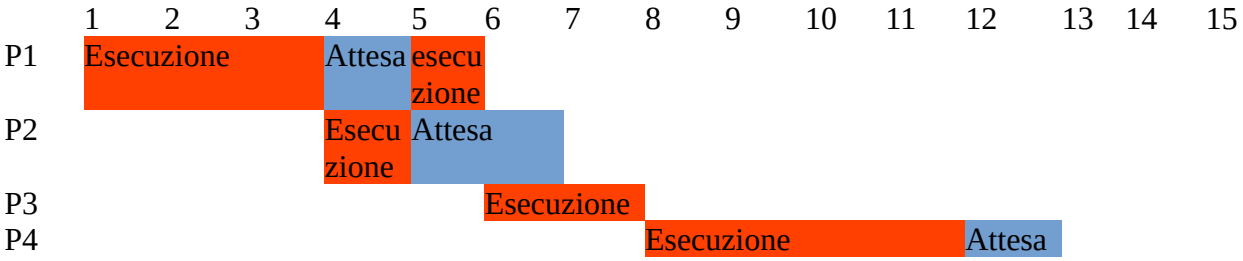
Processo Tempo di esecuzione Tempo di attesa Tempo di esecuzione dopo attesa P1 3 secondi 1 secondo 1 secondo P2 1 secondo 2 secondiP3 2 secondi --

MONO - TASKING



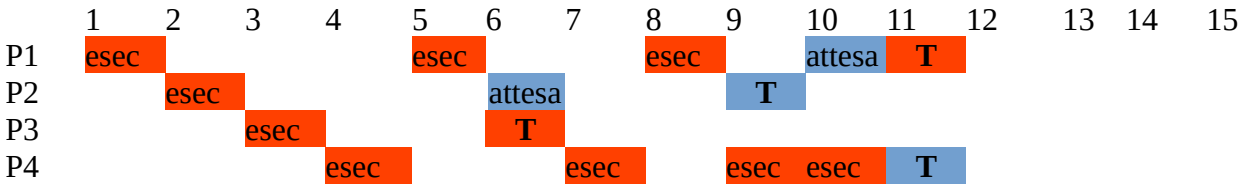
Il diagramma rappresenta come lo scheduler gestisca i processi con un approccio mono tasking. Quest’ultimo non consente il preemption e quindi un processo non può essere arrestato per dare priorità ad altri o eliminare i tempi di inutilizzo. Ne consegue che tutti i tempi di attesa input/output di ogni processo corrisponde al tempo di inutilizzo della CPU.
Il tempo totale per eseguire i 4 processi è 15 secondi di cui 4 sono di inutilizzo CPU.

MULTI-TASKING



In questo caso lo scheduler ha la possibilità di dare priorità ai vari processi ma soprattutto di eseguire un processo mentre un altro è in attesa di input/output. Questo permette un efficientamento nella gestione della CPU perché come vediamo in tabella se è vero che il processo in esecuzione è sempre e solo uno soltanto è vero anche che le fasi di attesa ed esecuzione (relative a processi diversi) possono avvenire contemporaneamente.
Il tempo totale per eseguire i processi è 12 secondi di cui 1 è di inutilizzo.

MULTI-TASKING (versione time sharing)



Legenda:
T=il processo è completato.

In questa versione lo scheduler mantiene la possibilità di interrompere un processo per dare priorità agli altri, ma a differenza del multi-tasking semplice assegna una porzione di tempo finita (time slice o quantum) ogni qual volta manda in esecuzione un processo. Nel grafico ho ipotizzato un time slice di 1 secondo (nella realtà avviene tutto molto più velocemente) e seguendo l'ordine di arrivo e quindi di priorità ho sviluppato le assegnazioni dello scheduler alla CPU.
Il tempo totale per eseguire i processi è 11 secondi e non ci sono momenti in cui il processore rimane fermo in attesa di input/output.

Conclusione:

Il modello multi – tasking versione time sharing risulta il più efficace di tutti visto che come dimostrato impiega meno tempo per portare a termine tutti i processi eliminando i tempi di inutilizzo del processore.