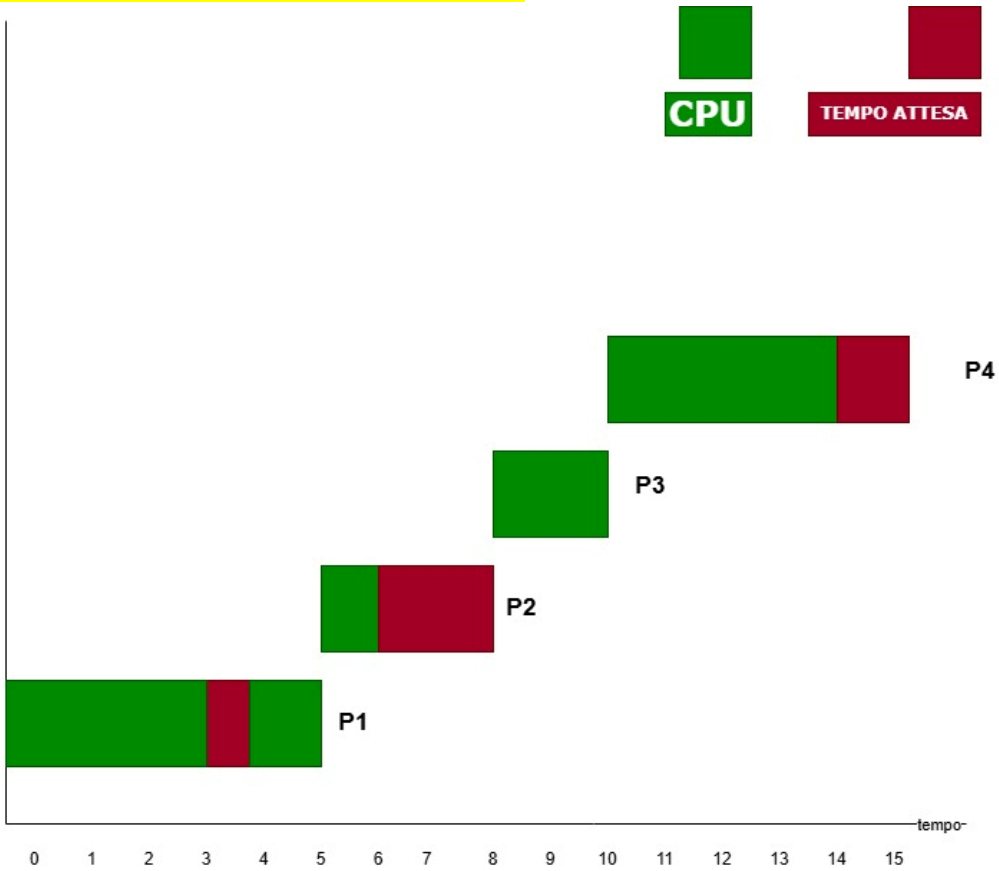
ESERCIZIO 1

INTRODUZIONE ED OBBIETTIVI:

L'esercitazione ha lo scopo di analizzare i meccanismi di pianificazione dell'utilizzo della CPU, osservando come la modalità di esecuzione dei processi influisca sull'efficienza complessiva del sistema operativo.
In particolare, il lavoro si concentra sul confronto tra diverse strategie di scheduling, a partire dall'approccio mono-tasking, a seguire il multi-tasking ed a finire il time-sharing.
L'obiettivo principale è comprendere come la CPU gestisce i processi, distinguendo tra tempo di esecuzione e tempo di attesa. Attraverso la simulazione e la rappresentazione grafica, vengono evidenziati i vantaggi e i limiti di ciascun approccio, con particolare attenzione all'impatto sul tempo totale di completamento e sull'utilizzo della CPU.

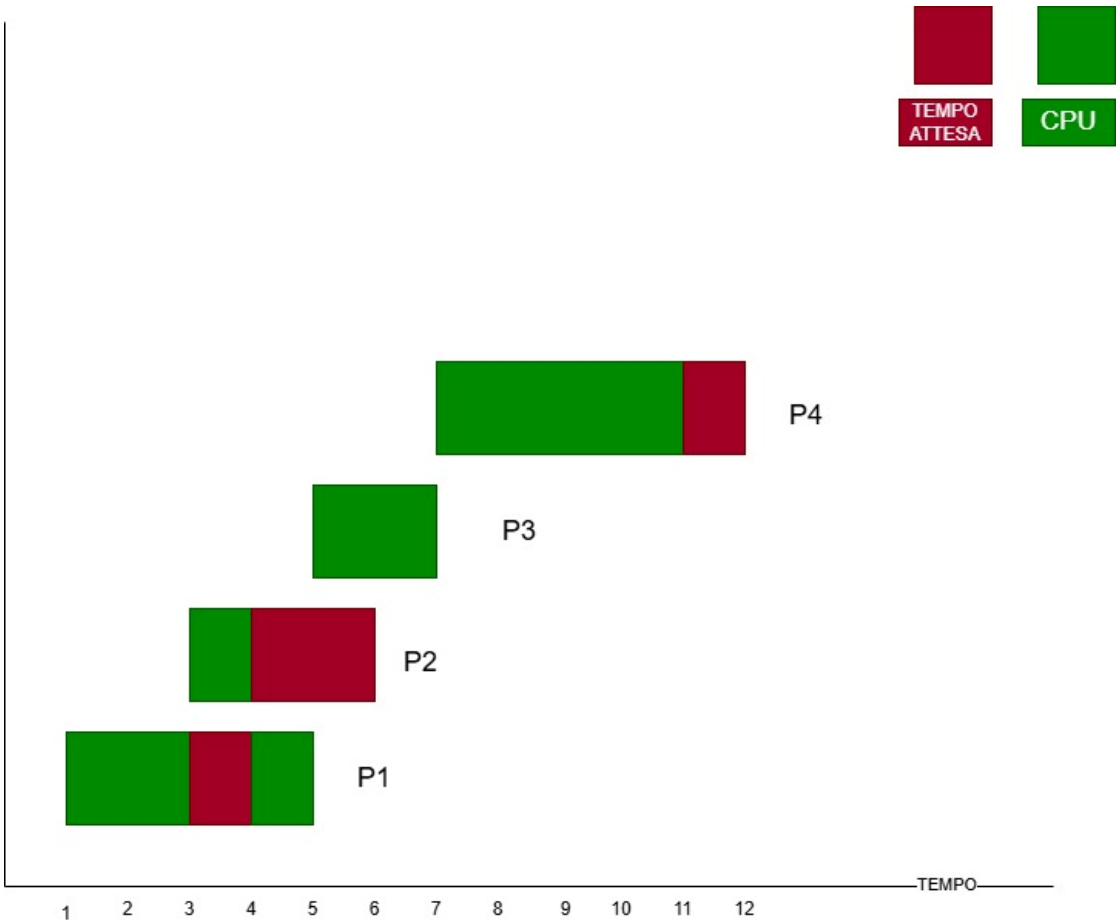
MODELLO MONO TASKING



Nel grafico realizzato, la parte verde rappresenta l'intervallo di tempo in cui la CPU è occupata con l'elaborazione del processo, mentre la parte rossa indica i periodi di attesa legati alle operazioni di input/output.
I processi P1, P2, P3 e P4 sono stati eseguiti in ordine sequenziale, rispettando il loro tempo di esecuzione e di attesa.

Nel processo mono-tasking possiamo osservare come la CPU gestisca un solo processo per volta portandolo a termine completamente prima di iniziare quello successivo.
Nel grafico si nota chiaramente che durante l'esecuzione di ogni processo (P1, P2, P3 e P4) la CPU lavora in modo sequenziale, senza sovrapposizioni.
Il tempo totale impiegato per completare tutti i processi è di circa 15 secondi, valore che mette in evidenza quanto questa modalità, pur essendo semplice e lineare, comporti tempi di attesa elevati e un utilizzo non ottimale delle risorse della CPU, che rimane inattiva durante le fasi di I/O.

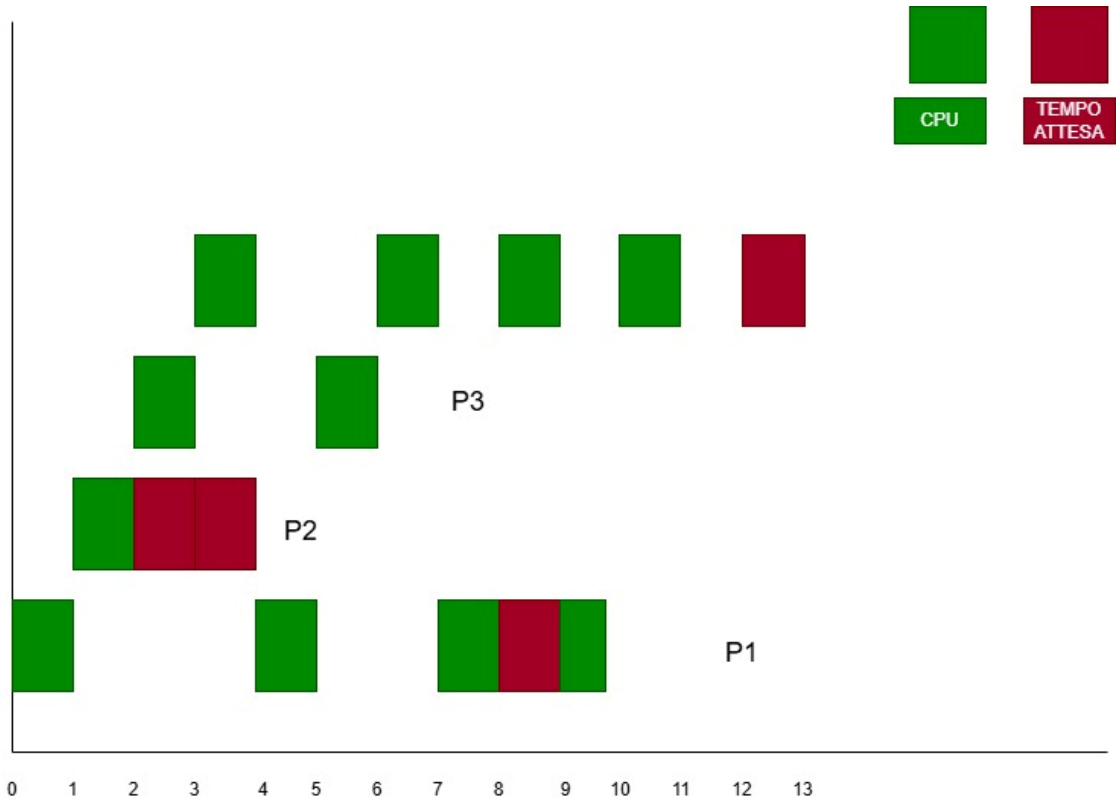
MODELLO MULTI-TASKING



Nel processo multi-tasking la CPU non lavora più su un solo compito alla volta, ma riesce a gestire più processi insieme, alternandole l'esecuzione
In pratica, mentre un processo è in attesa di un'operazione di input/output, la CPU passa a un altro, così da non restare mai ferma.
Nel grafico si nota come la CPU si sposti rapidamente tra i vari processi (P1, P2, P3 e P4), evitando di rimanere inattiva.

Come possiamo evincere dal grafico il tempo complessivo necessario per terminare tutti i processi si riduce mostrando un miglioramento evidente rispetto alle tempistiche del modello mono-tasking.
Il sistema risulta quindi più efficiente e reattivo, perché la CPU riesce a sfruttare meglio le proprie risorse e a lavorare in modo continuo, senza lunghi periodi di inattività.

MODELLO TIME SHARING



Nel time-sharing la CPU suddivide il proprio tempo in piccoli intervalli regolari (quanti) che vengono assegnati a turno ai vari processi attivi.
Se un processo non termina entro il tempo stabilito, viene momentaneamente sospeso e rimesso in coda, per poi riprendere al turno successivo. In questo modo tutti i processi procedono gradualmente, evitando che uno resti bloccato troppo a lungo; nel nostro caso, il quantum scelto consente di mantenere un buon ritmo: i processi ricevono turni regolari e la CPU non resta inutilizzata.

CONCLUSIONE
L'attività svolta ha permesso di comprendere in modo concreto come la gestione della CPU influenzi l'efficienza del sistema.

Attraverso la simulazione dei tre modelli è stato possibile osservare la progressiva ottimizzazione; l'esperienza ha evidenziato l'importanza di una corretta pianificazione della CPU per assicurare prestazioni stabili, tempi di risposta più rapidi e un utilizzo ottimale delle risorse.

ESERCIZIO 2

L'esercitazione ha avuto come scopo quello di analizzare il funzionamento dello scheduling della CPU, applicando in particolare la politica Round Robin con un time slice di 12 millisecondi.
L'obiettivo era comprendere come la CPU gestisca più processi contemporaneamente, alternandone l'esecuzione secondo intervalli di tempo regolari, e valutare l'impatto di questa strategia sui tempi di attesa e di turnaround.
Attraverso la simulazione sono stati osservati cinque processi (P1, P2, P3, P4 e P5) con tempi di arrivo e di esecuzione differenti. L'esperimento ha permesso di calcolare, passo dopo passo, il momento in cui ogni processo entra in CPU, quando viene sospeso e rimesso in coda ed infine quando termina la propria esecuzione.

P	t. arrivo	t. durata
P1	0	14
P2	30	16
P3	6	40
P4	46	26
P5	22	28

t.slice	Processo	inizio	fine	fine processo	Coda a fine time slice
1	P1	0	12		P3 P(12)
2	P3	12	24		P1(12) P(5) P3(12)
3	P1	24	26	FINE P1	P5 P3(12)
4	P5	26	38		P3(12) P2 P5(12)
5	P3	38	50		P2 P5(12) P4 P3(24)
6	P2	50	62		P5(12) P4 P3(24) P2(12)
7	P5	62	74		P4 P3(24) P2(12) P5(24)
8	P4	74	86		P3(24) P2(12) P5(24) P4(12)
9	P3	86	98		P2(12) P5(24) P4(12) P3(36)
10	P2	98	102	FINE P2	P5(24) P4(12) P3(36)
11	P5	102	106	FINE P5	P4(12) P3(36)
12	P4	106	112		P3(36) P4(24)
13	P3	118	122	FINE P3	P4(24)
14	P4	122	124	FINE P4	

processo	inizio	durata	fine	TURN AROUND T. TOTALE	T.ATTESA
P1	0	14	26	26	12
P2	30	16	102	72	56
P3	6	40	122	116	76
P4	46	26	124	78	52
P5	22	28	106	84	56
TOTALE				376	252
MEDIA				75.2	50,4

Il lavoro è stato svolto costruendo una tabella in Excel che riporta per ogni time slice, il processo in esecuzione, l'intervallo di tempo di inizio/fine e la coda dei processi in attesa alla fine del turno.
Ogni volta che un processo non completava la sua esecuzione entro i 12 ms, veniva interrotto e rimesso in coda, in attesa del turno successivo.
In questo modo, tutti i processi avanzavano gradualmente, senza che uno monopolizzasse il processore.
Infine nella parte inferiore della tabella sono stati calcolati i tempi di turnaround e di attesa per ciascun processo e ,successivamente, le medie complessive.
Questi risultati dimostrano il corretto funzionamento dello scheduling, evidenziando come la CPU riesca a mantenersi costantemente attiva e a distribuire il tempo di elaborazione in modo equilibrato tra tutti i processi.