MultiTasking

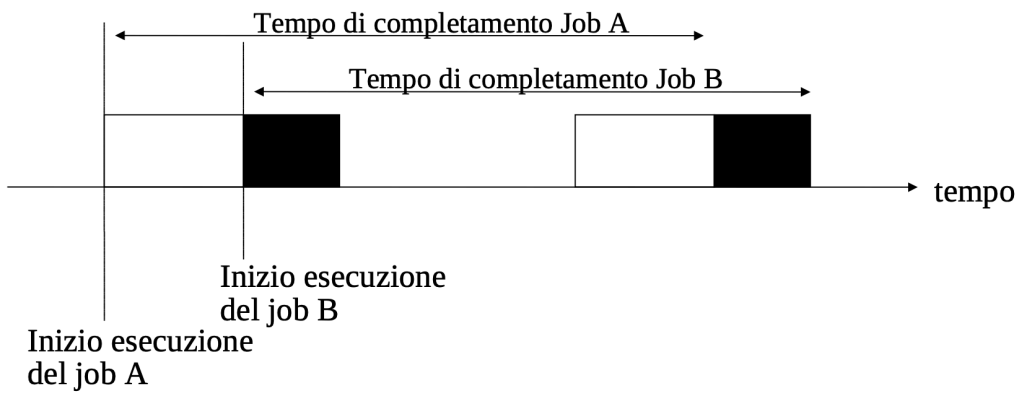
lunedì 13 marzo 2023 14:42

MULTITASKING

SI PARLA DI GESTIONE SIMULTANEA DI MOLTEPLICI JOB: MENTRE UN JOB A È ATTIVO, POSSO AVERE UN JOB B ATTIVO!

NEL MOMENTO IN CUI ABBIAMO UN JOB IN ESERCIZIO E QUESTO JOB VA A CHIAMARE IL MONITOR PER UN'INTERAZIONE CON UN DISPOSITIVO A QUESTO PUNTO LA CPU NON SERVE PIÙ A QUESTO JOB PER UN CERTO INTERVALLO DI TEMPO, LA CPU VIENE CEDUTA AD UN ALTRO JOB.

- gestione simultanea di molteplici job
- un job per volta impegna la CPU
- piu' job possono impegnare contemporaneamente i dispositivi



Ipotesi ⟹ job A e job B inoltrano richieste a dispositivi diversi

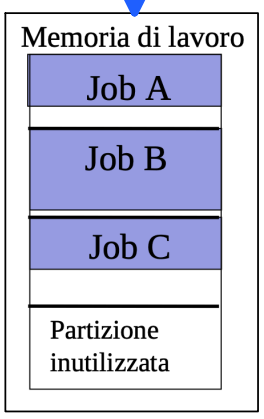
SE ABBIAMO PIÙ JOB ATTIVI, DOVE METTEREMO TUTTI QUESTI JOB NELLA NOSTRA ARCHITETTURA DI MEMORIZZAZIONE?

PER POTER ESEGUIRE LE ISTRUZIONI MACCHINA DEI JOB, ESSE DEVONO ESSERE PRESENTI ALL'INTERNO DELLA RAM, ALTRIMENTI LA NOSTRA CPU NON RIESCE AD ESEGUIRE IL fetch.

BENE, TUTTI QUESTI JOB DOVE POTEVANO ESSERE MEMORIZZATI?

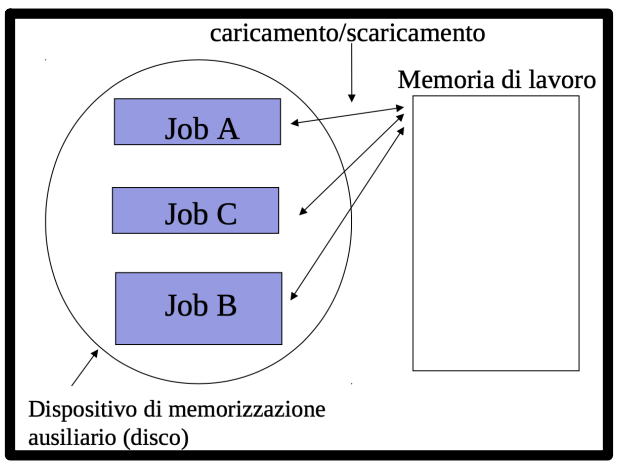
- ALL'INTERNO DELLA MEMORIA DI LAVORO

SE IL JOB A È IN ESECUZIONE, CHIAMA OVVIAMENTE IL MONITOR CHE DA QUALCHE PARTE IN MEMORIA È PRESENTE, EVENTUALMENTE IL MONITOR PUÒ TORNARE IL CONTROLLO AD UN ALTRO JOB.



CON PARTIZIONI MULTIPLE.

- DISCO AUSILIARIO



RAPIDO PASSAGGIO di dati dal disco STESSO alla memoria di lavoro, eccolo lì, come PRIMA.

NOI POTEVAMO caricare IL JOB A all'INTERNO della memoria E DARGLI IL CONTROLLO.

QUANDO IL JOB A VA A chiAMARE IL MONITOR, IL JOB A VIENE RIPORTATO FUORI dalla MEMORIA E NELLA STESSA VIENE CARICATO IL JOB C O IL JOB B.

QUESTE SONO SOLUZIONI CHE SONO ARRIVATE NEGLI ANNI.

Monoprogrammazione vs multiprogrammazione

SINGOLO JOB ATTIVO, finché NON COMPLETO L'ESECUZIONE di QUESTO JOB NON POSSO ATTIVARNE ALTRI.

MANTENIAMO PIÙ di UN JOB ATTIVO ALLO STESSO ISTANTE di TEMPO.
Poi MAGARI UN JOB STA ASPETTANDO UN dispositivo E UN ALTRO STA UTILIZZANDO la CPU.

	monoprogrammazione	multiprogrammazione
Utilizzo pocessore	17%	33%
Utilizzo del disco	33%	67%
Utilizzo stampante	33%	67%
Tempo totale	30 min	15 min
Throughput	6 job/ora	12 job/ora
Tempo di risposta medio	18 min	10 min

IN GENERALE SU UN SISTEMA OPERATIVO BATCH, NOI ABBIAMO CHE IL CONTROLLO RITORNA AL MONITOR IN FUNZIONE di SCELTE dell'APPLICAZIONE.

Quindi se l'APPLICAZIONE chiama UN dispositivo IL MONITOR PUO PRENDERE IL CONTROLLO E PASSARE poi IL CONTROLLO Ad un altro JOB.

Se questa cosa NON AVVIENE, LA CPU RESTA A VANTAGGIO del JOB che STA CORRETTAMENTE UTILIZZANDOLA.
Se questa termina noi possiamo PASSARE la CPU Ad un altro JOB PER ESEGUIRE delle ISTRUZIONI MACCHINA, altrimenti NO!

SE NOI SAPPIAMO GESTIRE LA MULTIPROGRAMMAZIONE, IN REALTÀ CON UN SISTEMA BATCH MULTIPROGRAMMATO NON SIAMO ANCORA IN GRADO DI STABILIRE QUALI DEBBANO ESSERE GLI INTERVALLI DI TEMPO IN CUI QUESTI JOB, debbano EFFETTIVAMENTE UTILIZZARE LE RISORSE DELL'HARDWARE.

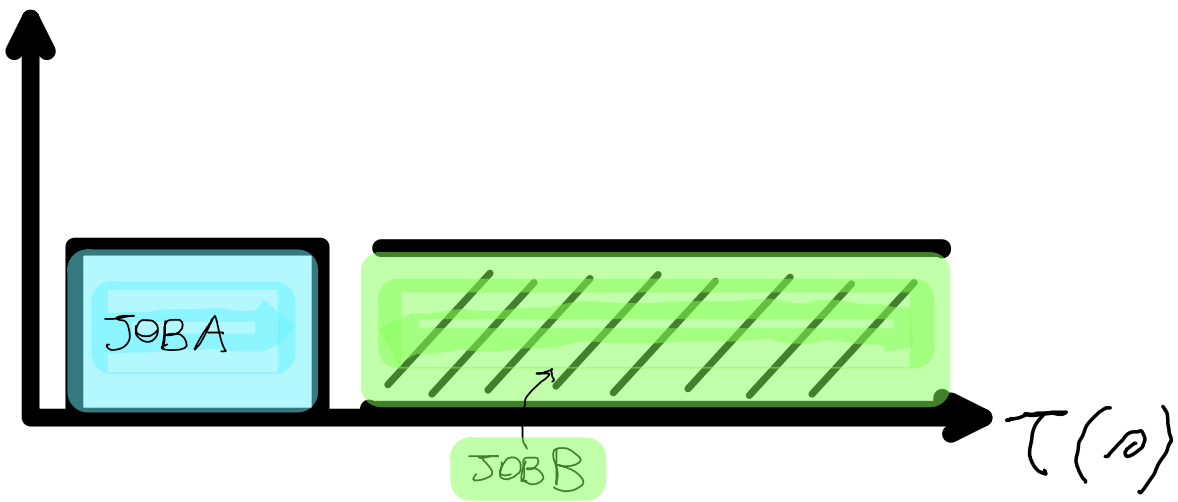
→ IN PARTICOLARE LA RISORSA PIÙ IMPORTANTE che NOI ABBIAMO che È LA CPU.

QUESTA COSA È OVVIAMENTE PROBLEMATICA PER UNA CLASSE di APPLICAZIONI:

APPLICAZIONI INTERATTIVE!

SE AVVIAMO L'ESECUZIONE di un JOB, CON frequenti RICHIESTE di INTERAZIONE CON UN dispositivo, IN UN'ARCHITETTURA BATCH MULTIPROGRAMMATA, questo JOB può AVERE delle PROBLEMATICHE.

ATTENZIONE: i JOB CON frequenti RICHIESTE di INTERAZIONE CON UN dispositivo, SONO QUELLI che TIPICAMENTE USIAMO SEMPRE.



Supponiamo che questo JOB B, MANTERRÀ l'utilizzo della CPU per lungo tempo.
Anche è in grado di prendere il CONTROLLO della CPU PER RICOMINCIARE EVENTUALMENTE la produzione di qualche OUTPUT che deve essere visualizzato da qualche utente, per esempio.
QUESTO È UN PROBLEMA quando ABBIAMO APPLICAZIONI di questo TIPO ATTIVE NEL SISTEMA.

ESISTE ANCHE IL RISCHIO DI SOTTOUTILIZZO DELLE RISORSE: LA CPU POTREBBE ESSERE OCCUPATA DA UN ALTRO JOB, E L'INTERAZIONE COL DISPOSITIVO NON POSSIAMO effettuarla!
ABBIAMO L'IMPOSSIBILITÀ di GESTIRE IN MANIERA CORRETTA, IL TIMING delle ATTIVITÀ che AVVENGONO all'INTERNO dell'ESECUZIONE di QUESTI JOB.

QUESTO È STATO IL MOTIVO che HA PORTATO ALLO SVILUPPO DI NUOVE famiglie di S.O

TIME SHARING. ✓