Introduzione

Componenti di un sistema

Quando si esegue un programma bisogna portarlo in memoria centrale e poi eseguirlo, la memoria secondaria è troppo lenta. Se i livelli superiori interagissero con quelli inferiori ci sarebbero problemi. L'interazione con la memoria secondaria varia in base al modello e alla ditta che l'ha prodotta. Certe cose non possono essere delegate ai programmi di alto livello in quanto aumenterebbe il rischio di errori o si scriverebbero programmi poco portabili. Le interfacce fornite dai sistemi operativi sono discrete. Richiede un utilizzo efficiente delle risorse fisiche, sia per motivi di correttezza che per motivi di efficienza. Se più processi sono in esecuzione concorrente, tutti sono in esecuzione sulla memoria centrale. Questa va gestita per fare in modo che tutti i programmi possono essere eseguiti in concorrenza senza andare in conflitto. Un altra ragione per l'uso efficiente delle risorse fisiche è la sicurezza e protezione. Il sistema operativo deve tenere conto di eventuale software prodotto per arrecare danni alla macchina. Se non ci fosse il sistema operativo, tutte queste cose dovrebbero essere gestite dalle singole applicazioni. Si utilizzano sistemi operativi anche per motivi di convenienza: deve garantire agli utenti un facile utilizzo della macchina.

Componenti del computer

- Hardware (CPU, memoria, dispositivi di I/O)
- Sistema operativo
- Applicazioni e programmi
- Utenti (persone, macchine o altri computer)

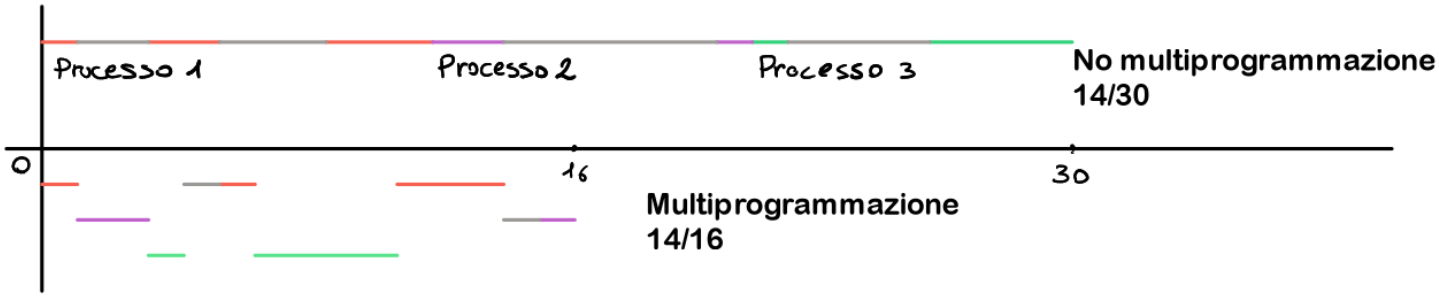
Componenti del SO

- Resource Allocator: gestisce e assegna le risorse
- Control Program: controlla l'esecuzione dei programmi lanciati dall'utente
- Kernel: il programma che è sempre in esecuzione: I primi computer erano poco efficienti, sia in termini di prestazione che di gestione di memoria. Un primo passo per risolvere questo problema è la multiprogrammazione. La multiprogrammazione consente alla macchina di eseguire più processi contemporaneamente al posto di usare un sistema sequenziale. Dato che il sistema operativo è un programma, ha bisogno di spazio in memoria (centrale) per funzionare.

Multiprogrammazione

Il processore è estremamente veloce, ma i comandi di I/O eseguiti dai job no, la multiprogrammazione fa si che il processore non perda tempo aspettando la fine dell'esecuzione dei singoli job. Quando viene inviato un comando di I/O viene mandato in esecuzione un altro job per far si che il processore non perda tempo. Ci sono processi nel sistema operativo che lo notificano dell'inizio di queste esecuzioni per poi fargli partire un altro job. Se job 2 richiede un altro comando di I/O il SO farà partire il job 3. Il processore non percepisce questo cambio di istruzioni/job.
Bisogna modificare i SO in maniera sostanziosa per farli funzionare con la multiprogrammazione. Dato che la memoria centrale è condivisa, il SO deve essere attento a usare dati relativi al processo che sta

eseguendo. Ci saranno anche situazioni quando i dati tra due processi vorranno essere condivisi. Il SO fornisce processi al processore, e determina anche la priorità. Ci sono diverse politiche che si appoggiano sugli stessi meccanismi.



Time sharing

Il time sharing è stato introdotto quando ci si è resi conto che è possibile interagire con il computer, modificando dati che il computer possiede solitamente in memoria secondaria. Diventa importante il tempo di reazione della macchina agli input dell'utente. Se è già in esecuzione un programma intensivo che ci mette molto tempo a completare. Quindi vengono aggiunti dei context switch aggiuntivi per quando scade il quanto di tempo, cioè il tempo deciso dal sistema operativo per portare a termine un processo, che non deve essere neanche troppo breve altrimenti nessun processo verrebbe mai portato a termine.
Api e system call: istruzioni che il SO fornisce e il processore eseguisce come se fossero istruzioni indivisibili (al più basso livello).

Sistemi Paralleli

Sistemi con più processori con CPU in comunicazione. Con sistemi paralleli si ha un maggiore. Il throughput è un indice di performance del sistema, ovvero quanti processi per unità di tempo vengono gestiti. Inoltre nel caso una delle CPU si guasti non ci sono grossi problemi. Ci sono sistemi paralleli simmetrici e asimmetrici. Architetture asimmetriche hanno un processore principale e altri dipendenti.

Sistemi Real Time

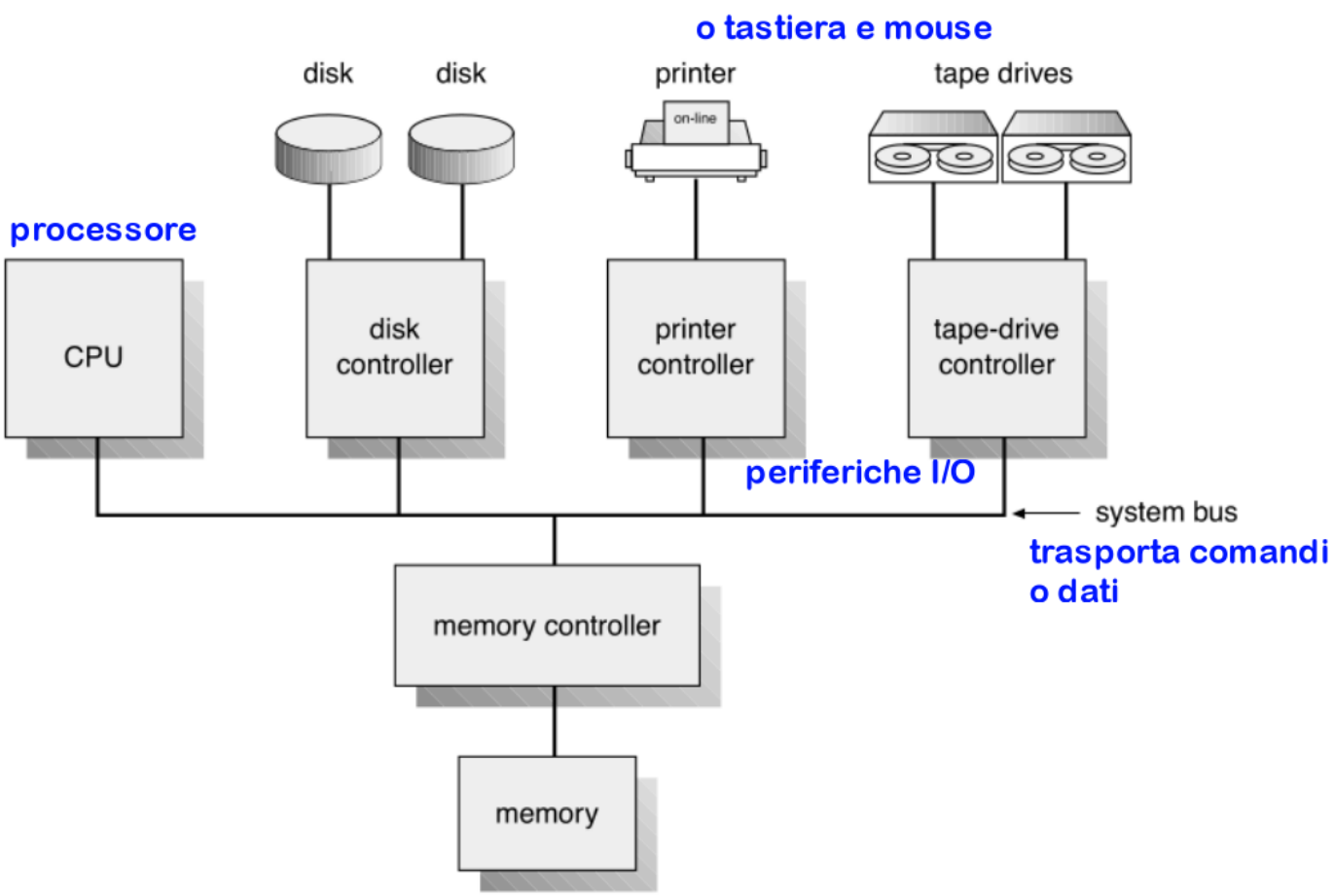
Ci sono macchine nelle quali non basta che un processo venga eseguito, è necessario che vengano rispettati dei vincoli temporali, ad esempio i sensori.

- Hard real time: il vincolo temporale deve assolutamente essere rispettato
- Soft Real time
  Il tempo di overhead è la percentuale di tempo in cui la CPU è utilizzata da parte del sistema operativo rispetto al tempo totale, usato nei sistemi di allarme

Memoria virtuale

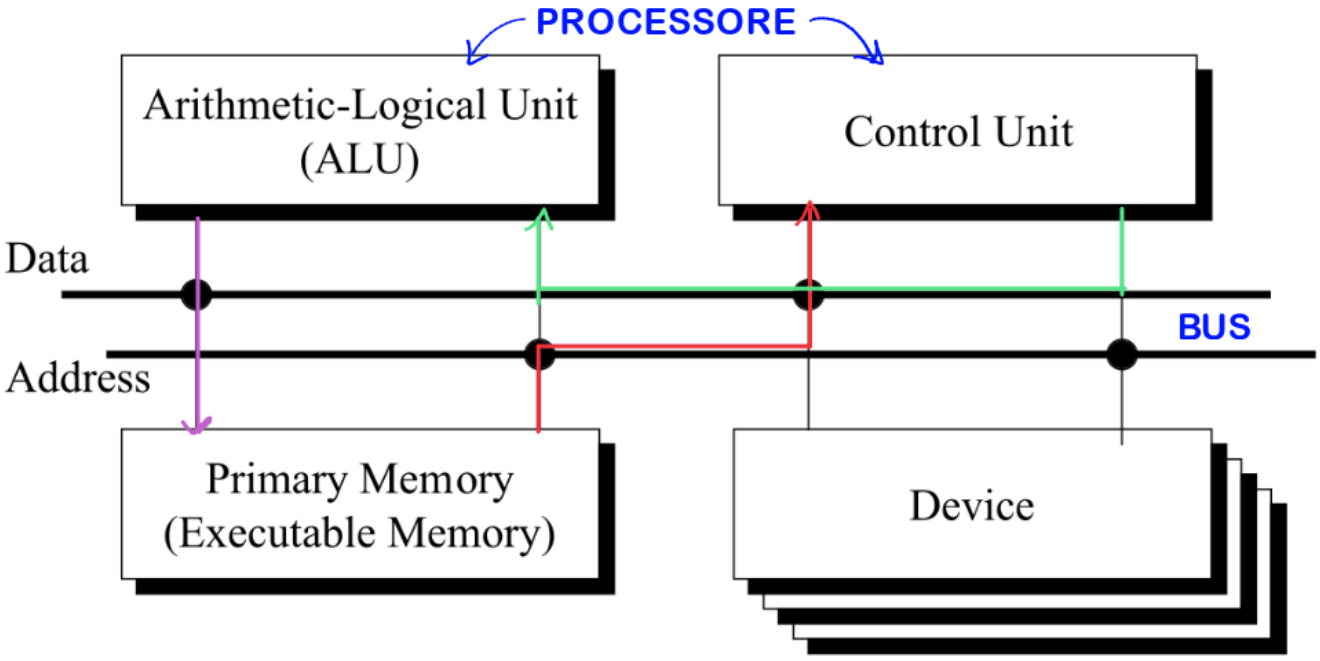
La memoria virtuale è un'architettura di sistema capace di simulare uno spazio di memoria centrale (memoria primaria) maggiore di quello fisicamente presente o disponibile, dando l'illusione all'utente di una quantità maggiore di memoria.

Architettura Computer



Il SO invia alla CPU le istruzioni, e questa in cicli esegure parte di esse.

Computer di Von Neumann



La ALU esegue le operazioni aritmetiche e booleane. La control unit è il cervello del processore, determina la prossima istruzione da eseguire dalla memoria centrale e invia i calcoli da fare alla ALU. La memoria centrale è un insieme ordinato di celle, e ad ogni cella corrisponde un indirizzo di memoria. In ogni cella è memorizzato un valore o un'istruzione.

Register

I registri sono una piccola memoria ad alta velocita' utilizzata per memorizzare i risultati temporanei e le informazioni di controllo necessarie al funzionamento dell'ALU. I più importanti sono:

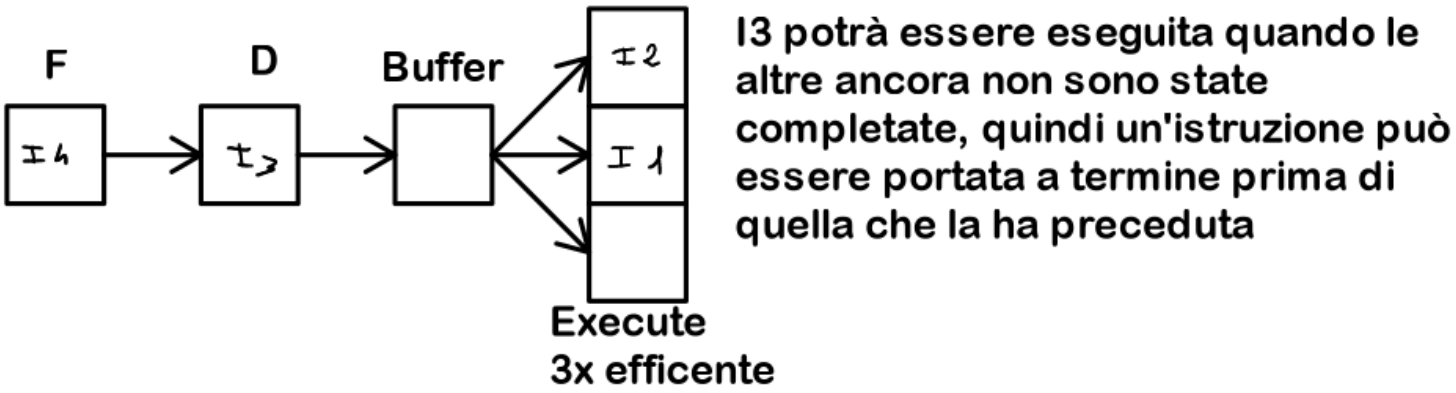
- Memory Address registry (MAR): contengono l'indirizzo di memoria da modificare;
- Memory Data registry (MDR): contengono il valore da inserire in memoria;
- Command registry: contengono il comando da far eseguire alla ALU.

Control Unit

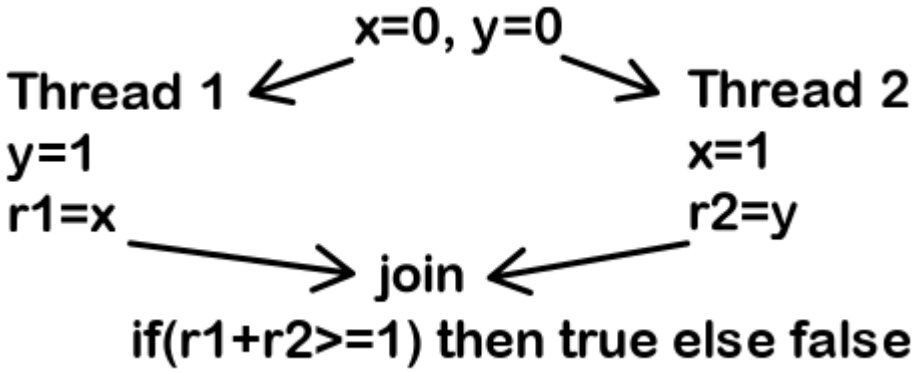
- Fetch Unit: recupera l'istruzione dalla memoria
- Decode: decodifica l'istruzione
- Execute: fa eseguire l'istruzione alla ALU
- Program Counter: contiene l'indirizzo di memoria della prossima istruzione da eseguire
- Instruction Registry: contiene l'istruzione

Pipelining

Dato che è sempre più difficile realizzare transistor piccoli, per migliorare le prestazioni di un computer si sfrutta la multiprogrammazione per eseguire parti diverse delle istruzioni contemporaneamente anziché una dopo l'altra come nel classico modo sequenziale. I computer moderni hanno 7/8 pipeline, il che li rende 7 volte più veloci di un computer senza multiprogrammazione. Si evita che la velocità del pipelining sia limitata da quella del processo più lento aumentando il numero di execute unit. Si definisce "superscalare" un processore in grado di avviare in esecuzione più di una istruzione in un singolo ciclo di clock.



Due istruzioni possono essere mandate in esecuzione contemporaneamente nella pipeline quando non c'è rischio che un istruzione B basata su A non venga portata a termine prima di A. Bisogna fare un grafo delle dipendenze e vedere le relazioni di dipendenza delle istruzioni. Il processore richiede tempo per fare questo calcolo e quindi l'efficenza guadagnata non è direttamente proporzionale al

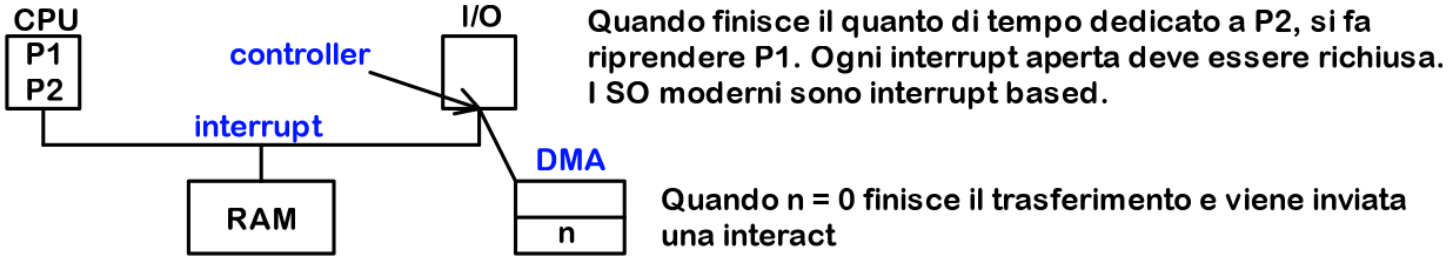


numero di esecutori.
In entrambi i casi la seconda istruzione è eseguita dopo la prima, ma se così non fosse (nel caso fossero indipendenti e quindi eseguite in parallelo), allora r1+r2=0. L'indipendenza tra le due istruzioni

deve dunque essere comunicata. I compiler devono capire dove le istruzioni possono essere eseguite in parallelo e in che ordine.

Interrupt

Le interrupt sono usate per assicurarsi che a un certo punto un'attività sia terminata e dunque un altro processore possa riprendere. Una volta il processore interrogava a intervalli regolari le periferiche di I/O per vedere se avessero terminato il processo, a volte interrogando la periferica inutilmente. Gli interrupt trasferiscono il controllo al servizio interrupt attraverso il vettore di interrupt, che contiene un indirizzo di memoria per ogni procedura di interrupt. Una procedura (o codice) di interrupt modifica lo stato di interruzione di un processo.



Direct Memory Access (DMA)

Sono usate per dispositivi di I/O ad alta velocità. Il DMA permette ai dispositivi di trasferire dati da e verso la memoria senza coinvolgere attivamente la CPU, il che porta a una maggiore efficienza e velocità complessiva del sistema. Funziona assegnando ai dispositivi periferici (I/O) il controllo temporaneo del bus di sistema per trasferire dati direttamente nella memoria principale.

Gerarchia di Memoria

- La cache è una piccola memoria molto veloce che si trova direttamente sui chip della CPU o vicino ad essa. È divisa in più livelli (L1, L2, L3) in base alla vicinanza alla CPU, con L1 che è la più vicina e più veloce. La cache è utilizzata per memorizzare temporaneamente dati e istruzioni che la CPU utilizza frequentemente, riducendo il tempo di accesso ai dati e migliorando le prestazioni complessive del sistema.
- La RAM è una memoria a accesso casuale in cui la CPU può leggere e scrivere dati in modo casuale. È più grande e più lenta della cache, ma è ancora molto più veloce dello storage di massa. Funge da ponte tra la cache e lo storage di massa, fornendo una quantità maggiore di spazio di archiviazione temporanea per dati e programmi. La RAM è volatile, il che significa che i dati vengono persi quando l'alimentazione viene spenta, quindi è utilizzata principalmente per memorizzare dati temporanei mentre il sistema è acceso.
- Lo storage di massa è costituito da dispositivi come hard disk e unità a stato solido. È il livello più grande e più lento della gerarchia della memoria, ma offre una capacità di archiviazione molto maggiore rispetto alla RAM. I dati vengono memorizzati permanentemente nello storage di massa, anche quando il sistema è spento. Lo storage di massa è utilizzato per archiviare il sistema operativo, i programmi e i dati a lungo termine.

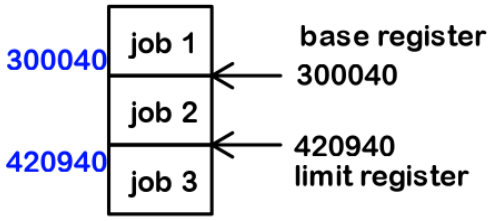
Principio di località

È la tendenza dei programmi a accedere ripetutamente a un insieme relativamente piccolo di risorse o a un'area limitata della memoria durante un breve periodo di tempo. Località temporale: si verifica quando una risorsa viene acceduta più volte nel breve termine. Località spaziale: si verifica quando una risorsa è vicina ad altre risorse precedentemente accedute.

Protezione Hardware

Parti di hardware sono state introdotte per venire contro alle esigenze del SO, quali:

- Operazione dual mode: differenziazione tra modalità utente e modalità monitor (o SO). L'utente potrà dialogare con I/O tramite system call fornite dal SO. Solo il driver del SO può dialogare direttamente con le periferiche. Viene usato un "mode bit" per distinguere tra utente (1) e monitor (0). Se un'istruzione privilegiata viene eseguita e il bit è 1, si fa partire l'interrupt.
- Protezione memoria: fornire protezione di memora per almeno il vettore di interrupt e le routine di interrupt dei servizi. Vengono quindi aggiunti 2 registri:
  - base register: contiene l'indirizzo di memoria più piccolo
  - limit register: contiene la dimensione dell'intervallo



Se job 2 modifica il contenuto di una cella, vogliamo essere sicuri che l'indirizzo sia tra 300040 e 420939,altrimenti prima di accedere alla memoria centrale bisogna aggiungere due istruzioni di controllo, che è tempo in più per eseguire un istruzione molto comune e che deve essere molto veloce. Per questo si agisce a livello di hardware.

Si aggiunge un circuito integrato nelle vicinanze del processore in cui tutte le volte viene prodotto un indirizzo (il processore interagisce con la memoria centrale).- CPU protection: si utilizza un timer che interrompe il computer dopo un certo periodo di tempo per assicurarsi che il sistema operativo mantenga il controllo. Il timer è solitamente usato per implementare il time sharing. per ridurre il tempo di esecuzione di questi processi ci si aiuta con l'hardware.