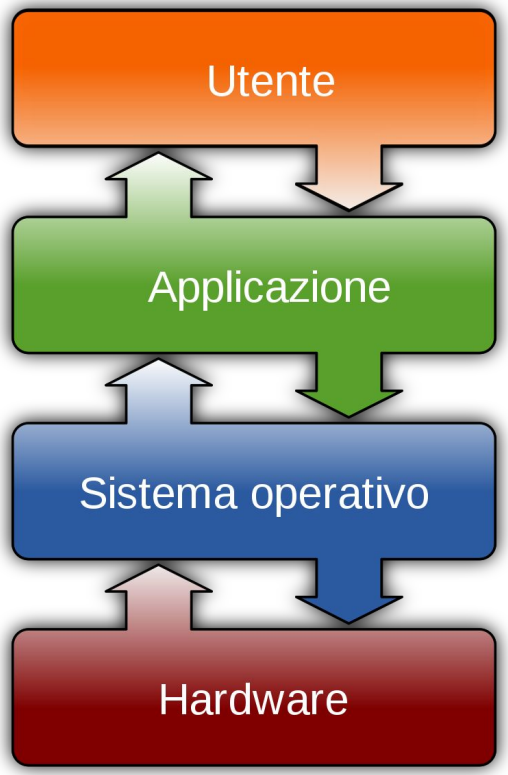
Architettura dei sistemi di elaborazione

Sistema operativo

Insieme di programmi che vengono caricati in memoria centrale all'avvio del sistema (boot).

Ha il compito di ottimizzare l'uso delle risorse del sistema, prendendone il controllo e assegnandole ai processi.

Funge da interfaccia tra l'utente e la macchina fisica.



Cosa è un processo?

Il programma è un'entità passiva: un insieme di byte contenente la codifica binaria delle istruzioni da eseguire.

Il processo è un'entità attiva, l'istanza di un programma in esecuzione.

Un processo è un programma in esecuzione.

Ogni attività all'interno del S.O. è rappresentata da un processo, che quindi è l'unità di lavoro all'interno del sistema.

Cosa è una risorsa?

I processi per avanzare hanno bisogno di utilizzare la CPU, la memoria RAM, i dispositivi di ingresso e uscita.

Possono anche avere bisogno di dati generati da altri processi.

Cosa è una risorsa?

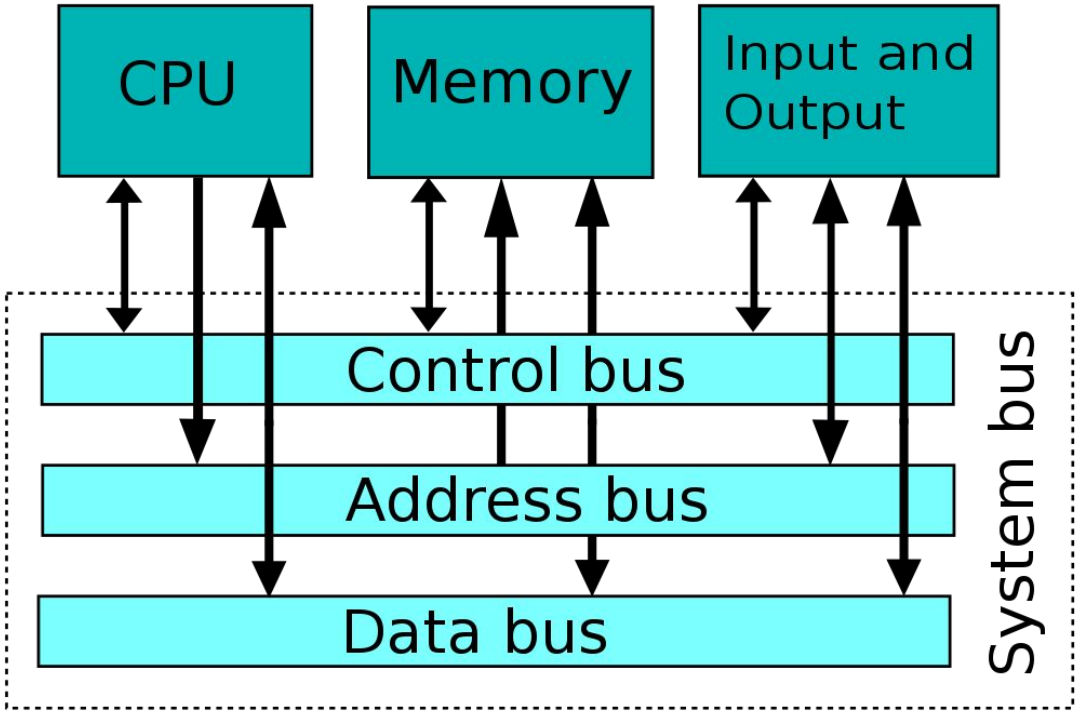
Una risorsa è una entità h/w o s/w necessaria all'avanzamento di un processo.

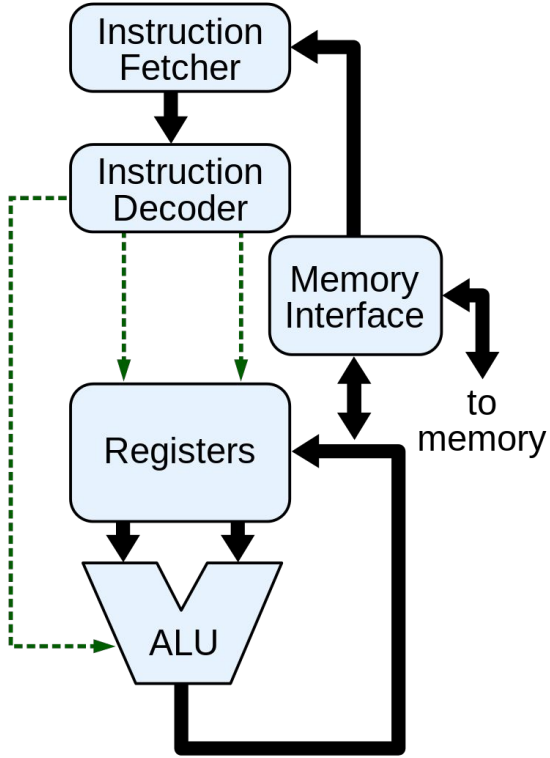
Le risorse h/w sono il processore, la memoria centrale e tutte le periferiche di I/O.

Le risorse s/w sono ad esempio dati generati da altri processi.

Servizi forniti dal S.O.

- Sviluppo di software: editor, compilatori, debugger, ecc.;
- Esecuzione di programmi;
- Accesso ai dispositivi di I/O;
- Accesso protetto alle risorse condivise;
- Gestione degli errori e dei malfunzionamenti;
- Gestione di statistiche e raccolta dati per addebiti agli utenti e per monitorare le prestazioni.





Di File:CPU block diagram.png: R. S. ShawSVG conversion by: Booyabazooka - Questo file deriva da: CPU block diagram.png, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=1362908

Protezione

Multiprogrammazione e multiutenza rendono necessari alcuni meccanismi HW per esercitare la protezione sulle risorse.

Le risorse allocate a programmi/utenti devono essere protette nei confronti di accessi illeciti di altri programmi/utenti.

Protezione

L’architettura h/w prevede un bit di modo (kernel: 0, user: 1).

Al verificarsi di un’interruzione, l’hardware commuta al kernel mode.

I programmi utente vengono eseguiti in user mode, il sistema operativo in kernel mode (supervisor, monitor mode)

Protezione

Le istruzioni privilegiate (operazioni di i/o, operazioni di gestione della memoria, ecc.) possono essere eseguite soltanto se il sistema si trova in kernel mode.

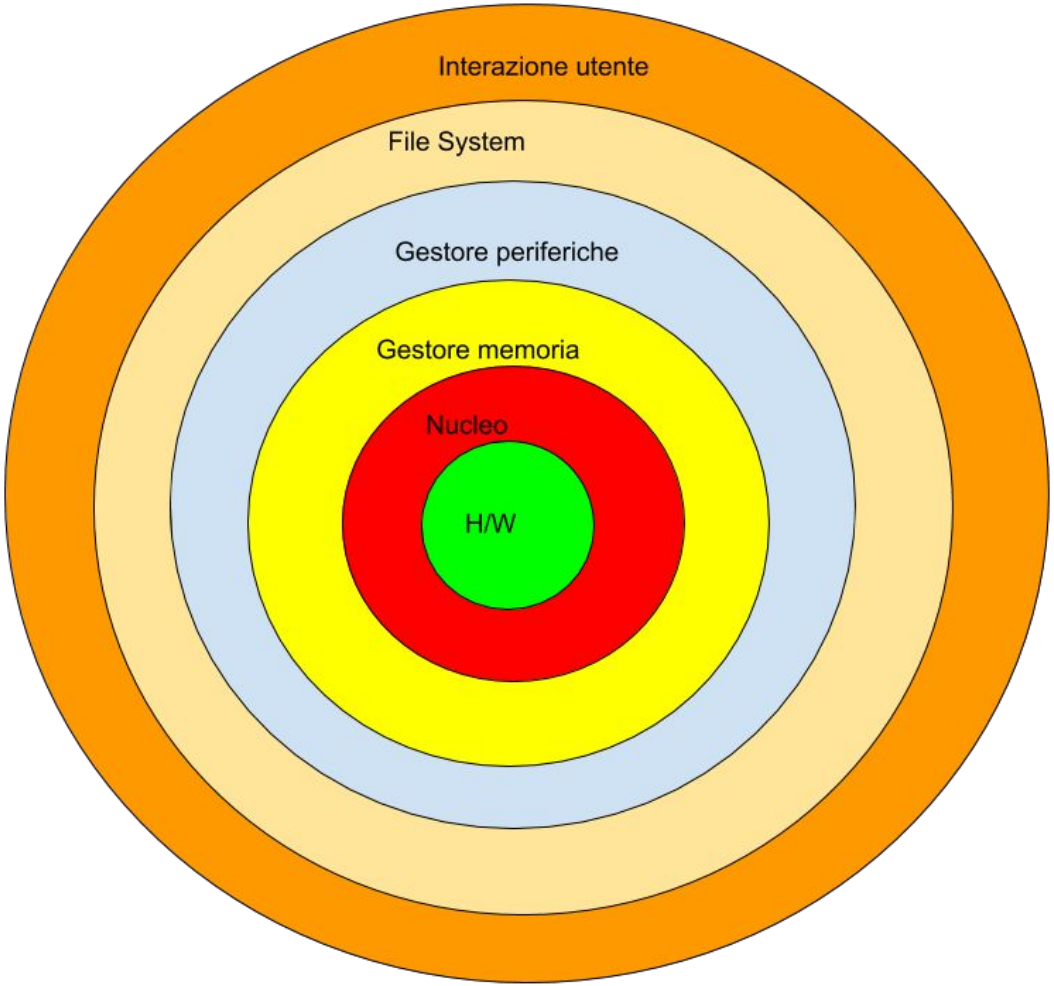
I programmi utente vengono eseguiti in user mode, se un programma utente tenta l'esecuzione di una istruzione privilegiata, viene generata una trap.

Per richiedere l'esecuzione di istruzioni i programmi utenti devono effettuare una "supervisor call" tramite l'invio di un'interruzione software al S.O.

Protezione

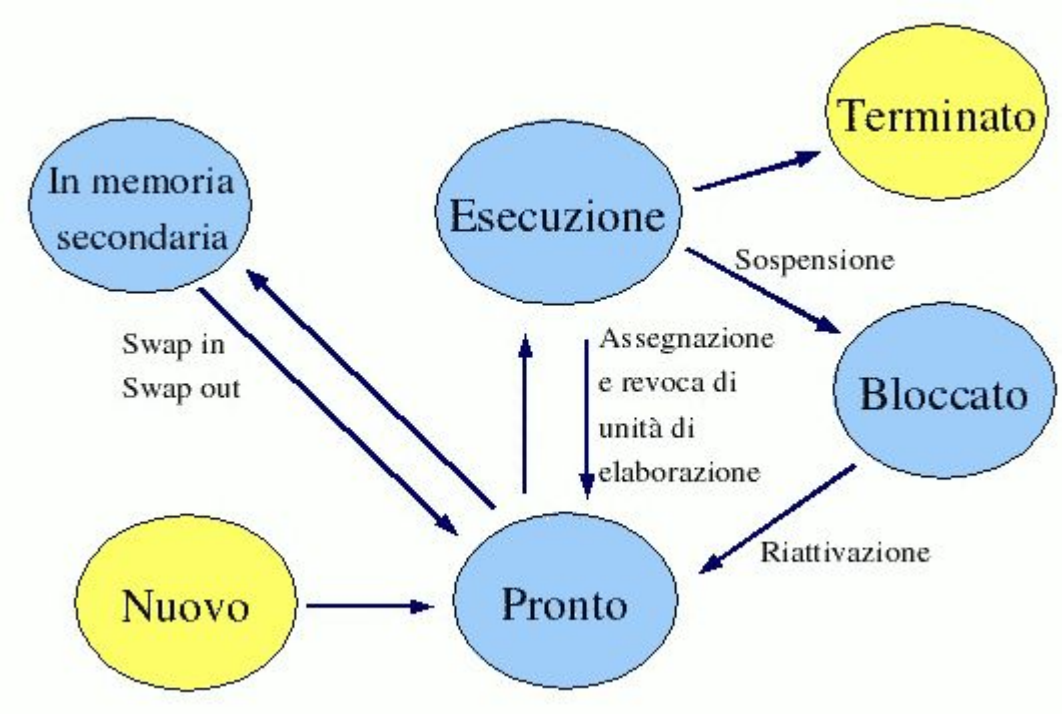
Questa comporta il salvataggio dello stato (PC, registri, bit di stato, etc.) del programma chiamante e trasferimento del controllo al S.O. Il S.O. esegue in modo kernel l'operazione richiesta (gestione dell'interruzione) e al termine dell'operazione, il controllo ritorna al programma chiamante (ritorno al modo user).

La comunicazione tra il programma chiamante ed il sistema operativo avviene mediante i parametri della system call, normalmente passati tramite registri e/o blocchi di memoria indirizzati da registri.


Interazione utente
File System
Gestore periferiche
Gestore memoria
Nucleo
H/W

Funzioni del nucleo

- I/O a basso livello (insieme delle RSI dei diversi dispositivi e SVC)
- Sincronizzazione tra i processi
- Gestione della risorsa processore



GFDL con disclaimer, https://it.wikipedia.org/w/index.php?curid=91809

Scheduler e Dispatcher

Il dispatcher (scheduler basso livello) ha il compito di estrarre tra i processi nella Ready List quello da far entrare in esecuzione, quindi il dispatcher commuta il contesto.

Lo scheduler ad alto livello modifica le priorità dei processi pronti e in attesa. Lo scheduler ed il dispatcher interagiscono tra loro per ottenere il risultato voluto secondo la politica adottata.

Il lavoro del dispatcher è strettamente correlato al meccanismo di gestione delle interruzioni.

Scheduler e Dispatcher

In un sistema monotasking, la routine di servizio di una interruzione (RSI) esegue il servizio richiesto (ad es. acquisisce un carattere), rimuove la causa che ha generato la richiesta di interruzione, esegue l'istruzione di ritorno da interrupt, che ripristina il contesto del programma che era stato interrotto, il quale può quindi proseguire la sua esecuzione.

Scheduler e Dispatcher

In un sistema multitasking il servizio di una interruzione può comportare invece la modifica dello stato di un processo (ad es. da attesa a pronto); le RSI terminano cedendo il controllo allo scheduler (anziché al processo interrotto); lo scheduler decide (in base alla politica adottata) se far proseguire il processo che era stato interrotto, o se portare in esecuzione un altro processo (scelto tra quelli pronti, ponendo nello stato pronto quello interrotto).

Process Control Block

- nome e/o id del processo
- stato del processo (attesa, pronto, ecc.)
- contesto h/w del processo (registri del processore)
- altre informazioni per la gestione del processo (risorse possedute, informazioni per la gestione della memoria, ecc.)
- link a altri PCB

Scheduling dei processi

I processi possono essere catalogati come CPU-bound se fanno un uso intensivo della CPU e poche operazioni di I/O, I/O bound se fanno molte operazioni di I/O ed utilizzano poco il processore. Spesso i processi alternano comportamenti I/O-bound e CPU-bound.

Scheduling dei processi

Lo scheduling può essere di tipo:

- non-preemptive o cooperativo: il processo in esecuzione rilascia volontariamente il processore (quando termina, o quando si blocca in attesa di I/O);
- preemptive: il processo in esecuzione è forzato a rilasciare il processore (e messo nella Ready List) a vantaggio di un altro processo; così si evita che un processo di tipo CPU-bound monopolizzi il processore per tempi troppo lunghi.

Scheduling dei processi

L’overhead del SO è la percentuale di tempo di CPU usato dal SO per svolgere le sue funzioni; un overhead troppo alto vanifica uno degli obiettivi principali dei SO: l’efficienza nell’uso del processore. Le operazioni di scheduling, a causa delle commutazioni di contesto, possono comportare overhead eccessivi.

Scheduling dei processi

Le diverse politiche dello scheduler corrispondono a diversi obiettivi (indici di prestazione), spesso contrastanti tra loro, ad esempio:

- massima utilizzazione del processore;
- massimo throughput (TH) = n. di processi/tempo;
- minimo turnaround time (TT) = tempo totale per terminare un processo (tempi di attesa + t. di esecuzione);
- …

Politiche di scheduling

- FCFS - First Come First Served (non preemptive).
- RR- Round Robin (preemptive)
- PS - Priority Scheduling
- MF – Multilevel Feedback Scheduling

Gestore della memoria

La gestione della memoria (effettuata dal SO usando gli accorgimenti presenti nell'hardware), consiste nel suddividere la memoria per consentire la presenza di più processi (presupposto indispensabile per il multitasking), cercando di per farci stare il numero maggiore possibile di processi (quanto maggiore è questo numero, tanto minore è la probabilità che la ready list sia vuota).

Gestore della memoria

Deve garantire:

Protezione: è necessario impedire che un processo acceda a locazioni di memoria di un altro processo (o del SO). La verifica non può essere fatta dal compilatore perché molti indirizzi vengono calcolati al momento della esecuzione (ad es. i metodi di indirizzamento per accedere agli elementi di un vettore) e i processi possono essere rilocati da una posizione di memoria ad un'altra. La verifica quindi va fatta al momento della esecuzione, utilizzando funzionalità che devono essere messe a disposizione dell'hardware.

Gestore della memoria

Trasparenza: l'operazione di allocazione deve essere invisibile al processo, in particolare il programmatore deve poter scrivere il codice senza preoccuparsi di come il programma verrà caricato in memoria.

Allocazione logica: le sezioni di memoria (codice e dati) assegnate a un programma devono essere viste da quest'ultimo come fisicamente contigue, anche qualora fossero poste in aree disgiunte di memoria.

Gestore della memoria

Condivisione: necessaria per consentire la condivisione di dati e di codice. La condivisione di dati è necessaria nel caso di processi cooperanti che fanno parte del medesimo programma ed hanno bisogno di dati comuni.

La condivisione di codice quando un insieme di subroutine deve essere utilizzato da più processi. In questo caso infatti conviene che di esse vi sia in memoria un'unica copia accessibile a tutti, piuttosto che averne tante copie replicate per ciascun processo.

Questo tipo di condivisione è possibile solo se il codice delle subroutine è rientrante, cioè se tutte le variabili da esse utilizzate sono memorizzate nello stack.

Memoria virtuale paginata

Ogni processo lavora in un proprio spazio di memoria virtuale, le cui dimensioni possono essere superiori alla dimensione della memoria fisica, completamente inaccessibile agli altri processi; anche il Sistema Operativo lavora in un proprio spazio di memoria virtuale.

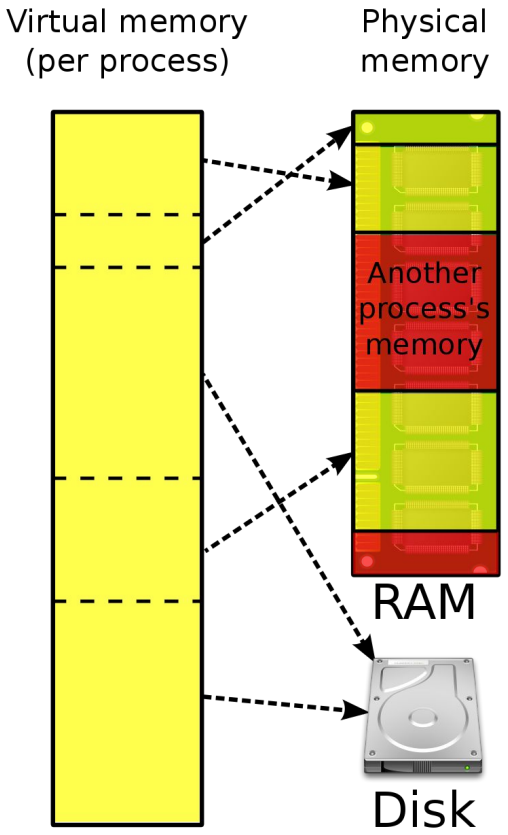
Memoria virtuale paginata

La memoria fisica viene divisa in blocchi uguali, detti frame (o pagine fisiche), di dimensioni spesso piccole (es: 4KB) e uguali alla dimensione di un blocco del disco.

Anche lo spazio di memoria indirizzato da ciascun processo (memoria virtuale) è diviso in blocchi delle medesime dimensioni, detti pagine (o pagine logiche o pagine virtuali)

Memoria virtuale paginata

Nella memoria fisica sono mappate solo alcune delle pagine di memoria virtuale di ciascun processo (resident set del processo stesso), le altre rimangono nella memoria secondaria e vengono prelevate quando necessario.



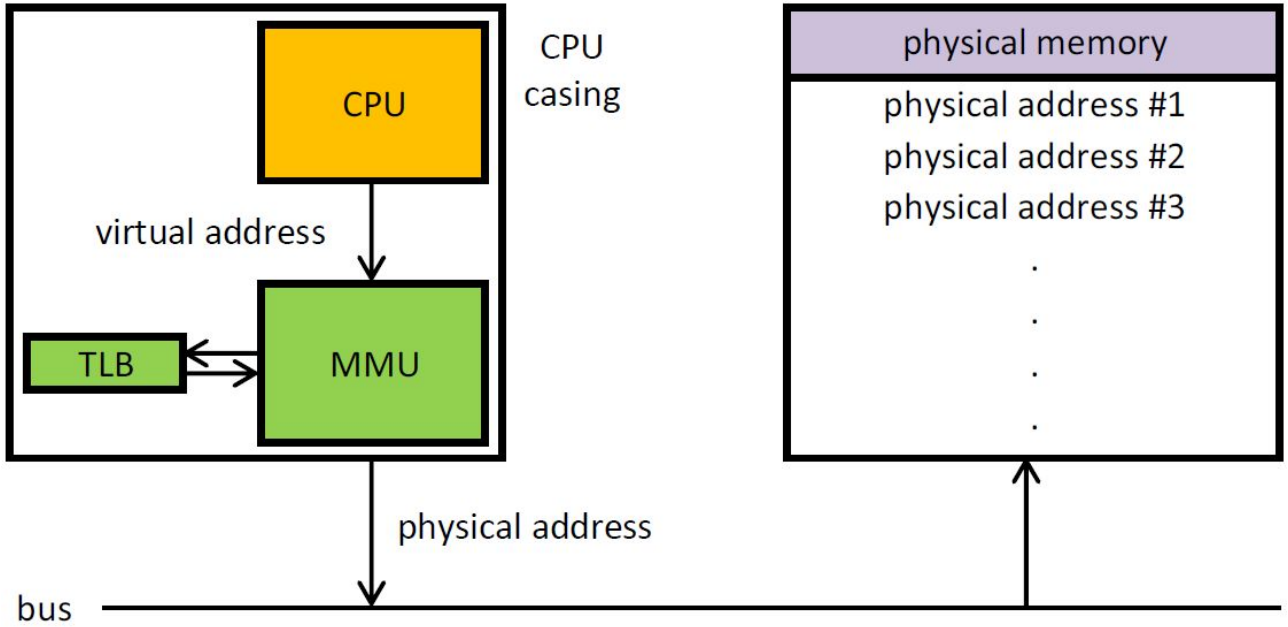
Memoria virtuale paginata

- Ogni processo tiene in memoria solo parte del suo codice e dei suoi dati;
- ogni processo può indirizzare più memoria di quella fisica disponibile senza la necessità di utilizzare tecniche di overlay;
- alcune pagine di memoria, assegnate al S.O., contengono codice condiviso tra i processi (es. DLL)

Memoria virtuale paginata

Sia gli indirizzi fisici che quelli virtuali sono divisi in numero pagina + offset.

Il MMU determina la corrispondenza tra pagina virtuale e pagina fisica



CPU: Central Processing Unit
MMU: Memory Management Unit
TLB: Translation lookaside buffer

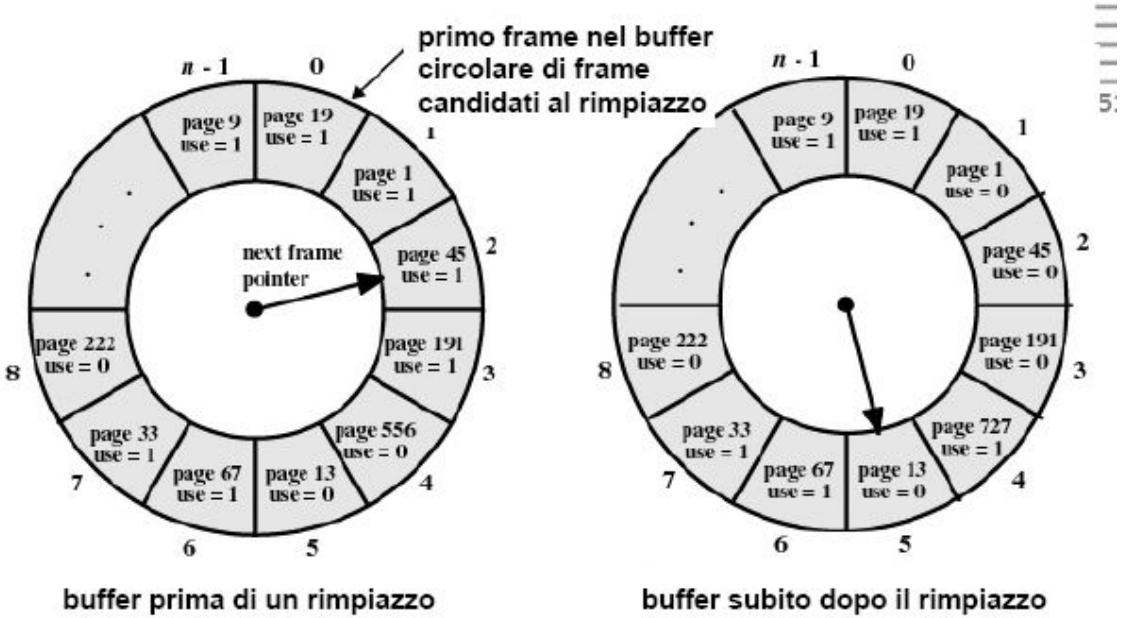
Politica di fetching

- demand paging: la pagina viene portata in memoria solo quando il processore chiede di accedere ad un suo indirizzo, provoca molti page fault all'avvio di ogni processo;
- prepaging: oltre alla pagina richiesta, si portano in memoria anche le pagine che risiedono su settori adiacenti del disco. È più efficiente trasferire da disco più settori/pagine consecutivi piuttosto che un settore alla volta (per via dei tempi di seek e di rotazione), ma c'è il rischio di portare in memoria pagine che non servono.

Politica di sostituzione

- Algoritmo ottimo
- Algoritmo LRU (Least Recently Used)
- Algoritmo FIFO (First In First Out)
- Algoritmo tipo clock

Algoritmo “clock”



buffer prima di un rimpiazzo

buffer subito dopo il rimpiazzo