# Sistemi

Andrea

November 29, 2024

## Contents



## 1 paginazione

La paginazione divide spazio logico e spazio fisico in aree predefinite. Page table svolge il ruolo di mappa. (binding runtime). Paginazione favorisce la condivisione di codice tra processi.

## 2 segmentazione

Lo spazio indirizzi è diviso in segmenti di **dimensione variabile**, che rispecchiano meglio la semantica del codice. Il compilatore produce i segmenti e questi vengono gestiti separatamente dal sistema operativo, allocati in parti non contigue della memoria, che quindi non è divisa a priori in frame, ma di volta in volta si cerca una porzione di memoria abbastanza capiente (come partizionamento dinamico con allocazione continua). La cpu genera un indirizzo logico già diviso in due parti (e non successivamente diviso dal S.O. come paginazione), queste due parti sono:

- segment number
- offset

Ogni segmento è mappato nella semgment table, dove leggiamo l'indirizzo fisico base e la lunghezza del segmento stesso. Ci sono due registri

- STBR: punta all'inizio della tabella dei segmenti

- STLR: indica numero dei segmenti usati dal programma

Binding avviene a runtime, possibilità di rilocare i segmenti, possibile condividere accesso ai segmenti. SOno presenti dei bit di validità e dei bit di accesso al segmento. I segmenti possono variare di lunghezza, problema gestione dinamica della memoria.

## 2.1 implementazione page table - + livelli e tlb

COme viene implementata la page table o la segment table? sono strutture dati potenzialmente pesanti. Struttura dati che viene acceduta molte volte, idealmente dovrebbero prevedere un accesso rapido, in registri veloci (di conseguenza). Tuttavia le dimensioni rendono difficile se non impossibile la possibilità di posizionarla in registri MMU veloci. Es:
Nel caso si parli di indirizzi virtuali da 32 bit (tra p+d), con pagina logiche da $2^12$ Byte = 4KB. Di conseguenza abbiamo $\frac{2^32}{2^12} = 2^20$ pagine, (indirizzo 32 bit = p 20 bit + d 12 bit)
Quindi la page table ha $2^20$ entry, in ogni entry c'è almeno il numero frame (oltre a qualche possibile bit), dati $2^32$ frame possibili, abbiamo 4MB di spazio occupato dalla page Table (i conti non tornano cit. prof).
Di conseguenza la page table sta in memoria, puntata da due registri (slide).
Per velocizzare affianco un caching, memoria dedicata a aparte che preserva alcune entry della page table di alcune pagine, per ridurre ovvero TLB (mem supplementare ad accesso rapido).
Cpu genera indirizzo, vedo se page ha entry in tlb (tlb hit) ho fatto, con costo base ridotto e rapido
Tlb-miss, non c'è in entry, devo accedere alla page table in memoria principale, doppio accesso alla memoria principale (+ accesso tlb fallimentare, ma poco pesante) Page-fault, ovvero pagina è in memoria secondaria, che andrà prtata in memoria principale, con ulteriorie costo.
Quali pagine nel tlb? ci sono algoritmi appositi.
costi: (slide) NOTA: 1.04 miglioramento rispetto a 2(non scritto) senza tlb
Tabella delle pagine presenta problema, molto ingombrante. Bisogna paginare a sua volte la page table, per ridurre occupazione. Rende necessaria una **tabella delle pagine esterna**. La cpu genera indirizzo, lo spezzo in due, offset d e poi pagina p. La parte p (numero di pagina) la devo dividere in due parti per recuperare in due parti, per capire dove recuperare la page table. per la paginazione a più livelli si può usare sempre la formula, con tlb contenenti pagine finali.

## 2.2 Tecnica della page table invertita

Uso la pag table per registrare nelle entry non le pagine ma bensì i frame, inverte ruolo pagine fisiche e logiche. Riduce spazio occupato in memoria per page table, in quanto spazio virtuale molto più grande. Lo svataggio sono i tempi accesso, in quanto non accedo più immediatamente alla pagina, il frame è l'indice, la ricerca è molto più costosa, indirizzo ti dà la pagina logica, devo

quindi cercare l'indirizzo della pagine logica al'interlo della page table, non ho la posizione!! (ndo cazzo sta? cerco). Tabelle di Hash.

## 2.3 frammentazione esterna

paginare i segmenti, combinando approccio,

## 2.4 memoria virtuale

memoria virtuale: separazione della memotria logica (slide) spazio logico può essere molto più grande della memoria fisica e riuscire ad eseguirlo, posso aumentare il grado di multiprogrammazione caricando soltando parzialmente in memoria principale, ciò comporta che parte della memoria logica viene tenuta in memoria secondaria.
Si carica una pagina in memoria quando necessario. Quando una pagina viene richiesta, quando cpu si riferisce a un indirizzo riferita a questa pagina. Se pagina non si trova in memoria principale va caricata, ergo necessario liberare il frame in caso di memoria piena. Operazione di page-in e page-out, paging in generale tra memoria principale e secondaria. In realtà paging e swapping sono due tecniche diverse swapping si riferisce a carico/scarico interi processi, paging solo di pagine, pager e swapper. Delle volte però si usano entrambe i termini con poca distanzione.
All'interno della page table abbiamo bit di validità, che indica se la pagine è caricata in memoria principale o meno. La prima volta la pagina sarà sicuramente in memoria secondaria (metodo su richiesta). Avviene un page fault e quindi si carica la memoria, liberando un frame scegliendo la pagina vittima. Trovato il frame libero, caricata la pagina, il sistema Operativo si occupa di aggiornare la page table, modificando il corrispondente bit di validità. calcolo dei tempi slide. Dobbiamo conoscere percentuale page fault, dipende dall'algoritmo di scelta della pagina vittima. (slide)