23-05-2023

Tabella dei frame

Traccia lo stato di occupazione di ogni frame. Ogni frame ha le seguenti informazioni:

- ***occupato/libero***: se è assegnato ad una pagina o no.
- se è assegnata ad una pagina, allora vuole sapere **l'informazione che contiene** e da ***quale processo*** è occupata

La tabella dei frame *viene consultata*:

- ogni volta che si **crea un nuovo processo** per creare la relativa tabella delle pagine di quel processo
  - L'associazione pagina->frame avviene solo quando una pagina viene chiamata in causa per la prima volta e quindi si ha page fault quando **avvio per la prima volta un programma** perchè il numero di quel programma si trova in memoria (*perchè aperto in quel preciso momento*).
  - *Per questo motivo i page fault non si possono annullare*
- ogni volta che un processo chiede di **allocare nuove pagine** che non sono caricati e non sono associati dei frame alle pagine richieste

> *Se ho un sistema di **più processi** e **più utenti** e creo page fault (pagina non associata ad alcun frame) allora bisogna scegliere una pagina da togliere per associrane una nuova: qual è la pagina che tolgo ad un processo?*
> *Bisogna fare delle valutazioni e rimuovere una determinata pagina e non una a caso.*

Progettazione di una tabella delle pagine

La tabella delle pagina è singola per ogni processo. Per tale motivo le dimensioni occupate in memoria aumentano e diminuisce velocità:

- Il mappaggio deve avvenire il più **velocemente possibile**
- lo **spazio degli indirizzi virtuali** (tabella delle pagine) è **grande** se il mappaggio è grande

> Si ha **accesso costante alla memoria** sia nella traduzione per accedere all'istruzione. Se l'accesso alla memoria diventa **lento**, questo potrebbe **rallentare le prestazioni**
> Più **è grande** la **tabella** delle pagine, più diventa delicata e pesante la gestione delle tabelle

Si ha **una sola tabella delle pagine formata da un array di registri**. I regsitri sono parti di memoria hardware e quindi sono veloci.

- Si ha un array di registri hardware con una sola voce per ogni pagina virtuale
- All'avvio del processo, il SO carica i registri con la tabella delle pagine del processo interessato e ,durante l'esecuzione, non è necessario accedere alla memoria della tabella ma vengono usati i registri
  - +E' semplice e si usa un array di registri (*evita l'accesso alla memoria*)
  - +Si guadagna in termini di velocità
  - -Il cambio di contesto (se la tabella è **MOLTO ESTESA**) diventa lento

Un alternativa a ciò è: si tiene una **tabella interamente caricata in memoria** e si ha bisogno di un *registro che punta all'inizio della tabella delle pagine*, detto **PTBR** (*Page Table Base Register*).

Differenze

- Con ***array di regsitri***: veloce e si evitano accessi alla memoria ma risulta dispendiosa (cambi di contesto) se la tabella è estesa
- con la tabella ***interamente caricata in memoria***: usa un solo registro (più veloce in caso di context switch) ma si devono fare gli accessi alla memoria (*almeno 2 volte -> **uno** per consultare la tabella delle pagine e **uno** per recuperare le informazioni necessarie*)

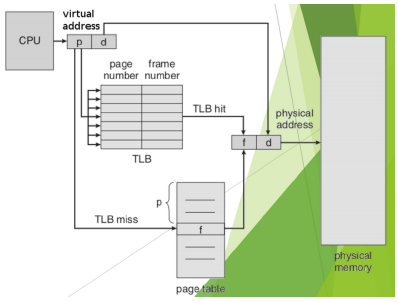
> *Ogni accesso alla memoria aumenta il rallentamento del sistema a livello complessivo*

Alternativa basata su un'osservazione: *i programmi in esecuzione fanno molti riferimenti a un **numero piccolo di pagine associate***, quindi vuol dire che accedono a un numero ristretto di voci nella tabella delle pagine e quindi vengono usate solo determinate voci nella tabella.

> *Ogni programma fa uso di un gran numero di riferimenti a piccolo numero di pagine, quindi non a tutte le sue pagine associate. Vuol dire che delle voci vengono frequentemente chiamate in causa mentre le altre no, o comunque poco usate.*

Le pagine più usate vengono ***messe in evidenza***:

- Si usa un apposito dispositivo (nella *MMU*) che mantiene *le pagine più frequentemente chiamate in causa*
- Il dispositivo è detto **MEMORIA ASSOCIATIVA** (*Translation Lookaside Buffer TLB*) ed è qui che avverranno le ricerche delle pagine, piuttosto che cercare nella tabella delle pagine visto che è **MOLTO PROBABILE** che la pagina si trovi proprio qui. (il funzionamento è identico alla *CACHE*)



Le **voci** nella TLB sono:

- il **numero di pagina virtuale**
- bit di **validità**, se il processo ha la validità per usare quella pagina oppure no
- **numero di frame associato**
- il **dirty bit** (bit di modifica)
- **codice di protezione** (come si può accedere a quella pagina)

La traduzione avviene accedendo alla TLB.

- Se **presente** nella TLB (***TLB HIT***) allora determina il frame fisico e fa l'operazione.
- Se **non presente** (***TLB MISS***) allora si accede e si va a ricercare all'interno della memoria. MMU non ha trovato la pagina nel TLB. In questo caso si cerca nella memoria nella tabella delle pagine.
  - La pagina ricercata è stata richiesta recentemente quindi **con molta probabilità verrà richiesta** in futuro. Per questo motivo deve essere **registrata nella TLB *DOPO*** averla trovata nella tabella

delle pagine e ***PRIMA*** di restituire l'*indirizzo fisico*
    - Se la TLB non ha spazio, allora devo sacrificare una pagina da sostituire e in questo caso verranno usati algoritmi che identificano quelle usate **meno recentemente**
- Alcune voci della TLB possono essere ***VINCOLATE***: cioè *non possono essere scartate* in caso di *TLB-miss*
- Alcune TLB memorizzano address-space identifiers(ASID) e identificano in maniera univoca un processo. Se la TLB non li supporta, allora quando viene selezionata una nuova tabella delle pagine, la TLB viene **invalidata** cioè si fa il ***FLUSH DELLA TLB***, ovvero il proprio contenuto non è più valido e viene presa una considerata una nuova tabella di riferimento

> *La TLB deve mantenere le pagine frequentemente utilizzate in modo che l'MMU cerca qui dentro visto che si tratta di record molto minori rispetto alla tabella delle pagine. In questo caso si accelerano le operazioni.*

Vantaggi TLB

Si valuta il **grado di successo** della TLB (*TLB HIT*). Se è alto allora il ricavo è sostanziale.

Esempio:

- tempo di accesso alla memoria = 100 nsec;
- tempo di accesso alla TLB = 20 nsec

Il **tempo effettivo di accesso** è:

- 120nsec per *TLB HIT*, non devo accedere alla memoria per ricercare la pagina
- 220nsec per *TLB MISS*, si accede alla memoria per cercare nella tabella delle pagine

Se per ipotesi si ha una TLB ratio (percentuale di successi) dell'80%:

- tempo(medio) effettivo di accesso: $0.8 \times 120 + 0.2 \times 220 = 140nsec$, ho 40nsec di ritardo (*rispetto al tempo di accesso alla memoria*)

In generale: (***ESAME***)

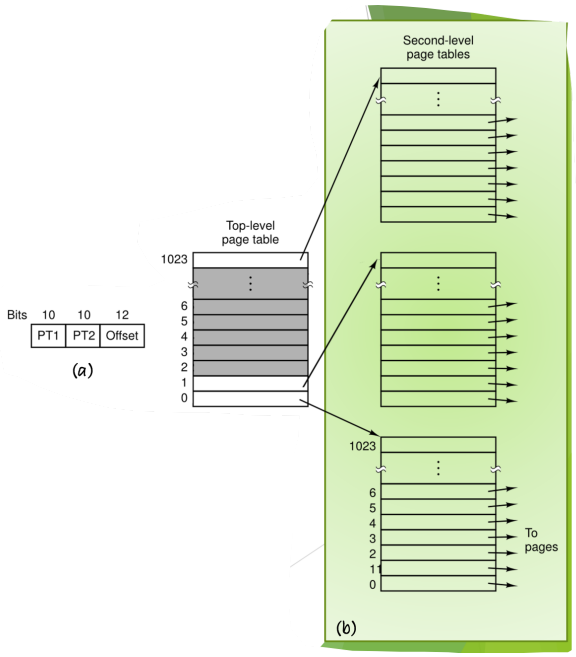
- tempo di accesso alla memoria: α
- tempo di accesso alla TLB: β
- TLB ratio: ε
    - $EAT = \varepsilon(\alpha + \beta) + (1 - \varepsilon)(2\alpha + \beta)$ -> ***Tempo di accesso effettivo medio***

Dimensioni

Come si gestisce lo spazio di indirizzi se questo è molto grande?

Si usa tabella delle pagine MULTILIVELLO: L'indirizzo virtuale (32 bit) è spezzato in **3 parti**:

1. PT1(10bit), PT2(10bit), Offset(12bit)
2. Ogni pagina è quindi da 4K
3. Vantaggi: si evita di mantenere tutte le tabelle delle pagine in memoria per tutto il tempo e ***si mantiene solo quello che è necessario in quel momento***.

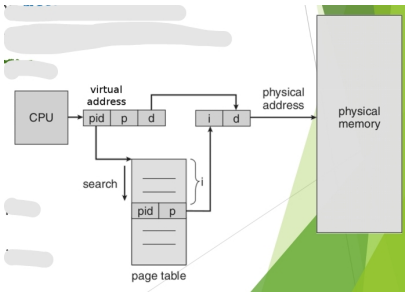


- La voce 0 punta a una tabella della pagine per il *testo del programma(?)*
- La vove 1 punta ai **dati del processo**
- L'ultima voce punta alla tabella delle pagine per lo **stack del processo**
- Tutte le altre voci (in grigio) sono **inutilizzate**

`PT1` = indice della tabella delle pagine di primo livello
`PT2` = indice della tabella delle pagine di secondo livello, è usato per cercare il numero del frame della pagina stessa

Se si usano molti più livelli allora potrebbero verificarsi dei rallentamenti e per tale motivo si non si deve esagerare.

Tabella delle pagine invertite



In questo apprccio: si mantiene una sola voce per frame fisico in memoria piuttosto che mannere una voce per pagina virtuale.
Ogni voce ha: indirizzo virtuale con le eventuali informazioni di quella pagina.

Ogni voce viene rappresentata da una coppia `(idprocesso, pagina_virtuale)` dove `pid` = identificativo dello spazio degli indirizzi.

> Quando si verifica un riferimento alla memoria, sarà rappresentato da (pid,npagina). Viene verificata e se c'è una corrispondenza, viene restituita la voce i-esima associata all'offset (`d`).
> Se non c'è corrispondenza, viene generato il *page fault*

Le tabelle invertite rappresentano dei limiti:

- Bisogna andare a ricercare si deve **scorrere l'intera tabella**, cioè svolgere un'**opezione di ricerca lenta**
- Si fa uso allora di una tabella HASH indicizzata sugli indirizzi virtuali.
  - consente di poter velocizzare la ricerca dei dati e si tampona sullo svantaggio della ricerca lenta
- Se si usa anche una TLB, si migliorano ancora di più le prestazioni

Aspetto fondamentale: cosa fare in caso di page fault?

Page fault

Vuol dire che la pagina non è all'interno della TLB. In questo caso si deve sostituire un frame associato a una pagina per associare la nuova pagina se la TLB è piena.
Quale pagina si rimuove? SI sceglie una ***VITTIMA*** da rimuovere per dare spazio alla nuova pagina

> L'**obiettivo**, comunque, è quello di minimizzare il numero di page fault in futuro
> *Basta capire e individuare quali sono le **pagine più utilizzate***.

Allora la soluzione ottimale e teorica è l'uso di un **algoritmo ottimale** (***OPT***):

- si deve scegliere la pagina da rimuovere che verrà referenziata in un furuto più lontano
- è **ottimale** ma **difficile da realizzare**
- dà comunque l'idea ottimale da seguire e un termine di paragone con le altre tipologie di algoritmi

Le due ***NOZIONI CHIAVE*** per gli algoritmi di sostituzione (*che hanno lo stesso obiettivo: scegliere la pagina che, con molta probabilità, evita i page fault nel breve futuro*) sono:

- la pagina da rimuovere è quella che ***non è stata usata recentemente***
- **fra 2 pagine** non usate recentemente, sostituire quella ***più anziana*** fra le due (confrontando il "*quando è stata usata l'ultima volta*", quindi referenziata)

> *Come si identifica se è stata referenziata recentemente o no? Si usano i 2 bit:* `referenziamento` *e di* `modifica`