Tabella delle pagine

Ne abbiamo 1 per processo, ~~indicano la sua dimensione totale~~ 2^m

- Un spazio virtuale: 2^m $\Rightarrow$ Numero di pagine nella Tabella 2^{m-n}
- Un ~~totale~~ pagine: 2^n

offset: n bit meno significativi dell'indirizzo

- Spazio indirizzi fisici: 2^t $\rightarrow$ abbiamo 2^{t-n} frame

NB $m > t$ SEMPRE

Esercizio 1 RAM = ?

$VM = 4\,MB = 2^{22}$ byte (c.v. è a 22 bit)

$N.pv = 2^{m-n}$

$\exists\ 2^{13}$ voci nella tabella pagine

frame è a 8 bit

$2^{13} = \frac{2^{22}}{2^n}$

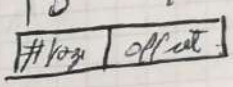
$2^n = \frac{2^{22}}{2^{13}} \rightarrow n = 9$

Il numero di frame è 2^8

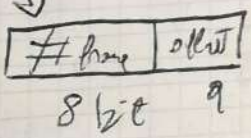
$RAM = 2^8 \cdot 2^9 = 128\,KB$ (bit pagg e frame coincidono)

Potevamo ragionare così

↳ MB = 2^{22} byte → i.v.



22 bit



? bit

8 + 9 = 17

i.l.

L'i.l. è a 17 bit

Quindi ... 2^{17} byte = 128 KB

Esercizio 2

VM = 1 GB

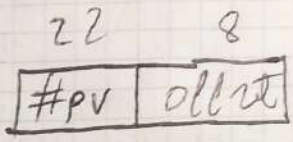
#pv a 22 bit

#i.l. a 20 bit

Quante frame?

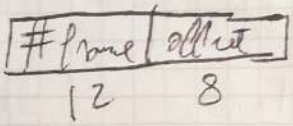
1 GB = 2^{30} byte 30 → m

i.v.



Ho 2^{12} frame

i.l.
20 bit



Voce di tabella

La tabella tiene anche altre informazioni oltre al numero di frame e il bit di presenza.

- Campo protezione: specifica cosa è consentito fare sulla pagina (lettura/scrittura e esecuzione)
- bit di modifica (dirty bit): è un flag che specifica se la pagina è stata modificata
  Per questo bit i va in contro alla politica della pagina ovvero sincronizzare le due cose in RAM e disco (se ci sono)
- bit di riferimento: è 0 se la pagina non è stata usata, 1 altrimenti.

E' previsto periodicamente l'azzeramento, è utile questo bit per gli algoritmi di sostituzione delle pagine. E' gestito dall'HW e sfruttato dal SO per l'azzeramento

- bit per disabilitare il meccanismo di una cache
  Viene disabilitato per le pagine che fanno mappatura per le porte di I/O
- bit di validità / allocazione
  Il SO tiene traccia quindi delle pagine allocate o meno, genera errori fatali nel caso si voglia accedere a pagine col bit disabilitato.

Tabella dei frame

Il SO tiene traccia dello stato di ogni frame tramite questa struttura dati.
Tiene due info per ogni frame
- Se è libero / occupato
- Occupato da chi?

La tabella è consultata in due momenti
- creazione nuovo processo (per creare la sua)
- processo chiede di allocare nuove pagine

Progettare una tabella delle pagine

Bisogna badare a < velocità / dimensione

1ª strategia
Avere un numero sufficiente di registri su cui mettere l'intera tabella. Se la tabella è grossa è impossibile.

2ª Caricare interamente in memoria la tabella e usare il PTBR.
Il PTBR è un registro speciale che punta alla tabella. Esso informa l'MMU su dove sta la tabella.
Il context switch diventa molto veloce e semplice: basta modificare PTBR

Svantaggio: Per prelevare un dato devo fare due accessi in memoria (1 accesso per consultare la tabella e 1 per consultare la word desiderata)
Il throughput di accesso alla memoria dimezza

Ottimizziamo la soluzione con la TLB
La TLB è un supporto HW che si comporta come una sorta di cache relativa alla tabella delle pagine e semplifica (se c'è hit) la traduzione
Essa sta dentro all'MMU, usa un numero ridotto di registri per tenere le seguenti voci

- #P.V.
- bit validità della voce in TLB
- codice protezione
- dirty bit
- #frame

Se la TLB contiene le info (TLB hit) usiamo la TLB per ottenere #frame in modo istantaneo e dovrà fare un solo accesso in memoria.
Altrimenti c'è TLB miss che ci porta a usare la page Table (2 accessi)
La TLB evita il 2° accesso in memoria molte volte

Un TLB miss provoca anche la scrittura della relativa voce nella TLB
La ricerca nella TLB è molto veloce essendo parallelizzata in HW

Anche qui ci sono algoritmi per scegliere/swappare pagine nella TLB

Nel caso di context switch poniamo o fare svuotamento (flush) ovvero il bit di validità diventa 0 in tutte le voci oppure:

- Usiamo ASID come campo in più nella voce come chiave con #PV ed evito svuotamento

ASID: address space id è un id dello spazio indirizzi associato alla pagina

Come spesso visto c'è la possibilità di avere

Voci vincolate

Voci ~~della pagina~~ della TLB "privilegiate" ovvero che non vengono buttate via dalla TLB durante gli algoritmi di mapping

- La TLB è una sorta di cache per le pagine della Tabella

Effective time access EAT

Esempio

tempo accesso memoria: 100 ns
tempo accesso TLB: 20 ns

Segue che il tempo effettivo di accesso varia
- 120 ns per TLB hit
- 220 ns per TLB miss

Avendo TLB ratio di 80% (percentuale successo)

$$t.m.a. = 0.8 \cdot 120\,ns + 0.2 \cdot 220\,ns = 140\,ns$$

In generale

tempo accesso memoria $= \alpha$
tempo accesso TLB $= \beta$
TLB ratio $= \varepsilon$

$$EAT = \varepsilon(\alpha+\beta) + (1-\varepsilon)(2\alpha+\beta)$$