Leggete il diario!

http://matteo.vaccari.name/so/diario.php

Dopo ogni lezione trovate:

- gli argomenti
- le sezioni del testo corrispondenti
- i lucidi
- documentazione addizionale
- esercizi

Tutte le letture riportate nel diario **sono parte del programma di esame**, a meno che non siano esplicitamente dichiarate *facoltative*

Problema: dimensione della page table

Tipica architettura a 32 bit, con pagine di 4K

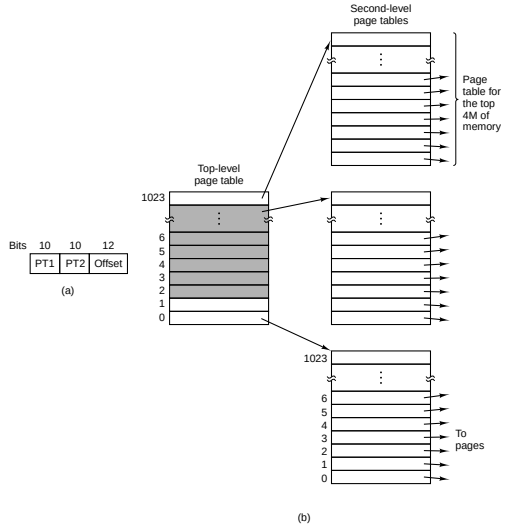
Spezziamo l'indirizzo virtuale in 20 (page index) + 12 (offset)

$\Rightarrow$ la page table ha 2^{20} righe

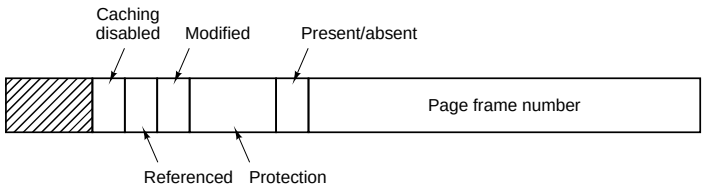
una riga occupa almeno 3 byte (# page frame + bits)

$\Rightarrow$ la page table occupa $3 * 2^{20}$ byte = 3MB!

Soluzione: multi-level page tables



Tipica riga di una page table



Problema: # accessi alla memoria

Gli accessi alla RAM sono costosi

Con mem virt paginata a 2 livelli, per un accesso rischio di farne 3

Soluzione: Translation Lookaside Buffer (TLB)

TLB

Il contenuto della TLB è relativo a *un* processo

Context switch fra processi $\Rightarrow$ svuoto (*flush*) la TLB

$\Rightarrow$ non conviene!

Ecco perché lo scheduler di Linux dà un bonus al processo corrente

Parentesi: ottimizzazione del TLB flush in Linux

Lo spazio kernel è mappato sugli stessi indirizzi in *tutti* i processi

I *kernel threads* sono processi "di servizio" che eseguono sempre in modo kernel

Se A, B sono processi normali e K_0, K_1 sono kernel threads

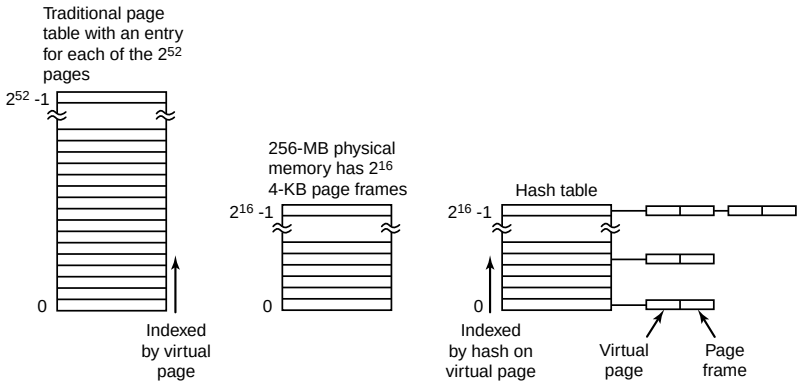
$A \rightarrow B$: flush (ovviamente)

$A \rightarrow ISR \rightarrow A$: no flush

$A \rightarrow K_0 \rightarrow K_1 \rightarrow A$: no flush

L'efficienza del SO dipende da tante piccole ottimizzazioni

Inverted page tables



Four times when OS involved with paging

- Process creation
  - create page table
- Process execution
  - MMU reset for new process
  - TLB flushed
- Page fault time
  - determine virtual address causing fault
  - swap target page out, needed page in
- Process termination time
  - release page table, pages

Page fault handling (i)

0. Hardware traps to kernel
1. General registers saved
2. OS determines which virtual page needed
3. OS checks validity of address, seeks page frame
4. If selected frame is dirty, write it to disk

Page fault handling (ii)

5. OS brings schedules new page in from disk
6. Page tables updated
7. Faulting instruction backed up to when it began
8. Faulting process scheduled
9. Registers restored
10. Program continues

Instruction backup

MOVE.L #6(A1), 2(A0)

Monitorare l'efficienza del MM

il comando /usr/bin/time(1)

il comando vmstat(1)