IF2230 Memory Management – Struktur Page Table

Operating Systems Concepts, 10th ed. Silberschatz, Galvin and Gagne 2018

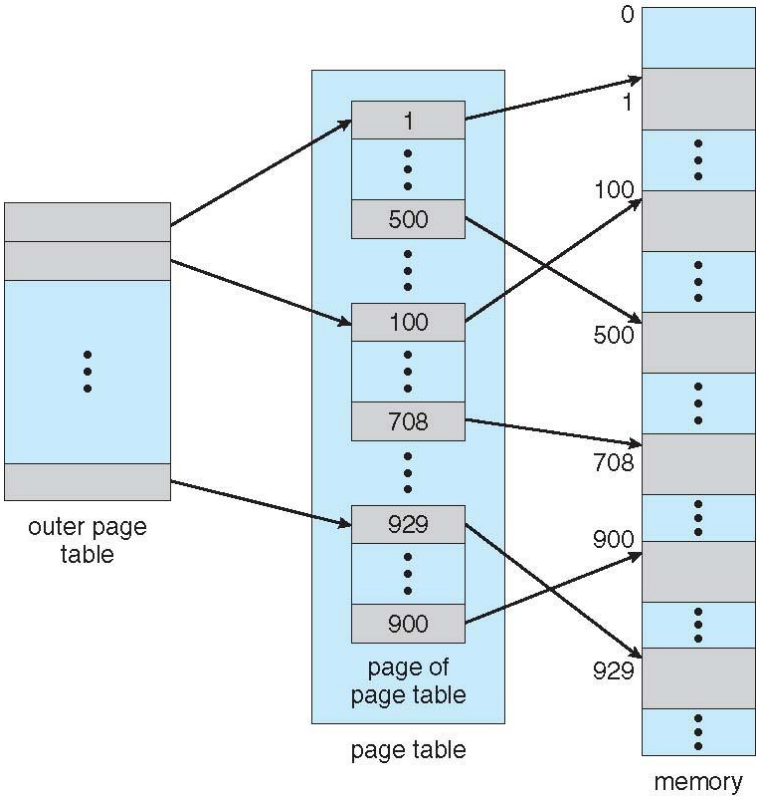
Structure of the Page Table

- Struktur memori untuk paging dapat berukuran sangat besar
  - Misal dengan 32-bit logical address space
  - Page size of 1 KB (2^{10})
  - Page table akan terdiri atas 4 juta entries ($2^{32} / 2^{10}$)
  - Jika setiap 4 bytes ➔ setiap proses memerlukan 16 MB physical address space hanya untuk page table saja

  - Solusi: membagi page table menjadi beberapa unit
    - Hierarchical Paging
    - Hashed Page Tables
    - Inverted Page Tables

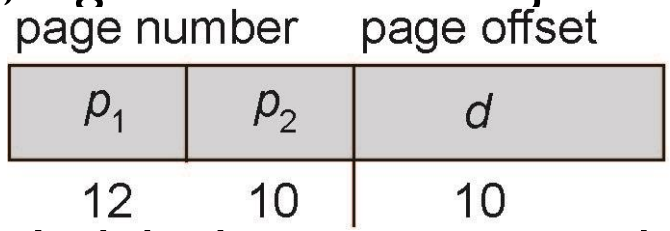
Hierarchical Page Tables

- logical address space dipecah menjadi multiple page tables
- Contoh: a two-level page table



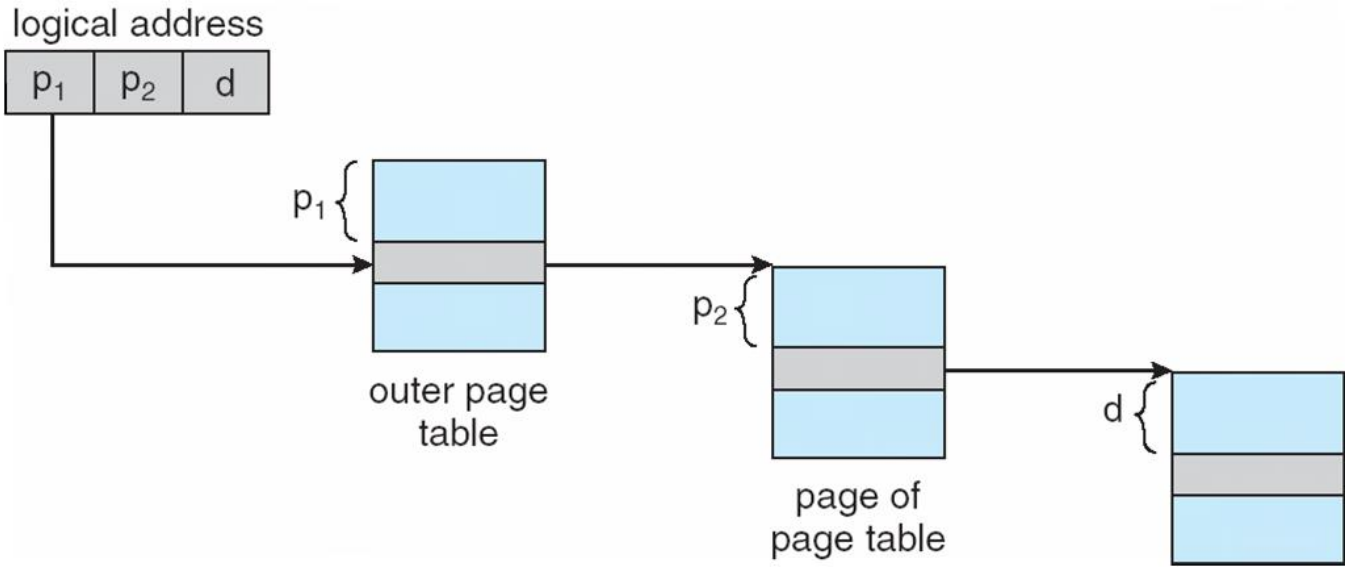
Two-Level Paging Example

- logical address (32-bit machine dengan 1K page size) dibagi menjadi :
  - page number yang terdiri atas 22 bits
  - a page offset yang terdiri atas10 bits
- Karena page table juga dibuat menjadi page, page number dibagi lagi menjadi:
  - 12-bit page number
  - 10-bit page offset
- Sehingga, logical address menjadi:



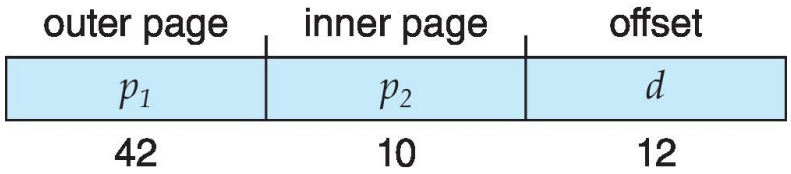
- p_1 adalah indeks ke outer page table, p_2 indeks ke inner page table
- Disebut juga **forward-mapped page table**

Address-Translation Scheme



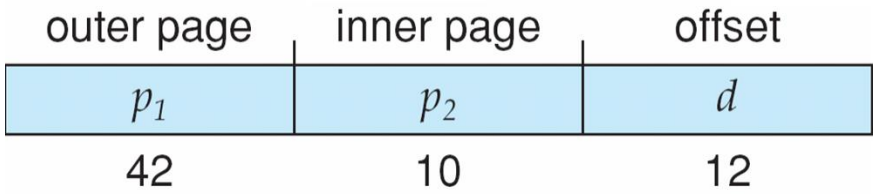
64-bit Logical Address Space

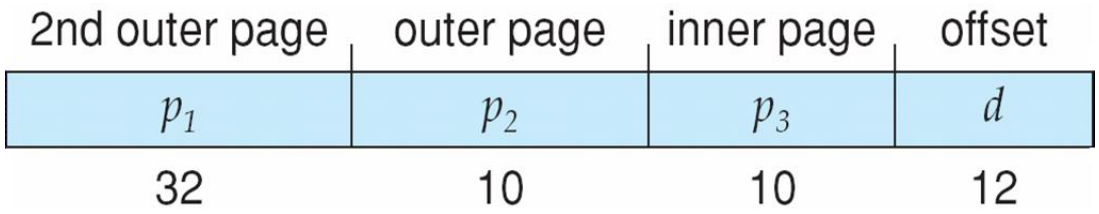
- Mekanisme two-level paging mungkin masih tidak cukup
- Jika ukuran page size 4 KB (2^{12})
  - Maka page table memiliki 2^{52} entries (untuk 64 bit address)
  - Pada skema two level, inner page tables dapat berukuran 2^{10} 4-byte entries
  - Address would look like



  - Outer page table memiliki 2^{42} entries atau 2^{44} bytes
  - Salah satu solusinya adalah 2^{nd} outer page table
  - Namun dengan seperti ini 2^{nd} outer page table masih berukuran 2^{34} bytes in size
    - Dan juga memerlukan 4 memory access untuk mengakses 1 physical memory location

Three-level Paging Scheme





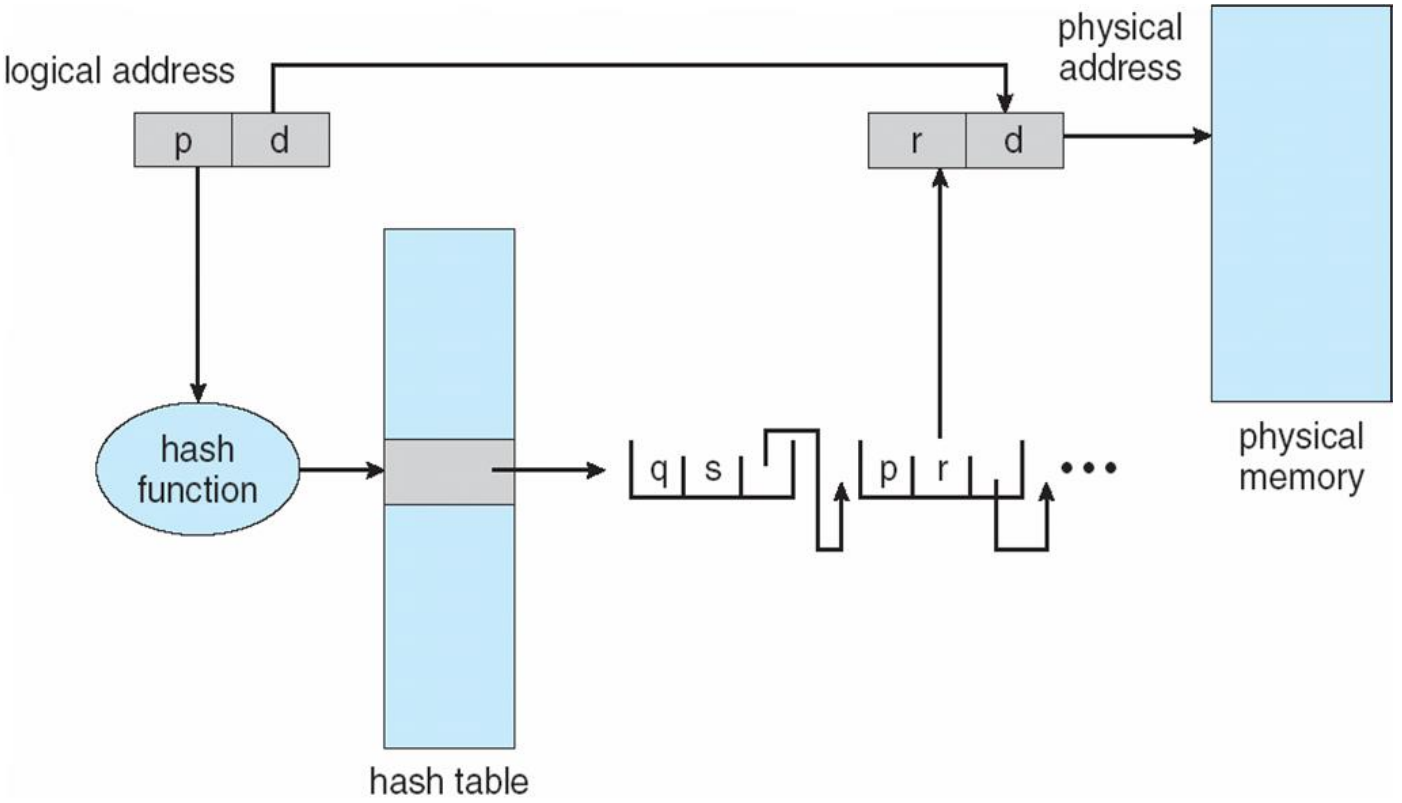
Hashed Page Tables

- Digunakan pada arsitektur dengan address spaces > 32 bits
- virtual page number di-hashed ke dalam page table
  - page table berisi linked-list element yang memiliki hash yg sama
- Setiap element berisi
  1. The virtual page number
  2. Nilai mapped mapped page frame
  3. pointer ke elemen berikutnya
- Virtual page numbers dibandingkan saat menelusuri linked-list element ini
  - Jika ditemukan, physical frame yang bersesuaian akan dibaca

Hashed Page Tables (Cont.)

- Variasi untuk 64-bit addresses adalah dengan **clustered page tables**
  - Mirip dengan hashed, namun setiap entri mengacu ke beberapa pages (misal 16), tidak hanya 1
  - Khususnya cocok untuk **sparse** address spaces (dimana memory references tidak contigu dan tersebar)

Hashed Page Table



Inverted Page Table

- Pada cara ini, dibalik, setiap proses tidak lagi mencatat page table dan men-track pemetaan dari logical page ke physical page, namun mentrak semua physical pages
- Satu entry untuk setiap real page of memory
- Entry terdiri atas virtual address dari page yang disimpan pada lokasi real memory tersebut, dengan informasi tentang proses yang memiliki page tersebut
- Mengurangi memori yang diperlukan untuk menyimpan setiap page table, namun menambah waktu yang diperlukan untuk pencarian saat ada akses page
- Menggunakan hash table agar pencarian dapat dilakukan hanya pada satu (atau beberapa) page-table entries
  - TLB can accelerate access
- Problem: bagaimana mengimplementasikan shared memory?
  - Hanya ada 1 mapping virtual address ke shared physical address

Inverted Page Table Architecture