Operating Systems Concepts

Main Memory: Paging TLB

CS 4375, Fall 2025

Instructor: MD Armanuzzaman (***Arman***)
marmanuzzaman@utep.edu

October 20, 2025

Summary

- Main Memory:
  - Fixed and Dynamic Memory Allocation
  - External and Internal Fragmentation
  - Address Binding
  - Hardware Address Protection
  - Paging
  - Segmentation

Agenda

- Main Memory:
    - Implementation of Paging
    - Effective Access Time
    - Multi-level Paging
    - Example: Intel 32-bit Architecture

Dynamic Storage-Allocation Problem

- How to satisfy a request of size n from a list of free holes?
  - **First-fit:** Allocate the first hole that is big enough
  - **Best-fit:** Allocate the smallest hole that is big enough; must search entire list, unless ordered by size. Produces the smallest leftover hole.
  - **Worst-fit:** Allocate the largest hole; must also search entire list. Produces the largest leftover hole.

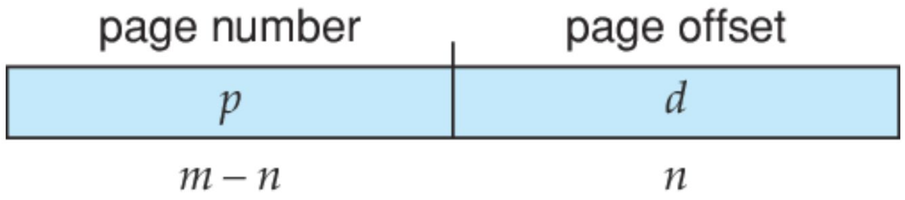
- First-fit is faster.
- Best-fit is better in terms of storage utilization.
- Worst-fit may lead to less fragmentation.

Dynamic Storage-Allocation Example

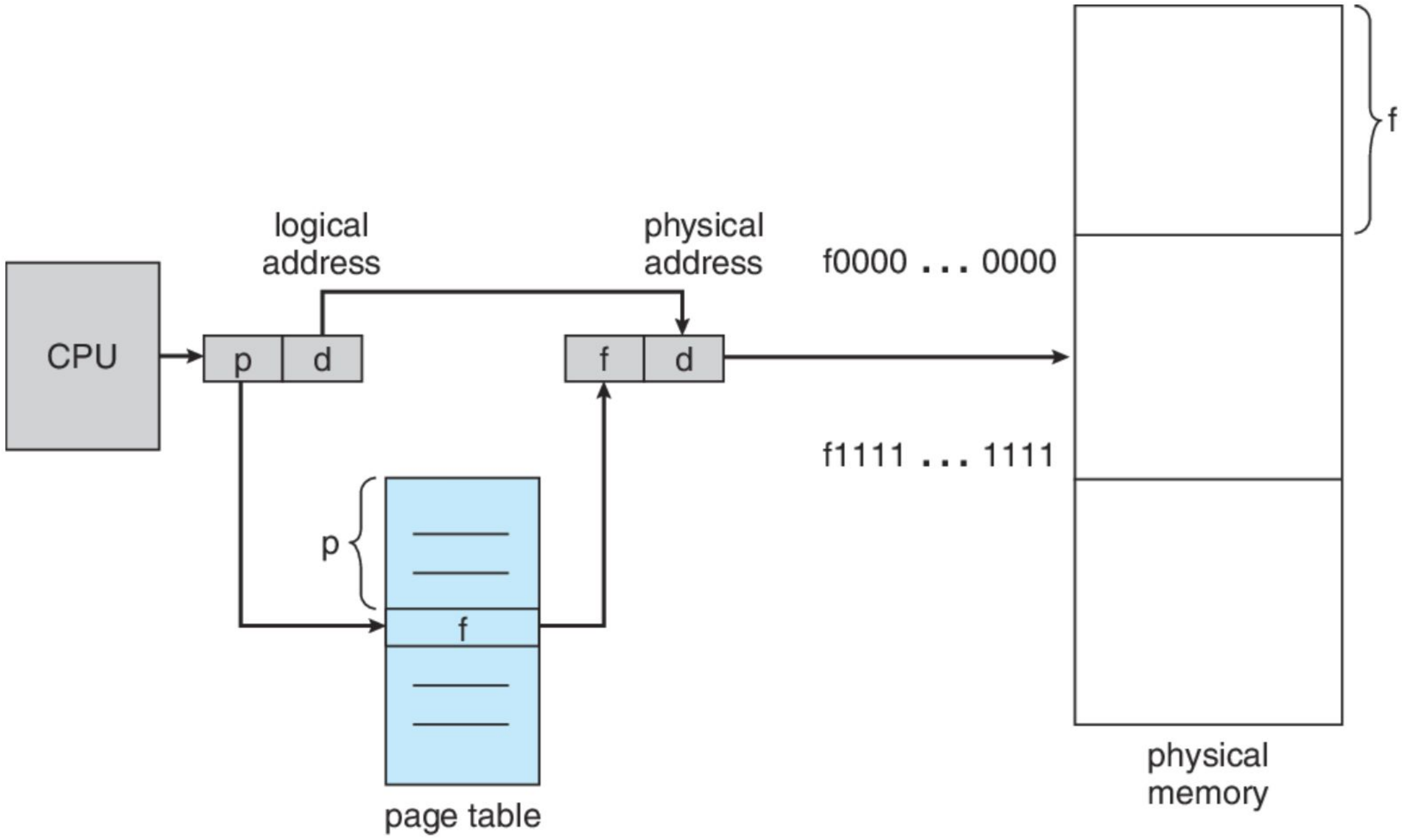
- Given five memory partitions of 100 KB, 500 KB, 200 KB, 300 KB, and 600 KB (in order), how would each of the first-fit, best-fit, and worst-fit algorithms place processes of 212 KB, 417 KB, 112 KB, and 426 KB (in order)?
- Which algorithm makes the most-efficient use of memory?

Paging Address Translation Scheme

- Address generated by CPU is divided into:
  - **Page number (p)**– used as an index into a page table which contains base address of each page in physical memory
  - **Page offset (d)**– combined with base address to define the physical memory address that is sent to the memory unit



Address Translation Architecture



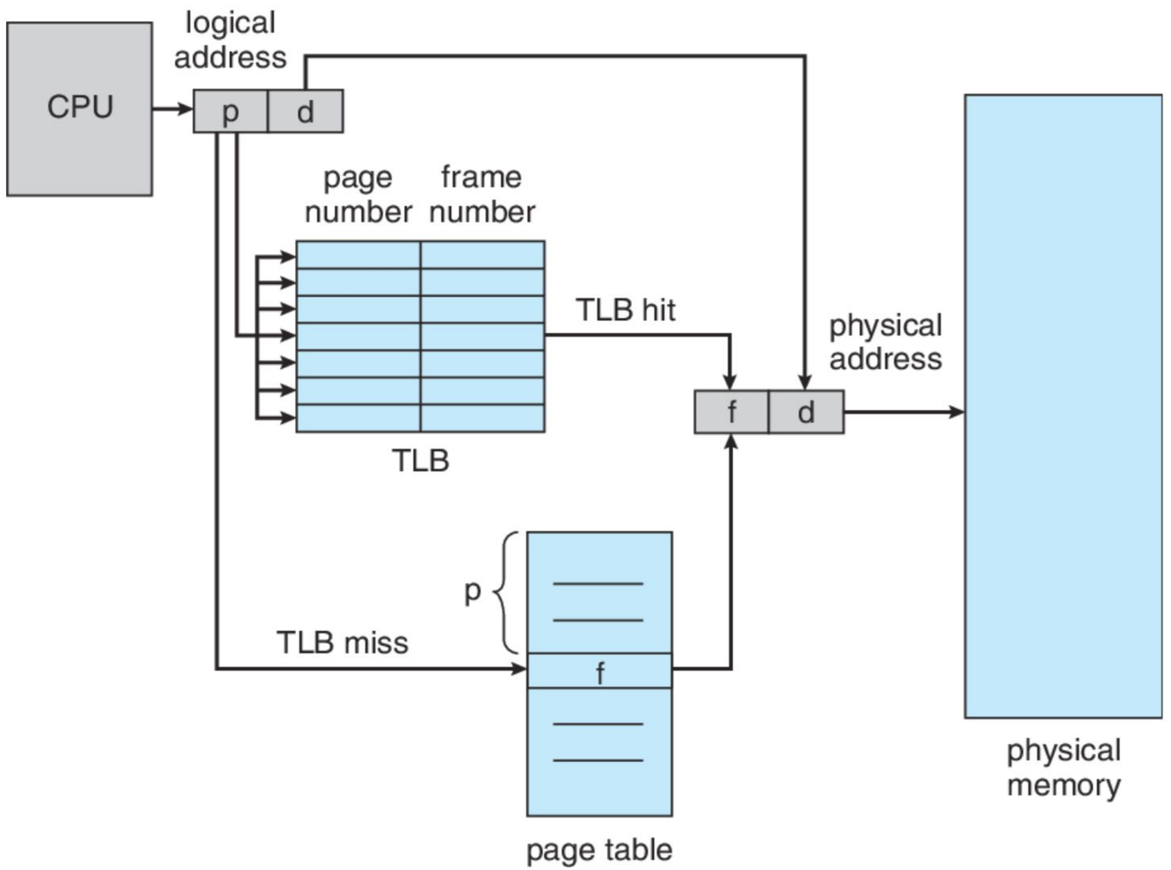
Implementation of Paging

- Page table is kept in main memory (why?)
- **Page-table base register (PTBR)** points to the page table
- **Page-table length register (PTLR)** indicates size of the page table
- In this scheme every data/instruction access requires two memory accesses
    - One for the page table and one for the data / instruction
- The two memory access problem can be solved by the use of a special fast-lookup hardware cache called **associative memory** or **translation lookaside buffers (TLBs)**

Implementation of Paging - TLB

- TLBs typically small (64 to 1,024 entries)
- On a **TLB miss**, value is loaded into the TLB for faster access next time
  - Replacement policies must be considered
  - Some entries can be wired down for permanent fast access
- Some TLBs store **A**ddress-**S**pace **ID**entifiers (ASIDs) in each TLB entry – uniquely identifies each process to provide address-space protection for that process
  - Otherwise need to flush at every context switch
- An order of magnitude faster than memory access

Implementation of Paging - TLB



Effective Access Time

- **α:** Hit ratio - percentage of times that a page number is found in the associative registers; ratio related to number of associative registers
- **M:** memory access time
- **ε:** associative lookup time
- Effective Access Time (EAT):

$$EAT = (\text{Hit Rate} \times \text{Hit Time}) + (\text{Miss Rate} \times \text{Miss Time})$$

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

$$\text{If } M \gg \varepsilon \rightarrow EAT = \alpha M + (1 - \alpha)(2M) = M + (1 - \alpha)M$$

- Example: consider α=80% and M=100ns

  EAT = 0.80 x 100 + 0.2 x 200 = 120ns

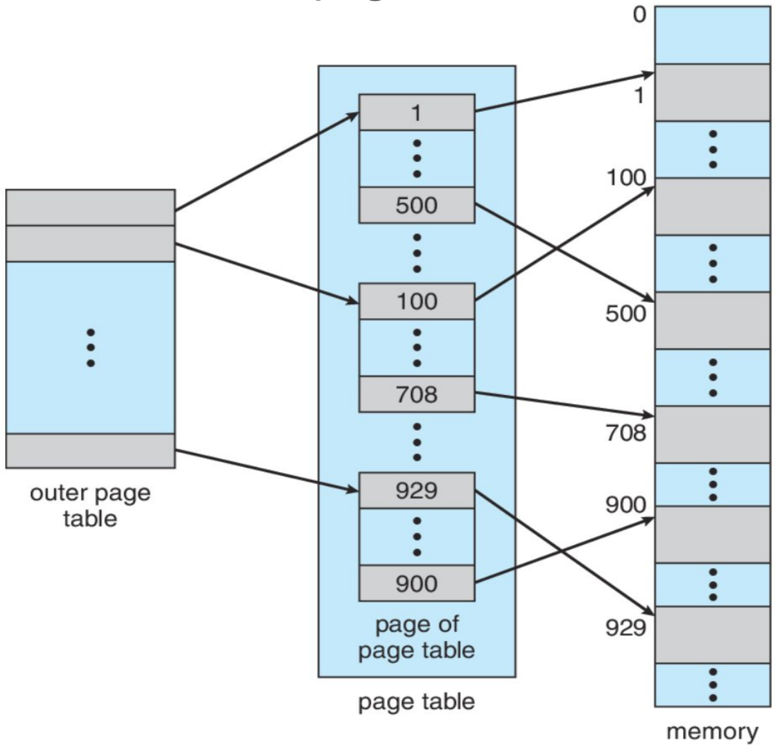
  If α=99%, EAT = 101ns

Structure of the Page Table

- Memory structures for paging can get huge using straightforward methods
- Consider a 32-bit logical address space as on modern computers (2^{32} Bytes of Memory)
- Page size of 4 KB (2^{12} Bytes)
- Page table would have 1 million entries (2^{32} / 2^{12})
- If each entry is 4 bytes → 4 MB of physical address space / memory for page table alone
    - That amount of memory used to cost a lot
    - Don't want to allocate that contiguously in main memory

Hierarchical Page Tables

- Break up the logical address space into multiple page tables
- A simple technique is a two-level page table

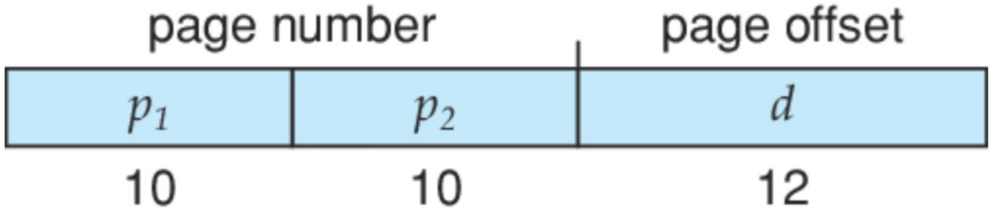


Two-Level Paging Example

- A logical address (on 32-bit machine with 4K page size) is divided into:
  - a page number consisting of 20 bits
  - a page offset consisting of 12 bits
- Since the page table is paged, the page number is further divided into:
  - a 10-bit page table index (within the outer page directory)
  - a 10-bit page index (within the inner page table)

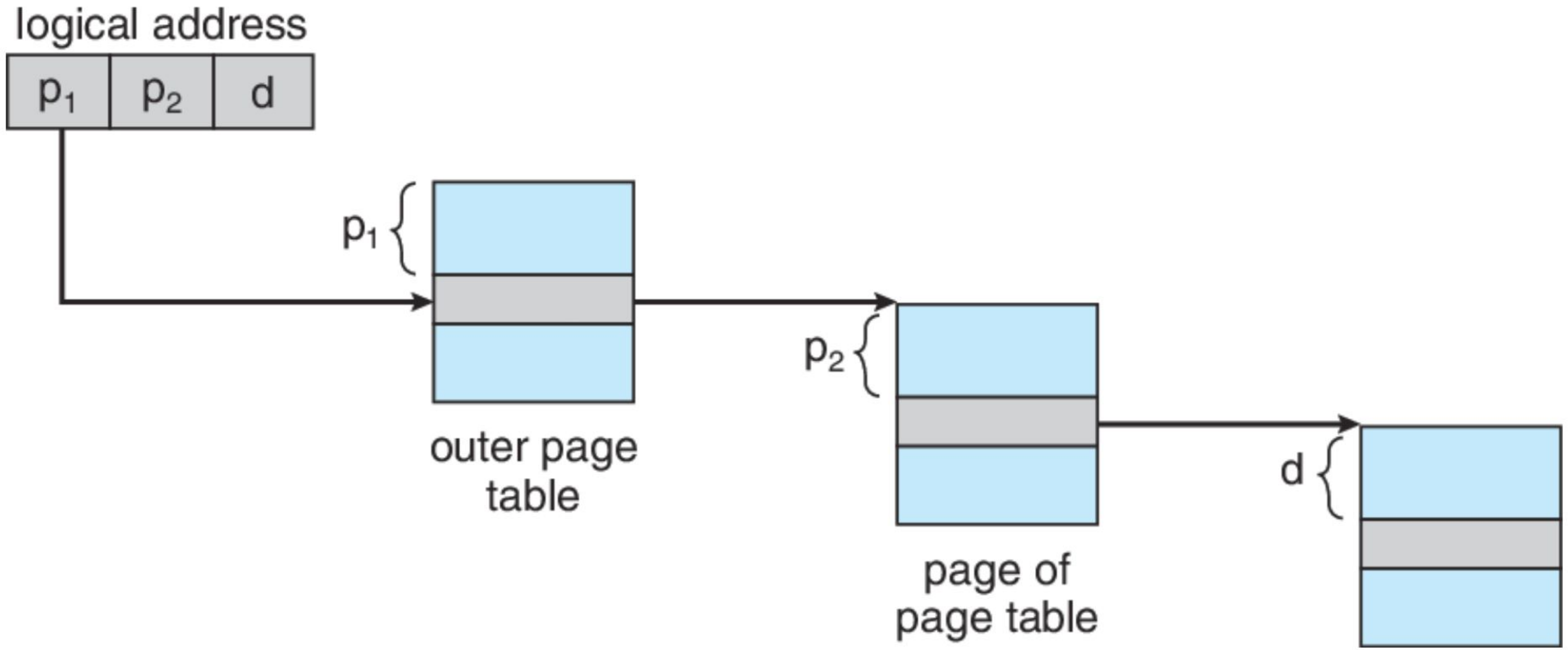
Two-Level Paging Example

- Thus, a logical address is as follows:



- where p_1 is an index into the outer page table, and p_2 is the displacement within the page of the inner page table
- Known as **forward-mapped page table**

Two-Level Paging Example: Address Translation

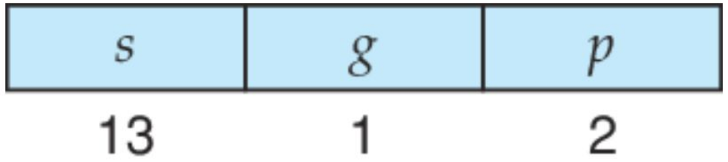


Example: The Intel IA-32 Architecture

- Supports both segmentation and segmentation with paging
  - Each segment can be 4 GB
  - Up to 16 K segments per process
  - Divided into two partitions
    - First partition of up to 8 K segments are private to process **kept in local descriptor table (LDT)**
    - Second partition of up to 8K segments shared among all processes **kept in global descriptor table (GDT)**

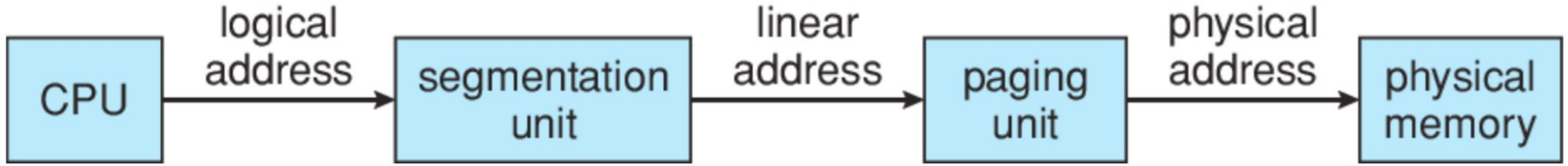
Example: The Intel IA-32 Architecture

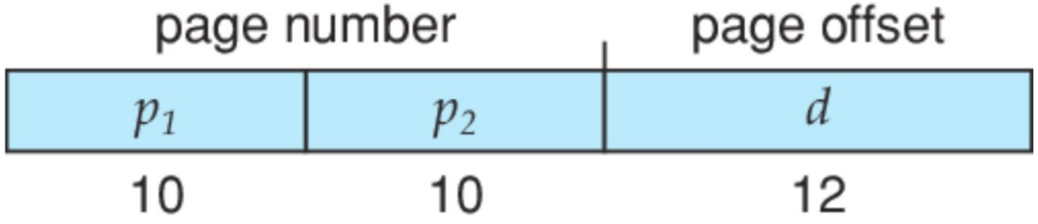
- CPU generates logical address
  - Selector given to segmentation unit
    - Which produces linear addresses



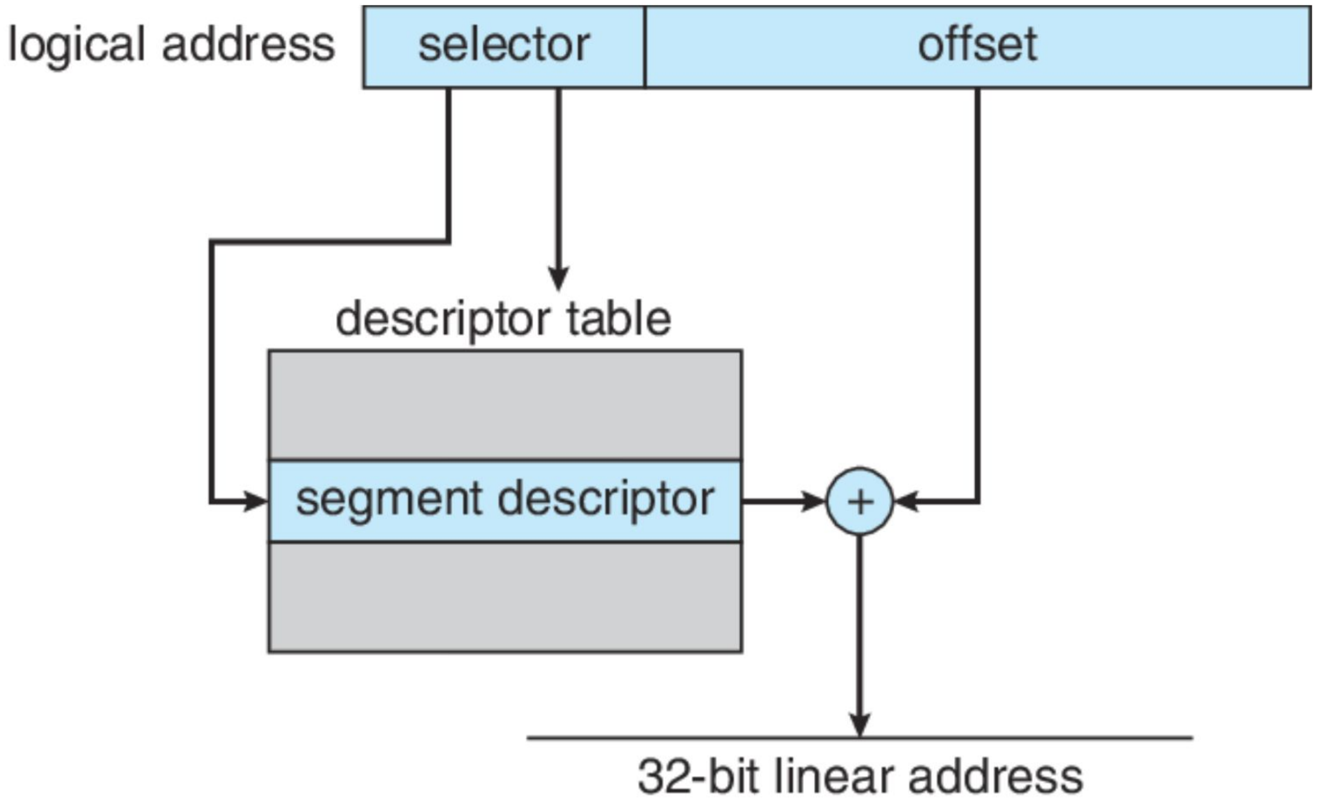
  - Linear address given to paging unit
    - Which generates physical address in main memory
    - Pages sizes can be 4 KB or 4 MB
  - Segmentation and paging units form equivalent of MMU

Logical to Physical Address Translation in IA-32

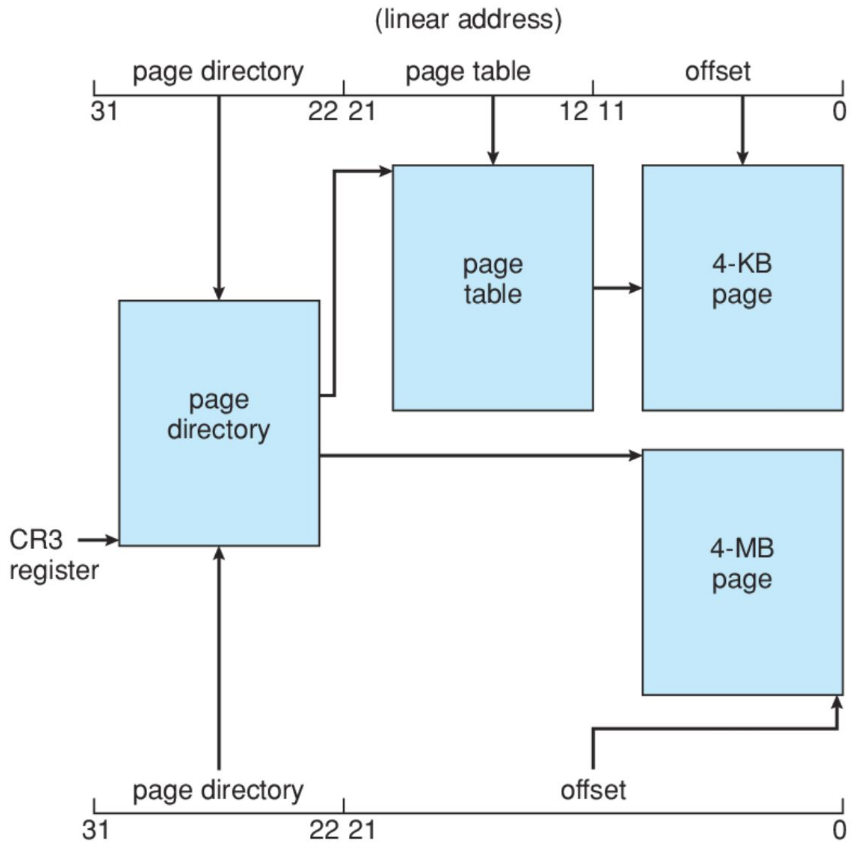




Intel IA-32 Segmentation



Intel IA-32 Paging Architecture



Acknowledgements

- "Operating Systems Concepts" book and supplementary material by A. Silberschatz, P. Galvin and G. Gagne
- "Operating Systems: Internals and Design Principles" book and supplementary material by W. Stallings
- "Modern Operating Systems" book and supplementary material by A. Tanenbaum
- R. Doursat and M. Yuksel from University of Nevada, Reno
- Farshad Ghanei from Illinois Tech
- T. Kosar and K. Dantu from University at Buffalo

Announcement

- Homework 3
  - **START EARLY**
  - Due on October 27th, 11.59 PM
- Quiz 4 next class