CS370 Operating Systems

Colorado State University

Yashwant K Malaiya

Fall 2016 Lecture 30

Main Memory

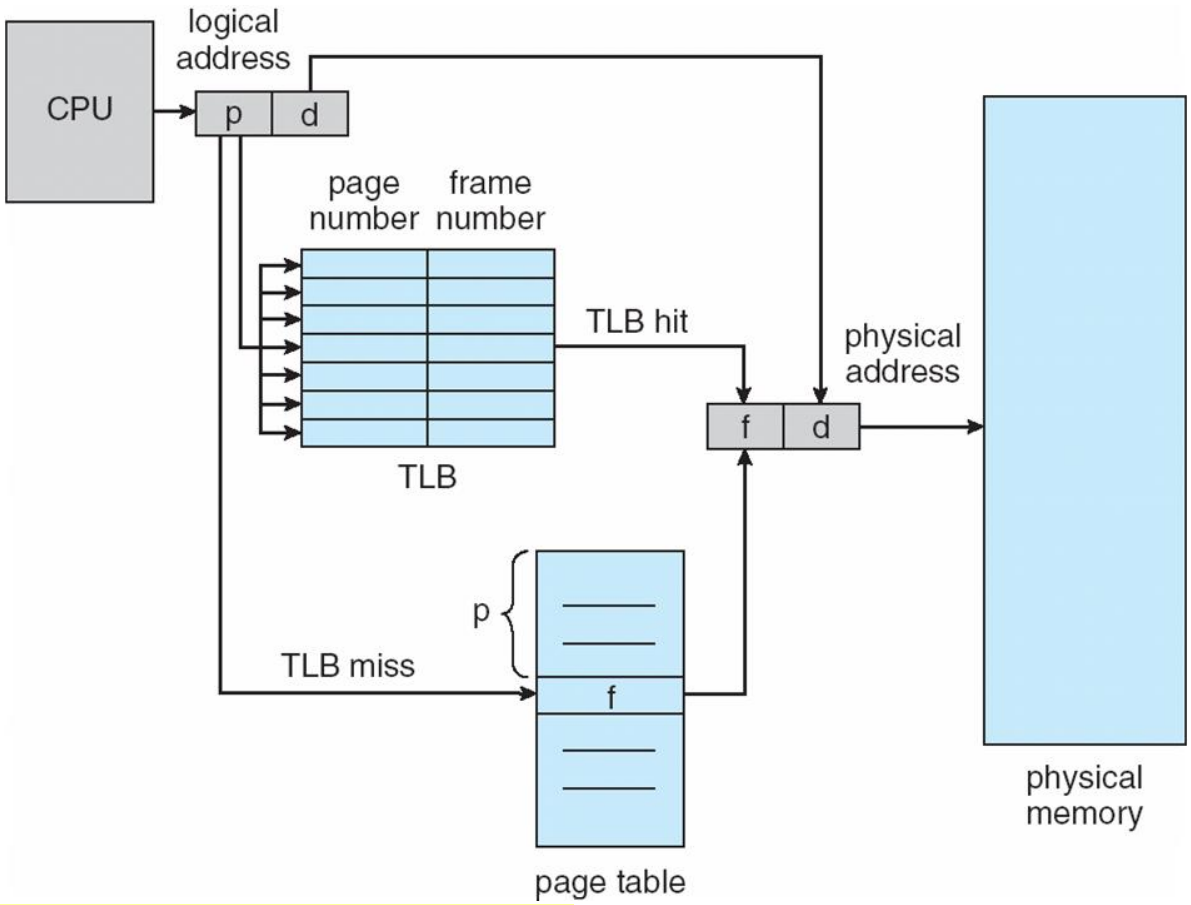
Slides based on

- **Text by Silberschatz, Galvin, Gagne**
- **Various sources**

Questions from last time

- Page table and page offset:
  - Logical address (m bits) = Page number, Page Offset (n bits)
  - Page size 2^n bytes, Logical memory size 2^m bytes
  - Page table maps Page number into Frame number
- Why is two memory access problem a big deal? cuts performance into about half!
- How to find the page table in memory? Page table base register
- Why need to flush TLB when there is a context switch? Because mapping is process specific, unless TLB can accommodate multiple processes
- Why use associative memory for TLBs? To see if the mapping for a specific page is there.

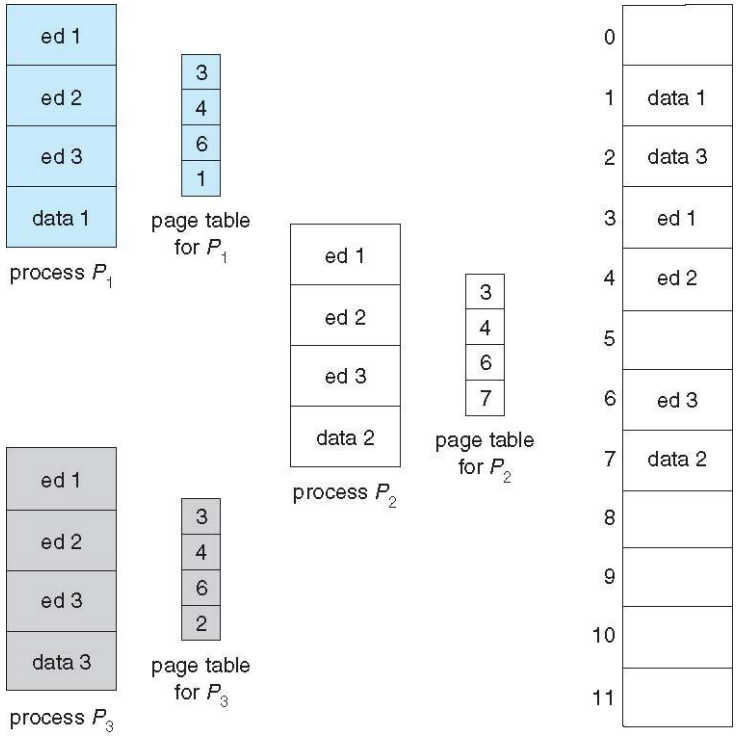
Paging Hardware With TLB



TLB Miss: page table access may be done using hardware or software

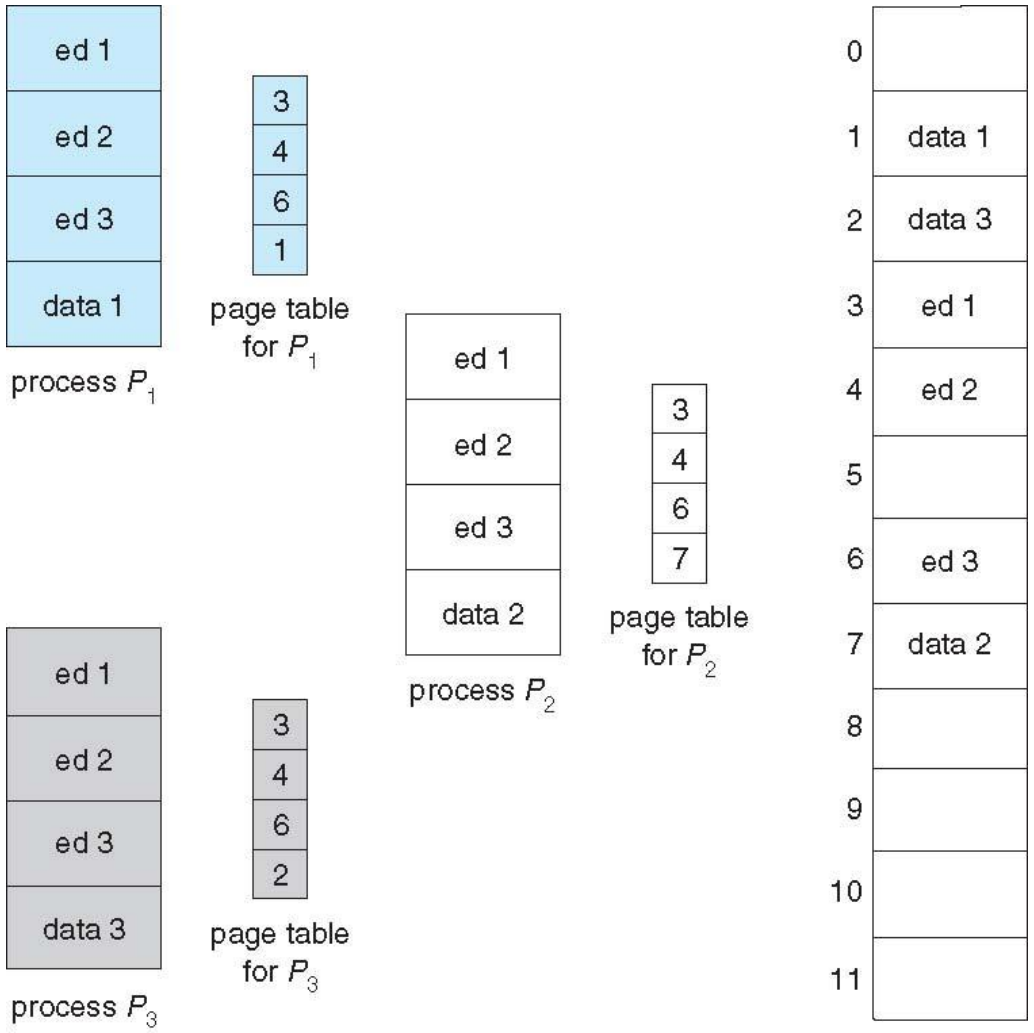
Questions from last time

- How are “pages” shared? Include in address space of both processes



- Difference between logical and virtual memory: Virtual memory is the term often used when a part of physical memory is implemented in disk. Details soon

Shared Pages Example



ed1, ed2, ed3
(3, 4, 6) shared

Colorado State University

Page Table Size

- Memory structures for paging can get huge using straight-forward methods
  - Consider a 32-bit logical address space as on recent processors 64-bit on 64-bit processors
  - Page size of 4 KB (2^{12}) entries
  - 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

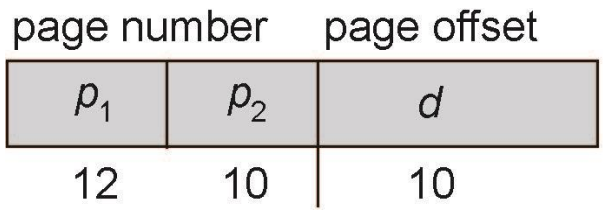
2^{10}	**1024 or 1 kibibyte**
2^{20}	1M mebibyte
2^{30}	1G gigibyte
2^{40}	1T tebibyte

Issues with large page tables

- Cannot allocate page table **contiguously** in memory
- Solutions:
  - Divide the page table into smaller pieces
  - **Page the page-table**
    - Hierarchical Paging

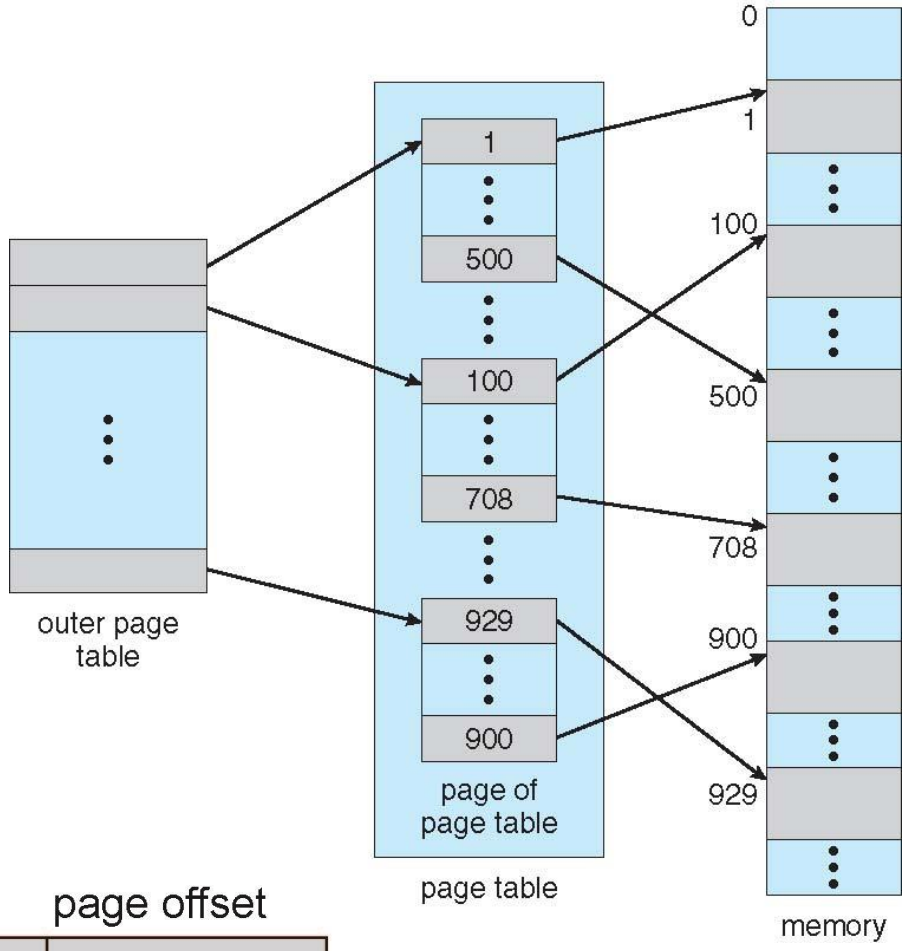
Hierarchical Page Tables

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



P1: indexes the outer page table
P2: page table: maps to frame

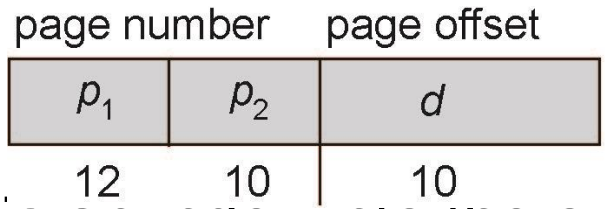
Two-Level Page-Table Scheme



page number		page offset
p_1	p_2	d
12	10	10

Two-Level Paging Example

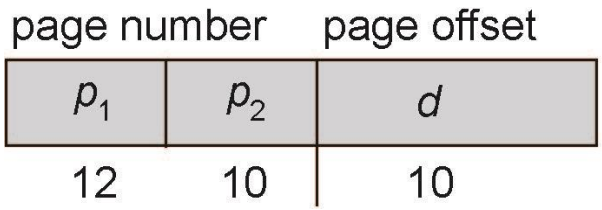
- A logical address (on 32-bit machine with 1K page size) is divided into:
  - a page number consisting of 22 bits
  - a page offset consisting of 10 bits
- Since the page table is paged, the page number is further divided into:
  - a 12-bit page number
  - a 10-bit page offset
- Thus, a logical address 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**

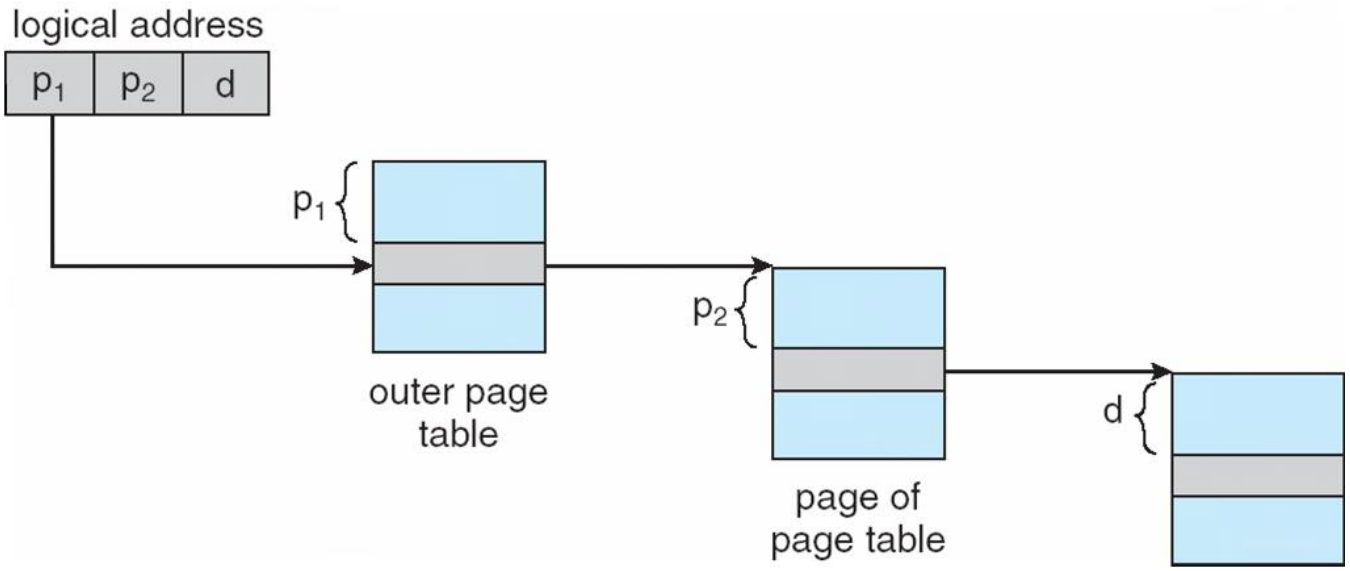
Two-Level Paging Example

- A logical address is as follows:



- One Outer page table: size 2^{12}
- Often only some of all possible 2^{12} Page tables needed (each of size 2^{10})

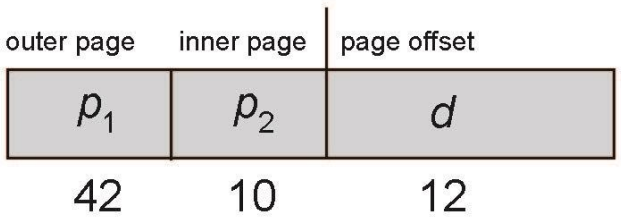
Address-Translation Scheme



If there is a hit in the TLB (say 95% of the time), then average access time will be close to slightly more than one memory access time.

64-bit Logical Address Space

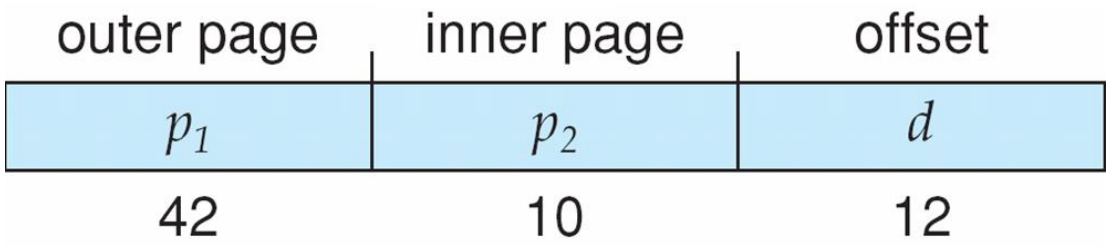
- Even two-level paging scheme not sufficient
- If page size is 4 KB (2^{12})
  - Then page table has 2^{52} entries
  - If two level scheme, inner page tables could be 2^{10} 4-byte entries
  - Address would look like



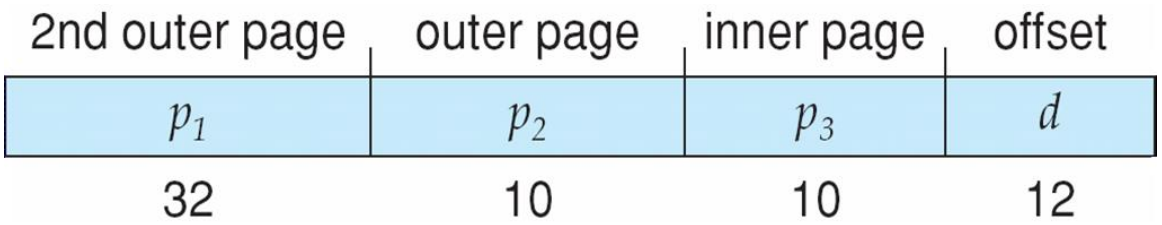
  - Outer page table has 2^{42} entries or 2^{44} bytes
  - One solution is to add a 2^{nd} outer page table
    - But in the following example the 2^{nd} outer page table is still 2^{34} bytes in size
      - And possibly 4 memory access to get to one physical memory location!

Full 64 bit physical memories not common yet

Three-level Paging Scheme



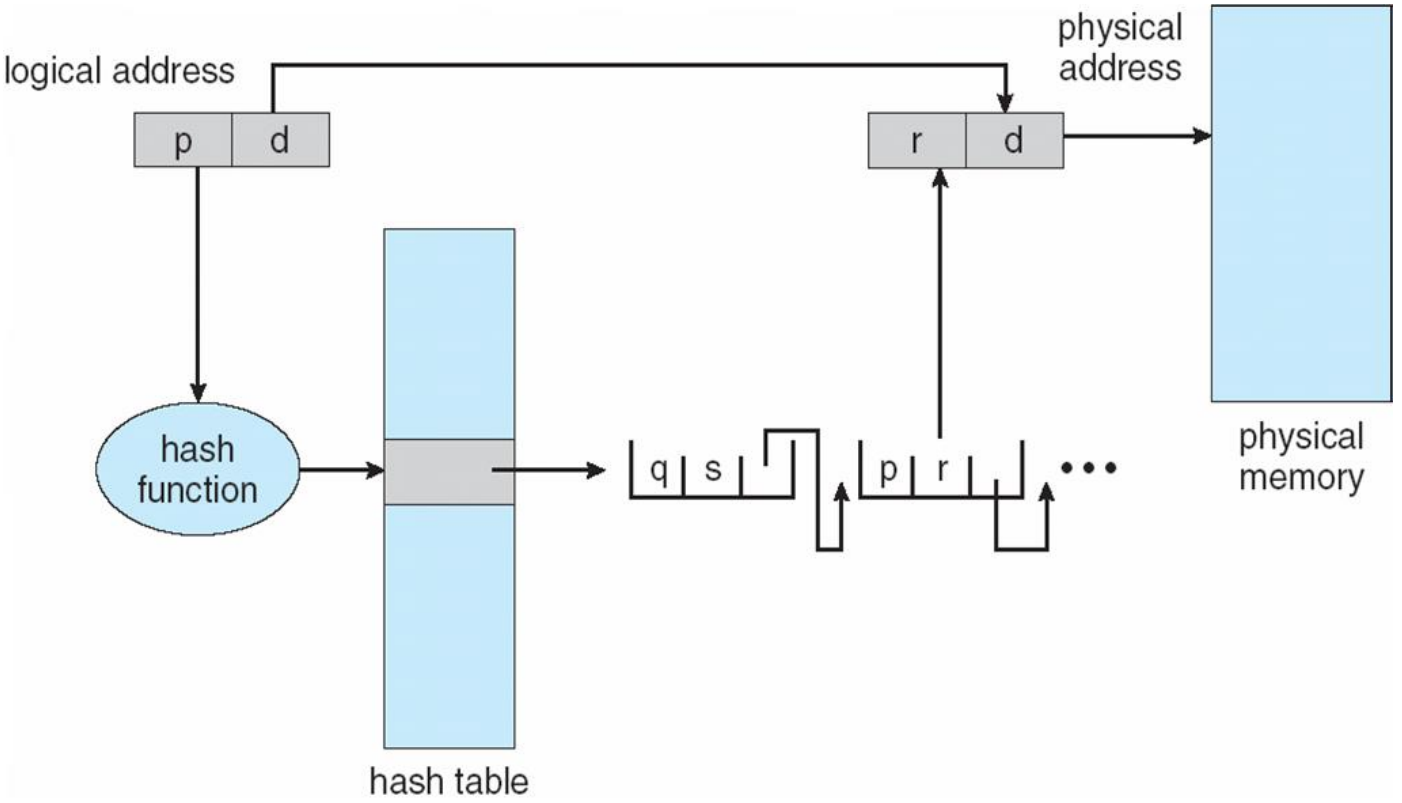
- Outer page table has 242 entries!
- Divide the outer page table into 2 levels
  - 4 memory accesses!



Hashed Page Tables

- Common in address spaces > 32 bits
- The virtual page number is hashed into a page table
  - This page table contains a chain of elements hashing to the same location
- Each element contains (1) the virtual page number (2) the value of the mapped page frame (3) a pointer to the next element
- Virtual page numbers are compared in this chain searching for a match
  - If a match is found, the corresponding physical frame is extracted
- Variation for 64-bit addresses is **clustered page tables**
  - Similar to hashed but each entry refers to several pages (such as 16) rather than 1
  - Especially useful for **sparse** address spaces (where memory references are non-contiguous and scattered)

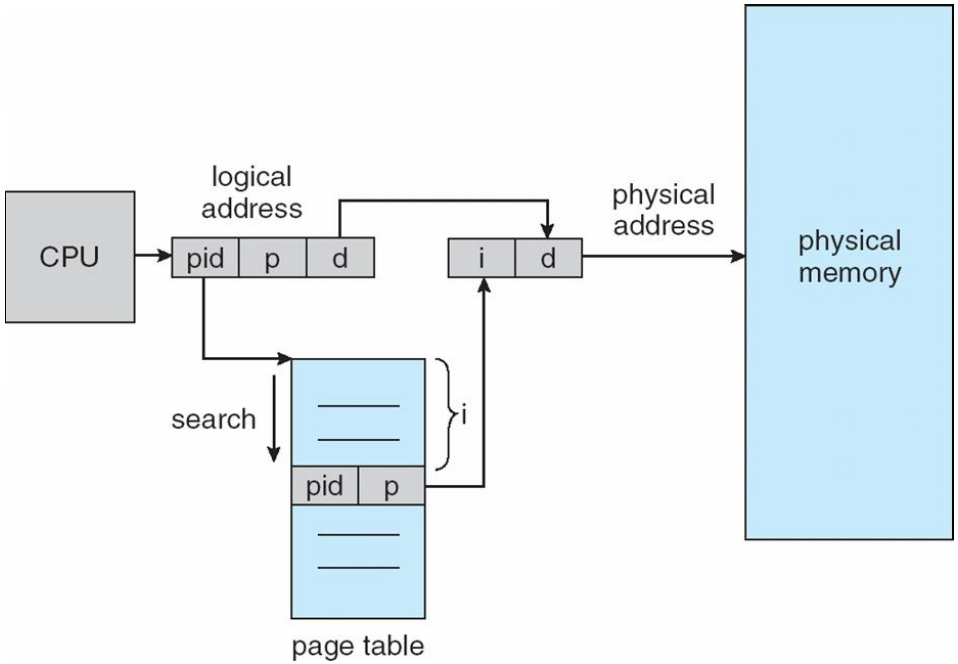
Hashed Page Table



This page table contains a chain of elements hashing to the same location.
Each element contains (1) the virtual page number (2) the value of the mapped page frame (3) a pointer to the next element

Inverted Page Table

- Rather than each process having a page table and keeping track of all possible logical pages, track all physical pages
  - One entry for each real page of memory
  - Entry consists of the virtual address of the page stored in that real memory location, with information about the process that owns that page



Search for pid, p, offset i is the physical frame address

Inverted Page Table

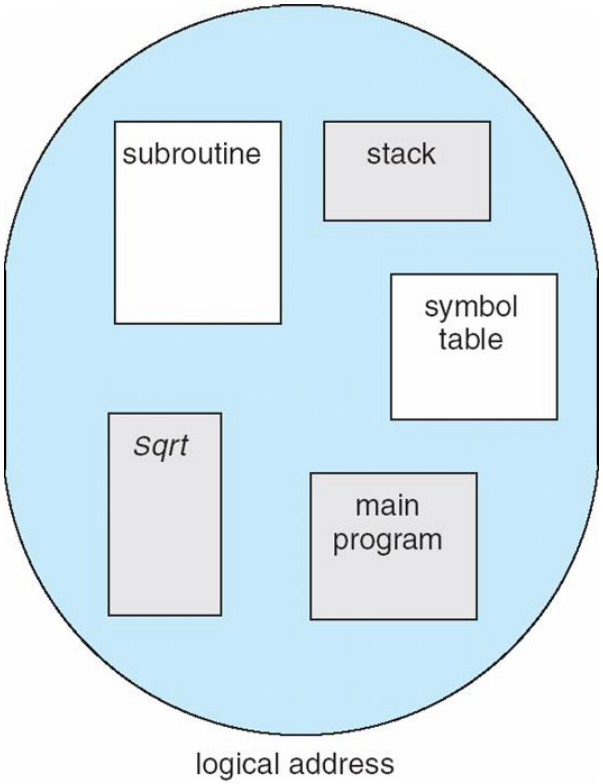
- Decreases memory needed to store each page table, but increases time needed to search the table when a page reference occurs
- But how to implement shared memory?
  - One mapping of a virtual address to the shared physical address. Not possible.

Segmentation Approach

Memory-management scheme that supports user view of memory

- A program is a collection of segments
  - A segment is a logical unit such as:
    main program
    procedure, function, method
    object
    local variables, global variables
    common block
    stack, arrays, symbol table

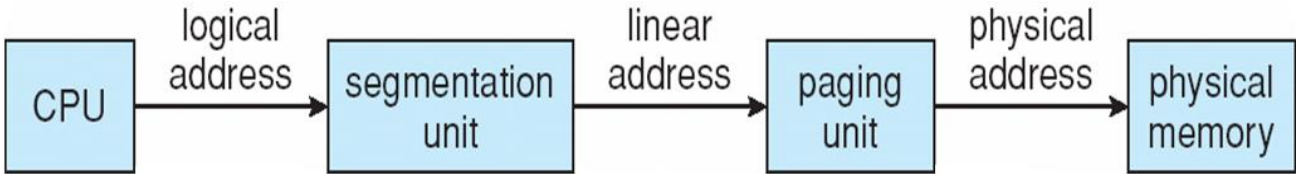
- Segment table
  - Segment-table base register (STBR)
  - Segment-table length register (STLR)
- segments vary in length, can very dynamically
- Segments may be paged
- Used for x86-32 bit
- Origin of term “segmentation fault”

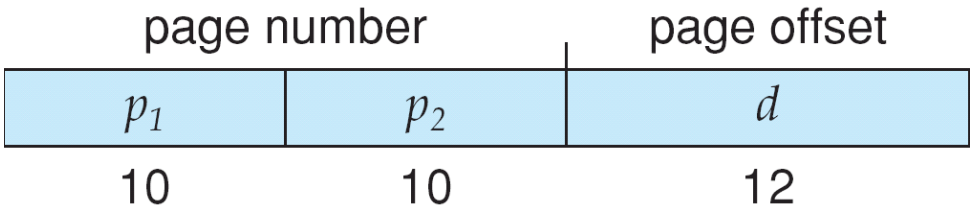


Examples

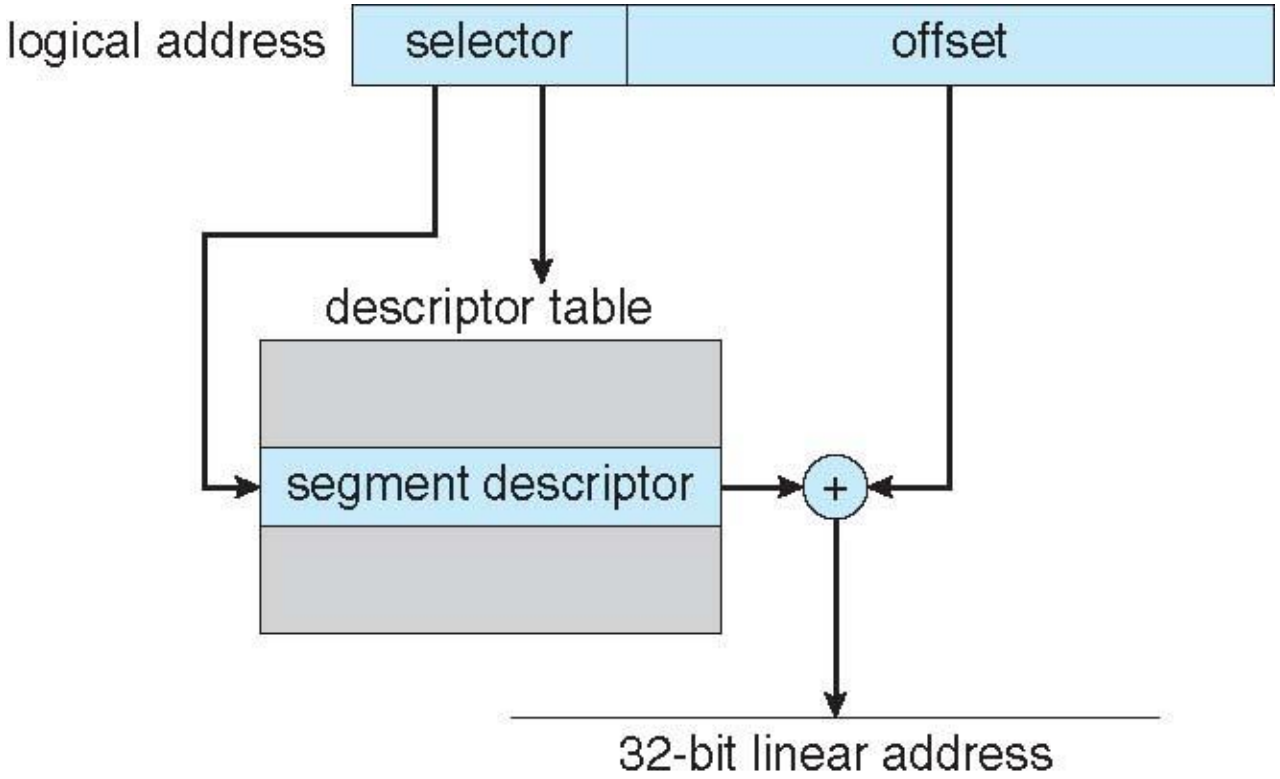
- Intel IA-32 (x386-Pentium)
- x86-64 (AMD, Intel)
- ARM

Logical to Physical Address Translation in IA-32

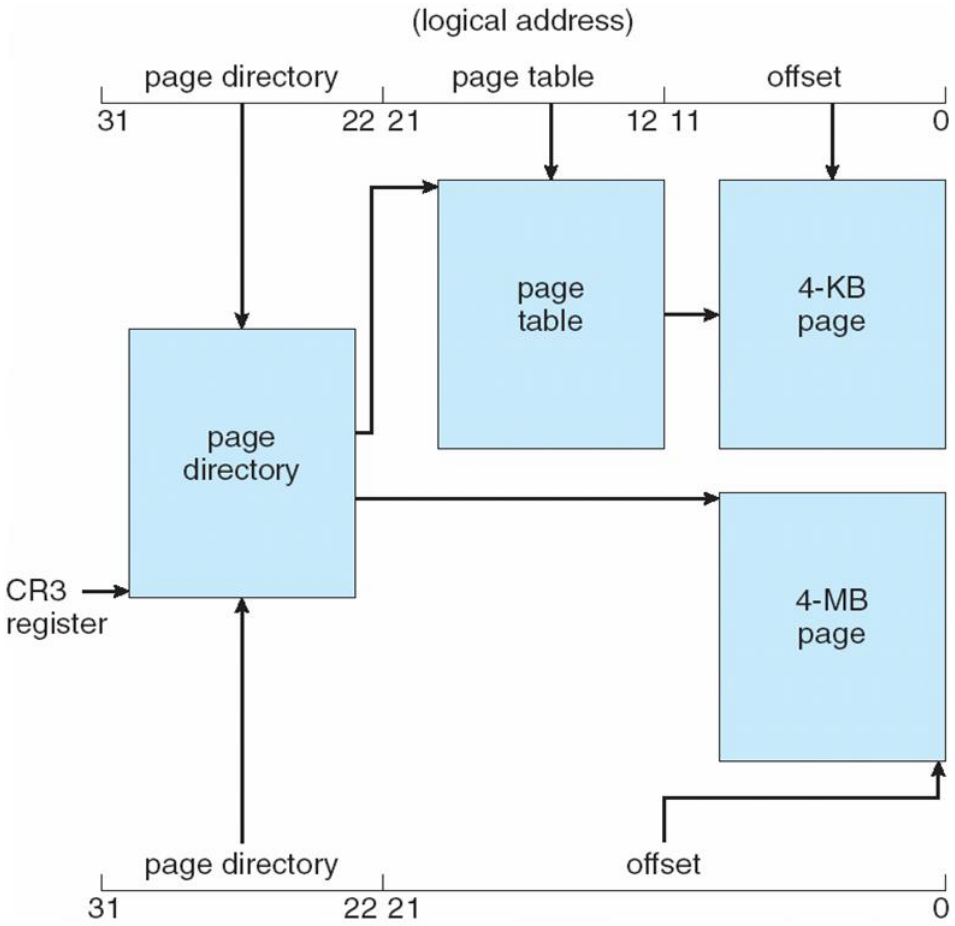




Intel IA-32 Segmentation

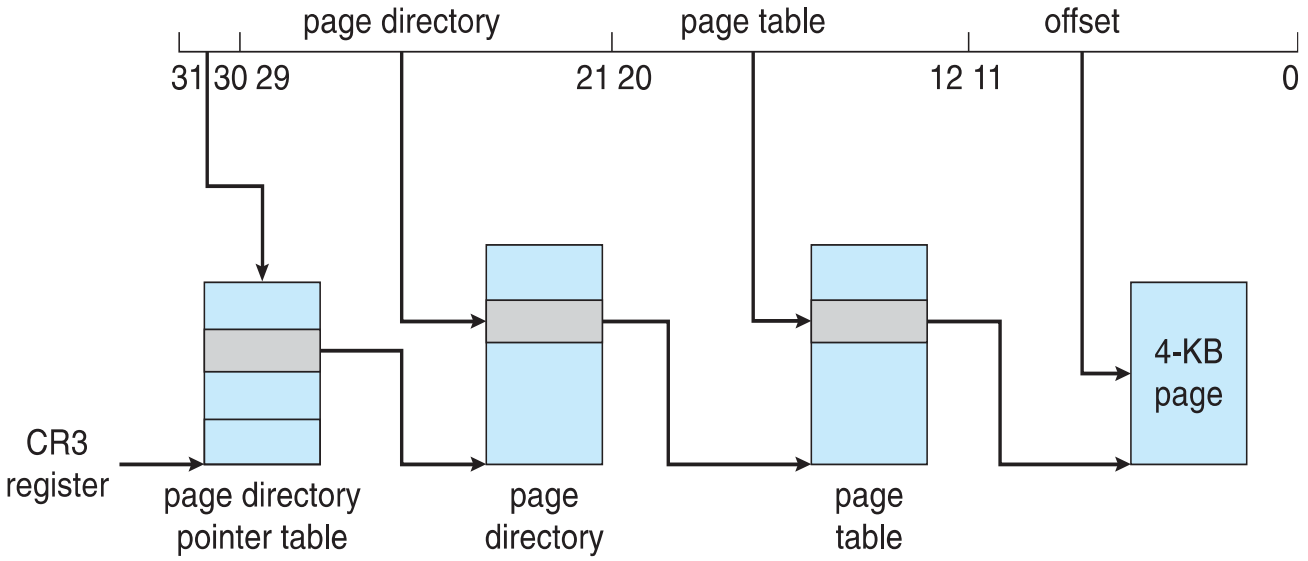


Intel IA-32 Paging Architecture



Intel IA-32 Page Address Extensions

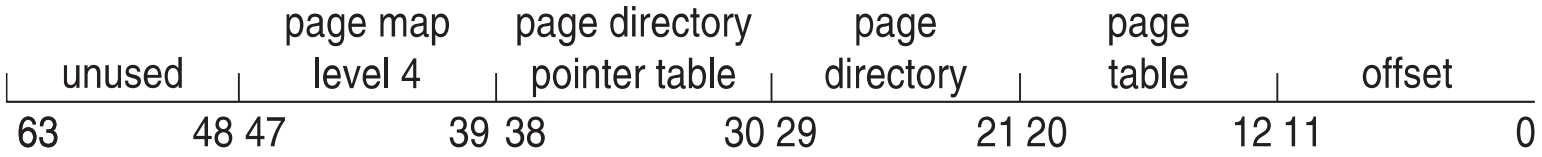
- 32-bit address limits led Intel to create **page address extension** (**PAE**), allowing 32-bit apps access to more than 4GB of memory space
  - Paging went to a 3-level scheme
  - Top two bits refer to a **page directory pointer table**
  - Page-directory and page-table entries moved to 64-bits in size
  - Net effect is increasing address space to 36 bits – 64GB of physical memory



Colorado State University

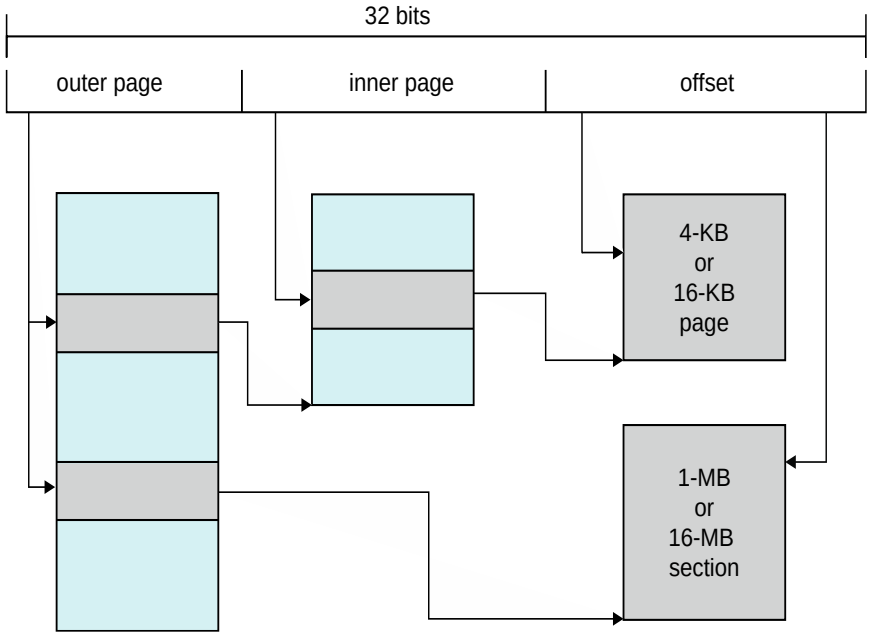
Intel x86-64

- Intel x86 architecture based on AMD 64 bit architecture
- 64 bits is ginormous (> 16 exabytes)
- In practice only implement 48 bit addressing
  - Page sizes of 4 KB, 2 MB, 1 GB
  - Four levels of paging hierarchy
- Can also use PAE so virtual addresses are 48 bits and physical addresses are 52 bits



Example: ARM Architecture

- Dominant mobile platform chip (Apple iOS and Google Android devices for example)
- Modern, energy efficient, 32-bit CPU
- 4 KB and 16 KB pages
- 1 MB and 16 MB pages (termed **sections**)
- One-level paging for sections, two-level for smaller pages
- Two levels of TLBs
  - Outer level has two micro TLBs (one data, one instruction)
  - Inner is single main TLB
  - First inner is checked, on miss outers are checked, and on miss page table walk performed by CPU



Colorado State University

CS370 Operating Systems

Colorado State University
Yashwant K Malaiya

Fall 2016 Lecture 30

Virtual Memory

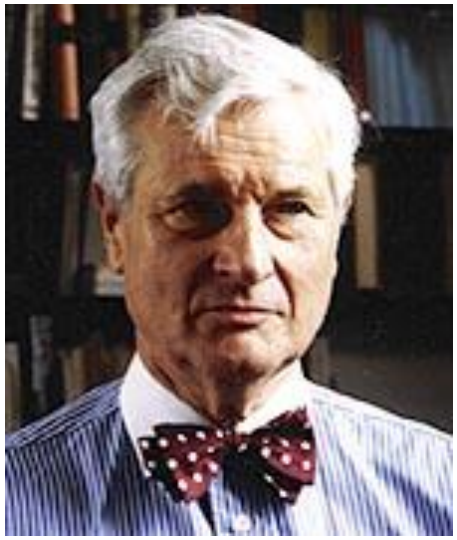
Slides based on

- Text by Silberschatz, Galvin, Gagne
- Various sources

Virtual Memory: Objectives

- Benefits of a virtual memory system
- Demand paging, page-replacement algorithms, and allocation of page frames
- The working-set model
- Relationship between shared memory and memory-mapped files
- To explore how kernel memory is managed

Fritz-Rudolf Güntsch



Fritz-Rudolf Güntsch (1925-2012) at the Technische Universität Berlin in 1956 in his doctoral thesis, *Logical Design of a Digital Computer with Multiple Asynchronous Rotating Drums and Automatic High Speed Memory Operation*.

First used in Atlas, Manchester, 1962

Windows 95

Chapter 9: Virtual Memory

- Background
- Demand Paging
- Copy-on-Write
- Page Replacement
- Allocation of Frames
- Thrashing
- Memory-Mapped Files
- Allocating Kernel Memory
- Other Considerations
- Operating-System Examples