OPERATING SYSTEMS

Memory Management

Chandravva Hebbi
Department of Computer Science

OPERATING SYSTEMS

- Inverted page table, Bigger pages

Chandravva Hebbi

Department of Computer Science

Slides Credits for all PPTs of this course

- The slides/diagrams in this course are an **adaptation**, **combination**, and **enhancement** of material from the following resources and persons:

1. Slides of Operating System Concepts, Abraham Silberschatz, Peter Baer Galvin, Greg Gagne - 9th edition 2013 and some slides from 10th edition 2018
2. Some conceptual text and diagram from Operating Systems - Internals and Design Principles, William Stallings, 9th edition 2018
3. Some presentation transcripts from A. Frank – P. Weisberg
4. Some conceptual text from Operating Systems: Three Easy Pieces, Remzi Arpaci-Dusseau, Andrea Arpaci Dusseau

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
- Decreases memory needed to store each page table, but increases time needed to search the table when a page reference occurs
- Use hash table to limit the search to one — or at most a few — page-table entries
  - TLB can accelerate access
- But how to implement shared memory?
  - One mapping of a virtual address to the shared physical address

Inverted Page Table

- A simplified version of the inverted page table used in the *IBM RT*.
- IBM was the first major company to use inverted page tables
  - starting with the IBM System 38 to RS/6000 and the current IBM Power CPUs.
  - For the IBM RT, each virtual address in the system consists of a triple:
    - <process-id, page-number, offset>.

Inverted Page Table

- Each inverted page-table entry is a pair <process-id, page-number> where the process-id assumes the role of the address-space identifier.
- When a memory reference occurs, part of the virtual address, consisting of <process-id, page number>
- The inverted page table is then searched for a match.
- If a match is found—say, at entry *i*—then the physical address <*i*, offset> is generated.
- If no match is found, then an illegal address access has been attempted.

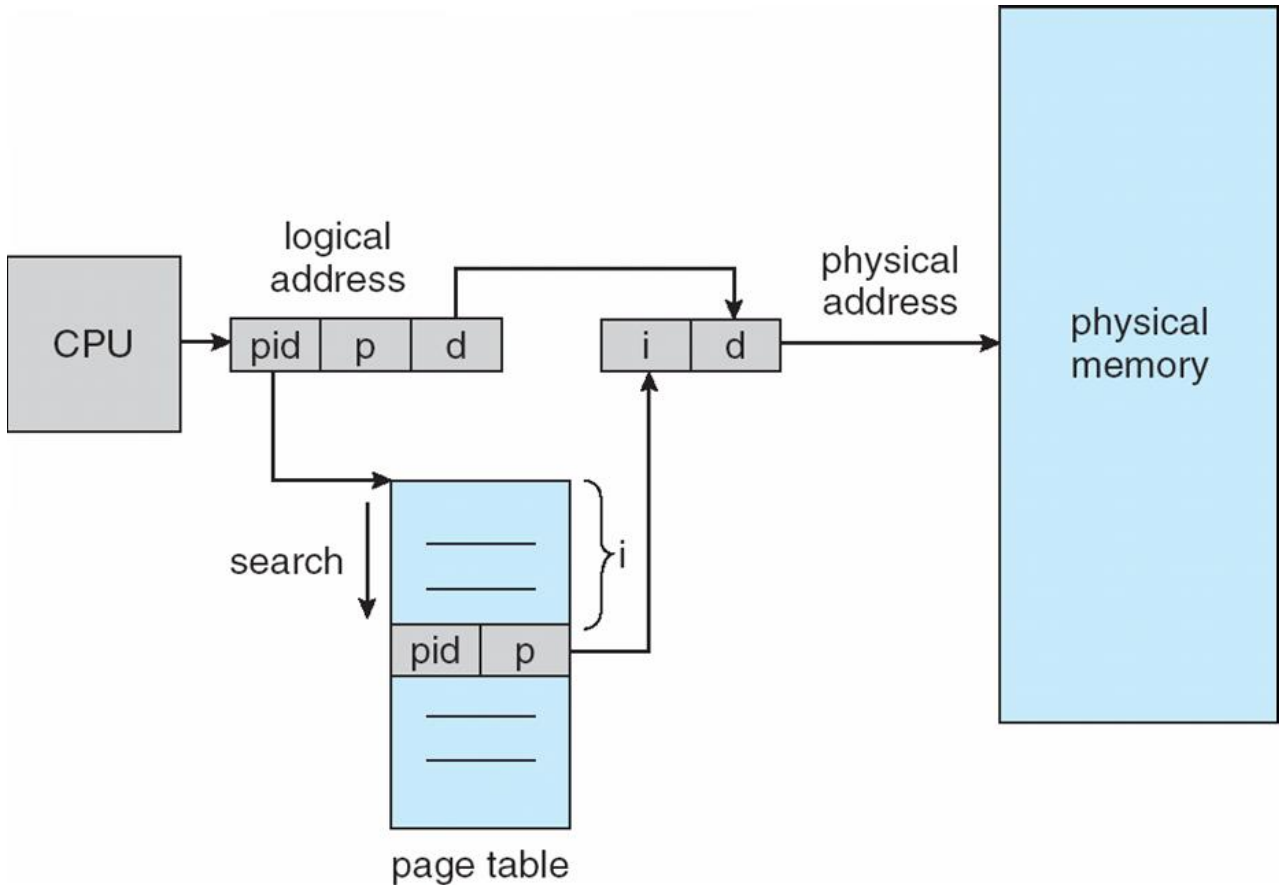
Inverted Page Table

- This scheme decreases the amount of memory needed to store each page table.
- It increases the amount of time needed to search the table when a page reference occurs.
- The inverted page table is sorted by physical address, but lookups occur on virtual addresses, the whole table might need to be searched before a match is found.
- Hash table is used to over come this problem where the number of search's are limited one or at most few.
- each access to the hash table adds a memory reference to the procedure.
- One virtual memory reference requires at least two real memory reads—one for the hash-table entry and one for the page table

Inverted Page Table

- Systems that use inverted page tables have difficulty in implementing shared memory.
- Shared memory is usually implemented as multiple virtual addresses are mapped to one physical address.
- There is only one virtual page entry for every physical page
- One physical page cannot have two (or more) shared virtual addresses.
- A simple technique for addressing this issue is to allow the page table to contain only one mapping of a virtual address to the shared physical address.
- references to virtual addresses that are not mapped result in page faults

Inverted Page Table Architecture



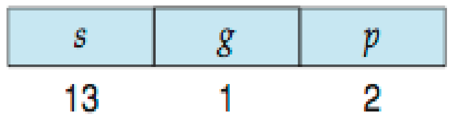
Used in 64-bit UltraSPARC (from Sun Micro Systems) and PowerPC (created by Apple-IBM-Motorola alliance)

Example: The Intel IA-32 Architecture

- Supports both segmentation and segmentation with paging
  - Each segment can be as large as 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**))
  - Machine has 6 segment registers
    - Segment register points to the appropriate entry in the LDT or GDT

Example: The Intel IA-32 Architecture (Cont.)

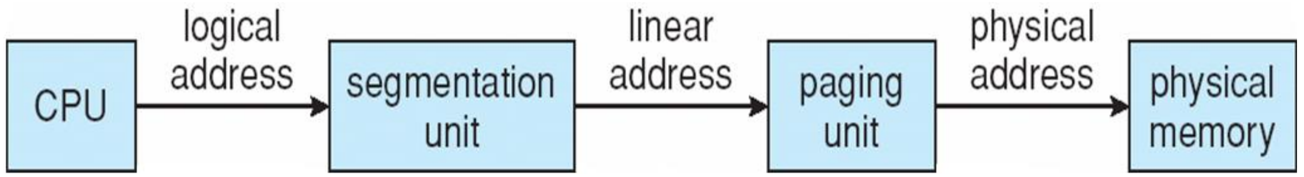
- CPU generates logical address
  - Selector given to segmentation unit
    - Which produces linear addresses
    - The logical address is a pair (**selector, offset**), where the selector is a 16-bit number:



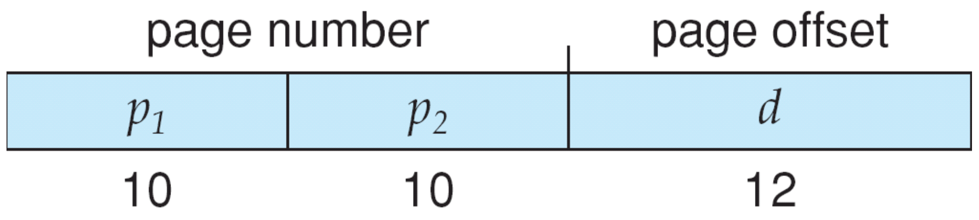
    - s indicates the segment number, g indicates whether the segment is in the GDT or LDT, p deals with protection
    - The offset is a 32-bit number specifying the location of the byte within the segment in question
  - Linear address given to paging unit
    - Which generates physical address in main memory
    - Paging units form equivalent of MMU
    - Pages sizes can be 4 KB or 4 MB

Logical to Physical Address Translation in IA-32

- Memory management in IA-32 systems is divided into two components – segmentation and paging
- Segmentation and Paging units form the equivalent of MMU

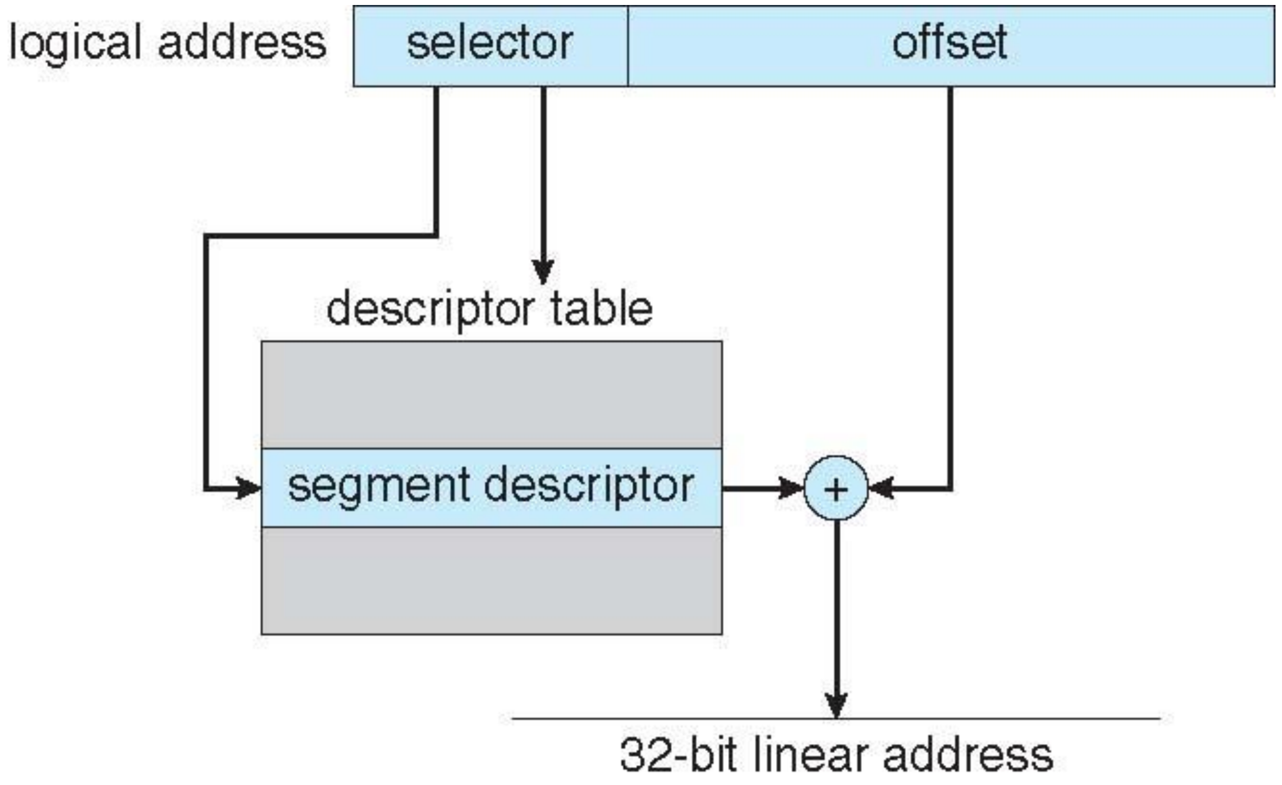


- IA-32 architecture allows a page size of either 4 KB or 4 MB.
- For 4-KB pages, IA-32 uses a two-level paging scheme in the division of the 32-bit linear address as shown below



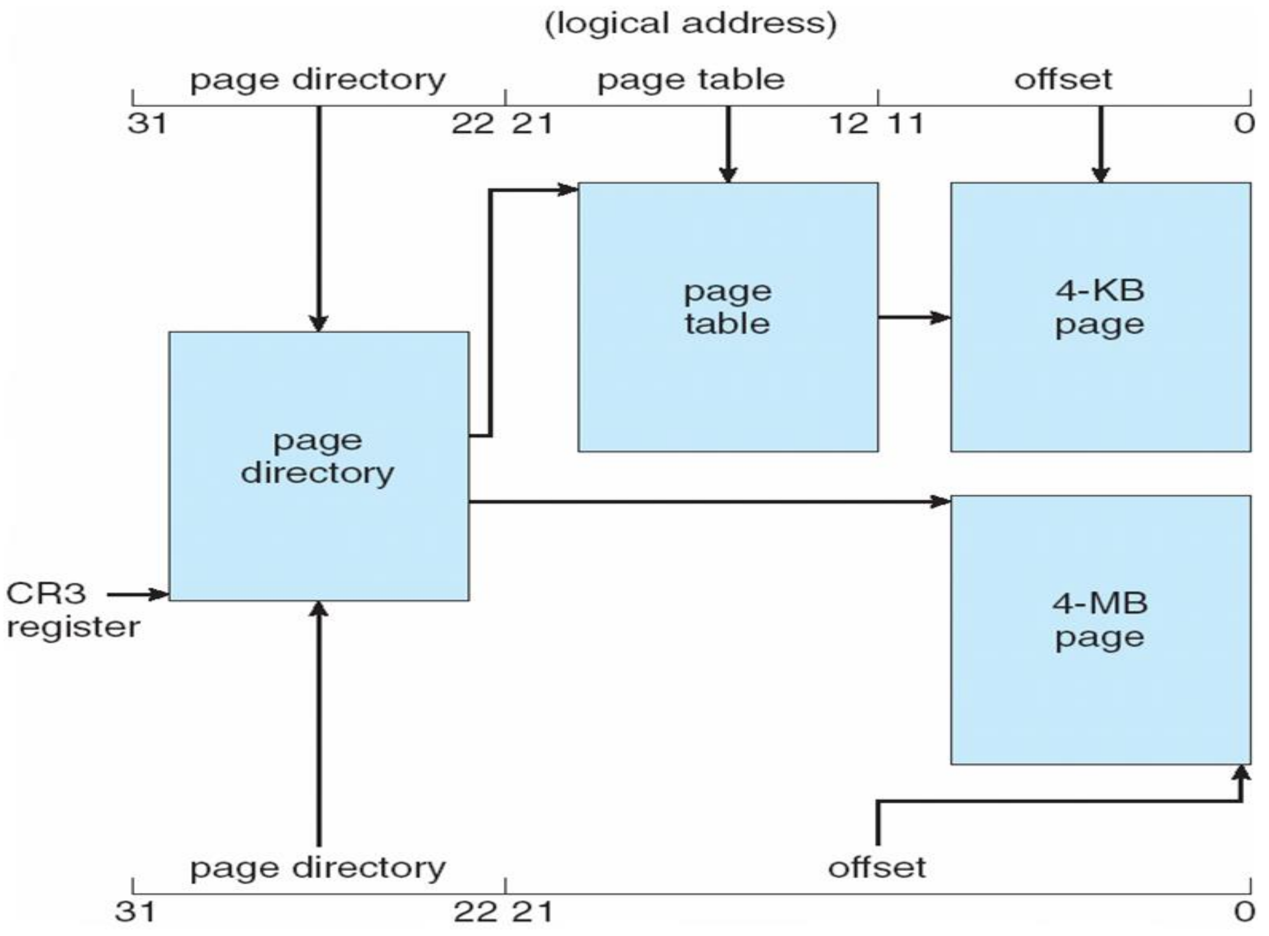
Intel IA-32 Segmentation

- The base and limit information about the segment in question is used to generate a linear address.
- The paging unit turns this linear address into a physical address.



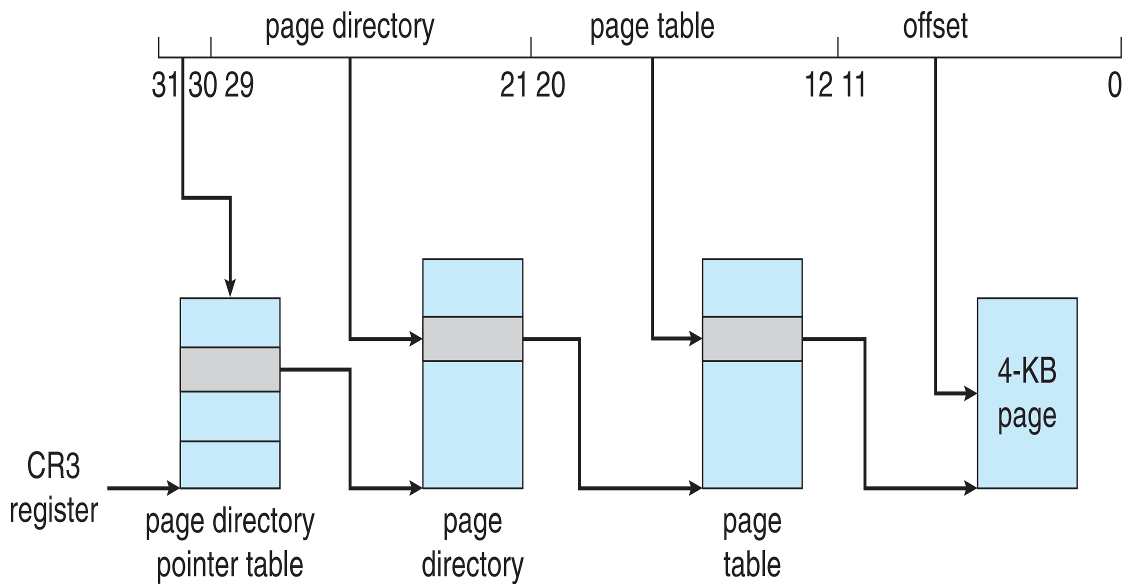
Intel IA-32 Paging Architecture

- The 10 high-order bits reference an entry in the outermost page table (page directory)
- The CR3 register points to the page directory for the current process.
- The page directory entry points to an inner page table
- Finally, the low-order bits 0–11 refer to the offset in the 4-KB page pointed to in the page table.
- One entry in the page directory is the Page Size flag, which—if set—indicates that the size of the page frame is 4 MB and not the standard 4 KB, so the 22 low-order bits in the linear address would refer to the offset in the 4-MB page frame.



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 from 32 to 64-bits in size
    - So base address of page tables and page frames extended from 20 to 24 bits.
  - Combined with 12-bit offset, Net effect is increasing address space to 36 bits – 64GB of physical memory



PES UNIVERSITY ONLINE

THANK YOU

Chandravva Hebbi
Department of Computer Science Engineering

chandravvahebbi@pes.edu