OPERATING SYSTEMS

Memory Management

Chandravva Hebbi
Department of Computer Science

OPERATING SYSTEMS

Case Study: Windows and Linux Memory Management

Chandravva Hebbi
Department of Computer Science

OPERATING SYSTEMS

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

Example: The Intel 32 and 64-bit Architectures

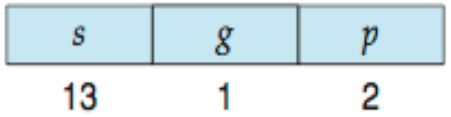
- Dominant industry chips
- Pentium CPUs are 32-bit and called IA-32 architecture
- Current Intel CPUs are 64-bit and called IA-64 architecture
- Many variations/versions in the chips exist, only the main ideas of memory management are discussed here
- Intel 64-bit chips are based on x86-64 architecture
- Most popular PC operating systems run on Intel Chips
  - Linux runs on several other architectures as well
  - Mobile Systems mostly use ARM 32-bit architecture (Advanced RISC Machine, originally Acorn RISC Machine)

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.)

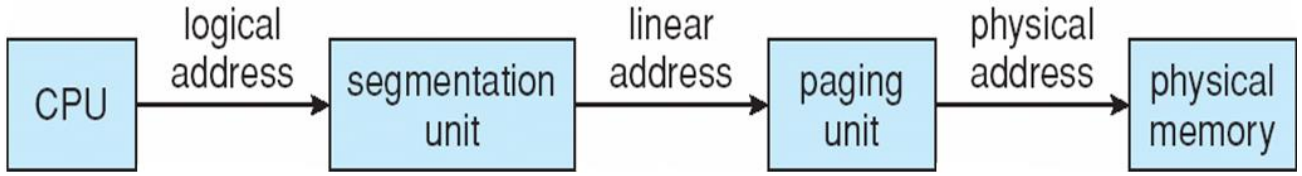
- CPU generates logical address
  - Selector given to segmentation unit
    - Which produces linear addresses
    - The logical address is a pair (**selector, offset**), where the selector is a 16-bit number:



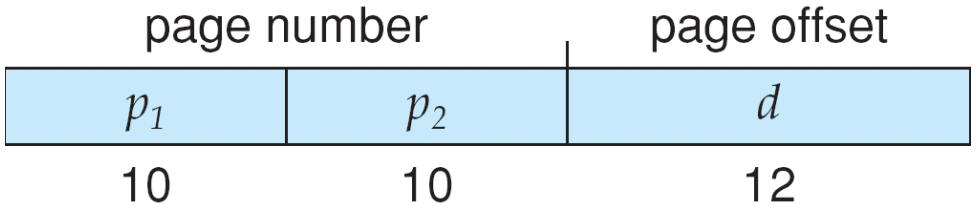
    - s indicates the segment number, g indicates whether the segment is in the GDT or LDT, p deals with protection
    - The offset is a 32-bit number specifying the location of the byte within the segment in question
  - Linear address given to paging unit
    - Which generates physical address in main 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

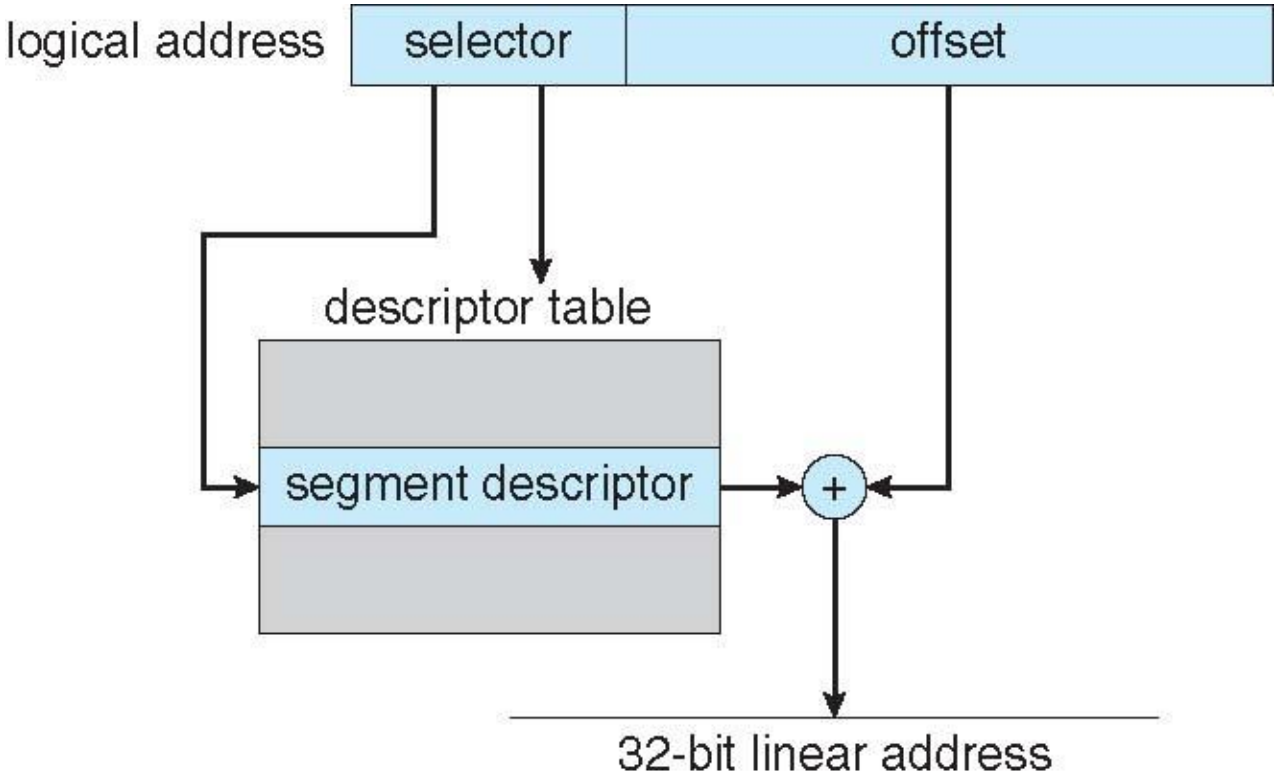


- IA-32 architecture allows a page size of either 4 KB or 4 MB.
- For 4-KB pages, IA-32 uses a two-level paging scheme in the division of the 32-bit linear address as shown below



Intel IA-32 Segmentation

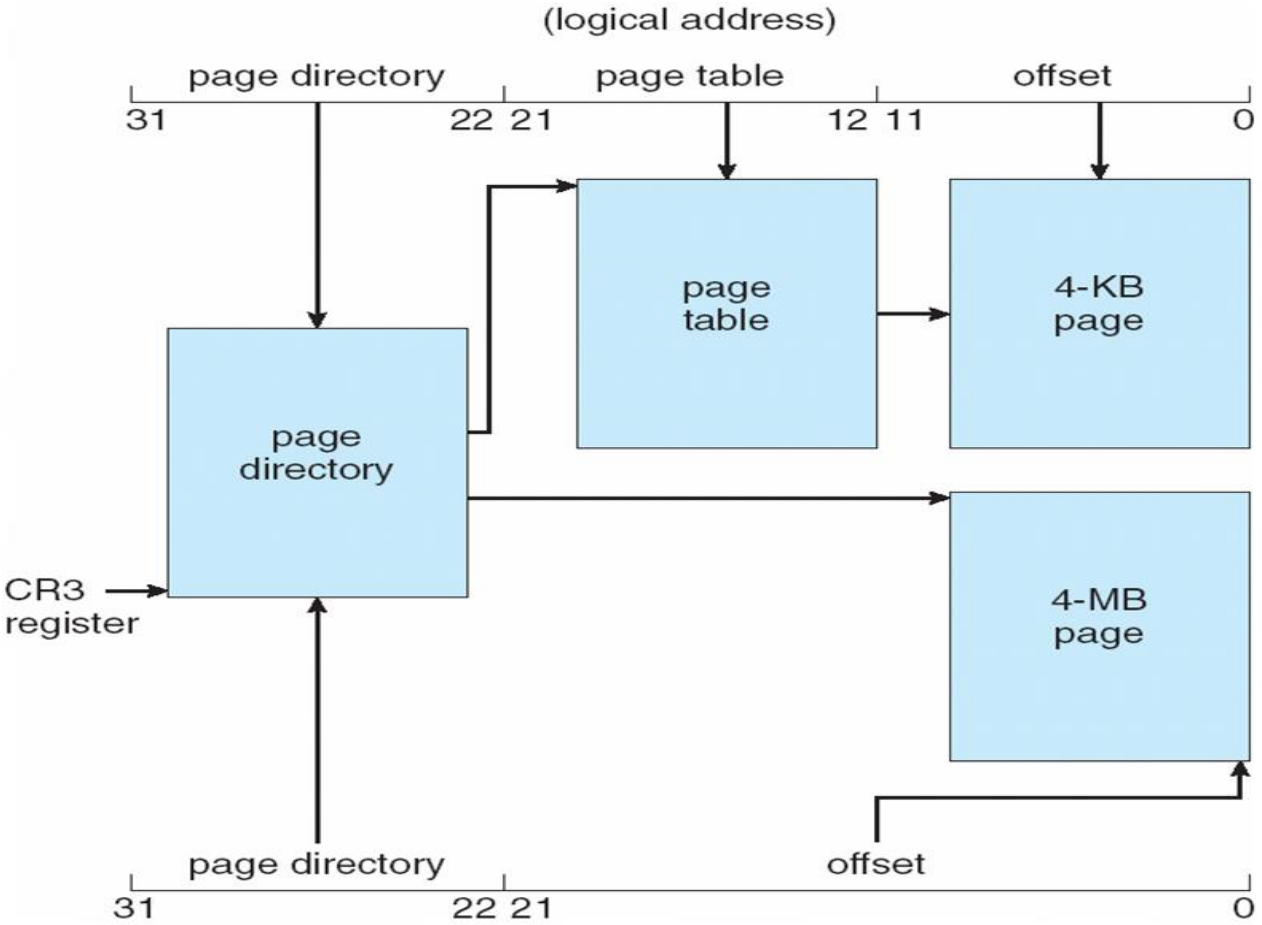
- The base and limit information about the segment in question is used to generate a **linear address**.
- The paging unit turns this linear address into a physical address.



OPERATING SYSTEMS

Intel IA-32 Paging Architecture

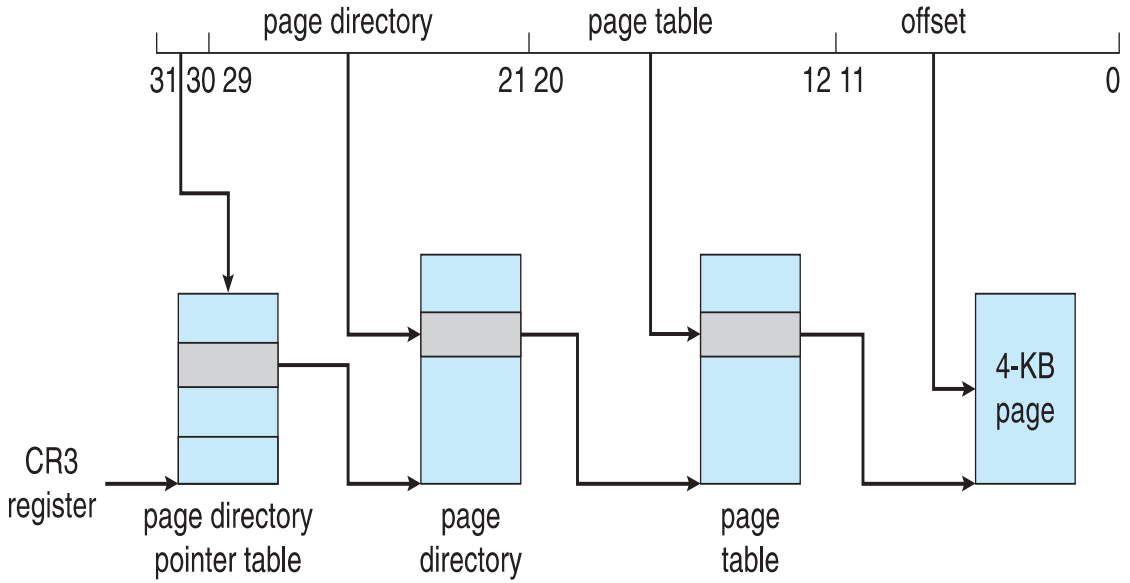
- The 10 high-order bits reference an entry in the outermost page table (page directory)
- The CR3 register points to the page directory for the current process.
- The page directory entry points to an inner page table
- Finally, the low-order bits 0–11 refer to the offset in the 4-KB page pointed to 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.



OPERATING SYSTEMS

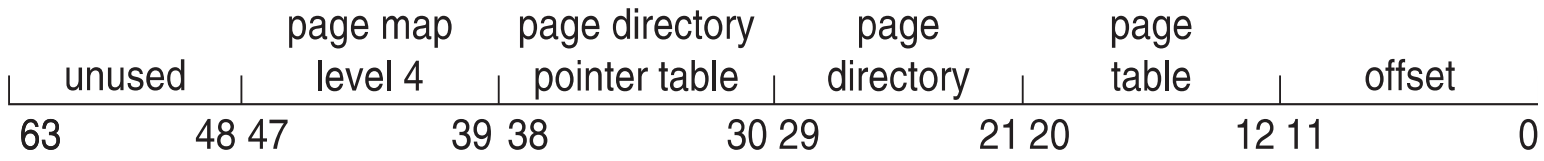
Intel IA-32 Page Address Extensions

- 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



Intel x86-64

- Current generation Intel x86 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



OPERATING SYSTEMS

Windows Memory Management

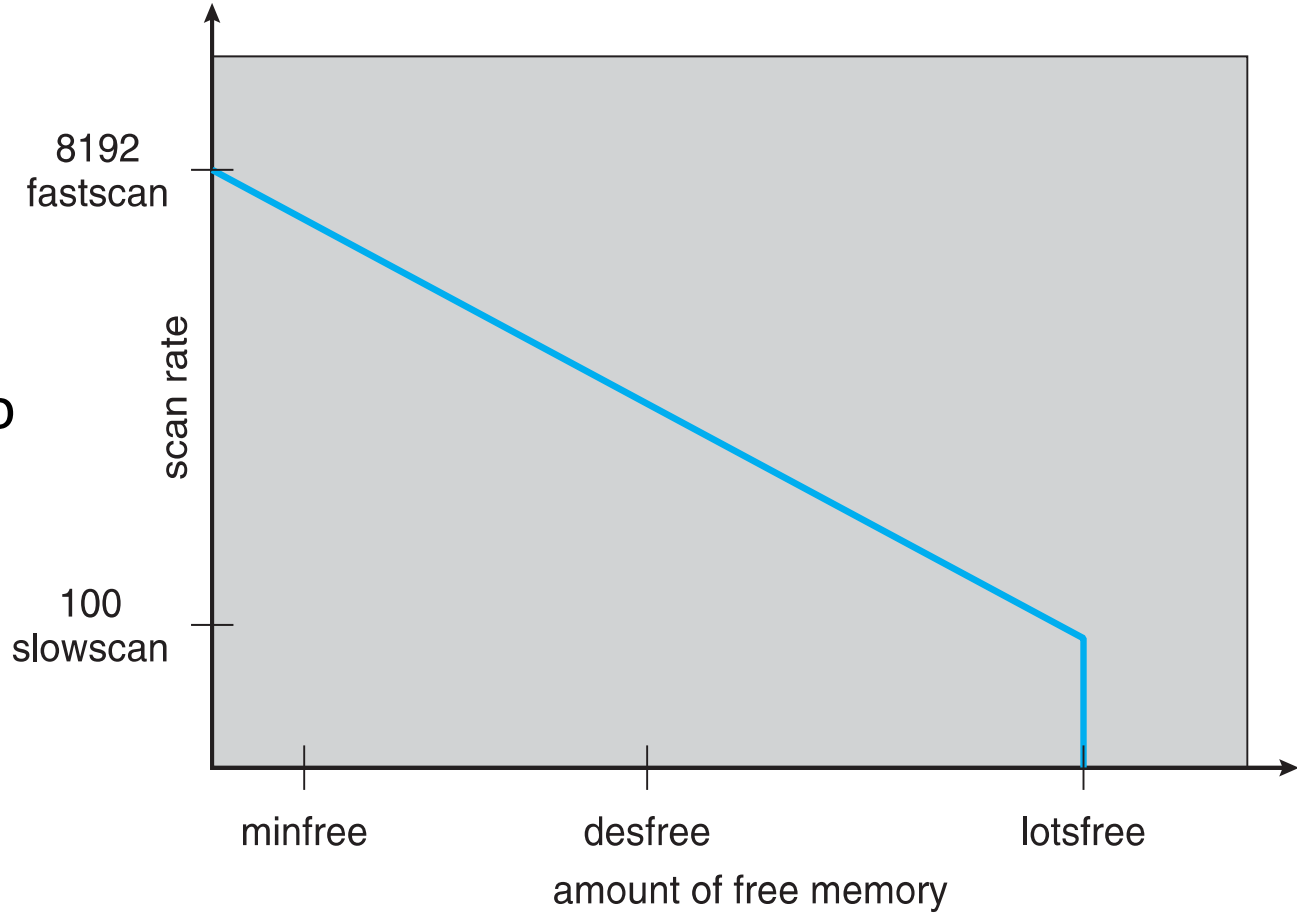
- Uses demand paging with **clustering**. Clustering brings in pages surrounding the faulting page
- Processes are assigned **working set minimum** and **working set maximum**
- Working set minimum is the minimum number of pages the process is guaranteed to have in memory
- A process may be assigned as many pages up to its working set maximum
- When the amount of free memory in the system falls below a threshold, **automatic working set trimming** is performed to restore the amount of free memory
- Working set trimming removes pages from processes that have pages in excess of their working set minimum

Solaris

- Maintains a list of free pages to assign faulting processes
- **Lotsfree** – threshold parameter (amount of free memory) to begin paging
  - Initial trigger for system paging to begin.
  - If the number of free pages falls below **lotsfree**, start **pageout**
- **Desfree** – threshold parameter to increasing paging
  - Amount of memory **desired** to be free at all times on the system.
- **Minfree** – threshold parameter to being swapping
  - Minimum acceptable memory level.
- Paging is performed by **pageout** process
- **Pageout** scans pages using modified clock algorithm (uses front hand & back hand clock)

Solaris 2 Page Scanner

- **Scanrate** is the rate at which pages are scanned. This ranges from **slowscan** (like 100 pages per sec) to **fastscan** (max of 8192 pages per sec)



The relationship of lotsfree is greater than desfree, which is greater than minfree, should be maintained at all times.

Solaris 2 Page Scanner

- **Pageout** is called more frequently depending upon the amount of free memory available
- **Priority paging** gives priority to process code pages over file-system page cache
  - Solaris allows processes and the page cache to share unused memory.
  - So, system performing many I/O operations can use most of the available memory for caching pages
  - Page scanner will reclaim pages from processes – rather than from the Page cache – when free memory ran low.

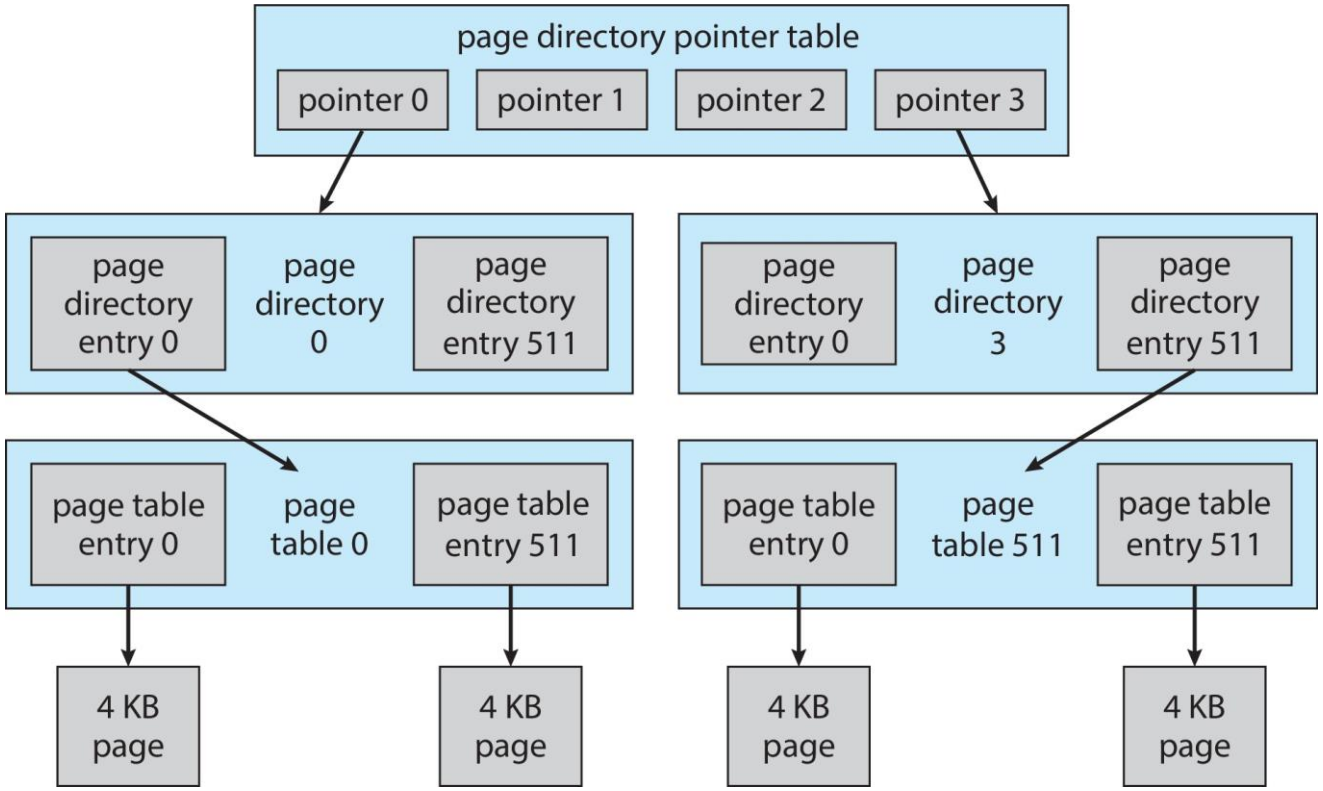
SUPPLEMENTARY SLIDES FROM HERE FOR ADDITIONAL READING BUT NOT INCLUDED FOR ISA/ESA

Windows Executive — Virtual Memory Manager

- The design of the VM manager assumes that the underlying hardware supports virtual to physical mapping a paging mechanism, transparent cache coherence on multiprocessor systems, and virtual addressing aliasing.
- The VM manager in Windows 7 uses a page-based management scheme with a page size of 4 KB.
- The Windows 7 VM manager uses a two step process to allocate memory
  - The first step reserves a portion of the process's address space
  - The second step commits the allocation by assigning space in the system's paging file(s)

OPERATING SYSTEMS

Virtual-Memory Layout

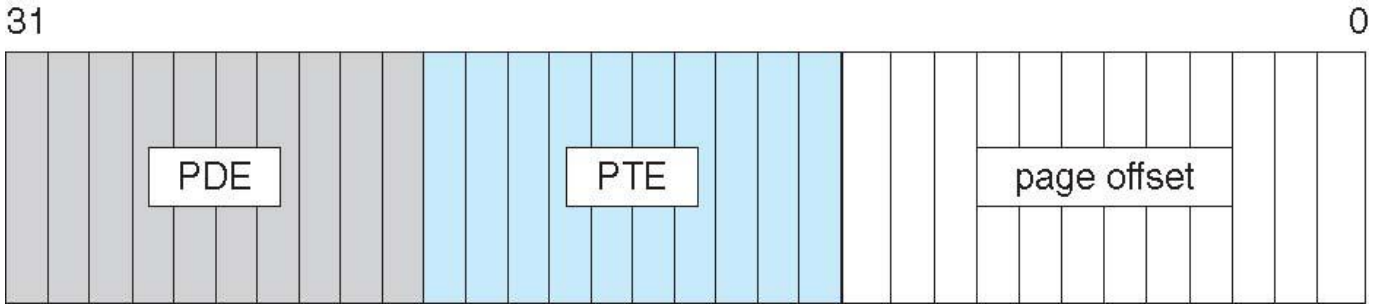


Virtual Memory Manager (Cont.)

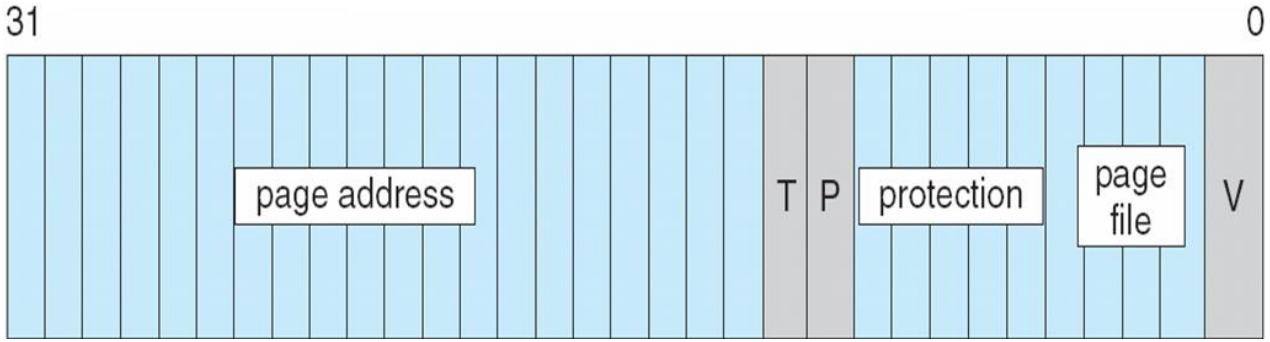
- The virtual address translation in Windows 7 uses several data structures
  - Each process has a *page directory* that contains 1024 *page directory entries* of size 4 bytes.
  - Each page directory entry points to a *page table* which contains 1024 *page table entries* (PTEs) of size 4 bytes.
  - Each PTE points to a 4 KB *page frame* in physical memory.
- A 10-bit integer can represent all the values form 0 to 1023, therefore, can select any entry in the page directory, or in a page table.
- This property is used when translating a virtual address pointer to a bye address in physical memory.
- A page can be in one of six states: valid, zeroed, free, standby, modified and bad.

Virtual-to-Physical Address Translation

- 10 bits for page directory entry, 10 bits for page table entry, and 12 bits for byte offset in page



Page File Page-Table Entry



5 bits for page protection, 20 bits for page frame address, 4 bits to select a paging file, and 3 bits that describe the page state.
V=0, T=0, P=0

OPERATING SYSTEMS

Linux Memory Management

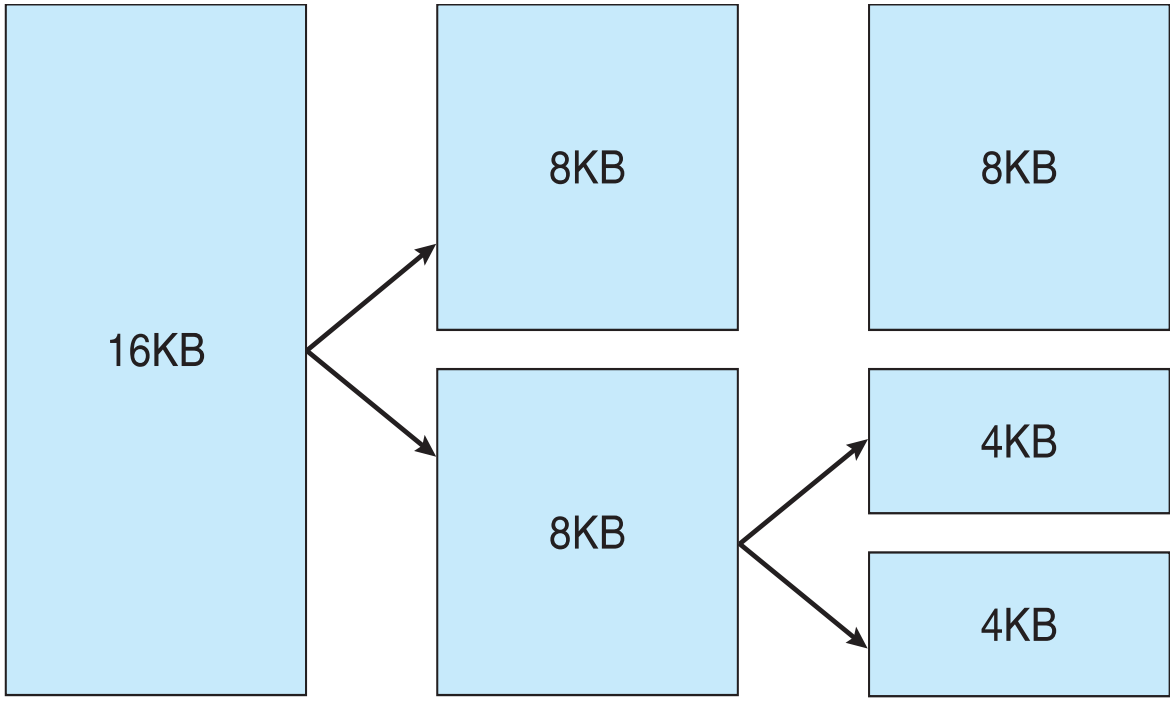
- Linux's physical memory-management system deals with allocating and freeing pages, groups of pages, and small blocks of memory
- It has additional mechanisms for handling virtual memory, memory mapped into the address space of running processes
- Splits memory into four different **zones** due to hardware characteristics
  - Architecture specific, for example on x86:

zone	physical memory
ZONE_DMA	< 16 MB
ZONE_NORMAL	16 .. 896 MB
ZONE_HIGHMEM	> 896 MB

Managing Physical Memory

- The page allocator allocates and frees all physical pages; it can allocate ranges of physically-contiguous pages on request
- The allocator uses a buddy-heap algorithm to keep track of available physical pages
  - Each allocatable memory region is paired with an adjacent partner
  - Whenever two allocated partner regions are both freed up they are combined to form a larger region
  - If a small memory request cannot be satisfied by allocating an existing small free region, then a larger free region will be subdivided into two partners to satisfy the request

Splitting of Memory in a Buddy Heap

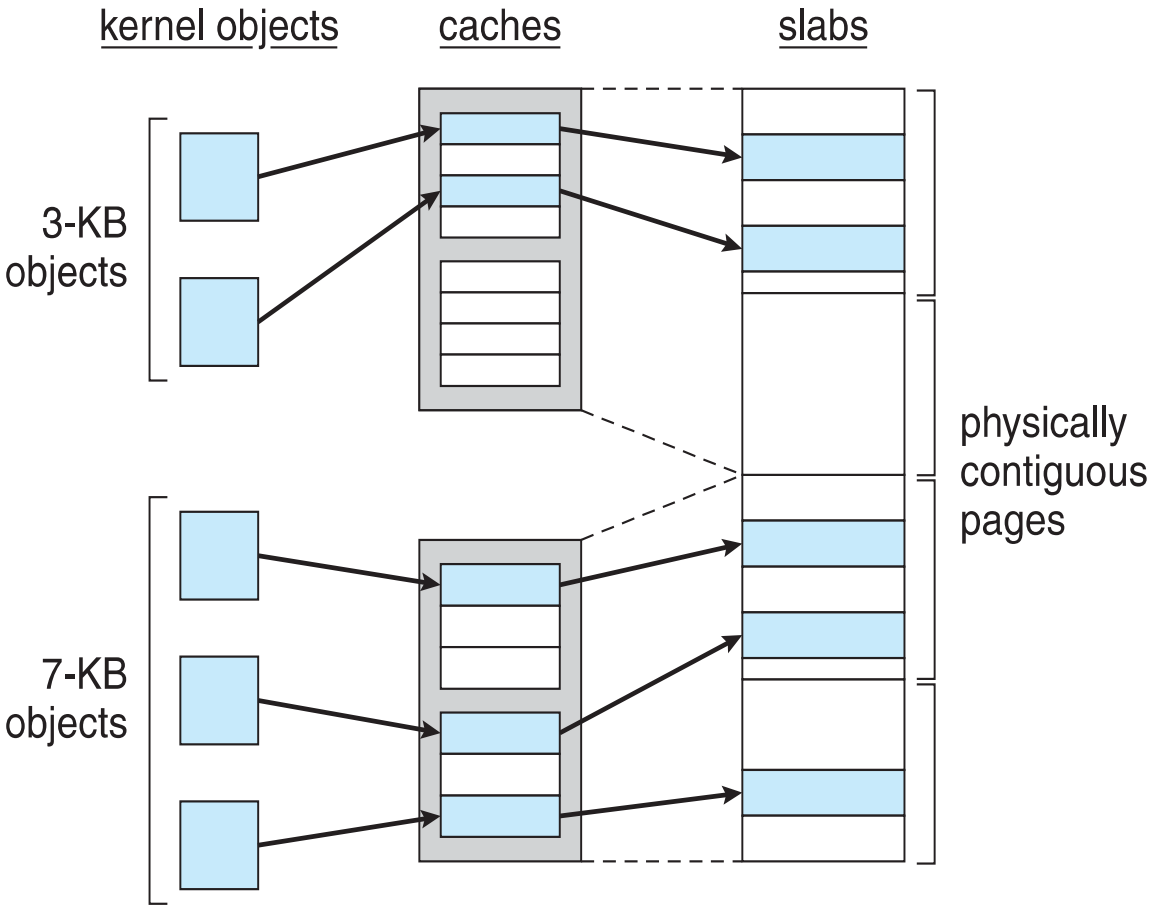


Managing Physical Memory (Cont.)

- Memory allocations in the Linux kernel occur either statically (drivers reserve a contiguous area of memory during system boot time) or dynamically (via the page allocator)
- Also uses **slab allocator** for kernel memory
- **Page cache** and virtual memory system also manage physical memory
  - Page cache is kernel's main cache for files and main mechanism for I/O to block devices
  - Page cache stores entire pages of file contents for local and network file I/O

OPERATING SYSTEMS

Slab Allocator in Linux



Virtual Memory

- The VM system maintains the address space visible to each process: It creates pages of virtual memory on demand, and manages the loading of those pages from disk or their swapping back out to disk as required.
- The VM manager maintains two separate views of a process's address space:
  - A logical view describing instructions concerning the layout of the address space
    - The address space consists of a set of non-overlapping regions, each representing a continuous, page-aligned subset of the address space
  - A physical view of each address space which is stored in the hardware page tables for the process

Virtual Memory (Cont.)

- Virtual memory regions are characterized by:
  - The backing store, which describes from where the pages for a region come; regions are usually backed by a file or by nothing (**demand-zero memory**)
  - The region's reaction to writes (page sharing or copy-on-write

Kernel Virtual Memory

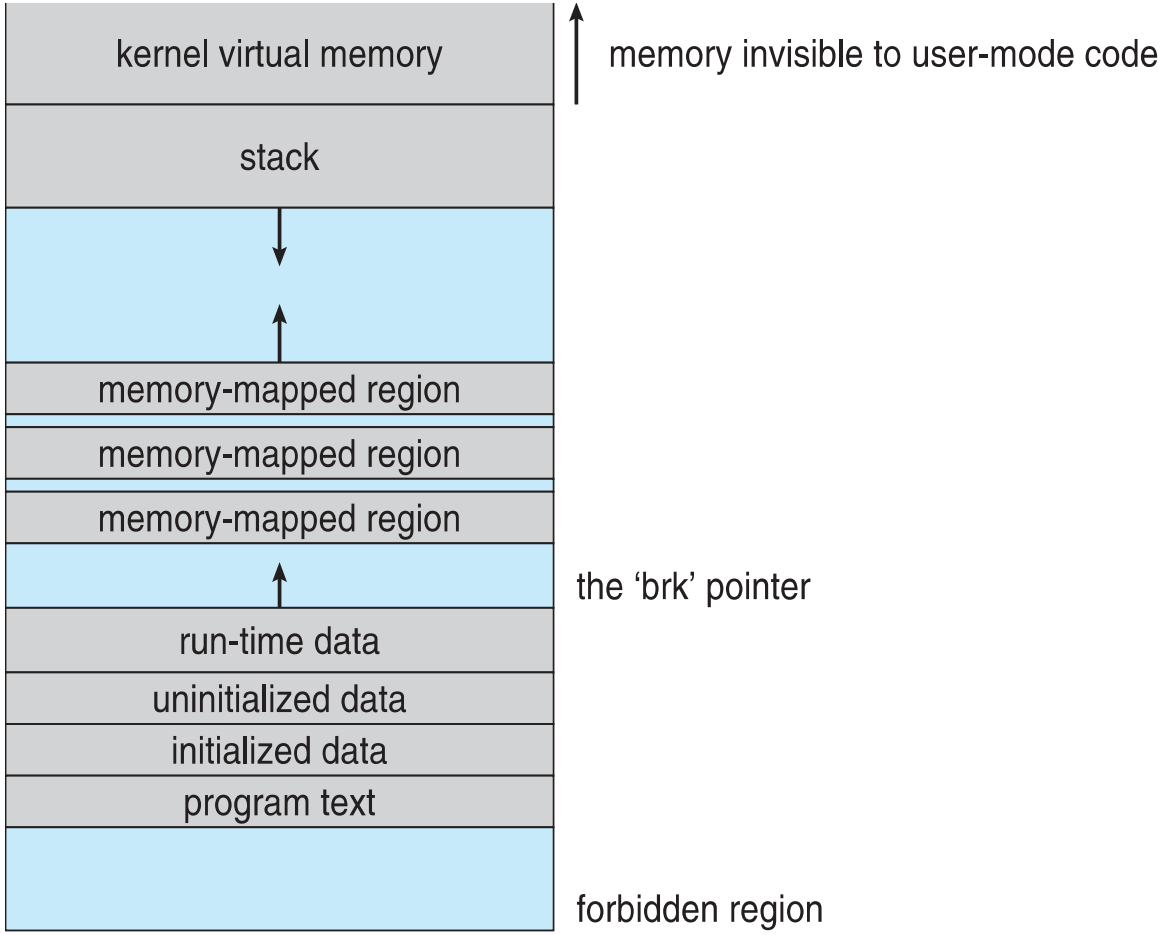
- The Linux kernel reserves a constant, architecture-dependent region of the virtual address space of every process for its own internal use
- This kernel virtual-memory area contains two regions:
  - A static area that contains page table references to every available physical page of memory in the system, so that there is a simple translation from physical to virtual addresses when running kernel code
  - The reminder of the reserved section is not reserved for any specific purpose; its page-table entries can be modified to point to any other areas of memory

Executing and Loading User Programs

- Linux maintains a table of functions for loading programs; it gives each function the opportunity to try loading the given file when an exec system call is made
- The registration of multiple loader routines allows Linux to support both the **ELF** and **a.out** binary formats
- Initially, binary-file pages are mapped into virtual memory
  - Only when a program tries to access a given page will a page fault result in that page being loaded into physical memory
- An ELF-format binary file consists of a header followed by several page-aligned sections
  - The ELF loader works by reading the header and mapping the sections of the file into separate regions of virtual memory

OPERATING SYSTEMS

Memory Layout for ELF Programs



THANK YOU

Chandravva Hebbi
Department of Computer Science Engineering

chandravvahebbi@pes.edu