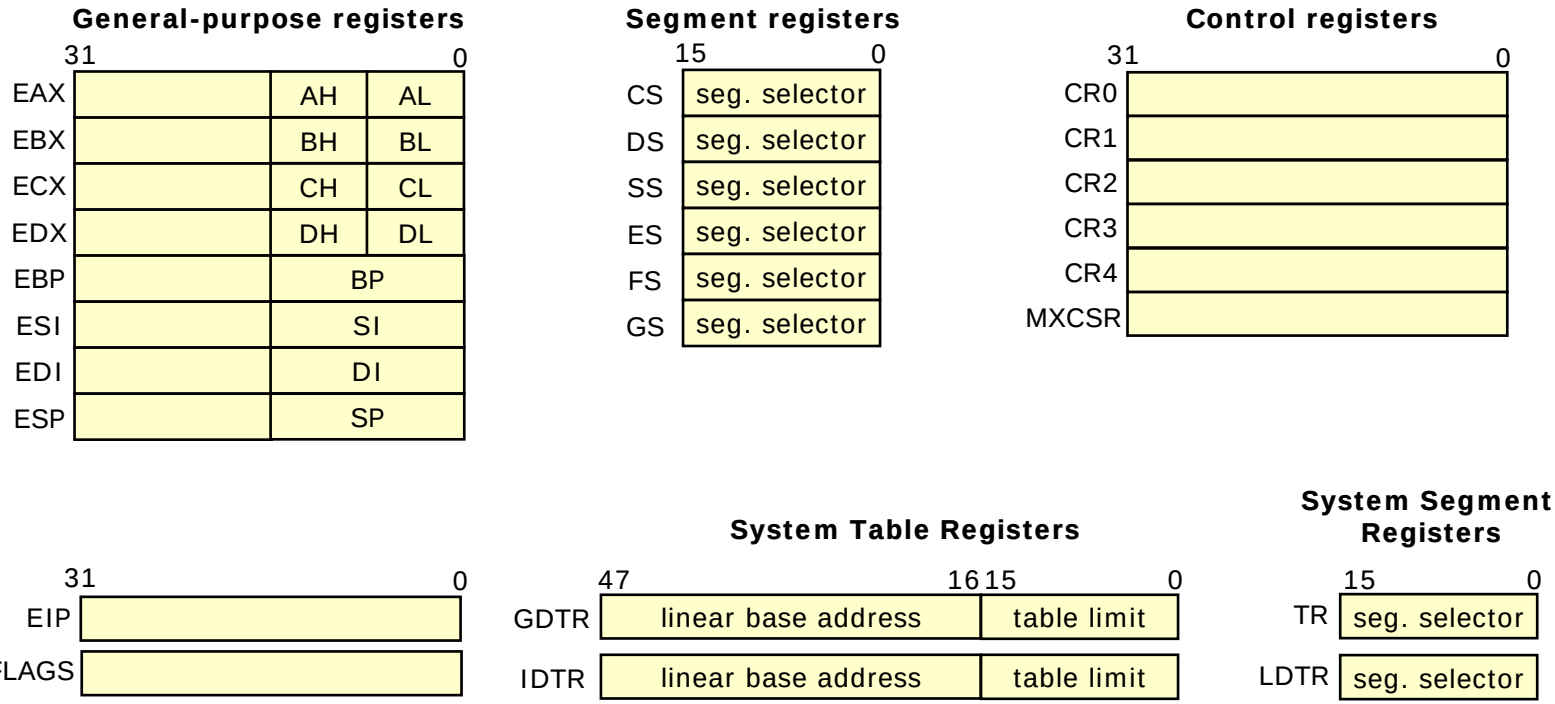
CS530: Graduate Operating Systems (Spring 2006)

IA32/Linux Virtual Memory Architecture

KAIST

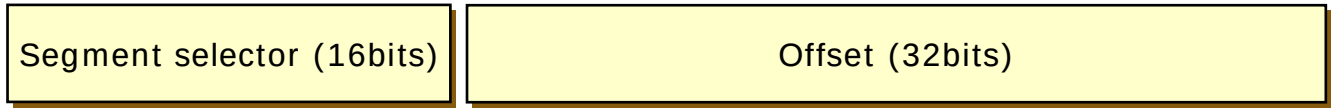
IA32 VM Architecture (1)

- **Basic execution environment**



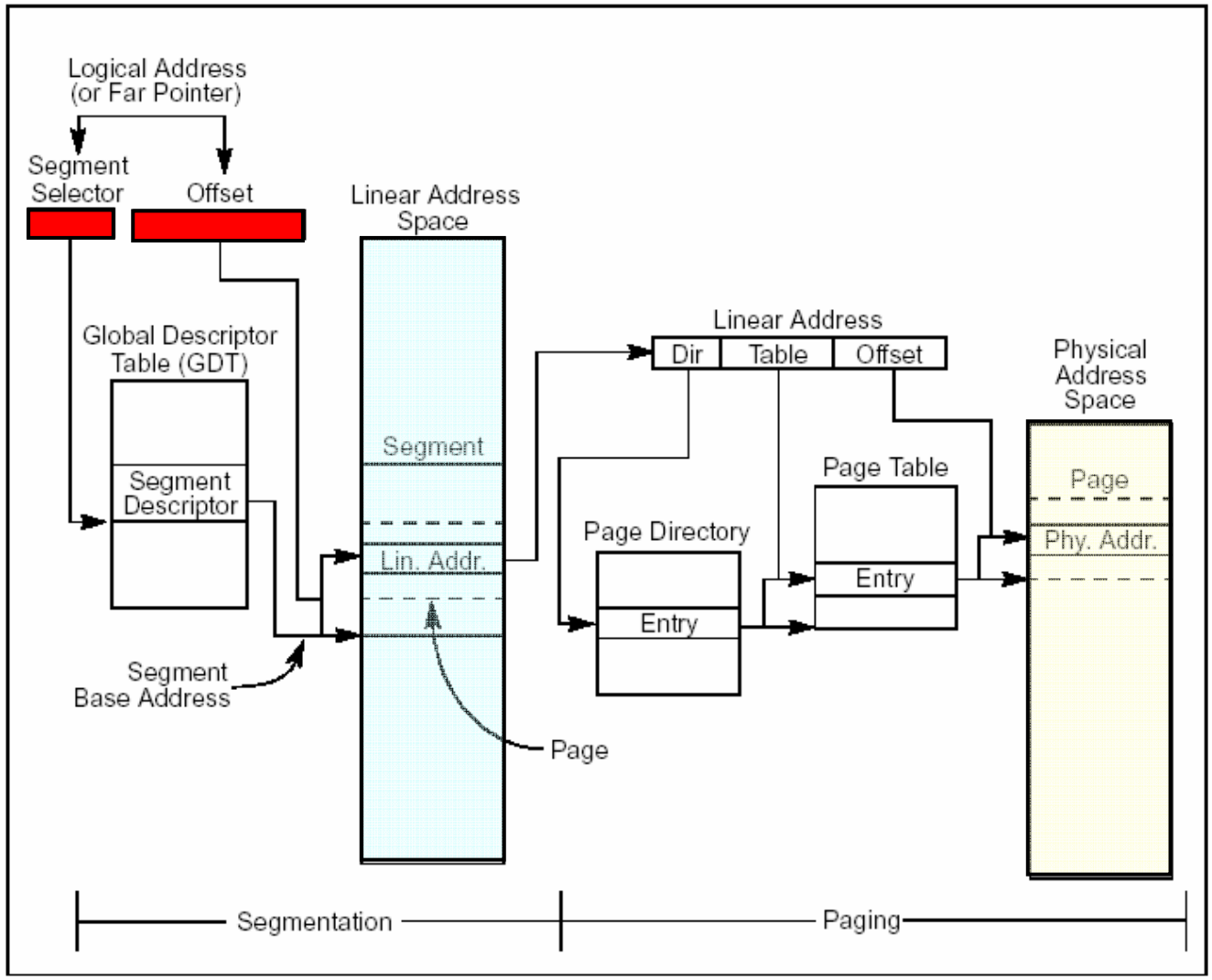
IA32 VM Architecture (2)

- **Logical address (far pointer)**
  - User's view, segmented



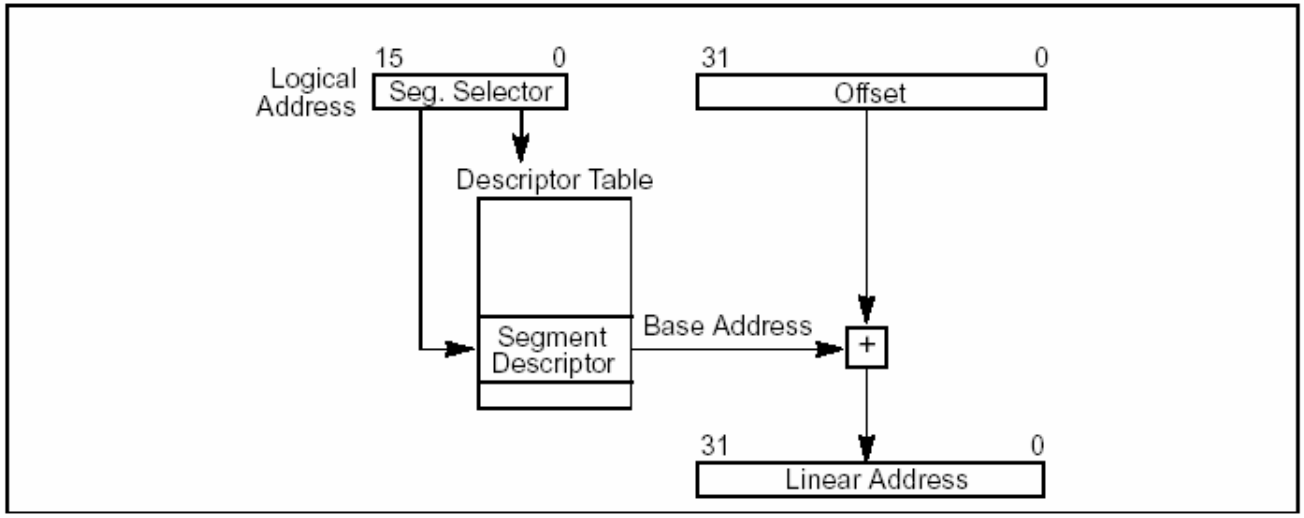
- **Linear address**
  - 32-bit, flat
- **Physical address**
  - 32-bit, flat
  - Pentium Pro and later processors support an extension of the physical address space to 2^{36} bytes.
  - Invoked with the physical address extension (PAE) flag located in CR4 register.

IA32 VM Architecture (3)



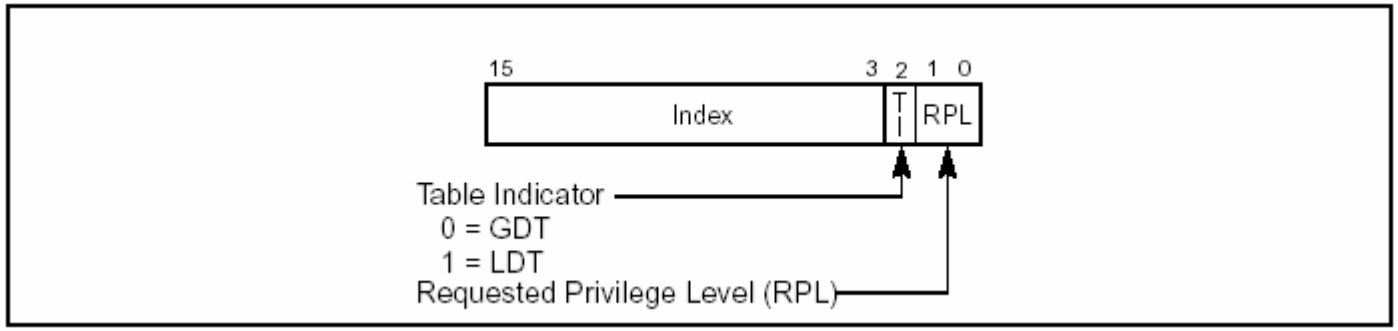
Segmentation (1)

- **Logical to linear address**
  - Locate the segment descriptor for the segment in the GDT or LDT using the segment selector.
  - Examine the segment descriptor to check the access rights and the offset is within the limits.
  - Adds the base address of the segment from the segment descriptor to the offset to form a linear address.



Segmentation (2)

- **Segment selector**



- **Segment registers**

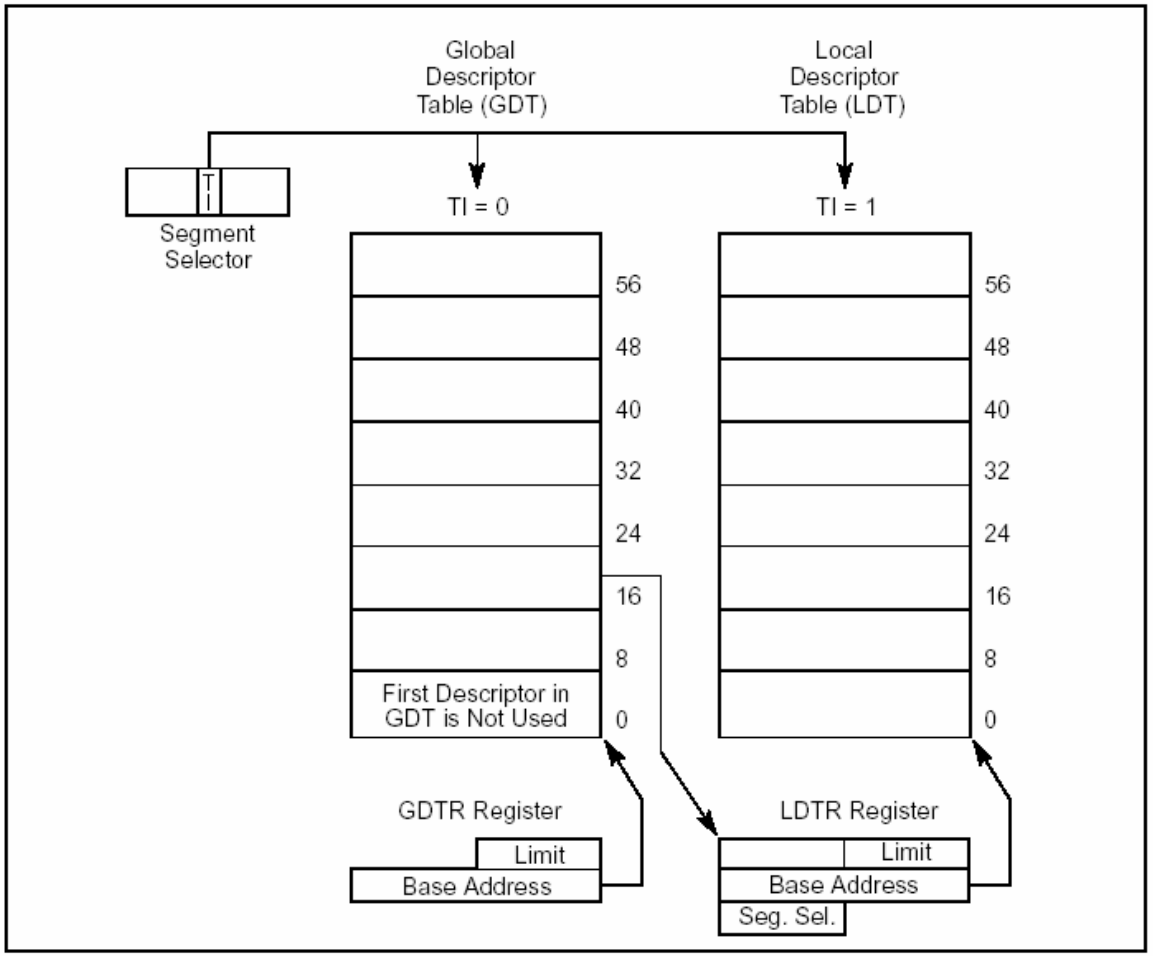
Visible Part	Hidden Part	
Segment Selector	Base Address, Limit, Access Information	CS
		SS
		DS
		ES
		FS
		GS

Segmentation (3)

- **Segment descriptor tables**
  - Each system must have one GDT (Global Descriptor Table), which may be used for all programs and tasks.
  - Optionally, one or more LDTs (Local Descriptor Tables) can be defined.
  - GDT is not a segment, but a data structure in the linear address space pointed to by the GDTR register.
  - LDT is located in a system segment of the LDT type.
  - GDT must contain a segment descriptor for the LDT segment.
  - The first descriptor in GDT is not used.
  - The LDTR register caches the segment descriptor of the current LDT segment.

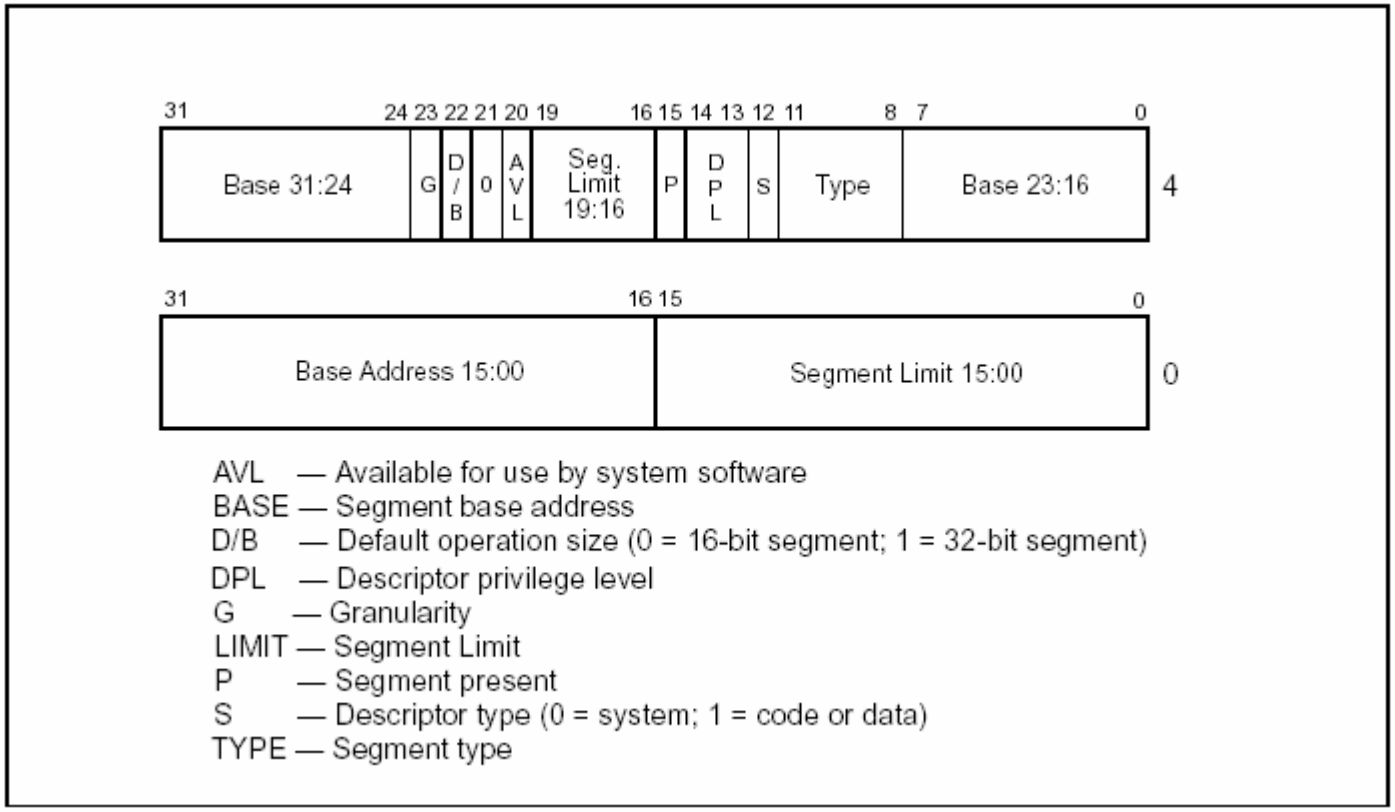
Segmentation (4)

- **Global and local descriptor tables**



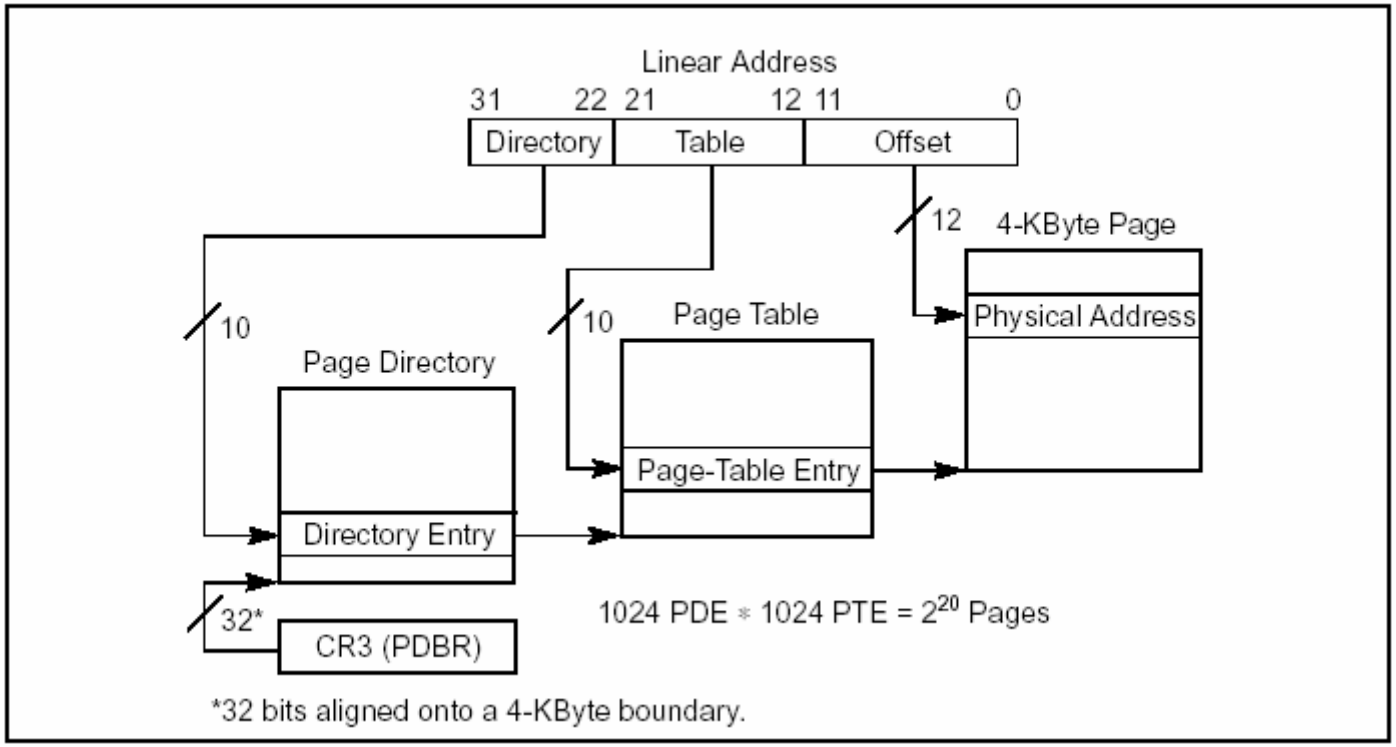
Segmentation (5)

- **Segment descriptor**



Paging (1)

- **Linear to physical address**
  - The physical address of the current page directory is stored in the CR3 register (a.k.a. page directory base register or PDBR).

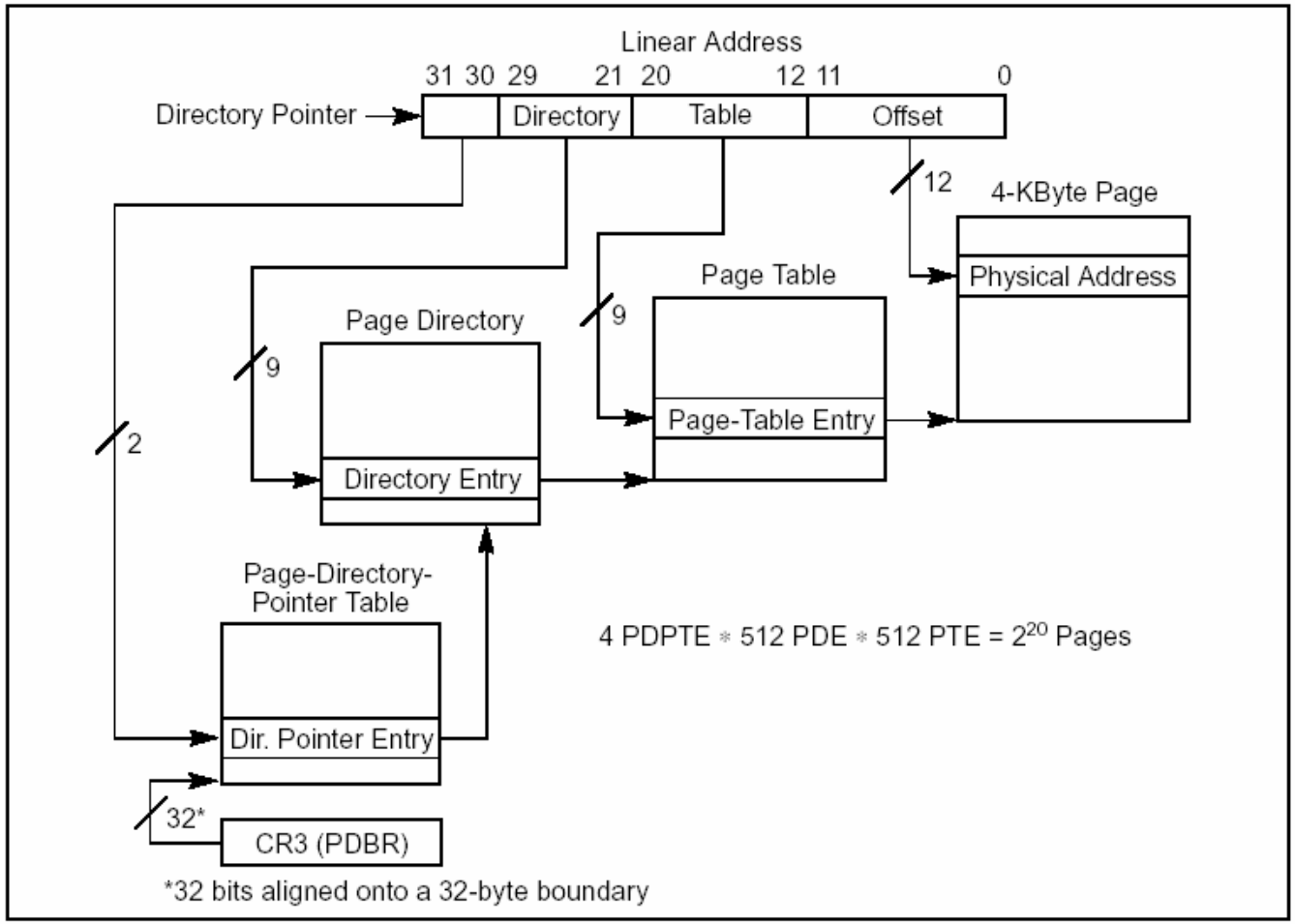


Paging (2)

- **Page tables and directories**
  - Page directory
    - An array of 32-bit page-directory entries (PDEs) contained in a 4KB page (1024 PDEs/page).
  - Page table
    - An array of 32-bit page-table entries (PTEs) contained in a 4KB page (1024 PTEs/page).
    - Page tables are not used for 2MB or 4MB pages.
  - Page
    - Supports page sizes of 4KB, 2MB, and 4MB.
  - Page-directory-pointer table
    - An array of four 64-bit entries pointing to a page directory.
    - Only used when the physical address extension is enabled.

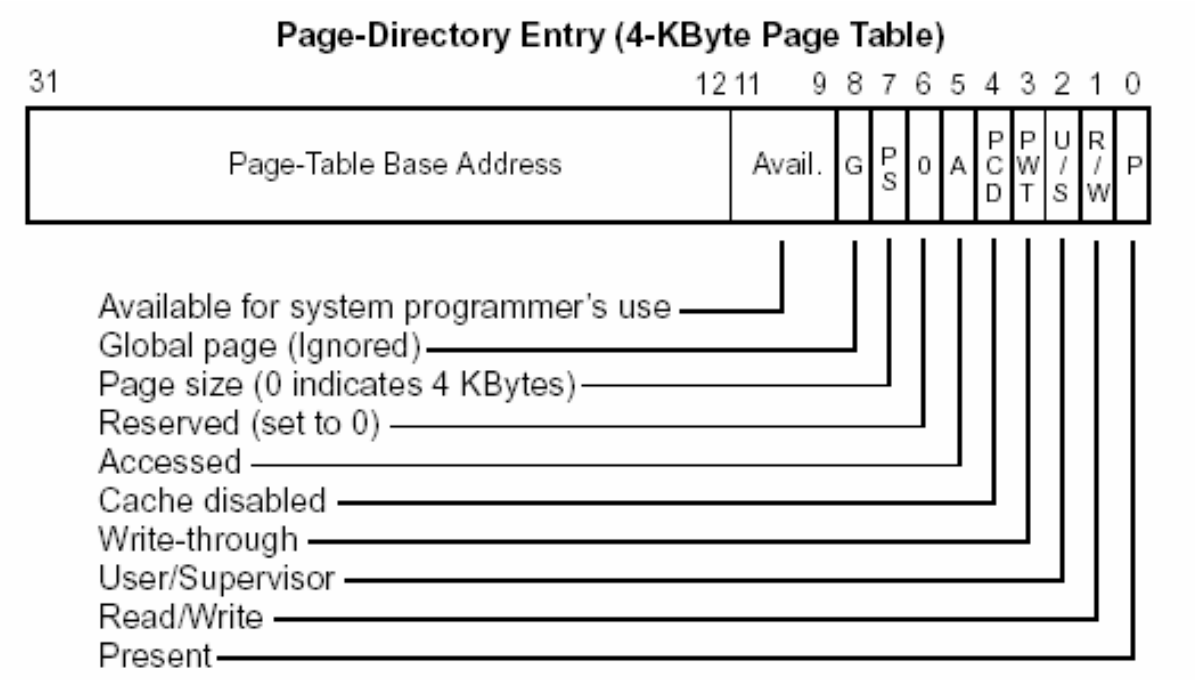
Paging (3)

- **Linear to physical address (PAE enabled)**



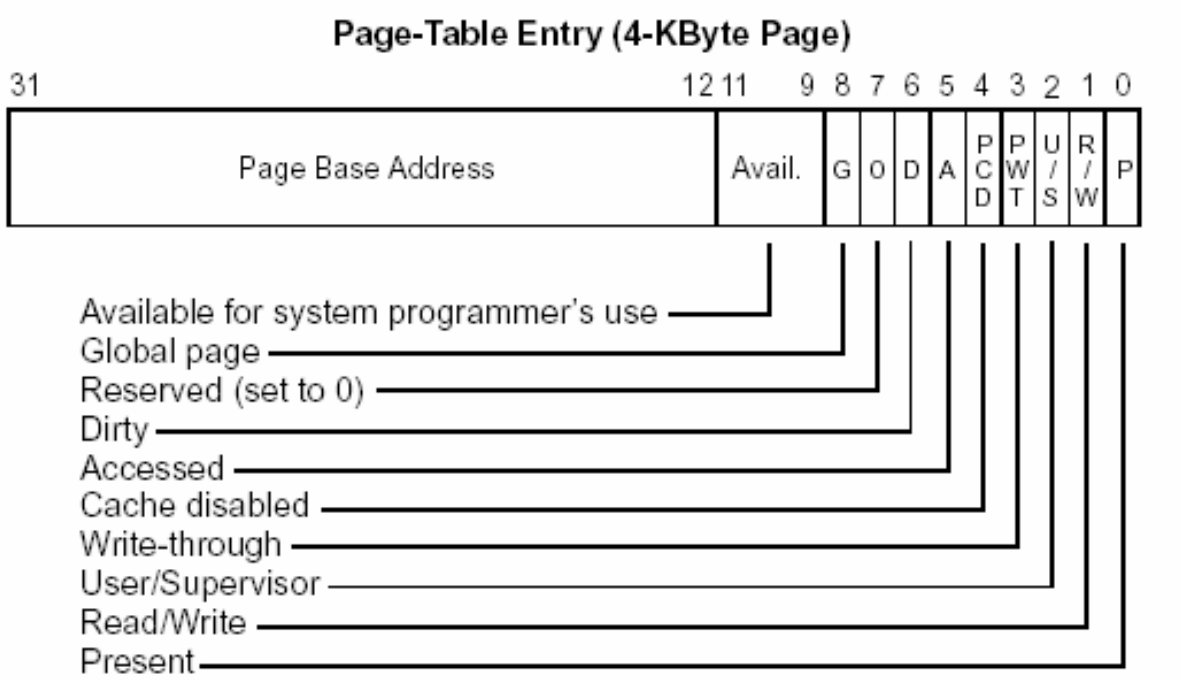
Paging (4)

- **Page directory entry (PDE)**



Paging (5)

- **Page table entry (PTE)**



Paging (6)

- **TLBs**
  - The P6 family and Pentium processors have separate TLBs for the data and instruction. (DTLB & ITLB)
  - All TLBs are automatically invalidated if the PDBR register is loaded.
    - by explicit MOV instruction
    - implicitly by executing a task switch
  - A specific page-table entry in the TLB can be invalidated using INVLPG instruction.
  - The page global enable (PGE) flag in CR4 and the global (G) flag of a PDE or PTE can be used to prevent frequently used pages from being automatically invalidated.

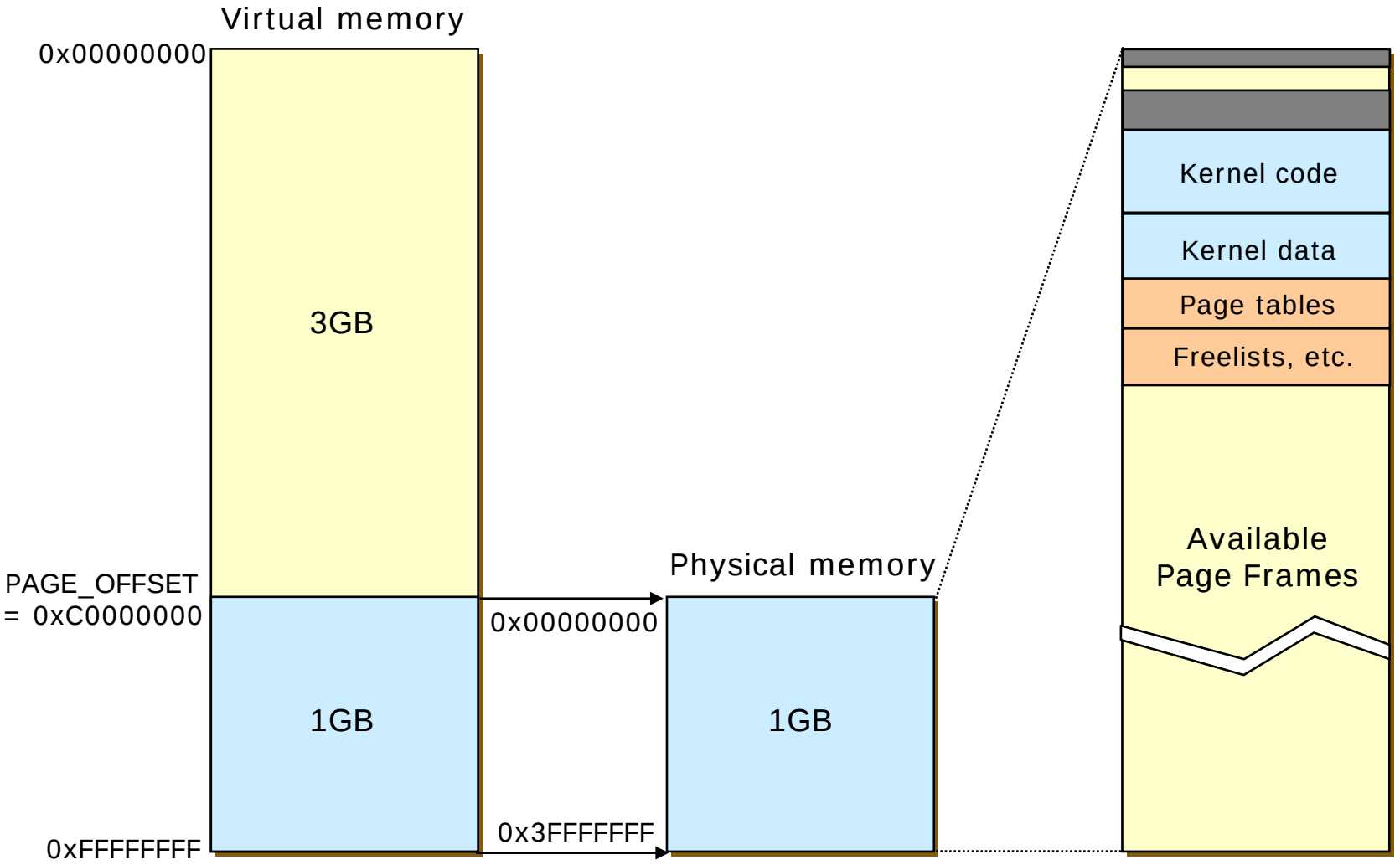
IA32 VM References

- **For more information, see**
  - IA-32 Intel Architecture Software Developer's Manual
    - Volume 1: Basic Architecture
    - Volume 2: Instruction Set Reference
    - Volume 3: System Programming Guide
  - Available at Intel's web site
    - http://www.intel.com/design/pentium4/manuals/

EM64T (IA32e)

- **Intel Extended Memory 64 Technology**
  - An enhancement to Intel's IA32 architecture (originally from AMD64)
  - IA32e 64-bit mode
    - Full support for 64-bit integers
    - Larger virtual/physical address space
      - » Currently 2^{48} virtual address space, up to 2^{64}
      - » Currently 2^{40} physical address space, up to 2^{52}
    - 64-bit wide GPRs and instruction pointers
    - Additional registers (8 → 16 GPRs, 8 → 16 XMM registers)
    - Extended and new instructions
    - Removal of older features (e.g., segmentation)
  - IA32e compatibility mode
    - 32-bit applications under 64-bit operating system
    - No recompilation is required.
  - Legacy mode for 16-bit or 32-bit OS
  - http://www.intel.com/technology/64bitextensions/

Linux VM Architecture (1)



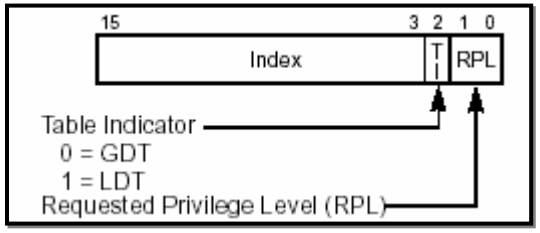
Linux VM Architecture (2)

- **Segmentation: Minimal approach**
  - For better portability across machines

GDT

0x00	NULL
0x08	(not used)
0x10	Kernel text from 0 (4GB)
0x18	Kernel data from 0 (4GB)
0x20	User text from 0 (4GB)
0x28	User data from 0 (4GB)
	(not used)
	(not used)
	Used for APM (4 entries)
	Used for PNPBIOS (8 entries)
0xa0	4 entries per CPU For TSS's & LDT's

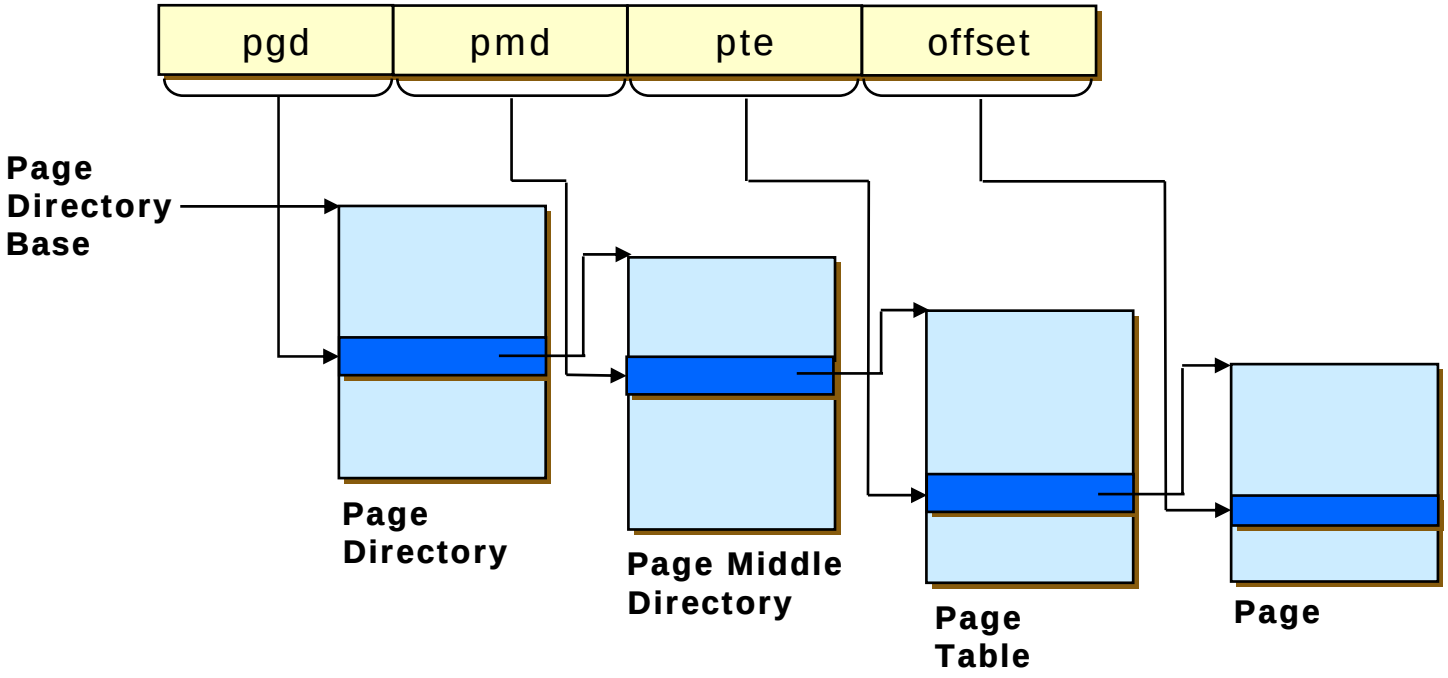
Segment selectors



__KERNEL_CS	0x10
__KERNEL_DS	0x18
__USER_CS	0x23
__USER_DS	0x2b

Linux VM Architecture (3)

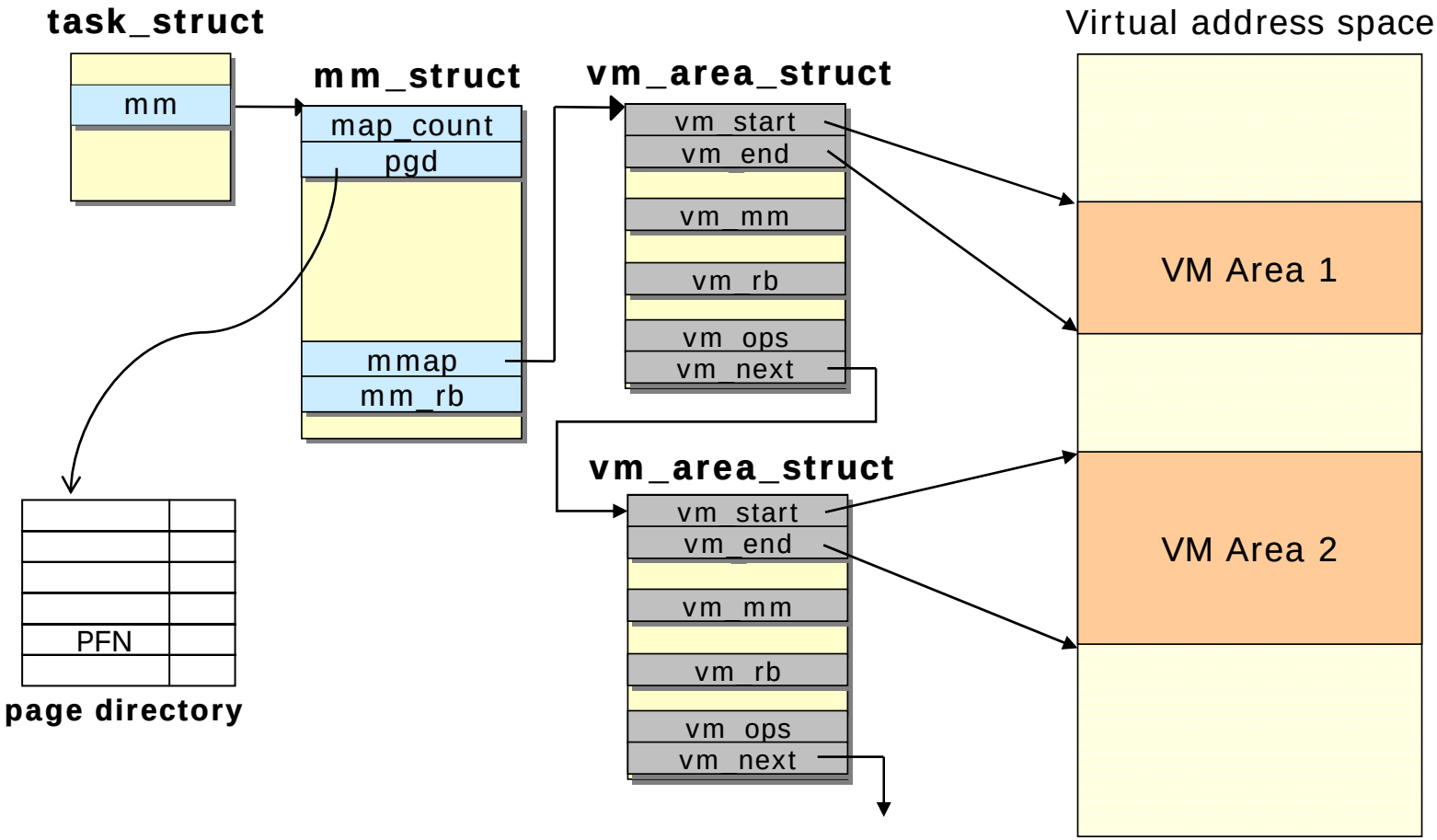
- **Paging: Three-level address translation**
  - In i386, the size of Page Middle Directory (PMD) is 1, if the physical address extension (PAE) flag is disabled.



Linux VM Architecture (4)

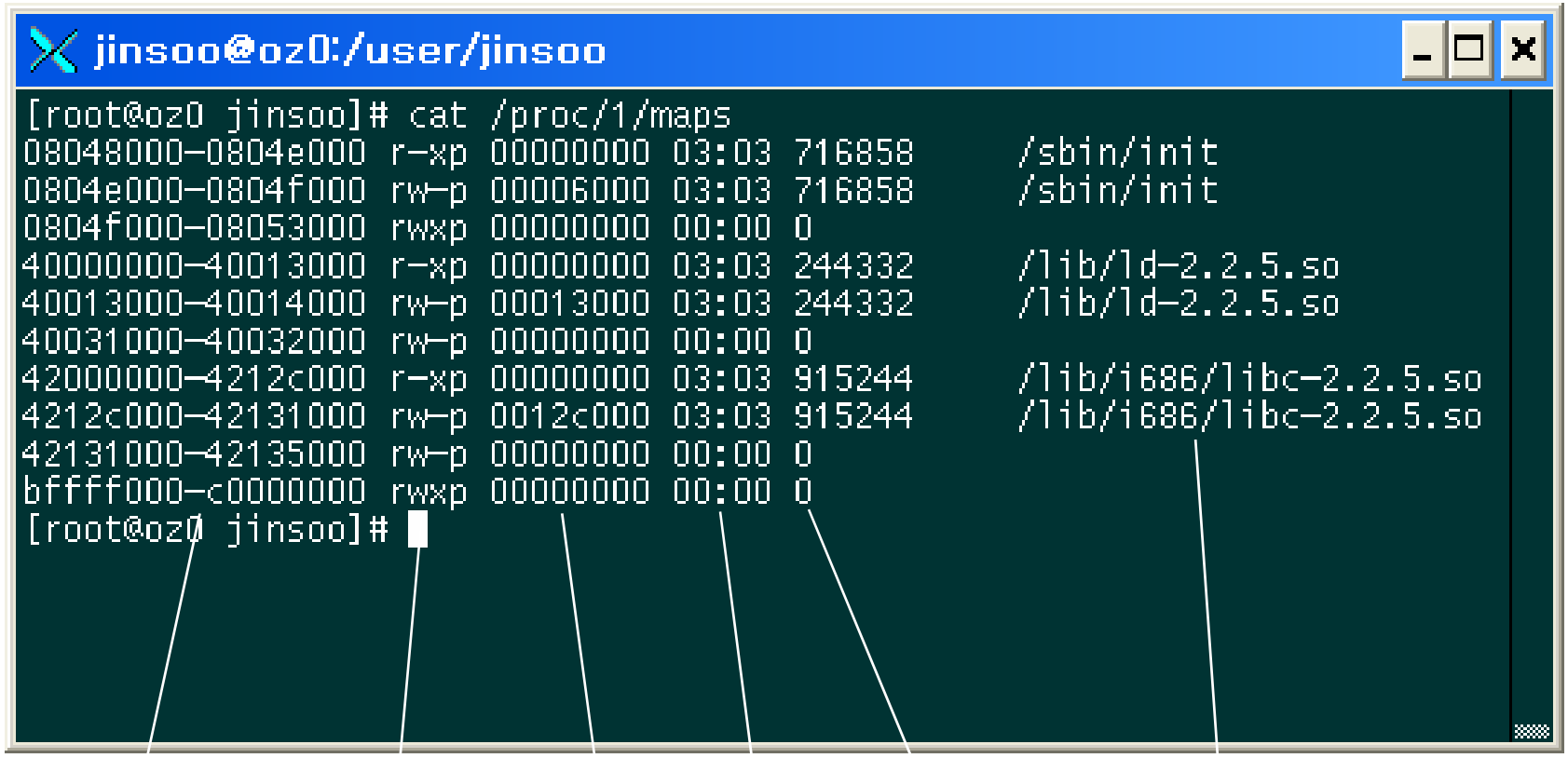
- **Virtual memory areas (VMA)**
  - Nonoverlapping regions, each region representing a continuous, page-aligned subset of the virtual address space.
  - Described by a single vm_area_struct
  - VMAs are linked into a balanced binary tree to allow fast lookup of the region corresponding to any virtual address.
    - VMAs form a red-black tree.

Linux VM Architecture (5)



Linux VM Architecture (6)

- **VMA example**



Linux Memory Management (1)

- **Page descriptors**
  - struct page
    - Flags, reference count, mapped or anonymous, etc.
  - For each physical page frame (32bytes)

- **Memory zones**
  - ZONE_DMA: below 16MB
  - ZONE_NORMAL: 16MB ~ 896MB
  - ZONE_HIGHMEM: above 896MB
  - struct zone

Linux Memory Management (2)

- **Zoned page frame allocator**
  - alloc_pages(), __get_free_pages()
  - Allocate 2^{order} contiguous physical pages
  - From buddy system
- **Slab allocator**
  - General purpose objects
    - 13 geometrically distributed lists of free memory areas
    - 32 bytes ~ 131,072 bytes
    - kmalloc()
  - Specific slab objects
    - kmem_cache_alloc()

Linux Memory Management (3)

- **Noncontiguous memory area management**
  - vmalloc()
  - Allocate a memory area in high memory.
  - Page tables are dynamically changed.
  - Virtually contiguous, but physically noncontiguous

Linux Page Replacement (1)

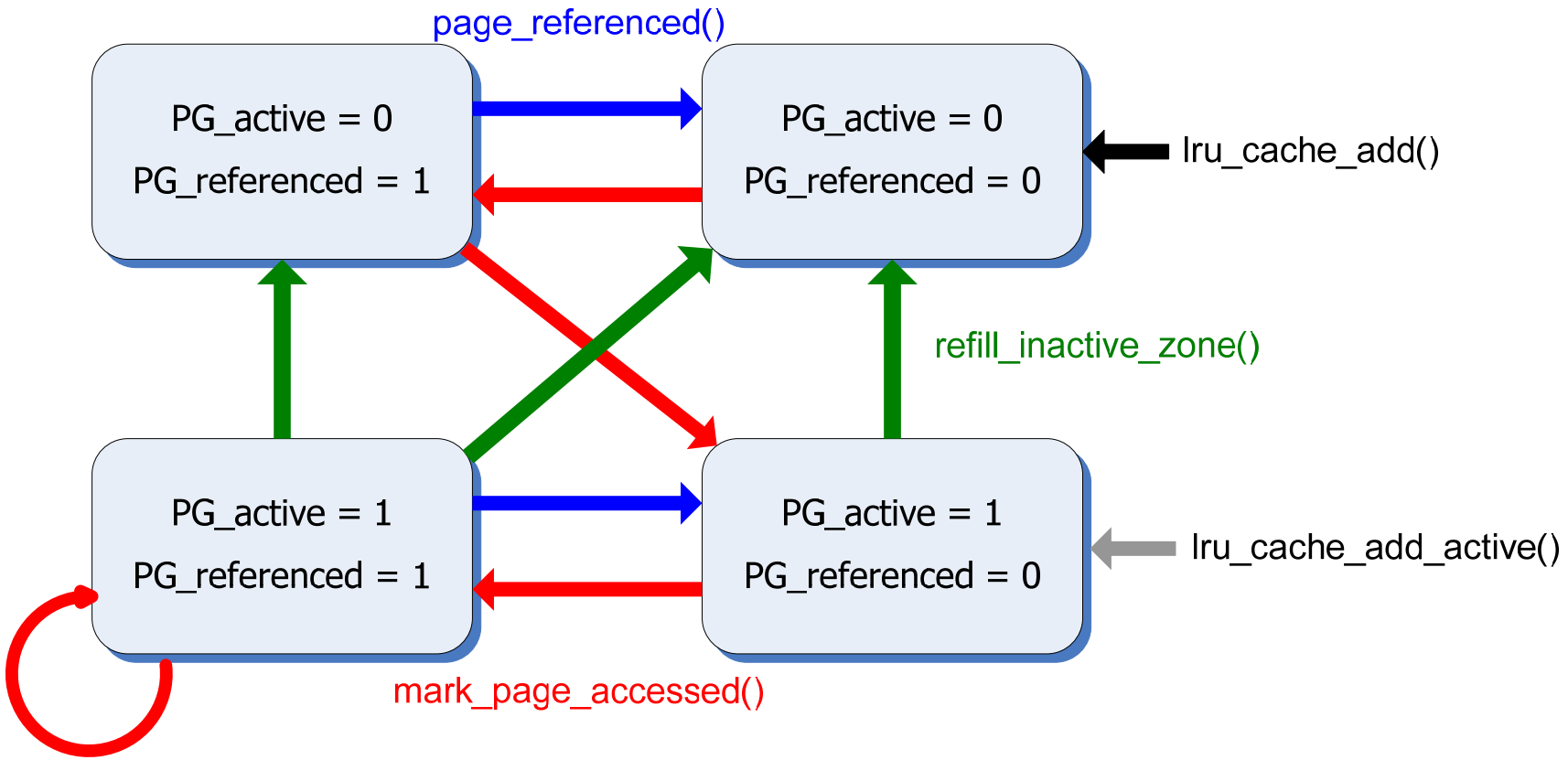
- **Target pages**
  - Swappable: backed by swap areas
    - Anonymous pages in user mode address space
  - Syncable: backed by file systems
    - Mapped pages in user mode address space
    - Pages included in the page cache
    - Block device buffer pages
    - Pages of some disk caches (inode cache, etc.)
  - Unreclaimable pages
    - Reserved pages (PG_reserved)
    - Temporarily locked pages (PG_locked)
    - Memory locked pages (VM_LOCKED)
    - Pages dynamically allocated by the kernel

Linux Page Replacement (2)

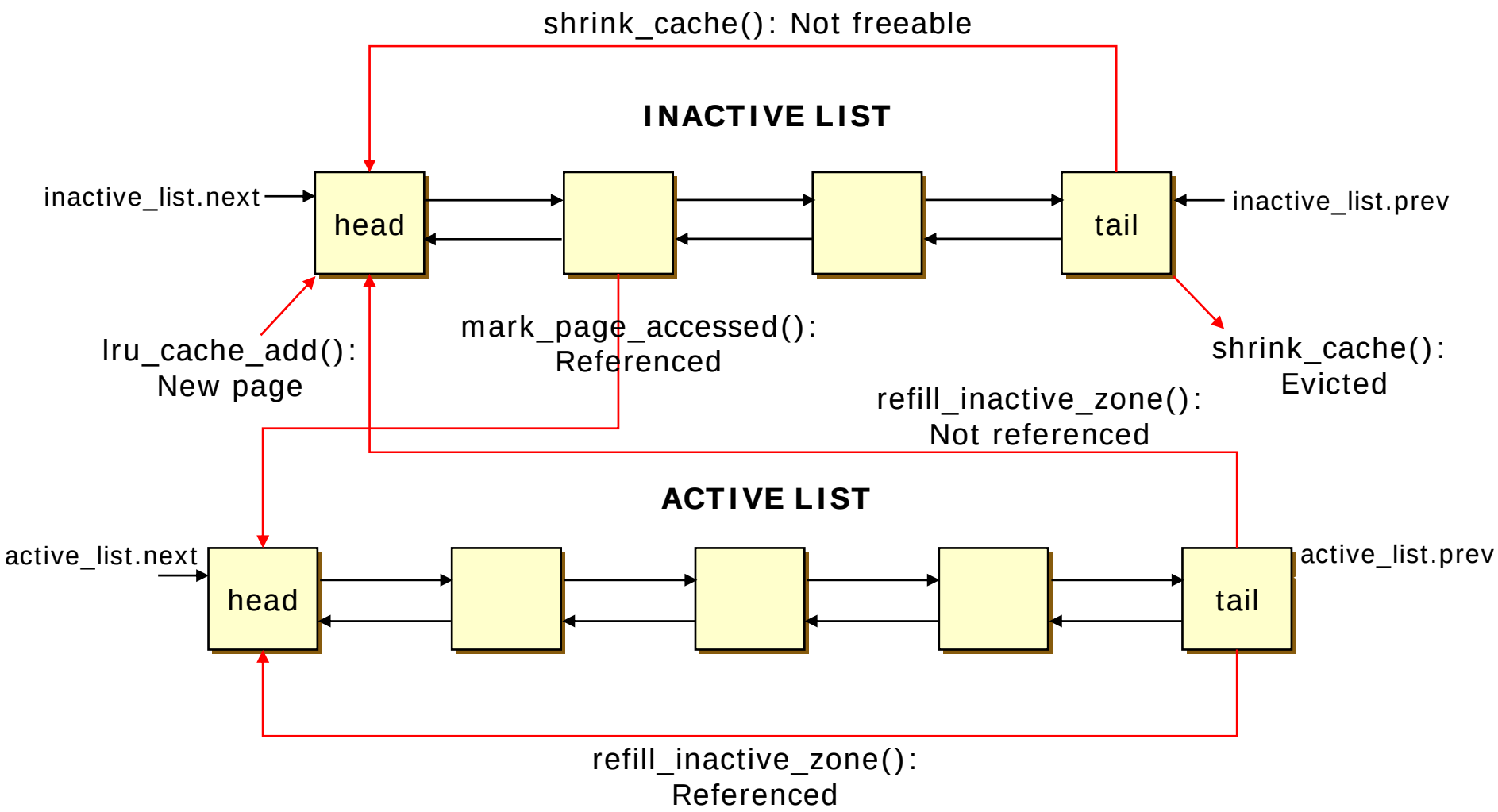
- **Two LRU lists**
  - inactive_list
    - Pages not known to be very important.
    - A dirty page can stay a few passes in the inactive_list while getting written to disk by the file system.
    - New pages are added to the head of the inactive_list.
  - active_list
    - Pages known to be referenced frequently.
    - Defines the working set.
  - When low on memory, we first refill some page from the active_list to the inactive_list head, and then we start freeing pages from the tail of the inactive_list.
  - The detected working set is moved in the active_list.

Linux Page Replacement (3)

- **Moving pages across the LRU lists**



Linux Page Replacement (4)



Discussion

- **IA32/Linux vs. VAX/VMS VM management**
  - What's similar?
  - What's different?