OPERATING SYSTEMS

Memory Management - 13

Nitin V Pujari
Faculty, Computer Science
Dean - IQAC, PES University

OPERATING SYSTEMS

Course Syllabus - Unit 3

Unit-3:Unit 3: Memory Management: Main Memory

Hardware and control structures, OS support, Address translation, Swapping, Memory Allocation (Partitioning, relocation), Fragmentation, Segmentation, Paging, TLBs context switches Virtual Memory - Demand Paging, Copy-on-Write, Page replacement policy - LRU (in comparison with FIFO & Optimal), Thrashing, design alternatives - inverted page tables, bigger pages. Case Study: Linux/Windows Memory

OPERATING SYSTEMS

Course Outline

25	Main Memory: Hardware and control structures, OS support, Address translation	8.1	64.2
26	Dynamic linking, Swapping	8.2	
27	Memory Allocation (Partitioning, relocation), Fragmentation	8.3	
28	Segmentation	8.4	
29	Paging: OS Support, TLBs, Address Translation	8.5	
30	Structure of the Page Table	8.6	
31	Design Alternatives - Inverted Page Tables, Bigger Pages	8.7-8.8	
32	Virtual Memory: Demand Paging, Copy-OnWrite	9.1-9.3	
33	Page replacement policy - LRU	9.4	
34	FIFO & Optimal	9.5	
35	Thrashing	9.6	
36	Case Study: Linux/ Windows Memory Management	9.10	

Topic Outline

- Virtual Memory - Page replacement
- What happens if there is no free Frame ?
- Basic Page Replacement
- Page and Frame Replacement Algorithms
- Graph of Page Faults versus the number of Frames

Topic Outline

- First-In-First-Out (FIFO) Algorithm
- FIFO illustrating Belady's Anomaly
- Optimal Page Replacement Algorithm
- Least Recently Used (LRU) Algorithm
- Use of a Stack to Record Most Recent Page References

Topic Outline

- LRU Algorithm Implementation
- LRU Approximation Algorithm
- Second-Chance (clock) Page-Replacement Algorithm
- Enhanced Second-Chance Algorithm
- Counting Algorithms
- Page-Buffering Algorithms

Topic Outline

- Applications and Page Replacement
- Allocation of Frames
- Fixed Allocation
- Priority Allocation
- Global vs. Local Allocation
- Non-Uniform Memory Access

Topic Outline

- Virtual Memory - Thrashing
- Demand Paging and Thrashing
- Working-Set Model
- Keeping Track of the Working Set
- Page-Fault Frequency

Topic Outline

- Memory Management – Case Study
- Operating System Examples
- Example: The Intel IA 32 Architecture
- Logical to Physical Address Translation in IA32
- Intel IA32 Segmentation

Topic Outline

- Intel IA32 Paging Architecture
- Intel IA32 Page Address Extension
- Intel X86-64
- Windows
- Solaris
- Solaris 2 Page Scanner

Operating System Examples

- H/W Support in Intel Architecture
- Windows
- Solaris

Example: The Intel IA 32 Architecture

- **IA-32** short for "**Intel Architecture, 32-bit**", sometimes also called **i386** is the 32-bit version of the x86 instruction set architecture, designed by Intel and first implemented in the 80386 microprocessor in 1985

- IA-32 is the first incarnation of x86 that supports 32-bit computing

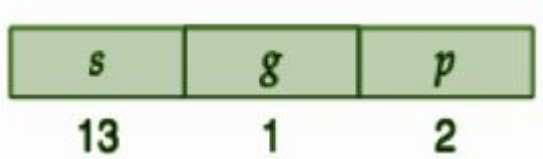
- As a result of the above statement, the "IA-32" term may be used as a metonym to refer to all x86 versions that support 32-bit computing

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)**

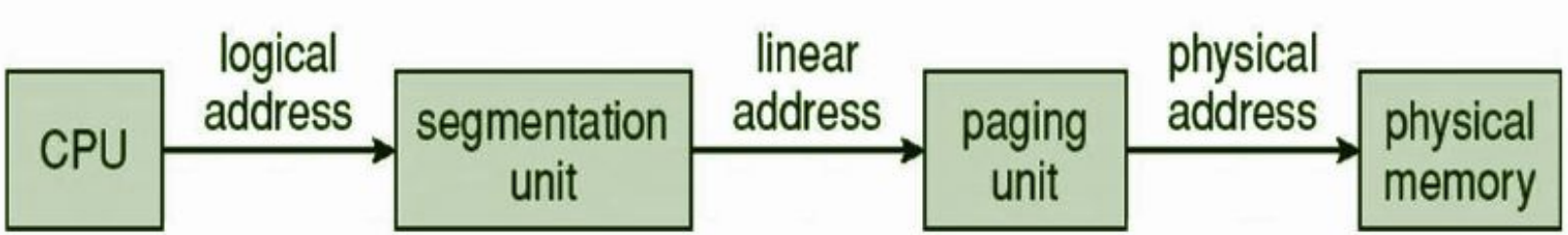
Example: The Intel IA 32 Architecture

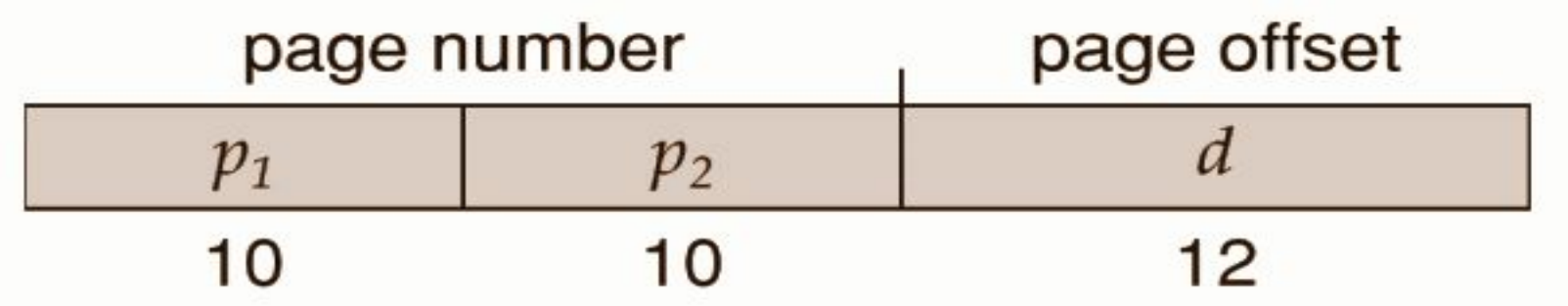
- CPU generates logical address
  - Selector given to segmentation unit which produces **Linear Addresses**



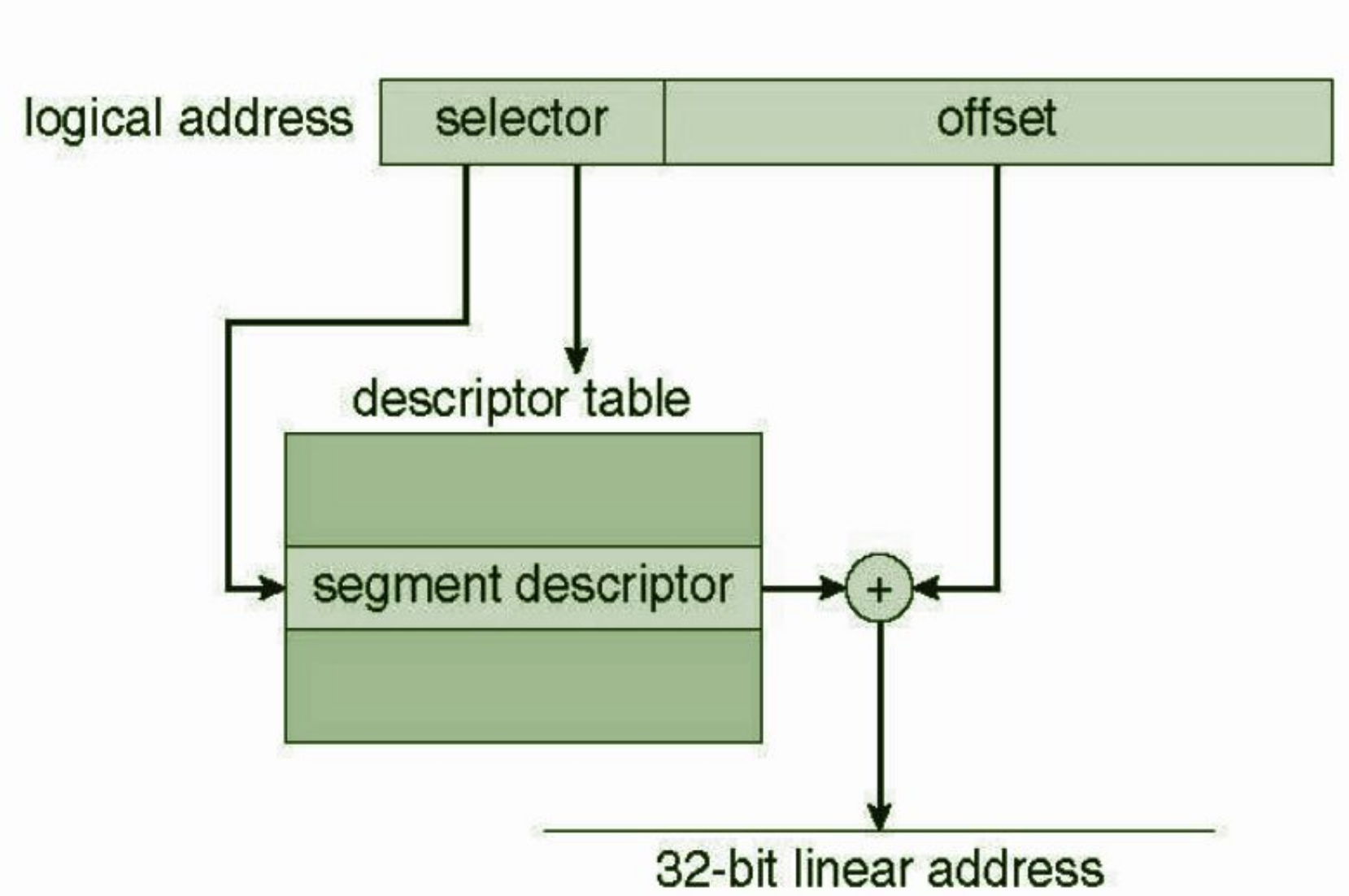
- Linear address is then given to **Paging Unit**
  - Which generates physical address in main memory
  - Paging units form equivalent of MMU
  - Pages sizes can be 4 KB or 4 MB

Logical to Physical Address Translation in IA32

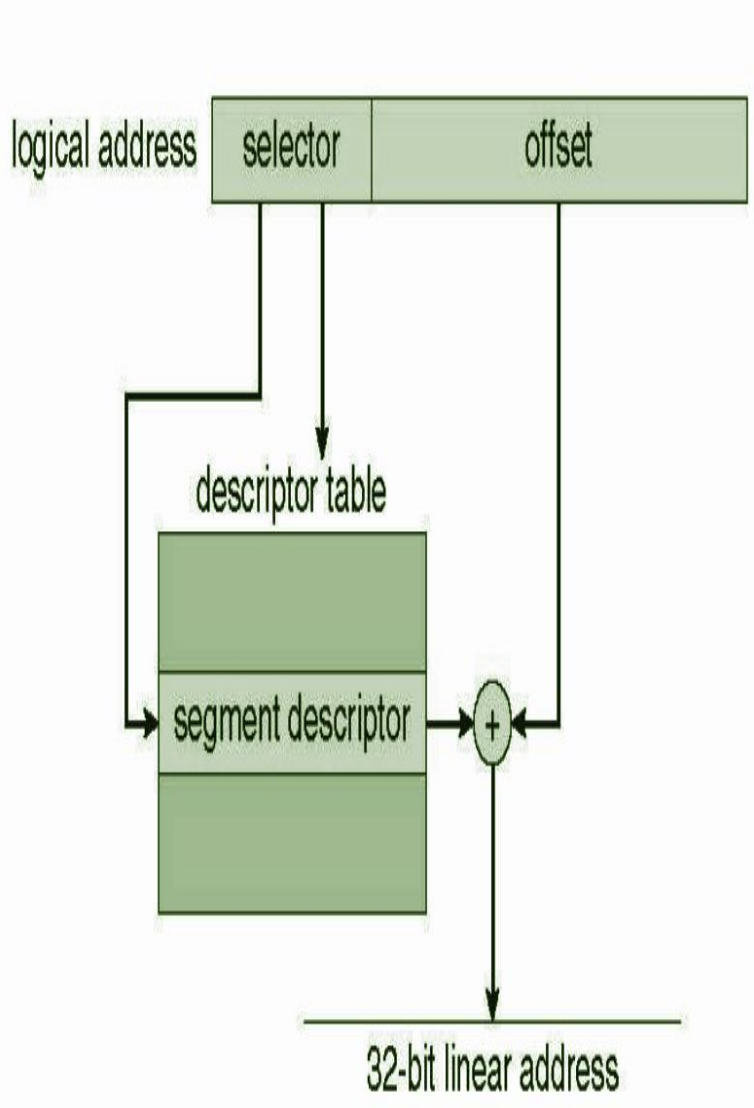




Intel IA32 Segmentation



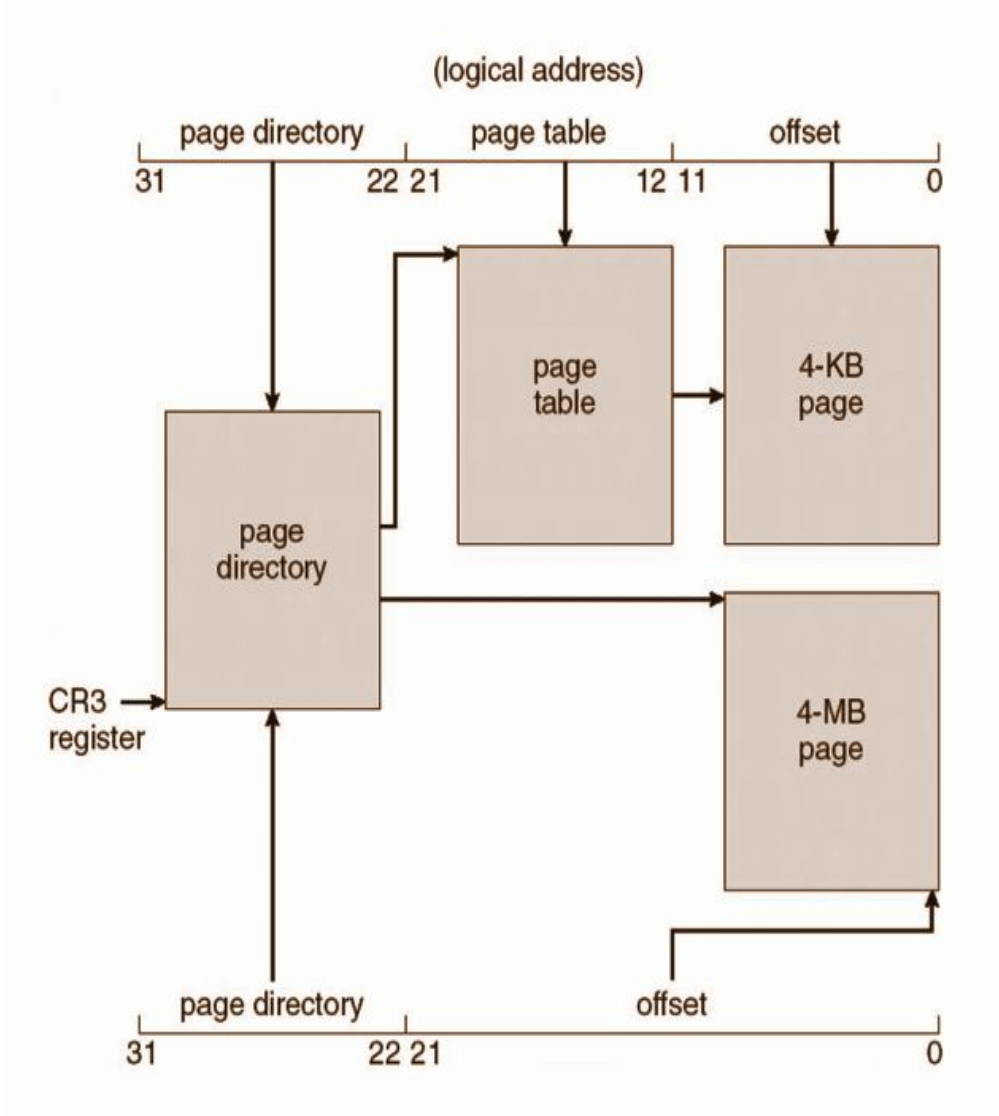
Intel IA32 Segmentation



- The segments are:
  - Regular segments
    - code and data segments
  - System segments
    - (TSS) task state segments
    - (LDT) local descriptor tables

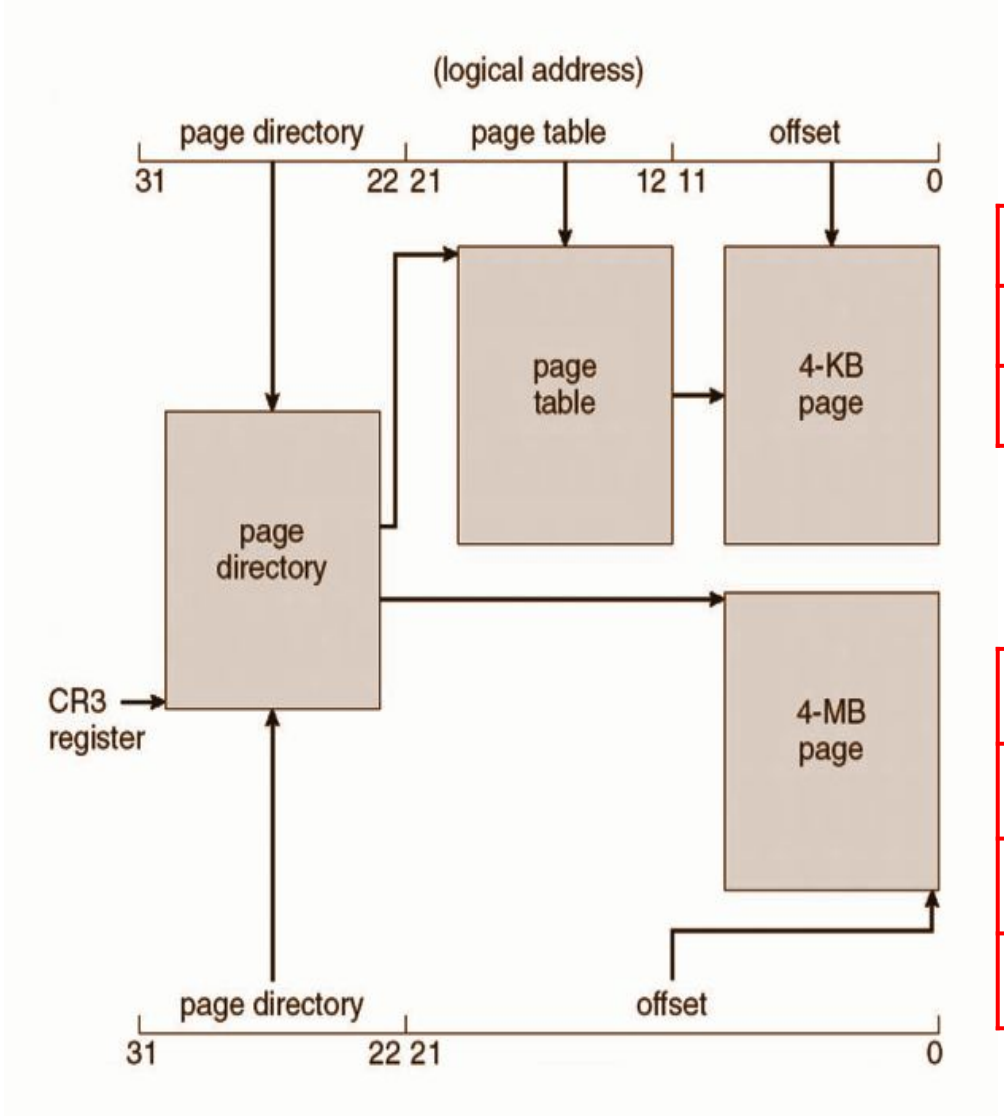
- **Characteristics of system segments**
  - System segments are task specific.
  - There is a Task State Segment (TSS) associated with each task in the system.
  - It contains the tss_struct (sched.h)
  - The size of the segment is that of the tss_struct excluding the i387_union (232 bytes)
  - It contains all the information necessary to restart the task.
  - The LDT's contain regular segment descriptors that are private to a task.

Intel IA32 Paging Architecture



- There are two levels of indirection in address translation by the paging unit.
- A page directory contains pointers to 1024 page tables. Each page table contains pointers to 1024 pages.
- The register CR3 contains the physical base address of the page directory and is stored as part of the TSS in the task_struct and is therefore loaded on each task switch.
- **TSS => Task State Segment**

Intel IA32 Paging Architecture



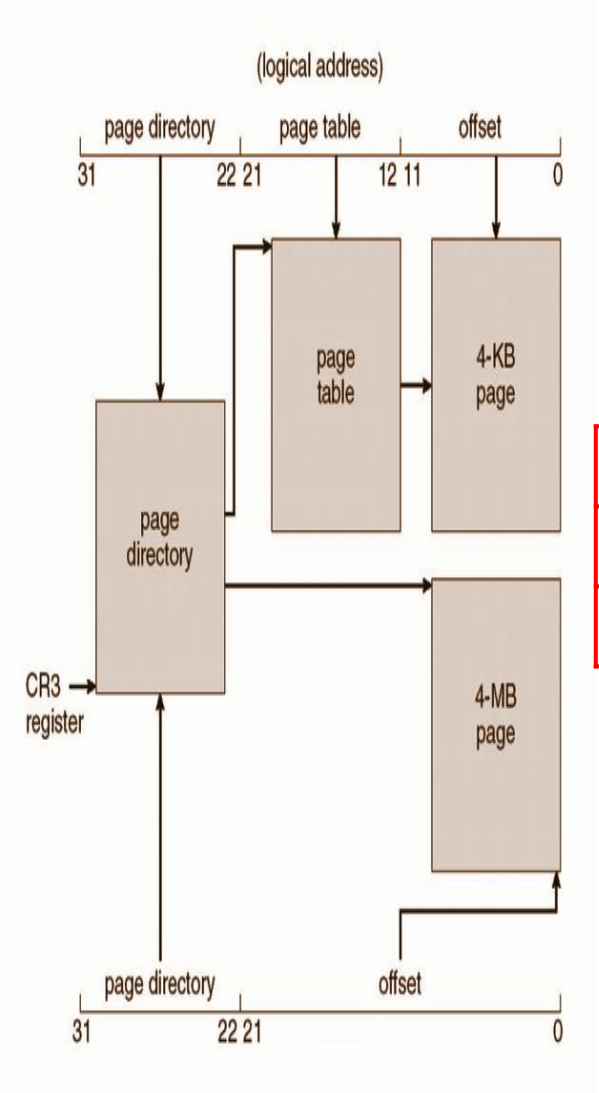
- A 32-bit Linear address is divided as follows

Directory	Table	Offset
Bit 31 to Bit 22	Bit 21 to Bit 12	Bit 11 to Bit 0
10 Bits	10 Bits	12 Bits

- Physical address is then computed (in hardware) as

Directory	Table
CR3 + Base	Points to the Base Table
Table Base + Table	Points to the Page Base
Physical Address =>	Page Base + Offset

Intel IA32 Paging Architecture



- Page directories (page tables) are page aligned so the lower 12 bits are used to store useful information about the page table (page) pointed to by the entry

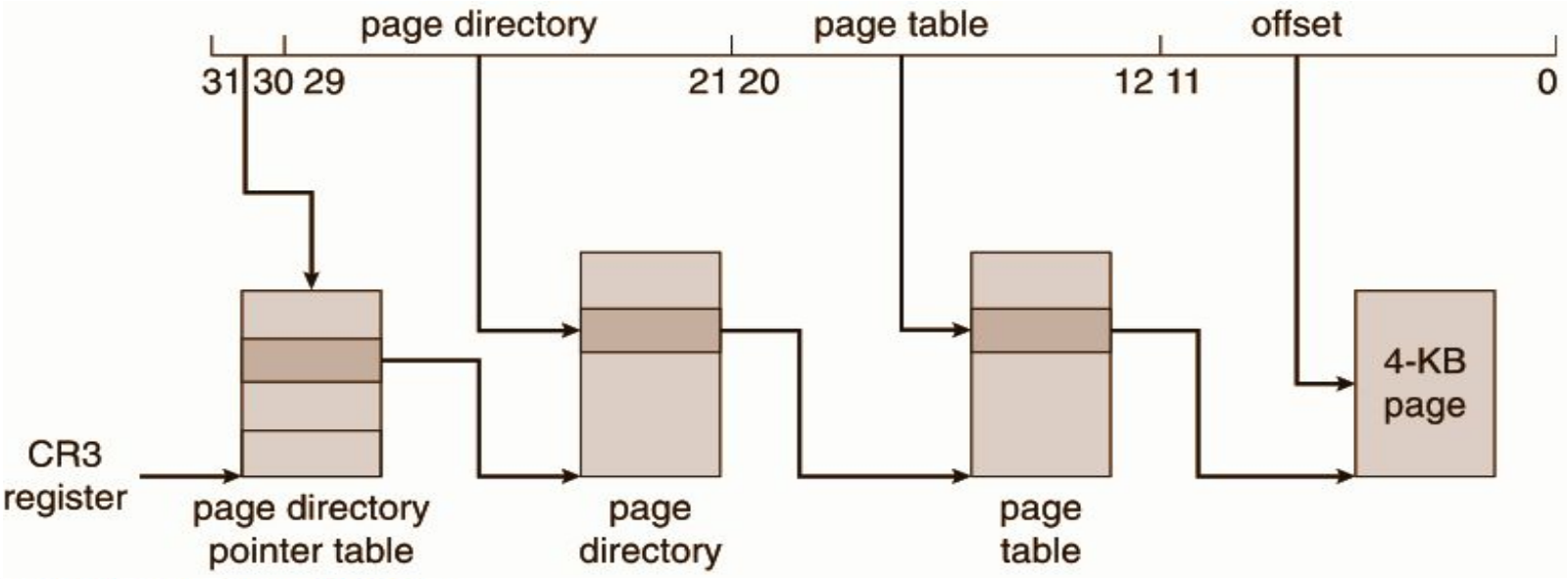
- Format for Page directory and Page table entries

Bit Positions										
31..12	11..9	8	7	6	5	4	3	2	1	0
Address	OS	0	0	D	A	0	0	U/S	R/W	P

- D => 1 means page is dirty (undefined for page directory entry)
- R/W => 0 means readonly for user.
- U/S => 1 means user page
- P => 1 means page is present in memory
- A => 1 means page has been accessed (set to 0 by aging)
- OS=> bits can be used for LRU etc, and are defined by the OS

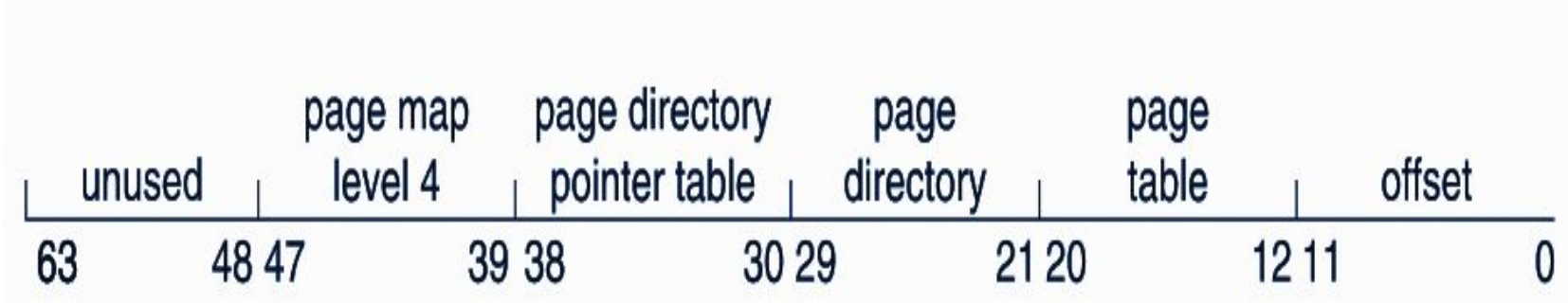
Intel IA32 Page Address Extension

- 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



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



Windows

- 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

Windows

- 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

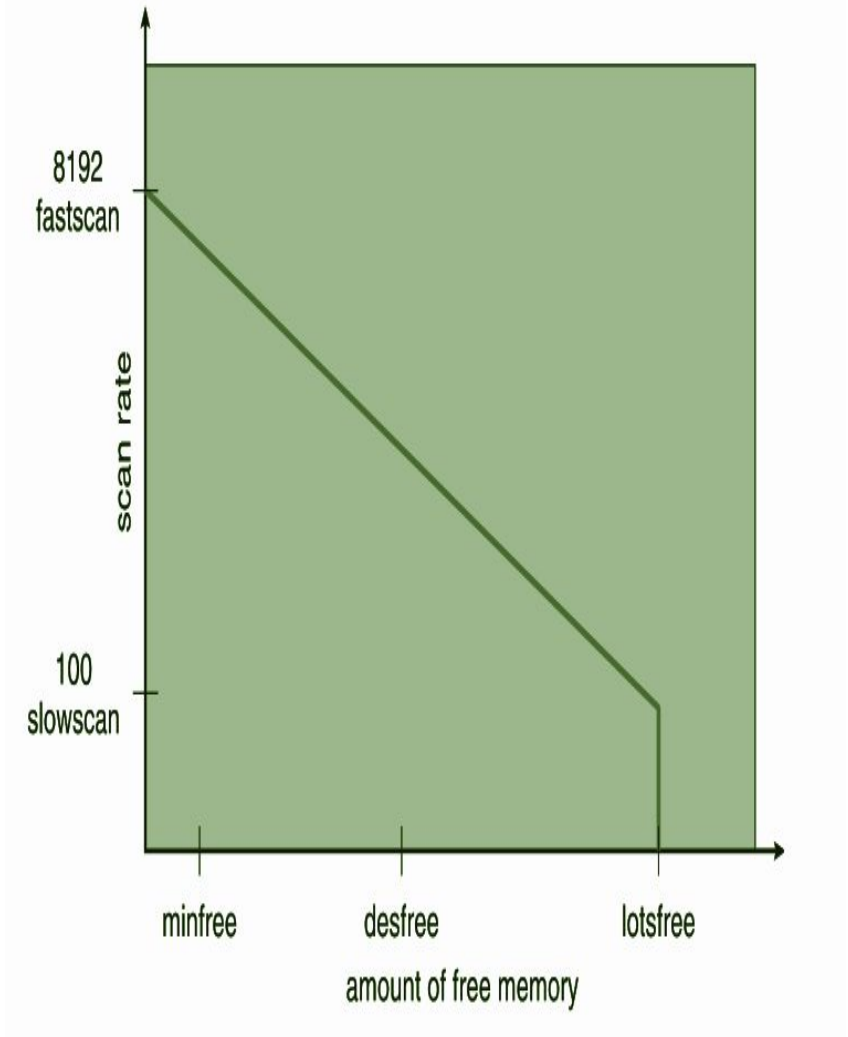
Solaris

- **Scanrate** is the rate at which pages are scanned. This ranges from **slowscan** to **fastscan**

- **Pageout** is called more frequently depending upon the amount of free memory available

- Priority paging gives priority to process code pages

Solaris 2 Page Scanner



- Maintains a list of free pages to assign faulting processes
- **lotsfree** => threshold parameter (amount of free memory) to begin paging
- **desfree** => threshold parameter to increasing paging
- **minfree** => Specifies the minimum acceptable memory level. When memory drops below this number, the system biases allocations toward allocations necessary to successfully complete pageout operations or to swap processes completely out of memory. Either allocation denies or blocks other allocation requests.
- Paging is performed by **pageout process**

Topic Outline

- Memory Management – Case Study
- Operating System Examples
- Example: The Intel IA 32 Architecture
- Logical to Physical Address Translation in IA32
- Intel IA32 Segmentation

Topic Outline

- Intel IA32 Paging Architecture
- Intel IA32 Page Address Extension
- Intel X86-64
- Windows
- Solaris
- Solaris 2 Page Scanner

THANK YOU

Nitin V Pujari
Faculty, Computer Science
Dean - IQAC, PES University

nitin.pujari@pes.edu

For Course Deliverables by the Anchor Faculty click on www.pesuacademy.com