Virtual Lecture
On
Memory Management in Linux

By

Prof. Nisha V.Kimmatkar
JSPM's
RSCOE Tathawade,Pune-3
Computer Engineering Department.

Two Parts

- Architecture Independent Memory should be flexible and portable enough across platforms

- Implementation for a specific architecture

Architecture Independent Memory Model

- Process virtual address space divided into pages
- Page size given in PAGE_SIZE macro in asm/page.h

  (4K for x86 and 8K for Alpha)
- The pages are divided between 4 segments
- User Code, User Data, Kernel Code, Kernel Data

- In User mode, access only User Code and User Data
- But in Kernel mode, access also needed for User Data

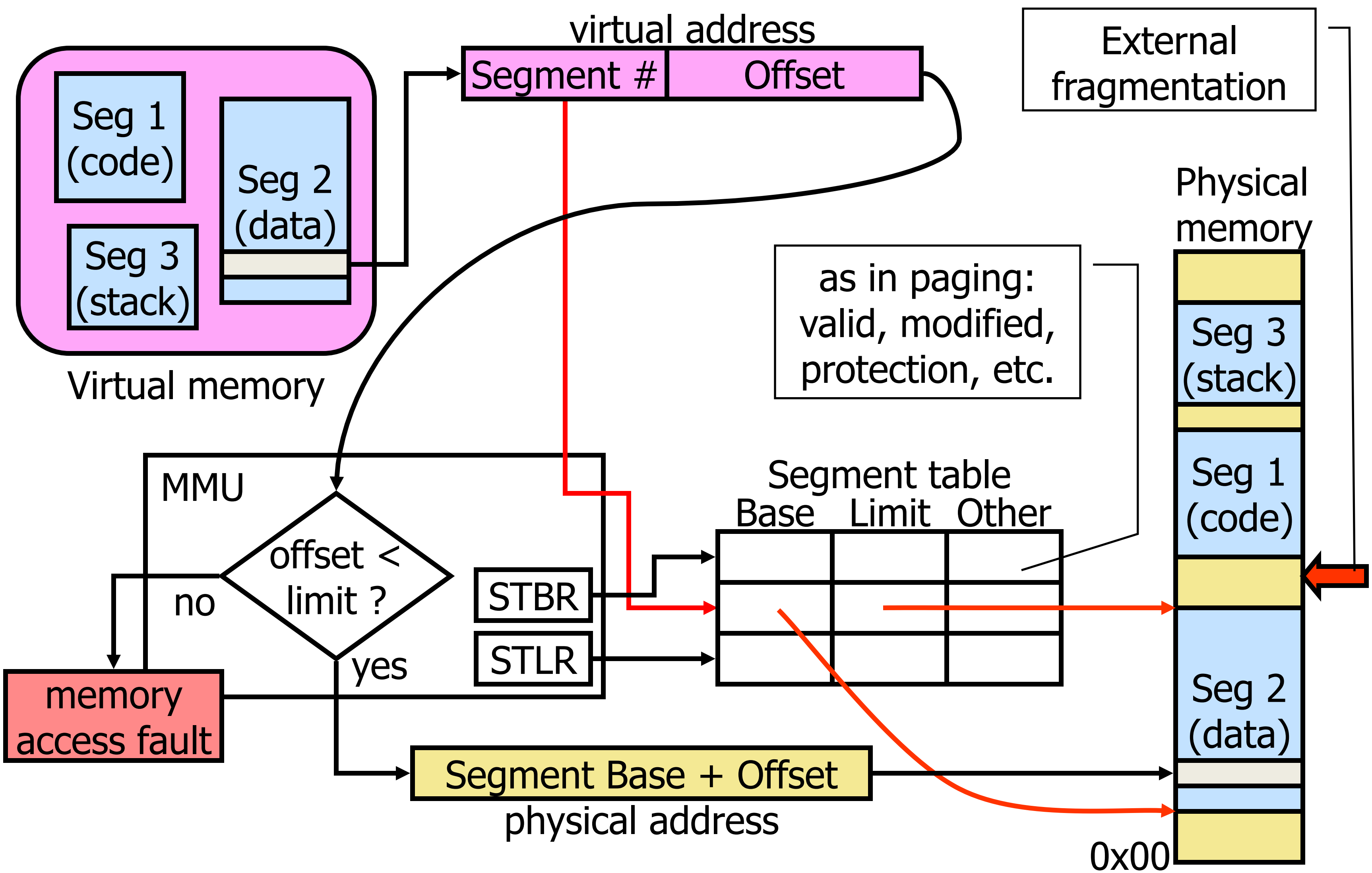
Linux memory management

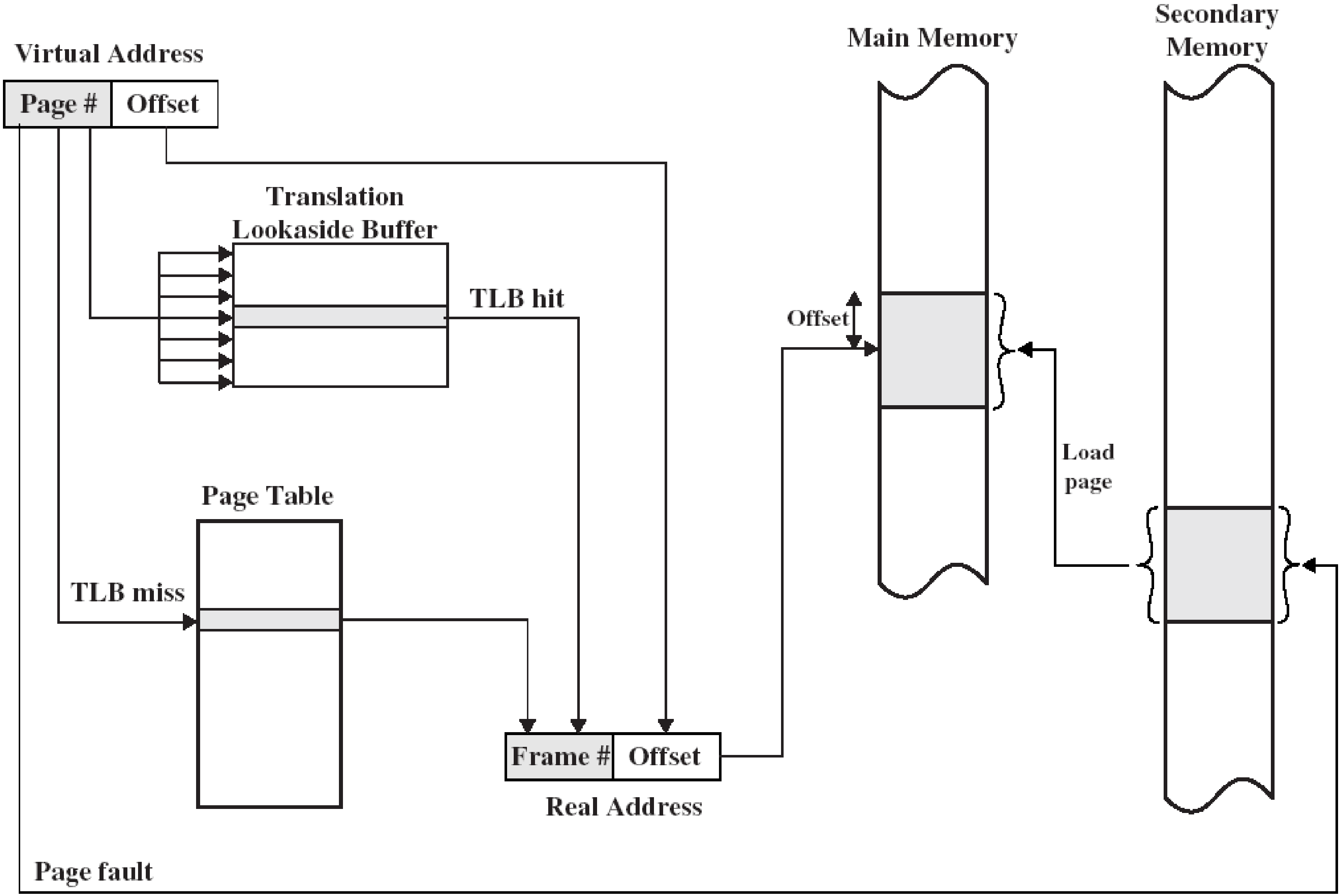
This part of the Linux kernel is relatively complex and is only presented in overview, the point is to familiarize yourself with the names and terminology

- Paging
- Physical and logical memory layout
- Contiguous frame management
- Non-contiguous frame management
- Process address space
- Memory descriptors
- Memory regions
- Page faults

- Intel x86 processes have segments.
- Linux tries to avoid using segmentation.
- Memory management is simpler when all processes use the same segment register values.
- Using segment registers is not portable to other processors

What is Segmentation





Use of a Transltion Lookaside Buffer

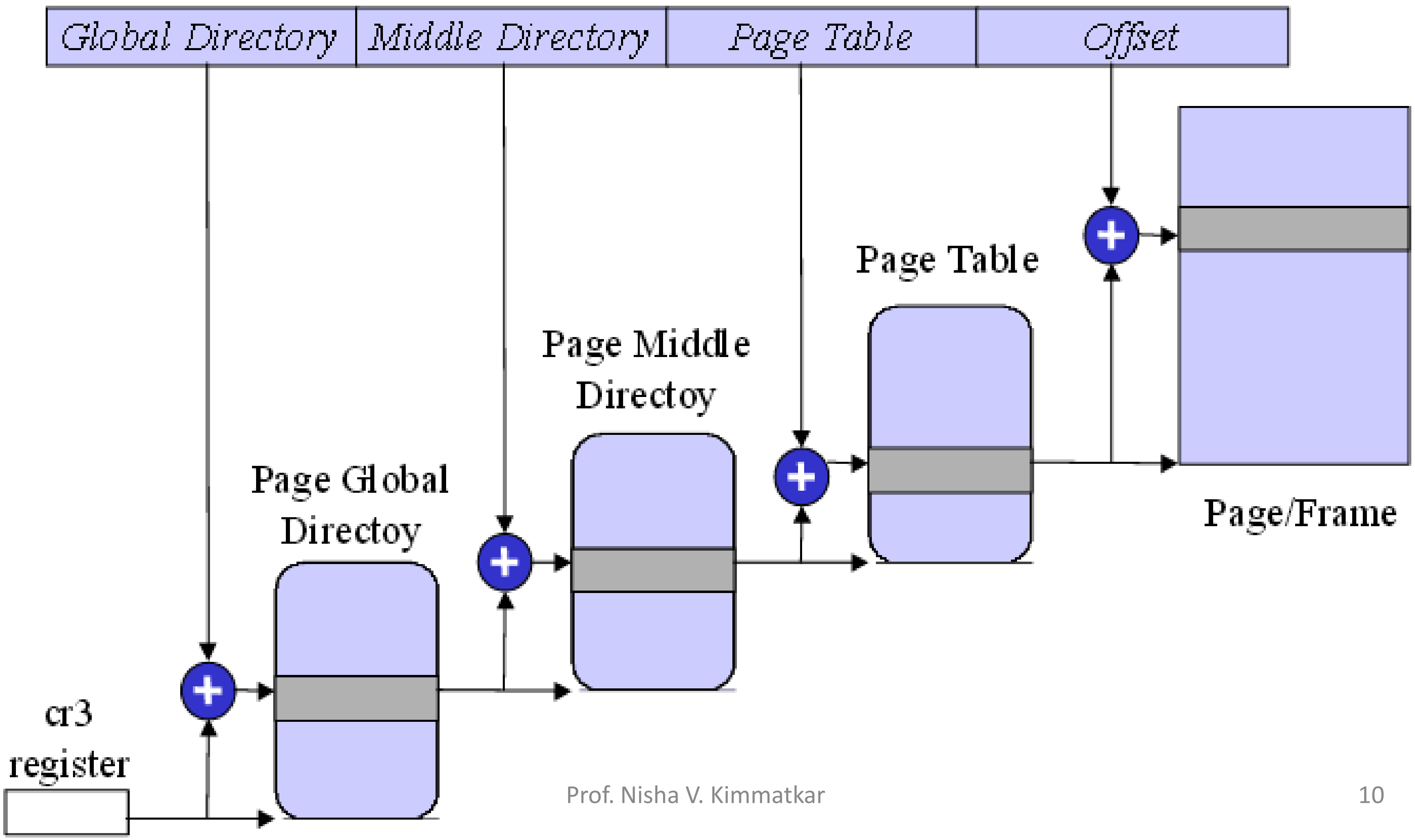
Linux uses paging

- 4k page size
- A three-level page table to handle 64-bit addresses
- On x86 processors
  1. only a two-level page table is actually used
  2. Paging is supported in hardware
  3. TLB is provided as well

Linux Memory Management

View of a logical address in Linux

(For x86 processors, *Middle Directory* is 0 bits)



Linux Kernel Memory Management

- Approximately the first two megabytes of physical memory are reserved

I. For the PC architecture and for OS text and data

II. The rest is available for paging

Page Fault

Segmentation

Paging

Contiguous Memory Allocation

Buddy Algorithm for large memory request

Slab Allocator for Small Memory request

Non Contigous Memory Allocation

reserved addresses

- The logical address space of a process is divided into two parts

A. 0x00000000 to PAGE_OFFSET-1 can be addressed in either user or kernel mode
B. PAGE_OFFSET to 0xffffffff can be addressed only in kernel mode
C. PAGE_OFFSET is usually 0xc0000000

Linux Page Frame Management

- The kernel keeps track of the current status of each page frame in an array of

Struct page descriptors,

one for each page frame

- Page frame descriptor array is called **mem_map**
- Keeps track of the usage count (== 0 is free, > 0 is used)
- Flags for dirty, locked, referenced, etc

- **The kernel allocates and release frame via**
  get_free_pages(gfp_mask, order)
  free_pages(addr, order)

Linux Page Frame Management

- In theory, paging eliminates the need for contiguous memory allocation, but…
  - Some operations like DMA ignores paging circuitry and accesses the address bus directly while transferring data
    - As an aside, some DMA can only write into certain addresses
  - Contiguous page frame allocation leaves kernel paging tables unchanged, preserving TLB and reducing effective access time
- As a result, Linux implements a mechanism for allocating contiguous page frames
  - *So how does it deal with external fragmentation?*

Contiguous Page Frame Allocation

- **Buddy system algorithm**

Contiguous Memory Area Allocation

- The buddy algorithm is fine for dealing with relatively large memory requests, but it how does the kernel satisfy its needs for small memory areas?
- In other words, the kernel must deal with internal fragmentation

- Slab allocator for dealing with small memory area allocation

- The slab allocator is a system of allocating memory that is optimized for the allocation and freeing of same-sized memory objects.
- The slab allocator organizes the memory into caches, slabs and objects.
- A cache consists of multiple slabs, and each slab is a contiguous region of memory containing objects of all the same size.

- When a cache is created, the creator specifies the object size and a name for the cache, as well as some flags.
- When an object is allocated, if there are no free objects in any slabs, a new slab is allocated. For small objects, a slab is a single page of memory.

Slab allocator

- Groups objects into caches
- A set of specific caches is created for kernel operation

Non-contiguous Memory Area Allocation

- Linux tries to avoid allocating noncontiguous memory areas, but for infrequent memory requests sometimes it makes sense to allocate noncontiguous memory areas
- Linux uses most of the reserved addresses above PAGE_OFFSET to map noncontiguous memory areas

Process Address Spaces

- The address space of a process consists of all logical addresses that the process is allowed to use.
- Each process address space is separate (unless shared)
- The kernel allocates logical addresses to a process in intervals called memory regions
- Memory regions have an initial logical address and a length, which is a multiple of 4096

Process Memory Descriptor

- All information related to the process address space is included in the memory descriptor (mm_struct) referenced by the mm field of the process descriptor.

- Some examples of included information
  - A pointer to the top level of the page table, the Page Global Directory, in field `pgd`
  - Number of page frames allocated to the process in field `rss`
  - Process' address space size in pages in field `total_vm`
  - Number of locked pages in field `locked_vm`
  - Number of processes sharing the same `mm_struct`, i.e., lightweight processes
- *Memory descriptors are allocated from the slab allocator cache using* `mm_alloc()`

Process Memory Region

- Linux represents a memory region (i.e., an interval of logical address space) with vm_area_struct

Page Fault Handler

- When a process requests more memory from the kernel, it only gets additional logical address space, not physical memory
- When a process tries to access its new logical address space, a page fault occurs to tell the kernel that the memory is actually needed (i.e., demand paging)

Steps in handling a page fault