OPERATING SYSTEM

Unit III — Address Spaces & Virtual Memory

Topics Covered in this Unit

Evolution of Memory Management: Early Uniprogramming, Static Relocation & The Virtual Address Space
Goals of Virtual Memory: Transparency, Efficiency & Protection / Process Isolation
Memory API & Pitfalls: Stack vs. Heap, malloc(), free(), Memory Leaks, Buffer Overflows & brk()/sbrk()
Hardware Address Translation: Base and Bounds Registers, Dynamic Relocation & Segmentation Mechanics
Free-Space Management: Splitting, Coalescing, Free Lists & Allocation Policies (First Fit, Best Fit, Buddy System)
Paging Architecture: Virtual Pages to Physical Frames (PFN), Linear Page Tables & Page Table Entries (PTEs)
Translation Lookaside Buffers (TLBs): Hardware Cache, TLB Hits/Misses, Context Switches (ASID) & Replacement
Multi-Level Page Tables & Hybrid Systems: Paging with Segments & Reducing Page Table Memory Overhead
Beyond Physical Memory (Swapping): Swap Space, Present Bit, Page Fault Exception & Complete Control Flow
Page Replacement Policies: FIFO, Belady's Anomaly, Optimal (MIN), LRU, Clock Algorithm & Thrashing
High-Yield University Exam Question Bank with Detailed Model Answers & Unit Summary

Compiled in a structured, easy-to-understand, textbook style format
Specially curated for B.Tech / B.E. / BCA / MCA / B.Sc Computer Science & Information Technology Students

COURSE SYLLABUS & MAPPING

[UNIT III SYLLABUS — PRESCRIBED UNIVERSITY CURRICULUM]

This section contains the official syllabus for Unit III. Verify with your university curriculum mapping.

Unit III Syllabus Breakdown:

Unit III Address Spaces: Early systems, multiprogramming and time sharing, the address space, virtualization of memory, memory api: types of memory, the malloc() call, the free() call, segmentation, fine-grained vs. coarsegrained segmentation, free-space management, paging, a memory trace, faster translations (TLBs), TLB basic algorithm, TLB issue: context switches, replacement policy, hybrid approach: paging and segments, beyond physical memory: mechanisms, swap space, the page fault, page fault control flow

• Course Code: _____ • Semester / Year: _____

TABLE OF CONTENTS

3.1 The Address Space & Virtual Memory Abstraction

Definition: Virtual Address Space

The Address Space is the operating system's abstraction of physical memory provided to a running process. It contains all the memory state of the running program (Code, Heap, Stack). Virtual Memory provides three vital guarantees:

1. Transparency: The program operates as if it possesses a private, dedicated, contiguous physical memory.

2. Efficiency: Virtualization is enforced with near-zero performance overhead using hardware MMU and TLB caching.

3. Protection / Isolation: Processes are strictly isolated from one another; a errant process cannot read or overwrite another process's or the kernel's memory.

Memory API & Common Memory Bugs

Memory Bug Category	Underlying Cause	Consequences & Detection Tools
Memory Leak	Allocating heap memory via `malloc()` but forgetting to call `free()` before losing pointer reference.	Slowly consumes available RAM, eventually crashing system via OOM (Out Of Memory) killer. Detected via Valgrind.
Use-After-Free (Dangling Pointer)	Reading or writing to a heap pointer after memory has been returned via `free()`.	Corrupts heap allocator metadata; causes undefined behavior and severe security vulnerabilities.
Double Free	Calling `free()` twice on the exact same allocated pointer.	Corrupts free list data structures, leading to catastrophic crash or heap exploitation.
Buffer Overflow	Writing beyond the allocated bounds of an array/buffer (e.g., `strcpy` without bounds check).	Overwrites adjacent stack frames, return addresses, or heap chunk headers (exploited in arbitrary code execution).

3.2 Segmentation & Free-Space Management

Definition: Segmentation

Segmentation divides the virtual address space into variable-sized logical units (Segments: Code, Heap, Stack). Each segment has a dedicated Base register (physical starting address) and Bounds / Limit

register (size of segment) in hardware MMU, eliminating the huge unused gap between heap and stack from physical RAM.

Coarse-Grained vs. Fine-Grained Segmentation

- • **Coarse-Grained Segmentation:** Divides memory into a small number of large monolithic segments (e.g., Code, Heap, Stack). Supported in early x86 architectures.
- • **Fine-Grained Segmentation:** Supports thousands of small variable segments (e.g., individual arrays or subroutines) requiring a Segment Table. Suffers heavily from External Fragmentation.

Free-Space Allocation Strategies

Allocation Strategy	Algorithmic Search Mechanism	Tradeoffs & Fragmentation Behavior
First Fit	Scans the free list from the beginning and allocates the very first free chunk large enough to hold request.	Extremely fast; tends to clutter the beginning of free list with tiny fragments.
Best Fit	Searches entire free list to find the chunk whose size is closest to requested size (minimizing leftover).	Minimizes wasted leftover, but incurs full list traversal and creates tiny unusable slivers.
Worst Fit	Searches entire list to find the largest available chunk, allocating request and keeping large leftover.	Leaves large leftovers, but performs full traversal and performs poorly in practice.
Buddy Allocator (Binary Buddy)	Divides memory pools by powers of 2. When freeing, adjacent buddy blocks are recursively coalesced.	Extremely fast allocation and coalescing in O(1)/O(log N); causes Internal Fragmentation.

3.3 Paging: Hardware Translation & Page Tables

Definition: Paging

Paging divides the virtual address space into fixed-sized units called Pages (typically 4 KB). Physical memory is divided into an array of identical fixed-sized slots called Physical Page Frames (PFNs). Paging completely eliminates External Fragmentation because any virtual page can be placed into any free physical frame.

VIRTUAL TO PHYSICAL ADDRESS TRANSLATION SCHEME

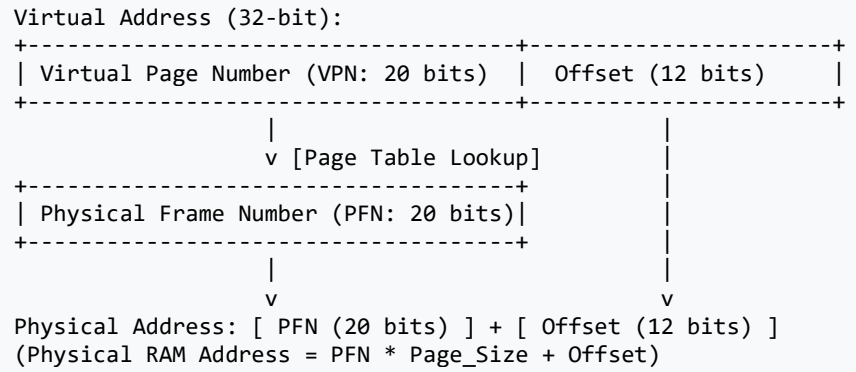

Figure: Hardware address translation mechanics with 4 KB page size (2^12 = 4096 bytes)

Page Table Entry (PTE) Control Bits

- • **Valid Bit:** Indicates whether the virtual page is legally mapped in the process address space (1 = valid; 0 = causes Segmentation Fault).
- • **Protection Bits (R/W/X):** Controls read, write, and execute permissions.
- • **Present Bit (P):** Indicates whether the page is physically in RAM (P=1) or has been swapped out to disk (P=0 → triggers Page Fault).
- • **Dirty Bit (D):** Set to 1 by hardware whenever a write occurs (indicates page must be written to disk before being evicted).
- • **Reference / Accessed Bit (A):** Set to 1 when page is read or written (used by LRU/Clock page replacement algorithms).

3.4 Faster Translations: Translation Lookaside Buffers (TLBs)

Definition: Translation Lookaside Buffer (TLB)

The TLB is a high-speed hardware associative cache built directly into the CPU's Memory Management Unit (MMU). It caches popular virtual-to-physical address translations (VPN → PFN), allowing memory accesses to execute in a single CPU cycle without reading the slow main memory page table on every access.

The Basic TLB Algorithm Flow

- **Step 1:** Extract VPN from virtual address and check if VPN exists in TLB.
- **Step 2 (TLB Hit):** Extract PFN directly from TLB, combine with offset, and access physical RAM (Zero extra memory overhead).

- **Step 3 (TLB Miss):** Hardware MMU (or software OS trap handler) walks the Page Table in main memory, retrieves PTE, inserts [VPN → PFN] into TLB, and retries the instruction.

TLB Issue: Handling Context Switches

- • **The Problem:** VPN 10 in Process A translates to physical frame 100, but VPN 10 in Process B translates to physical frame 500. Without handling, Process B could access Process A's private memory!
- • **Solution 1 (Flushing TLB):** On every context switch, set all TLB valid bits to 0. Simple, but incurs high TLB miss penalty after every context switch as new process warms up cache.
- • **Solution 2 (Address Space Identifier - ASID):** Hardware adds an 8-bit or 16-bit ASID tag to every TLB entry. The MMU only matches a TLB entry if (VPN matches AND ASID matches current process ID), allowing multiple processes to share the TLB simultaneously.

3.5 Beyond Physical Memory: Swapping & The Page Fault

Definition: The Page Fault Mechanism

When a process accesses a virtual page whose Present Bit in the PTE is 0, the MMU hardware generates a Page Fault Trap to the kernel. The OS handles the fault by allocating a free physical frame, reading the missing page from Swap Space on disk via I/O, updating the PTE (PFN, Present=1), and resuming the faulting user instruction.

Page Replacement Policies

Replacement Algorithm	Selection Criteria	Properties & Real-World Feasibility
Optimal (Belady's MIN)	Evicts the page that will NOT be accessed for the longest time in the future.	Theoretical benchmark providing lowest possible miss rate; impossible to implement in practice (requires future clairvoyance).
FIFO (First-In, First-Out)	Evicts the oldest page brought into memory.	Simple; suffers from Belady's Anomaly (increasing physical frames can paradoxically increase page fault count).
Least Recently Used (LRU)	Evicts the page that has not been accessed for the longest time in the past.	Approximates Optimal by relying on temporal locality; perfect LRU hardware stack is too expensive to maintain per memory access.

Replacement Algorithm	Selection Criteria	Properties & Real-World Feasibility
Clock Algorithm (Second-Chance LRU)	Maintains circular list of frames with a rotating hand. If Reference Bit = 1, clears to 0 and advances; if 0, evicts page.	Efficient, realistic hardware-software approximation of LRU used in modern OS kernels.

- • **Thrashing:** When the collective working sets of all running processes exceed total physical RAM, the system spends virtually 100% of its time swapping pages in and out from disk rather than executing instructions. Solved by reducing degree of multiprogramming.

3.6 High-Yield University Exam Question Bank & Model Answers

Part A: Short Answer Questions (2–3 Marks)

- **Q1. Differentiate between Internal and External Fragmentation.**

Answer: Internal Fragmentation occurs when allocated memory blocks are larger than requested (e.g., in fixed-size paging, process needs 1KB but gets a 4KB page, wasting 3KB inside). External Fragmentation occurs when free memory is broken into tiny scattered slivers between allocated segments, making it impossible to satisfy a large contiguous request even though total free memory is sufficient.

- **Q2. What is a Translation Lookaside Buffer (TLB)? What is its purpose?**

Answer: A TLB is a high-speed associative hardware cache in the MMU that stores recent virtual-to-physical address mappings (VPN → PFN). It enables single-cycle memory translations on a TLB Hit, bypassing slow multi-level page table traversals in physical RAM.

- **Q3. Explain Belady's Anomaly. Which page replacement algorithm exhibits it?**

Answer: Belady's Anomaly is the counter-intuitive phenomenon where increasing the number of physical page frames results in an INCREASE in the total number of page faults. It occurs in the FIFO page replacement algorithm because FIFO does not belong to the class of stack algorithms.

- **Q4. How does an Address Space Identifier (ASID) optimize context switches?**

Answer: An ASID tags each TLB translation entry with a process ID. During context switches, the OS does not need to flush the entire TLB; the hardware MMU simply checks that the ASID matches the currently running process, preserving cached translations.

Part B: Long Answer Questions (8–10 Marks)

- **Q5. Explain the Paging hardware translation mechanism in detail. Describe the structure of a Page Table Entry (PTE) and calculate physical address from a virtual address.**

Model Answer Outline:

- 1. Paging concept: Virtual pages and physical page frames (PFNs) eliminating external fragmentation.
- 2. Address division: 32-bit virtual address divided into Virtual Page Number (VPN) and Offset (e.g., 20-bit VPN + 12-bit Offset for 4KB pages).
- 3. Page Table Entry (PTE) bit fields: PFN, Valid bit, Protection (R/W/X) bits, Present bit, Dirty bit, and Reference bit.
- 4. Numerical example: Given Virtual Address 0x00003045, Page size 4KB (Offset = 0x045, VPN = 3). If Page Table maps VPN 3 → PFN 7, Physical Address = (7 × 4096) + 0x045 = 0x00007045.
- 5. Complete hardware translation block diagram showing MMU traversal.

- **Q6. Describe the Page Fault handling mechanism and Virtual Memory Swapping. Explain the Clock (Second-Chance) Page Replacement algorithm with a diagram.**

Model Answer Outline:

- 1. Concept of Virtual Memory Swapping: Swap space on disk and the Present bit in PTE.
- 2. Step-by-step Page Fault Control Flow: (1) MMU detects Present=0 → (2) Hardware Page Fault Exception trap → (3) OS allocates physical frame (running page replacement if full) → (4) Issue disk I/O to read page → (5) Update PTE PFN and Present=1 → (6) Return from trap and retry instruction.
- 3. Clock Algorithm: Circular buffer of frames, reference bit evaluation, clearing bit on pass, and selecting victim frame.
- 4. Definition of Thrashing, Working Set Model, and mitigation techniques.

3.7 Unit III Summary & Quick Revision Sheet

Memory Management Domain	Key Theoretical & Hardware Takeaways for Exams
Address Spaces	Transparency, Efficiency, Isolation. Heap (malloc/free), Stack (local vars), Segmentation (Base/Bounds).
Free-Space Policies	First Fit (Fast), Best Fit (Small leftover), Buddy Allocator (Power of 2 blocks + fast coalescing).
Paging Mechanics	Fixed 4KB pages. VA = [VPN | Offset]. PA = [PFN | Offset]. PTE (Valid, Present, Dirty, Ref, R/W).
TLB Accelerators	MMU cache for translations. Hits = 1 cycle. Miss = Page walk. Context switch: Flush vs ASID tag.
Swapping & Page Faults	Present=0 triggers Page Fault trap → OS reads swap disk → updates PTE. Clock Algorithm (Second-chance LRU approximation).