Operating Systems
Course Code: 7120300200 4
Demand Segmentation & Overlay.

by -
Asst. Prof. Minal Rajwar

What is Segmentation?

- Segmentation is a memory management technique where the **logical address space is divided into segments**.
- Each segment has a **name (or number)** and a **length**.
- A logical address includes two parts:
  - the **segment number** and
  - the **offset** within that segment.

What is Demand Segmentation?

- It loads **segments only when they are used**, not all at once.
- Reduces memory usage by **avoiding loading unused segments**.
- Similar to **demand paging**, but with **variable-sized segments** instead of fixed-size pages.

Pros

- ★ **Efficient Memory Usage**: Only active segments occupy RAM.
- ★ **Faster Startup**: Program can begin running without all segments loaded.
- ★ **Reduced I/O**: Avoids loading unnecessary segments from disk.

Cons

- ★ **Segment Faults**: Like page faults, if a required segment is not in memory, it must be loaded, which slows down execution.
- ★ **Complex Implementation**: Needs support for segment tracking, fault handling, and loading mechanisms.
- ★ **Still Prone to External Fragmentation**: Because segments are variable in size

Overlay in Memory Mange

- **Overlay memory management** is a technique used to run programs **larger than available physical memory**.
- It works by loading **only necessary parts (overlays)** of a program into memory, while keeping the rest on disk.
- When different parts are needed, the system **swaps overlays in and out** of memory as required.
- Used when:
  - The program is **too large** for memory.
  - You want to **optimize memory usage** by not loading the entire program at once.

How it Works:

A program is split into **smaller parts called overlays**.

- Only the **active overlay** is loaded in memory.
- When another part of the program is needed, the current overlay is **swapped out** and the new one is loaded.

Pros

- ★ **Better Memory Utilization**: Multiple programs or parts of a program can fit in limited memory.
- ★ **Reduced Fragmentation**: Easier to manage smaller chunks.
- ★ **Faster Memory Access**: Only required parts are loaded, reducing access overhead.
- ★ **Lower Memory Footprint**: Saves space by loading smaller sections.
- ★ **Improved System Performance**: Less memory swapping improves efficiency.

Cons

- ★ **Increased Complexity**: More complex design and software/hardware support required.
- ★ **Overhead**: Time and resources are needed to manage overlays.
- ★ **Code/Data Fragmentation**: Program performance may drop if overlays are not managed properly.
- ★ **Limited Address Space**: There's a limit to how many overlays can be handled at once.

Types of Overlay Memory Management

- **Overlay memory management** is a technique used to run programs **larger than available physical memory**.
- It works by loading **only necessary parts (overlays)** of a program into memory, while keeping the rest on disk.
- When different parts are needed, the system **swaps overlays in and out** of memory as required.
- Used when:
  - The program is **too large** for memory.
  - You want to **optimize memory usage** by not loading the entire program at once.

Comparison : Segmentation, Paging & Overlay.

Feature/ Aspect	Segmentation	Paging	Overlays
Basic Idea	Divides memory into **variable-sized segments**	Divides memory into **fixed-sized pages**	Manually divides programs to fit into limited memory
Memory Division	Logical divisions based on program structure (code, stack, heap)	Physical/logical memory split into equal-sized pages	Programmer/user defines which parts are loaded
Address Format	Segment number + offset	Page number + offset	Logical addresses with manual mapping
Fragmentation Type	External fragmentation	Internal fragmentation	Can avoid fragmentation but complex to manage

Feature/ Aspect	Segmentation	Paging	Overlays
Mapping Mechanism	Segment Table	Page Table	No hardware support; manual mapping
Ease of Use	Easier than overlays, supports modular programming	Transparent to the programmer	Complex and manual
Protection and Sharing	Better support for protection/sharing of segments	Difficult (page-level)	Limited
Hardware Support	Requires segment table and segment registers	Requires page table and TLB	No specific hardware support
Use Case	Used in early Intel architectures (e.g., x86)	Widely used in modern systems	Used in early systems with very limited memory

Discussion & Revision

- Which memory management technique divides memory into fixed-size blocks?
- Which method causes external fragmentation?
- Which method requires manual loading of program parts into memory?
- What is used to map pages in paging?
- Which technique divides programs based on logical parts like code, stack and data?