# CSE 4733/6733 - Operating System 1

Stephen A. Torri, Ph.D.

Mississippi State University





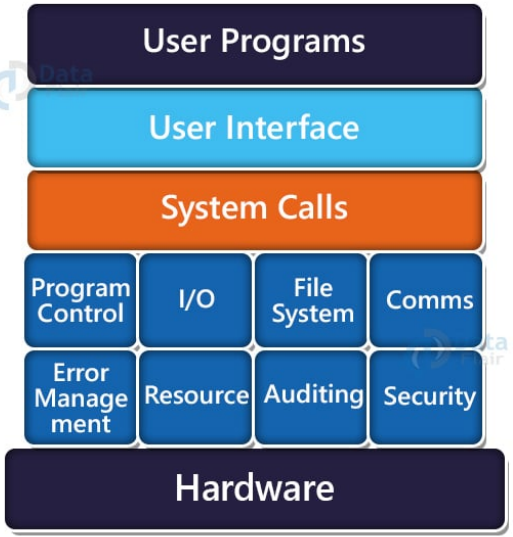
CSE 4733/6733 - Operating System 1

Stephen A. Torri, Ph.D.

Mississippi State University

# Introduction

# Introduction








Introduction


1. We'll look at how segmentation divides a process's memory into logical units
2. Work through some examples of address translation using segment registers
3. Discuss the advantages and drawbacks of using segmentation. This background will be valuable as we compare segmentation with other memory management techniques such as paging

# Definition





Definition

# Segmentation

## Definition [1]

- Segmentation is a memory management technique that divides the address space into variable-sized segments.
- Each segment represents a logical unit (such as code, data, or stack) and functions as an independent virtual address space.
- In modern 64-bit systems, segmentation is mostly disabled—system memory is treated as one contiguous block, with segment bases set to zero.





Segmentation
Definition [1]

- Segmentation is a memory management technique that divides the address space into variable-sized segments.
- Each segment represents a logical unit (such as code, data, or stack) and functions as an independent virtual address space.
- In modern 64-bit systems, segmentation is mostly disabled—system memory is treated as one contiguous block, with segment bases set to zero.

1. Introduced by Intel in 1978, segmentation originally used a 16-bit segment register to point to memory segments of up to 64 KB. This method allowed for flexible memory organization in early systems. (Intel Architecture Manual, Volume 1, 2.2.1)
2. In 64-bit mode, segmentation is generally disabled. The processor sets the base of segments such as CS, DS, ES, and SS to zero, effectively creating a flat memory model where the linear address equals the effective address. Segmented and real mode addressing is not available in 64-bit systems [2].

# Segmentation

## Properties

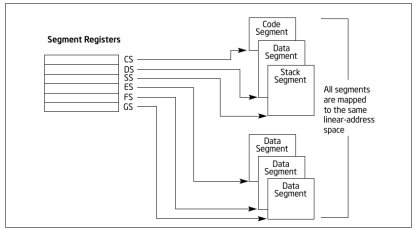



Figure 3-7. Use of Segment Registers in Segmented Memory Model

- A segment is a contiguous block of a process's virtual address space, defined by a base address and a limit.[2]
- Segmentation allows the operating system to map each segment independently into physical memory, even if they are noncontiguous.[2]
- Each segment functions as its own virtual address space, which facilitates modular design and protection.





Segmentation
Properties
A segment is a contiguous block of a process's virtual address space, defined by a base address and a limit.[2]
Segmentation allows the operating system to map each segment independently into physical memory, even if they are noncontiguous.[2]
Each segment functions as its own virtual address space, which facilitates modular design and protection.

1. Segmentation ensures that only the necessary portions of the virtual address space are mapped into physical memory. This minimizes wasted space and provides the flexibility to relocate segments to different areas of physical memory.

# Segmentation

## Properties

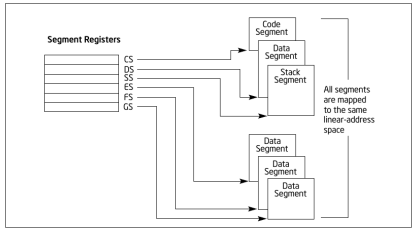
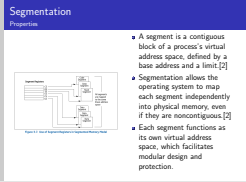



Figure 3-7. Use of Segment Registers in Segmented Memory Model

- A segment is a contiguous block of a process's virtual address space, defined by a base address and a limit.[2]
- Segmentation allows the operating system to map each segment independently into physical memory, even if they are noncontiguous.[2]
- Each segment functions as its own virtual address space, which facilitates modular design and protection.








1. **Core Message:**
   Segmentation divides a process's virtual address space into variable-sized segments, each defined by a base and a limit, allowing independent mapping and protection.

   **Supporting Evidence:**
   - Segmentation was introduced in the late 1970s (e.g., by Intel) to overcome the limitations of small address spaces in early processors.
   - It minimizes internal fragmentation by allocating only the necessary amount of memory for each logical unit (code, data, stack).
   - This approach also simplifies memory protection by isolating different components of a process.

# Segmentation

## Properties

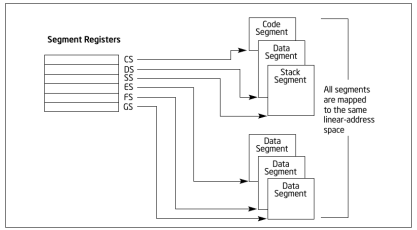
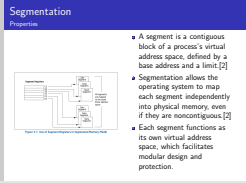



Figure 3-7. Use of Segment Registers in Segmented Memory Model

- A segment is a contiguous block of a process's virtual address space, defined by a base address and a limit.[2]
- Segmentation allows the operating system to map each segment independently into physical memory, even if they are noncontiguous.[2]
- Each segment functions as its own virtual address space, which facilitates modular design and protection.








1. **Broader Context:**
   - Modern 64-bit systems typically use a flat memory model with paging; however, the principles of segmentation (flexible allocation, isolation, and protection) remain fundamental to understanding OS memory management.
   - Recognizing how segmentation works provides insight into the evolution of memory management techniques used by operating systems to map virtual memory to physical memory.

# Segmentation

## Segmented Memory Model Image

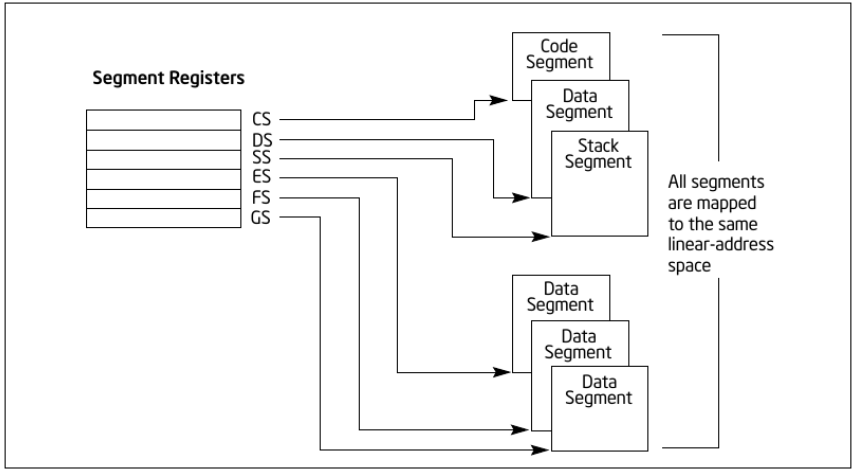
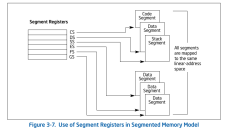



Figure 3-7. Use of Segment Registers in Segmented Memory Model





Segmentation
Segmented Memory Model Image



# Segmentation

## Flat Memory Model Image

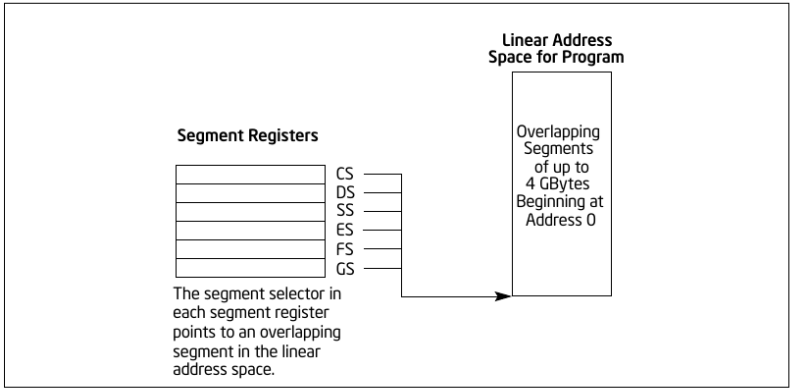
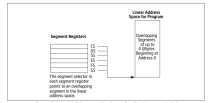



Figure 3-6. Use of Segment Registers for Flat Memory Model





Segmentation
Flat Memory Model Image



1. The flat memory model treats the entire memory as one contiguous block of addresses. In modern 64-bit mode, segmentation is largely disabled—this means that the base addresses for segments such as CS, DS, ES, and SS are set to zero. As a result, the effective address is equal to the linear address, which simplifies the address translation process [2].

# Segmentation

## Flat Memory Model Image

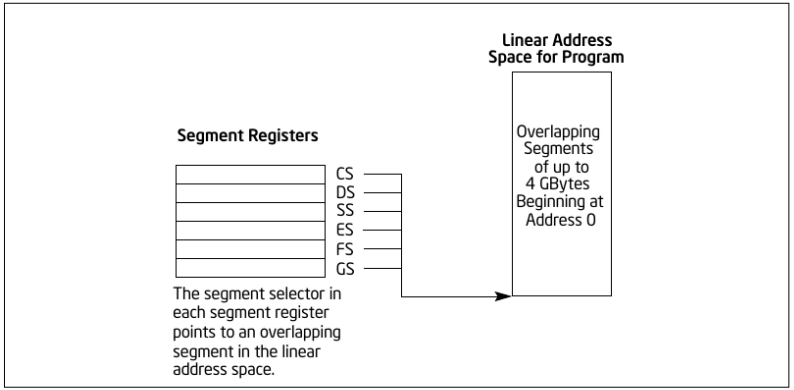
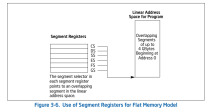



Figure 3-6. Use of Segment Registers for Flat Memory Model





Segmentation
Flat Memory Model Image



1. This design was adopted to reduce both hardware and software complexity. With a flat memory model:
   - The processor avoids the extra step of adding a segment base to an offset, leading to faster address computation.
   - Programming becomes more straightforward as developers no longer need to manage multiple segment registers or deal with noncontiguous memory areas.
   - It pairs effectively with paging, the memory management scheme used in modern operating systems, allowing for efficient mapping, protection, and isolation of memory pages.

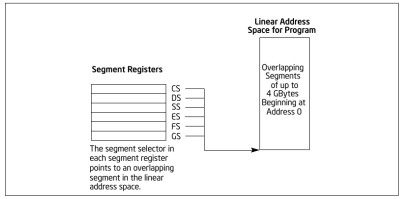



Figure 3-6. Use of Segment Registers for Flat Memory Model

## Segmentation Fault

A segmentation fault occurs when a program attempts to access memory that it is not permitted to access or accesses memory in an illegal way (for example, trying to write to a read-only location). Such faults usually lead to program termination to prevent further errors or corruption [3].





Segmentation Fault
Definition
Segmentation Fault
A segmentation fault occurs when a program attempts to access memory that it is not permitted to access or accesses memory in an illegal way (for example, trying to write to a read-only location). Such faults usually lead to program termination to prevent further errors or corruption [3].

1. **Key Points:**
   - A segmentation fault is a type of memory access error.
   - It happens when a program accesses a memory area that it is not allowed to (e.g., an address outside its allocated range) or uses improper access permissions.
   - The operating system detects these violations and terminates the program to protect the system from unintended behavior.

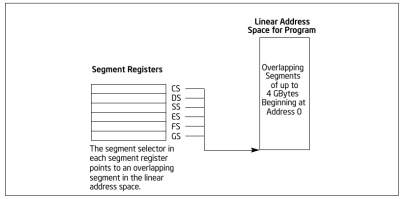



Figure 3-6. Use of Segment Registers for Flat Memory Model

## Segmentation Fault

A segmentation fault occurs when a program attempts to access memory that it is not permitted to access or accesses memory in an illegal way (for example, trying to write to a read-only location). Such faults usually lead to program termination to prevent further errors or corruption [3].





Segmentation Fault
Definition
Segmentation Fault
A segmentation fault occurs when a program attempts to access memory that it is not permitted to access or accesses memory in an illegal way (for example, trying to write to a read-only location). Such faults usually lead to program termination to prevent further errors or corruption [3].

1. **Proper Memory Allocation:** Always ensure that you allocate memory correctly (using functions like `malloc` in C or `new` in C++) and free it when it's no longer needed.
2. **Pointer Management:** Ensure that pointers are initialized properly and check for NULL before dereferencing. Avoid pointer arithmetic that could lead to invalid memory access.
3. **Bounds Checking:** Always perform bounds checking when accessing arrays or buffers to prevent accessing memory outside the allocated area.
4. **Use Debugging Tools:** Use tools like Valgrind or address sanitizers to detect illegal memory access during development.
5. **Code Reviews and Testing:** Regularly check and test your code, especially for edge cases where memory access might be unsafe.

# Advantages of Segmentation [1]

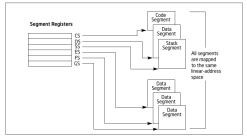


Figure 3-7. Use of Segment Registers in Segmented Memory Model

- Eliminates internal fragmentation by allocating only the exact amount of memory needed.
- Uses compact segment tables that consume less memory than traditional paging tables.
- Involves lower processing overhead compared to managing numerous page table entries.
- Simplifies relocation since segments can be moved individually rather than large contiguous blocks.





Advantages of Segmentation [1]
- Eliminates internal fragmentation by allocating only the exact amount of memory needed.
- Uses compact segment tables that consume less memory than traditional paging tables.
- Involves lower processing overhead compared to managing numerous page table entries.
- Simplifies relocation since segments can be moved individually rather than large contiguous blocks.

1. **Key Advantages Explained:**
   - **No Internal Fragmentation:** Unlike fixed-size paging, segmentation allocates memory according to the precise size of the data, thereby reducing wasted space.
   - **Compact Segment Tables:** The data structures used for segmentation (segment tables) are typically smaller and simpler, which saves memory.
   - **Reduced Processing Overhead:** With fewer and more flexible entries than page tables, the CPU spends less time on memory management tasks.
   - **Simpler Relocation:** Segments, being logical divisions of the address space, can be moved easily in physical memory without complex recalculations for every individual page.

# Advantages of Segmentation [1]

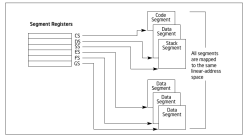


Figure 3-7. Use of Segment Registers in Segmented Memory Model

- Eliminates internal fragmentation by allocating only the exact amount of memory needed.
- Uses compact segment tables that consume less memory than traditional paging tables.
- Involves lower processing overhead compared to managing numerous page table entries.
- Simplifies relocation since segments can be moved individually rather than large contiguous blocks.





Advantages of Segmentation [1]
- Eliminates internal fragmentation by allocating only the exact amount of memory needed.
- Uses compact segment tables that consume less memory than traditional paging tables.
- Involves lower processing overhead compared to managing numerous page table entries.
- Simplifies relocation since segments can be moved individually rather than large contiguous blocks.

1. **Focus for Students:**
   - Understand how segmentation's variable-sized allocation avoids the wasted space seen in paging.
   - Recognize the trade-offs between memory management simplicity and the potential for external fragmentation.
   - Consider why these advantages, while significant in earlier systems, may be less impactful in modern architectures that favor paging.

# Disadvantages of Segmentation[1]

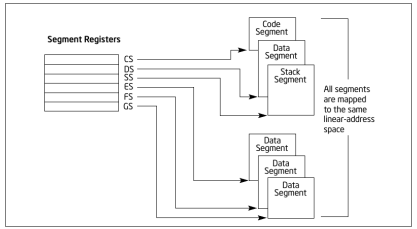


Figure 3-7. Use of Segment Registers in Segmented Memory Model

- Segmentation causes external fragmentation to the point that modern x86-64 servers treat it as a legacy application and only support it for backward compatibility.
- As fragmentation worsens, system performance degrades significantly.





Disadvantages of Segmentation[1]
- Segmentation causes external fragmentation to the point that modern x86-64 servers treat it as a legacy application and only support it for backward compatibility.
- As fragmentation worsens, system performance degrades significantly.

1. **Expanded Explanation:**
   - **External Fragmentation:** In a segmented memory system, memory is allocated in variable-sized blocks. Over time, as segments are allocated and deallocated, small unused gaps can appear between segments. These gaps are too small to be useful for new allocations even though the total free memory may be sufficient.
   - **Impact on Performance:** As these gaps accumulate, the system may struggle to find contiguous blocks of memory for new segments. This can lead to increased overhead as the operating system tries to manage and, if necessary, compact the memory to reduce fragmentation.
   - **Legacy Usage:** Due to these issues, modern systems—particularly in x86-64 architecture—treat

# Disadvantages of Segmentation[1]

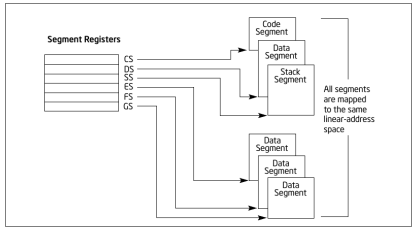


Figure 3-7. Use of Segment Registers in Segmented Memory Model

- Segmentation causes external fragmentation to the point that modern x86-64 servers treat it as a legacy application and only support it for backward compatibility.
- As fragmentation worsens, system performance degrades significantly.





Disadvantages of Segmentation[1]
- Segmentation causes external fragmentation to the point that modern x86-64 servers treat it as a legacy application and only support it for backward compatibility.
- As fragmentation worsens, system performance degrades significantly.

1. **Key Points for Students:**
   - Understand that while segmentation allows for flexible memory allocation, its use can lead to inefficient memory usage through external fragmentation.
   - Recognize that external fragmentation means memory can become unusable even when there appears to be enough total free space.
   - Note that this fragmentation issue is one of the main reasons modern operating systems have shifted to flat memory models combined with paging.

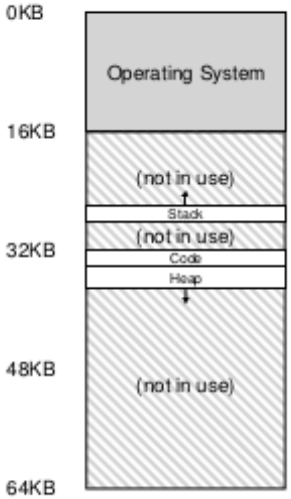
# Example



Example

# Setup

Ref: P. Bridges, University of New Mexico



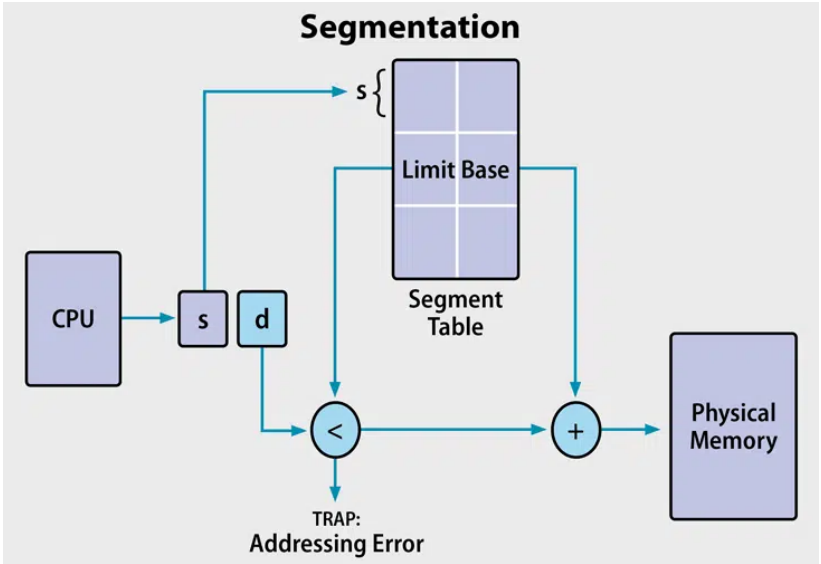

| Name | Index | Base | Size |
|---|---|---|---|
| Heap (ES) | 00b | 34K | 2K |
| Code (CS) | 01b | 32K | 2K |
| Stack (SS) | 10b | 28K | 2K |





Setup
Ref: P. Bridges, University of New Mexico
0KB
Operating System
16KB
(not in use)
Stack
(not in use)
32KB
Code
Heap
48KB
(not in use)
64KB
Name Index Base Size
Heap (ES) 00b 34K 2K
Code (CS) 01b 32K 2K
Stack (SS) 10b 28K 2K

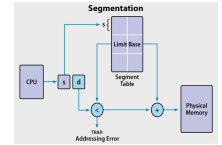
# Segmentation Addressing Diagram







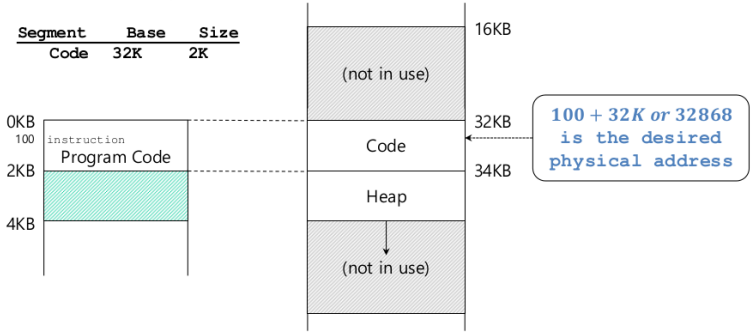

Segmentation Addressing Diagram



# Code Address Translation on Segmentation

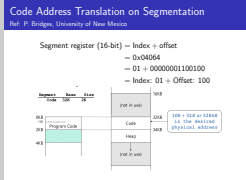
Ref: P. Bridges, University of New Mexico

$$\begin{aligned}\text{Segment register (16-bit)} &= \text{Index} + \text{offset} \\ &= 0x04064 \\ &= 01 + 00000001100100 \\ &= \text{Index: } 01 + \text{Offset: } 100\end{aligned}$$







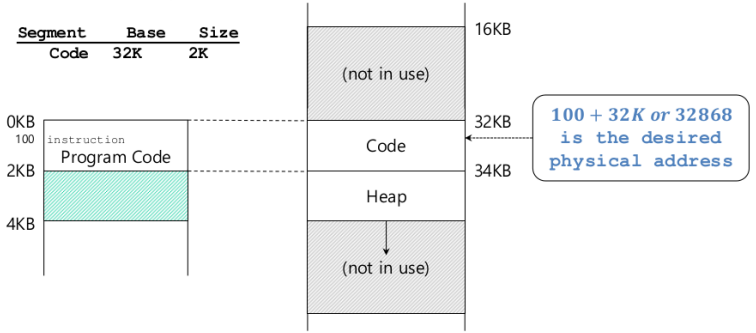




1. **Key Points:**
   - The segmentation mechanism uses a 16-bit segment register that is conceptually divided into two parts: an **Index** and an **Offset**.
   - In our example, the 16-bit value is 0x04064. Here, the first part (Index) is 01 and the second part (Offset) is 00000001100100, which we interpret as 100.
   - The Index indicates which segment is being referenced—in this case, the code segment. The Offset specifies the exact location within that segment.
   - Typically, for the code segment, the segment's base address is set to 0, so the effective (or linear) address is equal to the offset.

# Code Address Translation on Segmentation

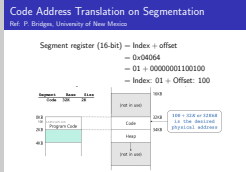
Ref: P. Bridges, University of New Mexico

$$\begin{aligned}\text{Segment register (16-bit)} &= \text{Index} + \text{offset} \\ &= 0x04064 \\ &= 01 + 00000001100100 \\ &= \text{Index: } 01 + \text{Offset: } 100\end{aligned}$$







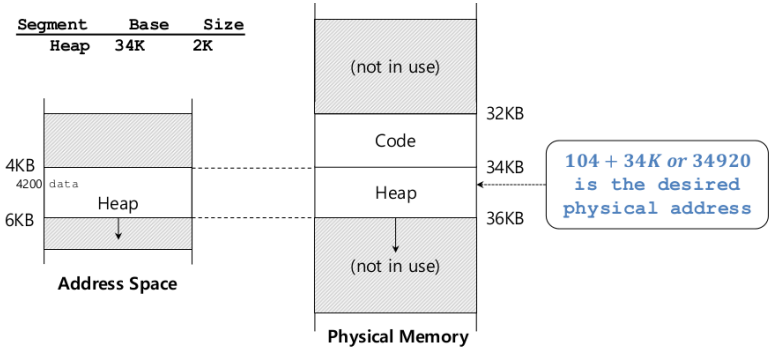




1. **Context for Students:**
   - Understand that this mechanism was essential in earlier architectures for mapping segmented addresses to physical addresses.
   - Note how the separation into Index and Offset allows the operating system to maintain different segments (like code, data, stack) in separate areas of physical memory.
   - Remember that, in modern 64-bit systems, segmentation is mostly disabled, and a flat memory model is used; however, knowing this process is key to understanding the evolution of memory management in operating systems.

# Heap Address Translation on Segmentation

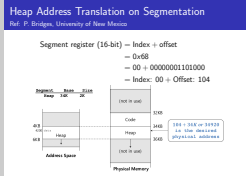
Ref: P. Bridges, University of New Mexico

$$\begin{aligned}\text{Segment register (16-bit)} &= \text{Index} + \text{offset} \\ &= 0x68 \\ &= 00 + 00000001101000 \\ &= \text{Index: } 00 + \text{Offset: } 104\end{aligned}$$







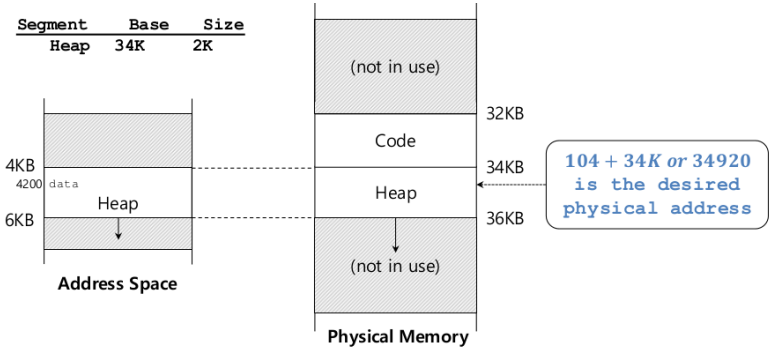




1. **Identifying the Heap Segment:** The index value of 00 identifies the segment in the segment table corresponding to the heap. This indicates that we are accessing the heap segment.
2. **Context for Memory Management:** This process illustrates how segmentation allows the operating system to divide a process's virtual address space into logical segments. Every segment, like the heap, can occupy any location in physical memory but is accessed via its segment index offset.

# Heap Address Translation on Segmentation

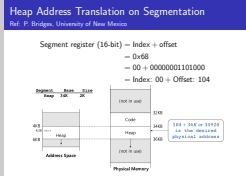
Ref: P. Bridges, University of New Mexico

$$\begin{aligned}\text{Segment register (16-bit)} &= \text{Index} + \text{offset} \\ &= 0x68 \\ &= 00 + 00000001101000 \\ &= \text{Index: } 00 + \text{Offset: } 104\end{aligned}$$








1. **Key Points for Students:**
   - Understand that the segmentation mechanism breaks the address into an index (to select the segment) and an offset (to locate the exact position within that segment).
   - Recognize that in this example, the heap segment is identified by index 00, and the offset 104 tells us the position within the heap.
   - Note that the operating system uses the segment table to map the segment (here, the heap) to a specific virtual address (starting at 0x34816 in this case), allowing for flexible memory management.

# Heap Address Translation on Segmentation

Ref: P. Bridges, University of New Mexico

- **External Fragmentation**: little holes of free space in physical memory that make it difficult to allocate new segments:
  - There is 24KB free, but not in one contiguous segment.
  - The OS cannot satisfy the 20KB request
- **Compaction**: rearranging the exiting segments in physical memory.
  - Compaction is costly.
    - Stop running process.
    - Copy data to somewhere.
    - Change segment register value.
- The more segments you have, the worse it is.

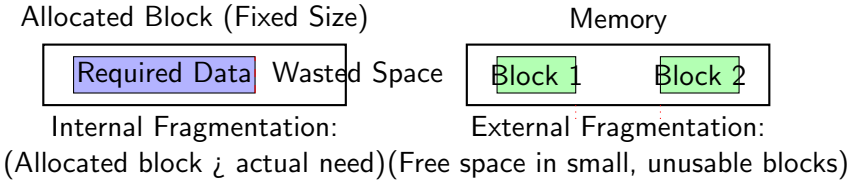




Heap Address Translation on Segmentation
Ref: P. Bridges, University of New Mexico
External Fragmentation: little holes of free space in physical memory that make it difficult to allocate new segments:
There is 24KB free, but not in one contiguous segment.
The OS cannot satisfy the 20KB request
Compaction: rearranging the exiting segments in physical memory.
Compaction is costly.
Stop running process.
Copy data to somewhere.
Change segment register value.
The more segments you have, the worse it is.

# Fragmentation in Memory Management

## Internal vs External Fragmentation





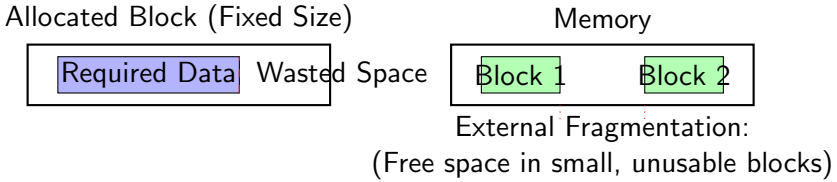



Fragmentation in Memory Management
Internal vs External Fragmentation


1. **Segmentation and Internal Fragmentation:** One of the advantages of segmentation is that it allows variable-sized allocations. This means that each segment can be allocated based on the exact memory needed for its contents, thus minimizing internal fragmentation (the wasted space inside an allocated segment).
2. **Segmentation and External Fragmentation:** However, as segments are allocated and deallocated over time, the available memory may become fragmented into small, noncontiguous blocks. Even though the overall free memory might be sufficient for a new allocation, if it isn't in one contiguous block, the allocation may fail. This phenomenon is known as external fragmentation.

# Fragmentation in Memory Management

## Internal vs External Fragmentation

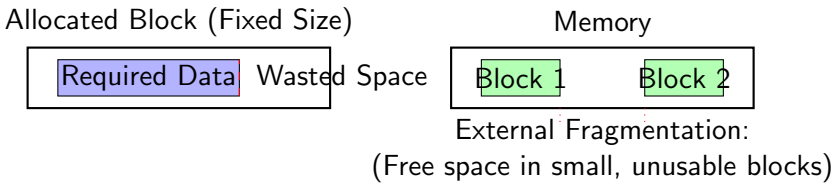







Fragmentation in Memory Management
Internal vs External Fragmentation


1. **Additional Considerations:** Sometimes, segments are allocated with extra space to accommodate future growth. While this strategy can reduce the need for frequent relocations, it may also contribute to memory waste if that additional space is never used, potentially worsening external fragmentation.





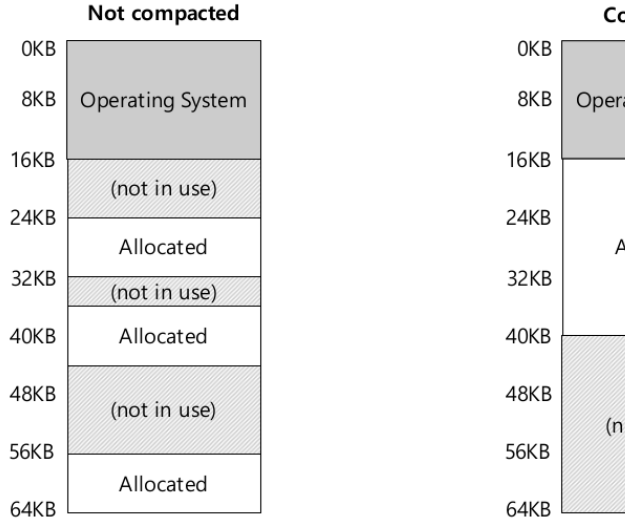


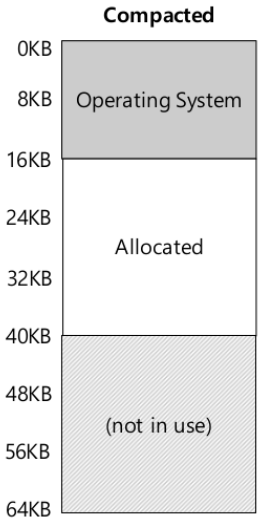


1. **Focus for Students:**
   - Understand that segmentation is designed to avoid internal fragmentation by matching allocation sizes to program needs.
   - Recognize that despite this benefit, segmentation can lead to external fragmentation as memory is dynamically allocated and freed over time.
   - Consider how modern memory management techniques, such as paging, help mitigate external fragmentation by dividing memory into fixed-size pages.
   - Reflect on the trade-offs in memory management strategies: variable-sized allocations (segmentation) versus fixed-sized allocations (paging).

# Memory Compaction

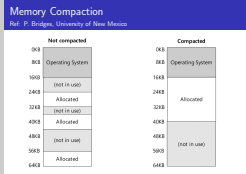
Ref: P. Bridges, University of New Mexico







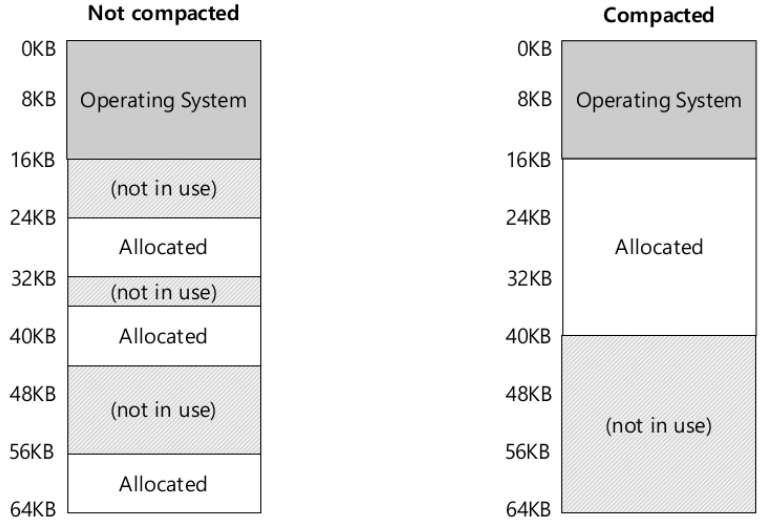




1. **Overview of Memory Compaction:**
   - Memory compaction is a technique used by the operating system to consolidate free memory.
   - Its primary goal is to counteract external fragmentation by rearranging memory segments so that free memory is available as one large contiguous block.

   **Why is Compaction Necessary?**
   - Over time, as processes allocate and free memory, the available free space becomes fragmented into small, scattered gaps.
   - Even if the total free memory is sufficient to meet a request, the lack of contiguous space may prevent the allocation of large segments.

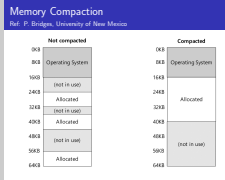
# Memory Compaction

Ref: P. Bridges, University of New Mexico







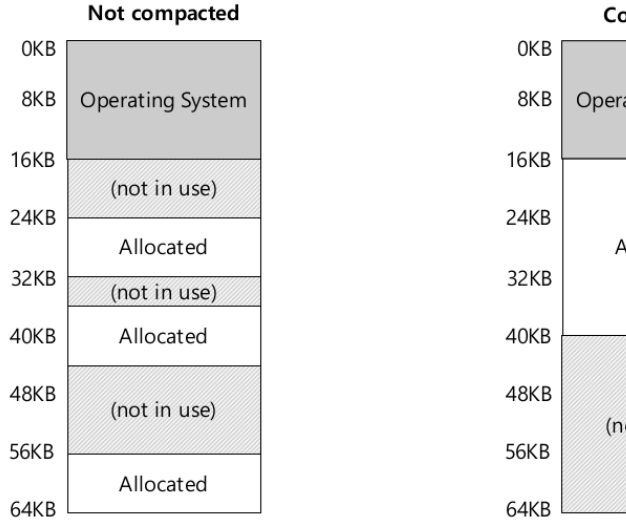




1. **The Process of Compaction:**
   - The OS temporarily halts running processes to move memory segments safely.
   - Segments are copied from their current locations into new contiguous regions.
   - After the data is moved, the system updates the segment registers and any associated pointers to reflect the new addresses.

   **Trade-Offs and Considerations:**
   - While compaction can improve the usability of free memory, it is resource-intensive.
   - The process involves extra CPU overhead memory bandwidth usage and may cause a temporary pause or performance degradation in running applications.

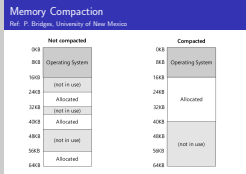
# Memory Compaction

Ref: P. Bridges, University of New Mexico








1. **Takeaways for Students:**
   - Understand that compaction is a remedy for external fragmentation in segmentation-based systems.
   - Recognize the trade-off between optimizing memory utilization and incurring performance overhead.
   - Compare this with modern paging systems, which address fragmentation differently.

# Conclusion

# Conclusion

Ref: P. Bridges, University of New Mexico

Segmentation is variable length allocation

- Just like malloc free lists, with many of the same problems.
- It's useful and flexible but hard to manage well.
- Particularly when you have lots of segments (e.g., from either lot of segments per process or lots of processes)





Conclusion
Ref: P. Bridges, University of New Mexico
Segmentation is variable length allocation
- Just like malloc free lists, with many of the same problems.
- It's useful and flexible but hard to manage well.
- Particularly when you have lots of segments (e.g., from either lot of segments per process or lots of processes)

# Conclusion (cont.)

Ref: P. Bridges, University of New Mexico

Modern OSes make only minimal use of segmentation

- 32-bit mode x86 (introduced with 80286) can use segments extensively, but most OSes (e.g., Windows and Linux) don't.
- 64-bit mode x86 forces most segments to have a base address of 0.
- With a very narrow exception usually used for thread-specific data.





Conclusion (cont.)
Ref: P. Bridges, University of New Mexico
Modern OSes make only minimal use of segmentation
- 32-bit mode x86 (introduced with 80286) can use segments extensively, but most OSes (e.g., Windows and Linux) don't.
- 64-bit mode x86 forces most segments to have a base address of 0.
- With a very narrow exception usually used for thread-specific data.

# References I


C. Taylor.
Paging vs. segmentation: Core differences explained, 2023.
https://www.enterprisestorageforum.com/hardware/paging-and-segmentation/.

Intel.
*Intel 64 and IA-32 Architectures Software Developer's Manual Combined Volumes: 1, 2A, 2B, 2C, 2D, 3A, 3B, 3C, 3D and 4.*
2200 Mission College Blvd. Santa Clara, CA 95054, 2021 [Online].
https://cdrdv2.intel.com/v1/dl/getContent/671200.

Srinivas.
How to diagnose and locate segmentation faults in x86 assembly, 2021.
https://resources.infosecinstitute.com/topic/how-to-diagnose-and-locate-segmentation-faults-in-x86-assembly/.







References I