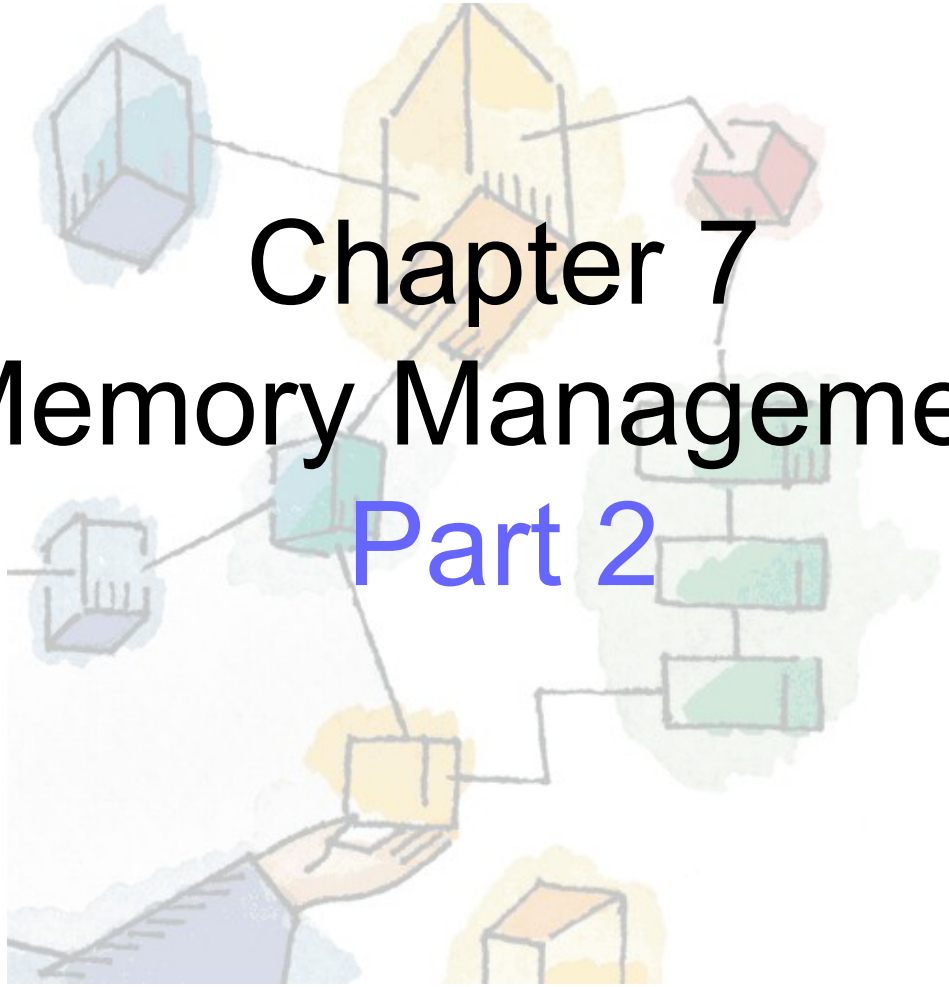
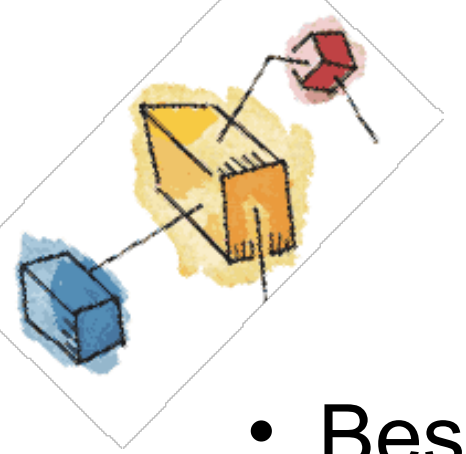
*Operating Systems:*
*Internals and Design Principles, 8/E*
William Stallings



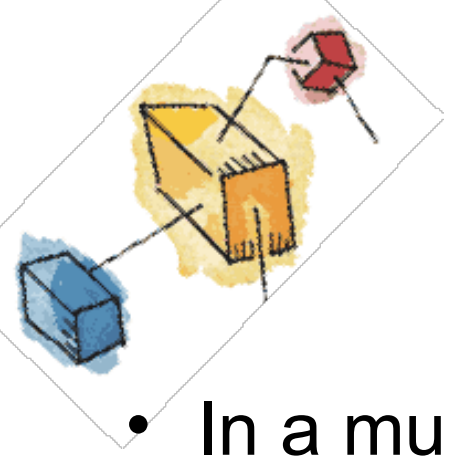
# Chapter 7
# Memory Management
## Part 2

Patricia Roy
Manatee Community College, Venice, FL



# Placement Algorithm

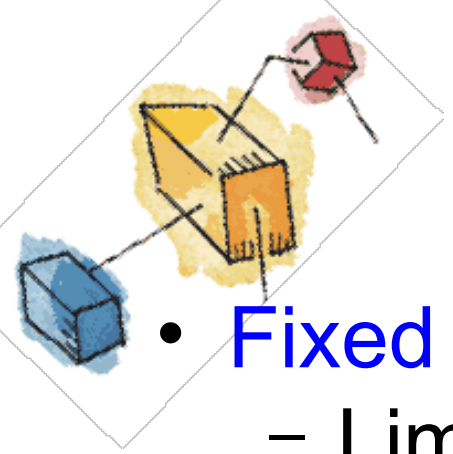
- Best-Fit
- First-Fit
- Next-Fit
- Compaction



# Replacement Algorithm

- In a multi-programming system, we may arrive at a point where all of the processes in main memory are in blocked state, and insufficient memory for additional process even after compaction.

- To avoid wasting processor time:
  - OS will swap one of the processes out of main memory to make room for a new process
  - But, which process to be swapped / replaced???



# Drawbacks

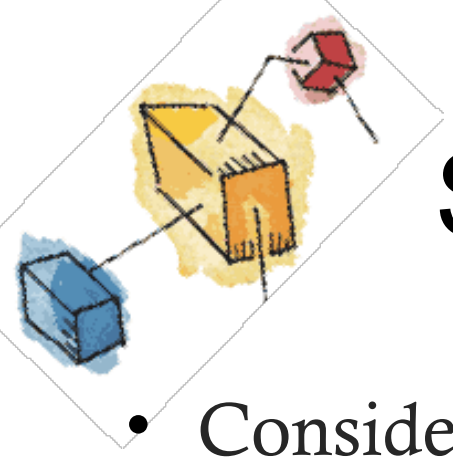
- Fixed partition:
  - Limits the number of active processes
  - May use the space inefficiently (in case of poor match)
  - Internal fragmentation

- Dynamic partition:
  - More complex to maintain
  - Incur overhead from compaction
  - External fragmentation

# Solution: Buddy System

- Comprises of fixed and dynamic partitioning
- Space available for allocation is treated as a single block
- Memory blocks are available of size *words,* $2^K$ , $L \leq K \leq U$, where

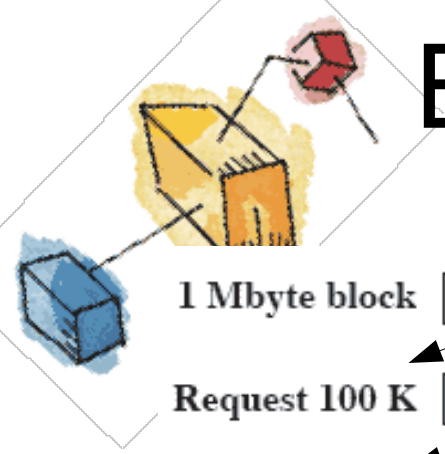
  - $2^L$ = *smallest size block that is allocated*
  - $2^U$ = largest size block that is allocated; generally 2U is the size of the entire memory available for allocation



# Solution: Buddy System

- Consider a single block of size 2^U (Maximum)
- If a request of size *s* such that
  - 2(U-1) < s< 2U is made, then the entire block is allocated.
  - Otherwise the block is split into 2 equal buddies of size 2(U-1).
  - If 2(U-2) < s <= 2(U-1), then the request is allocated to one of the two buddies.
    - Otherwise one of the buddies is split into half again

# Example of Buddy System



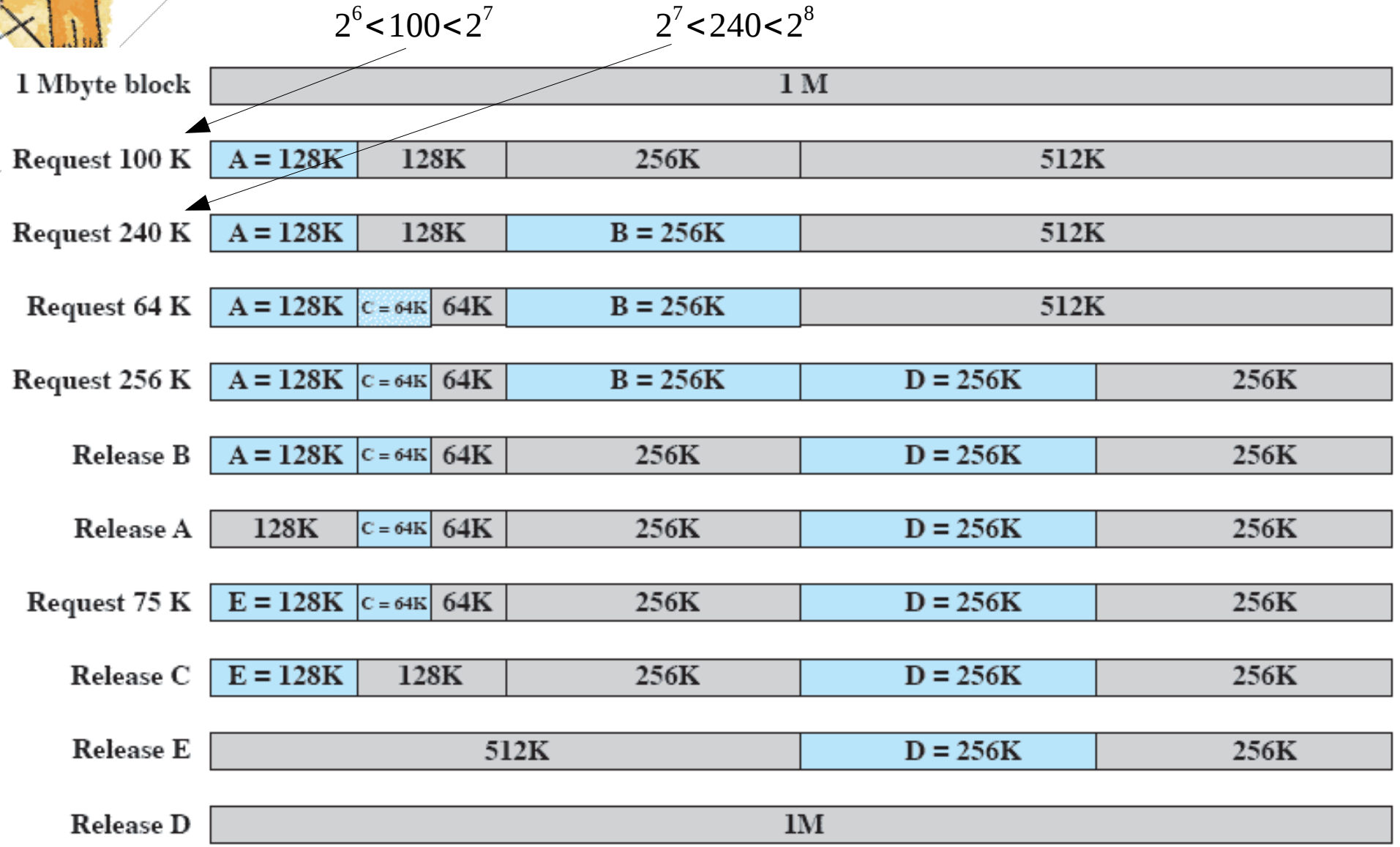
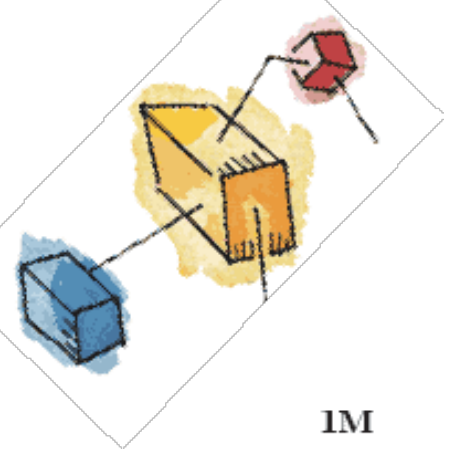



**Figure 7.6 Example of Buddy System**



# Tree Representation of Buddy System

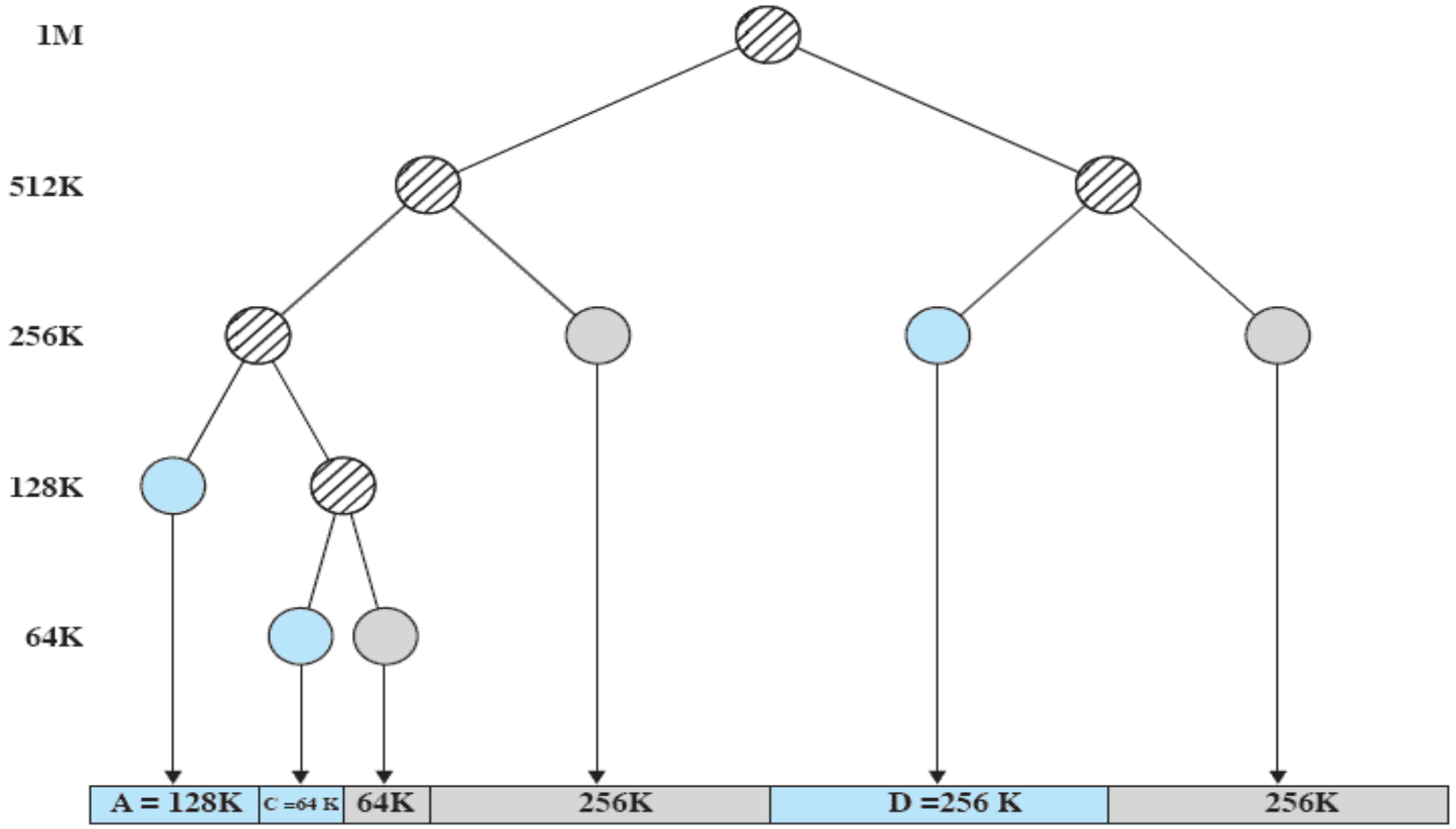
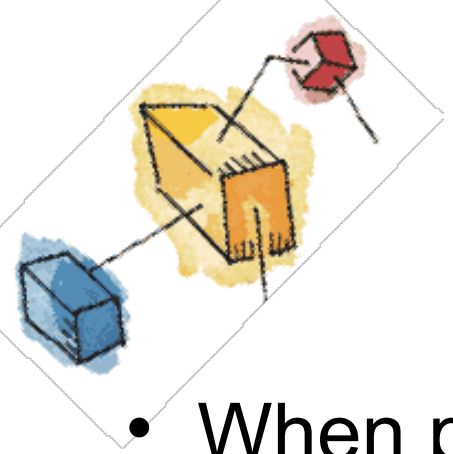



Figure 7.7 Tree Representation of Buddy System

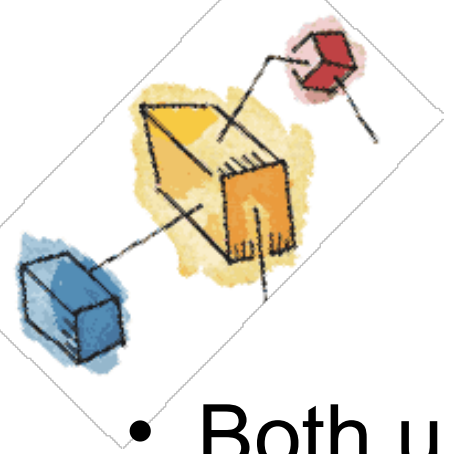


# Relocation

- When program loaded into memory the actual (absolute) memory locations are determined
- A process may occupy different partitions which means different absolute memory locations during execution (from swapping)
- The location referenced by a process will change each time a process is swapped in or shifted.

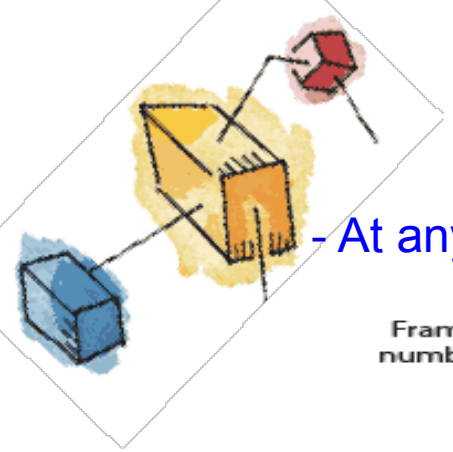
# Addresses

- Solution to the relocation problem:
  - Logical
    - Reference to a memory location independent of the current assignment of data to memory
    - Translation must be made to the physical address (actual location in main memory)
  - Relative
    - Address expressed as a location relative to some known point (e.g: value in processor register)



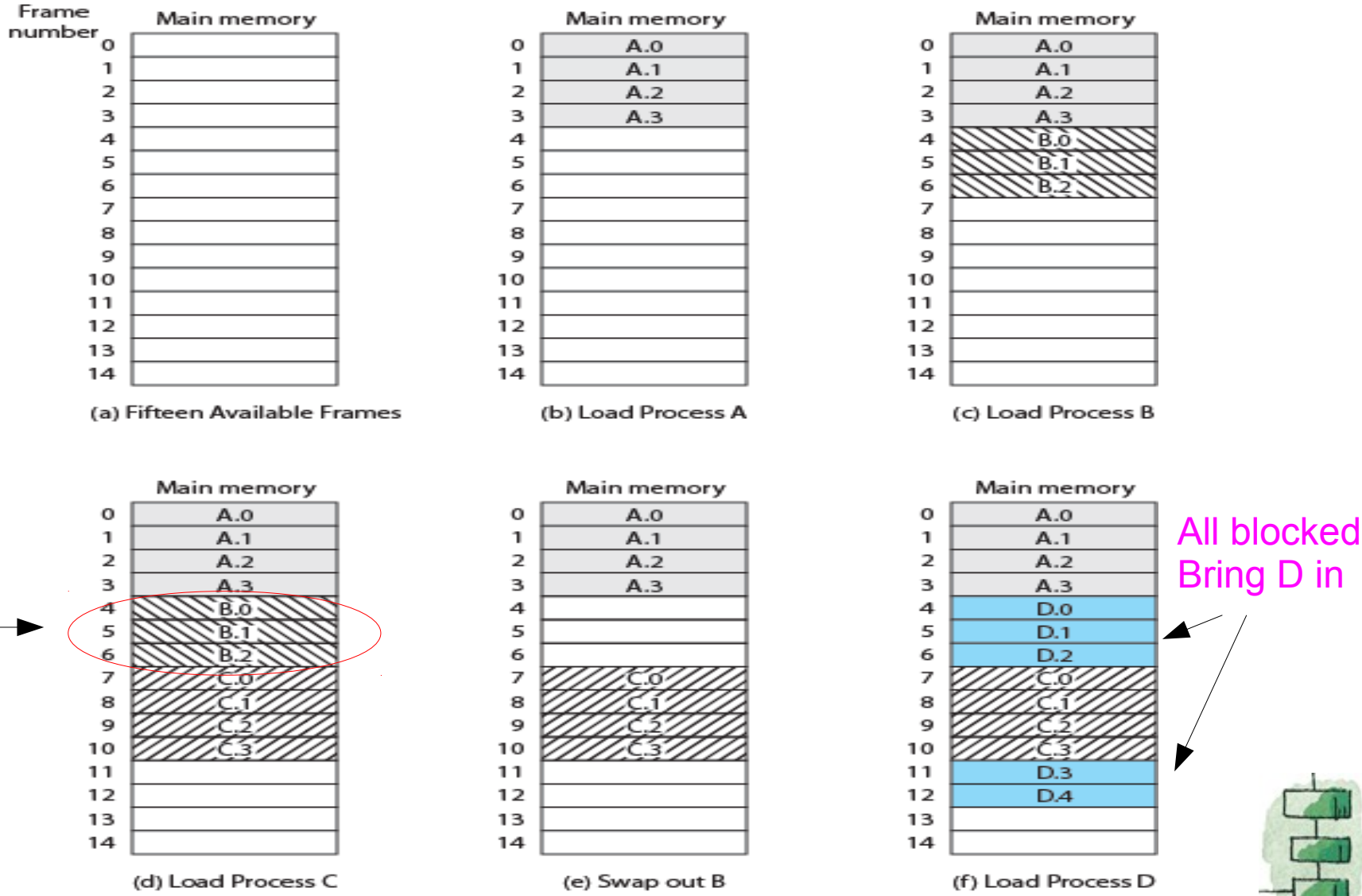
# Paging

- Both unequal fixed-size and variable-size partitions are inefficient in the use of memory
- Suppose partition memory into small equal fixed-size chunks and divide each process into small fixed-size chunks of the same
- The chunks of a process are called pages and chunks of memory are called frames



# Process and Frames

- At any given point in time, some of the frames are in used, and some are free

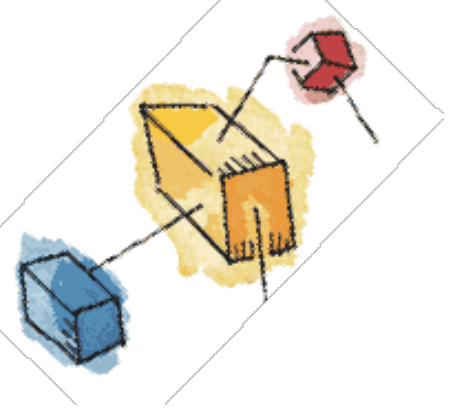



Suspended!
Swapped out

All blocked
Bring D in



Figure 7.9 Assignment of Process Pages to Free Frames



# Page Table

| | |
|---|---|
| 0 | 0 |
| 1 | 1 |
| 2 | 2 |
| 3 | 3 |

Process A page table

| | |
|---|---|
| 0 | — |
| 1 | — |
| 2 | — |

Process B page table

| | |
|---|---|
| 0 | 7 |
| 1 | 8 |
| 2 | 9 |
| 3 | 10 |

Process C page table

| | |
|---|---|
| 0 | 4 |
| 1 | 5 |
| 2 | 6 |
| 3 | 11 |
| 4 | 12 |

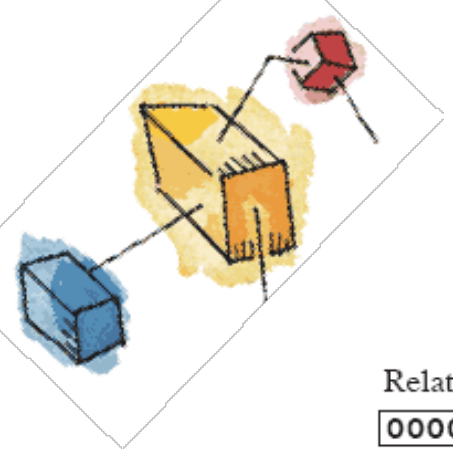
Process D page table

| |
|---|
| 13 |
| 14 |

Free frame list

Show the frame location for each page of the process
Need translation from logical-to-physical address by processor
Logical Address = (page number, offset)
Physical Address = (frame number, offset)

**Figure 7.10 Data Structures for the Example of Figure 7.9 at Time Epoch (f)**



# Logical Addresses

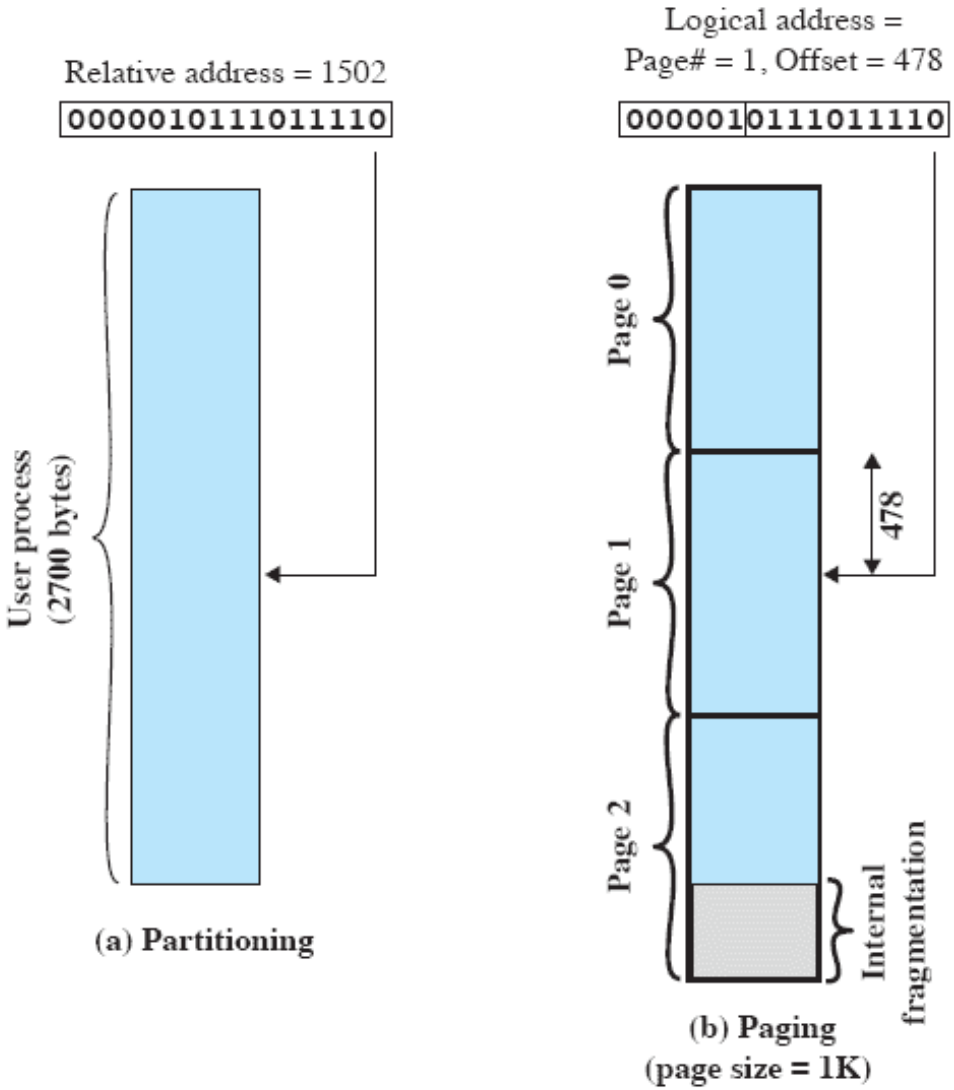
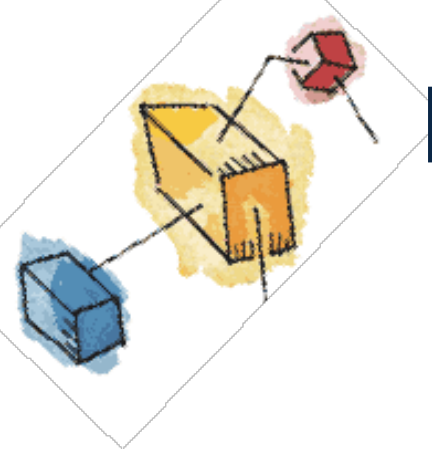
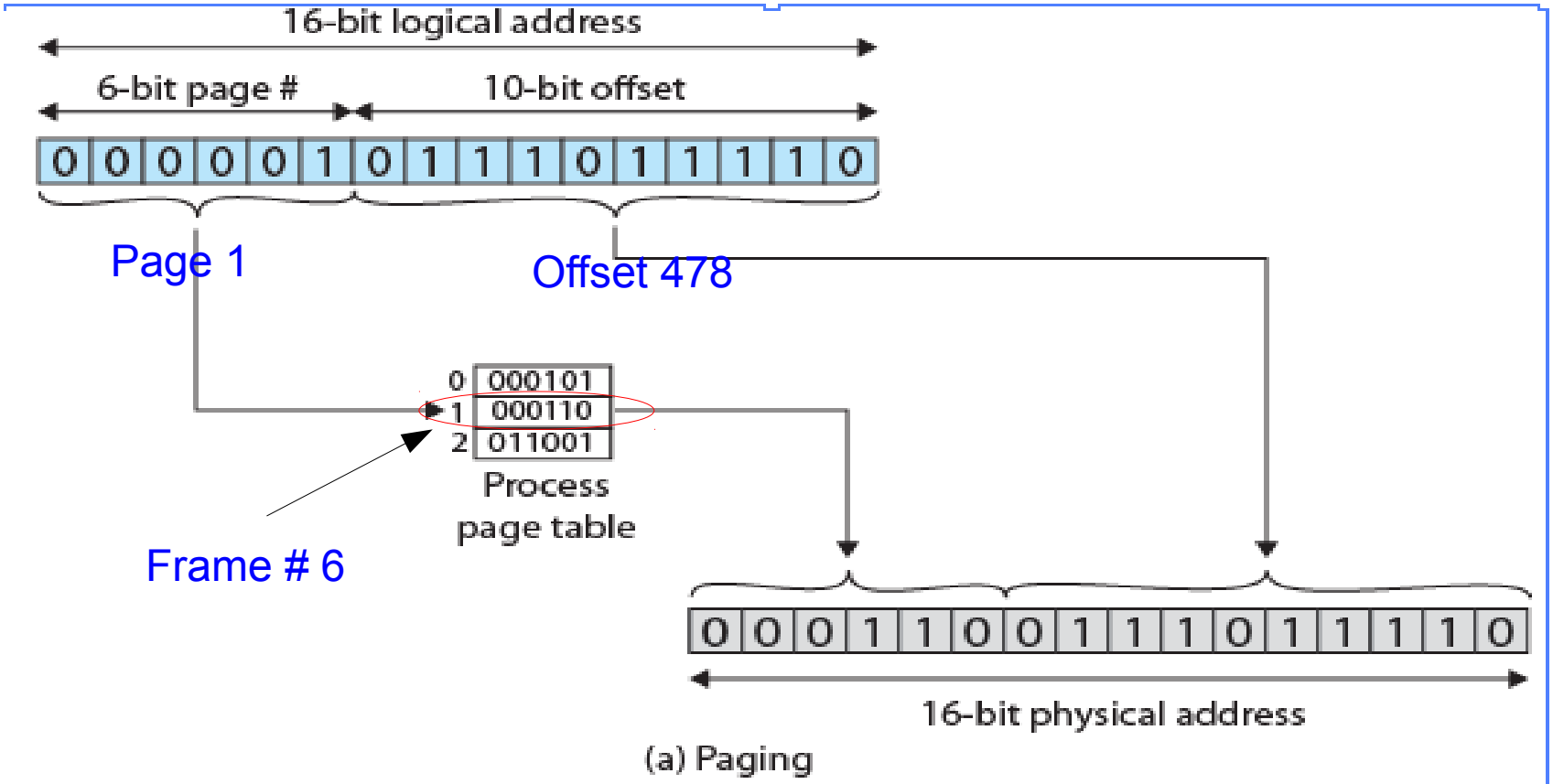


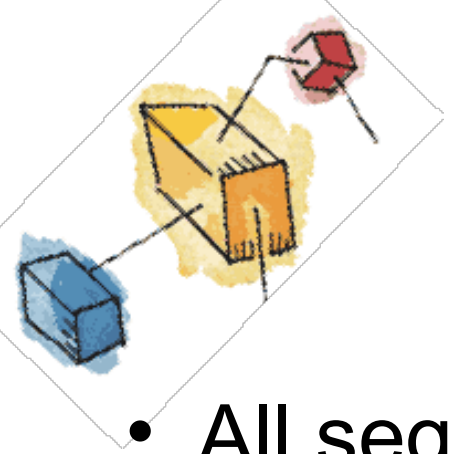

**Figure 7.11 Logical Addresses**



# Logical-to-Physical Address Translation - Paging



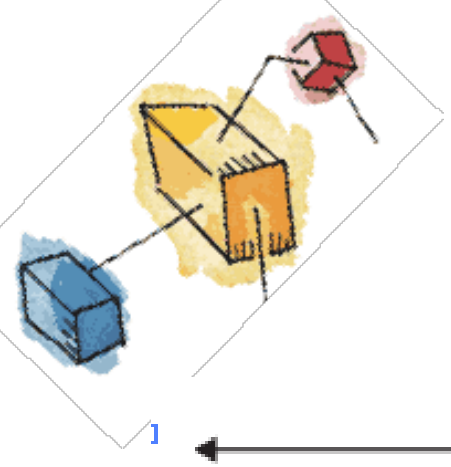

(a) Paging

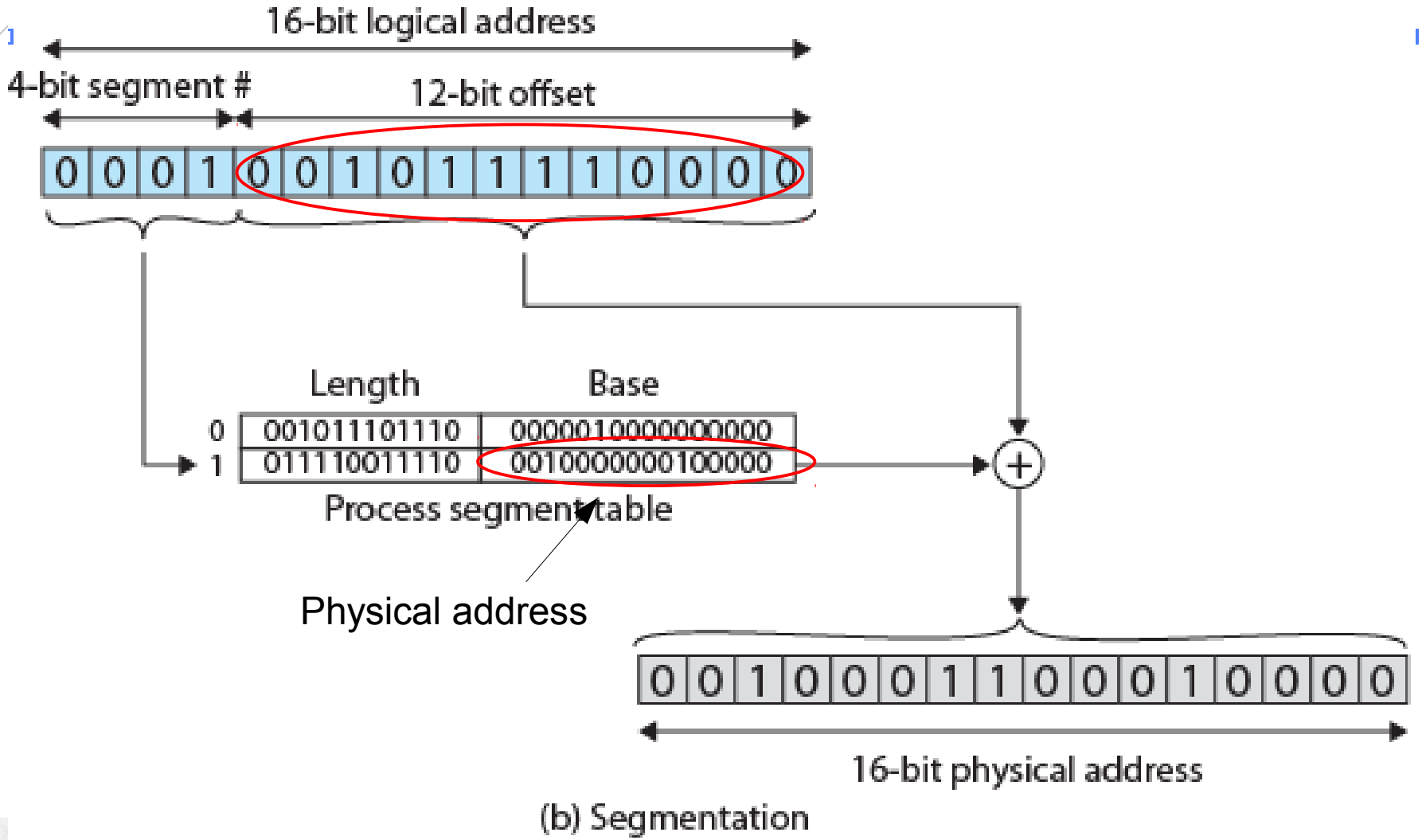


# Segmentation

- All segments of all programs do not have to be of the same length
- There is a maximum segment length
- Addressing consist of two parts - a segment number and an offset
- Since segments are not equal, segmentation is similar to dynamic partitioning

# Logical-to-Physical Address Translation - Segmentation






(b) Segmentation