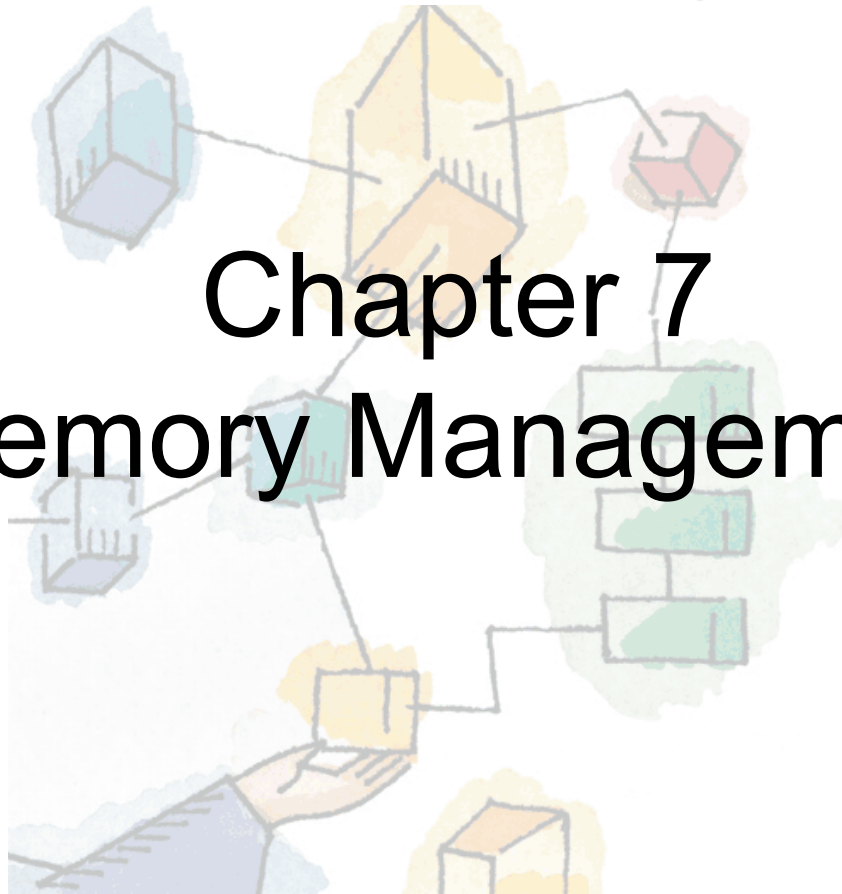


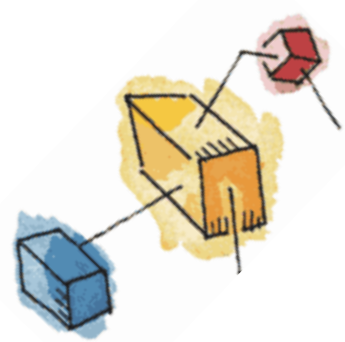
*Operating Systems:
Internals and Design Principles*
William Stallings

Chapter 7
Memory Management



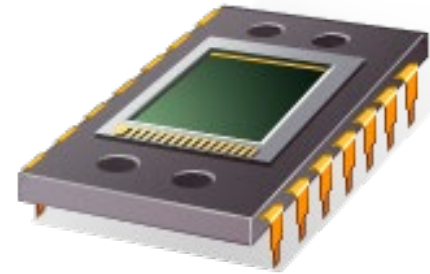
Outline

- Memory management requirements
- Contiguous partitioning
 - Fixed partitioning
 - Dynamic partitioning
 - Buddy system
- Non-contiguous allocation
 - Paging
 - Segmentation



Memory Management Requirements

- Memory management is intended to deal with the following issues:
 - Allocation/Relocation
 - Protection
 - Sharing



Allocation/Relocation

- Programmers typically do not know in advance which other programs will be resident in main memory at the time of execution of their program
- Active processes need to be able to be swapped in and out of main memory in order to maximize processor utilization
- A process may be swapped to disk and return to main memory at a different location (relocated)



Addresses



Logical

- reference to a memory location independent of the current assignment of data to memory

Relative

- address is expressed as a location relative to some known point

Physical or Absolute

- actual location in main memory
- 
- 

Addressing Requirements

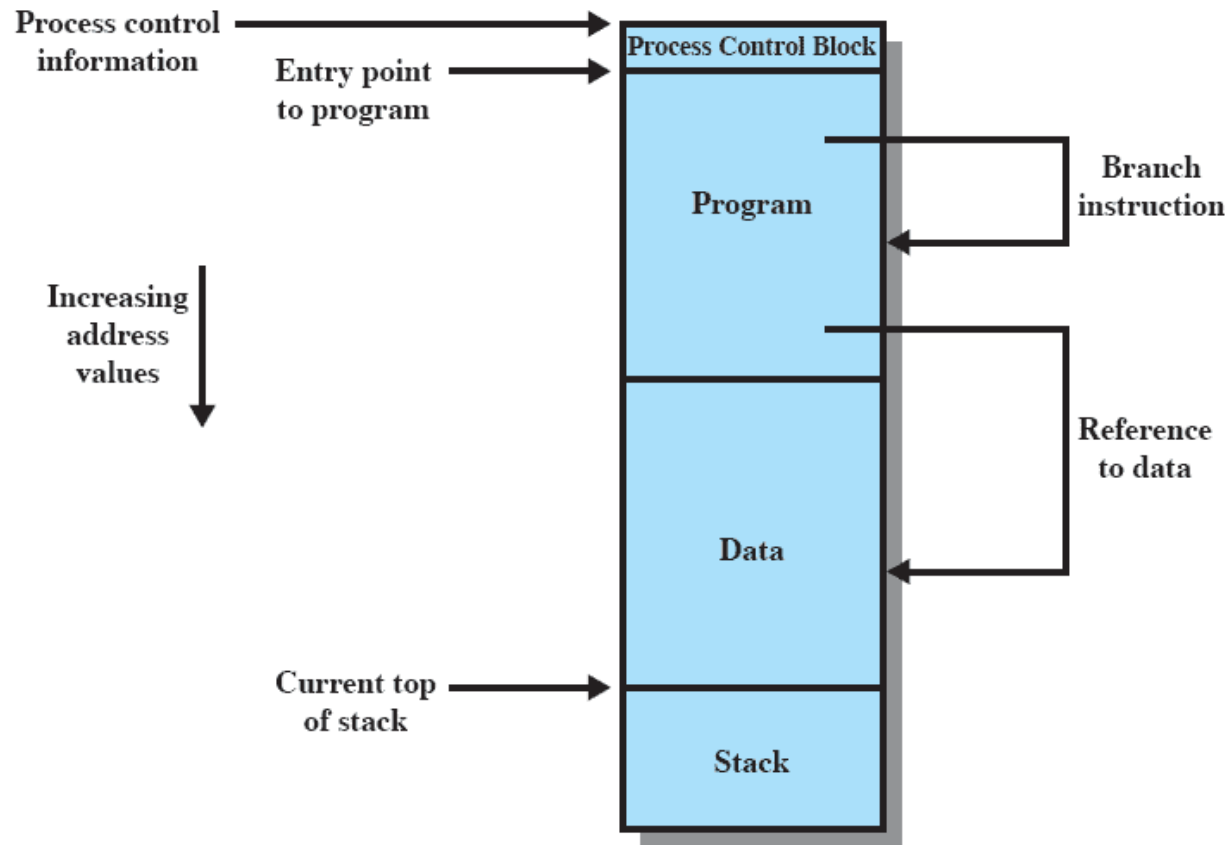
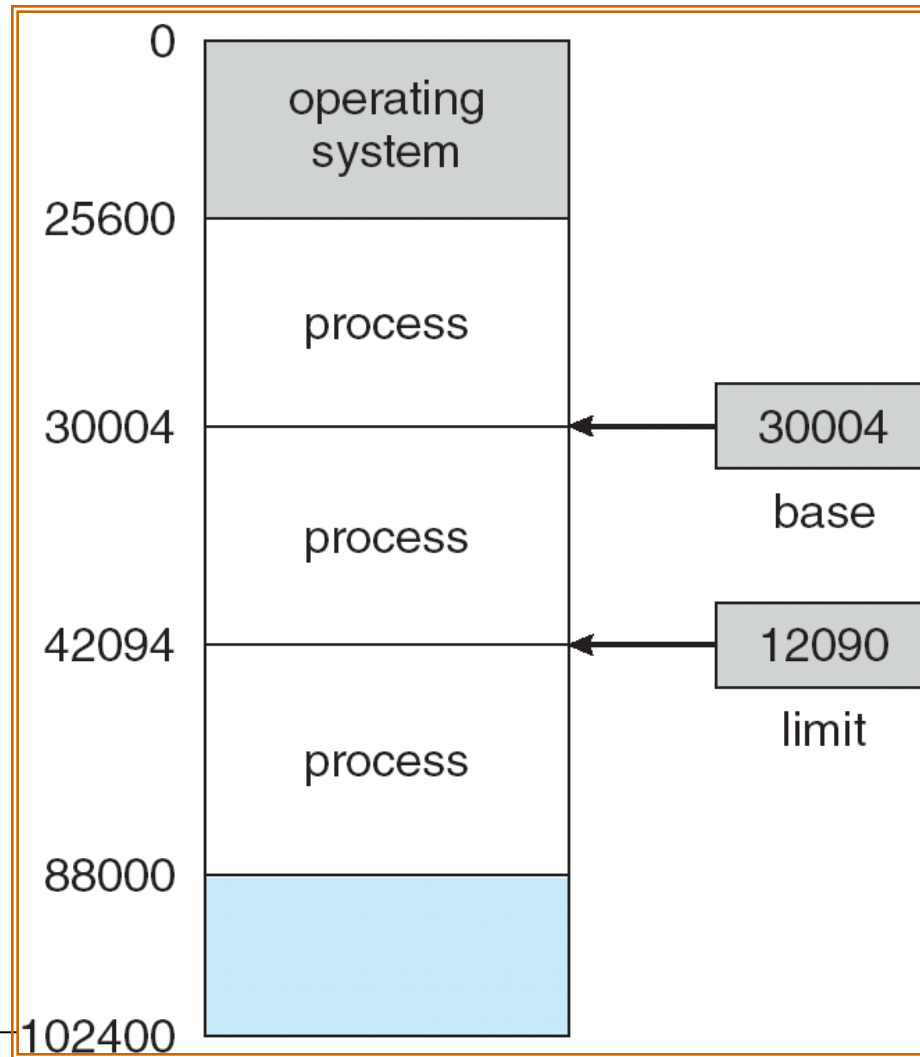


Figure 7.1 Addressing Requirements for a Process

Contiguous Allocation



Relocation

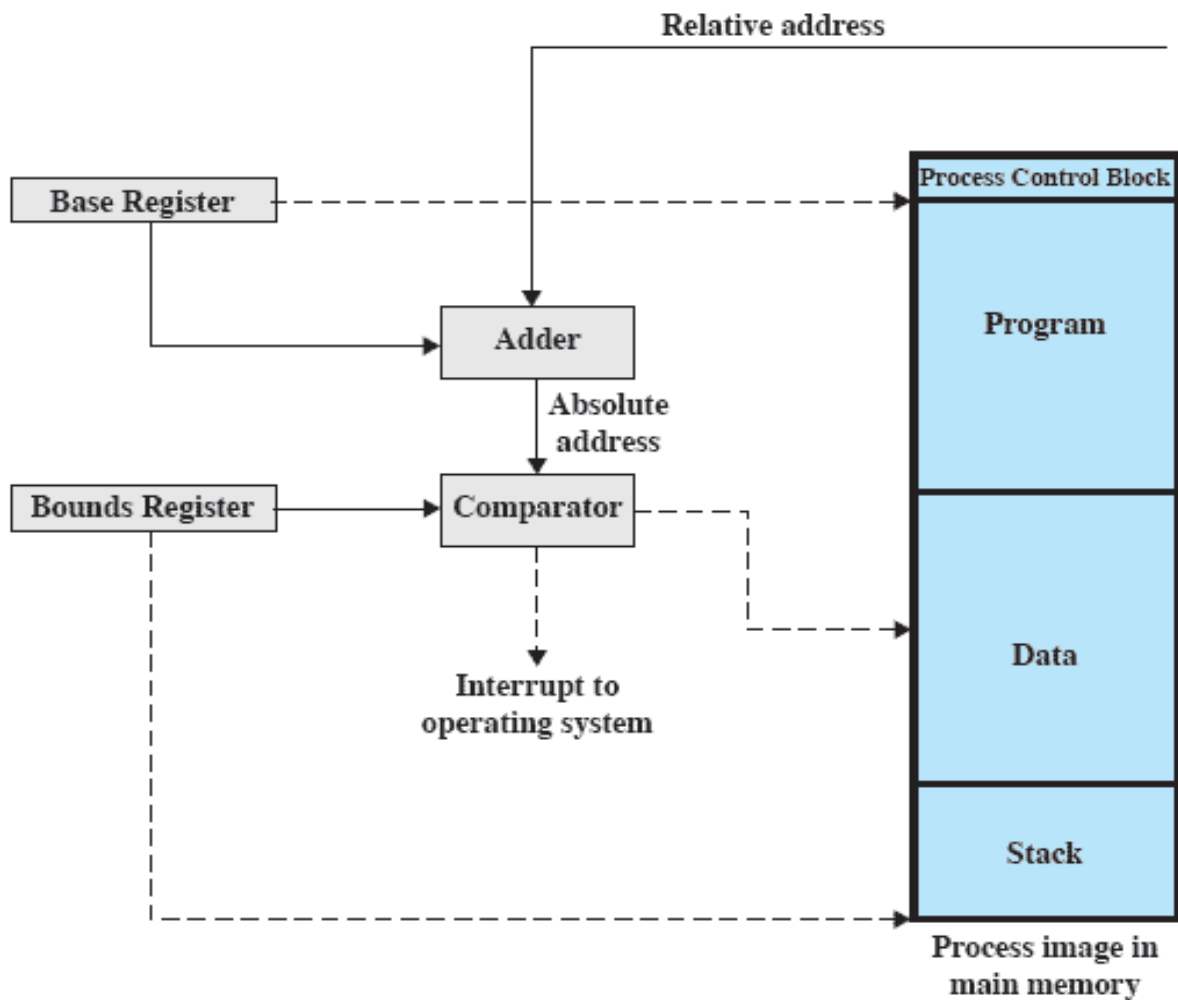
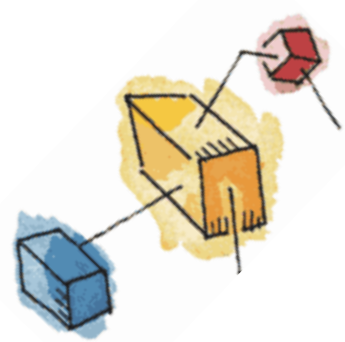


Figure 7.8 Hardware Support for Relocation

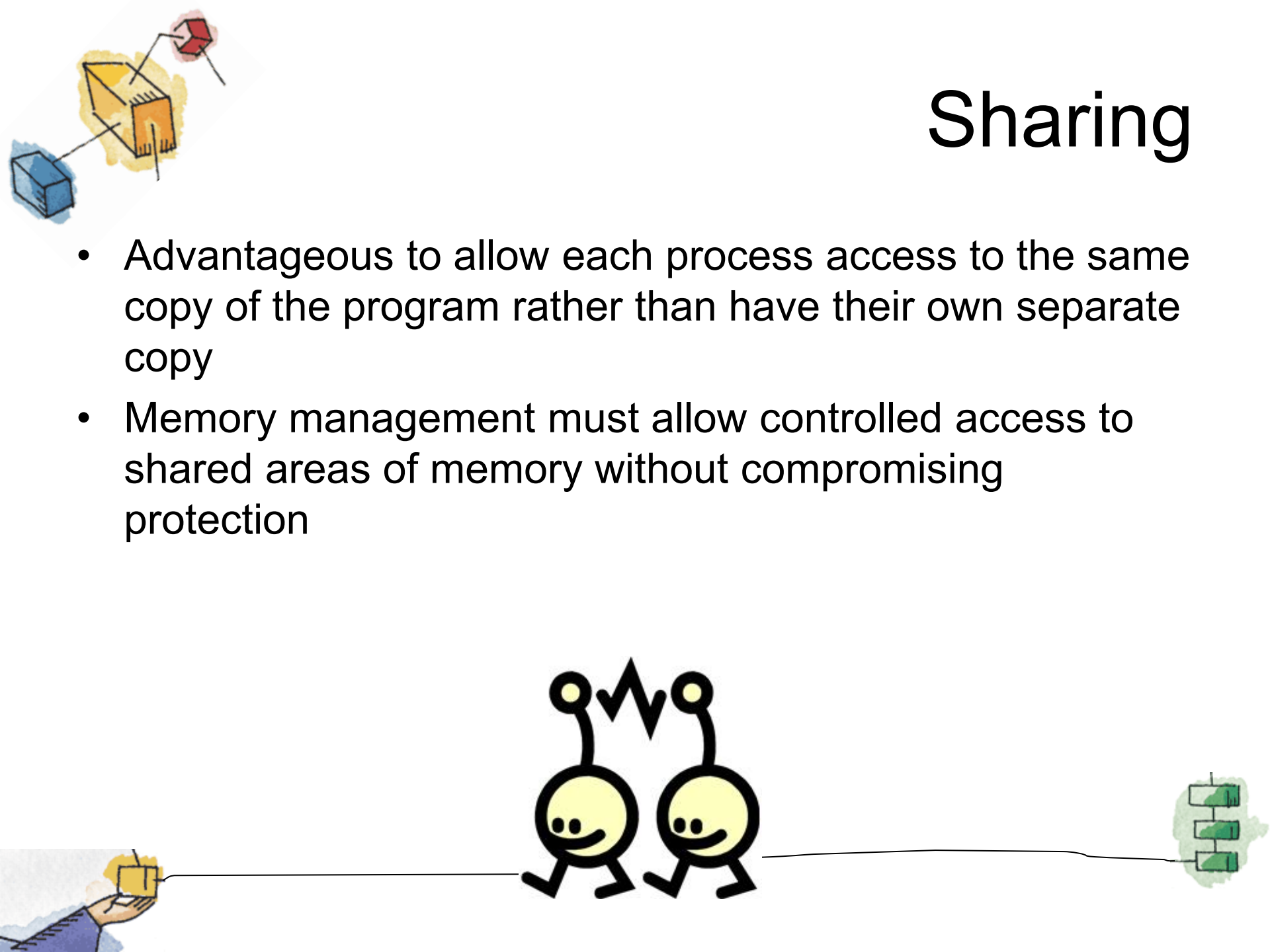
Protection

- Processes need to acquire permission to reference memory locations for reading or writing purposes
- Location of a program in main memory is unpredictable
- Memory references generated by a process must be checked at run time
- Mechanisms that support relocation also support protection



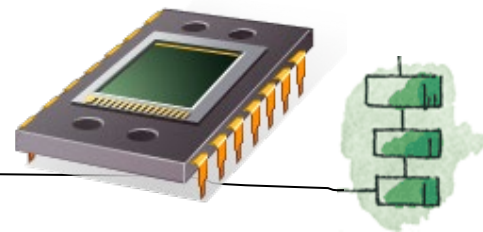
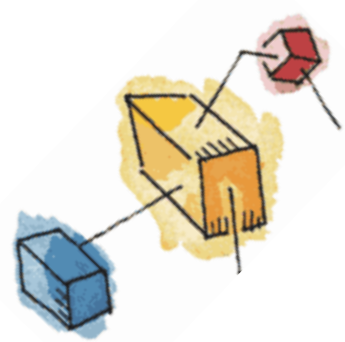
Sharing

- Advantageous to allow each process access to the same copy of the program rather than have their own separate copy
- Memory management must allow controlled access to shared areas of memory without compromising protection



Memory Management

- Memory management brings processes into main memory for execution by the processor
 - Contiguous memory partitioning
 - » OS divides memory up into relatively large partitions
 - » One process per partition
 - Paging and segmentation
 - Non-contiguous memory allocation
 - OS divides memory up into relatively small blocks
 - Process loaded in smaller pieces



Memory Partitioning

Technique

Description

Strengths

Weaknesses

Fixed Partitioning

Main memory is divided into a number of static partitions at system generation time. A process may be loaded into a partition of equal or greater size.

Simple to implement; little operating system overhead.

Inefficient use of memory due to internal fragmentation; maximum number of active processes is fixed.

Dynamic Partitioning

Partitions are created dynamically, so that each process is loaded into a partition of exactly the same size as that process.

No internal fragmentation; more efficient use of main memory.

Inefficient use of processor due to the need for compaction to counter external fragmentation.

Simple Paging

Main memory is divided into a number of equal-size frames. Each process is divided into a number of equal-size pages of the same length as frames. A process is loaded by loading all of its pages into available, not necessarily contiguous, frames.

No external fragmentation.

A small amount of internal fragmentation.

Simple Segmentation

Each process is divided into a number of segments. A process is loaded by loading all of its segments into dynamic partitions that need not be contiguous.

No internal fragmentation; improved memory utilization and reduced overhead compared to dynamic partitioning.

External fragmentation.

Virtual Memory Paging

As with simple paging, except that it is not necessary to load all of the pages of a process. Nonresident pages that are needed are brought in later automatically.

No external fragmentation; higher degree of multiprogramming; large virtual address space.

Overhead of complex memory management.

Virtual Memory Segmentation

As with simple segmentation, except that it is not necessary to load all of the segments of a process. Nonresident segments that are needed are brought in later automatically.

No internal fragmentation, higher degree of multiprogramming; large virtual address space; protection and sharing support.

Overhead of complex memory management.

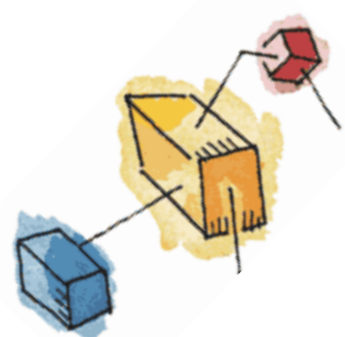


Fixed Partitioning

- Equal-size partitions
 - any process whose size is less than or equal to the partition size can be loaded into an available partition

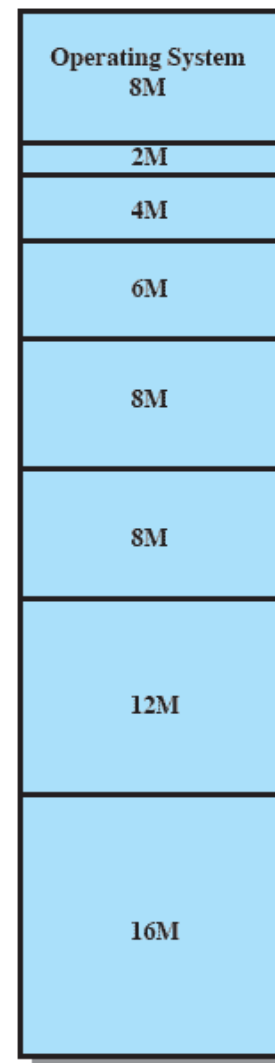


(a) Equal-size partitions

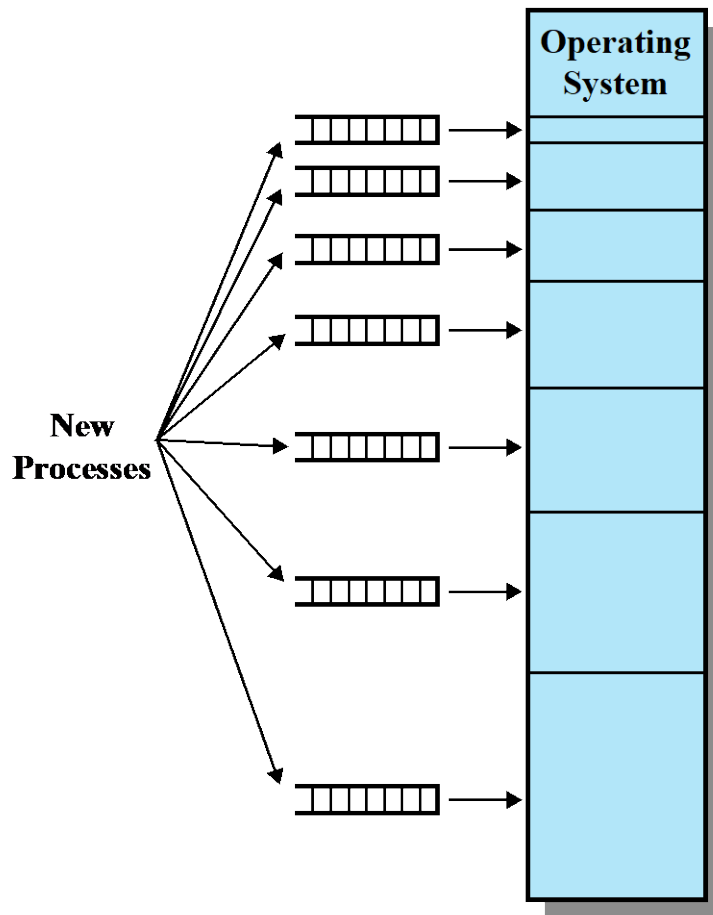


Unequal Size Partitions

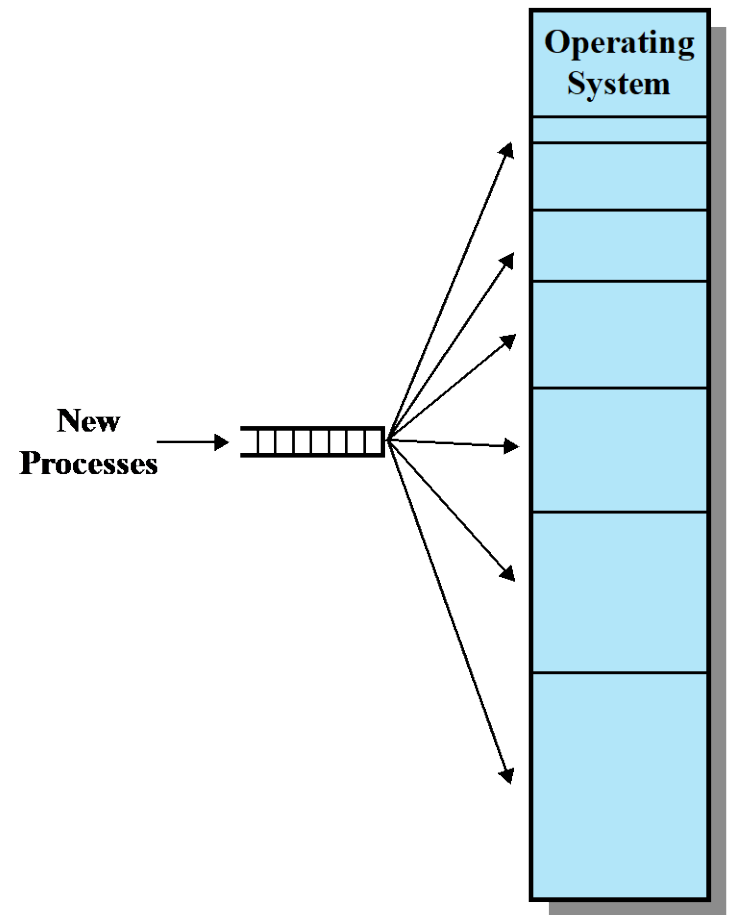
- Using unequal size partitions helps lessen the problems
 - programs up to 16M can be accommodated without overlays
 - partitions smaller than 8M allow smaller programs to be accommodated with less internal fragmentation



(b) Unequal-size partitions



(a) One process queue per partition

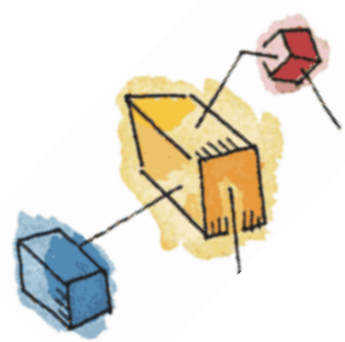


(b) Single queue

Figure 7.3 Memory Assignment for Fixed Partitioning

Disadvantages

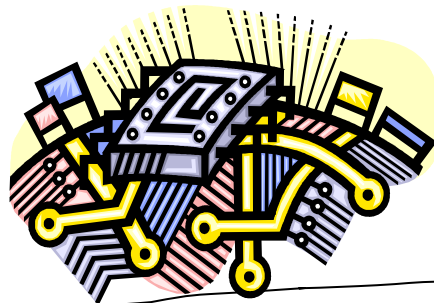
- The number of partitions specified at system generation time limits the number of active processes in the system
- A program may be too big to fit in a partition
- Main memory utilization is inefficient
 - any program, regardless of size, occupies an entire partition
 - ***internal fragmentation***
 - wasted space due to the block of data loaded being smaller than the partition



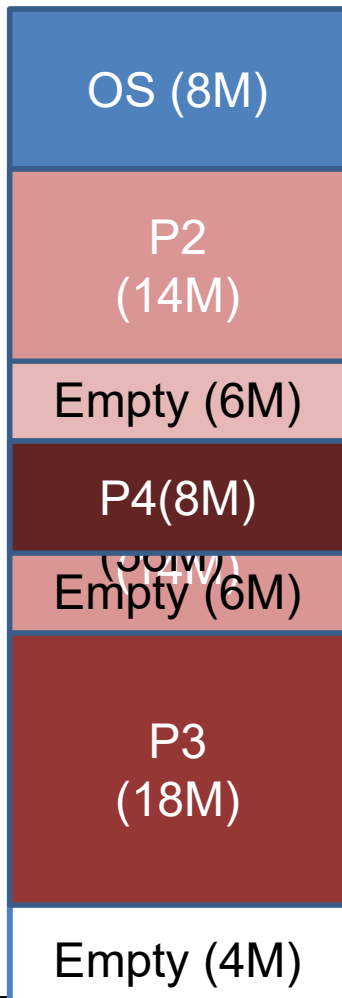
An illustration in the top-left corner showing a large yellow cube with a blue cube and a red cube attached to its sides, representing memory partitions.

Dynamic Partitioning

- Partitions are of variable length and number
- Process is allocated exactly as much memory as it requires
- This technique was used by IBM's mainframe operating system, OS/MVT



Effect of Dynamic Partitioning



- **External Fragmentation**
 - Memory external to processes is fragmented
 - Memory becomes more and more fragmented & utilization declines
- **Compaction**
 - Technique for overcoming external fragmentation
 - OS moves processes so that they are contiguous
 - Free memory is together in one block
 - Time consuming and wastes CPU time



Placement Algorithms

Best-fit

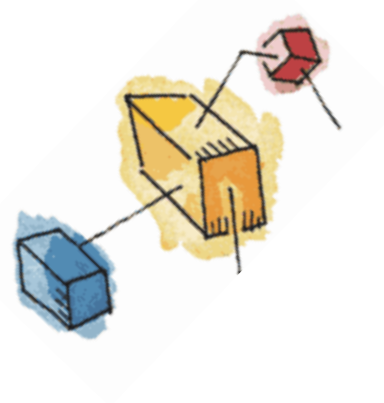
- chooses the block that is closest in size to the request

First-fit

- begins to scan memory from the beginning and chooses the first available block that is large enough

Next-fit

- begins to scan memory from the location of the last placement and chooses the next available block that is large enough



Allocation

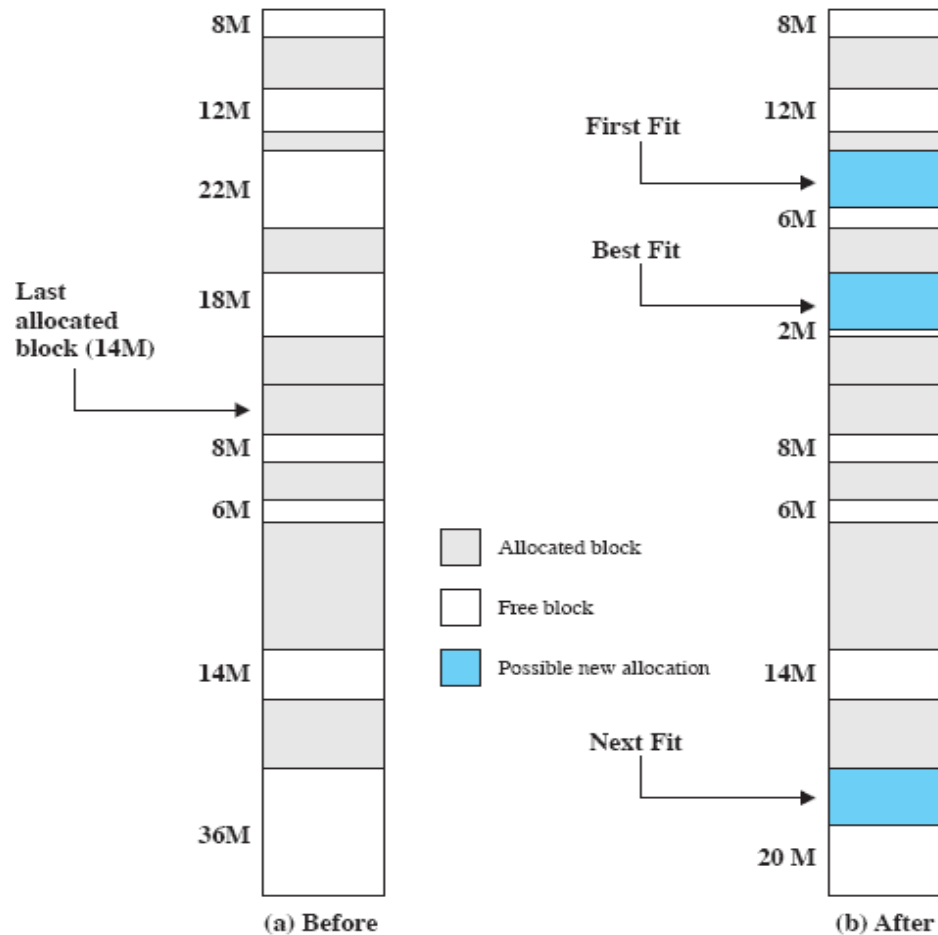
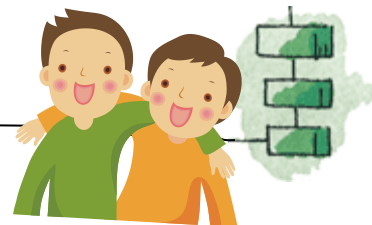
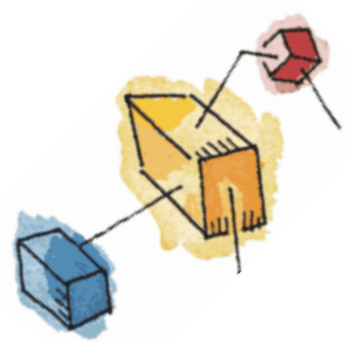
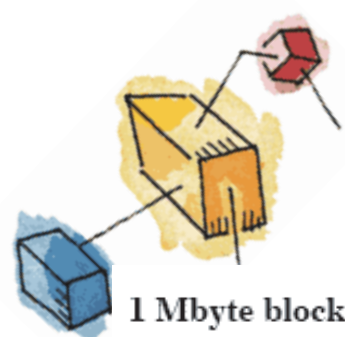


Figure 7.5 Example Memory Configuration before and after Allocation of 16-Mbyte Block

Buddy System

- Comprised of fixed and dynamic partitioning schemes
- Space available for allocation is treated as a single block
- Memory blocks are available of size 2^K words, $L \leq K \leq U$, where
 - 2^L = *smallest size block that is allocated*
 - 2^U = largest size block that is allocated; generally 2^U is the size of the entire memory available for allocation





Buddy System Example

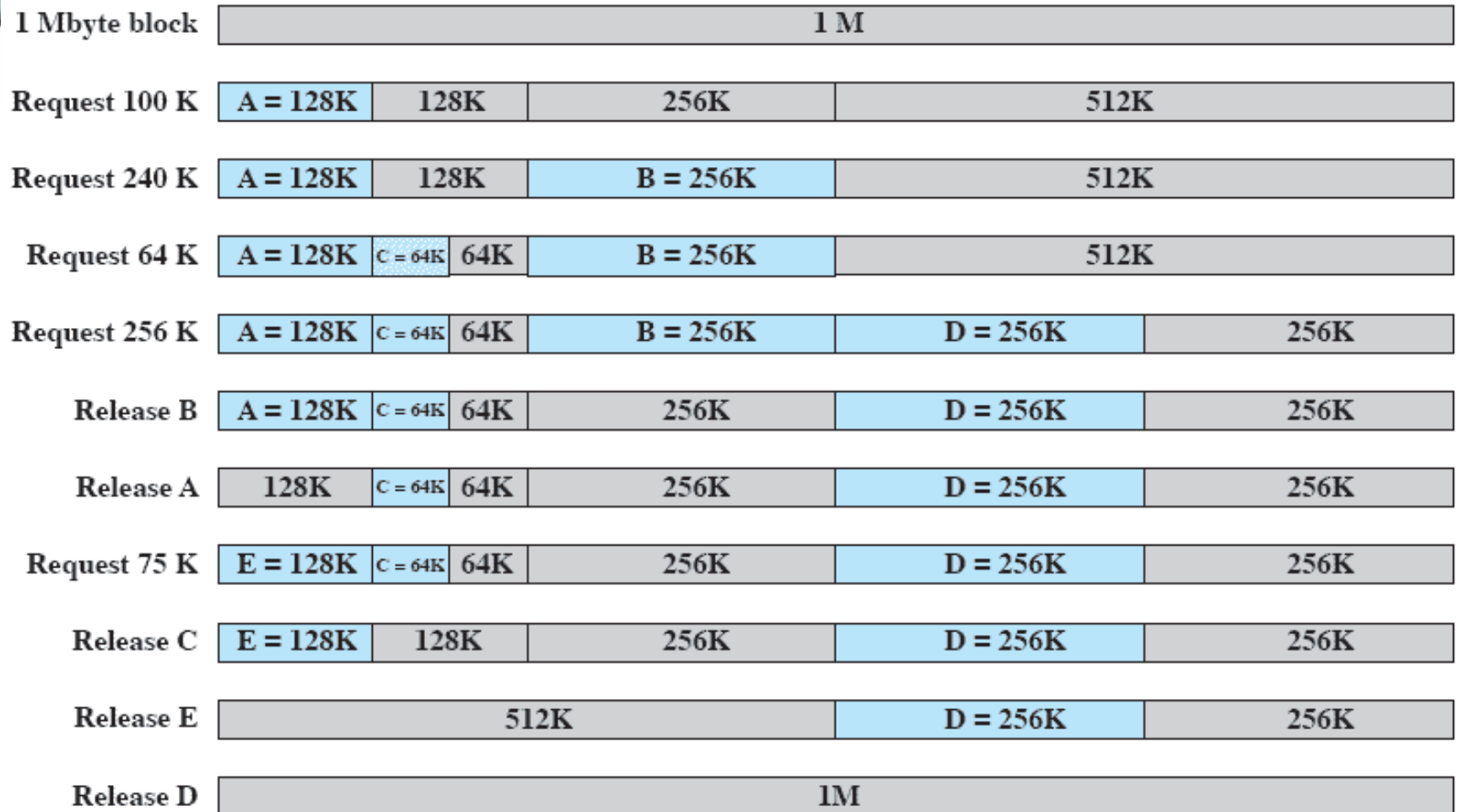


Figure 7.6 Example of Buddy System

Tree Representation

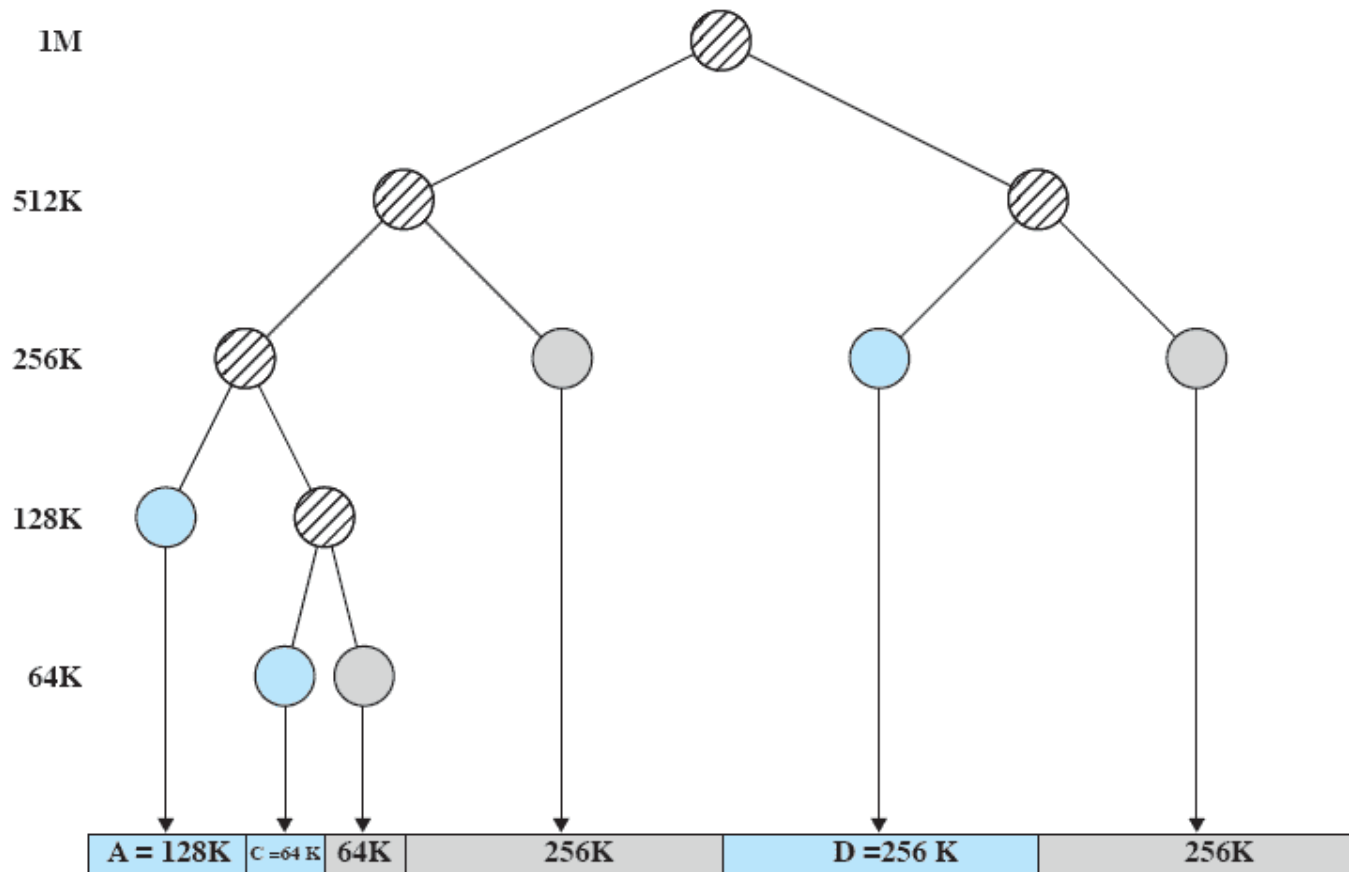


Figure 7.7 Tree Representation of Buddy System

Paging

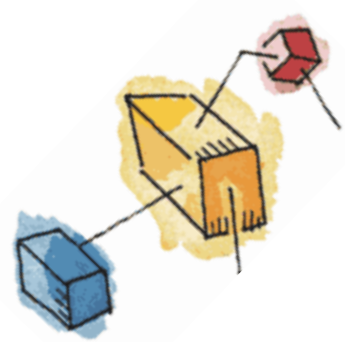
- Partition memory into equal fixed-size chunks that are relatively small
- Process is also divided into small fixed-size chunks of the same size

Pages

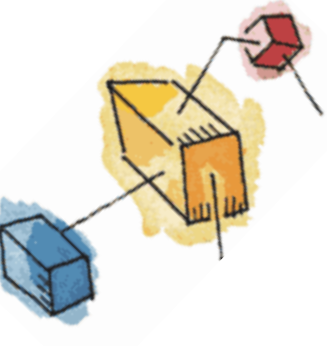
- chunks of a process

Frames


- chunks of memory



Processes and Frames

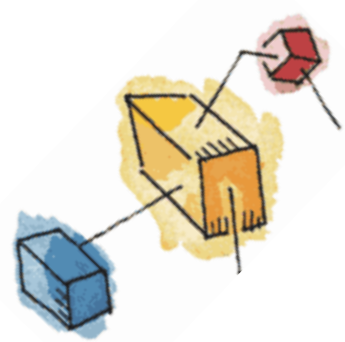


Frame number	Main memory
0	A.0
1	A.1
2	A.2
3	A.3
4	D.0
5	D.1
6	D.2
7	C.0
8	C.1
9	C.2
10	C.3
11	D.3
12	D.4
13	
14	

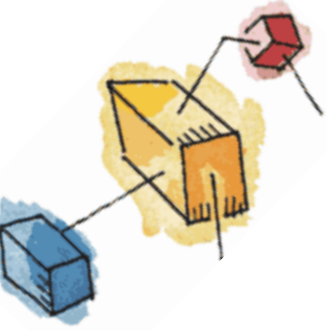


Page Table

- Maintained by operating system for each process
- Contains the frame location for each page in the process
- Used by processor to produce a physical address



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

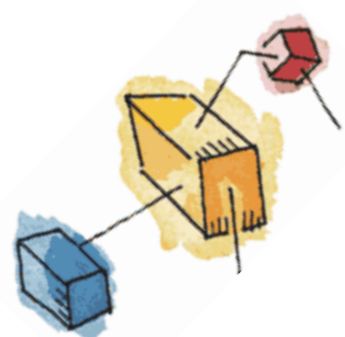
Process D
page table

13
14

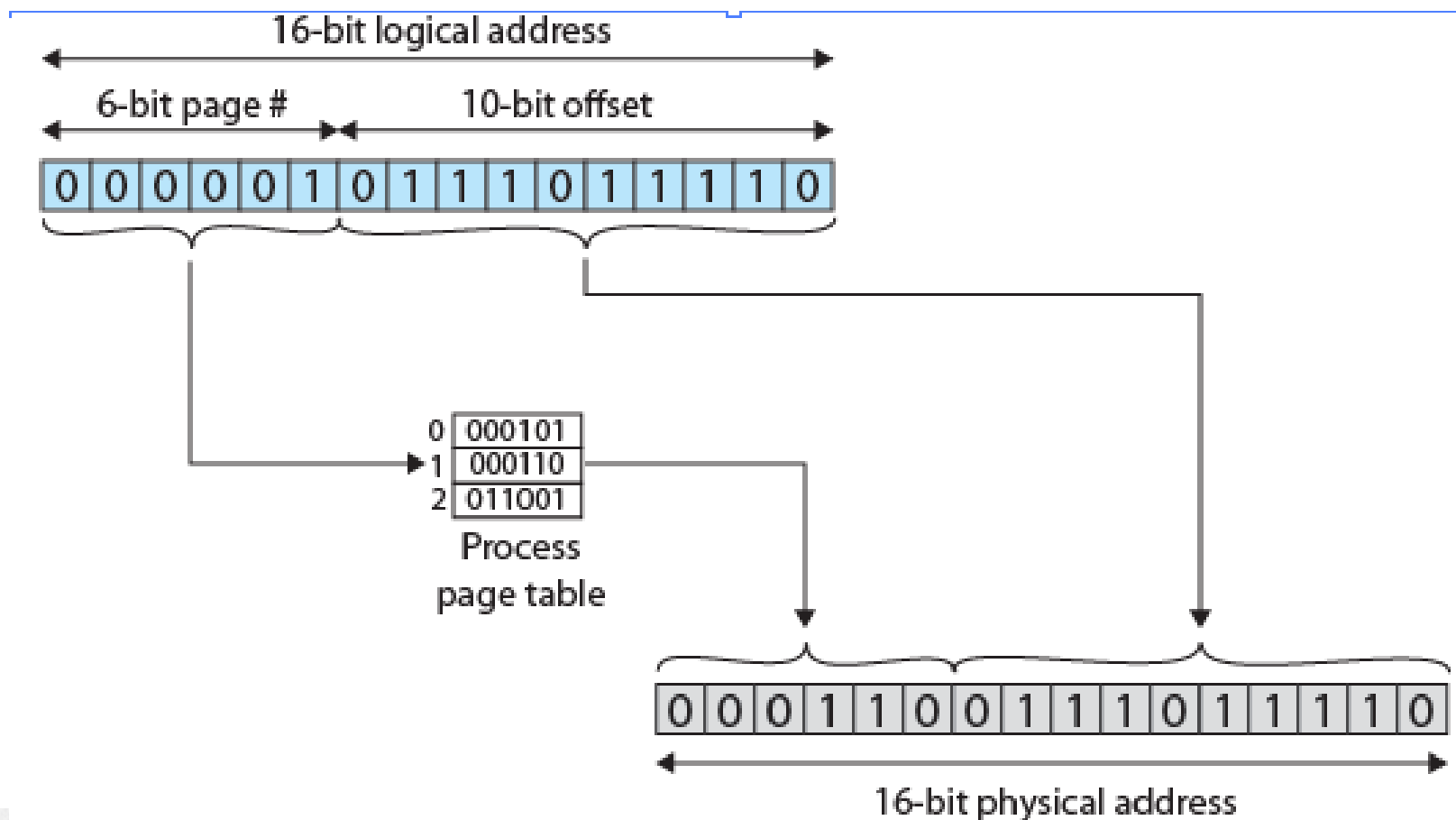
Free frame
list

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





Logical-to-Physical Address Translation - Paging



(a) Paging



Logical Addresses

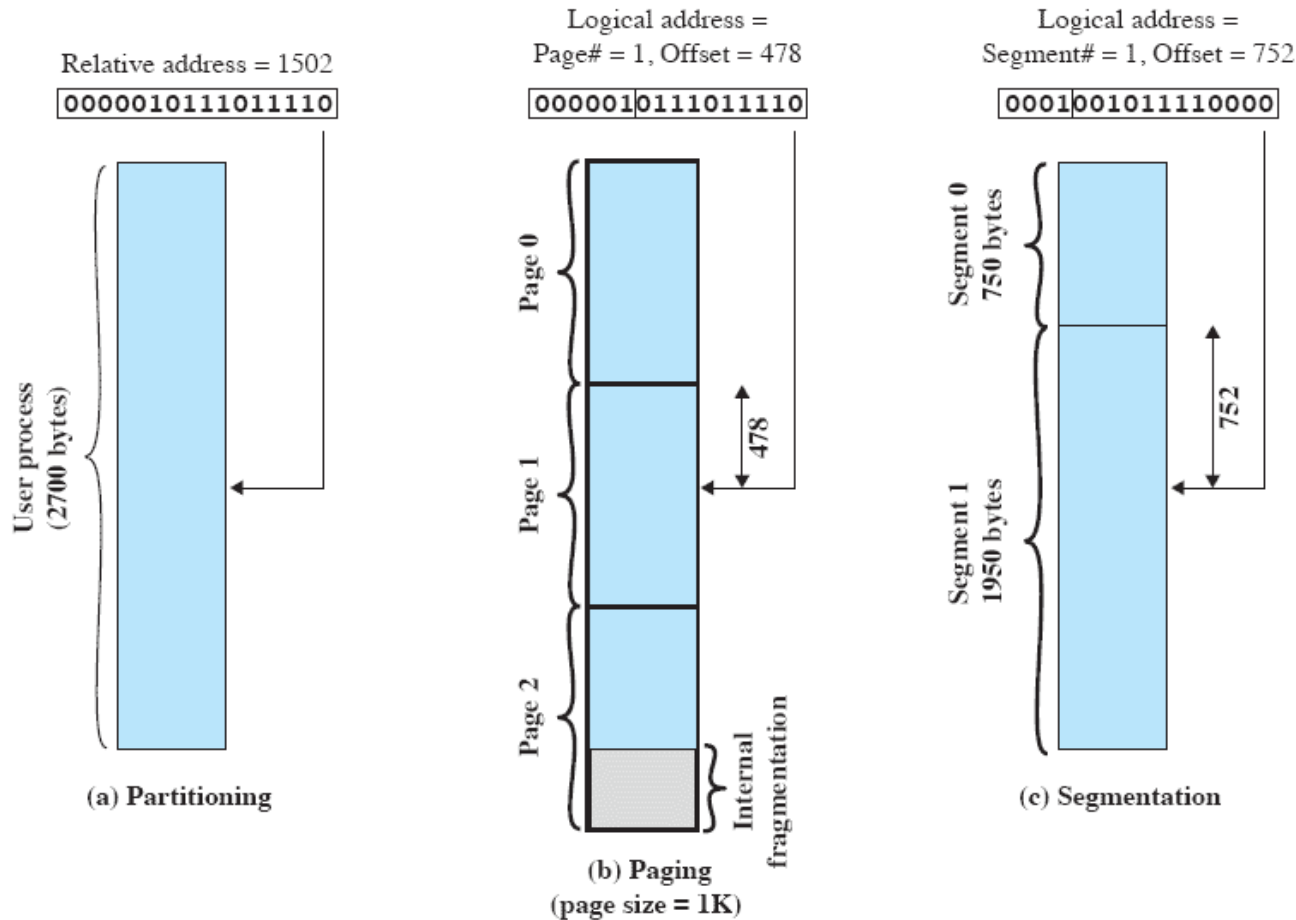
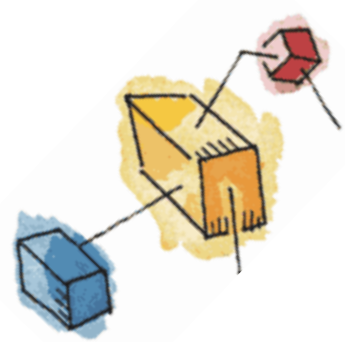


Figure 7.11 Logical Addresses

Segmentation

- A program can be subdivided into segments
 - may vary in length
 - there is a maximum length
- Addressing consists of two parts:
 - segment number
 - an offset
- Similar to dynamic partitioning
 - Eliminates internal fragmentation
 - But have issues of external fragmentation



Segmentation

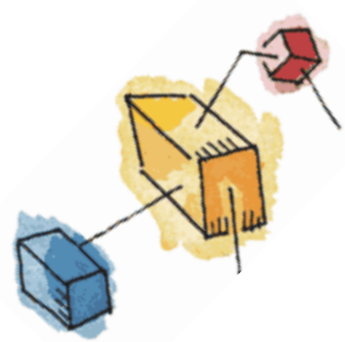
- Visible, and provided as a convenience for organizing programs and data
- However, there is no simple relationship between logical addresses and physical addresses
- The following steps are needed for address translation:

Extract the segment number as the leftmost n bits of the logical address

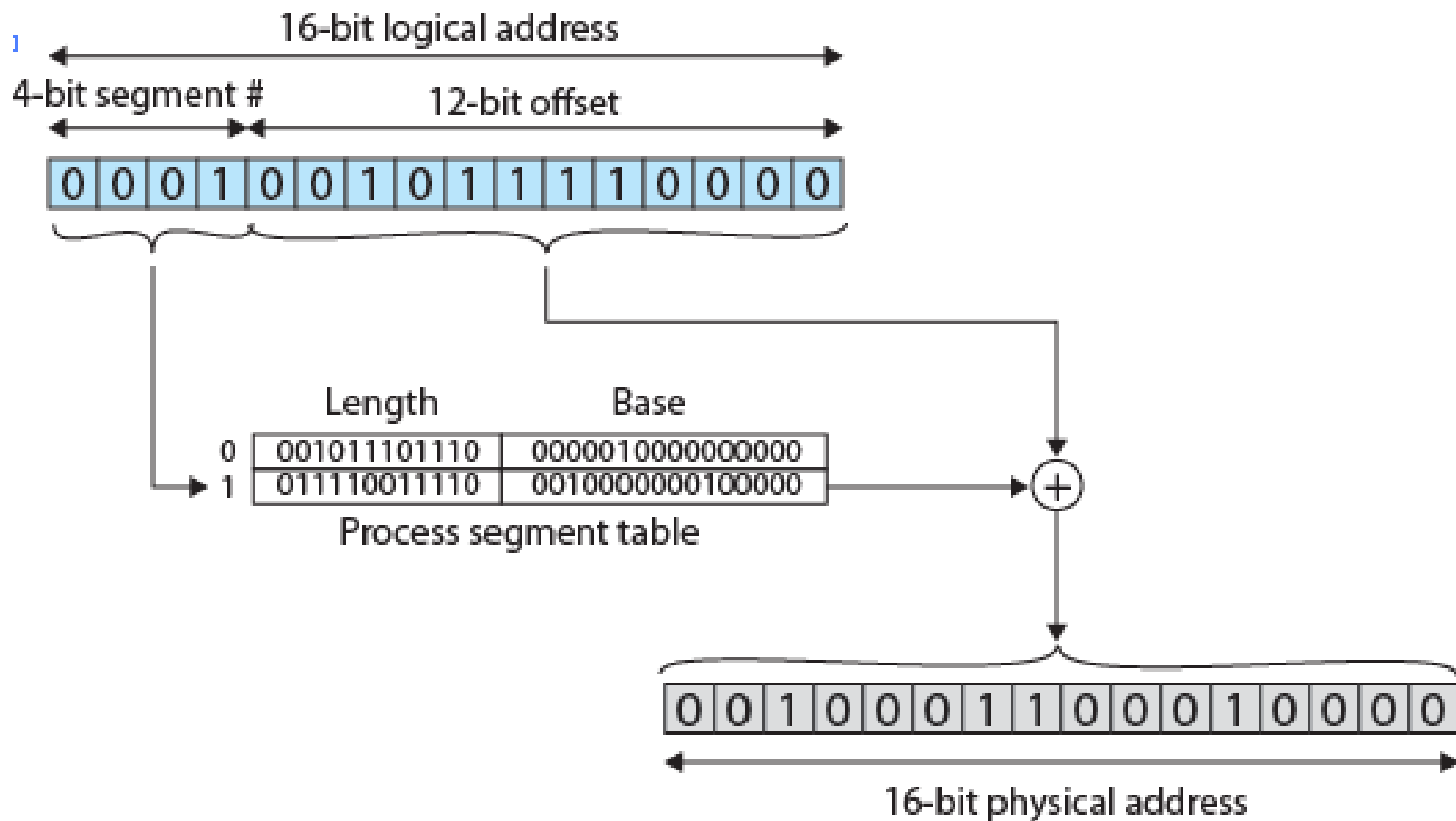
Use the segment number as an index into the process segment table to find the starting physical address of the segment

Compare the offset, expressed in the rightmost m bits, to the length of the segment. If the offset is greater than or equal to the length, the address is invalid

The desired physical address is the sum of the starting physical address of the segment plus the offset



Logical-to-Physical Address Translation - Segmentation



Logical Addresses

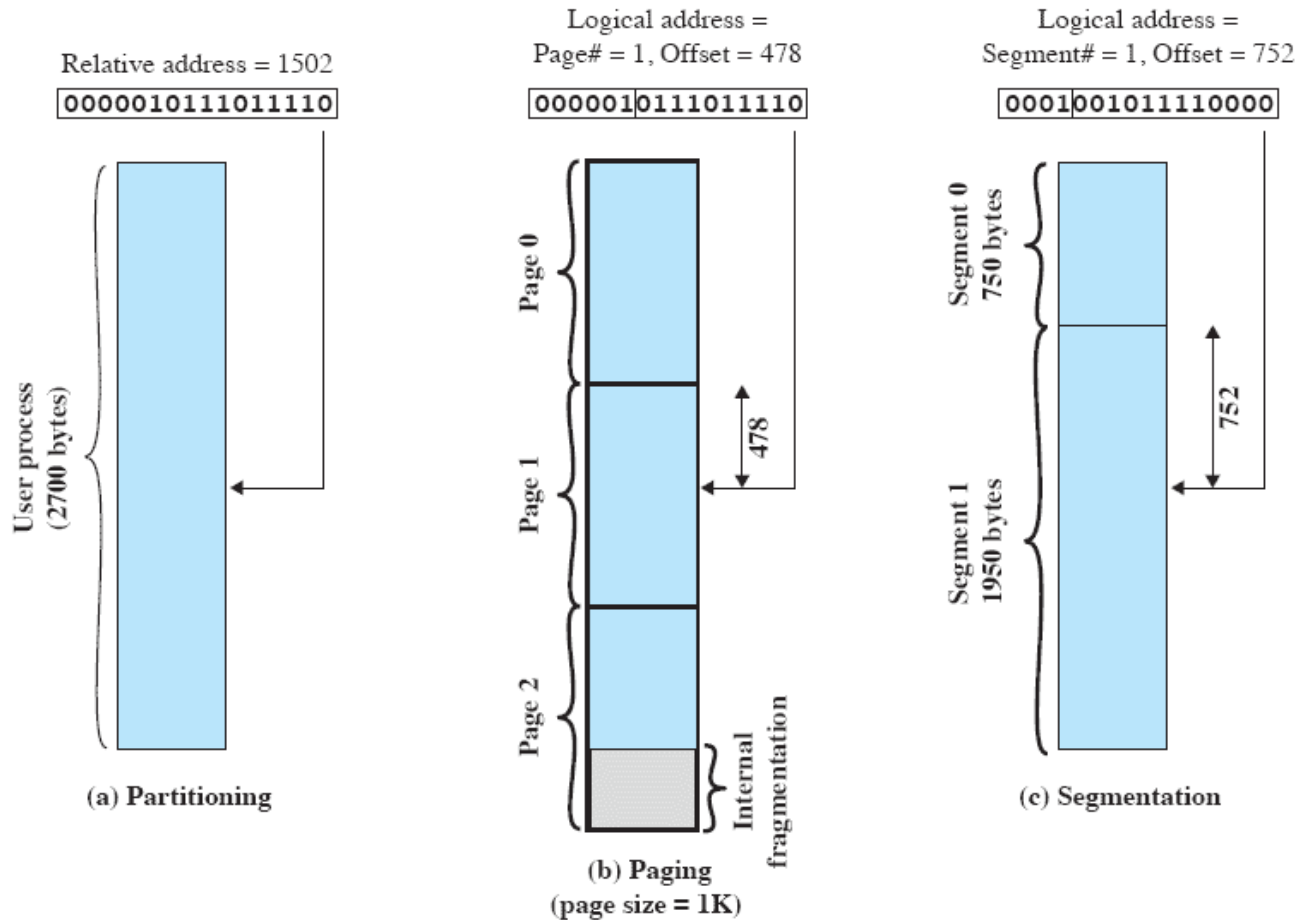
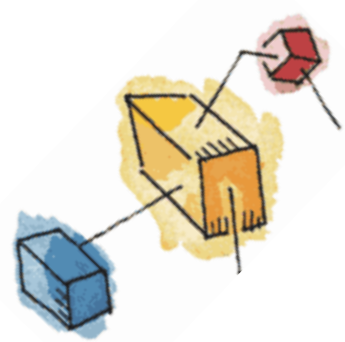


Figure 7.11 Logical Addresses

Summary

- Memory management requirements
 - Relocation
 - Logical/physical organization
 - Protection
 - Sharing
- Contiguous partitioning
 - Fixed partitioning
 - Dynamic partitioning
 - Buddy system
- Non-contiguous allocation
 - Paging
 - Segmentation

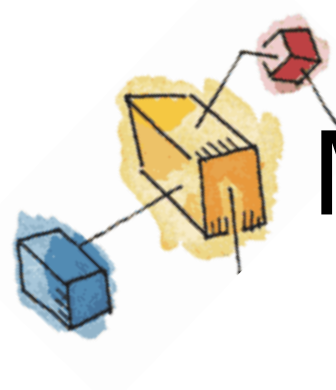


Summary

- Memory Management

- one of the most important and complex tasks of an operating system
- needs to be treated as a resource to be allocated to and shared among a number of active processes
- desirable to maintain as many processes in main memory as possible
- desirable to free programmers from size restriction in program development
 - basic tools are paging and segmentation (possible to combine)





Memory Management Terms

Frame	A fixed-length block of main memory.
Page	A fixed-length block of data that resides in secondary memory (such as disk). A page of data may temporarily be copied into a frame of main memory.
Segment	A variable-length block of data that resides in secondary memory. An entire segment may temporarily be copied into an available region of main memory (segmentation) or the segment may be divided into pages which can be individually copied into main memory (combined segmentation and paging).

