Operating Systems: Internals and Design Principles

Chapter 7 Memory Management

Eighth Edition

William Stallings

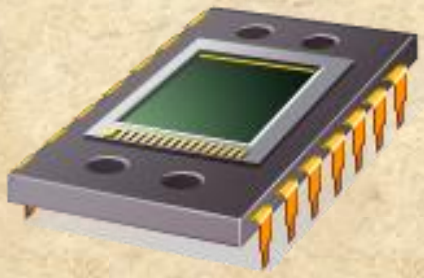
Frame	**A fixed-length block of main memory.**
Page	**A fixed-length block of data that resides in secondary memory (such as disk). A page of data may temporarily be copied into a frame of main memory.**
Segment	**A variable-length block of data that resides in secondary memory. An entire segment may 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).**

Table 7.1

Memory Management Terms

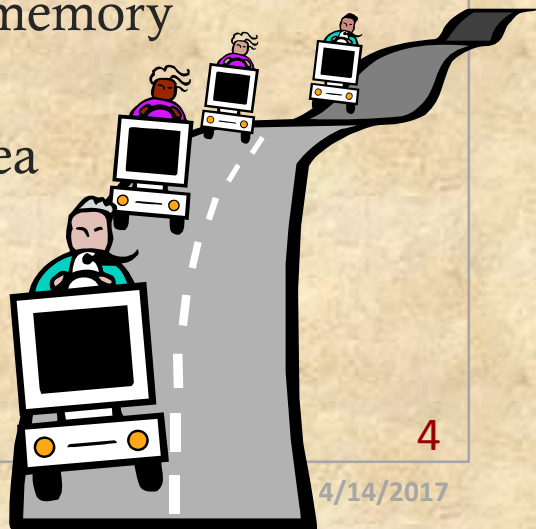
Memory Management Requirements

- Memory management is intended to satisfy the following requirements:
  - Relocation
  - Protection
  - Sharing
  - Logical organization
  - Physical organization



Relocation

- Programmers typically do not know in advance which other programs will be resident in main memory at the time of execution of their programs
- Active processes need to be able to be swapped in and out of main memory in order to maximize processor utilization
- Specifying that a process must be placed in the same memory region when it is swapped back in would be limiting
  - may need to *relocate* the process to a different area of memory



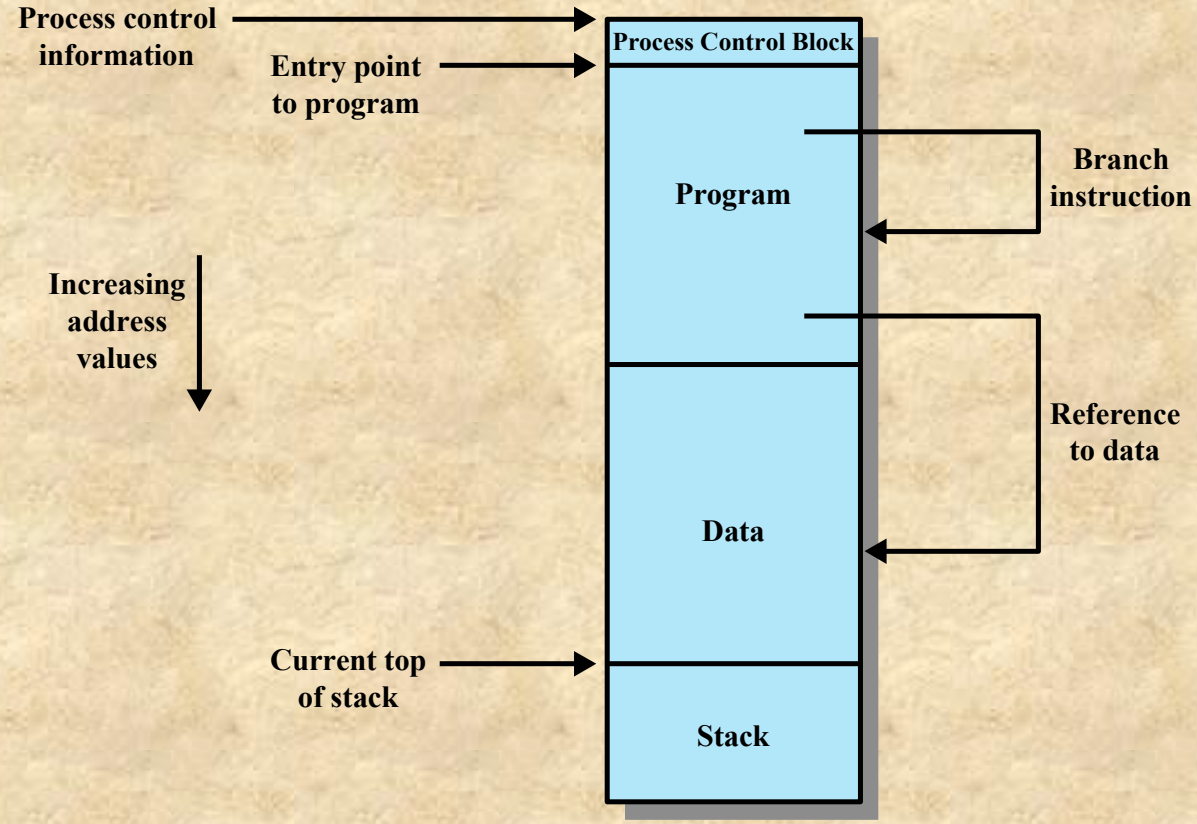


Figure 7.1 Addressing Requirements for a Process

Protection

- Each process should be protected against unwanted interference by other processes
- Processes need to acquire permission to reference memory locations in other process for reading or writing purposes
- Since the location of a program in main memory is unpredictable, it is impossible to check absolute address at compile time to assure protection
- All memory references generated by a process must be checked at run time to ensure that they refer only to the memory space allocated to that process
- Mechanisms that support relocation also support protection
- The memory protection requirement must be satisfied by the hardware

Sharing

- If a number of processes are executing the same program, it is advantageous to allow each process access to the same copy of the program rather than have their own separate copy
- The memory management must allow controlled access to shared areas of memory without compromising protection
- The mechanisms used to support relocation support sharing capabilities

Logical Organization

- Main memory is organized as a linear address space, which does not correspond to the way in which programs are typically constructed

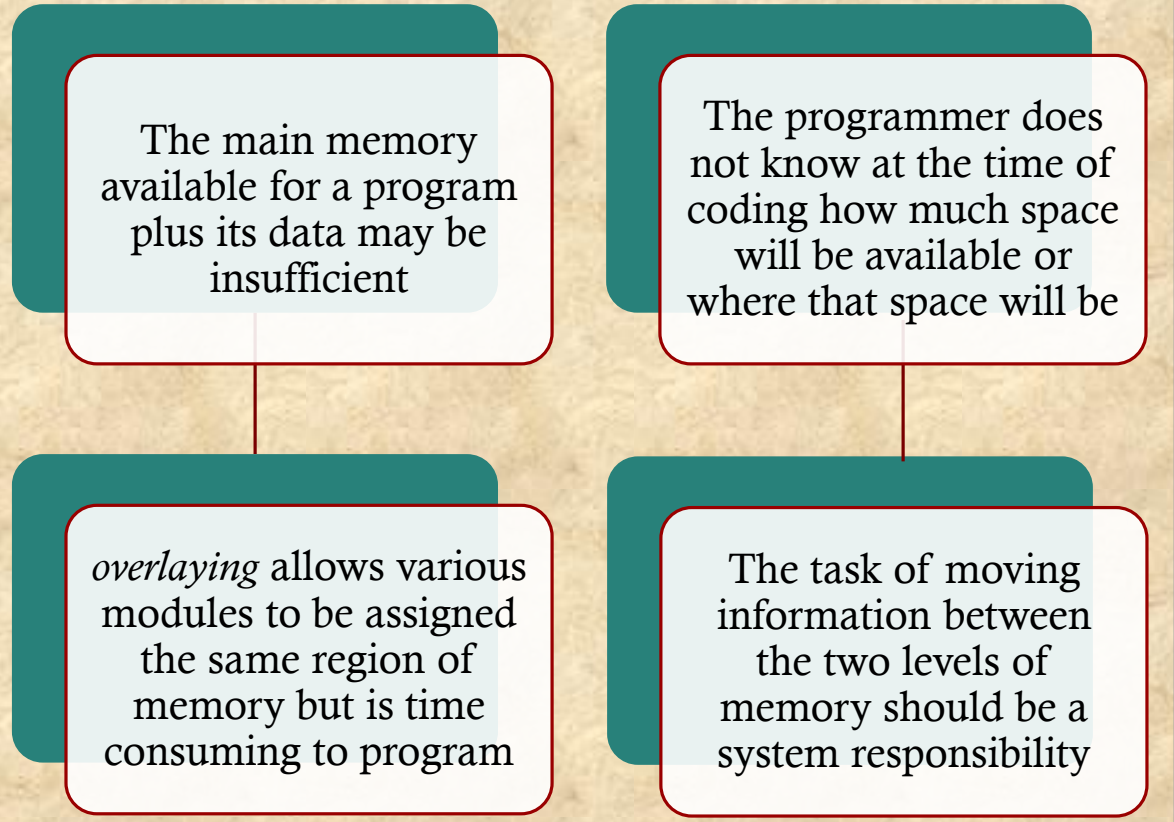
Programs are typically organized in modules

- modules can be written and compiled independently, with all references from one module to another resolved by the system at run time
- different degrees of protection can be given to different modules (read-only, execute-only)
- modules can be shared among processes

- Segmentation is the tool that most readily satisfies these requirements

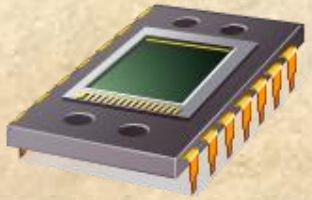
Physical Organization

- Computer memory is organized into at least two levels: main memory and secondary memory
- Secondary memory can be provided for long-term storage of programs and data, while a smaller main memory holds programs and data currently in use



Memory Partitioning

- The principal operation of memory management brings processes into main memory for execution by the processor
  - involves a scheme known as virtual memory
  - based on basic techniques: segmentation and paging
- Partitioning
  - a simple technique that does not involve virtual memory
  - used in several variations in some now-obsolete operating systems
- Simple paging and simple segmentation

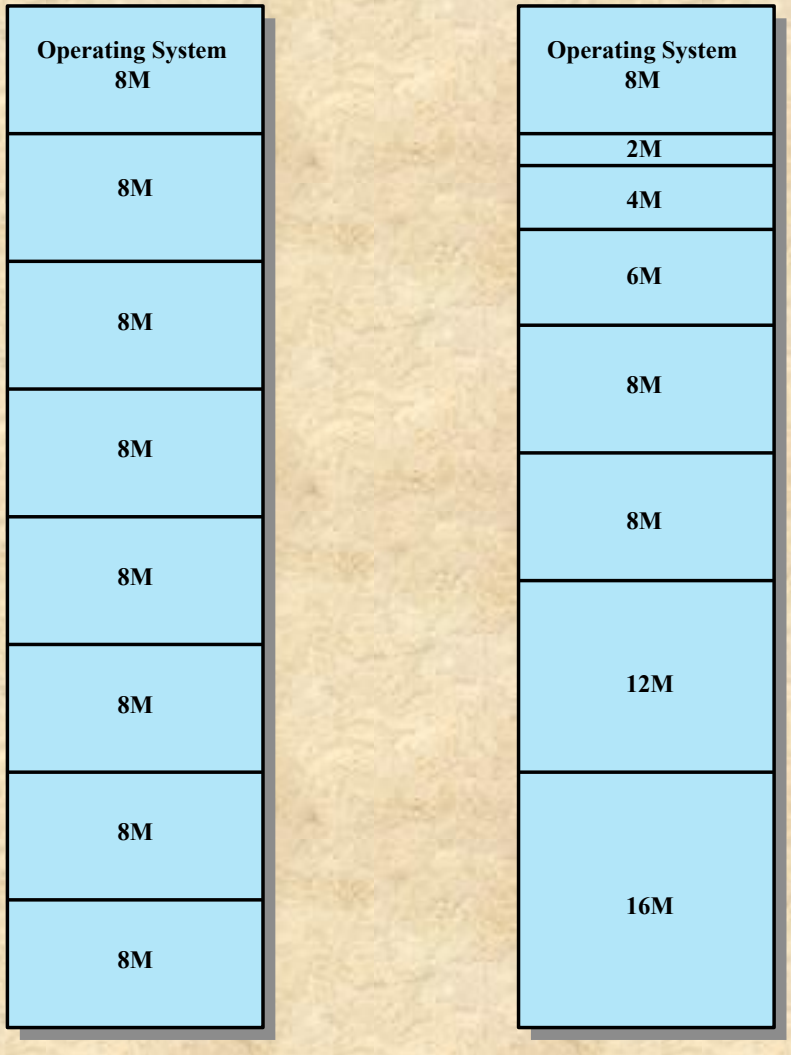


Technique	Description	Strengths	Weaknesses
Fixed Partitioning	Main memory is divided into a number of static partitions at system generation time. A process may be loaded into a partition of equal or greater size.	Simple to implement; little operating system overhead.	Inefficient use of memory due to 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.

Table 7.2

Memory Management Techniques

(Table is on page 345 in textbook)



(a) Equal-size partitions

(b) Unequal-size partitions

Figure 7.2 Example of Fixed Partitioning of a 64-Mbyte Memory

Disadvantages of Equal-sized Fixed Partitions

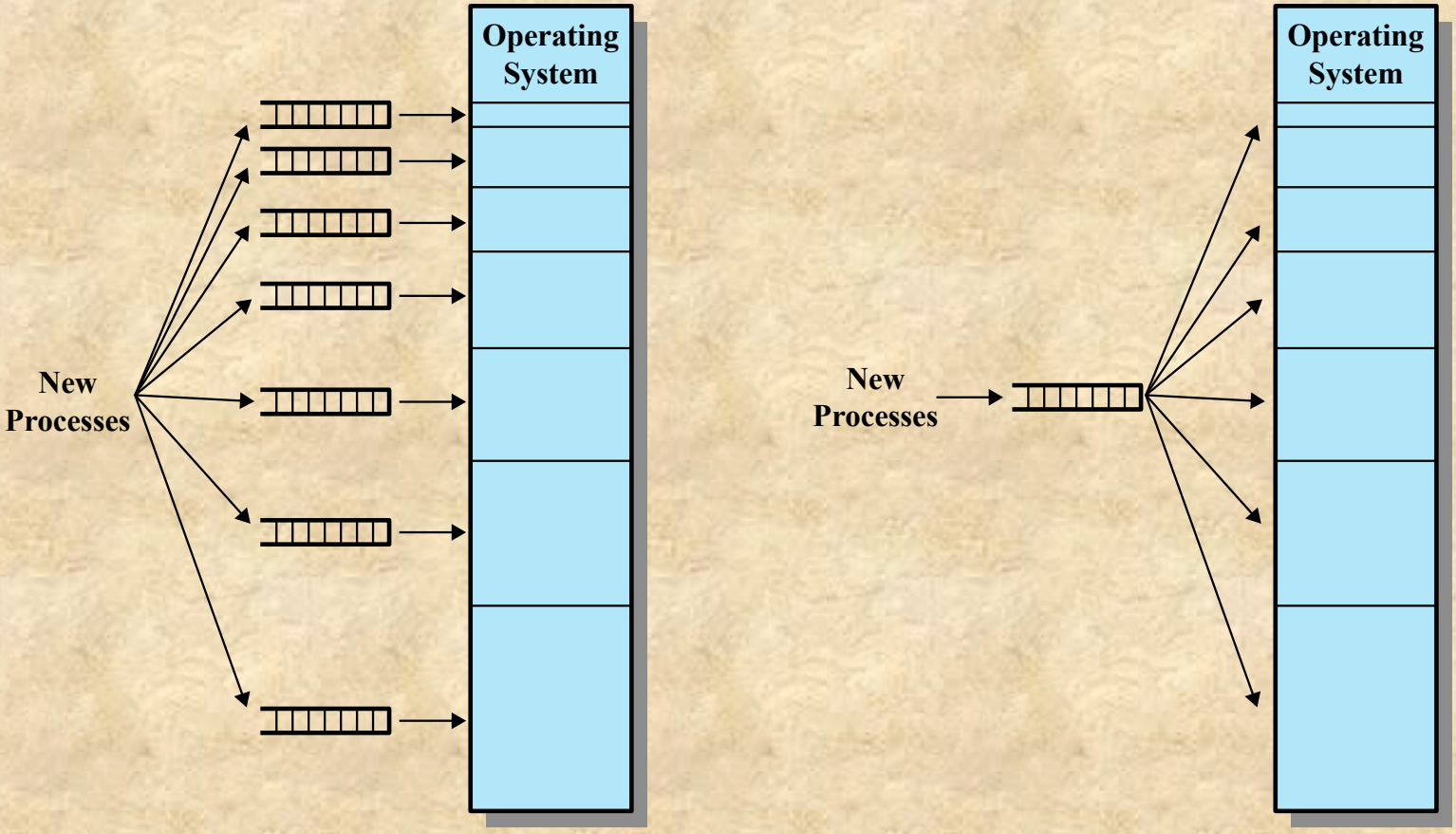
- A program may be too big to fit in a partition
  - program needs to be designed with the use of overlays
  - when a module is needed that is not present, the user's program must load that module into the program's partition, overlaying whatever programs or data are there
- Main memory utilization is 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

Placement Policies of Partitions

- Equal-size partitions
  - As long as there is any available partition, a process can be loaded into that partition
  - If all partitions are occupied with processes that are not ready to run, then one of these processes must be swapped out to make room for a new process
  - Relatively simple and require minimal OS software and processing overhead

Placement Policies of Partitions

- Unequal-size partitions: Assign each process to the smallest partition within which it will fit
  - A scheduling queue for each partition to hold swapped-out processes destined for that partition (Figure 7.3a)
    - Minimize wasted memory within a partition (internal fragmentation)
    - Not optimum from the point of view of the system as a whole
  - A single queue for all processes (Figure 7.3b)
    - If all partitions are occupied
      - Swapping out of the smallest partition that will hold the incoming process
      - May consider other factors, such as priority, and a preference for swapping out blocked processes versus ready processes



(a) One process queue per partition

(b) Single queue

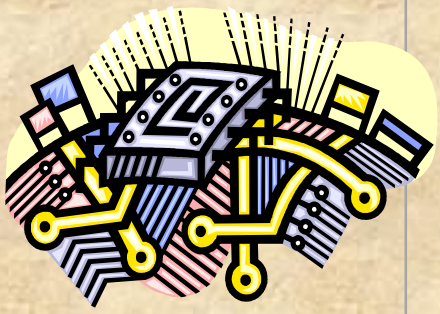
Figure 7.3 Memory Assignment for Fixed Partitioning

Disadvantages of Fixed Partitions

- The number of partitions specified at system generation time limits the number of active (not suspended) processes in the system
- Because partition sizes are preset at system generation time, small jobs will not utilize partition space efficiently

Dynamic Partitioning

- The partitions are of variable length and number
- A process is allocated exactly as much memory as it requires
- Used by IBM's mainframe operating system, OS/MVT(Multiprogramming with a Variable Number of Tasks)
- External fragmentation may occur



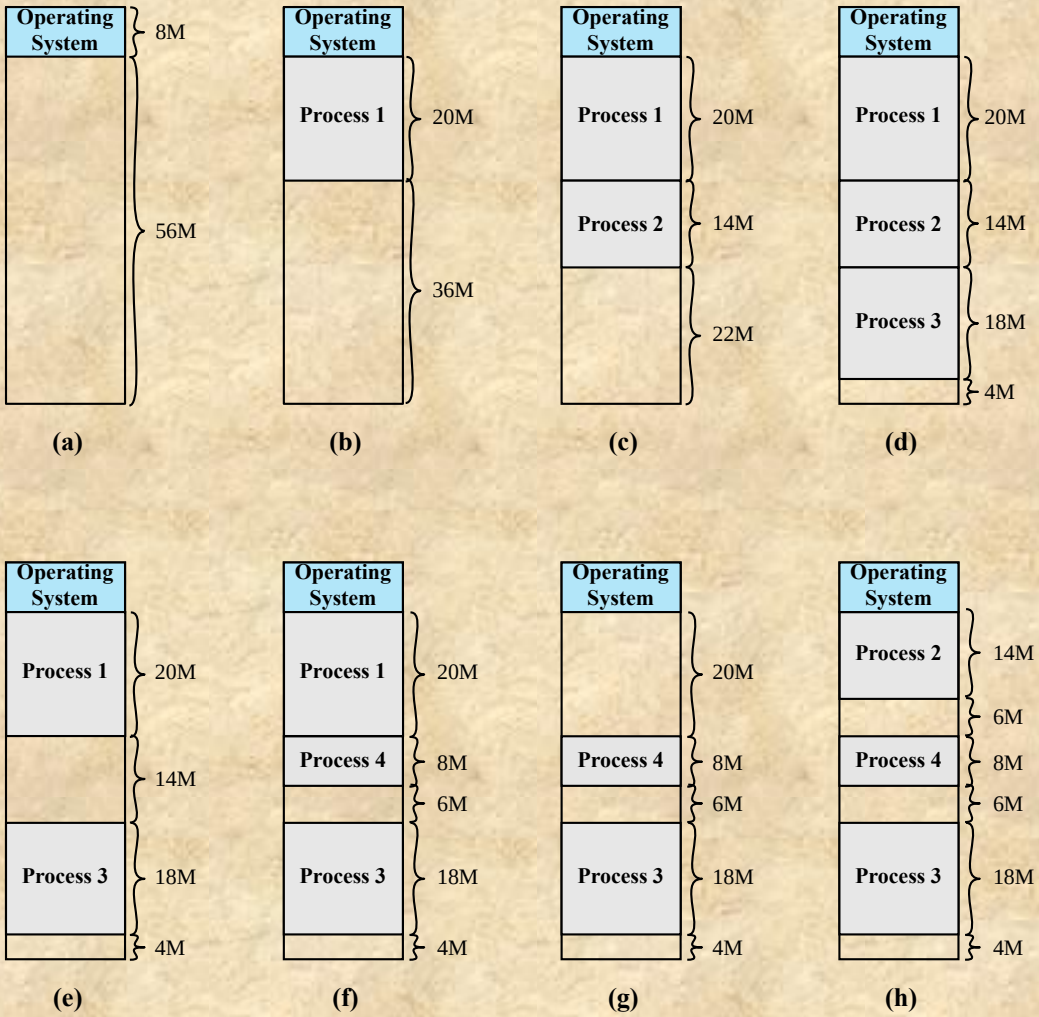


Figure 7.4 The Effect of Dynamic Partitioning

Dynamic Partitioning

External Fragmentation

- memory becomes more and more fragmented
- memory utilization declines

Compaction

- technique for overcoming external fragmentation
- OS shifts processes so that they are contiguous
- free memory is together in one block
- time consuming and wastes CPU time

Placement Algorithms

Best-fit	First-fit	Next-fit
• chooses the block that is closest in size to the request	• begins to scan memory from the beginning and chooses the first available block that is large enough	• begins to scan memory from the location of the last placement and chooses the next available block that is large enough

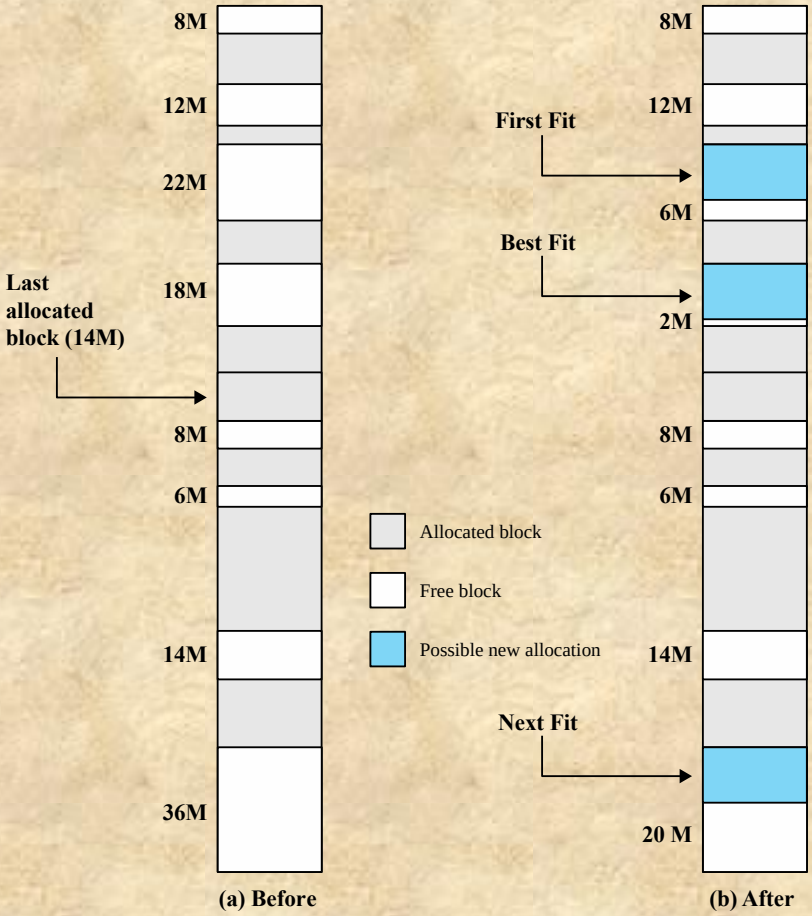


Figure 7.5 Example Memory Configuration before and after Allocation of 16-Mbyte Block

Buddy System

- A compromise of fixed and dynamic partitioning schemes
- 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
- At any time, the buddy system maintains a list of holes (unallocated blocks) of each size 2^i
  - A *hole* may be removed from the (*i + 1) list by splitting it in half to create two buddies of* size 2^i *in the i list*

Buddy Algorithm

- If a request of size s such that $2^{U-1} < s \leq 2^{U}$ is made, then the entire block is allocated
- Otherwise, the block is split into two equal buddies of size 2^{U-1}
  - *If $2^{U-2} < s \leq 2^{U-1}$, then the request is allocated to one of the two buddies*
  - *Otherwise, one* of the buddies is split in half again
  - This process continues until the smallest block greater than or equal to *s is generated and allocated to the request*
  - *Whenever a pair of buddies on the i list both become unallocated,* they are removed from that list and coalesced into a single block on the *(i + 1) list*

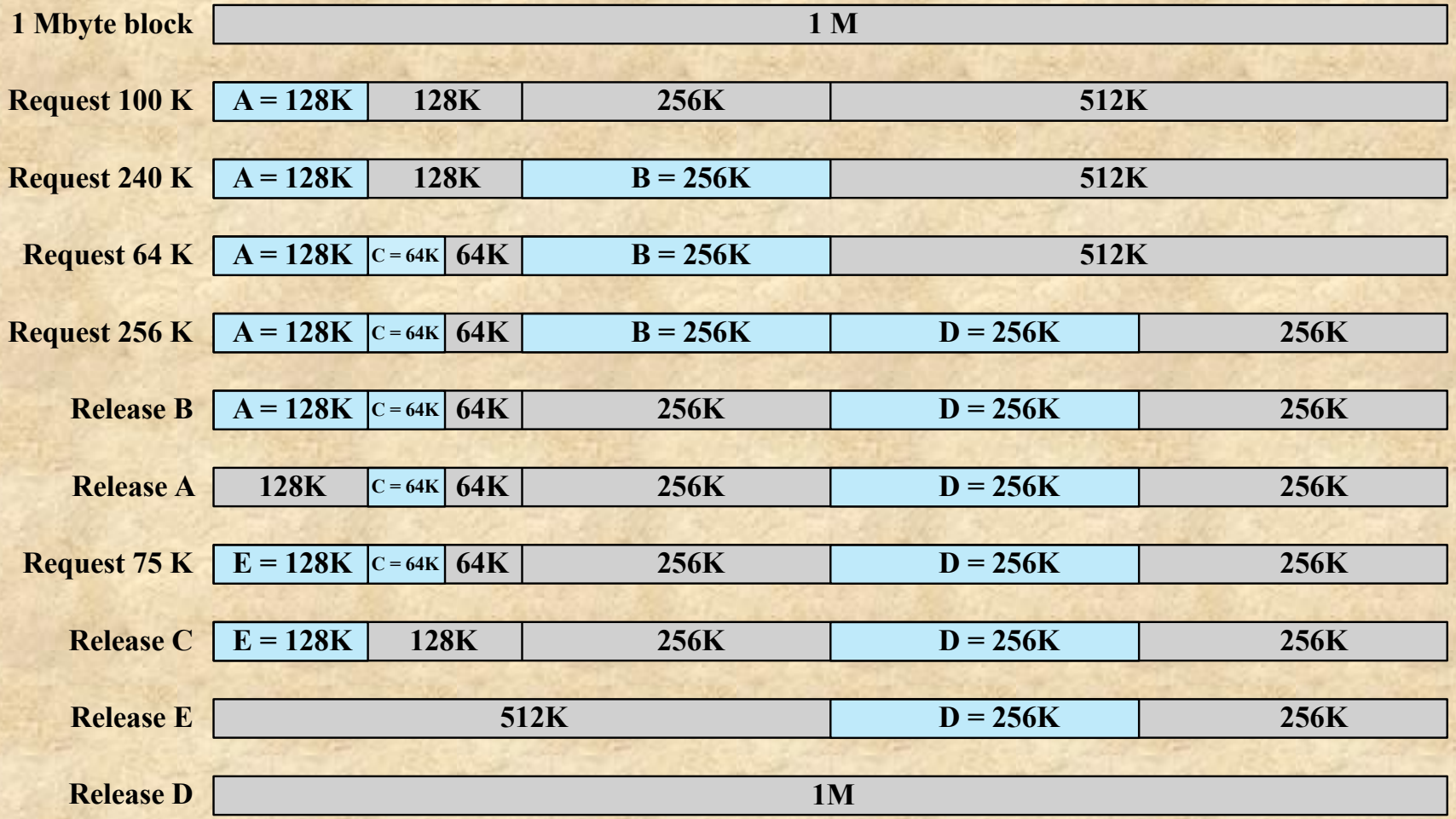


Figure 7.6 Example of Buddy System

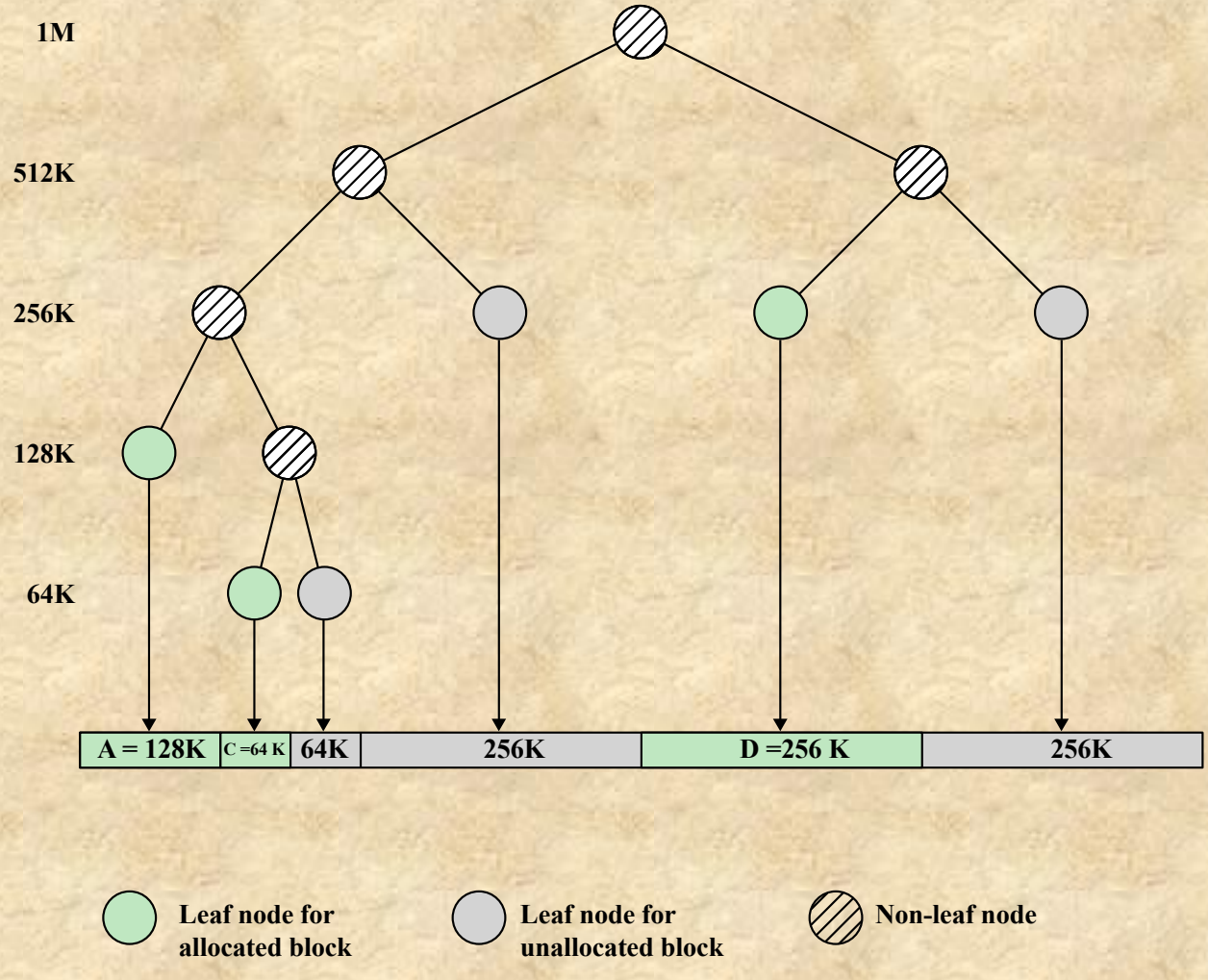


Figure 7.7 Tree Representation of Buddy System

Addresses

Logical

- A reference to a memory location independent of the current assignment of data to memory
- A translation must be made to a physical address before the memor y access can be achieved

Relative

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

Physical or Absolute

- An actual location in main memory

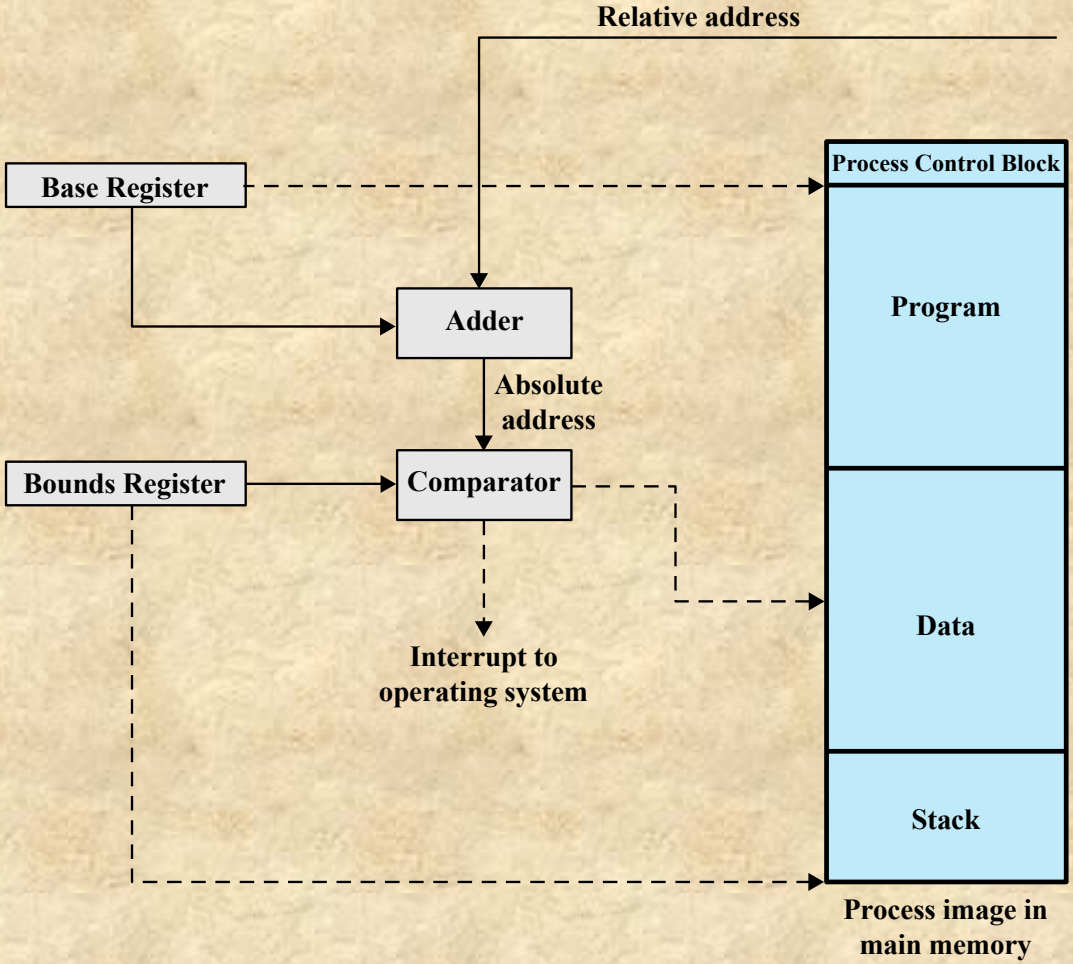


Figure 7.8 Hardware Support for Relocation

Paging

- Partition main memory into equal fixed-size chunks that are relatively small
- Each process is also divided into small fixed-size chunks of the same size
- Small internal fragmentation consisting of only a fraction of the last page of a process and no external fragmentation

Pages	[Page] Frames
• the chunks of a process	• available chunks of memory

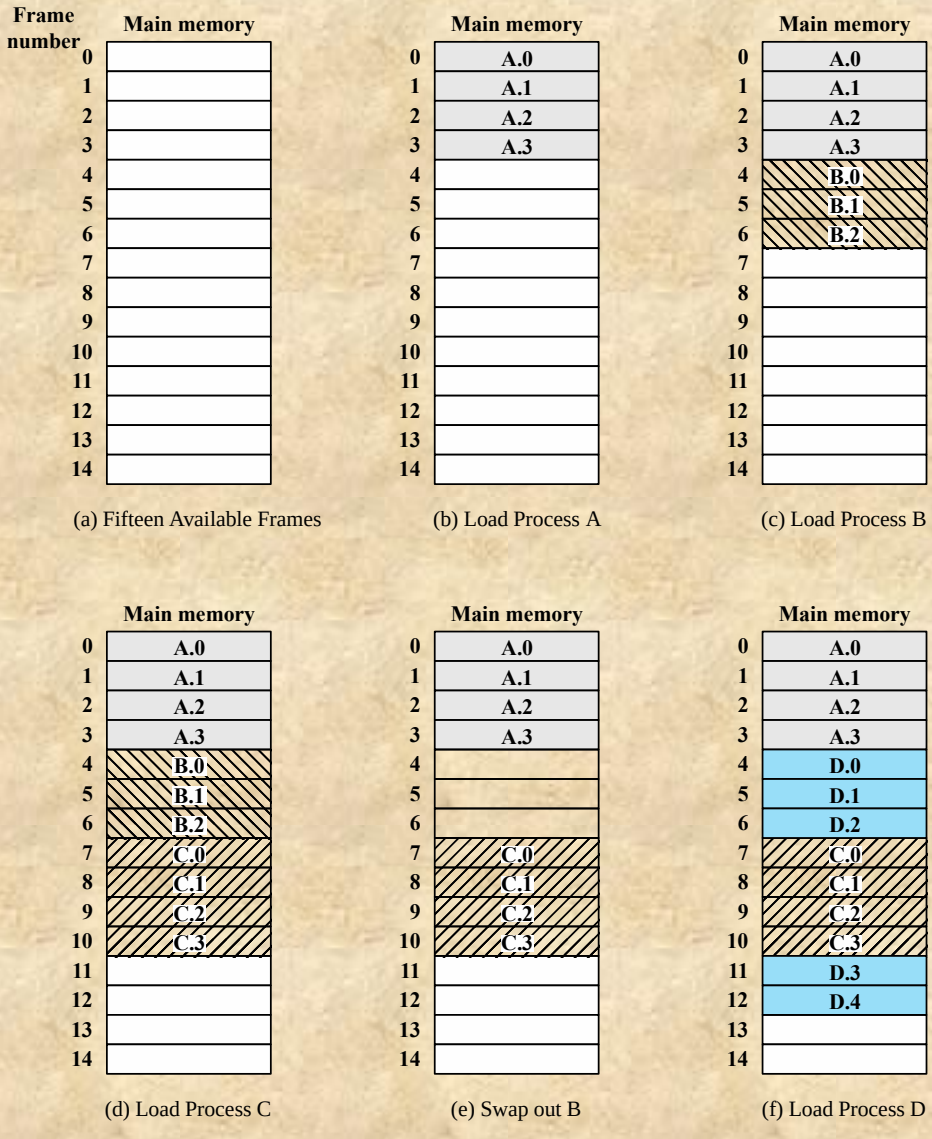


Figure 7.9 Assignment of Process Pages to Free Frames

Page Table

- Maintained by the operating system for each process
- Contains the frame location for each page in the process
- Each logical address consists of a page number and an offset within the page
- The processor uses the page table to produce a physical address: (frame number, offset)

- Indexed by the page number (starting at page 0)
- Each page table entry contains the frame number in main memory, if any
- Similar to fixed partitioning where the partitions are rather small and a program may occupy more than one partition which need not be contiguous

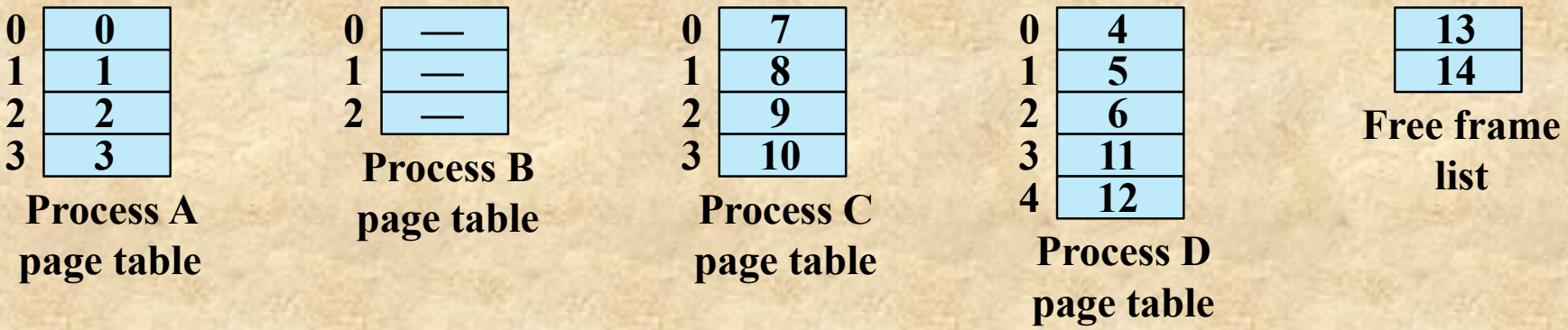


Figure 7.10 Data Structures for the Example of Figure 7.9 at Time Epoch (f)

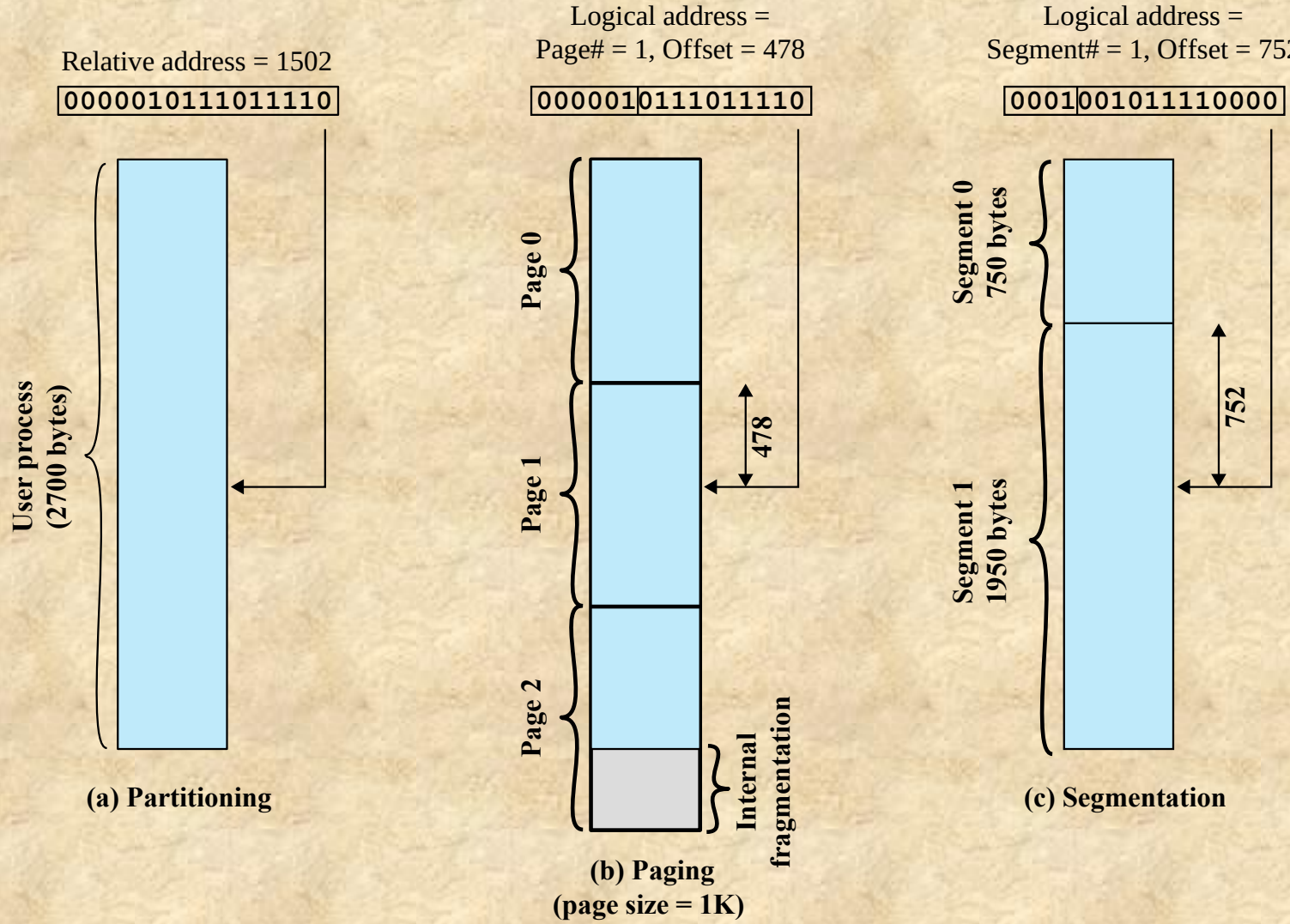


Figure 7.11 Logical Addresses

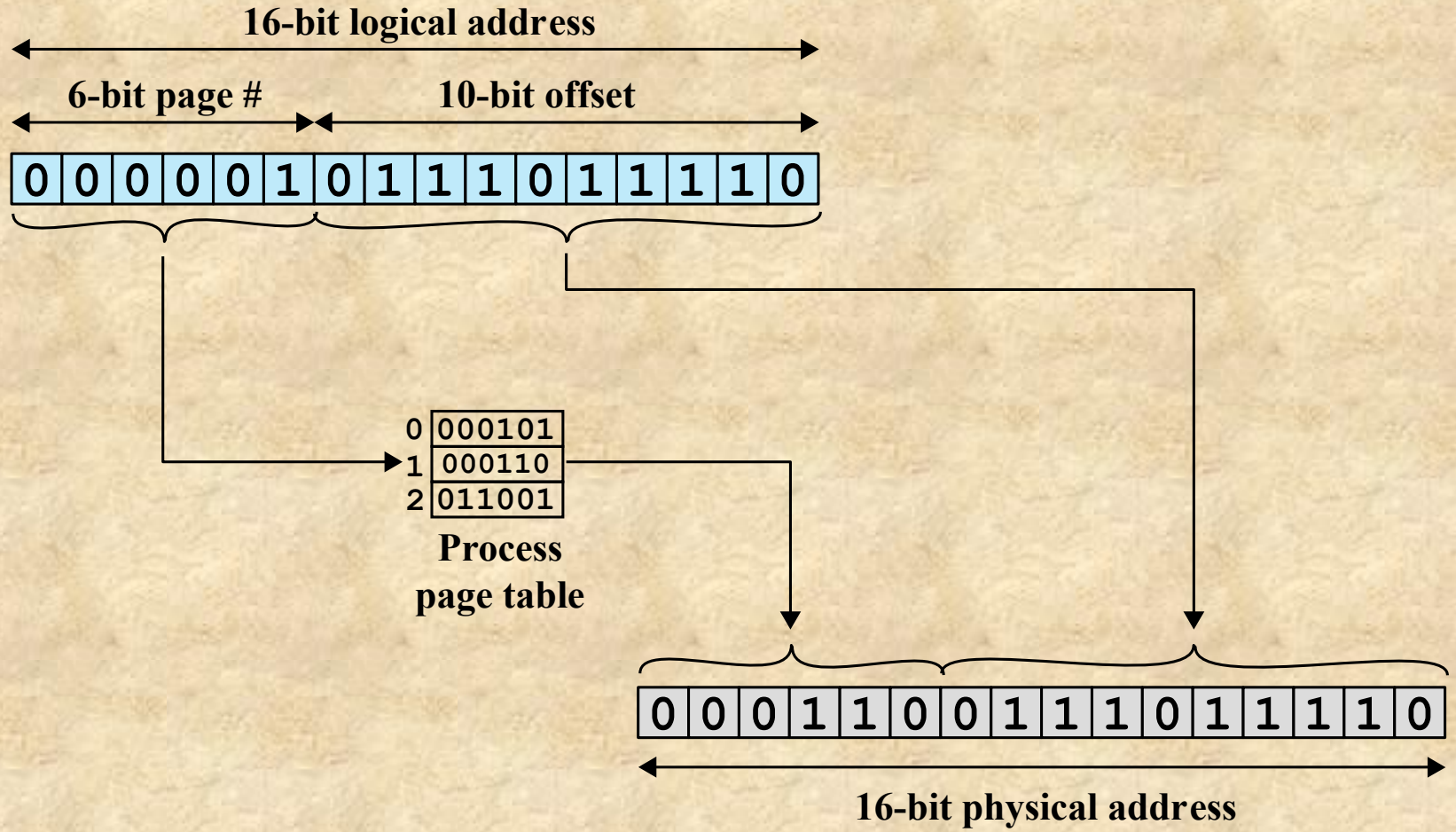


Figure 7.12 Examples of Logical-to-Physical Address Translation

Segmentation

- A program can be subdivided into segments
  - may vary in length
  - there is a maximum length
- Addressing consists of two parts:
  - a segment number
  - an offset

Segmentation

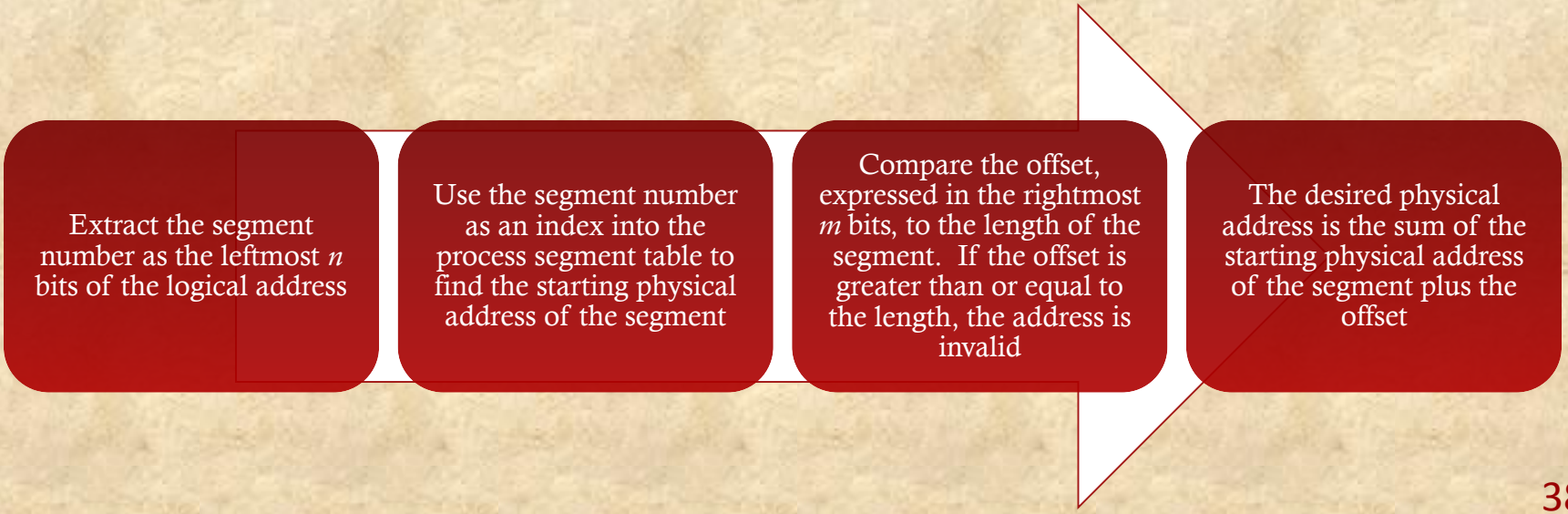
- Similar to dynamic partitioning
  - A program may occupy more than one partition, which need not be contiguous
  - No internal fragmentation, but suffers from external fragmentation
    - because a process is broken up into a number of smaller pieces, the external fragmentation should be less

Segmentation

- Usually visible whereas paging is invisible to the programmer
- Provided as a convenience for organizing programs and data
- Typically the programmer will assign programs and data to different segments
- For purposes of modular programming the program or data may be further broken down into multiple segments
  - the principal inconvenience of this service is that the programmer must be aware of the maximum segment size limitation

Address Translation

- Analogous to paging, a simple segmentation scheme would make use of a segment table for each process addresses
- The entry should also provide the starting address and the length of the segment to assure that invalid addresses are not used
- The following steps are needed for address translation:



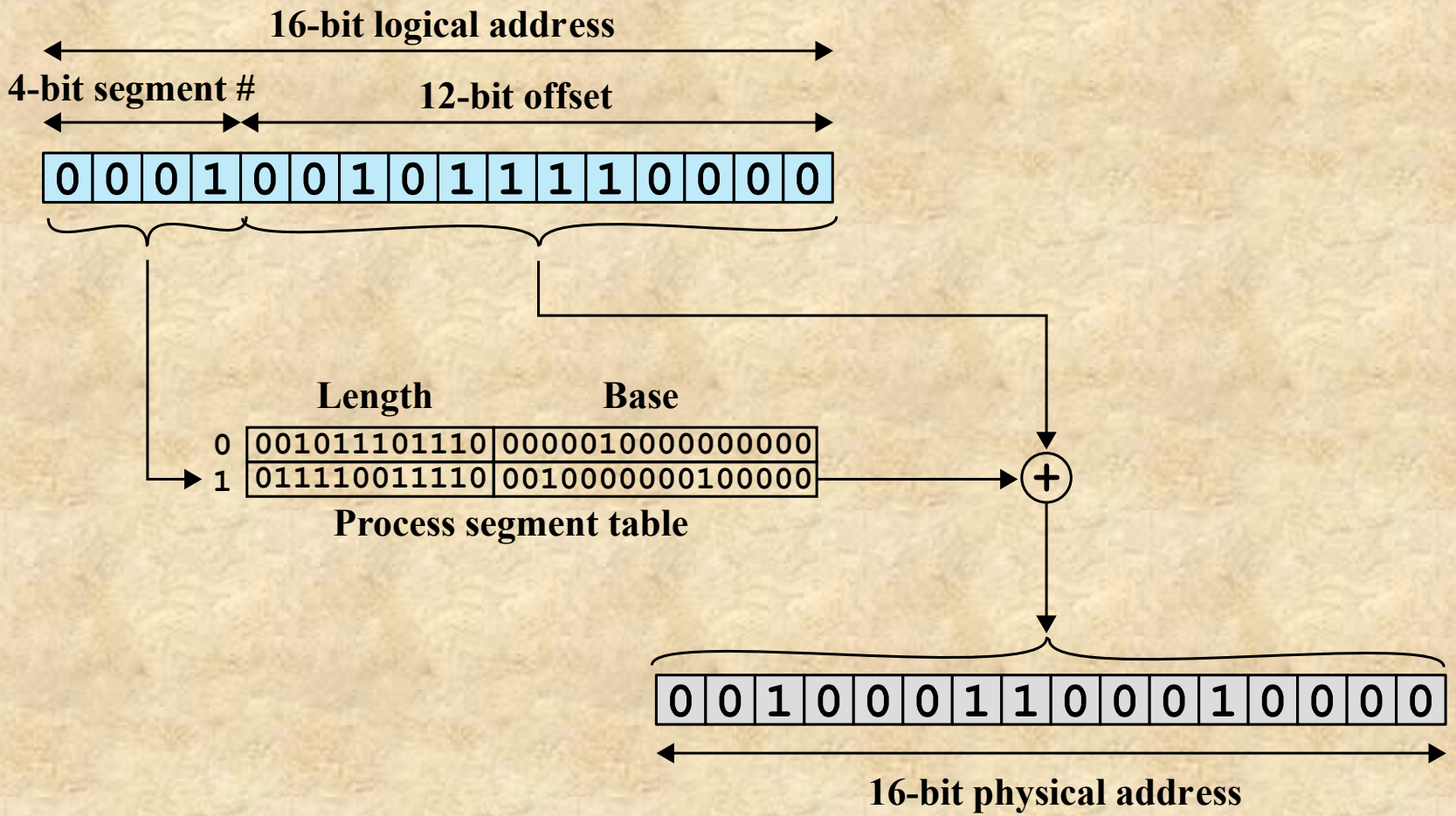


Figure 7.12 Examples of Logical-to-Physical Address Translation

Summary

- Memory management requirements
  - relocation
  - protection
  - sharing
  - logical organization
  - physical organization
- Paging
- Memory partitioning
  - fixed partitioning
  - dynamic partitioning
  - buddy system
  - relocation
- Segmentation