FONTYS UNIVERSITY OF APPLIED SCIENCES › FOR SOCIETY

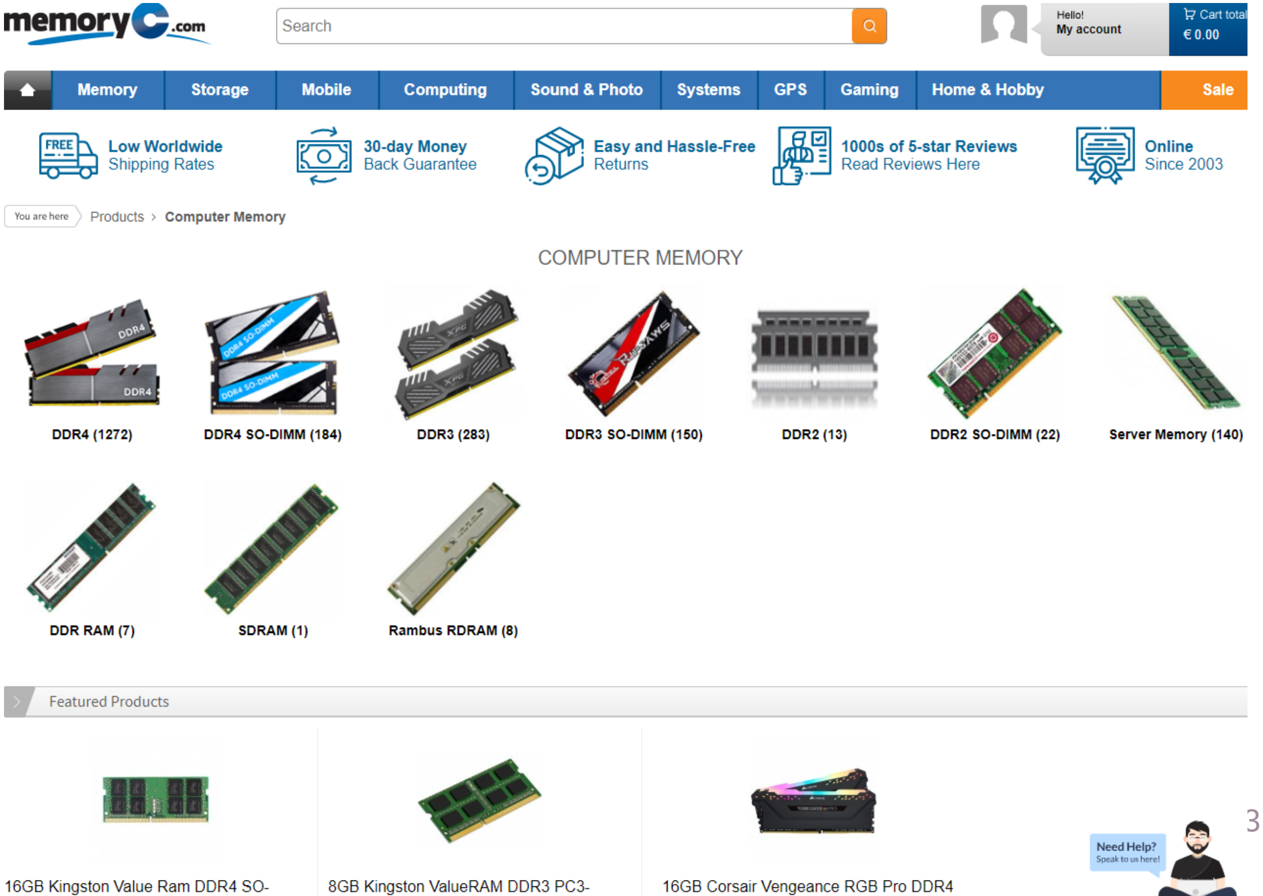
Chapter 7 – Memory management

Lecture 5

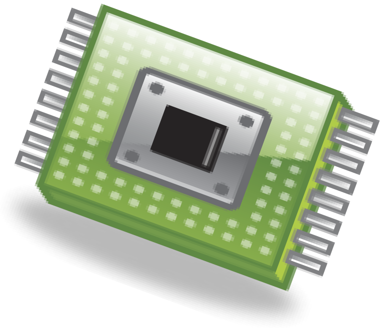
Roadmap

- **Basic requirements of Memory Management**
- Memory Partitioning
- Basic blocks of memory management
  - Paging
  - Segmentation

Why do we need memory management?



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).

Memory Management Requirements

- Relocation
- Protection
- Sharing
- Logical organisation
- Physical organisation

Requirements: Relocation

- The programmer does not know where the program will be placed in memory when it is executed,
  - it may be swapped to disk and return to main memory at a different location (relocated)
- Memory references must be translated to the actual physical memory address

Addressing

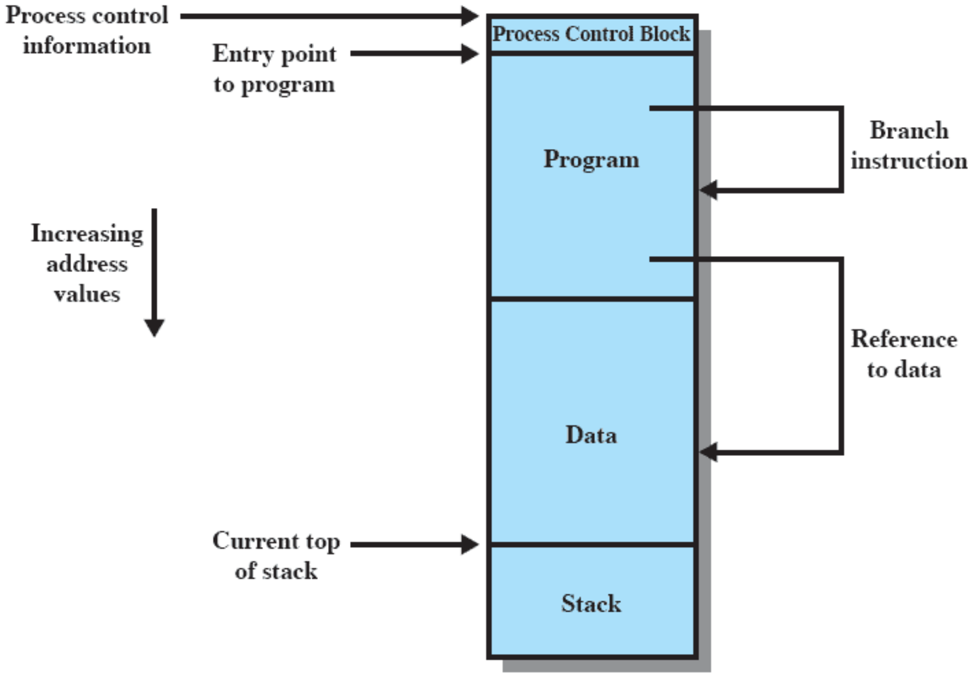


Figure 7.1 Addressing Requirements for a Process

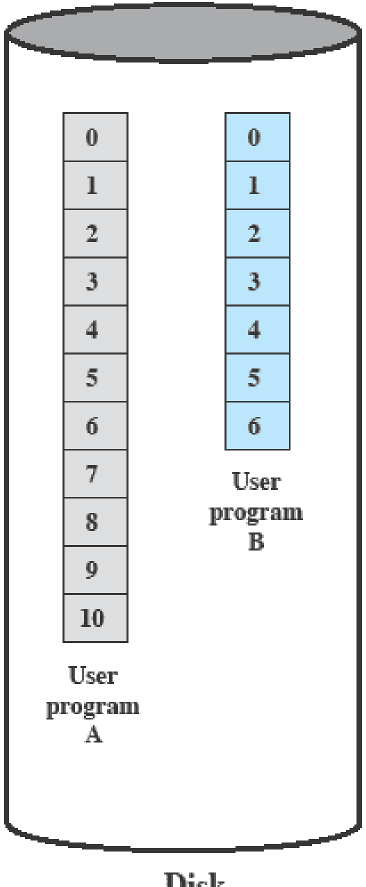
Requirements: Protection

- Processes should not be able to reference memory locations in another process without permission
- Impossible to check absolute addresses at compile time → Must be checked at run time

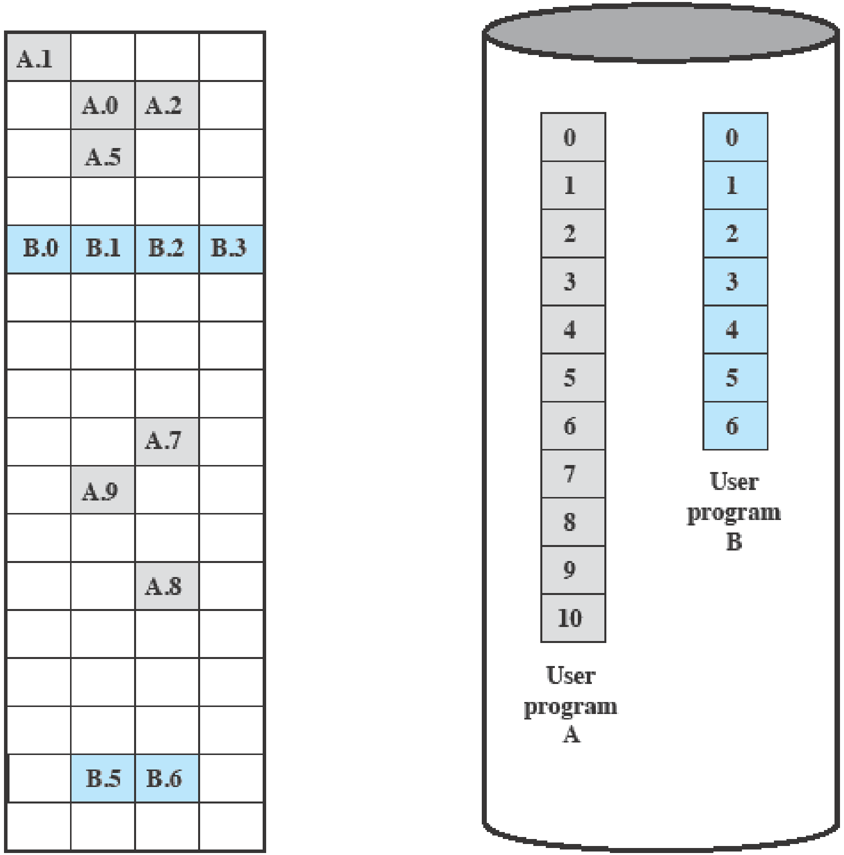
Requirements: Sharing

- Allow several processes to access the same portion of memory
- Better to allow each process access to the same copy of the program rather than have their own separate copy

Requirements: Logical Organization



Requirements: Physical Organization



Roadmap

- Basic requirements of Memory Management
- **Memory Partitioning**
- Basic blocks of memory management
  - Paging
  - Segmentation

Partitioning

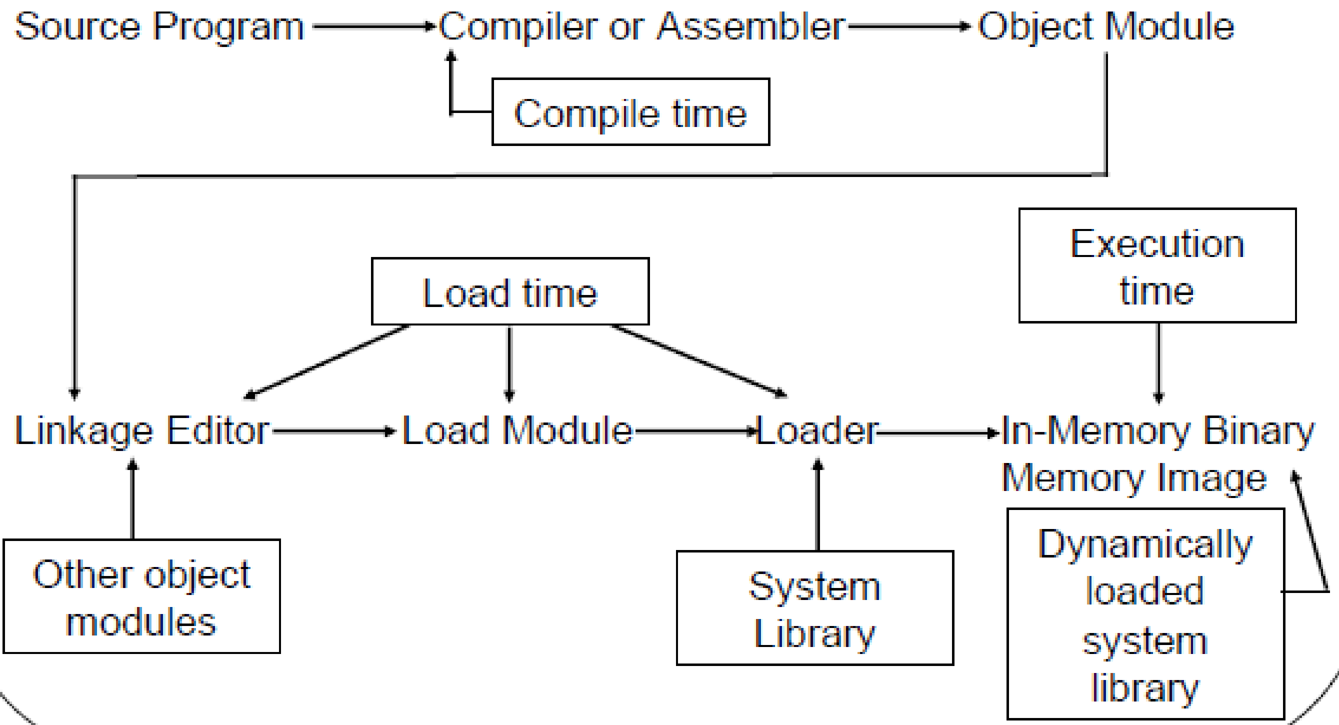
- An early method of managing memory
  - Pre-virtual memory
  - Not used much now
- **But**, it will clarify the later discussion of virtual memory if we look first at partitioning
  - Virtual Memory has evolved from the partitioning methods

Types of Partitioning

- Fixed Partitioning
- Dynamic Partitioning
- Simple Paging
- Simple Segmentation
- Virtual Memory Paging
- Virtual Memory Segmentation

Background

- User programs go through several steps before being executed:



Fixed Partitioning

- Equal-size partitions (see fig 7.3a)
  - Any process whose size is less than or equal to the partition size can be loaded into an available partition
- The operating system can swap a process out of a partition
  - If none are in a ready or running state

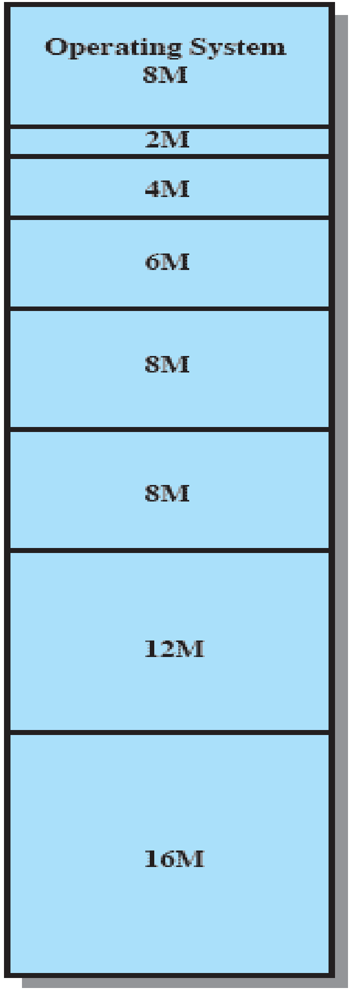
(a) Equal-size partitions

Fixed Partitioning Problems

- A program may not fit in a partition.
  - The programmer must design the program with overlays
- Main memory use is inefficient.
  - Any program, no matter how small, occupies an entire partition.
  - This is results in ***internal fragmentation.***

Solution – Unequal Size Partitions

- Lessens both problems
  - but doesn't solve completely
- In Figure:
  - Programs up to 16M can be accommodated without overlay
  - Smaller programs can be placed in smaller partitions, reducing internal fragmentation

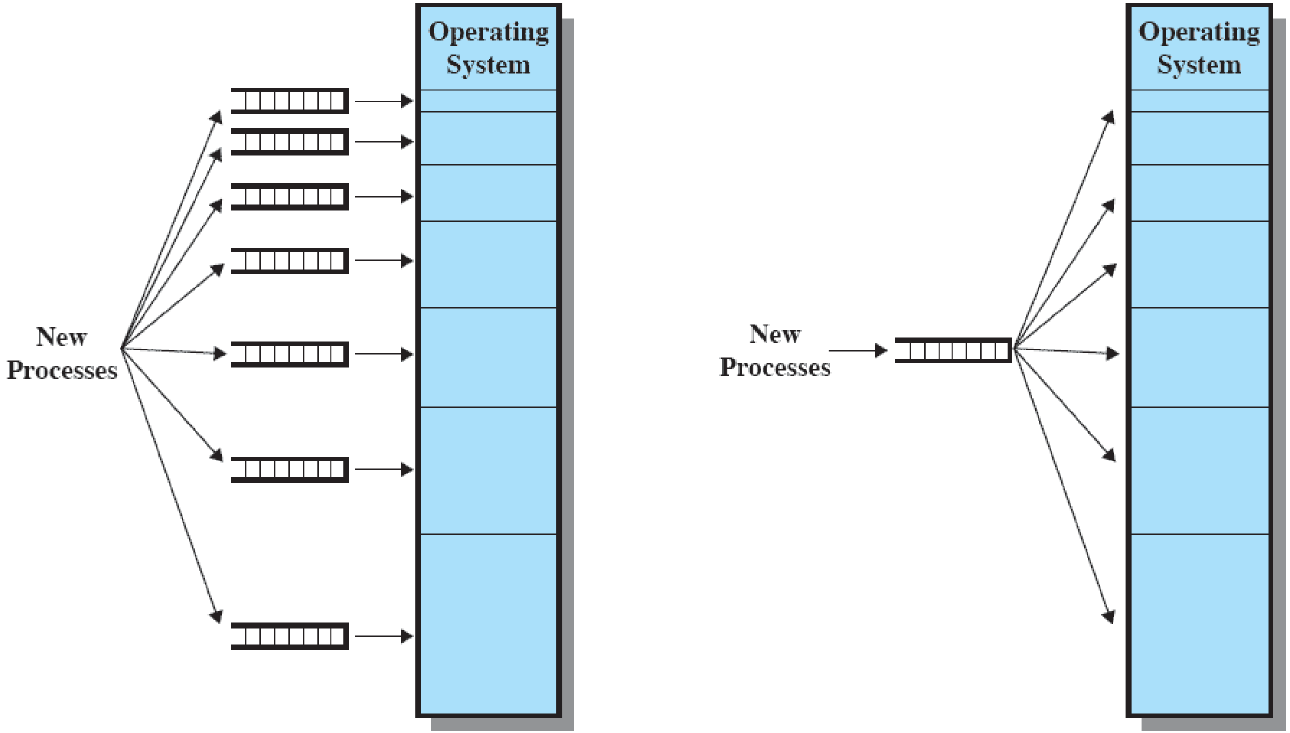


(b) Unequal-size partitions

Placement Algorithm

- Equal-size
  - Placement is trivial (no options)
- Unequal-size
  - Can assign each process to the smallest partition within which it will fit
  - Queue for each partition
  - Processes are assigned in such a way as to minimize wasted memory within a partition

Memory assignment - Fixed Partitioning



(a) One process queue per partition

(b) Single queue

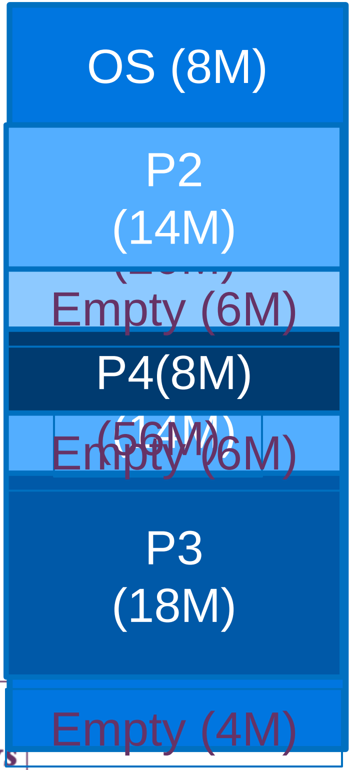
Remaining problems

- How many active processes?
- What about fragmentation?

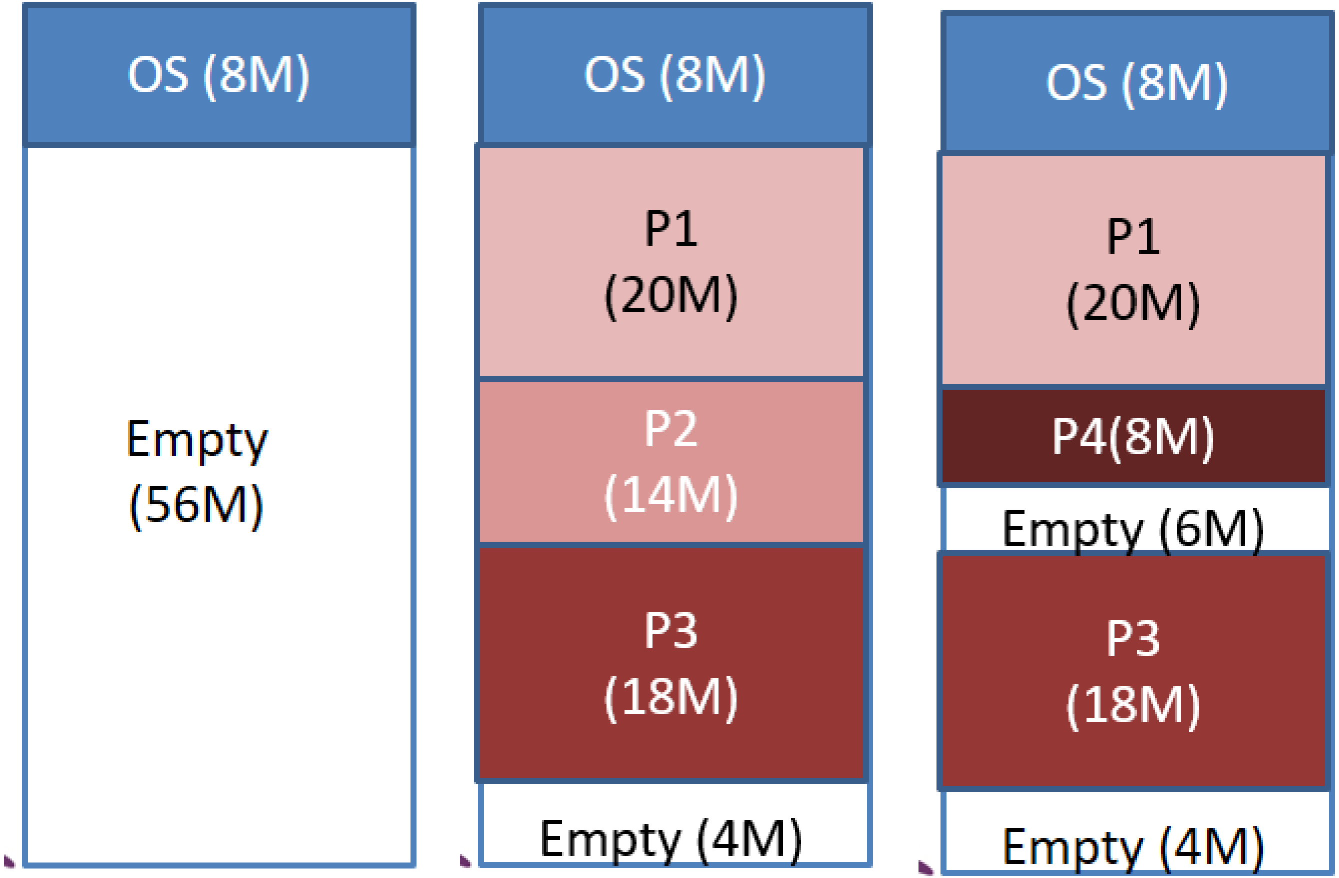
Dynamic Partitioning

- Partitions are of variable length and number
- Process is allocated exactly as much memory as required

Dynamic Partitioning Example



- ***External Fragmentation***
  - Memory external to all processes is fragmented
- Can resolve using ***compaction***
  - OS moves processes so that they are contiguous
  - Time consuming and wastes CPU time


OS (8M)
Empty
(56M)
OS (8M)
P1
(20M)
P2
(14M)
P3
(18M)
Empty (4M)
OS (8M)
P1
(20M)
P4(8M)
Empty (6M)
P3
(18M)
Empty (4M)

Dynamic Partitioning

- Operating system must decide which free block to allocate to a process
- Best-fit algorithm
  - Chooses the block that is closest in size to the request
  - Worst performer overall
  - Since smallest block is found for process, the smallest amount of fragmentation is left
  - Memory compaction must be done more often

Dynamic Partitioning

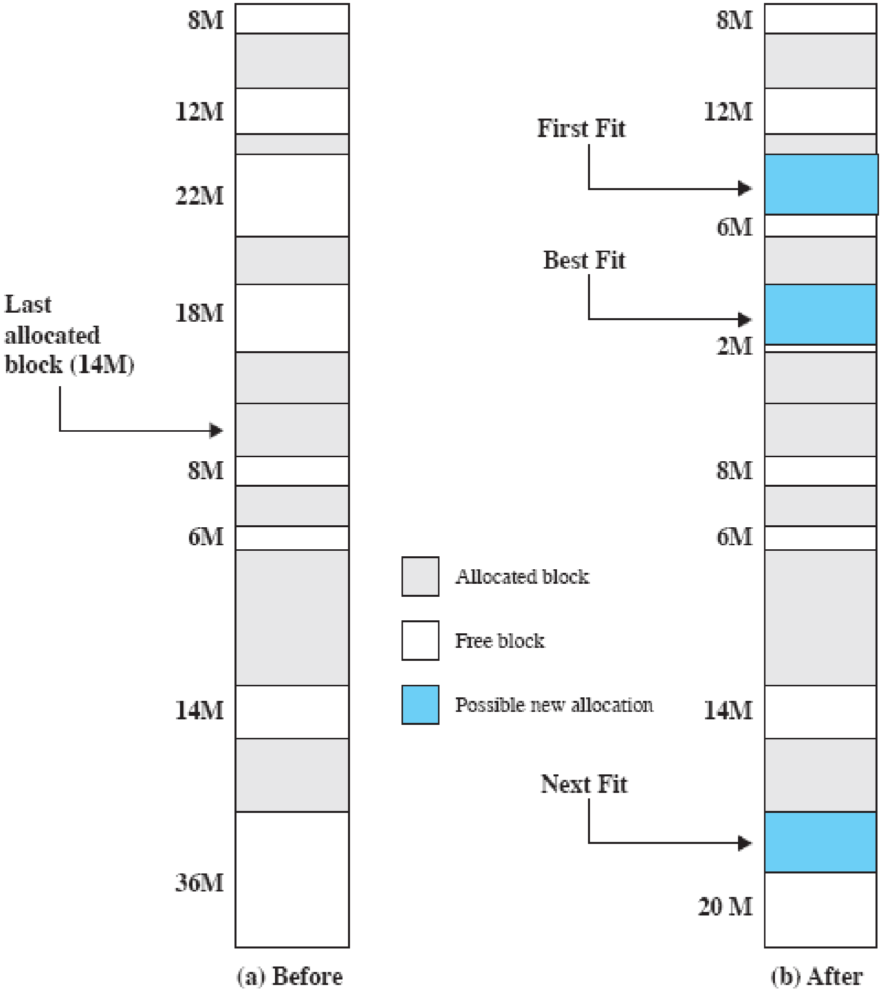
- First-fit algorithm
  - Scans memory from the beginning and chooses the first available block that is large enough
  - Fastest
  - May have many process loaded in the front end of memory that must be searched over when trying to find a free block

Dynamic Partitioning

- Next-fit
  - Scans memory from the location of the last placement
  - More often allocate a block of memory at the end of memory where the largest block is found
  - The largest block of memory is broken up into smaller blocks
  - Compaction is required to obtain a large block at the end of memory

Allocation

Where to place 16Mb



Break

Types of Partitioning

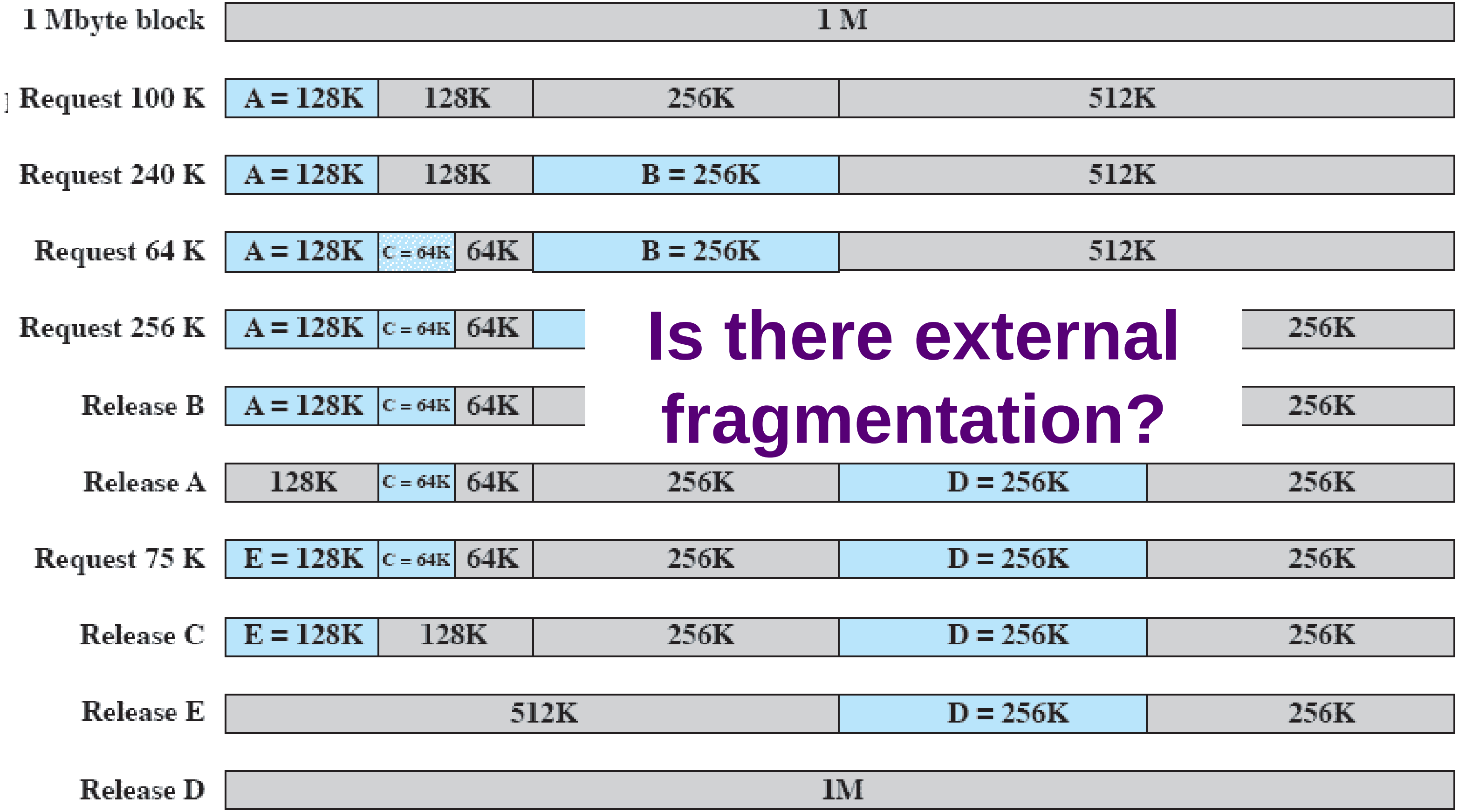
- Fixed Partitioning
- Dynamic Partitioning
- Simple Paging
- Simple Segmentation
- Virtual Memory Paging
- Virtual Memory Segmentation

Buddy System

- Entire space available is treated as a single block of 2^U
- If a request of size *s* where $2^{U-1} < s <= 2^U$
  - entire block is allocated
- Otherwise block is split into two equal buddies
  - Process continues until smallest block greater than or equal to *s* is generated

Fontys

Example of Buddy System



Is there external fragmentation?

Tree Representation of Buddy System

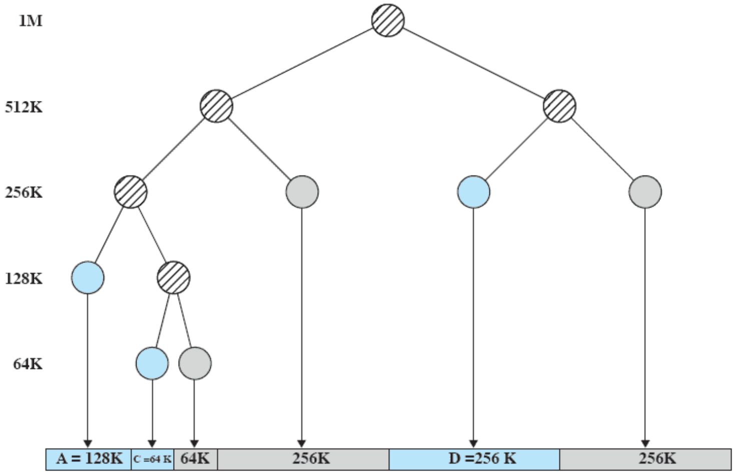


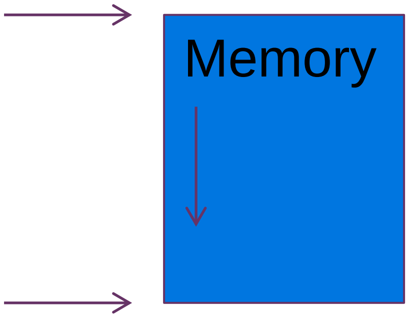
Figure 7.7 Tree Representation of Buddy System

Addresses

- Logical
  - Reference to a memory location independent of the current assignment of data to memory.
- Relative
  - Address expressed as a location relative to some known point.
- Physical or Absolute
  - The absolute address or actual location in main memory.

Registers Used during Execution

- Base register
  - Starting address for the process
- Bounds register
  - Ending location of the process
- These values are set when the process is loaded or when the process is swapped in



Relative to Physical addressing

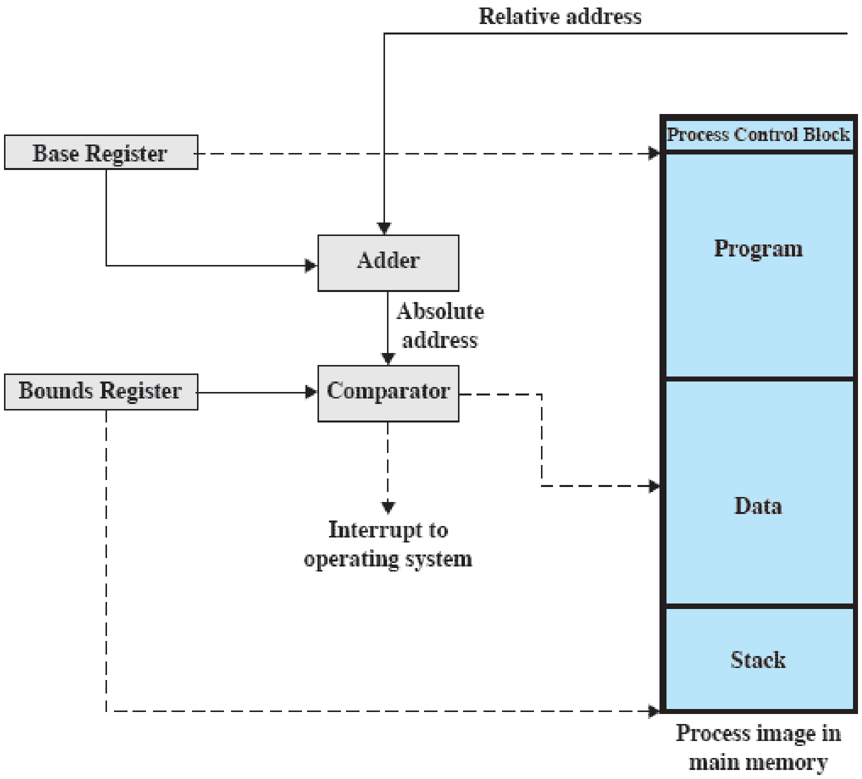


Figure 7.8 Hardware Support for Relocation

Roadmap

- Basic requirements of Memory Management
- Memory Partitioning
- **Basic blocks of memory management**
  - **Paging**
  - **Segmentation**

Paging

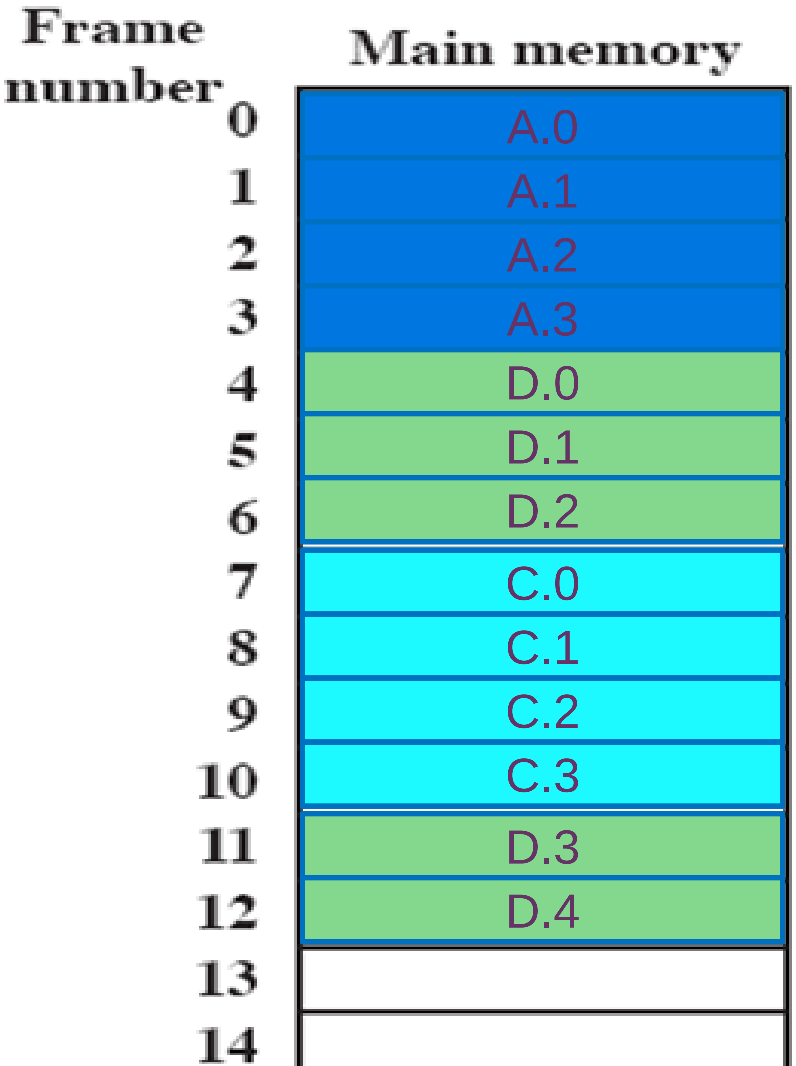
Partition memory into small equal fixed-size chunks and divide each process into the *same size* chunks

- The chunks of a process are called ***pages***
- The chunks of memory are called ***frames***

Paging

- Operating system maintains a page table for each process
  - Contains the frame location for each page in the process
  - Memory address consist of a page number and offset within the page

Processes and Frames



Page Table

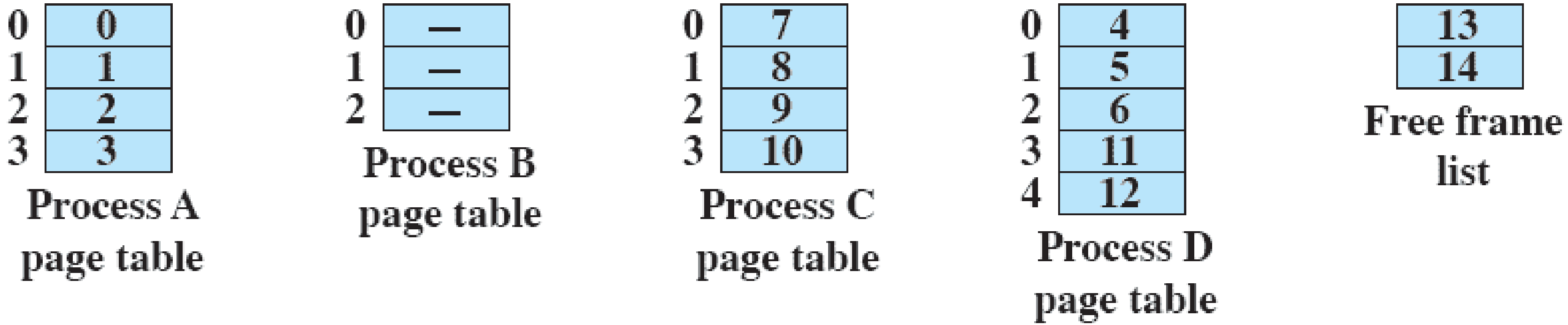


Figure 7.10 Data Structures for the Example of Figure 7.9 at Time Epoch (f)

Logical Addresses

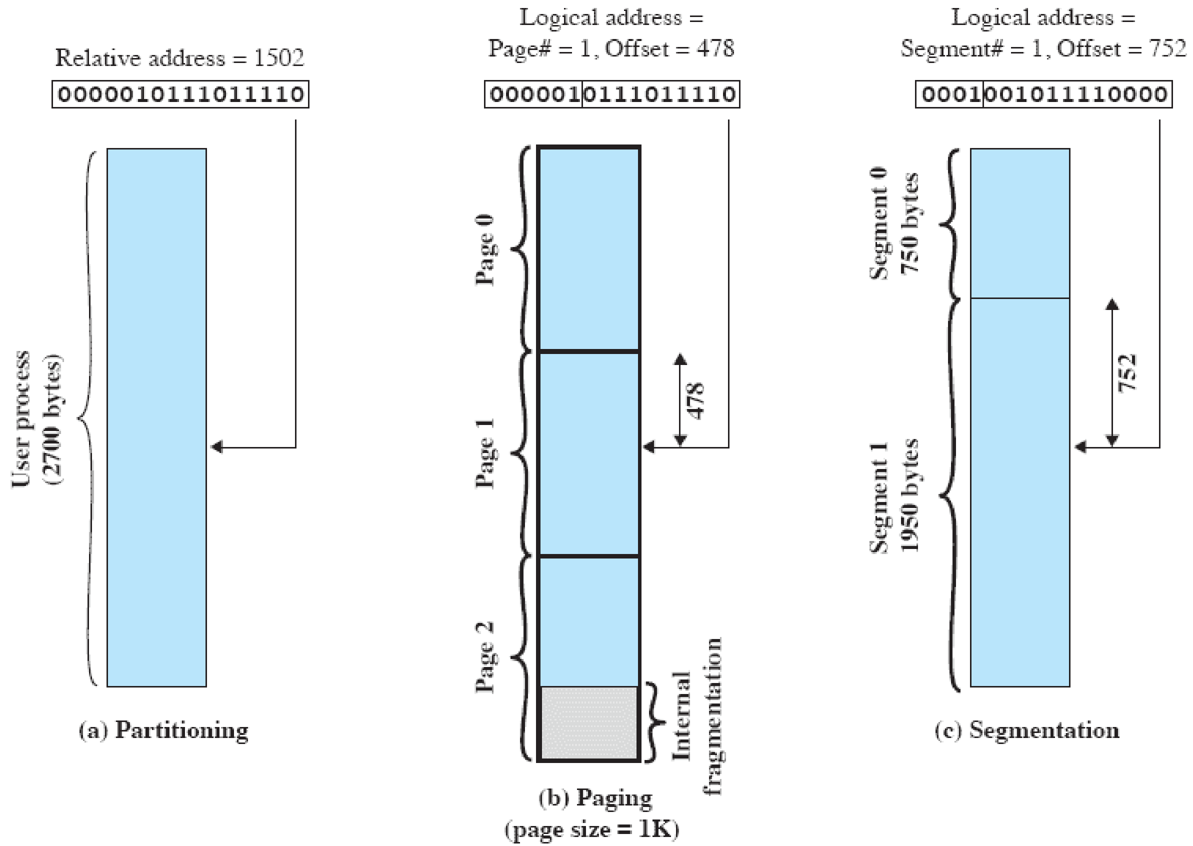
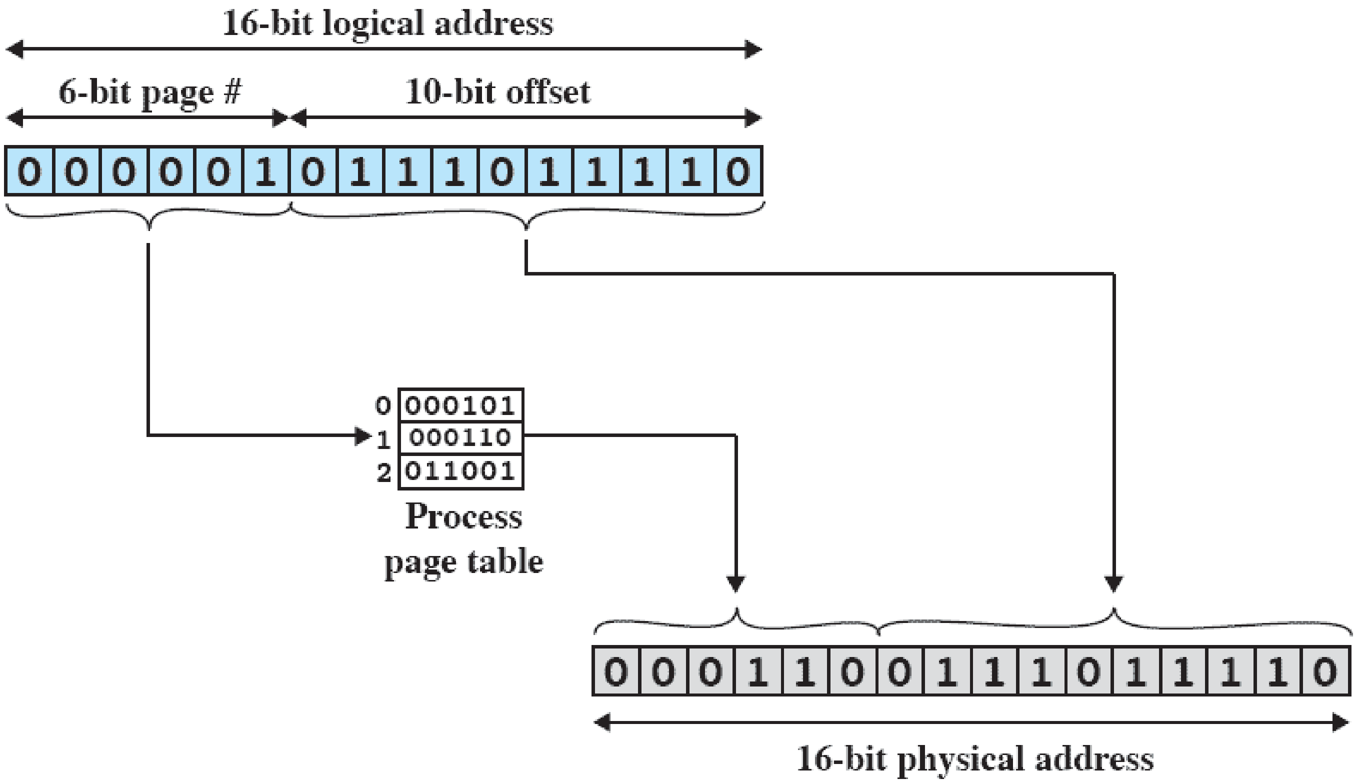


Figure 7.11 Logical Addresses

Paging



(a) Paging

Fontys

Segmentation

- A program can be subdivided into segments; Module, procedure, stack, data, file, etc. Segments may vary in length
  - There is a maximum segment length
- Addressing consist of two parts
  - a segment number and
  - an offset
- Segmentation is similar to dynamic partitioning

Segmentation

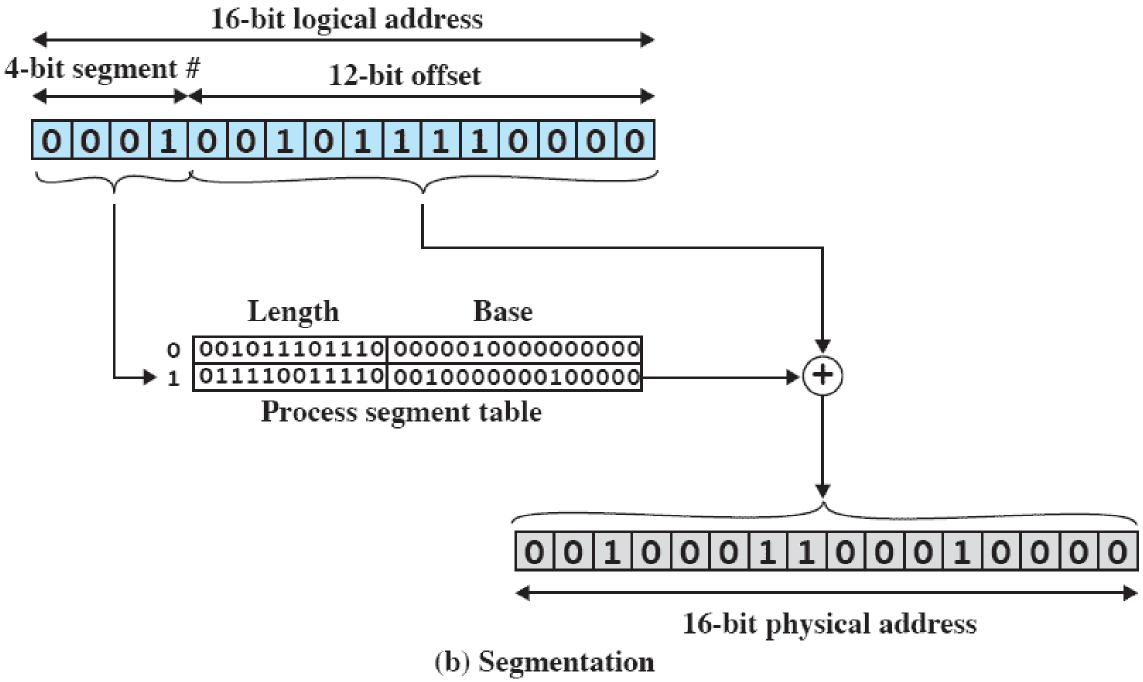


Figure 7.12 Examples of Logical-to-Physical Address Translation

Random selection
&
Practical assignment explanation

Questions?