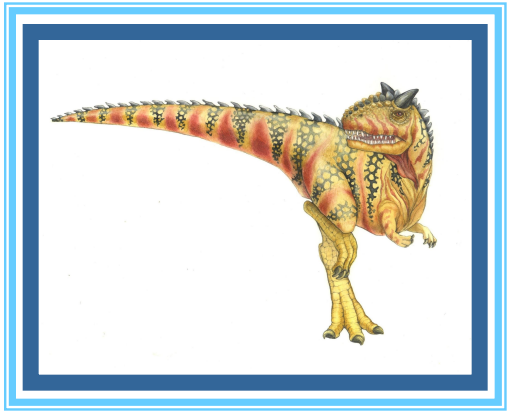
Chapter 8: Main Memory



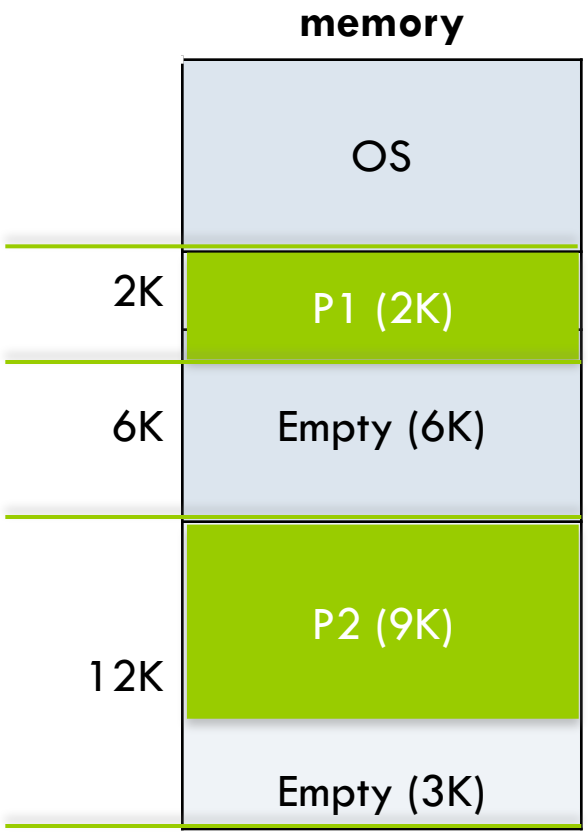
Recap

- Swapping
- Contiguous Memory Allocation
  - Fixed size partitioning
  - Variable size partitioning
- Dynamic Storage-Allocation Problem
  - First Fit
  - Best Fit
  - Worst Fit
- Fragmentation
  - Internal Fragmentation
  - External Fragmentation
- Compaction

Fragmentation

- **External Fragmentation** – total memory space exists to satisfy a request, but it is not contiguous
- **Internal Fragmentation** – allocated memory may be slightly larger than requested memory; this size difference is memory internal to a partition, but not being used

Fragmentation



If a whole partition is currently not being used, then it is called *an* ***external fragmentation.***

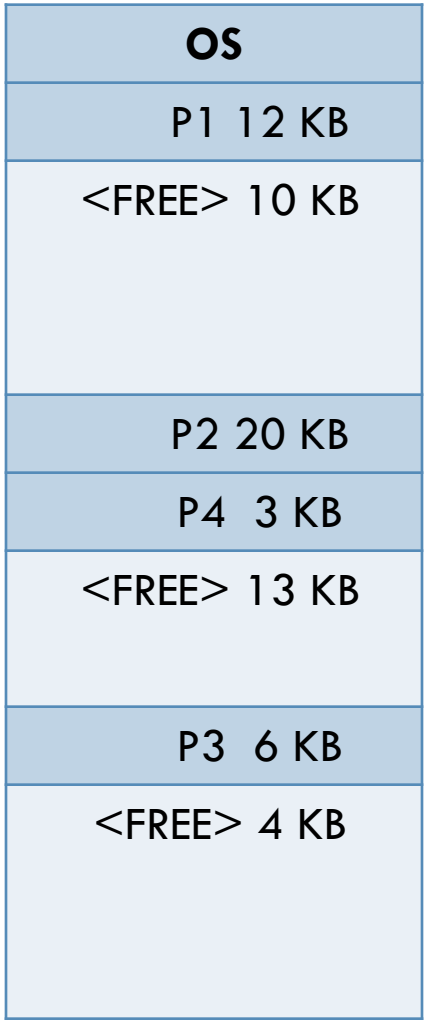
If a partition is being used by a process requiring some memory smaller than the partition size, then it is called an ***internal fragmentation.***

Compaction

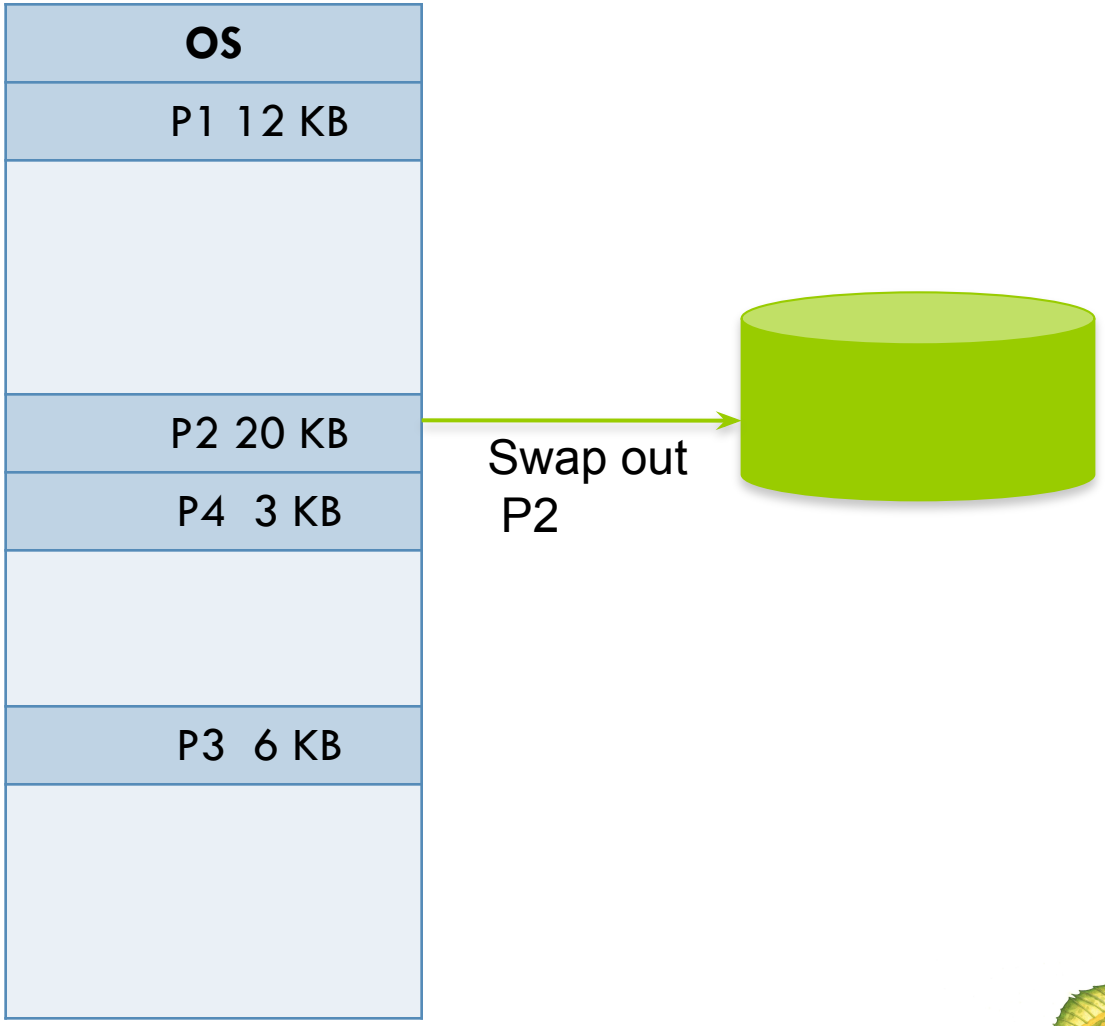
- Compaction is a method to overcome the external fragmentation problem.
- All free blocks are brought together as one large block of free space.
- Compaction requires dynamic relocation.
- Certainly, compaction has a cost and selection of an optimal compaction strategy is difficult.
- One method for compaction is swapping out those processes that are to be moved within the memory, and swapping them into different memory locations

Compaction

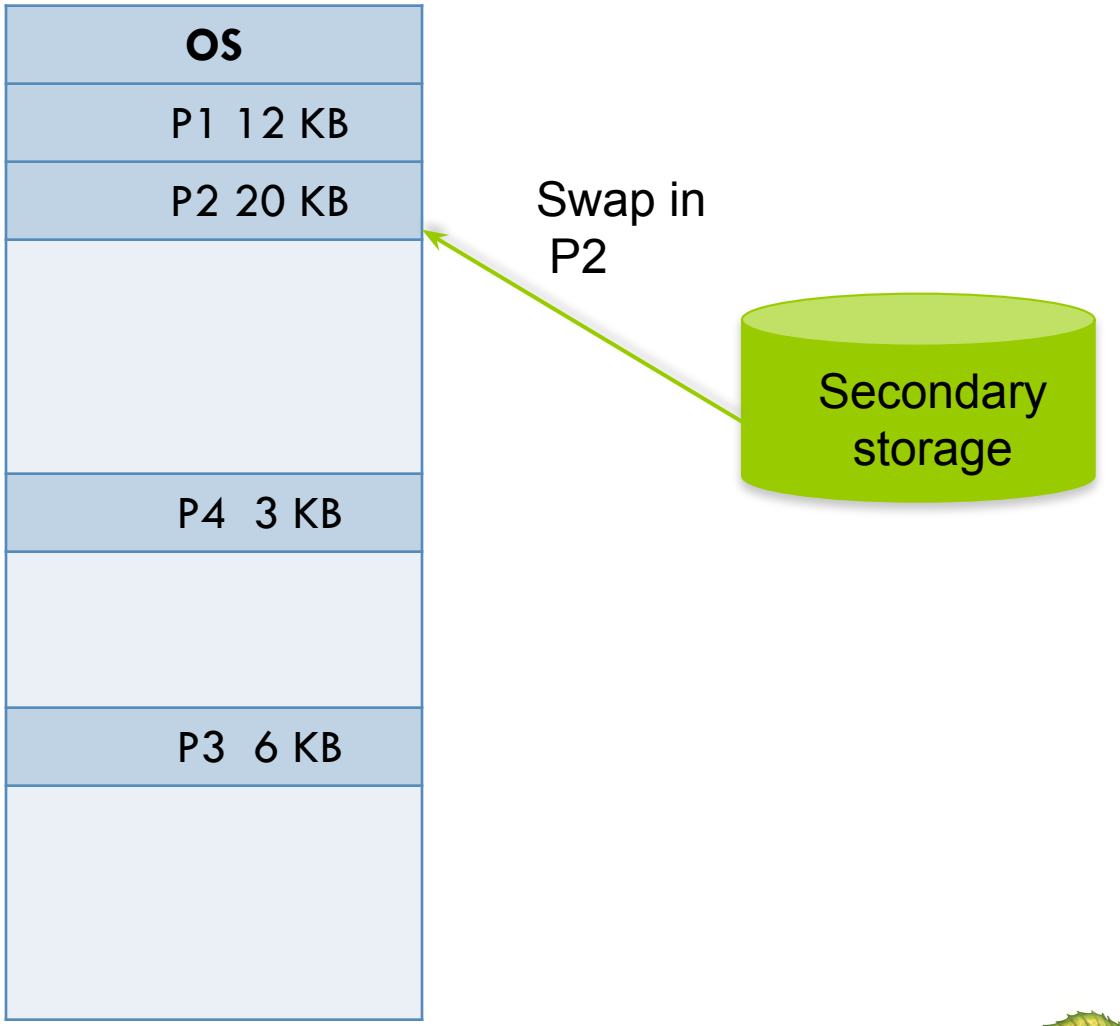
Memory mapping before compaction



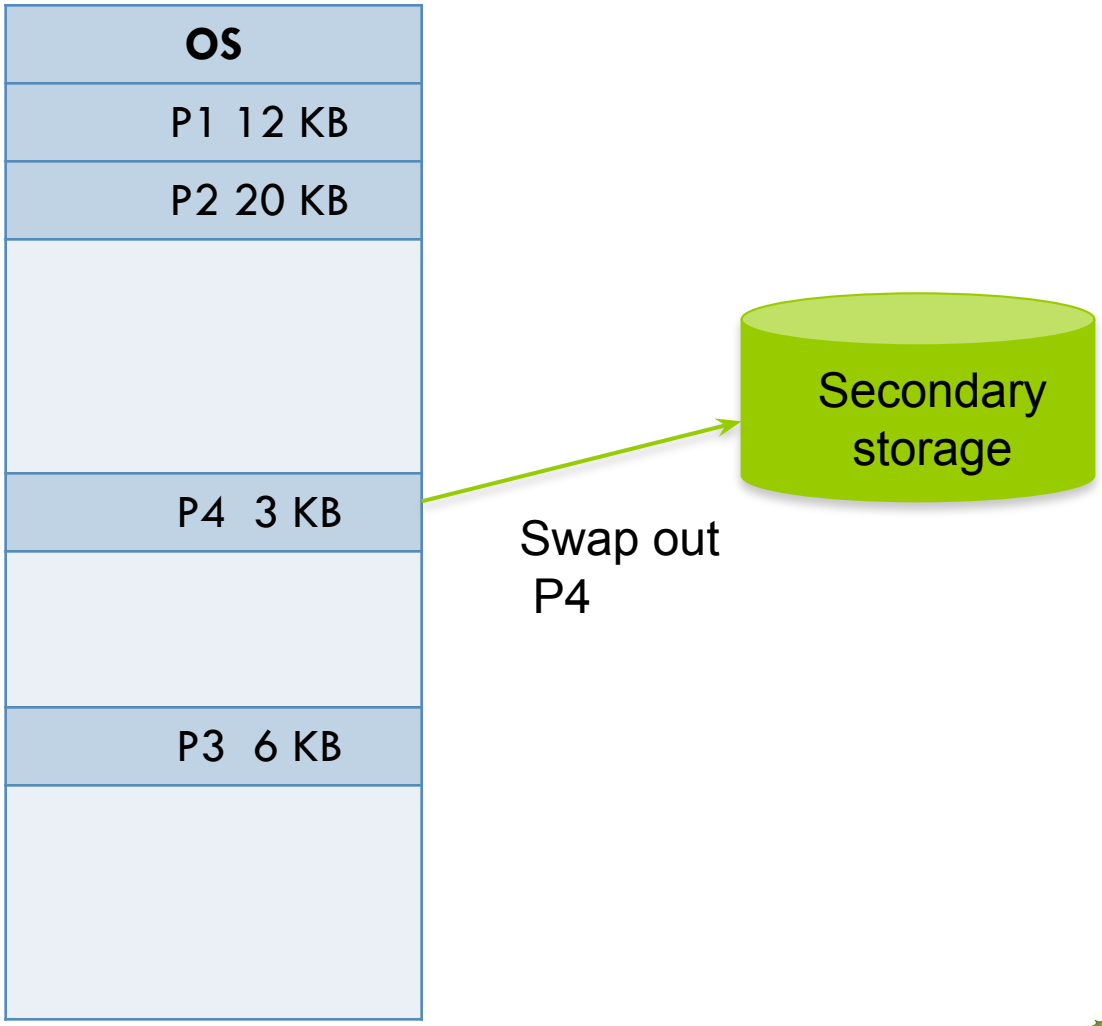
Compaction Example



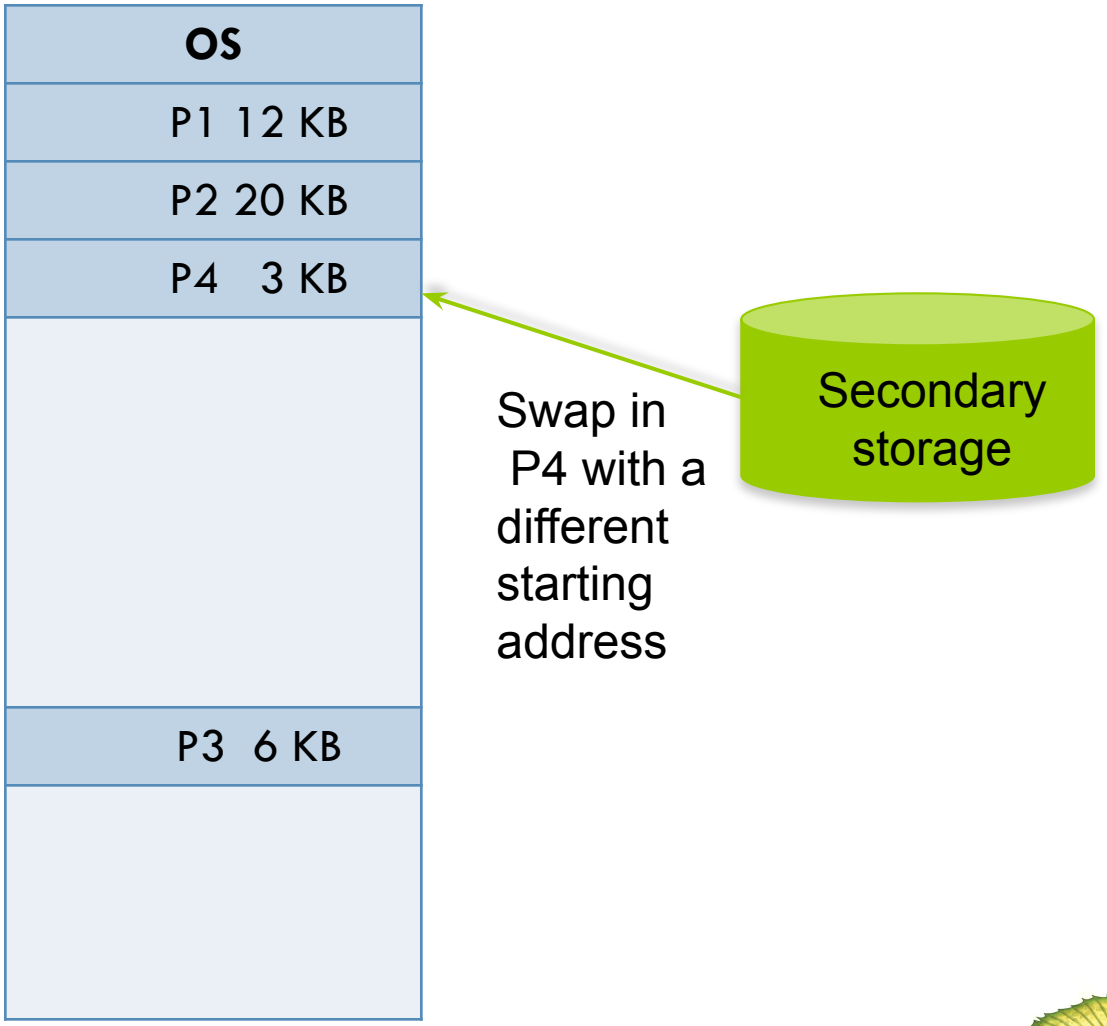
Compaction (Cont.)



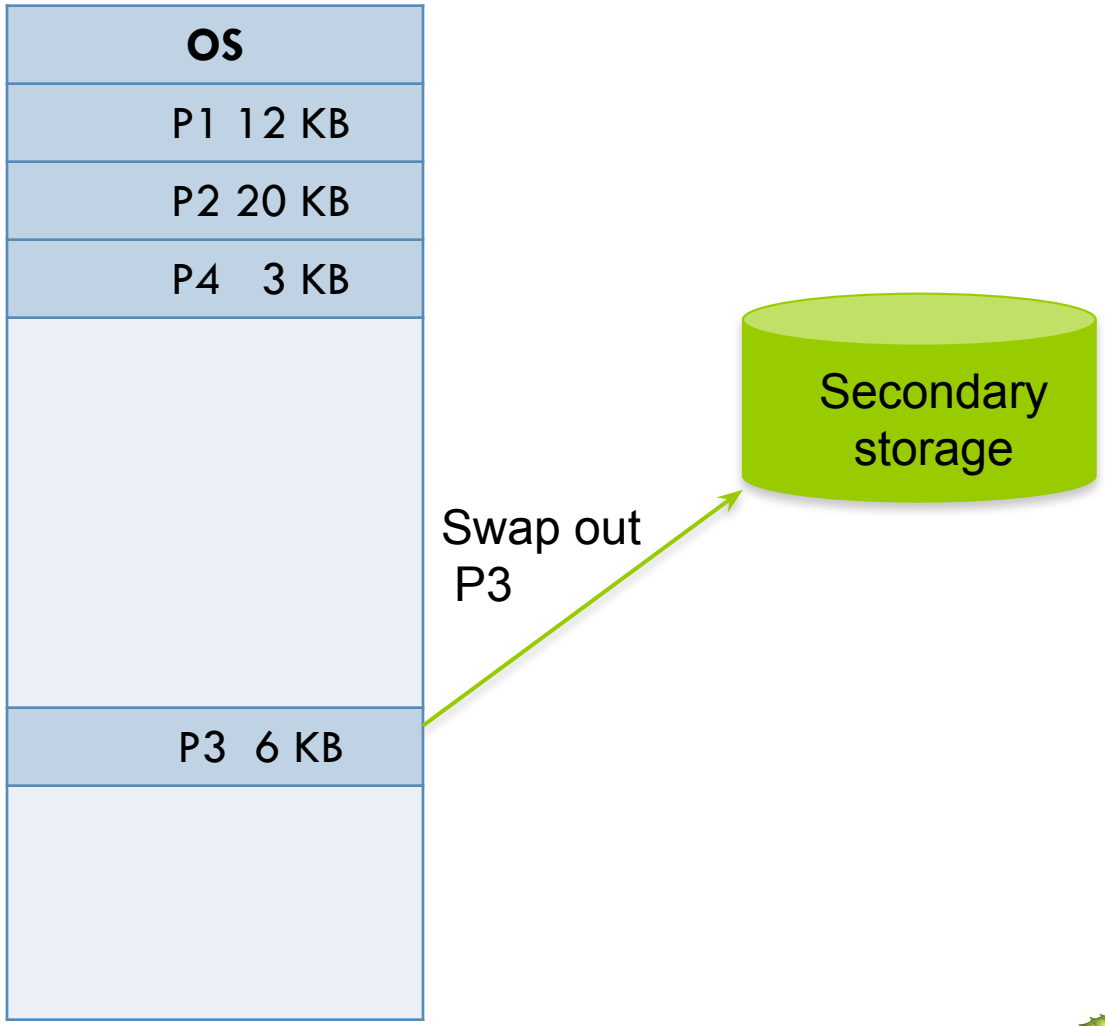
Compaction Example (Cont.)



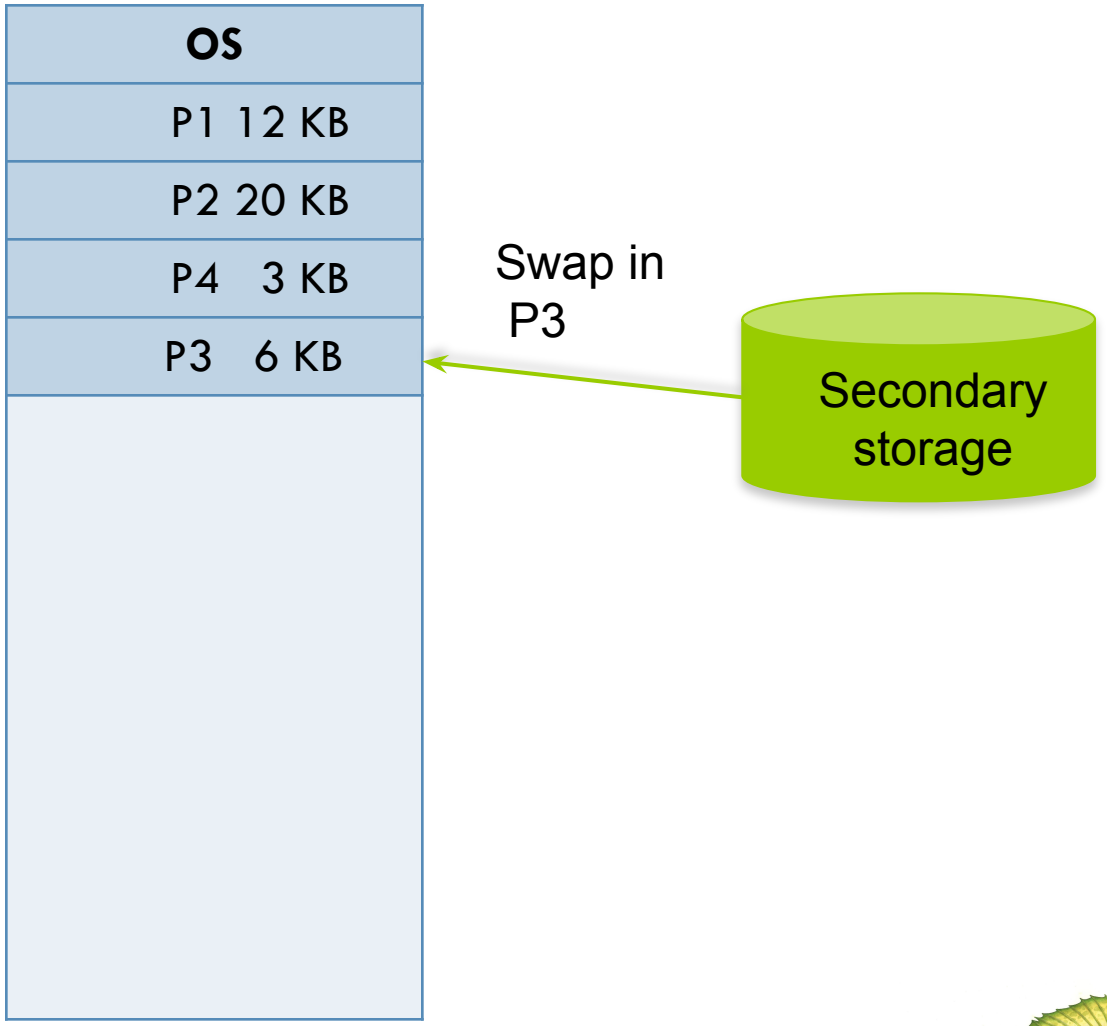
Compaction Example (Cont.)



Compaction Example (Cont.)

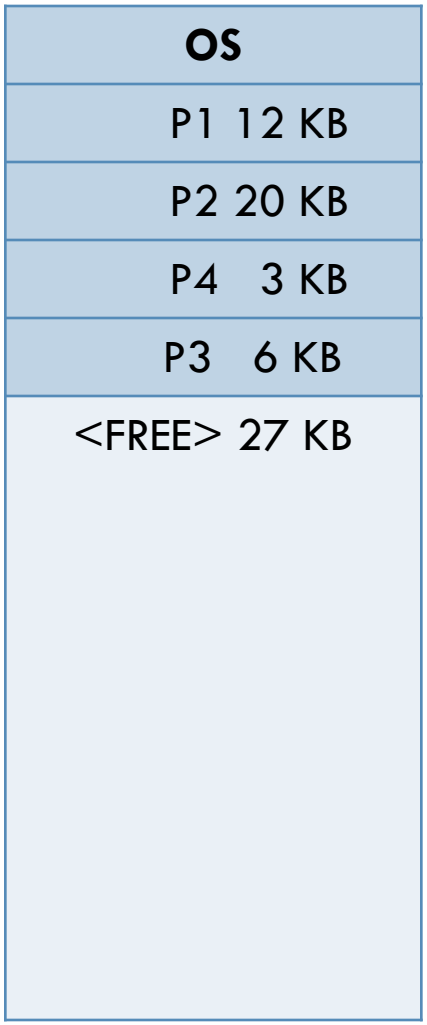


Compaction Example (Cont.)



Compaction Example (Cont.)

Memory mapping after compaction



Now P5 of 15KB can be loaded here

Compaction Example (Cont.)

OS
P1 12 KB
P2 20 KB
P4 3 KB
P3 6 KB
P5 12 KB
<FREE> 12 KB

P5 of 15KB is loaded

How does user view Memory

A process is divided into Segments. The chunks that a program is divided into are not necessarily to be of the same sizes are called segments. It is a non- Contiguous memory allocation scheme to processes.

Segmentation gives user's view of the process which paging does not give. Here the user's view is mapped to physical memory.

Thus the differentiation comes between the logical and physical memory.

Do users view memory as linear array of bytes some containing instruction and other containing data???

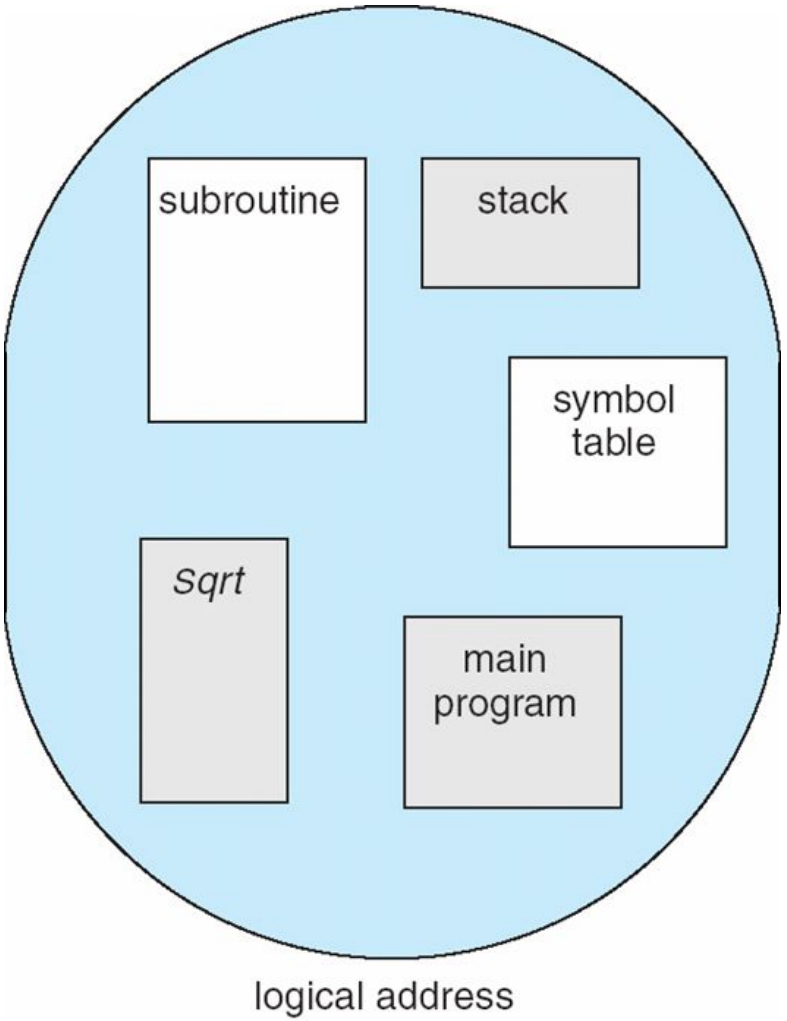
No…Rather they would see it as collection of segments

Segmentation

- Memory-management scheme that supports user view of memory
- A program is a collection of segments
  - A segment is a logical unit such as:

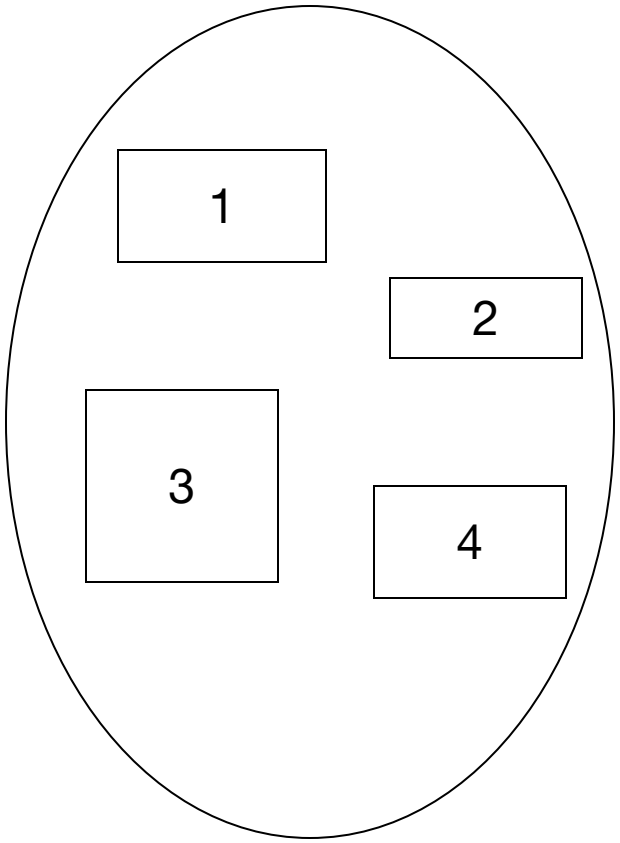
main program
procedure
function
method
object
local variables, global variables
common block
stack
symbol table
arrays

Segmentation

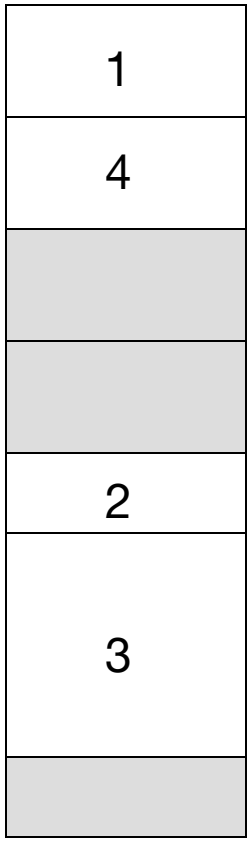


- User View of logical memory
  - Linear array of bytes
  - A collection of variable-sized entities
    - User thinks in terms of "subroutines", "stack", "symbol table", "main program" which are somehow located somewhere in memory.
- Segmentation supports this user view. The logical address space is a collection of segments.

Logical View of Segmentation



user space



physical memory space

Segmentation

- Although the user can refer to objects in the program by a two-dimensional address, the actual physical address is still a one-dimensional sequence
- Thus, we need to map the segment number
- This mapping is effected by a **segment table**
- In order to protect the memory space, each entry in segment table has a **segment base** and a **segment limit**

Segmentation Architecture

- Logical address consists of a two tuple:

  <segment-number, offset>,

- **Segment table** – maps two-dimensional physical addresses; each table entry has:
  - **base** – contains the starting physical address where the segments reside in memory
  - **limit** – specifies the length of the segment
- **Segment-table base register (STBR)** points to the segment table's location in memory
- **Segment-table length register (STLR)** indicates number of segments used by a program;

  segment number ***s*** is legal if ***s*** < **STLR**

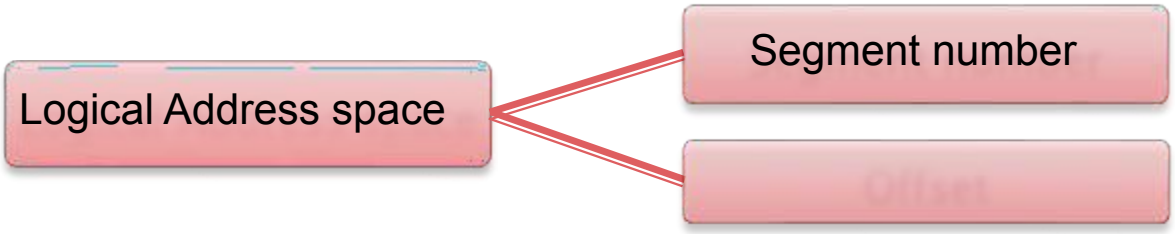
Segmentation Architecture

- **Segments are variable-sized**
  - Dynamic memory allocation required (first fit, best fit, worst fit).
- **External fragmentation**
  - In the worst case the largest hole may not be large enough to fit in a new segment.
- **Each process has its own segment table**
  - Each process has its own page table. The size of the segment table is determined by the number of segments, whereas the size of the page table depends on the total amount of memory occupied.
- Segment table located in the main memory

Segmentation Architecture (Cont.)

- Protection
  - With each entry in segment table associate:
    - validation bit = 0 ⇒ illegal segment
    - read/write/execute privileges
- Protection bits associated with segments; code sharing occurs at segment level
- Since segments vary in length, memory allocation is a dynamic storage-allocation problem

Logical Addressing in Segmentation



The mapping of the logical address to the physical address is done with the help of the segment table.

the length of the segment

SEGMENT TABLE

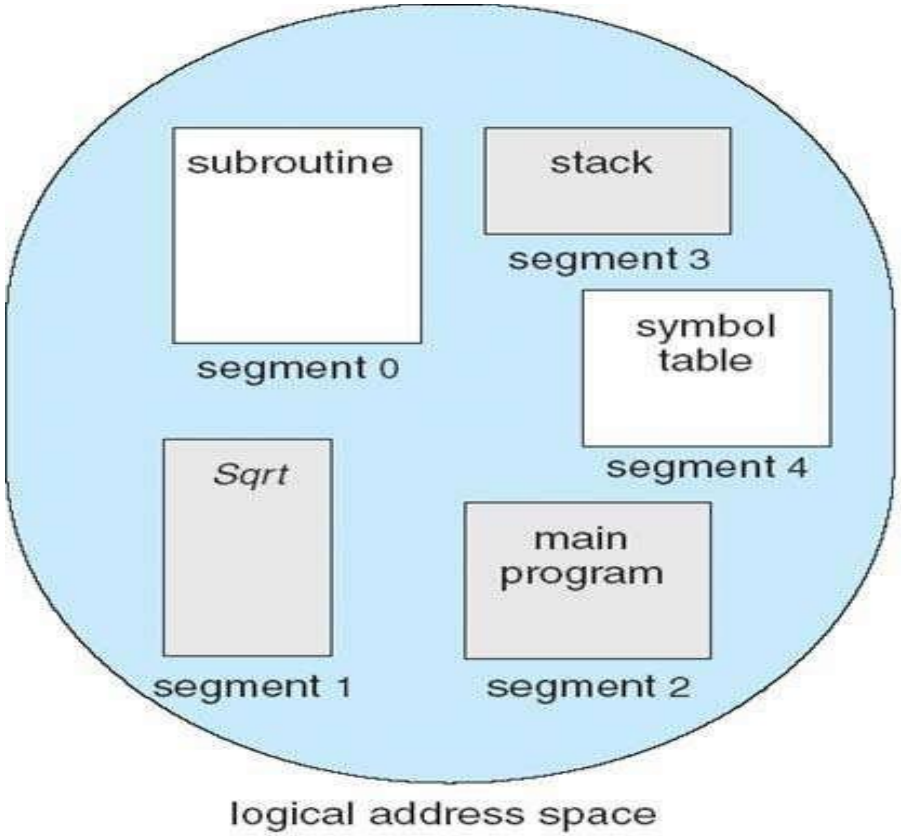
Segment Limit	Segment Base	Other bits

starting address of the corresponding segment in main memory

A bit is needed to determine if the segment is already in main memory (P)
Another bit is needed to determine if the segment has been modified since it was loaded in main memory (M)

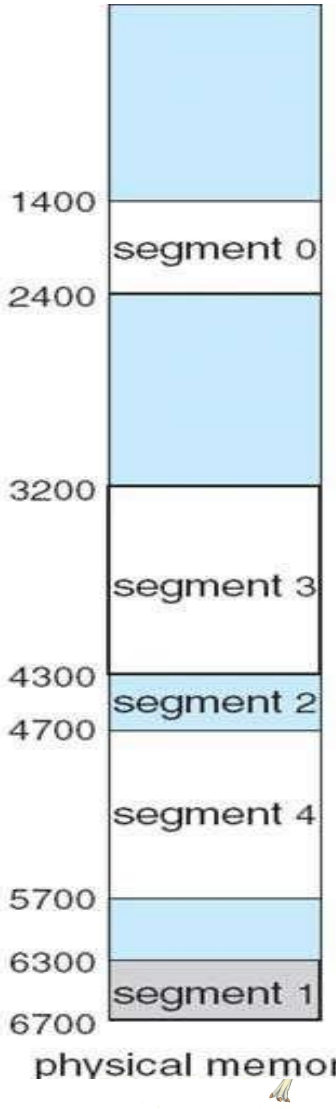
Example of Segmentation

A segmentation example is shown in the following diagram

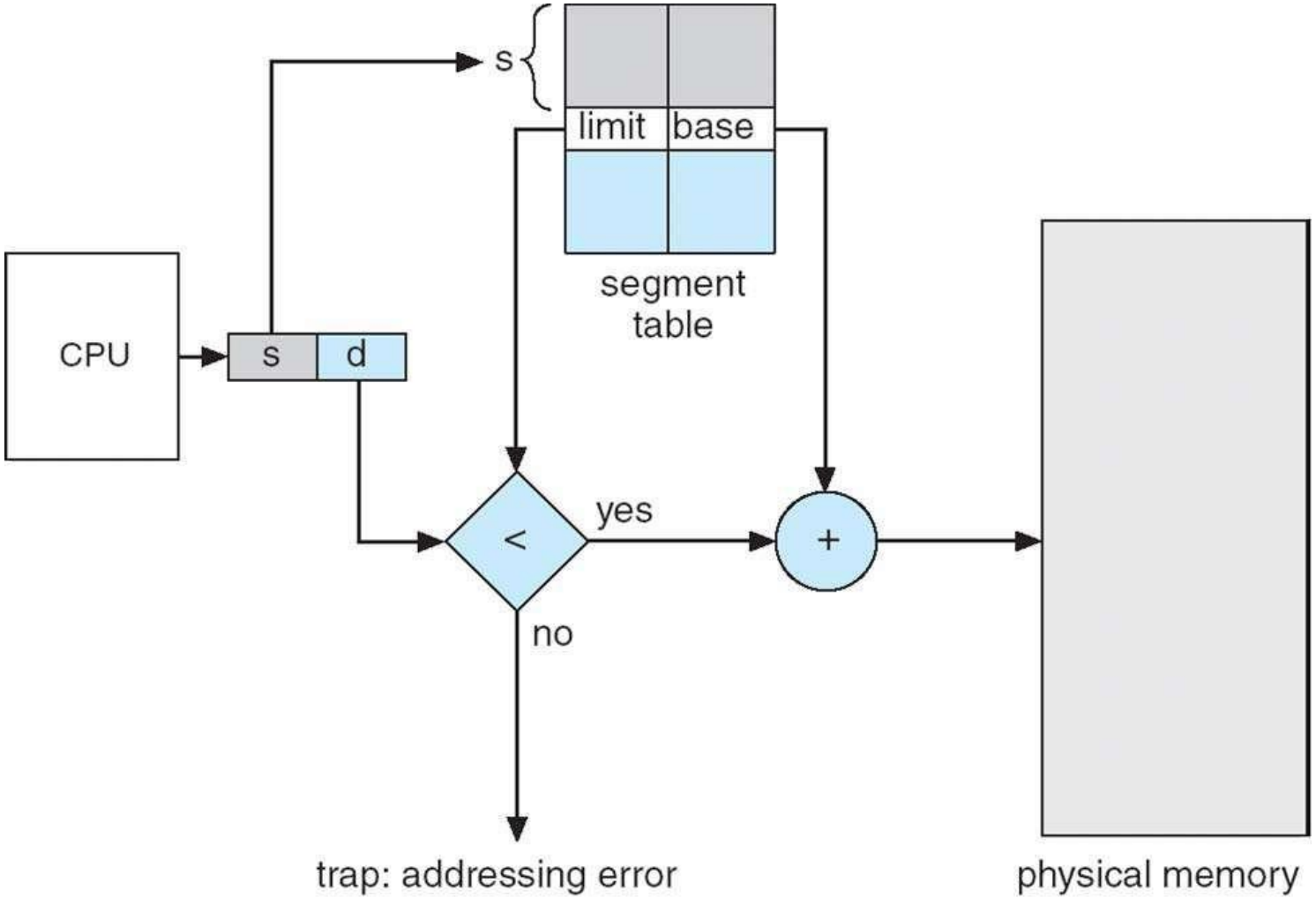


	limit	base
0	1000	1400
1	400	6300
2	400	4300
3	1100	3200
4	1000	4700

segment table



Segmentation Hardware



Protection and Sharing

- Segmentation lends itself to the implementation of protection and sharing policies
- Each entry has a base address and length so inadvertent memory access can be controlled
- Sharing can be achieved by segments referencing multiple processes
- Two processes that need to share access to a single segment would have the same segment name and address in their segment tables.

Disadvantages of Segmentation

- External fragmentation.
- Costly memory management algorithm
- Unequal size of segments is not good in the case of swapping.