



PES
UNIVERSITY
ONLINE

Nitin V Pujari
Faculty, Computer Science
Dean - IQAC, PES University

OPERATING SYSTEMS

Memory Management -3

OPERATING SYSTEMS

Course Syllabus - Unit 3



Unit-3: Unit 3: Memory Management: Main Memory

Hardware and control structures, OS support, Address translation, Swapping, Memory Allocation (Partitioning, relocation), Fragmentation, Segmentation, Paging, TLBs context switches

Virtual Memory - Demand Paging, Copy-on-Write, Page replacement policy - LRU (in comparison with FIFO & Optimal), Thrashing, design alternatives - inverted page tables, bigger pages.

Case Study: Linux/Windows Memory

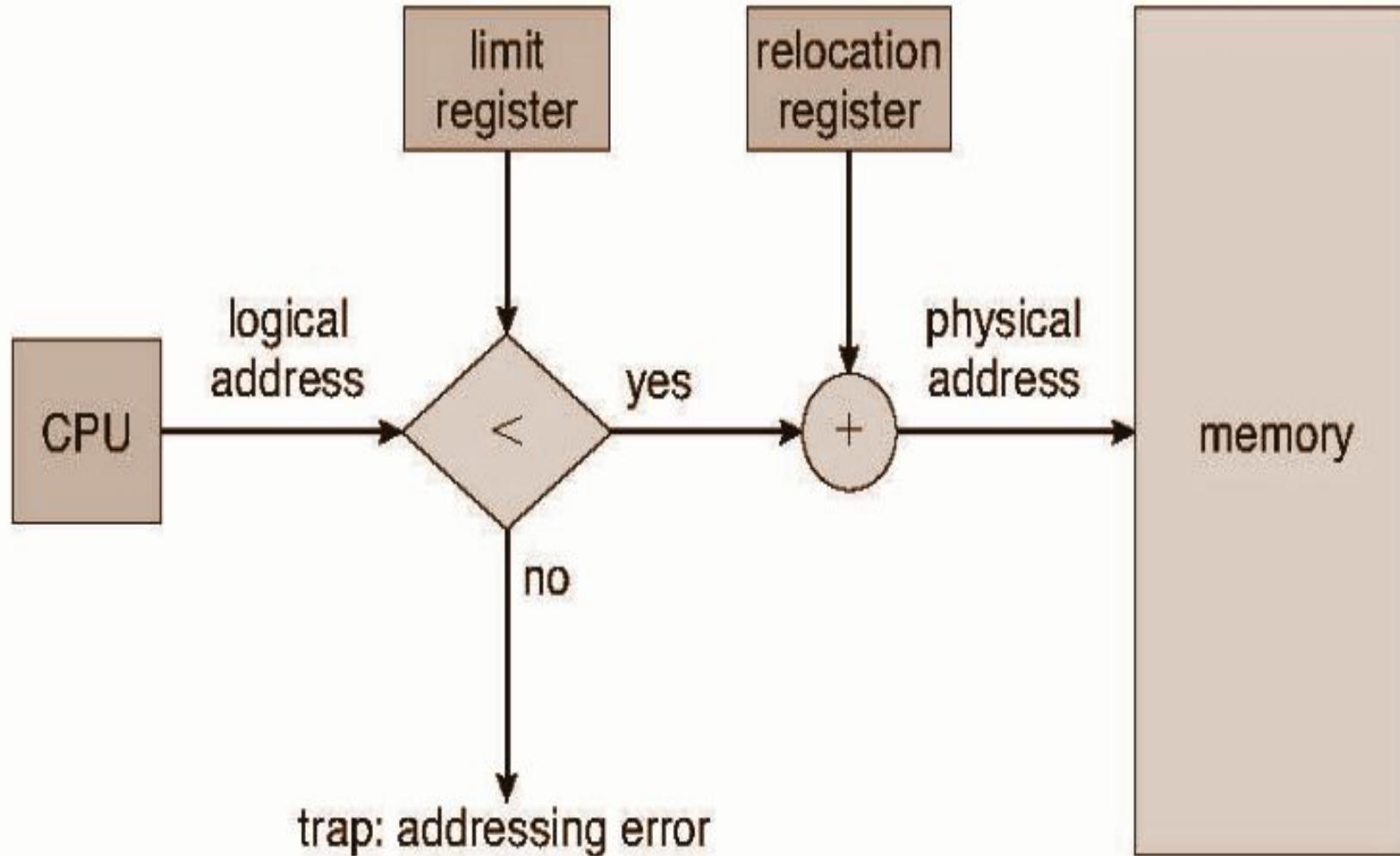
OPERATING SYSTEMS

Course Outline

25	Main Memory: Hardware and control structures, OS support, Address translation	8.1	64.2
26	Dynamic linking, Swapping	8.2	
27	Memory Allocation (Partitioning, relocation), Fragmentation	8.3	
28	Segmentation	8.4	
29	Paging: OS Support, TLBs, Address Translation	8.5	
30	Structure of the Page Table	8.6	
31	Design Alternatives - Inverted Page Tables, Bigger Pages	8.7-8.8	
32	Virtual Memory: Demand Paging, Copy-OnWrite	9.1-9.3	
33	Page replacement policy - LRU	9.4	
34	FIFO & Optimal	9.5	
35	Thrashing	9.6	
36	Case Study: Linux/ Windows Memory Management	9.10	

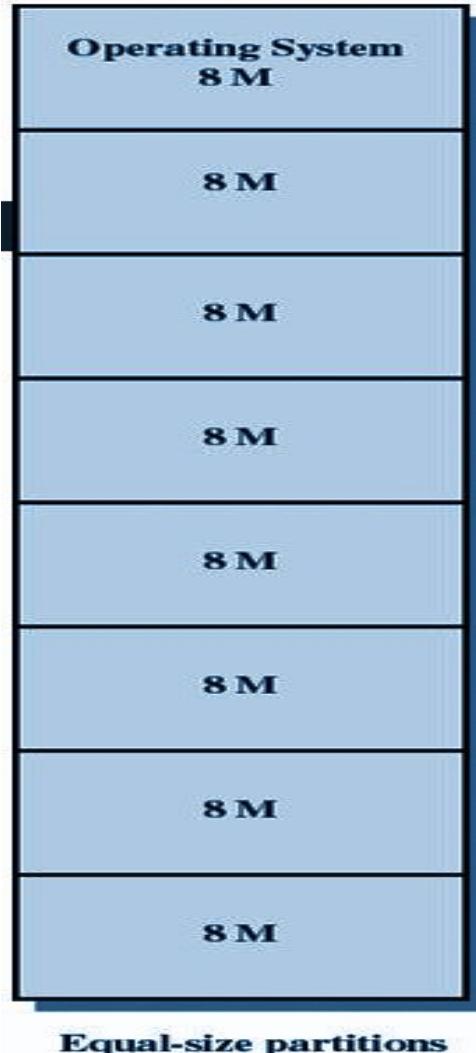
- Contiguous Allocation
- Hardware Support for Relocation and Base Register
- Multiple Partition Allocation
- Fragmentation

- Relocation registers used to protect user processes from each other, and from changing operating-system code and data
- Base register contains value of smallest physical address
- Limit register contains range of logical addresses => each logical address must be less than the Limit register
- MMU maps logical address dynamically
- It can then allow actions such as kernel code being transient and kernel changing size

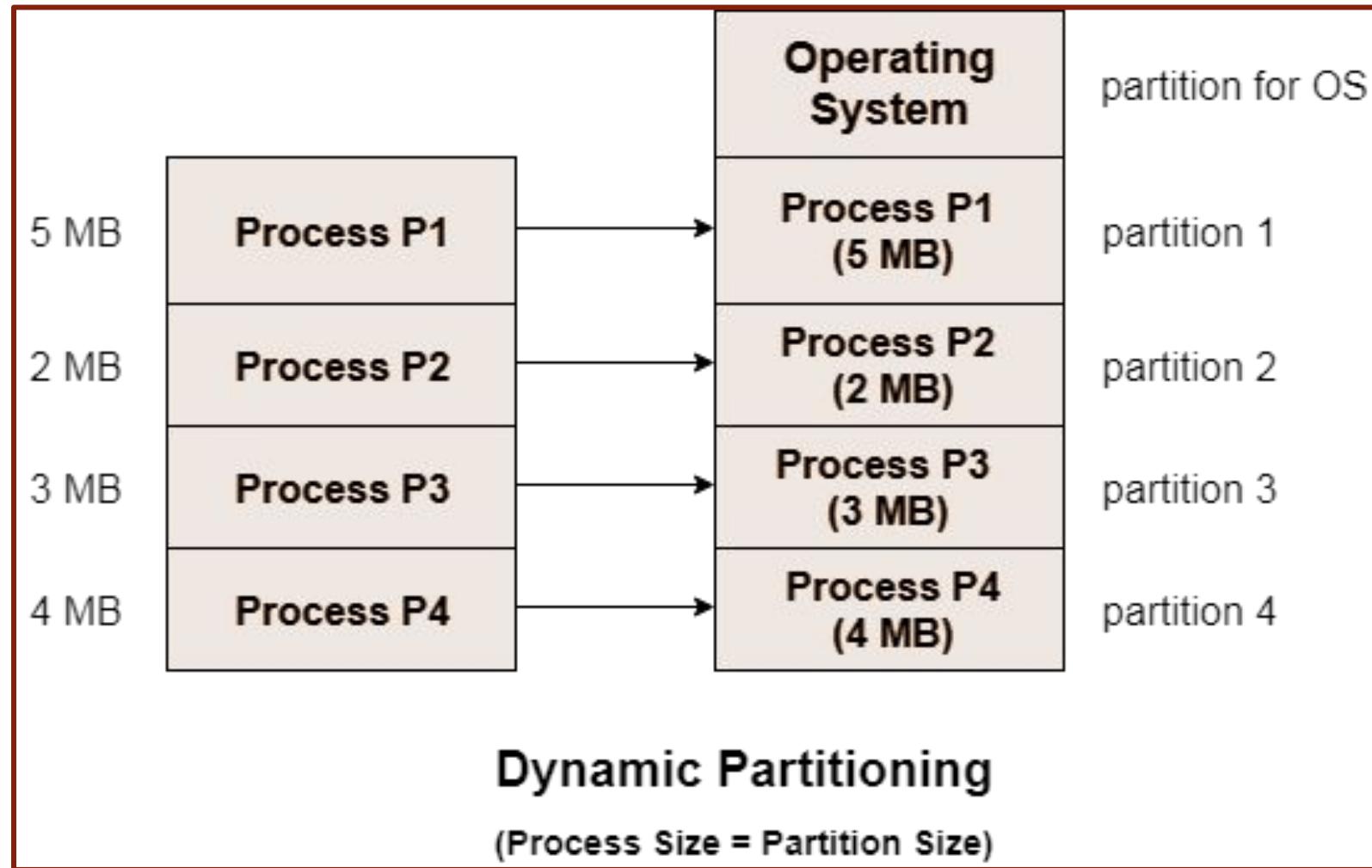


- Multiple-partition allocation
 - Degree of multiprogramming limited by number of partitions - Fixed or Variable
 - Variable Partition sizes for efficiency (sized to a given process' needs)
 - Hole or Contiguous Chunk (CC) – block of available memory; CC of various size are scattered throughout memory

Multiple Partition Allocation - Static or Fixed Partitions

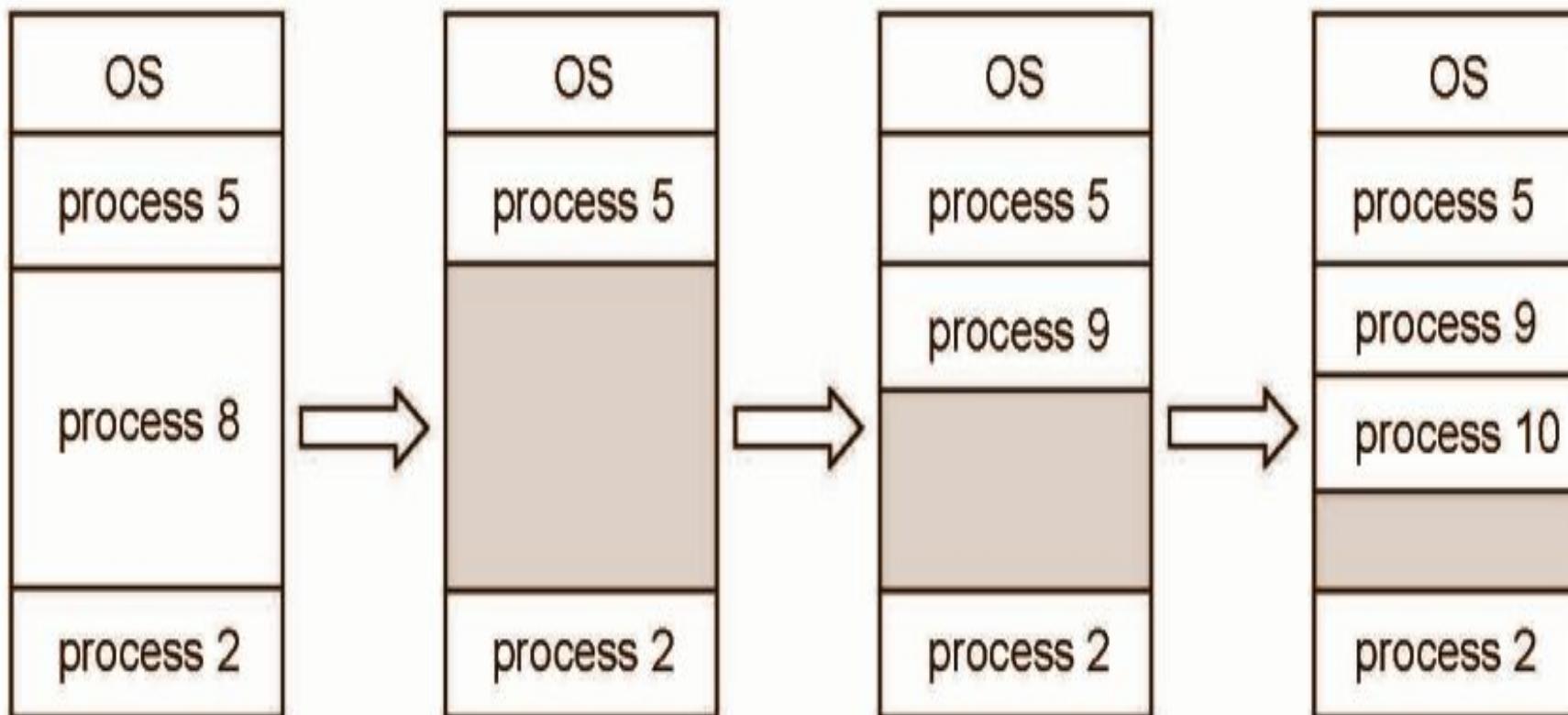


- Dynamic or Variable - Partition allocation



- When a process arrives, it is allocated memory from a free CC large enough to accommodate it
- Process exiting frees its partition, adjacent free partitions combined
- Operating system maintains information about:
 - allocated partitions
 - free CC or partitions

Multiple Partition Allocation - Dynamic or Variable Partitions



How to satisfy a request of size n from a list of free CC ?

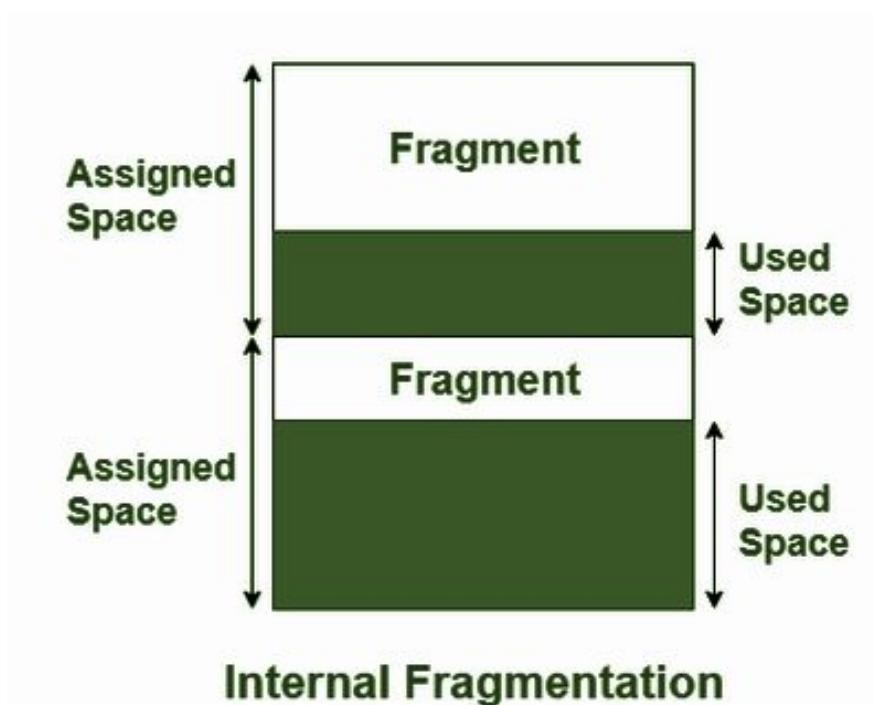
- **First-fit:** Allocate the first CC that is big enough to accomodate the request of size n
- **Best-fit:** Allocate the smallest CC that is big enough; must search entire list, unless ordered by size. It produces the smallest leftover CC
- **Worst-fit:** Allocate the largest hole; must also search entire list. It produces the largest leftover hole

Multiple Partition Allocation - Additional Input

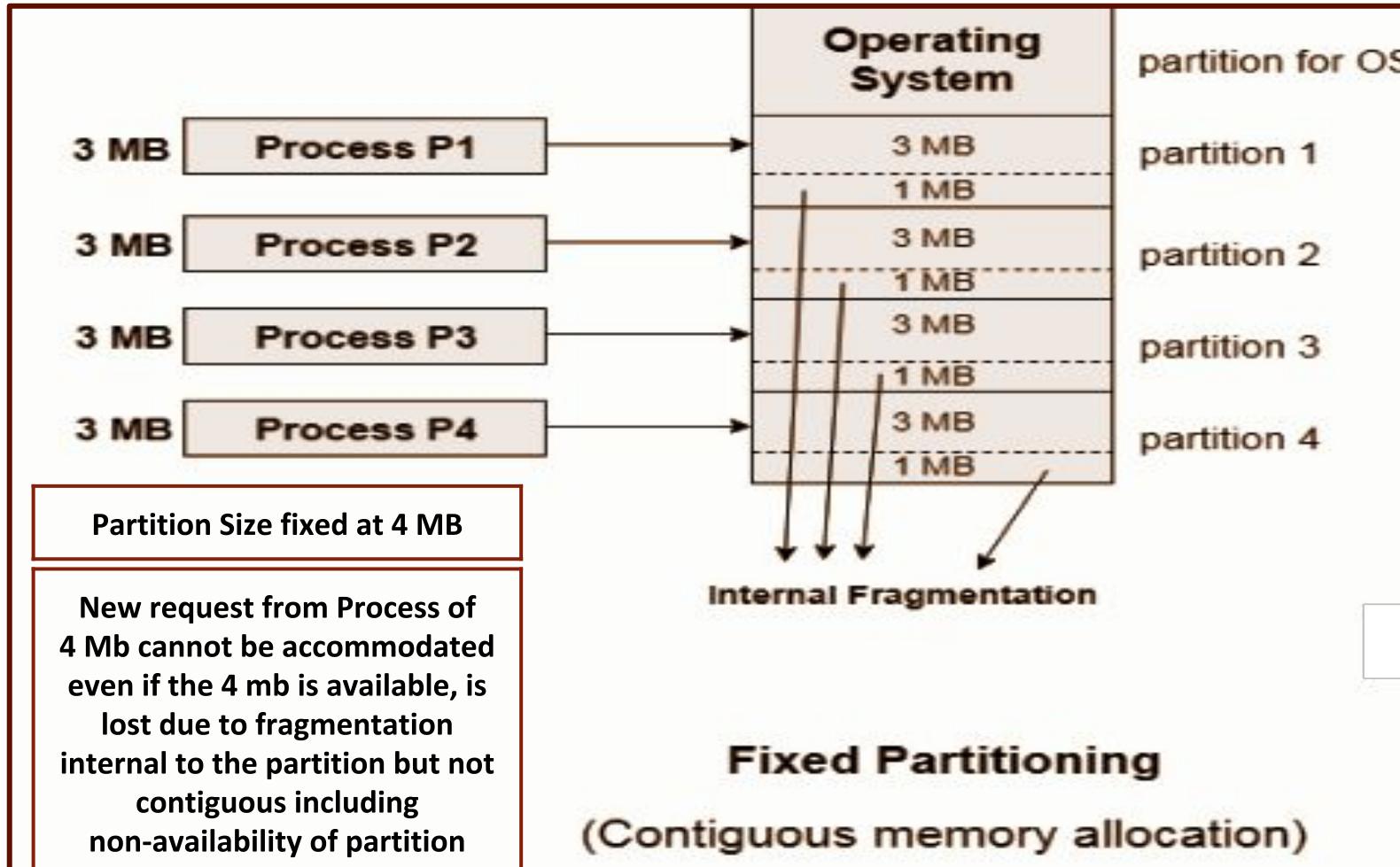
- **Buddy System:** In buddy system, sizes of free blocks are in form of integral power of 2.
- Example: 2, 4, 8, 16 etc. Up to the size of memory.
- When a free block of size $2k$ is requested, a free block from the list of free blocks of size $2k$ is allocated.
- If no free block of size $2k$ is available, the block of next larger size, $2k+1$ is split in **two halves** called **buddies** to satisfy the request.
- **Next fit:** Next fit is a modified version of first fit. It begins as first fit to find a free partition. When called next time it starts searching from where it left off, not from the beginning.

- **Fragmentation** => is an unwanted problem where the memory blocks cannot be allocated to the processes due to their small size and the blocks remain unused.
- It can also be understood as when the processes are loaded and removed from the memory they create free space or hole in the memory and these small blocks cannot be allocated to new upcoming processes and results in inefficient use of memory.

- **Internal Fragmentation** => allocated memory may be slightly larger than requested memory; this size difference is memory internal to a partition, but not being used



- Static or Fixed - Partition allocation



- **External Fragmentation** => total memory space exists to satisfy a request, but it is not contiguous : New Request of 426K must wait

100 Kb	Free
417 Kb	Occupied
88 Kb	Free
200 Kb	Free
300 Kb	Free
212 Kb	Occupied
112 Kb	Occupied
276 Kb	Free

Internal Fragmentation	External Fragmentation
In internal fragmentation, fixed-sized memory blocks square measure appointed to process.	In external fragmentation, variable-sized memory blocks square measure appointed to method.
The solution of internal fragmentation is best-fit block.	Solution of external fragmentation is compaction, paging and segmentation.

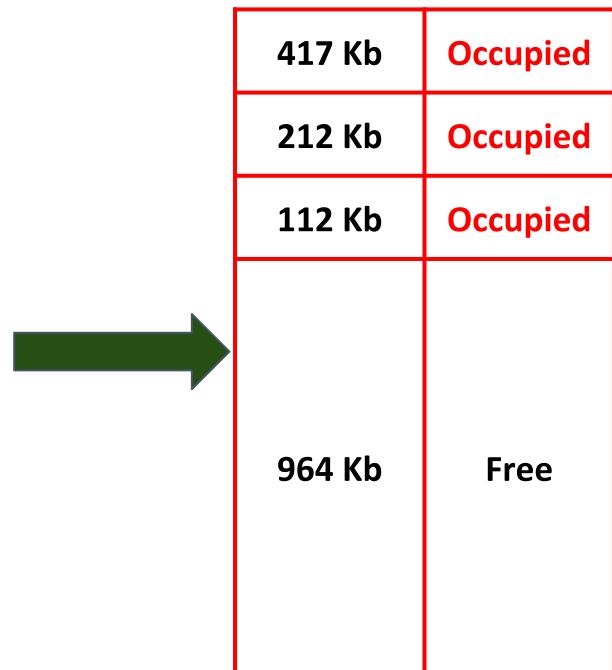
- **First Fit** analysis reveals that given N blocks allocated, $0.5 N$ blocks lost to fragmentation
 - $1/3$ may be unusable -> 50-percent rule

Fragmentation

- Reduce external fragmentation by **Compaction**
- Shuffle memory contents to place all free memory together in one large block
- Compaction is possible only if relocation is dynamic, and is done at execution time
 - I/O problem
 - Latch job in memory while it is involved in I/O
 - Do I/O only into OS buffers

- **Compaction - Example**
- Shuffle memory contents to place all free memory together in one large block. New Request of 426K

100 Kb	Free
417 Kb	Occupied
88 Kb	Free
200 Kb	Free
300 Kb	Free
212 Kb	Occupied
112 Kb	Occupied
276 Kb	Free



Compaction in Progress

417 Kb	Occupied
212 Kb	Occupied
112 Kb	Occupied
426 Kb	Occupied
538 Kb	Free

New request of 426 Kb
Accommodated after
compaction

Multiple Partition Allocation - Example

- Given five memory partitions of 100Kb, 500Kb, 200Kb, 300Kb, 600Kb (in order), how would the first-fit, best-fit, and worst-fit algorithms place processes of 212 Kb, 417 Kb, 112 Kb, and 426 Kb (in order) ? Which algorithm makes the most efficient use of memory?

First-fit:

212K is put in 500K partition

417K is put in 600K partition

112K is put in 288K partition (new partition 288K = 500K - 212K)

426K must wait

Best-fit:

212K is put in 300K partition

417K is put in 500K partition

112K is put in 200K partition

426K is put in 600K partition

In this example, best-fit turns out to be the best.

Worst-fit:

212K is put in 600K partition

417K is put in 500K partition

112K is put in 388K partition

426K must wait

Multiple Partition Allocation - Example

First-fit:

212K is put in 500K partition

417K is put in 600K partition

112K is put in 288K partition (new partition 288K = 500K - 212K)

426K must wait

212 Kb	417 Kb	112 Kb	426 Kb
--------	--------	--------	--------

100 Kb	Free
500 Kb	Free
200 Kb	Free
300 Kb	Free
600 Kb	Free

100 Kb	Free
212 Kb	Occupied
288 Kb	Free
200 Kb	Free
300 Kb	Free

100 Kb	Free
212 Kb	Occupied
288 Kb	Free
200 Kb	Free
300 Kb	Free

100 Kb	Free
212 Kb	Occupied
112 Kb	Occupied
176 Kb	Free
200 Kb	Free

100 Kb	Free
300 Kb	Free
417 Kb	Occupied
183 Kb	Free
417 Kb	Occupied

100 Kb	Free
183 Kb	Free

First-fit

Multiple Partition Allocation - Example

Best-fit:

212K is put in 300K partition

417K is put in 500K partition

112K is put in 200K partition

426K is put in 600K partition

212 Kb		417 Kb		112 Kb		426 Kb	
100 Kb	Free	100 Kb	Free	100 Kb	Free	100 Kb	Free
500 Kb	Free	500 Kb	Free	417 Kb	Occupied	417 Kb	Occupied
200 Kb	Free	200 Kb	Free	83 Kb	Free	83 Kb	Free
300 Kb	Free	212 Kb	Occupied	200 Kb	Free	112 Kb	Occupied
600 Kb	Free	88 Kb	Free	212 Kb	Occupied	88 Kb	Free
		600 Kb	Free	88 Kb	Free	212 Kb	Occupied
Best-fit				600 Kb	Free	88 Kb	Free
						426 Kb	Occupied
						174 Kb	Free

Multiple Partition Allocation - Example

Worst-fit:

212K is put in 600K partition

417K is put in 500K partition

112K is put in 388K partition

426K must wait

212 Kb		417 Kb		112 Kb		426 Kb	
100 Kb	Free	100 Kb	Free	100 Kb	Free	100 Kb	Free
500 Kb	Free	500 Kb	Free	417 Kb	Occupied	417 Kb	Occupied
200 Kb	Free	200 Kb	Free	88 Kb	Free	88 Kb	Free
300 Kb	Free	300 Kb	Free	200 Kb	Free	200 Kb	Free
600 Kb	Free	212 Kb	Occupied	300 Kb	Free	300 Kb	Free
		388 Kb	Free	212 Kb	Occupied	212 Kb	Occupied
				388 Kb	Free	112 Kb	Occupied
						276 Kb	Free

Worst-fit



THANK YOU

**Nitin V Pujari
Faculty, Computer Science
Dean - IQAC, PES University**

nitin.pujari@pes.edu

For Course Deliverables by the Anchor Faculty click on www.pesuacademy.com and complete reading assignments provided by the Anchor on Edmodo