OPERATING SYSTEM (315319)

UNIT IV

MEMORY MANAGEMENT

Unit Scope

- Basic memory management: fixed and variable partitioning
- Free-space management: bitmap and linked list
- Swapping, compaction and fragmentation
- First Fit, Best Fit and Worst Fit
- Paging and segmentation
- Virtual memory, demand paging and page fault
- FIFO, LRU and Optimal page replacement

Learning Outcomes

TLO	Expected capability
4.1	Compare fixed and variable memory partitioning.
4.2	Differentiate bitmap and linked-list techniques.
4.3	Explain working of partitioning algorithms.
4.4	Calculate page faults for a given page-reference string.
Assessment weightage Unit IV carries 16 marks and 12 learning hours in the syllabus; R = 2, U = 6, A = 8.	

1. Basic Memory Management

- Memory management tracks which memory locations are allocated or free.
- It decides how much memory is allocated, which process receives memory and when released space is updated.
- The supplied notes divide techniques into contiguous and non-contiguous memory management.

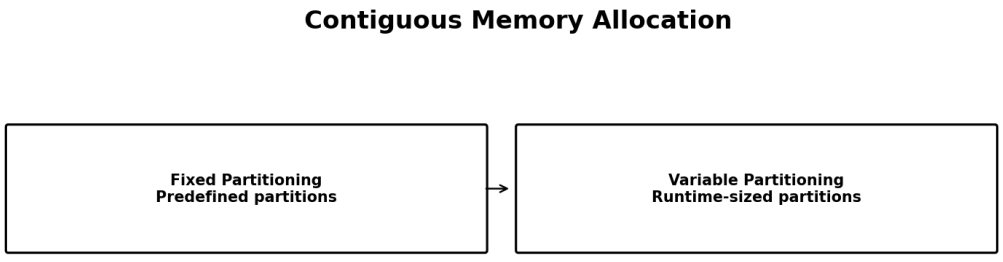

Fig. 4.1 Contiguous allocation: fixed vs variable partitioning

2. Fixed Partitioning

- The number of non-overlapping partitions is fixed before execution or during system configuration.
- Each partition contains one process; spanning is not allowed in this model.
- Partition sizes may be equal or unequal.
- Unused space inside an allocated partition is internal fragmentation.

Advantages	Disadvantages
Easy to implement	Internal fragmentation
Little OS overhead	Process-size limitation
Simple tracking	Degree of multiprogramming limitation

3. Variable Partitioning

- Partitions are created at run time according to process requirements.
- Partition size is matched to process need, reducing internal fragmentation.
- The number of partitions varies with processes and available memory.
- External fragmentation can occur as processes enter and leave memory.

Advantages	Disadvantages
No internal fragmentation in the basic model	Difficult implementation
Flexible degree of multiprogramming	External fragmentation
Flexible process size	Requires free-hole management

Exam Focus Compare fixed and variable partitioning using at least four points.

4. Free-Space Management

- The OS maintains information about free space so released blocks can be reused.
- The supplied notes describe bitmap and linked-list approaches.

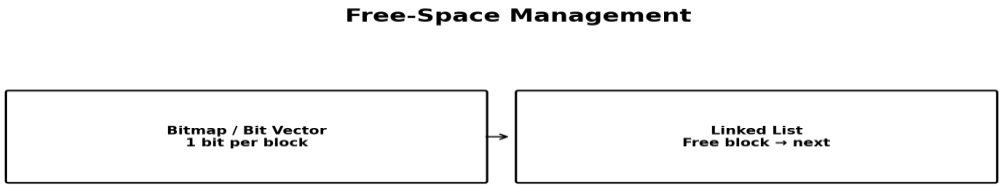

Fig. 4.2 Bitmap and linked-list free-space management

Technique	Working	Key point
Bitmap / Bit Vector	One bit corresponds to each block; source uses 0 = allocated and 1 = free.	Compact representation.
Linked List	Free blocks are linked by pointers.	Traversal may require I/O.

Source example The supplied notes give the 16-bit bitmap 0000111000000110, with 1 indicating a free block.

5. Swapping

- Swapping moves a process between main memory and secondary storage to manage limited RAM.
- A process may be swapped out to free memory and later swapped back in.
- It supports multiprogramming but introduces transfer overhead.

6. Compaction

- Compaction combines scattered free holes into a larger contiguous block.
- It is useful for reducing external fragmentation in variable partitioning.
- Relocation support and processing time are required.

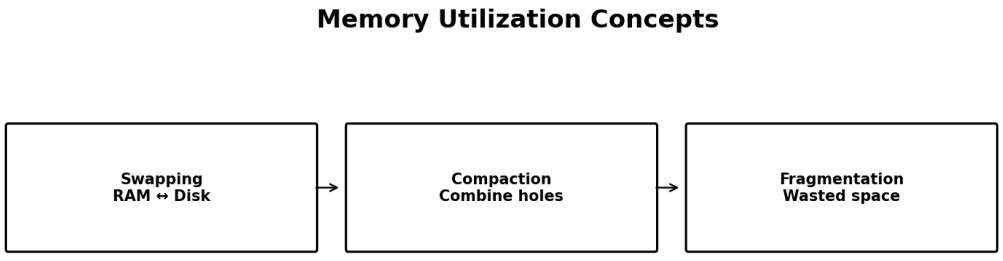

Fig. 4.3 Swapping, compaction and fragmentation

7. Fragmentation

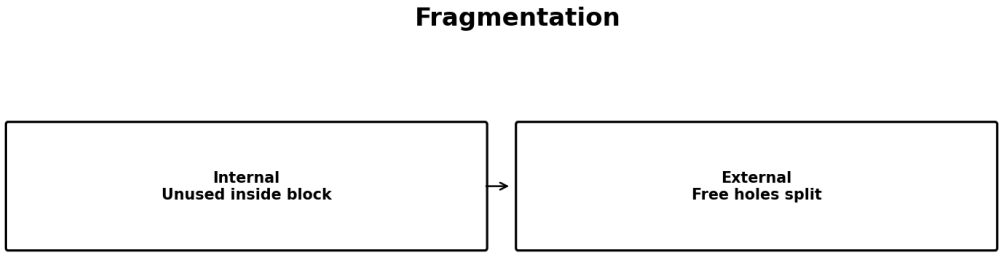

Fig. 4.4 Internal and external fragmentation

Type	Meaning	Typical association
Internal	Unused space inside an allocated block.	Fixed-size partitions / paging
External	Free space is available but split into non-contiguous holes.	Variable-size contiguous allocation / segmentation

8. Partitioning Algorithms

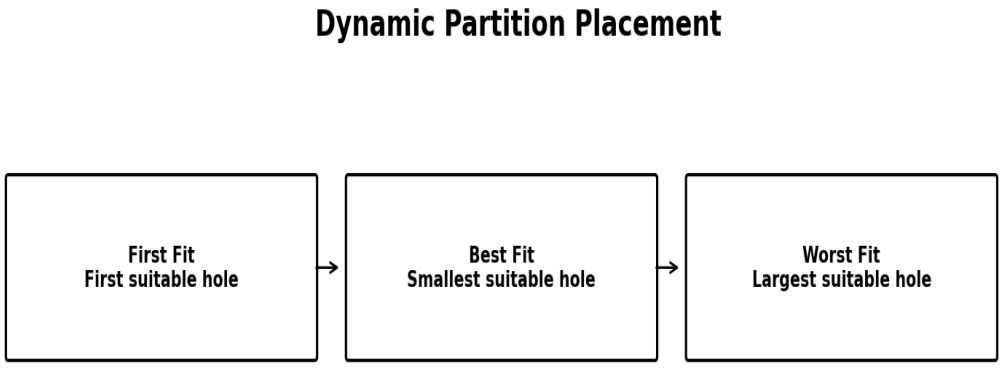

Fig. 4.5 First Fit, Best Fit and Worst Fit

Algorithm	Selection rule	Key point
First Fit	First hole large enough.	Simple and generally quick.
Best Fit	Smallest hole large enough.	May create many small leftover holes.
Worst Fit	Largest hole available.	Leaves a relatively large remainder.

8.1 Worked Example

Free holes	100 KB	500 KB	200 KB	300 KB	600 KB
Process	212 KB	417 KB	112 KB	426 KB	—

- First Fit: 212→500; 417→600; 112→remaining 288; 426 cannot fit.
- Best Fit: 212→300; 417→500; 112→200; 426→600.
- Worst Fit: 212→600; 417→500; 112→remaining 388; 426 cannot fit.

Supplementary example This numerical First/Best/Worst Fit example is teaching enrichment aligned with the syllabus; the uploaded notes do not contain a numerical fit example.

9. Non-contiguous Memory Management

- Non-contiguous techniques allow parts of a process to occupy different physical memory locations.

10. Paging

- Paging eliminates the need for contiguous physical allocation.
- Physical memory is divided into fixed-size frames; processes are divided into fixed-size pages.
- Pages can be placed in available frames.
- Logical address = page number (p) + page offset (d). Physical address = frame number (f) + frame offset (d).

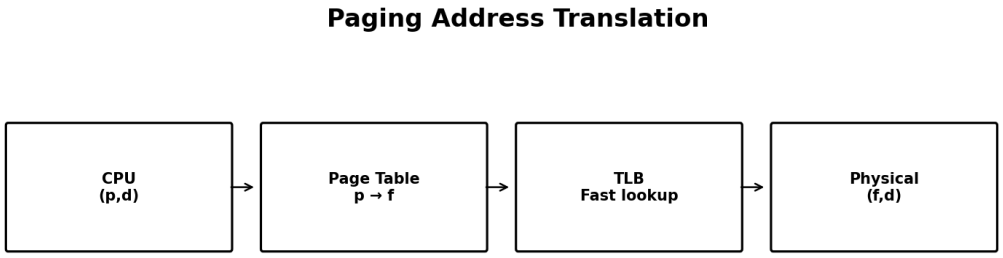

Fig. 4.6 Paging address translation

10.1 Page Table and TLB

Component	Function
Page Table	Maps page number to the frame containing that page.
PTBR	Contains the base address of the page table.
TLB	Small, fast associative lookup cache.
PTE	May contain frame number, present/absent, protection, reference and dirty bits.
Advantages	**Disadvantages**
Non-contiguous process storage	Internal fragmentation
Solves external fragmentation	Page-table overhead
Supports virtual memory	Address-translation overhead

11. Segmentation

- A process is divided into variable-size logical segments representing related program modules.
- Each segment occupies a contiguous block, while different segments can be placed non-contiguously overall.
- Logical address = segment number (s) + offset (d).
- Segment table stores base address and limit.

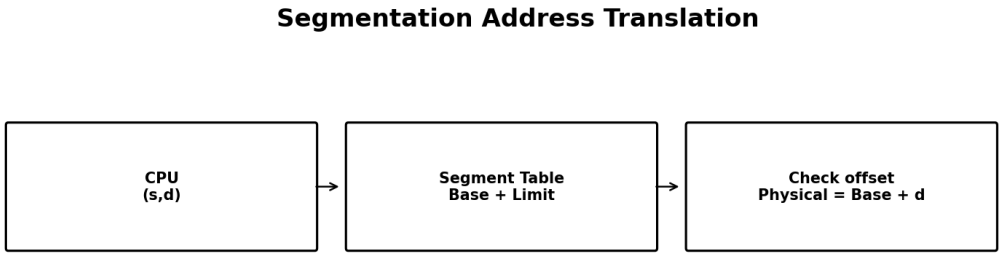

Fig. 4.7 Segmentation address translation

Feature	Paging	Segmentation
Size	Fixed-size	Variable-size
Table	Page → frame	Base + limit
Fragmentation	Internal possible	External possible
Address	p + d	s + d

12. Virtual Memory

- Virtual memory allows secondary memory to be addressed as though it were part of main memory.
- Logical/virtual addresses are translated to physical addresses at run time.
- Not all pages or segments need to be present in main memory simultaneously.
- The supplied notes state demand paging and demand segmentation as implementation approaches.

Key Idea Only required pages/segments need to be loaded into main memory when required.

13. Demand Paging and Page Fault

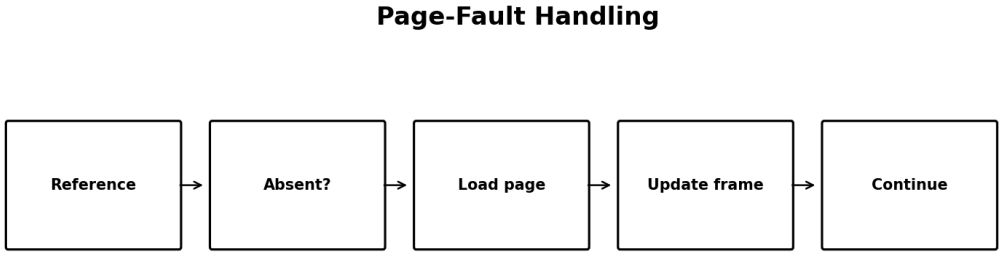

Fig. 4.8 Page-fault handling

- A page fault occurs when a process references a page not currently in physical memory.
- The OS locates the page in secondary storage and loads it into a suitable frame.
- Execution continues after the missing page is made available.

14. Page Replacement Algorithms

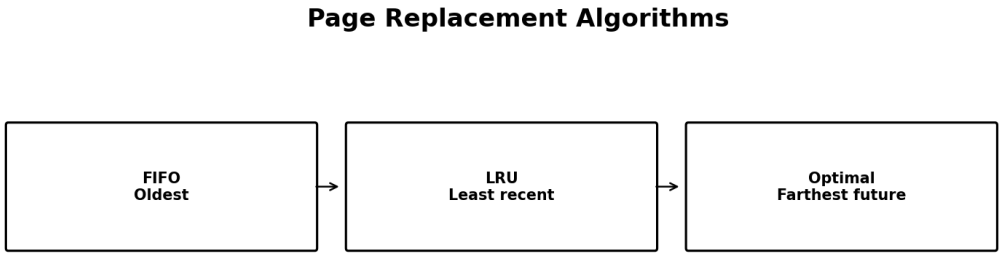

Fig. 4.9 FIFO, LRU and Optimal

Algorithm	Rule	Observation
FIFO	Replace the oldest page.	Simple; may suffer Belady's anomaly.
LRU	Replace the least recently used page.	Uses recent-reference history.
Optimal	Replace the page whose next use is farthest in the future.	Theoretical benchmark; future is not known in practice.

15. Page-Fault Calculation Workflow

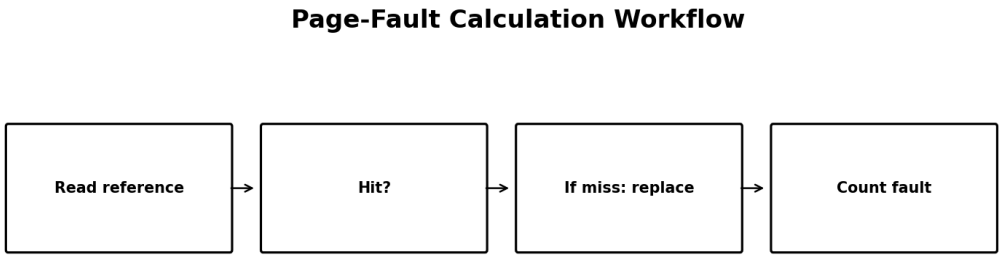

Fig. 4.10 Page-fault calculation workflow

- Write the reference string and frame count.
- Process references from left to right.
- Mark Hit if page is already present.
- For a miss, use an empty frame or replace according to the selected algorithm.
- Count all misses to obtain total page faults.

16. Worked / Practice Examples from the Source

- Reference string: 7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2; 4 frames.

Algorithm	Page faults reported in source
Optimal	6
LRU	6
FIFO	6

- Second source practice: 4, 7, 6, 1, 7, 6, 1, 2, 7, 2; 3 frames.

Algorithm	Page faults reported in source
Optimal	5
LRU	6
FIFO	6

17. Source Calculation Quality Check

Important The source reports "6/14 = 0.85" for one page-fault ratio. Arithmetically, 6/14 ≈ 0.43. The source's page-fault count is retained, while any ratio should be recalculated from the actual reference-string length before final publication.

- The source also contains typographical wording such as "Fage Fault"; this has been cleaned in the publishing version.

18. Quick Revision

Topic	Remember
Fixed partition	Fixed number; internal fragmentation.
Variable partition	Runtime-sized; external fragmentation.
Bitmap	One bit per block; 0=allocated, 1=free in source.
Linked list	Free blocks linked by pointers.
Swapping	Process moved between RAM and secondary storage.
Compaction	Combine scattered holes.
First Fit	First suitable hole.
Best Fit	Smallest suitable hole.
Worst Fit	Largest suitable hole.
Paging	Fixed pages/frames; page → frame.
Segmentation	Variable segments; base + limit.
Virtual memory	Logical address space supported using secondary storage.
Page fault	Referenced page absent from RAM.
FIFO	Oldest page.
LRU	Least recently used.
Optimal	Farthest future use.

19. Question-Paper Mapping: Unit IV

Paper	Mapped Unit IV questions
Summer 2026	Q1(e) fragmentation; Q1(f) virtual memory; Q3(c) paging vs segmentation; Q4(d) bitmap; Q5(c) paging hardware; Q6(b) FIFO/Optimal page faults.
Winter 2025	Q1(g) page fault; Q2(b) variable partitioning; Q2(c) bitmap/linked list; Q4(a) First/Best/Worst Fit; Q4(c) paging vs segmentation; Q6(c) FIFO/Optimal page faults.
Past-paper observation Across the two supplied papers, Unit IV generated 13 sub-question opportunities, the highest among the five units. This is a past-paper observation, not a prediction.	

20. Exam Checklist

- Compare fixed and variable partitioning.
- Differentiate bitmap and linked-list techniques.
- Explain swapping, compaction and fragmentation.
- Explain First Fit, Best Fit and Worst Fit with examples.
- Explain paging and paging hardware.
- Differentiate paging and segmentation.
- Define virtual memory, demand paging and page fault.
- Calculate page faults using FIFO, LRU and Optimal.
- Explain page table, PTBR, PTE and TLB.