(Operating Systems: 22516)

Unit V: Memory Management
(Weightage: 14 marks)

— Pranjal Sane

5.1

Q. Free space management Techniques (6 marks / 4 marks)

⇒ **Free space management (02)**

A file system is responsible to allocate free blocks to the file therefore it has to keep track of all the free blocks present in disk.

Free space management is critical aspect of operating system as it involves managing the available storage space on the hard disk or secondary storage devices.

The operating system uses various techniques to manage free space and optimize the use of storage device.

List of free space management techniques. (02)

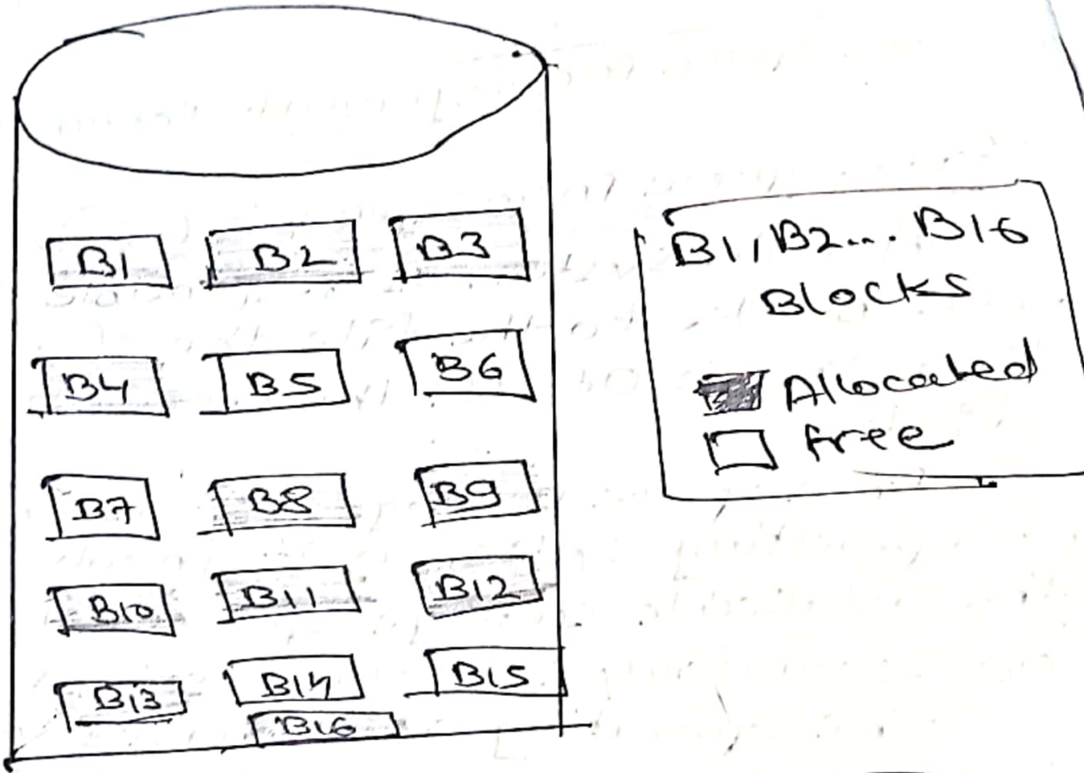
- Bit vector or Bitmap
- Linked List
- Grouping
- Counting

1) Bit vector or Bitmap (02)

- The free space list is implemented as bitmap or bit vector.
- It is series of collection of bits where each bit corresponds to disk block.
- The bit can take two value 0 & 1.
- If block is free 0 & if block is allocated 1

For example,

The given instance of disk blocks on disk allocated in Figure. which are colored are 1111000111111001 can be represented by



Advantages

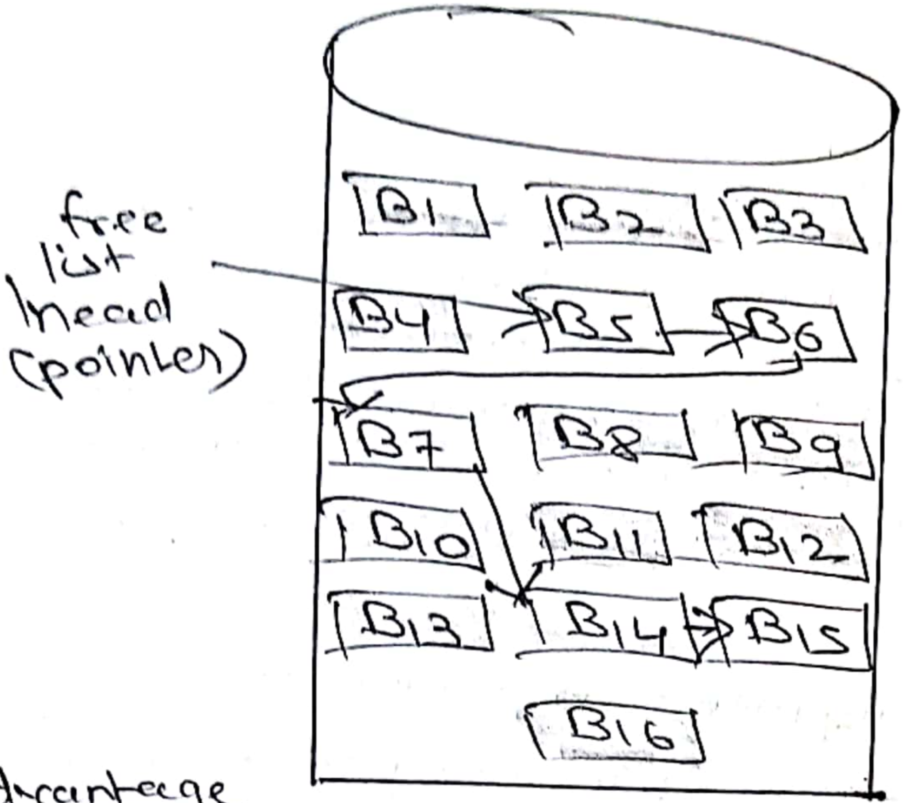
- Easy to understand & implement
- Efficient to find first free block by scanning the non-zero word, then identifying first bit

DisAdvantage

- finding free blocks may become slower as it requires scanning of blocks
- more memory space if the storage or disk is large

2) Linked-List (02)

- In this approach, the free disk blocks are linked together i.e a free block contains pointer to next free block.
- The block number of very first disk block is stored at seperate location on disk and is also cached in memory.
- In this approach, link all the disk blocks together, keeping a pointer to first free block.
- This block contains a pointer to next free disk block and so on.



Advantage

- Total available space is efficiently used in method.
- Dynamic allocation in Linked List is easy.

DisAdvantage

- As Linked List size increase, headache of memorizing pointers increases.
- Not efficient while iteration of each block.

Introduction

Memory Management

Memory management is how the operating system controls and divides memory between different programs running on computer.

It ensures that the main memory is used efficiently and helps manage the transfer of data between the memory and disk while programs are running.

Functions of Memory ~~Mem~~ Management

1) Keep tracking of every memory location.
2) Track of whether memory is utilize properly.
3) Track how much memory is allocated.
4) Track of whether memory is allocated or not.
5) Takes decision which process will get memory and when.

Partitioning in Memory Management

Partitioning in main memory management is how a computer's memory is divide to run multiple programs.

The main memory (RAM) is split into sections, and each section holds one program.

There are two methods

→ Non-Contiguous memory allocation.

→ Contiguous memory allocation

Contiguous memory allocation

- Each program gets one big, continuos block of memory.
- The entire program must be in memory to run.
- Contiguoes memory can be divided into.

Fixed Partitioning | Variable Partitioning

Fixed Partitioning (Static)

Fixed partitioning divides memory into a set number or fixed-size blocks.

Each block holds one process, and the partition sizes are set when system starts and do not change.

Key points

- Each partition has a fixed size.
- If a process doesn't use the entire partition the unused space is wasted (called internal fragementation).
- The numbers of partitions limits how many processes can run at same time.

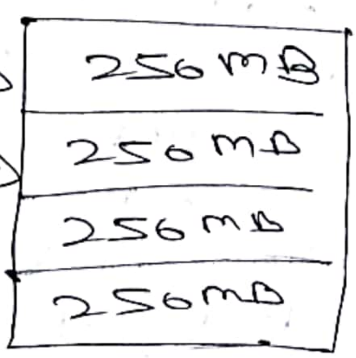
Example

Computer has 1024 MB
divided into 256MB (4 part)

Process A = 100MB
(156 MB unused)

Process B = 200MB
(56MB unused)

Process C = 300MB ✗
(cant run because larger enough, even though total free memory)



Variable Partitioning

Variable Partitioning allocate memory dynami based on exact amount a process needs.
Unlike fixed partitioning, the memory is not divided into fixed blocks. instead the System creates partitions as processes arrive and frees them when they finish.

Key points :

- Partition size vary depending on the process need.
- Memory is used more efficiently because it allocate exactly what's required.
- No internal fragmentation, but external fragmentation can happen when small gap form as processes finish and leaves holes in memory.

Example: System 1024 MB Ram

Process X (100MB) allocated, 924 MB free
Process Y (300MB) allocated, 624 MB fre
Process Z (150MB) allocated, 474 MB fre

If Process Y finishes and frees 300 MB, the memory split into two parts: 300 MB of free space & 174 MB elsewhere.
If new process W needs 350 MB it can't be allocated because available memory seperate smaller blocks, not a single large block.
This is external fragmentation

7) First Fit, Best Fit & Worst Fit are methods used by operating system to allocate memory to processes

1) First Fit

Chooses the first available memory block that is big enough for process.

In simple, takes the first memory block big enough to process.

Example, process needs 200MB, available blocks are 300MB & 400MB, it takes 300MB block.

100MB 500MB
200MB 30MB 150MB

2) Best Fit

It choose the smallest block that can fit the process.

Example, Process need: 200MB.
Blocks are 250MB & 400MB,
It takes 250MB

3) Worst Fit

largest block.

200MB (need) 250 & 500

500MB block taken

Questions

Consider the following memory map and assume a new process P4 comes with memory requirements of 6KB.

Locate (Draw) this process in memory

i) First Fit

ii) Best Fit

iii) Worst Fit

O.S
P1
<FREE> 12KB
P2
<FREE> 19KB
P3
<FREE> 7KB

memory

⇒ —6 marks

First Fit

Allocate the first free block to new process P4.

OS
P1
P4 6KB <FREE> 6KB
P2
<FREE> 19KB
P3
<FREE> 7KB

—02—

Best Fit

Allocate smallest free block that is big enough to accomodate new process P4.

OS
P1
<FREE> 12KB
P2
<FREE> 19KB
P3
P4 6KB <FREE> 1KB

—02—

Worst Fit

Allocate largest free block to new process P4

OS
P1
<FREE> 12KB
P2
P4 6KB <FREE> 13KB
P3
<FREE> 7KB

—02—

—06 marks ✓

Scanned with CamScanner

.2 Virtual memory - Introduction to Paging, segmentation, Fragementation, Pagefault.

Questions

Q1 Define Virtual memory (2m)

Q2 Difference between paging and segmentation. (2m/4m).

Q3 Describe concept of virtual memory with respect to paging. Also draw paging hardware diagram and describe its working with example. (6m).

Q4 Define the term fragmentation in terms of memory. (2m).

Q5 Define paging & segmentation. (2m).

Q6 Define fragmentation, Explain Internal & External fragmentation. (4m)

Q7 Define following terms
i) Virtual memory
ii) Paging (2m).
iii) memory compaction

Virtual memory

- Virtual memory lets computer use part of its hardrive to act like extra RAM when physical (RAM) is full.
- It moves data temporarily from RAM to disk storage to free up space.
- Programmers have not to worry about how much RAM the computer has, as virtual memory makes it seem like there's more memory available
- This makes programming easier since developers can focus on coding without worrying about memory limits.

Paging

Paging is memory management scheme that permi the physical address space of process to be non contiguous.

Logical memory is divided into blocks of same size called as pages.

Paging is a storage mechanism used in OS to retrive processes from secondary storage to main memory as pages.

The process of retriveing processes in form of pages from secondary storage into main memory is known as paging.

Segmentation

Segmentation is memory management schem that permits dividing logical address space into multiple segments.

Segmentation divides the computer's physica memory and program address space into segments.

page fault

A Page fault is typically occurs when a process attempt to access memory in virtual address space that is does not own.

Page fault is the condition in which a running process refers to page that is not loaded in memory (main).

Memory compaction

The process of shuffling the memory contents so as to place all free memory together in one large block is known as memory compaction.

Question: Describe the concept of virtual memory with respect to paging. Also draw paging hardware diagram and describe its working with example ... 6m.

Virtual Memory with respect to paging

Paging is memory management technique that divides both physical memory (RAM) and memory into fixed size blocks called pages.

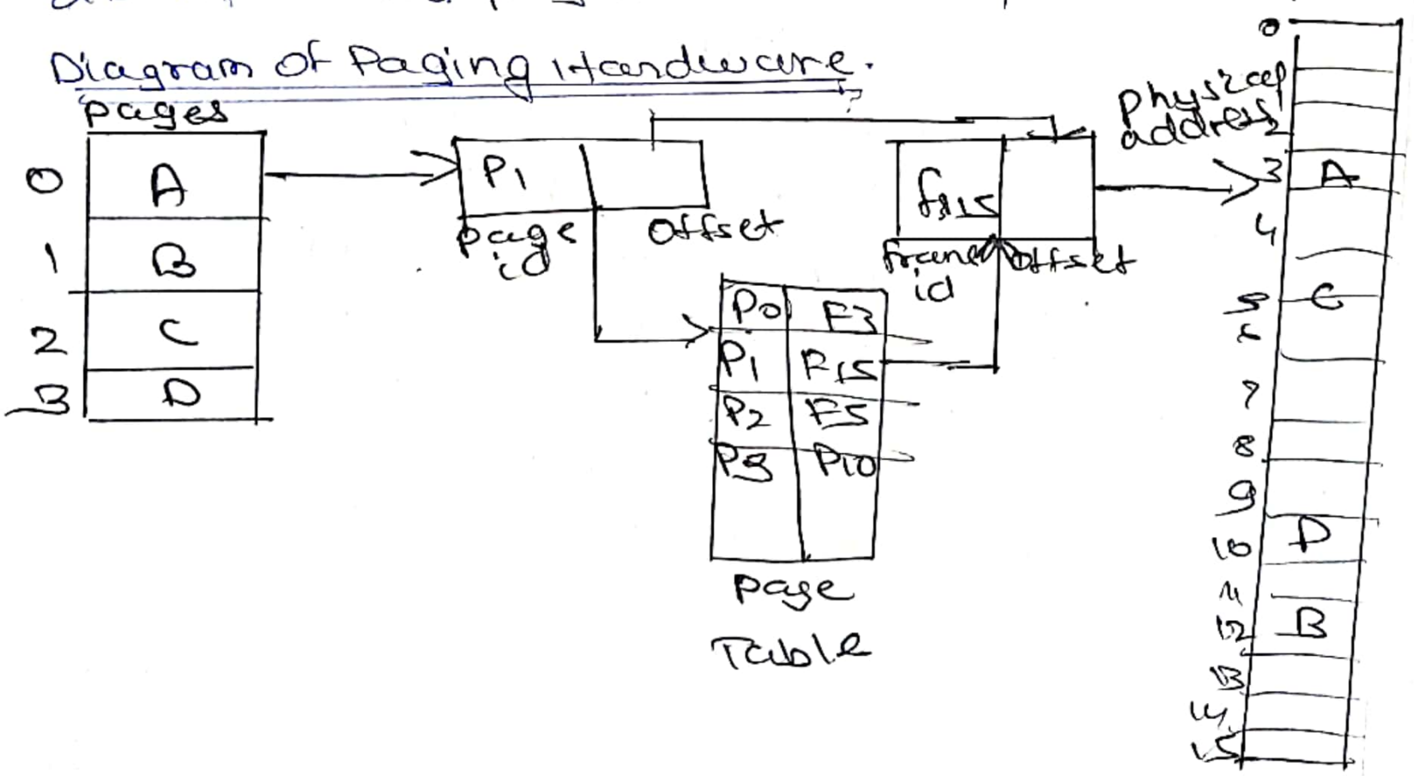
The method allows Operating system to load pages into RAM only as needed, instead of loading the entire program at once.

Paging is key in implementation virtual memory, as it allows for larger programs to run on smaller physical memory.

When a program runs, only the necessary pages (parts of program) are brought into RAM. If page isn't currently in RAM but it needed, a page fault occurs.

The OS will then load that page from disk storage to RAM, allowing the program to continue smoothly without worrying about limited physical memory.

Diagram of Paging Hardware.



How logical address is translated into physical address

1) ~~Doe~~ Logical memory (Pages)

↳ Logical memory divided into fixed-size blocks called pages (A, B, C, D)

↳ Each pages has P|D (P_0, P_1, P_2, P_3) and offset address points specific part.

2) Page Table

↳ Crucial part of paging. It stores mapping between Page ID & Frame ID.

↳ Example P_0 maps F3, P1 maps F15.

3) Physical Memory (Frames)

↳ Physical memory divides into fixed size blocks called frames.

↳ It contains the actual data corresponding to logical pages.

↳ For instance A is mapped to frame F3

4) Address Translation

When program requests data using logical address (Page ID + offset), the memory management unit (mmu) uses the page Table to corresponding Frame ID in physical memory.

Final Physical is combine of Frame ID & offset within that frame.

Advantages : 1) Avoids External fragmentation
2) Efficient memory usage

Parameters	Paging	Segmentation
memory structure	Divides the process address space into fixed size pages.	Divides the process's address space into variable-size segments.
memory size	Pages are of fixed size	~~Page~~ Segment are of varying size
Speed	Paging is faster in accessing memory.	Segmentation is slower due to variable-size of segments.
Fragmentation	Paging can lead to internal fragmentation if page has unused space.	segmentation can lead to external fragmentation when memory gaps appear bet^n segments.
Accountibility	The operating system manages memory by dividing into pages.	Compiler divides memory into segments and assign actual or virtual address.
Flexibility	Same size so limited flexibility.	more ~~size~~ flexibility as segments can be different in size.
Data storage	Page tables store the mapping of page to memory.	Segment tables store the mapping of segment to memory
Logical Address	A logical address is divided into page number and offset	A logical address is divided into a segment number and offset
Size Determination	Size is determined by available memory and is fixed for all pages.	The user or program defines segments size which can vary.

Scanned with CamScanner

Question : Define fragmentation. Explain In. & External fragmentation. (4M)

Fragmentation

Fragmentation is problem in operating system where memory is used inefficiently, leaving small gaps that cannot be used.

This is leads to wasted memory space, making it difficult to allocate memory for new processes even where there is enoug memory available.

There are two types of Fragmentation

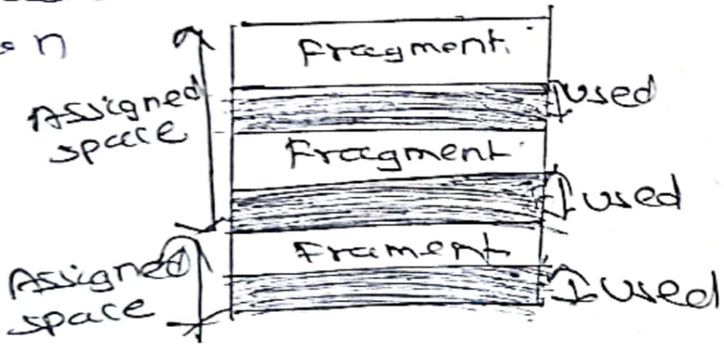
→ Internal Fragmentation

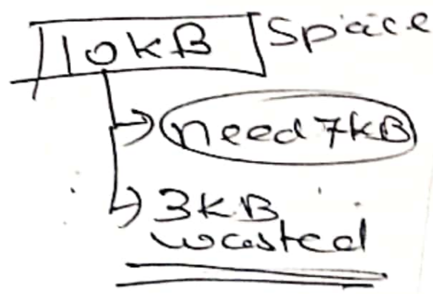
→ External Fragmentation.

Internal Fragmentation

Internal Fragmentations happen when memory is divided into fixed-size blocks, but a process doesn't need all the space in block.

The leftover space inside the block is wasted, and this waste is Internal Fragmentation

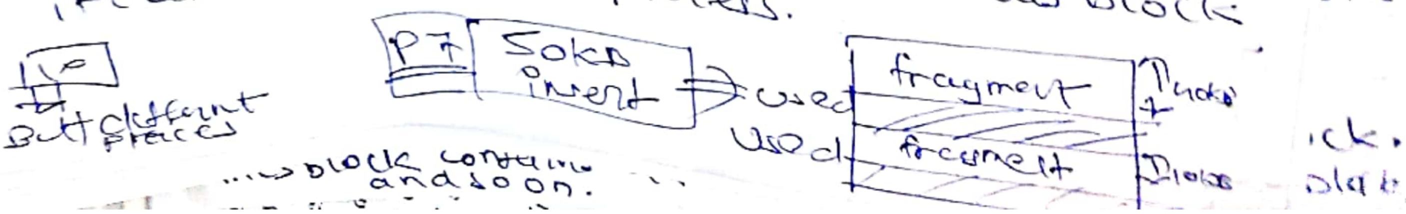




External Fragmentation

External fragmentation when there is enough total memory to meet the needs of process, but scattered in small pieces.

since the memory is not in continuous block it can't be used by process.



Difference Between Internal and External Fragmentation

Aspect	Internal Fragmentation	External Fragmentation
Definition	Occurs when memory blocks are allocated in **fixed sizes** and the process uses **less memory** than allocated, wasting space inside the block.	Happens when there is enough **free memory** but it is split into small **non-contiguous pieces**, making it hard to allocate to a process.
Cause	Caused by **fixed-size memory allocation** (e.g., paging).	Caused by **dynamic memory allocation**, leading to scattered free spaces.
Memory Wasted	Memory is wasted **inside** the allocated block.	Memory is wasted **between** allocated blocks.
Occurs In	Systems using **fixed-size blocks** for allocation.	Systems with **variable-sized blocks** or frequent memory allocation/deallocation.
Solution	Use **dynamic memory allocation** to fit the process size more closely.	Use techniques like **compaction, paging, or segmentation** to consolidate or manage scattered free space.
Example	Allocating 4KB memory to a process needing only 3.6KB, wasting 0.4KB.	A process needs 50KB, but 55KB is available in small chunks (e.g., 20KB + 15KB + 10KB).

Internal Fragmentation	External Fragmentation
Fixed-size memory blocks are given to processes.	Variable-size memory blocks are given to processes.
Happens when a process is smaller than the memory block allocated to it.	Happens when memory blocks are **freed**, but small gaps are left behind.
Solution: **Best-fit** allocation.	Solution: **Compaction** or **paging**.
Occurs with **fixed-size** memory partitions (e.g., paging).	Occurs with **variable-size** memory partitions (e.g., segmentation).
Wasted memory **inside** a block is called internal fragmentation.	Wasted memory **between** blocks is called external fragmentation.
Happens in **paging** and **fixed partitioning**.	Happens in **segmentation** and **dynamic partitioning**.
Caused by allocating more memory than needed.	Caused when non-contiguous memory gaps are too small for new processes.
Common with the **worst-fit** memory allocation method.	Common with **best-fit** and **first-fit** memory allocation methods.

c)	**Explain the following terms with respect to memory management:** i) **Dynamic relocation** ii) **Swapping**		**4M** *Each term explanation 2M*
Ans.	**i) Dynamic Relocation** When a program gets swapped out to a disk memory, then it is not always possible that when it is swapped back into main memory then it occupies the previous memory location, since the location may still be occupied by another process. We may need to relocate the process to a different area of memory. Thus there is a possibility that program may be moved in main memory due to swapping. **ii) Swapping** Swapping is mechanism in which a process can be swapped temporarily out of main memory (or move) to secondary storage (disk) and make that memory available to other processes. At some later time, the system swaps back the process from the secondary storage to main memory.		

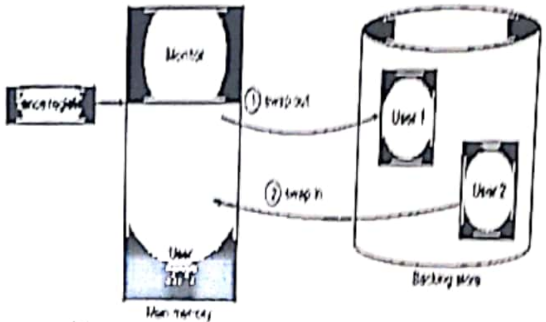


Fig: swapping in memory management

5.3 Page Replacement Algorithm: FIFO, LRU, Optimal.

Demand Paging

pagefault: If needed memory page is not in main memory

page hit: If needed page is already in main memory.

FIFO

Frame size = 3   3 2 1 3 4 1 6 2 4 3 4 2 1 4 5 2 1 3 4.

Page	3	2	1	3	4	1	6	2	4	3	4	2	1	4	5	2	1	3	4
f1	3	3	3	3	4	4	4	4	4	3	3	3	3	3	5	5	5	5	4
f2		2	2	2	2	2	6	6	6	6	4	4	4	4	4	2	2	2	2
f3			1	1	1	1	1	2	2	2	2	2	1	1	1	1	1	3	3
Result	fault	f	f	Hit	f	Hit	f	f	Hit	f	f	Hit	f	Hit	[illegible]	f	Hit	f	f

page fault = 13 = $\frac{13}{19}$

Hit = 6/19

Page	5	6	7	8	9	7	8	5	9	7	8	7	9	6	5	6
f1	5	5	5	8	8	8	8	8	8	7	7	7	7	6	6	6
f2		6	6	6	9	9	9	9	9	9	8	8	8	8	5	5
f3			7	7	7	7	7	5	5	5	5	5	9	9	9	9
Result	f	f	f	f	f	Hit	Hit	f	Hit	f	f	Hit	f	f	f	Hit

page fault = 11   5 = Hit

Scanned with CamScanner

Page Algorithm of Replacement

Page	7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1	7	0	1
f1	7	7	7	2	2	2	2	4	4	4	0	0	0	0	0	0	0	7	7	7
f2		0	0	0	0	3	3	3	2	2	2	2	2	1	1	1	1	1	0	0
f3			1	1	1	1	0	0	0	3	3	3	3	3	2	2	2	2	2	1
Result	f	f	f	f	H	f	f	f	f	f	f	H	H	f	f	H	H	f	f	f

Page fault = 15

Page	0	1	2	3	0	1	2	3	0	1	2	3	4	5	6	7
f1	0	0	0	3	3	3	2	2	2	1	1	1	4	5	5	7
f2		1	1	1	0	0	0	3	3	3	2	2	2	4	4	4
f3			2	2	2	1	1	1	0	0	0	3	3	3	6	6
Result	f	f	f	f	f	f	f	f	f	f	f	f	f	f	f	f

Page fault = 16

the specific task ... is executed ... times ...

LRU Page Replacement Algorithm:

Replace a page which is not in used for long time.

Page frame	3	2	1	3	4	1	6	2	4	3	4	2	1	4	5	2	1	3	4
f1	3	3	3	3	3	3	6	6	6	3	3	3	1	1	1	2	2	2	4
f2		2	2	2	4	4	4	2	2	2	2	2	2	2	5	5	3	3	3
f3			1	1	1	1	1	1	4	4	4	4	4	4	4	4	1	1	1
Dese Ht	f	f	f	H	f	H	f	f	f	f	H	H	f	H	F	F	F	F	F

14 = page fault.

page	5	6	7	8	9	7	8	5	9	7	8	7	9	6	5	6
f1	5	5	5	8	8	8	8	8	8	7	7	7	7	7	5	5
f2		6	6	6	9	9	9	5	5	5	8	8	8	6	6	6
f3			7	7	7	7	7	7	9	9	9	9	9	9	9	9
Dest	f	f	f	f	f	H	H	F	F	F	F	H	H	F	F	H

page fault = 11

page	7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1	7	0	1
f1	7	7	7	2	2	2	2	4	4	4	0	0	0	1	1	1	1	1	1	1
f2		0	0	0	0	0	0	0	0	3	3	3	3	3	3	0	0	0	0	0
f3			1	1	1	3	3	3	2	2	2	2	2	2	2	2	2	7	7	7
Result	f	f	f	f	H	f	H	f	f	f	f	H	H	f	H	f	H	f	H	H

page fault=12

page	3	8	2	3	7	6	3	8	5	7	6	8	3	7	2	5	6	3	8	2
f1	3	3	3	3	3	3	3	3	3	7	7	7	3	3	3	5	5	5	8	8
f2		8	8	8	7	7	7	8	8	8	6	6	6	7	7	7	6	6	6	2
f3			2	2	2	6	6	6	5	5	5	8	8	8	2	2	2	3	3	3
Result	f	f	f	H	f	f	H	f	f	f	f	f	f	f	f	f	f	f	f	f

page fault=18

2nd Optimal Page Replacement Algorithm:

Replace page that will be used-not in near future.

Page	3	2	1	3	4	1	6	2	4	3	4	2	1	4	5	2	1	3	4
f1	3	3	3	3	4	4	4	4	4	4	4	4	4	4	5	5	5	5	5
f2		2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	3	[illegible]
f3			1	1	1	1	6	6	6	3	3	3	1	1	1	1	1	3	4
Result	F	F	F	H	F	H	F	H	H	F	H	H	F	H	F	H	H	F	F

Page fault = 10

Page	5	6	7	8	9	7	8	5	9	7	8	9	6	5	6
f1	5	5	5	5	9	9	9	9	9	9	9	9	9	5	5
f2		6	6	8	8	8	8	5	5	5	8	8	8	8	8
f3			7	7	7	7	7	7	7	7	7	7	6	6	6
Result	F	F	F	F	F	H	H	F	H	H	F	H	F	F	H

Page fault = 09

... of Some examples of File

... file : Sequence of chaaracters organized ... files.

page	7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1	7	0	1
f_1	7	7	7	2	2	2	2	2	2	2	2	2	2	2	2	2	2	7	7	7
f_2		0	0	0	0	0	0	4	4	4	0	0	0	0	0	0	0	0	0	0
f_3			1	1	1	3	3	3	3	3	3	3	3	1	1	1	1	1	1	1
Result	F	F	F	F	H	F	H	F	H	H	F	H	H	F	H	H	H	F	H	H

page fault = 09

page	4	1	2	3	1	4	1	5	2	3	4	2	1	3	4	1	2	5	1	2
f_1	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	2	2	2	2
f_2		1	1	1	1	1	1	5	2	2	2	2	1	1	1	1	1	1	1	1
f_3			2	3	3	3	3	3	3	3	3	3	3	3	3	3	3	5	5	5
Result	F	F	F	F	H	H	H	F	F	H	H	H	F	H	H	H	F	F	H	H

page fault = 09