Date
12/03/24

UNIT-3

DEADLOCKS

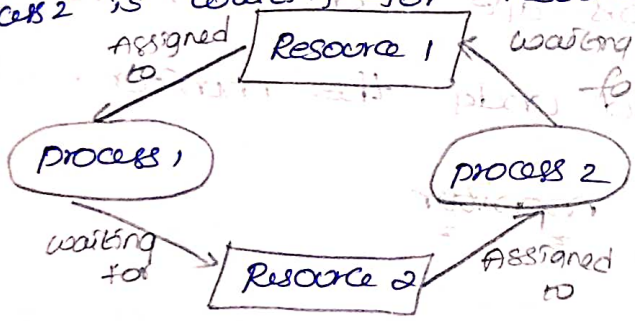
* Introduction of Deadlocks

→ Deadlock is a situation where a set of processes are blocked because each process is holding a resource & waiting for another resource req acquired by some other process.

→ consider an example when two trains are coming toward each other on the same track and there is only one track, none of the trains can move once they are in front of each other.

→ A similar situation occurs in operating systems when there are two or more processes that hold some resources & wait for resources held by other.

→ For eg:- in the below diagram, process 1 is holding Resource 1 & waiting the resource 2 which is acquired by process 2 & process 2 is waiting for resource 2.



* Consumable Resources

→ consumable resources are created with help of users & programmers.

→ consumable resources must & should needed to use for the process.

→ finally discarded consumable resources from the private area.

→ Sometimes consumable resources overlooded into main memory or secondary memory.

* Reusable Resources

→ Reusable resources are created with help of using developers. Reusable resources shared by different users.

→ Reusable resources are available in main memory for secondary memory.

* Deadlock conditions & Deadlock preventions

→ To properly maintaining of deadlock required to follow the 4 conditions are:

1, Mutual exclusion

2, Hold & wait

3, No preemptive

4, Circular wait

→ Sometimes these 4 conditions are called deadlock preventions:

* mutual Exclusion

→ Initially consider two processes one common resource. process names are P_1 & P_2 and common resource name is P.

→ P_1 enters into critical section for the purpose of execution.

→ P_1 properly is used common resource & successfully execution under the critical section.

→ Process P_1 execution time does not allow any other process into critical section.

→ P_2 ee process enter into critical section for purpose of execution.

→ P_2 property is used common resource & successfully executed under critical section.

→ process P_2 executed time does not allow any other process into critical section.

* conclusion :-

→ Mutual exclusion property does not support more no. of processes & resources.

→ To overcome this drawback we are using 'Hold & wait' condition.

* Hold & wait

→ According to Hold & wait, consider two processes P_1, P_2 & resources R_1 & R_2.

→ Hold & wait is implemented based on any process request waiting for one resource & hold another resource.

→ P_1 process enter into critical section for the purpose of execution.

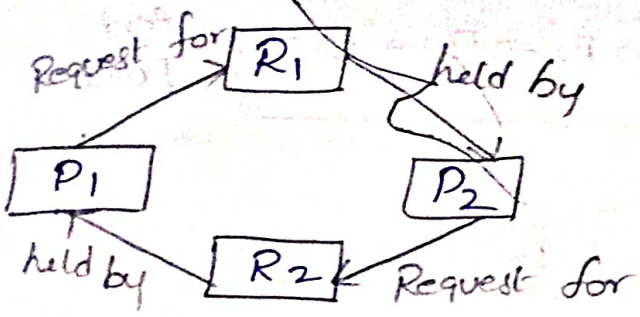
→ P_1 sends send a request to R_1 resources & already held by R_2 resource.

→ Finally process P_1 successfully completed under critical section.

→ P_2 process enter into critical section for the purpose of execution.

→ P_2 process send a request to R_2 resource & already held by R_1 resource.

→ Finally process P_2 successfully completed under critical section.



* No preemptive

→ Initially consider more than one process & more than one resource.

→ According to no preemption once enter any process into critical section & must & should complete under critical section.

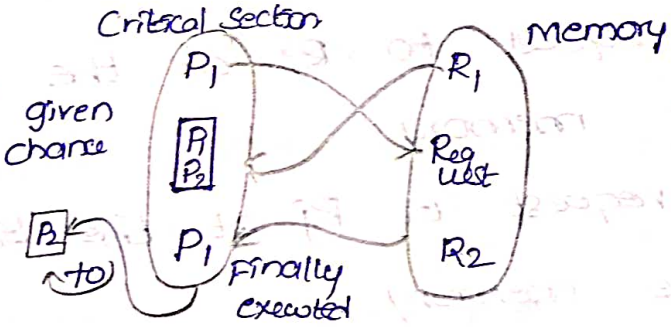
→ Any process execution time does not allow any another process into critical section.

* example :-

· consider two process P_1 & P_2 and resource R_1, R_2.

* conclusion :-

· P_1 execoted time does not allow P_2 into critical section.



c) circular wait :-

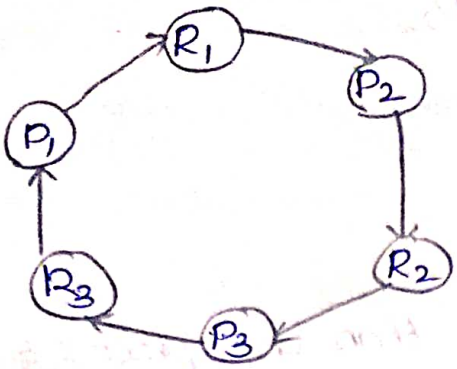
→ According to circular wait consider more number of processors & more number of resources.

→ Circular wait is implemented with the help of using two protocols are.

1. $F(R_i) >= F(R_j)$.

2. $F(R_j) >= F(R_i)$

Start point $P_1 \rightarrow R_1 \rightarrow P_2 \rightarrow R_2 \rightarrow P_3 \rightarrow R_3 \rightarrow P_1$ End point



→ Consider the two process P_1 & P_2 & Resources R_1 & R_2.

→ Process P_1 request to R_j at the time R_i provides R_i (or) R_1 by the memory.

→ P_1 Sends a request to R_i at the time provides R_j by the memory.

→ Finally P_1 successfully completed execution under the processor using two conditions.

→ P_2 sends a request to R_i at the time R_j provides by the memory

→ P_2 sends a request to R_j at the time R_i provides by the memory.

→ Finally P_2 process successfully completed execution under the processors using two conditions / protocols.

→ Sometimes, circular wait is implemented based on the circular queue.

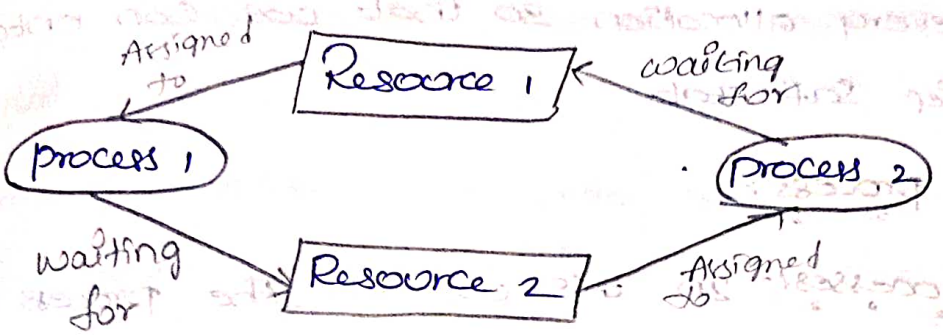
→ Circular queue is always maintained between starting & ending points. are same.

* Deadlock Detection And Recovery

* Deadlock Detection

1. If Resources have single instance:

→ In this case for Deadlock detection we can run an algorithm to check for cycle in the Resource Allocation Graph. presence of cycle in the graph is the sufficient condition for deadlock



→ In the Above diagram, resource 1 & resource 2 have single instances.

→ There is a cycle $R_1 \rightarrow P_1 \rightarrow R_2 \rightarrow P_2$. So. Deadlock is confirmed.

→ If there are multiple instances of resources we check the cycle is formed or not in the system. but that is not enough to detect the deadlock we also need to use some safety algorithm on the system so that we can convert the resource allocation graph into the request & the allocation matrix.

* Deadlock Recovery:-

1. For resources:-

→ Preempt the resource: In this we take the resource from one process to the process that needs it to finish

its execution & after the execution is completed, the process soon releases the resource. In this the resource selection is difficult & the matching of the resources is also difficult.

* 2. Rollback to safe state:- To enter into deadlock, the system goes through several state. In this, the operating system can easily roll back the system to the earlier safe state. To do so we require to implement checkpoints at every state. At that time when we need to rollback every allocation so that we can enter into the earlier safe state.

2. for a process:

* kill a processes:- In this we kill the process due to which deadlock occured. But the selection of the process to kill is a tough task. In this, the operating system mainly kills that process, which does not want more till now.

* kill all process:- Kill all the process is not a suitable approach, we can use this approach when the problem becomes critical. By killing all the process the system efficiency will be decreased & we have to execute all the processes further from the start.

* Deadlock Avoidance:- In deadlock avoidance the request for any resources will be granted if the resulting state of the system doesn't cause deadlock in the system.

→ The state of the system will continously be checked for safe & unsafe states.

→ In order to avoid deadlocks, the process must tell OS, the maximum number of resources a process can request to complete its execution.

→ the simplest & most useful approach states that the process should declare the maximum number of resources of each type it may ever need.

→ the deadlock Avoidance algorithm examines the resource allocation so that there can never be a circular wait condition.

* Banker's Algorithm / Safety Algorithm / Deadlock Avoidance Algorithm

1. Initially consider a available / work & finish vector Available / work represented as 'm' & finish represented as 'n'. finish [i] = 1 - - - n and finish [i] = false;

2. Find an 'i' such that

- finish [i] = false;
- Need [i] <= work;

if 'i' exists go to step 3.

else go to step 4.

3. work = work + Allocation

go to step 2.

4. Finish [i] = true for all processes.

* Request Resource Algorithm

1. Request [i] = Need [i]

go to step-2. otherwise error condition

2. Request [i] <- Available / work.

go to step-3. otherwise wait for shorter / longer time

3. Available = Available - Request

Allocation = Allocation + Request

Need [i] = Need [i] - Request

* Deadlock Avoidance

Process	Allocation			max Allocation		
	A	B	C	A	B	C
P_0	0	1	0	7	5	3
P_1	2	0	0	3	2	2
P_2	3	0	2	9	0	2
P_3	2	1	1	2	2	2
P_4	0	0	2	4	3	3

A has 10 instances

B has 5 instances

C has 7 instances

Sol:- ~~Need to check for~~ P_4 : Need[i] <= work.

A instances - Total Allocation value.

work A = 10 - (10 + 2 + 3 + 2 + 0)

= 10 - 7

= 3

work B = 65 - (1 + 0 + 0 + 1 + 0)

= 5 - 2

= 3

work C = 2 (0 + 0 + 2 + 1 + 2)

= 7 - 5

= 2

work vector = [A B C] = [3 3 2]

Need vector = max Allocation - Allocation

P_0 = [7 5 3] - [0 1 0]

= [7 4 3]

P_1 = [3 2 2] - [2 0 0]

= [1 2 2]

P_2 = [9 0 2] - [3 0 2]

= [6 0 0]

$P_3 = [2\ 2\ 2] - [2\ 1\ 1] \rightarrow [0\ 1\ 1]$

$P_4 = [4\ 3\ 3] - [0\ 0\ 2] \rightarrow [4\ 3\ 1]$

$$\begin{bmatrix} 7 & 4 & 3 \\ 1 & 2 & 3 \\ 6 & 0 & 0 \\ 0 & 1 & 1 \\ 4 & 3 & 1 \end{bmatrix}$$

these two values applying on different processes for the purpose of safe (or) not safe order to check.

for P_0

Need [i] <= work

P_0 = Need vector of P_0 <= work vector

= [7 4 3] <= [3 2 2]

P_0 = Above condition is not satisfied

∴ It is unsafe.

P_1 = Need vector of P_1 <= work vector

= [1 2 2] <= [2 0 0]

∴ Above condition is satisfied

P_1 is safe.

New work = work + Allocation of P_1

= [3 3 2] + [2 0 0]

= [5 3 2] present

P_2 = need vector of P_2 <= work vector

= [6 0 0] <= [5 3 2]

Above condition is not satisfied

∴ It is unsafe.

P_3 = Need vector of P_3 <= work vector

= [0 1 1] <= [5 3 2 0]

∴ It is safe.

New work = work + Allocation

= [5 3 2] + [2 1 1]

= [7 4 3]

P_4 = Need vector of P_4 <= present work vector

= [4 3 1] <= [7 4 3]

P_4 is safe

New work = work + Allocation

= [2 4 3] + [0 0 2]

= [7 4 5]

As P_0 is unsafe. we need to change into safe.

P_0 = need of P_0 = present work vector

= [7 4 3] <= [7 4 5]

P_0 is safe

work = work + Allocation (P_0)

= [7 4 5] + [0 1 0]

= [7 5 5]

P_2 = need vector of P_2 <= work vector

= [6 0 0] <= [7 5 5]

= P_2 is safe

New work = present work + Allocation

= [7 5 5] + [3 0 2]

= [1 5 7]

Safe order => P_1 P_3 P_4 P_0 P_2

* Deadlock Detection :-

Step 1 :- consider a work / available vector & finish vector work available vector is represented as 'm' & finish vector represented as "n".

finish [i] = 1 ---- n

Step 2 : Finish of [i] = false

Allocation [i] != 0 == 0

Step 3 :- Find an i such that

Request [i] <= work;

if i exists go to Step 4

else

go to Step 5

Step 4 :- work = work + Allocation;

finish [i] = true

go to Step 3

Step 5 :- Finish [i] = false;

Eg :-

process	Request			Allocation		
	A	B	C	A	B	C
P_0	0	0	0	0	1	0
P_1	2	0	2	2	0	0
P_2	0	0	0	3	0	3
P_3	1	0	0	2	1	1
P_4	2	0	2	0	0	2

A has 7 instances

B has 2 instances

C has 6 instances

Sol:- Need to check for P_0

Request [i] $P_0 \leq$ Work

work A = 7 - [0 + 2 + 0 + 0 + 2]
(3)

= 7 - 5 7

= 2 0

work B = 3 - [0 + 0 + 0 + 0 + 0]
(1)

= 3 2 0

work C = 6 - [0 + 2 + 0 + 0 + 2]
(0) (3) (1)

= 6 - 6

= 0

work vector = [A B C] = [0 0 0]

Request [i] $P_0 \leq$ work

[0 0 0] $\leq$ [0 0 0]

It is safe

New work = work + Allocation

= [0 0 0] + [0 1 0]

= [0 1 0]

New work is applied to P_0

Request [i] $P_1 \leq$ work

[2 0 2] $\leq$ [0 0 0]

It is unsafe

No needed to find out new work because it is unsafe

Request [i] $P_2 \leq$ work

[0 0 0] $\leq$ [0 1 0]

It is safe

New work = work + Allocation

= [0 1 0] + [3 0 3]

= [3 1 3] New work is applied to P_2

Request [i] $P_3 \leq$ work

[1 0 0] $\leq$ [3 1 3]

It is safe

New work = work + Allocation

= [3 1 3] + [2 1 1]

= [5 2 4] New work is applied to P_4

Request [i] P_4 <= work

[2 0 2] <= [5 2 4]

It is safe.

New work = work + Allocation

= [5 2 4] + [0 0 2]

= [5 2 6]

Finally, the end of the detection

Safe order = [P_0 P_2 P_3 P_4]

unsafe order = [P_1]

* Deadlock Recovery

→ All the processes are successfully executed using diff resources from the memory.

→ executed processes are release different resources & same resources require to send from processor to main memory

→ sometimes processes are not successfully completed due to system mod functions failures (IO device failure, disk failure, Read & write failures. memory device failures. power failures)

→ Above system failures are rectify with the help of using check point technique.

→ Checkpoint technique provides to operations to

the user.

Redo undo

→ Redo operation is taking one care of to contain the transaction from failures occurred of the applications.

→ undo operations is taking care of prevent previous Statements of the transaction (or) process

Eg: consider a two account values are in A & B

Initially A as maintained 5000/-

Initially B as maintained 4000/-

Here needed to transfer 1000/- from account A to B

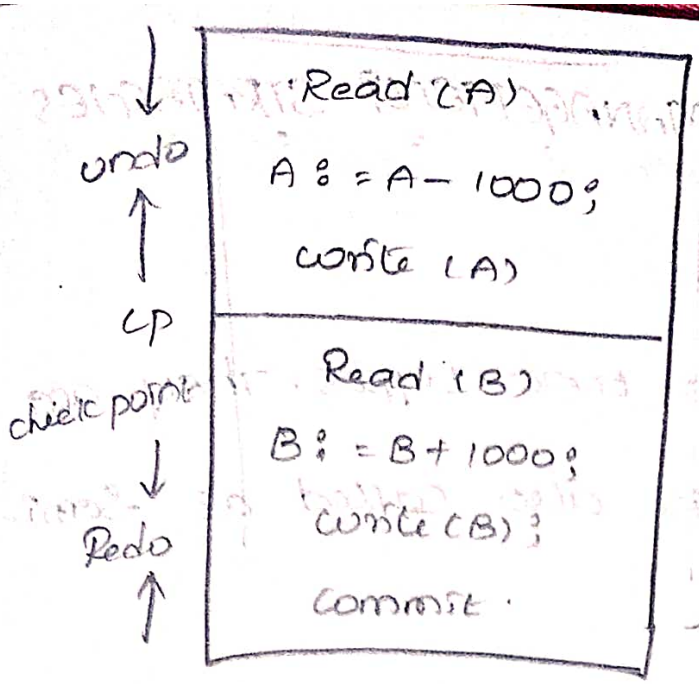
→ To transfer any amount of information from one account to another account using transaction.

→ Transaction means execution of unit of application. this application needed to use two operations

1) Read() 2) write()

→ Read() operation is used to retrieving of any amount value (or) Account value from database to private area.

→ write() operation is used to retrieving send back any required information from Seco private area to Secondary memory.



→ Some of the processors are not recovery from deadlock at that time corresponding process killed from the main memory (or) Secondary memory.

p. 11 4. MEMORY MANAGEMENT STRATAGIES

* INTRODUCTION

→ Memory is divided into two types. They are

1. primary memory. It is also called as Semi-conductor memory.

2. Secondary Memory :- It is also called as magnetic memory.

→ primary memory maintains base values

→ Secondary memory maintains limit values

→ Seee Memory Management is the process of controlling & co-ordinating computer memory. assigning portions known as blocks to various running programs to optimize the overall performance of the system.

* Basic concepts :-

→ Logical Address :-

→ It is the address generated by cpu while a program is running is referred as Logical Address.

→ The logical address is virtual as it does not exist physically.

Hence. it is also called as virtual Address.

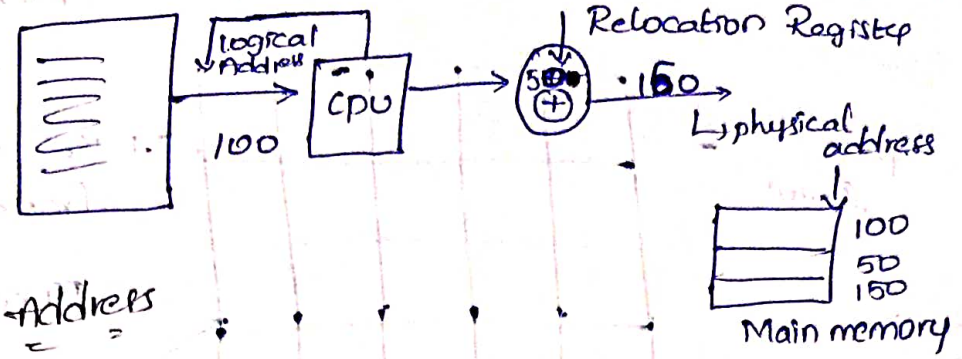
→ This address is used as a reference to access the physical memory location.

* Relocation :-

→ It is the process of assigning load Addresses for position dependent code & data of a program & adjusting the code & data to reflect the assigned addresses.

(or)

→ Relocation Registers a mechanism is used to convert any logical or virtual Address converted into physical address. Eg:



* physical Address

→ physical Address identifies a physical location in a memory.

→ MMU (memory management unit) computes the physical address for the corresponding logical address.

physical address is calculated using mathematical formula. physical address = logical / virtual Address + Relocation register address content value

Eg: physical address PI = 120 + 50 = 170

* Buffer manager :-

→ Buffer manager is a software layer. Software layer is appeared on main memory.

→ According to buffer manager available memory is divided into equal number of partitions. Each & every partition is called page or frame.

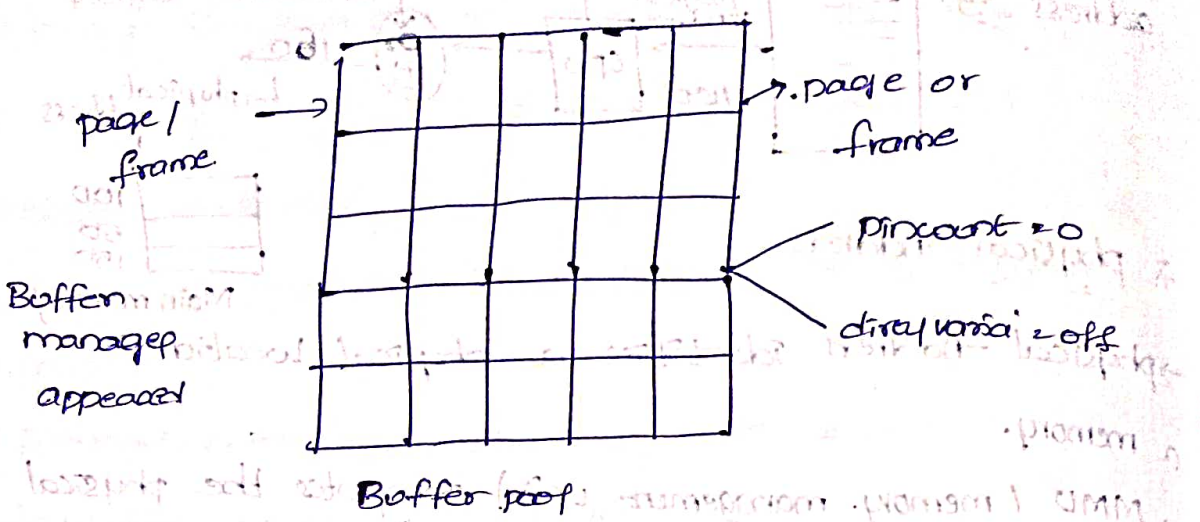
→ Sometimes main memory gets overloaded at that time needed to transfer some of the pages from main memory to secondary memory.

→ Transformation is done based on page replacement algorithms.

→ Once found any page fault must & should transfer corresponding page from main memory to secondary

memory with help of Swapping.

→ Sometimes bringing required pages from Secondary memory to main memory using Swapping.

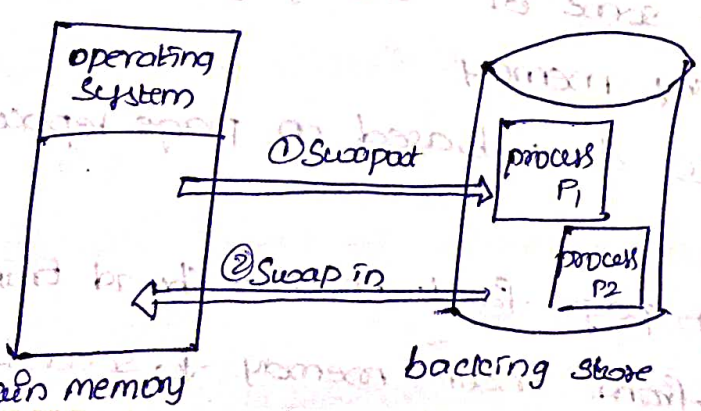


* Swapping :-

→ Swapping is a mechanism in which a process can be swapped temporarily out of main memory to Secondary ~~mem~~ storage and make that memory available to other ~~oth~~ processes which is known as swap-out operation.

→ At some later time, the system swaps back the process from the secondary storage to main memory which is known as swap-in operation

→ Though performance is usually affected by swapping process but it helps in running multiple & big processes in parallel & that's the reason swapping is also known as a technique for memory compaction.



* Contiguous memory Allocation:-

→ In the contiguous memory allocation both the operating system & the user must reside in the main memory.

→ the main memory is divided into two portions one portion is for the operating & other is for the user program.

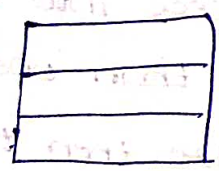
→ In the contiguous memory allocation when any user process request for the memory a single section of the contiguous memory block is given to that process according to its need.

→ We can achieve contiguous memory allocation by dividing memory into the fixed-sized partition.

→ A single process is allocated in that fixed size single partition.

But this will increase the degree of multiprocess programming means more than one process in the main memory that bounds the number of fixed partition done in memory.

Internal fragmentation increases because of the Contiguous memory allocation



Process

memory pages

* Fixed Sized partition :- In the fixed sized partition the System divides memory into fixed size partition here entire partition is allocated to a process & if there is some wastage inside the partition is allocated to a process & if there is some wastage inside the partition then it is called internal fragmentation.

→ Advantage :- management or book keeping is easy.

→ Disadvantage :- Internal fragmentation

* variable size partition :- In the variable size partition, the memory is treated as one unit & space allocated to a process is exactly the same as required & the leftover space can be reused again.

→ Advantage :- There is no internal fragmentation.

* Memory Allocation Algorithms :-

→ In partition Allocation, when there is more than one partition freely available to accomodate a process request, a partition must be selected.

→ To choose a particular partition allocation method is needed. A partition allocation method is considered better if it avoids internal fragmentation.

→ When it is time to load a process into the main memory & if there is more than one free block of memory of sufficient size then the OS decides which free block to allocate.

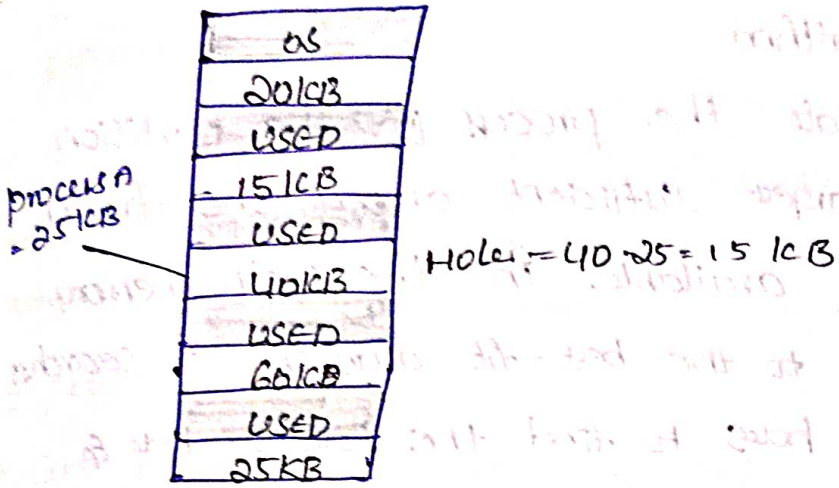
→ memory Allocation Algorithm is divided into 3 types.

1) First fit Algorithm
2) Best fit Algorithm
3) Worst fit Algorithm

* First Fit Algorithm

→ According to first fit Algorithm, it scans memory from the beginning & chooses the first available block that is large enough.

→ In the first fit, the partition is allocated which is the first sufficient block from the top of main memory



* Advantages of first fit Algorithm:-

→ First fit Algorithm minimized cost & time duration.

* Drawbacks of first fit Algorithm:-

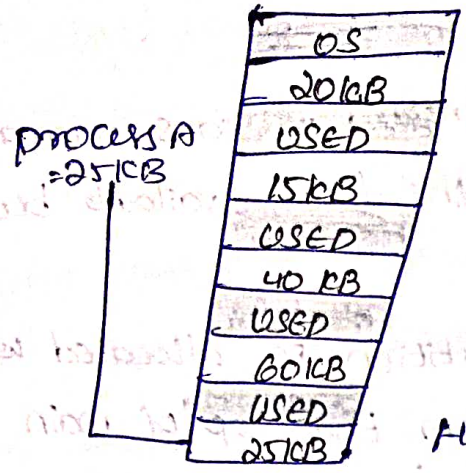
→ Internal fragmentation appears.

→ Internal fragmentation means process size should maintain lesser size value compared with partition size value.

* Best fit Algorithm

→ Best fit Allocation the process to the partition which is the first smallest sufficient partition among the free available partition.

→ It searches the entires list of holes to find the smallest hole whose size is greater than or equal to the size of the process.

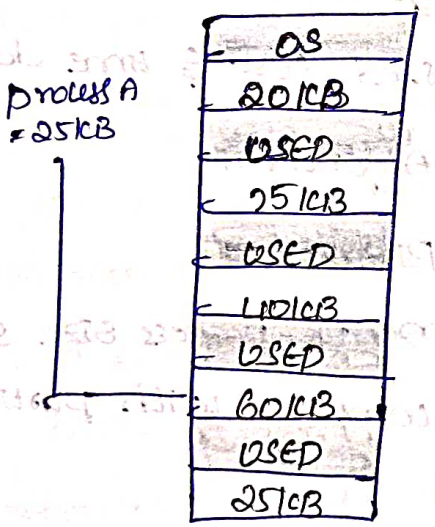


Hole = 25-25 = 0KB

* Worst fit Algorithm

→ Worst fit Allocate the process to the partition which is the largest sufficient among the freely available partitions available in the main memory.

→ It is opposite to the best-fit algorithm. It searches the entire list of holes to find the largest hole & allocate it to process.



Hole = 60-25
= 35 KB

* Advantages

→ The main advantage of worst fit algorithm is process of more than one process or job is done.

* Disadvantages

→ It has internal fragmentation

→ more time is required

→ It is costlier.

* conclusion :-

→ To overcome the drawback (Internal Fragmentation) we follow other Alternative algorithms

1) paging

2) Segmentation

3) combination of paging & Segmentation

* compaction :-

→ It is the process of collecting all free space of main memory & allocation of free space information added at end of main memory.

* paging :-

→ In computer operating systems, paging is a memory management scheme by which a computer stores and retrieves data from secondary storage for use in main memory.

→ In this scheme, the operating system retrieves data from secondary storage in same size blocks called pages.

→ paging is a memory management scheme that eliminates the need for contiguous allocation of physical memory.

→ This scheme permits the physical address space of a process to be non-contiguous.

* Logical Address or Virtual Address :- An address generated by the CPU.

* Logical Address Space or Virtual Address space :- The set of all logical addresses generated by a program.

* physical Address :- An address actually available on memory unit.

* physical Address space :- the set of all physical addresses corresponding to the logical addresses.

→ The physical Address space is conceptually divided into a number of fixed-size blocks. called frames.

→ the logical address space is also splitted into fixed size blocks. called pages.

* page size = frame size. ✓

* According to paging technique any user application is divided into no. of pertical partitions.

→ Each and every partition is technically caused page & frame.

→ page & frames are required to send into processor for the purpose of execution.

→ CPU assigns same address value to the user application (no. of pages) is called virtual address.

→ According to paging technique logical address values are divided into two partitions → page no of
→ paging technique provides mapping table to the → Displacement/offset
user.

→ mapping table is maintained two attribute value
↳ page no: 2, frame no

→ According to paging technique physical address is divided into two partitions.

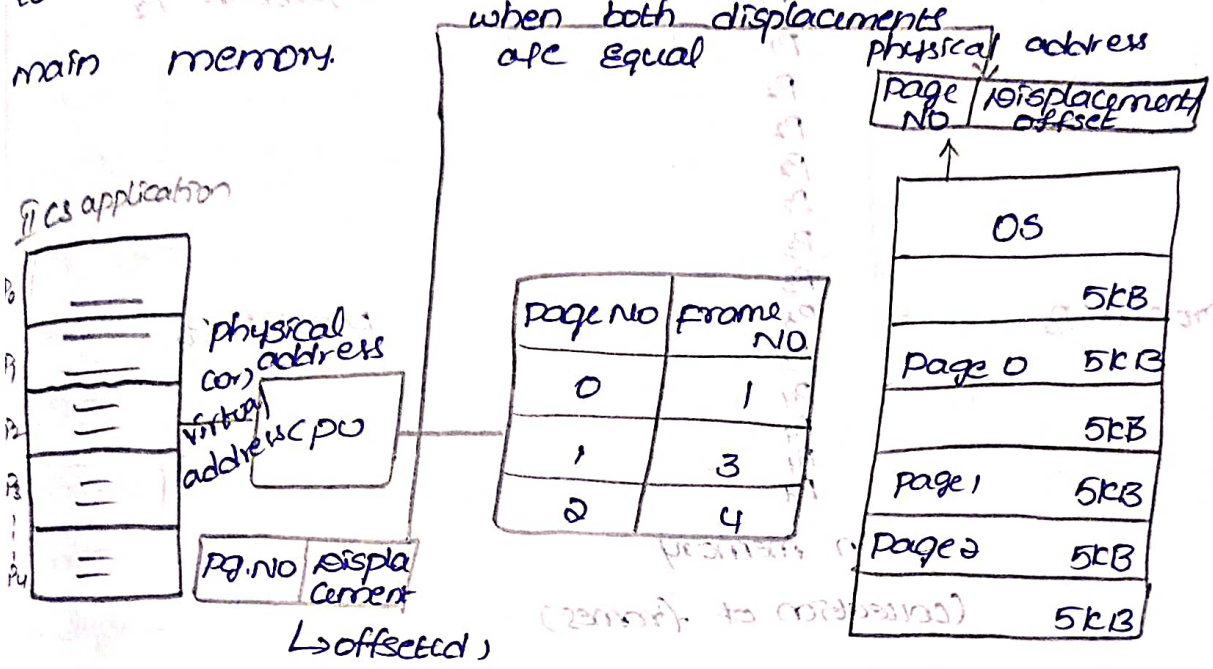
Frame
Page no

Displacement / offset (D)

→ page frame numbers are required to send into frames of the main memory.

→ paging mapping table supported to transfer page no. into frames of the memory.

→ when both displogical address offset value & physical address offset values are Equal at the time to transfer page numbers into frames of the main memory.



→ address generated by CPU is divided into:

→ page number (p): Number of bits required to represent the pages in Logical Address space or page number.

→ page offset(d): Number of bits required to represent particular word in a page or page size of logical address space or word number of a page or page offset.

→ physical address is divided into:-

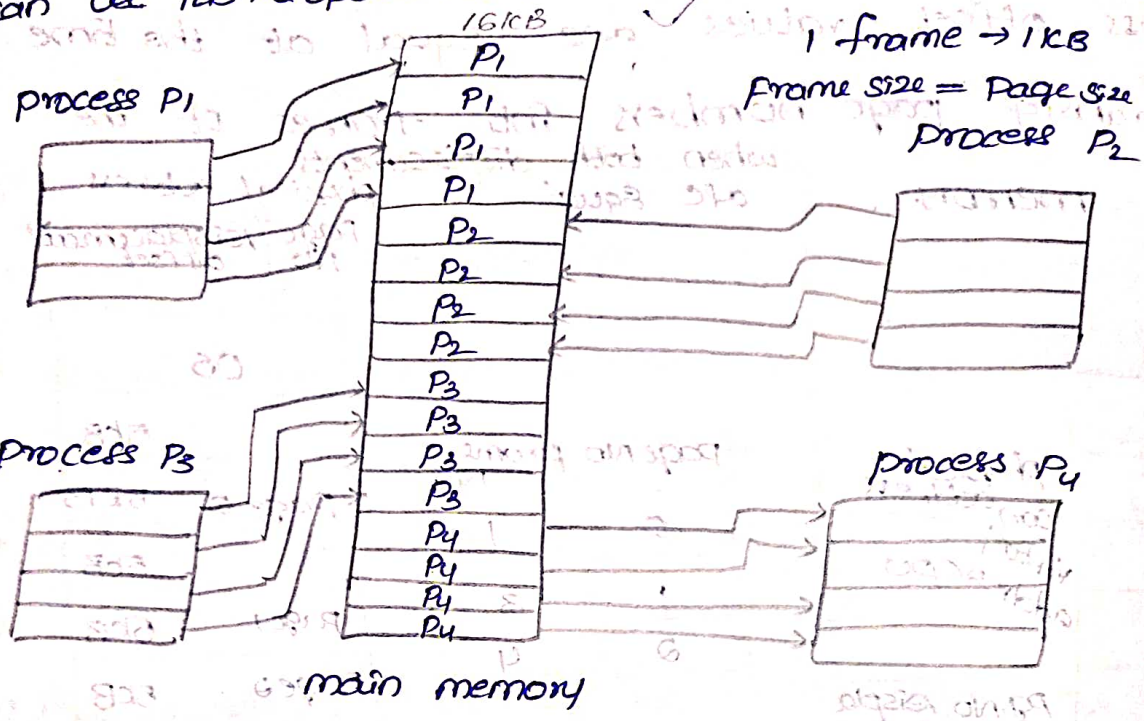
→ frame number(f): Number of bits required to represent the frame of physical address space or frame number.

→ frame offset(d): Number of bits required to represent particular word in a frame or frame size of physical address space or word number of a frame or frame offset.

→ The hardware implementation of page table can be done by using dedicated registers.

But the usage of register for the page table is satisfactory only if page table is small

If page table contain large number of entries then we can use TLB, a special, small fast lookup hardware cache.



* paging Hardware:-

→ The hardware support for paging. every address generated by the cpu is divided into two parts: a page number and a page offset.

→ The page number is used an index into a page table. The page table contains the base address of each page in physical memory.

* SEGMENTATION:-

→ In operating systems segmentation is a memory management technique in which the memory is divided into the variable size pages.

→ Each part is known as segment which can be allocated to a process

→ the details about each segment are stored in a table called as segment table. segment table is stored in one of the segments.

→ segment table contains mainly two information about segment.

1. Base Address :- It contains the starting physical address where the segments reside in memory

2. Limit :- It is the length of the segment.

→ According to segmentation technique initially consider any user application

→ user application is divided into no. of partitions

→ Each & Every partitions is called segments.

→ According to segmentation technique, Available logical address is divided into two partitions. values

1) Segment number

2) Object or displacement

→ Segmentation technique provides segmentation map table

→ Segmentation map table provides 3 attributes to user

1) Segment

2) Base Address

3) Limit Value

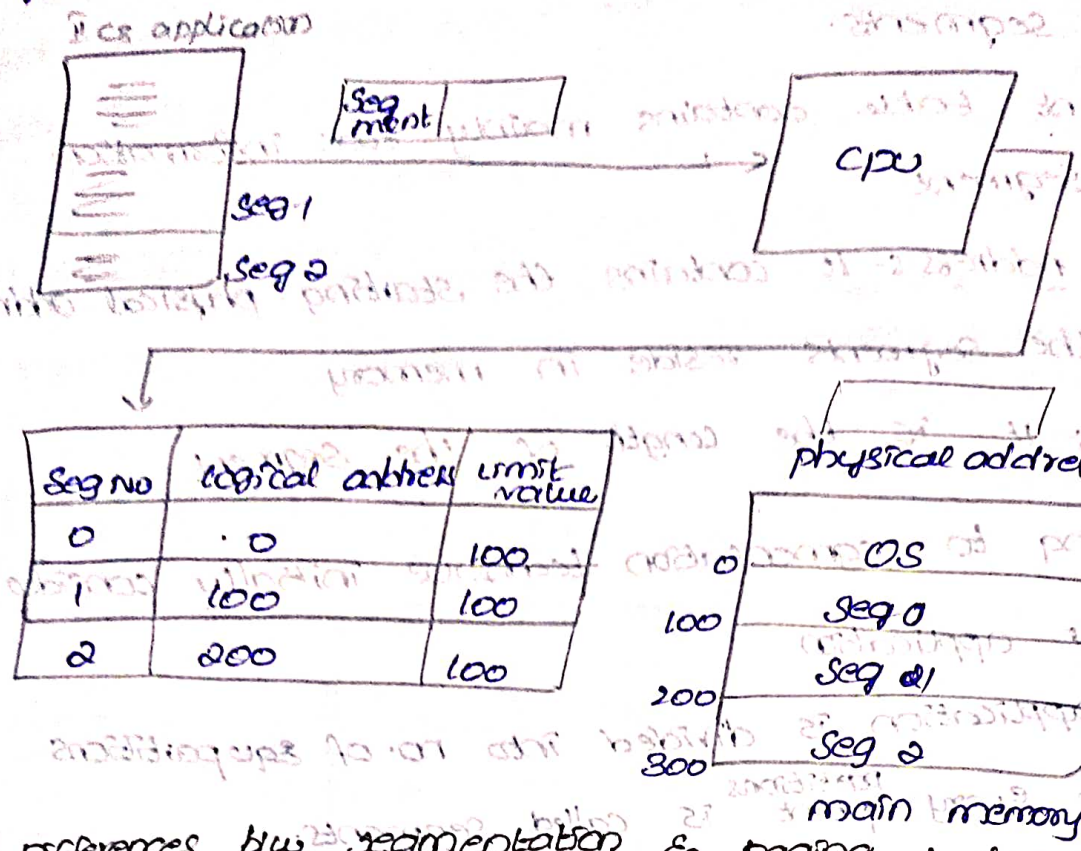
→ According to segmentation technique Available physical address is divided into 2 partitions

1) Base address

2) Offset / displacement

→ When logical address offset value & physical address values are equal then transfer segment information from segmentation table to base address of main memory.

→ Segmentation technique suffering with External fragmentation.



* Differences b/w segmentation & paging techniques.

Paging	Segmentation
→ Initially require any process or job	→ Initially require any process or job.
→ Available process is divided into number of pages.	→ Available process is divided into number of segments
→ According to paging technique logical address contains page number & offset / displacement	→ According to segment techniq logical address contain segment no. & offset / displacement
→ maintain page table	→ maintain segment table
→ It maintain 2 attributes value pages number & frame value.	→ It maintain 3 attribute value segment number, limit & base val
→ According to page technique physical address contain frame number & offset / displacement	→ According to segmentation technique physical address contain base & displacement
→ It is suffering with internal fragmentation.	→ It is suffering with external fragmentation.

Eg: Refer page back diagram

eg: segmentation block diagram

* Translation Look-Aside Buffer :-

→ The Translation Look aside buffer (TLB) is associative, high speed memory.

→ Each entry in TLB consists of two parts one is a key & another is a value

→ The TLB is used with page tables in the following way. The TLB contains a few of the page-table entries.

→ When a Logical address is generated by the CPU, its page number is presented to the TLB. If the page number is immediately available is used to access memory.

→ If the page number is not in the TLB a memory reference to the page table must be made. Depending on the CPU, this may be done automatically in hardware or an interrupt to the operating system.

Page replacement Algorithms

→ These are divided into different types

→ FIFO stands for First In First Out.

→ It is a page replacement Algorithm

→ Page Replacement Algorithm is used to find out page fault and fault page from the main memory & then replace the fault page from the main memory to secondary memory. using swapping technique

→ Fault page finded of the main memory based on different replacement algorithms.

Algorithm for FIFO

1) Start the process

2) Declare the variables

3) Read the no. of pages and no. of frames

4) Allocate the pages to frames

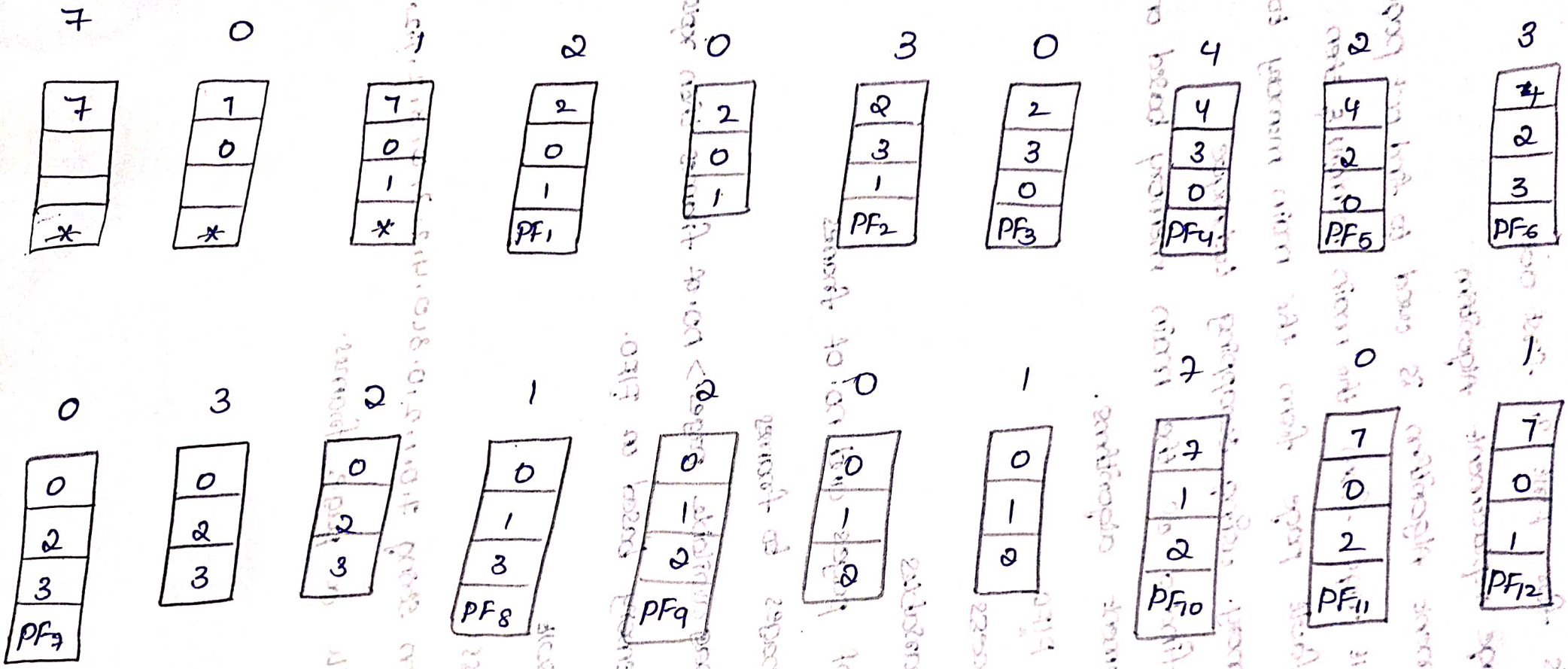
5) If no. of pag available pages > no. of frames then xplace the pags of memory based on FIFO.

6) Display the result.

7) Stop the process.

Eg:- consider a given string 7,0,1,2,0,3,0,4,2,3,0,3,2,1,2,0.

1, 7, 0, 1 consts a page frames.



→ Total Number of Page faults = 12

→ Page fault rate = no. of page faults / number of pages in the string = 12/20

= 0.6

* LRU (least Recently used)

→ LRU stands for least Recently used

→ It is a replacement algorithm

→ page replacement algorithm is used to find out page fault and fault page from the main memory to secondary memory. using swapping technique.

→ fault page finded of the main memory based on different replacement algorithms.

* LRU Algorithm

1) Start the process.

2) According to LRU algorithm we need to search or scan available main memory at least once.

3) According to LRU algorithm, the algorithm checks for

i) Recently used pages (RU)

(ii) pre Recently used pages (PRU)

iii) least Recently used pages (LRU)

4) once find least recently used pages replace the least recently used pages.

5) Insert new pages in place of least recently used pages

6) Display the result

7) Stop the process.

Eg:- consider page string 7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 1, 2, 0, 1, 7, 0, 1 with 3 page frames.

7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1

7	7	7	2	2	2	2	2	2	2	2	2	2	2	2	2	2
	0	0	0	0	0	0	4	4	4	0	0	0	1	1	1	1
		1	1	1	3	3	3	3	3	3	3	3	3	3	0	3
			PF_1		PF_2		PF_3			PF_4			PF_5		PF_6	

7 0 1

7	7	1
1	0	0
3	3	3
PF_7	PF_8	PF_9

* Optimal page Replacement Algorithm

1, It is a page replacement algorithm

2, optimal algorithm initially required a string (collection of pages)

3, Available pages needed to allocate, available frames of memory.

4, According to optimal algorithm needed to inserting new pages into frames of memory. At that time most of should require replacement of pages.

5, Replacement of pages is done based on "least value."

6, Sometimes replacement of pages is done based on least count value (which page maintain least values).

7, Display the result

8, Stop the process.

consider page string 7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, ~~1, 2, 0, 1, 7, 0~~ 3

with 3 page frames.

7 0 1 2 0 3 0 4 2 3 0 3 2 3

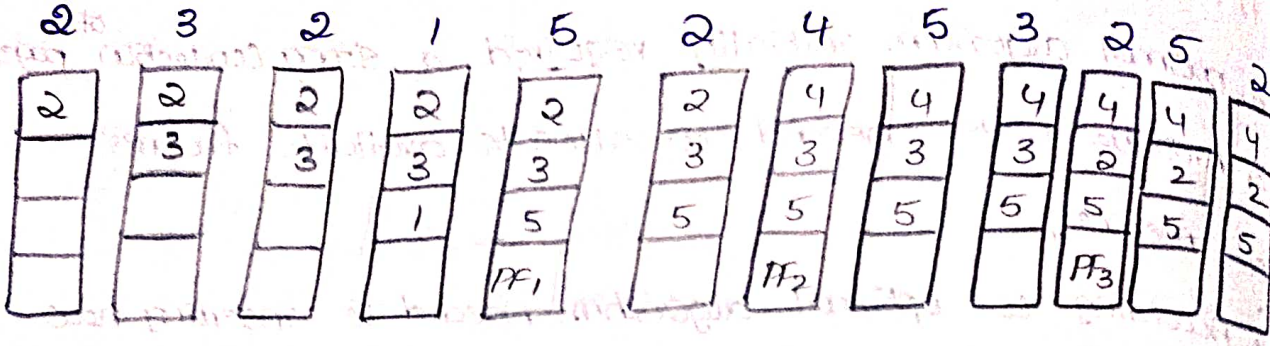
7	7	7	7	7	7	7	7	7	7	7	7	7	7
	0	0	2	2	3	3	4	4	3	3	3	3	3
		1	1	0	0	0	0	2	2	0	0	2	2
			PF_1	PF_2	PF_3		PF_4	PF_5	PF_6	PF_7		PF_8	

No. of page faults = 8

page fault rate = page faults / number of pages

in the string = 8/14 = ~~0.28~~ 0.57

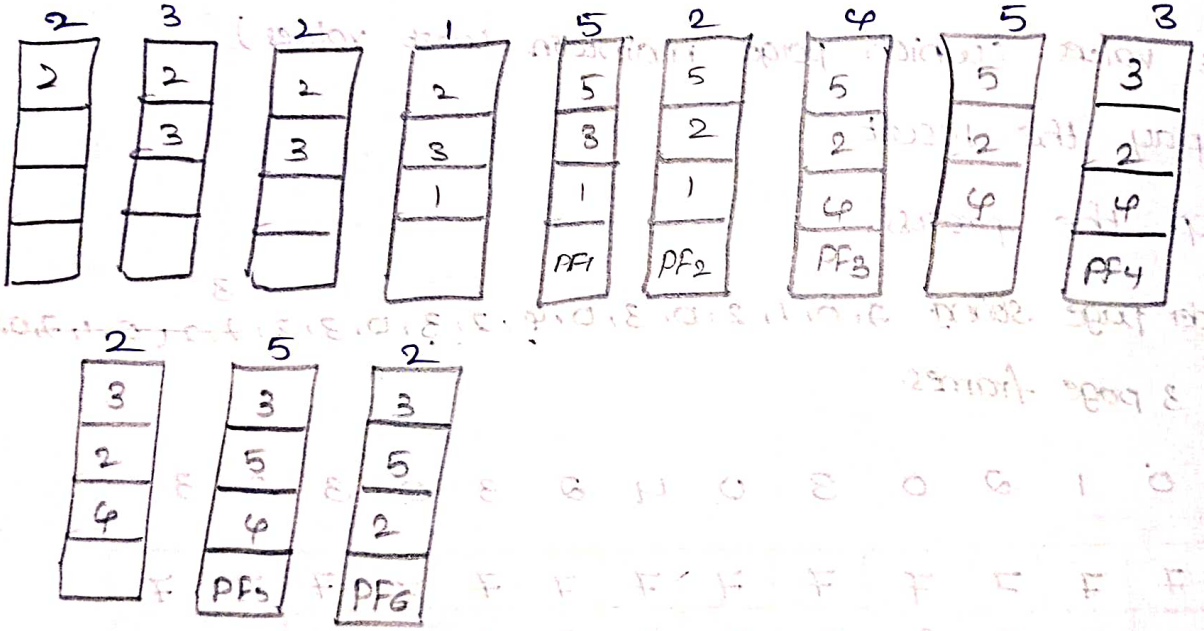
Consider a string 2, 3, 2, 1, 5, 2, 4, 5, 3, 2, 5, 2 using 3 frames using optimal & fifo algorithms



Total no. of page faults = 3

page fault rate = $\frac{\text{No. of page faults}}{\text{Total no of string}} = \frac{3}{12} = 0.25$

* FIFO



No. of page faults = 6

Page fault rate = $\frac{\text{No. of page faults}}{\text{No. of page in string}} = \frac{6}{12} = \frac{1}{2} = 0.5$

* LFU (Least Frequently used)

→ LFU stands for least frequently used

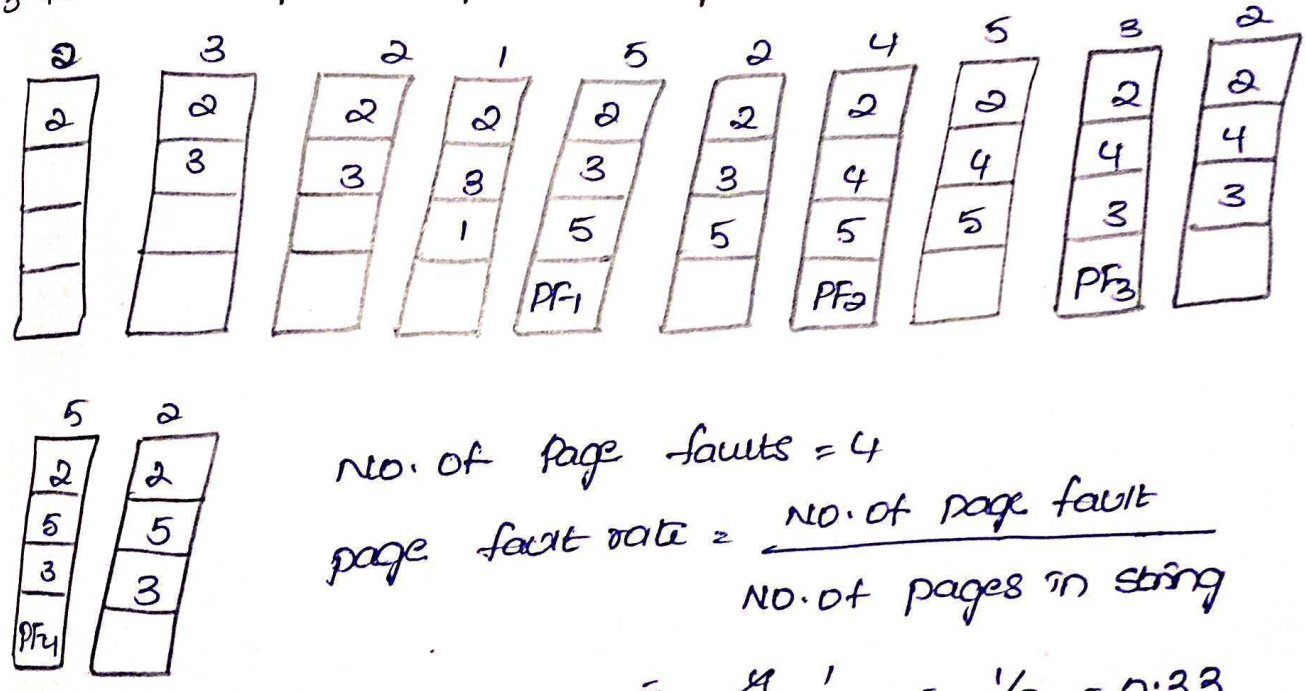
→ It is a page replacement algorithm

→ Page replacement algorithm is used to replace the any one of the page from main memory. on LFU

* Algorithm for LFU

1) Start the process
2) Declare the variables
3) Allocate no. of pages to frames of memory
4) According to LFU, must and should replace the one of the page from the main memory.
5) Replacement is done based on different conditions as
   i) Replacement the pages using least count value pages
   ii) Does not use present replacement pages
   iii) Does not use cache memory pages (longest time pages)
6) Display the result
7) Stop the process

Eg:- Consider a string 2, 3, 2, 1, 5, 2, 4, 5, 3, 2, 5, 2 with 3 frames by using LFU algorithm.



No. of page faults = 4

page fault rate = $\frac{\text{No. of page fault}}{\text{No. of pages in string}}$

$= \frac{4}{12} \; \frac{1}{3} = 1/3 = 0.33$