Ch#: Virtual memory

What is virtual memory?

=> Virtual memory is a storage technique that provides user an illusion of having a very big main memory.

=> This is done by considering a part of secondary memory as the main memory

=> used to execute process that are not in main memory.

=> Virtual memory is a separation of user logical memory from physical memory.

=> In this method, we keep only a part of the process in the memory and rest part on the secondary disk.

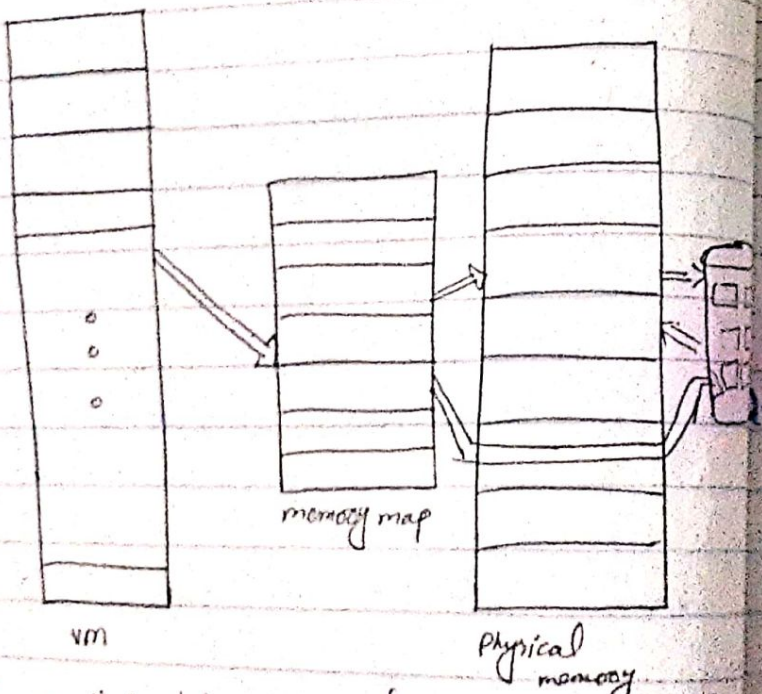
Why virtual memory needed?

- Because main memory has limited size we can do swapping for process replacement, but swapping has a drawback, it decreases the degree of multi-programming.

To overcome this problem, we use the concept of virtual memory.

because it allows instead of loading
fig. process in the main memory, the OS
load the different parts of more than one
process in the main memory.
- By doing so, the degree of multi-programming will be increased and CPU utilization will also be increased.

Diagram:



⇒ It is abstracts physical memory into an extremely large uniform array of storage

Pros:

- Degree of multi-programming increases.
- Bigger sizes programs can easily be executed.
- Reduces the external fragmentation.
- Allows only a part of the program execution
- Allows processes to share files easily and to implement shared memory.
- Less number of I/O would be needed to load or swap each user program into memory.

Cons:

- System becomes more slower because swapping takes time.
- The user will have the lesser hard disk space for its use.

How VM implemented?

- VM in OS is commonly implemented by "demand paging"
- Can also be implemented in a "Segmentation system."

Demand Paging

Definition:

- Demand paging is a technique where pages are loaded from secondary memory to main memory only when they are demanded during program execution. Pages that are never demanded.
- it is a combination of paging & swapping.
- Also known as Lazy swapper because swapping is done only when CPU requires

Main requirement:

- Complete program should be present in the backing storage in form of pages.

=> Virtual memory is commonly implemented in demand paging.

=> it decrease the swap time and the amount of physical memory needed.

=> The demand paging system depends on the page table implementation because the page table help map logical memory to physical memory.

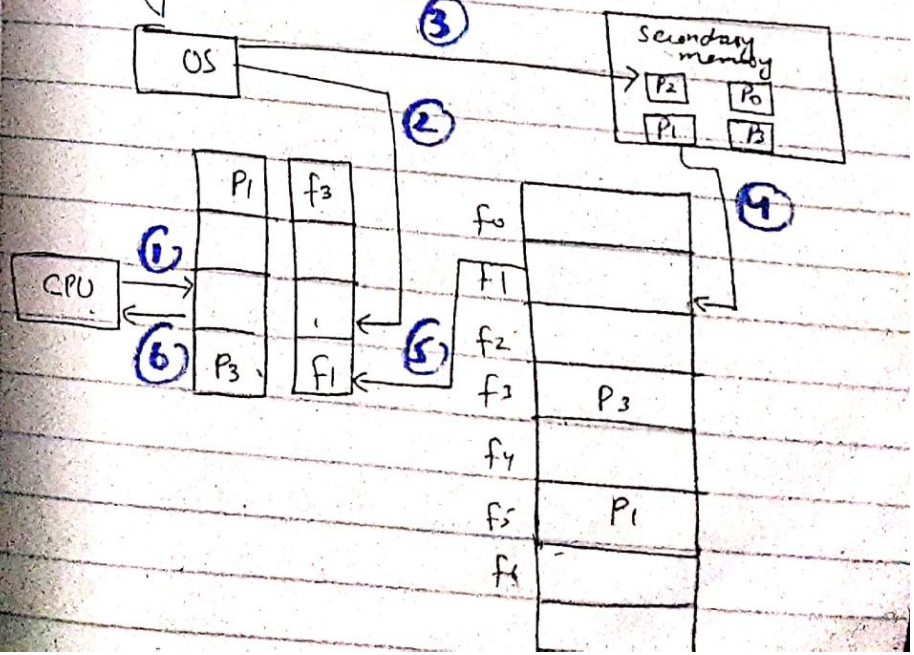
=> Bitwise operators are implemented in the page table to indicate that pages are valid or invalid.

=> All valid pages exist in primary memory.

=> All invalid pages are exist in secondary memory.

Examples:-

- Suppose we have to execute a process P having four pages P_0, P_1, P_2 and P_3. Currently we have Pages P_1 and P_3 in page table.



1) If the CPU wants to access page P of a process P, it will first search page in the page table.
2) As the page table does not contain this page so it will be a trap or page fault.
3) The Control goes to the operating system as soon as the trap is generated and Context Switching happens.
4) The OS will put the process in a waiting/blocked state. The OS will now search that page in the Secondary memory.
5) The OS will then read the page from the backing store and load it to the main memory.
6) Next, the OS will update the page table entry accordingly.
7) Finally, the Control is taken back from the OS, and the execution of the process is resumed.

=> Hence, the OS follows these steps when a page fault occurs and the required page is brought into memory.

Effective memory access time:

=> When the page fault rate is 'P' while executing any process, then the effective memory access time is calculated as follows:

EMA = (P) * (S) + (1-P) * m

where:

- P is the page fault rate.
- S is the page fault service time.
- m is the main memory access time.

=> EAT $\propto$ Page fault rate.

Pros

- memory is utilized efficiently.
- Avoids external fragmentation.
- Less I/o needed for demand paging.
- This process is not constrained by the size of physical memory.
- It becomes easier to share the page.

- With this technique, pages of the process that are never called are never loaded.
- No Compaction is required in demand paging

Cons:

- There is an increase in overheads due to interrupts and page tables.
- Memory access time in demand paging is long.

Q. What is page fault?

- page fault dominates more like error. it mainly occurs when any program tries to access the data or the code that is in the address space of the program, but that data is not currently located in the RAM of the system.
- So basically when the page referenced by the CPU is not found in the main memory then this situation is termed as page fault.
- Whenever any page fault occurs, then the required page has to be fetched from the secondary memory into the main memory.

Page Replacement

- Page replacement is a process of swapping out an existing page from the frame of a main memory and replacing it with the required page.
- Page replacement is required when:-
  - All the frames of main memory are already occupied.

Page replacement algorithm

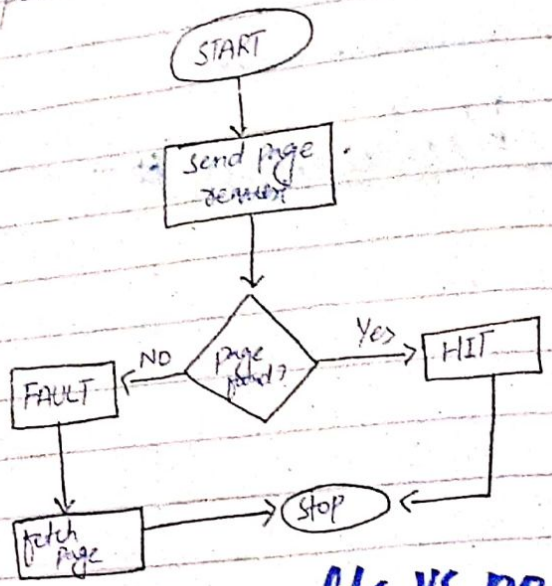
- The algorithms used to select the page to be replaced are called page replacement algorithms.
- The efficiency lies in the least time that is needed for a page to be paged in

Types:

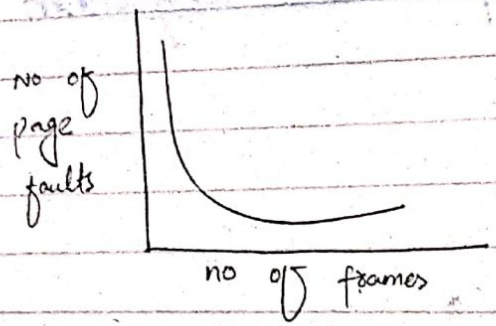
- Local page replacement (Local strategy)
- Global page replacement strategy

Why we need?

- The main goal of page replacement algo. is to provide lowest page fault rate.



No of page faults vs no of frames:



Algorithms:

- FIFO
- LRU
- LFU (Counting based algorithm)
- Second chance
- Optimal replacement
- MRU

i) FIFO:

- FIFO stands for first in first out page replacement algorithm.
- This is the simplest page replacement algorithm.
- In this OS keeps track of all pages in the memory in a queue, oldest page is in the front of the queue.
- When a page needs to be replaced page in the front of the queue is selected for removal.
- Not effective way

Pros:

- Simple and easy to use
- FIFO does not cause overhead.

Cons:

- Poor performance
- Does not use the frequency of the last used time and just replaces the oldest page.
- Suffers from Belady's anomaly.
- Inefficient use of memory

Example:

- Reference String : 1, 3, 0, 3, 5, 6
- no of page slots: 3

P	1	3	0	3	5	6
f1	1	1	1	1	5	5
f2	-	3	3	3	3	6
f3	-	-	0	0	0	0
	X	X	X	V	X	X

no of page faults = 5

no of hit = 1

page fault probability = 5/6

2) Optimal Replacement:

- In this algorithm, pages are replaced which are not used for the longest duration of time in the future.
- This algorithm replaces the page which will not be referred for so long in future.

Pros:

- Excellent efficiency
- Less Complexity
- Simple data structure can be used to implement.
- Less no of page faults.
- Can be used to measure the performance of other algorithm.

Cons:

- More time Consuming
- Difficult for error handling
- Need future awareness of the program, which is not possible every time.

Example:

Reference String: 7, 0, 1, 2 0 3 0 4 2 3 0 3 2

No of Page slots: 4

P	7	0	1	2	0	3	0	4	2	3	0	3	2
f1	7	7	7	7	7	3	3	3	3	3	3	3	3
f2	-	0	0	0	0	0	0	0	0	0	0	0	0
f3	-	-	1	1	1	1	1	4	4	4	4	4	4
f4	-	-	-	2	2	2	2	2	2	2	2	2	2
	X	X	X	X	✓	X	✓	X	✓	✓	✓	✓	✓

no of page faults = 6
no of hit = 7
no of total pages = 13

(3) LRU

- page will be replaced which is least recently used.
- it replace the page that has not been referenced for the longest time.
- LRU works on the idea that pages that have been most heavily used in the past few instructions are most likely to be used heavily in the next few instructions too.
- we are looking in the past which page has not been used longest time.

Pros

- Does not suffer from Belady's anomaly.
- it is open for full analysis
- more efficient

Cons:

- Expensive and more Complexity.
- Need for additional data structure.

Example:

Reference String: 7 0 1 2 0 3 0 4 2 3 0 2
No of page slots: 4

P	7	0	1	2	0	3	0	4	2	3	0	3	2
f1	7	7	7	7	7	3	3	3	3	3	3	3	3
f2	-	0	0	0	0	0	0	0	0	0	0	0	0
f3	-	-	1	1	1	1	1	4	4	4	4	4	4
f4	-	-	-	2	2	2	2	2	2	2	2	2	2
	X	X	X	X	✓	X	✓	X	✓	✓	✓	✓	✓

no of page fault: 6
no of hit: 7

(4) LFU :-

- In LFU, the page with the minimum count / frequency is selected for replacement with the page that needs to enter into the system.
- A counter is assigned to every page that is loaded into the memory.
- LFU algorithm is sometimes also combined with LRU replacement algorithm & then implemented.
- if the frequency of pages remains constant, the page that comes first is replaced first.

Example:

no of page slot = 3

Reference String: 7 0 1 2 0 3 0 4 2 3 0 3 2

P.	7	0	1	2	0	3	0	4	2	3	0	3
f1	7	7	7	2	2	2	2	4	4	3	3	1
f2	-	0	0	0	0	0	0	0	0	0	0	0
f3	-	-	1	1	1	3	3	3	2	2	2	2
	x	x	x	x	✓	x	✓	x	x	x	✓	✓

Frequency: 0 = 2 3 4 | 1 = 1 0 | 2 = 1 0 1 2 | 3 = 1 0 1 2 | 4 = 1

(5) MRU

- Also known as fetch-and-discard algorithm.
- Deal with the case such as sequential scanning when pattern

Example:

RS : 7 0 1 2 0 3 0 4 2 3 0 3 2 1 2

frame: 3

P	7	0	1	2	0	3	0	4	2	3	0	3	2	1	2
f1	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7
f2	-	0	0	0	0	3	0	4	4	4	4	4	4	4	4
f3	-	-	1	2	2	2	2	2	2	3	0	3	2	1	2
	x	x	x	x	✓	x	x	x	✓	x	x	x	x	x	✓

no of page fault: 13

no of hit = 2

(6) Second Chance Algorithm

- Modification of FIFO replacement algorithm
- Drawback in FIFO is that even if page is in use, it may be replaced due to its arrival time
- A reference bit or use bit is used to give information about regarding whether page has been used.
- Initially reference bit for all pages are 0.
- As the page is referenced, bit is set to 1 to indicate that the page is in use.
- A page whose reference bit 1 will not be replaced, and will be given second chance

Example:

Frame = 3

Reference string: 2 3 2 1 5 2 4 5 3

frame	2	3	2	1	5	2	4	5	3
f1	2^0	2^0	2^1	2^1	2^0	2^1	2^0	2^0	3^0
f2		3^0	3^0	3^0	5^0	5^0	5^0	5^1	5^0
f3	-	-	1^0	1^0	1^0	1^0	4^0	4^0	4^0
	✗	✗	✓	✗	✗	✓	✗	✓	✗

PF = 6, Hit = 3, PF = 6/9

Belady's Anomaly

Also called FIFO anomaly.

- Belady's anomaly is the phenomenon of increasing the number of page faults on increasing the number of frames in main memory. it is not always happen but for some reference string
- Following algorithms suffer from Belady's anomaly:
  - FIFO
  - Random page replacement
  - Second Chance
- An algorithm suffers from belady's anomaly iff it does not follow stack property.

#Why stack based algorithm do not suffer from belady's anomaly?

- Because these type of algorithms assign a priority to a page that is independent of the number of page frames.
- A stack-based algorithm is one for which it can be shown that the set of pages in memory for N frames is always a subset of the set of pages that would be in memory with N+1 frames.

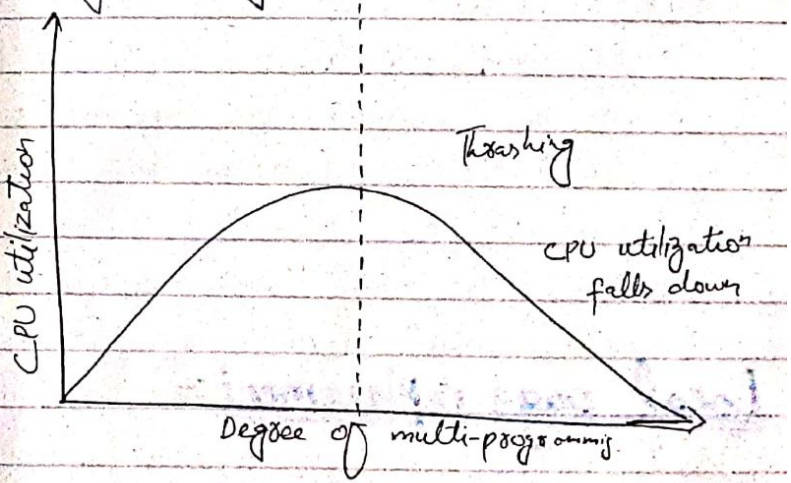
- if the number of frames increases then these n pages will still be the most recently referenced pages.
- While in FIFO, if a page b came in physical memory before a page - a then priority of replacement of b is greater than a, but this is not independent the number of page frames.

=> Because stack based algorithms make their page replacement decisions based on future references rather than simply relying on past references like the FIFO.

Thrashing

=> A situation when system is spending most of the time in servicing page faults (swapping) than executing the actual process.

=> In degree of multiprogramming, Thrashing occurs because page faults are more in degree of multi-programming, then CPU spend most of his time in swapping the pages between main memory and the secondary memory.



Causes:

- When more processes in memory as compared to the available no of frames in memory. OR Lack of frames
- High degree of multi-programming.
- Page replacement policy.

Effect of Thrashing:-

⇒ At the time, when Thrashing starts then the operating system tries to apply either the
- Global page replacement algorithm
- Local page replacement algorithm

Global page replacement algorithm

- Global page replacement can bring any page, it tries to bring more pages whenever thrashing is found.
- But what actually will happen is that no process gets enough frames, and as a result, the thrashing will increase more and more.
- Therefore the global page replacement algorithm is not suitable when thrashing happens.

Local page replacement:-

- Unlike the global page algorithm, local page replacement will select pages which only belong to that process.
- So there is a chance to reduce the thrashing.
- But it is proven that there are many disadvantages if we use local page replacement.
- Therefore, local page replacement is just an alternative to global page replacement in a thrashing scenario.

Techniques to Handle Thrashing

- We already have local replacement that is better than global replacement to avoid thrashing.
- But it also has disadvantages and not suggestable. Some more techniques are:
  - Working-set model
  - Page fault frequency

Working set model:-

- This model is based on locality model.

Locality model:-

- A locality is a set of pages that are actively used together.
- The locality model states that as a process executes, it moves from one locality

Def: working set model is a conceptual model proposed by Peter Denning to prevent thrashing.

to avoid:

- A program is generally composed of several different localities which may overlap.
- Working set model is a dynamic page replacement algorithm that allocates frames to a particular process assuming that the nearest future of pages will be used is a closed approximation of the recent past pages in memory.
- The model uses a parameter Δ to define the working-set window.

Working set: The set of pages in the most recent page reference.

(working set window) Δ : No of pages in the most recent page reference.

Example:

(i) Page references: 2 6 1 5 7 7 7 5 1 (t1)

Δ = 10

WS(t1) = {1, 2, 5, 6, 7}

(2) Page references = 3 4 4 3 4 3 4 4 4 (t2)

Δ = 10

WS(t2) = {3, 4}

- if a page is in active use, it will be in working set.
- if it is no longer used, it will be dropped from the working set.

Example:

2 6 1 5 7 7 7 5 1, 6 2 3 4 1 2 3 4 4 4 3 4 3 4 4 4 1 3 2 3

Δ = 10

WS(t1) = {1, 2, 5, 6, 7}

=> Through working set we can suspension and resume of the process can be done and prevent from thrashing.

Example:

Assume working set of window set is 4, initially pages in working set {e, d, a} {Assume earlier r-s is ... e, e, d, a ...} what is the no of hits and average no of frames used by working set of next to ref of page is {e, d, a, c, c, d, b, c, e, c, e, a, d}

t1 - {e, d, a, c} - miss - 4

t2 - {d, a, c} - Hit - 3

t3 - {a, c, d} - Hit - 3

t4 - [a, d, b] → miss - 3
t5 - [d, b, c] - H - 3
t6 - [d, b, e, c] - m - 4
t7 - [b, e, c] - H - 3
t8 - [e, c] - H - 2
t9 - [a, e, c] - m - 3
t10 - [c, e, a] - m - 3

(1) No of hits = 5

(2) No of frames on average = $\frac{32}{10}$ = 3.2

D = ws(t1) + ws(t2) + ...
working set size

Disadvantages:

- Monitoring the working set is a very clumsy work.

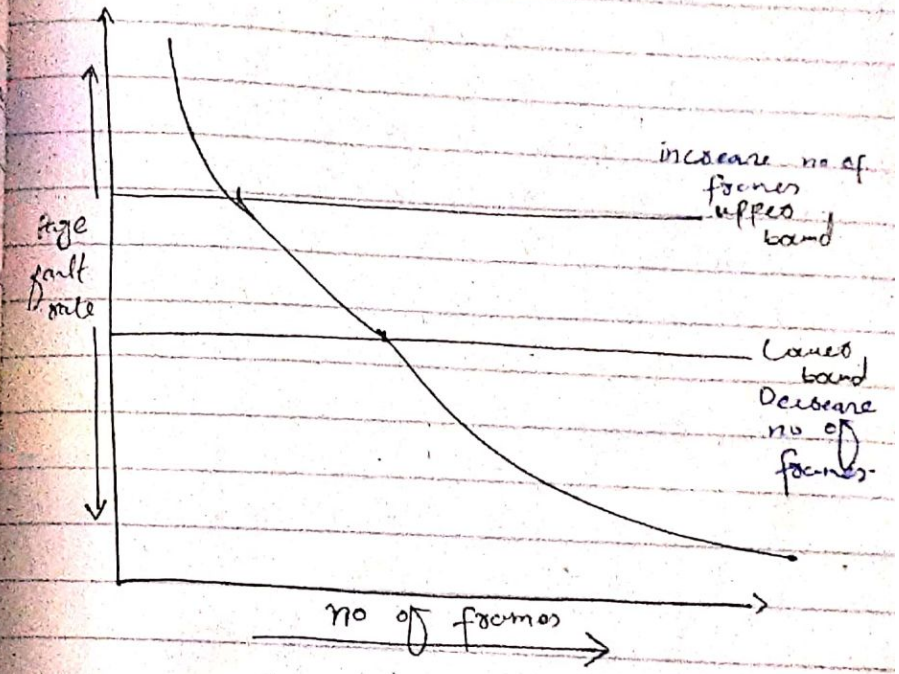
Note:

- if enough empty frames in physical memory then initiate another process.
- D > empty frames then suspend the process

- In working set model processes are divided into two groups:
  - Active
  - Inactive
- When a process is active its entire working set must always be in memory; never execute a thread whose working set is not resident.
- When a process becomes inactive, its working set can migrate to disk.
- Threads from inactive process never scheduled for execution.
- balance set:
- The collection of active processes is called the balance set.
- The system must have a mechanism for gradually moving processes into and out of the balance set.
- As working sets change, the balance set must be adjusted.

2) Page fault frequency:

- it is direct approach for handling thrashing.
- In this method we monitors the page fault rate of each process.
- High page fault rate:
  - indicates that process has very few frames allocated to it.
- Low page fault rate:
  - indicates that process has large no of frames allocated.
- In order to prevent thrashing, upper & lower limits has been set on the desired page fault rate.
- if page fault rate < lower limit
  - frames can be removed from the process
- if page fault rate > upper limit
  - more frames allocate to those process
- if there is no free frames then one of the process to be suspended and frames allocated to these can be reallocated to other process.



- What is belady's anomaly? (repeat)
- FIFO and LRU both use previous information in page replacement policy? How is the one different from another than? (repeat)
- Differentiate blw LFU and MFU algorithm
- What is demand paging? (repeat)
- What is virtual memory? (repeat)
- Under what circumstances do page fault occur?
- How page fault frequency can be used as method of Thrashing?
- What is the cause of Thrashing?
- Why page table is needed?
- What is page fault? (repeat)
- Define hit ratio. (repeat)
- What is reference bit or dirty bit?
- What is Thrashing?
- What causes page fault?

long

- Discuss different Threading issues in detail.

Q# Apply LRU and optimal algorithm.
7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 1, 2, 0, 1, 7, 0, 1
frame size is 3.

Q#2. Apply LRU & optimal algorithm
4, 3, 1, 3, 1, 1, 1, 3, 4, 2, 1, 5, 6, 2, 1, 2, 3, 7, 6, 3, 2, 1, 2, 3, 6
frame size is a) 5 b) 6

Q#3) Apply LRU, FIFO & Optimal.
6, 7, 0, 2, 1, 2, 3, 4, 2, 1, 5, 6, 2, 1, 2, 3, 7, 6, 3, 2, 1, 2, 3, 6
frame size a) 3, b) 4, c) 6

Q#4) Apply LRU, LFU & FIFO.
4, 3, 2, 1, 5, 1, 2, 3, 2, 3, 2, 3, 4, 5, 5, 5
frame size 3.

Q#5) Apply FIFO & optimal.
1, 3, 4, 2, 4, 5, 6, 4, 2, 3, 1, 7, 8
frame size 3.

Q#6) Apply LFU and optimal
1, 2, 3, 4, 1, 2, 3, 5, 1, 3, 2, 1, 4, 5, 6, 3, 2
frame size is 3.

Q#7) Apply LRU, FIFO & optimal
8, 2, 5, 1, 2, 8, 4, 6, 1, 7, 2, 1, 6, 1, 8, 7, 8, 2, 1, 7
frame size a) 3, b) 4, c) 5

CamScanner

Q#8) Apply Optimal Page replacement.
1,2,1,4,3,4,2,1,5,6,2,1,2,3,7,6,3,2,1,2,3
frame size 4 and 7.

Q#9) Apply LRU and FIFO
3,1,9,2,4,8,6,9,2,1,3,7,5,8,1,9,3,4,2
frame size 3.

Q#: What is reference bit or dirty bit?

- It is used in page replacement algorithm
- if no frames are free, two pages transfers are required (one out and one in).
- This situation effectively doubles the page fault service time and increases the effective access time.
- We can reduce this overhead by using a modify bit or dirty bit.
- it is associated with each page or frame
- it indicates whether a page has been modified or not
- it is set by the hardware and used to show the page is modified after being loaded into the cache.

What is reference String?

- An algorithm can be evaluated by running it on a particular string of memory references and computing the number of page faults. The string of memory references is called a reference string.
- it can be generated artificially.

What is equal allocation?

- The easiest way to split m frames among n processes is to give everyone an equal share, m/n frames.
- for instance, if there are 93 frames and five processes, each process will get 18 frames. The three leftover frames can be used as a free frame buffer pool.
- This is called equal allocation.

What is proportional allocation?

- In proportional allocation, we allocate available memory to each process according to its size.
- if total no of frames are m, we allocate ai frames to process pi, where ai is approximately.

$$a_i = s_i / S * m$$

What is pure demand paging?

- In the extreme case, we can start executing a process with no pages in memory.
- When the OS sets the instruction pointer to the first instruction of the process, which is on a non-memory resident page, then page fault occurs immediately.
- Pages are loaded with on demanding of the process. This is called pop.

What is Copy-on-Write?

- Copy-on-write is a benefit of virtual memory
- COW allows both parent and child process to initially share the same page in memory
- if either process modifies a shared page, only then is the page copied.
- COW allows more efficient process creation as only modified pages are copied.
- Free pages are allocated from a pool of zeroed-out pages.
