Virtual Memory

One important goal in now-a-days computing environment is to keep many processes in memory simultaneously to follow multiprogramming, and use resources efficiently especially main memory.

Pure Demand Paging/Demand Paging

- we can start executing a process with no pages in memory. When the operating system sets the instruction pointer to the first instruction of the process, which is on a non-memory-resident page, the process immediately faults for the page. After this page is brought into memory, the process continues to execute, faulting as necessary until every page that it needs is in memory. At that it can execute with no more faults. This scheme is **Pure Demand Paging:** never bring a page into memory until it is required.
- The **Locality of Reference** helps Demand Paging in performing reasonably.
- A demand-paging system is extension to a paging system with swapping.
- But rather than swapping the entire process into memory, we use a **Lazy Swapper**. A lazy swapper never swaps a page into memory unless that page will be needed.
- We use the term **Pager** instead of **swapper** in demand paging.

Q Which of the following is incorrect for virtual memory? **(NET-JAN-2017)**
a) Large programs can be written **b)** More I/O is required
c) More addressable memory available **d)** Faster and easy swapping of process
Answer: (B)

Q Page making process from main memory to disk is called **(NET-DEC-2010)**
(A) Interruption **(B)** Termination **(C)** Swapping **(D)** None of the above
Answer: (C)

Q If the executing program size is greater than the existing RAM of a computer, it is still possible to execute the program if the OS supports: **(NET-DEC-2008)**
(A) multitasking **(B)** virtual memory **(C)** paging system **(D)** none of the above
Answer: (B)

Q Moving Process from main memory to disk is called: **(NET-JUNE-2005)**
(A) Caching **(B)** Termination **(C)** Swapping **(D)** Interruption
Answer: (C)

Advantage

- A program would no longer be constrained by the amount of physical memory that is available, Allows the execution of processes that are not completely in main memory, i.e. process can be larger than main memory.
- More programs could be run at the same time as use of main memory is less.
- Less I/O would be needed to load or swap user programs into memory, so each user program would run faster.
- Virtual memory also allows processes to share files easily and to implement shared memory.

Disadvantages

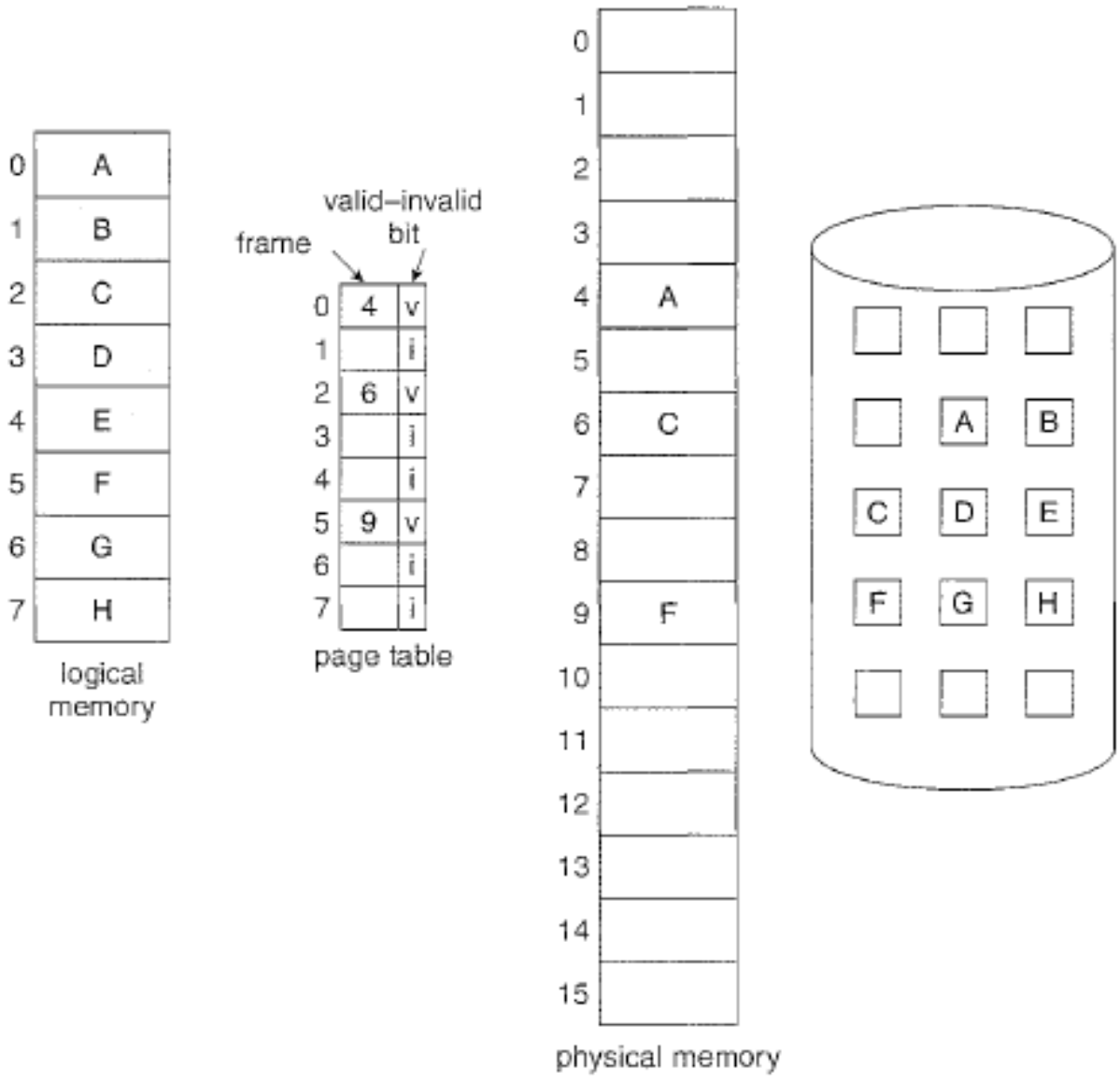
- Virtual memory is not easy to implement.
- It may substantially decrease performance if it is used carelessly (Thrashing)

Q Which of the following memory allocation scheme suffers from external fragmentation? **(NET-DEC-2012)**
(A) Segmentation **(B)** Pure demand paging **(C)** Swapping **(D)** Paging
Answer: (A)

Q Which of the following statements is false? **(GATE-2001) (1 Marks)**
(A) Virtual memory implements the translation of a program's address space into physical memory address space
(B) Virtual memory allows each program to exceed the size of the primary memory
(C) Virtual memory increases the degree of multiprogramming
(D) Virtual memory reduces the context switching overhead
Answer: (A)

Basic Concepts

- When a process is started execution, no page is loaded into main memory before demand so, it avoids reading into memory pages that will not be used anyway, decreasing the swap time and the amount of physical memory needed.
- Now we need some hardware to distinguish between which pages are in memory and which are not, so **valid -invalid bit scheme** can be used for it.
- This time, however, when this bit is set to "**valid**" the associated page is both legal and in memory. If the bit is set to "**invalid**" the page either is not valid (that is, not in the logical address space of the process) or is valid but is currently on the disk.



- The page-table entry for a page that is brought into memory is set as usual but the page-table entry for a page that is not currently in memory is either simply marked invalid.

Page Fault: - When a process tries to access a page that is not in main memory then a **Page Fault Occurs.**

Steps to handle Page Fault

- If the reference was invalid, we terminate the process. If it was valid, but we have not yet brought in that page, we now page it in.
- We find a free frame (by taking one from the free-frame list, for example).
- We schedule a disk operation to read the desired page into the newly allocated frame.
- When the disk read is complete, we modify the internal table kept with the process and the page table to indicate that the page is now in memory.
- We restart the instruction that was interrupted by the trap. The process can now access the page as though it had always been in memory.

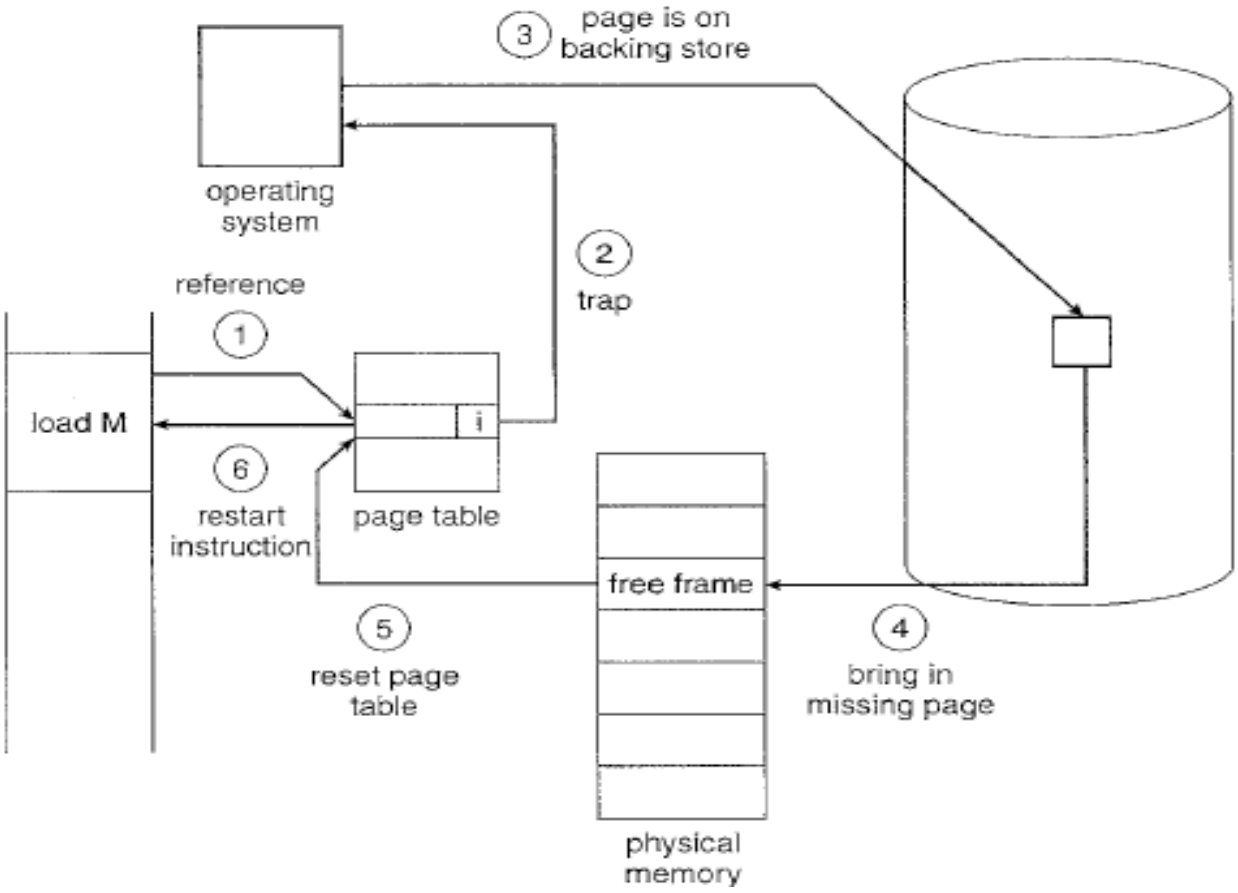


Figure: Page Fault Handling

- A crucial requirement for demand paging is the ability to restart any instruction after a page fault. Because we save the state (registers, condition code, instruction counter) of the interrupted process when a page fault occurs, we must be able to restart the process in exactly the same place and state.

Q A page fault **(NET-DEC-2009)**
(A) is an error specific page.
(B) is an access to the page not currently in memory.
(C) occur when a page program occurs in a page memory.
(D) page used in the previous page reference.
Answer: B

Q A page fault ..... **(NET-JUNE-2006)**
(A) is an error in specific page
(B) is an access to the page not currently in main memory
(C) occurs when a page program accesses a page of memory
(D) is reference to the page which belongs to another program
Answer: (B)

Performance of Demand Paging

- **Effective Access time for Demand Paging:** (1 - p) x ma + p x page fault service time.
- Here, p: Page fault rate or probability of a page fault.
- ma is memory access time.

Q In a paged memory, the page hit ratio is 0.40. The time required to access a page in secondary memory is equal to 120 ns. The time required to access a page in primary memory is 15 ns. The average time required to access a page is _____. **(NET-JULY-2018)**

(a) 105 **(b)** 68 **(c)** 75 **(d)** 78

Answer: (D)

Q Consider a process executing on an operating system that uses demand paging. The average time for a memory access in the system is M units if the corresponding memory page is available in memory, and D units if the memory access causes a page fault. It has been experimental measured that the average time taken for a memory access in the process is X units.

Which one of the following is the correct expression for the page fault rate experienced by the process? **(GATE-2018) (2 Marks)**

(A) (D – M) / (X – M) **(B)** (X – M) / (D – M)

(C) (D – X) / (D – M) **(D)** (X – M) / (D – X)

Answer: (B)

Q Suppose that the number of instructions executed between page fault is directly proportional to the number of page frames allocated to a program. If the available memory is doubled, the mean interval between page faults is also doubled. Further, consider that a normal instruction takes one microsecond, but if a page fault occurs, it takes 2001 microseconds. If a program takes 60 sec to run, during which time it gets 15,000-page faults, how long would it take to run if twice as much memory were available**? (NET-DEC-2015)**

A) 60 sec **b)** 30 sec **c)** 45 sec **d)** 10 sec

Answer: (C)

Q In a demand paging memory system, page table is held in registers. The time taken to service a page fault is 8 msec. if an empty frame is available or if the replaced page is not modified, and it takes 20 msec., if the replaced page is modified. What is the average access time to service a page fault assuming that the page to be replaced is modified 70 of the time? **(NET-DEC-2014)**

(A) 11.6 msec. **(B)** 16.4 msec. **(C)** 28 msec. **(D)** 14 msec.

Answer: (B)

Q Virtual memory is **(NET-JUNE-2011)**

(A) related to virtual reality **(B)** a form of ROM

(C) a form of RAM **(D)** None of the above

Answer: (C)

Q Let the page fault service time be 10ms in a computer with average memory access time being 20ns. If one-page fault is generated for every 10^6 memory accesses, what is the effective access time for the memory? **(GATE-2011) (1 Marks)**

(A) 21ns **(B)** 30ns **(C)** 23ns **(D)** 35ns

Answer: (B)

Q The memory allocation scheme subjected to "external" fragmentation is: (**NET-JUNE-2006)**

(A) Segmentation **(B)** Swapping

(C) Demand paging **(D)** Multiple contiguous fixed partitions

Answer: (A)

Q Suppose the time to service a page fault is on the average 10 milliseconds, while a memory access takes 1 microsecond. Then a 99.99% hit ratio results in average memory access time of? (GATE-2000) (1 Marks).

(A)1.9999 milliseconds **(B)**1 millisecond

(C)9.999 microseconds **(D)**1.9999 microseconds

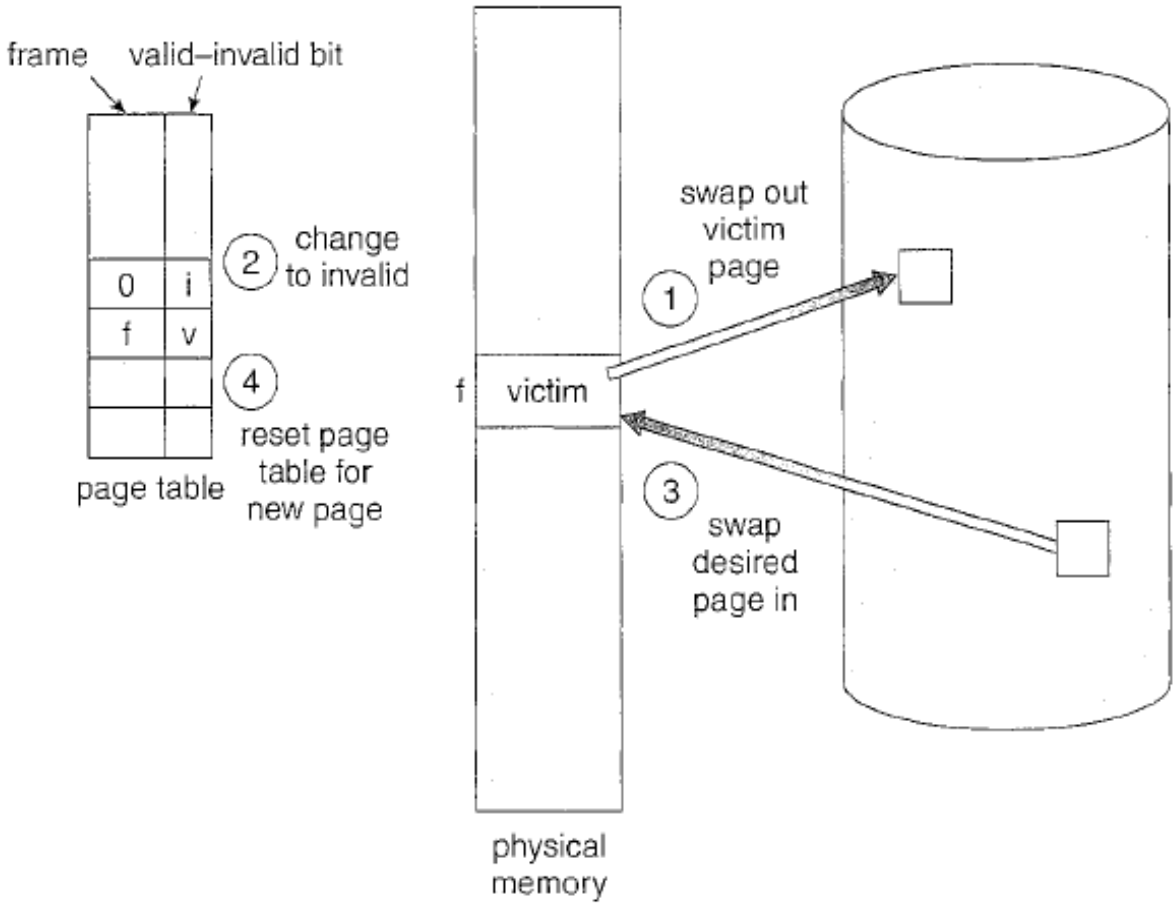
Answer: (D)

Page Replacement

With the increase in multiprogramming each process will get less amount of space in main memory so, the rate of page faults may rise.

thus, to reduce the degree of multiprogramming the operating system swaps out processes from the memory freeing the frames and thus the process that requires to execute can now execute.

- If no frames are free, two-page transfers (one out and one in) are required. This situation effectively doubles the page-fault service time and increases the effective access time accordingly.
- We can reduce this overhead by using a **Modify bit or Dirty Bit**. When this scheme is used, each page or frame has a modify bit associated with it in the hardware.
- The modify bit for a page is set by the hardware whenever any word or byte in the page is written into, indicating that the page has been modified.
- **If the bit is set**: the page has been modified since it was read in from the disk. In this case, we must write the page to the disk.
- **If the modify bit is not set**: however, the page has not been modified since it was read into memory. In this case, we need not write the memory page to the disk: it is already there.



Now, we must solve two major problems to implement demand paging: We must be able to develop a **frame allocation algorithm** and a **page replacement algorithm**.

- Frame Allocation will decide how much frames to allocate to a process.
- Page Replacement will decide which page to replace next.

Q Dirty bit is used to show the **(NET-DEC-2018)**
a) Wrong Page
b) Page with corrupted data
c) Page with low frequency occurrence
d) Page that is modified after being loaded into cache memory
Answer: (D)

Q A virtual memory has a page size of 1K words. There are eight pages and four blocks. The associative memory page table contains the following entries:

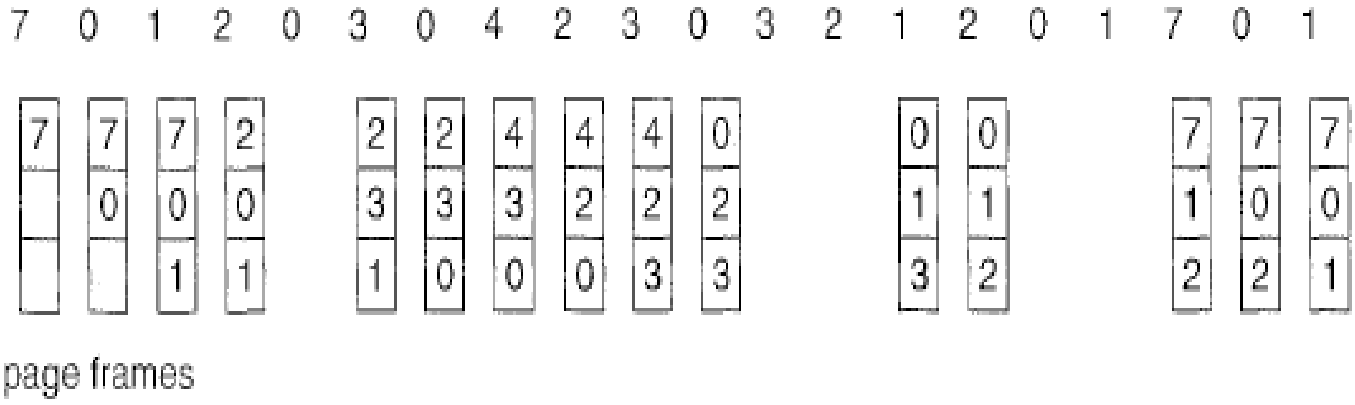
Page	Block
0	3
2	1
5	2
7	0

Which of the following list of virtual addresses (in decimal) will not cause any page fault if referenced by the CPU? **(NET-DEC-2015)**
a) 1024, 3072, 4096, 6144
b) 1234, 4012, 5000, 6200
c) 1020, 3012, 6120, 8100
d) 2021, 4050, 5112, 7100
Answer: (C)

Page Replacement Algorithms

First In First Out Page Replacement Algorithm: - A FIFO replacement algorithm associates with each page, the time when that page was brought into memory. When a page must be replaced, the oldest page is chosen. i.e. the first page that came into the memory will be replaced first.

Example:



- In the above example the number of page fault is 15.
- The FIFO page-replacement algorithm is easy to understand and program. However, its performance is not always good.
- **Belady's Anomaly:** for some page-replacement algorithms, the page-fault rate may increase as the number of allocated frames increases.

Q Suppose for a process P, references to pages occur in order are 1, 2, 4, 5, 2, 1, 2, 4. Assume that the main memory can accommodate 3 pages and the main memory already has the pages 1 and 2 in the order 1-first, 2-second. At this moment, assume fifo page replacement algorithm is used then the number of page faults that occur to complete the execution of process P is (NET-DEC-2018)

(A) 4 **(B)** 3 **(C)** 5 **(D)** 6

Answer: (C)

Q. In which one of the following page replacement algorithms it is possible for the page fault rate to increase even when the number of allocated frames increases? (GATE-2016) (1 Marks)

(1) LRU (Least Recently Used **(2)** OPT (Optimal Page Replacement)

(3) MRU (Most Recently Used) **(4)** FIFO (First In First Out)

Answer: (D)

Q Consider the reference string 0 1 2 3 0 1 4 0 1 2 3 4 If FIFO page replacement algorithm is used, then the number of page faults with three-page frames and four-page frames are _____ and _____ respectively. **(NET-JUNE-2016)**

a) 10, 9 **b)** 9, 9 **c)** 10, 10 **d)** 9,10

Answer: (D)

Q Consider the following page trace: 4, 3, 2, 1, 4, 3, 5, 4, 3, 2, 1, 5. Percentage of page fault that would occur if FIFO page replacement algorithm is used with number of frames for the JOB m = 4 will be **(NET-JUNE-2012)**

(A) 8 **(B)** 9 **(C)** 10 **(D)** 12

Answer: (C)

Q A system uses FIFO policy for page replacement. It has 4-page frames with no pages loaded to begin with. The system first accesses 100 distinct pages in some order and then accesses the same 100 pages but now in the reverse order. How many page faults will occur? **(GATE-2010) (1 Marks)**

(A) 196 **(B)** 192 **(C)** 197 **(D)** 195

Answer: (A)

Q In which one of the following page replacement policies, Belady's anomaly may occur? **(GATE-2009) (1 Marks)**

(A) FIFO **(B)** Optimal **(C)** LRU **(D)** MRU

Answer: (A)

Q A virtual memory system uses First in First Out (FIFO) page replacement policy and allocates a fixed number of frames to a process. Consider the following statements: **(GATE-2007) (2 Marks)**

P: Increasing the number of page frames allocated to a process sometimes increases the page fault rate.

Q: Some programs do not exhibit locality of reference.

Which one of the following is TRUE?

(A) Both P and Q are true, and Q is the reason for P

(B) Both P and Q are true, but Q is not the reason for P.

(C) P is false, but Q is true

(D) Both P and Q are false

Answer: (B)

Q Which page replacement policy suffers from Belady's anomaly? **(NET-JUNE-2007)**
(A) LRU **(B)** LFU **(C)** FIFO **(D)** OPTIMAL
Answer: (C)

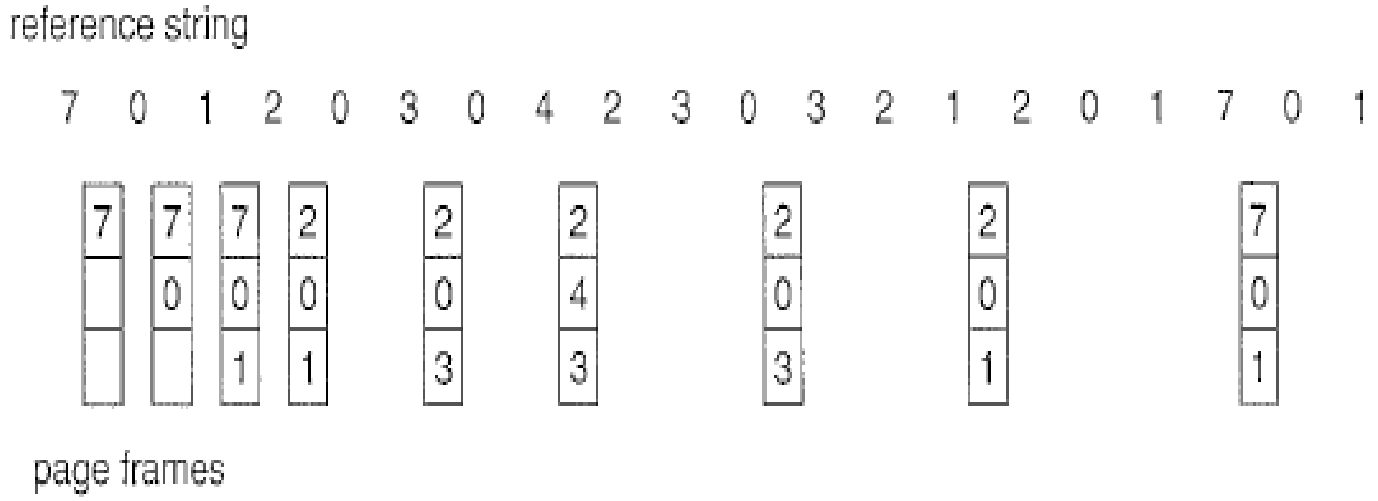
Q A program has five virtual pages, numbered from 0 to 4. If the pages are referenced in the order 012301401234, with three-page frames, the total number of page faults with FIFO will be equal to: **(NET-DEC-2007)**
(A) 0 **(B)** 4 **(C)** 6 **(D)** 9
Answer: (D)

Q Consider a virtual memory system with FIFO page replacement policy. For an arbitrary page access pattern, increasing the number of page frames in main memory will (GATE-2001) (1 Marks)
(A) always decrease the number of page faults
(B) always increase the number of page faults
(C) sometimes increase the number of page faults
(D) never affect the number of page faults
Answer: (C)

Optimal Page Replacement Algorithm

- Replace the page that will not be used for the longest period of time.
- It has the lowest page-fault rate of all algorithms and will never suffer from Belady's anomaly.
- Use of this page-replacement algorithm guarantees the lowest possible page fault rate for a fixed number of frames.

Example:



- The number of page faults occurred: 9
- Unfortunately, the optimal page-replacement algorithm is difficult to implement, because it requires future knowledge of the reference string.
- It is mainly used for comparison studies.

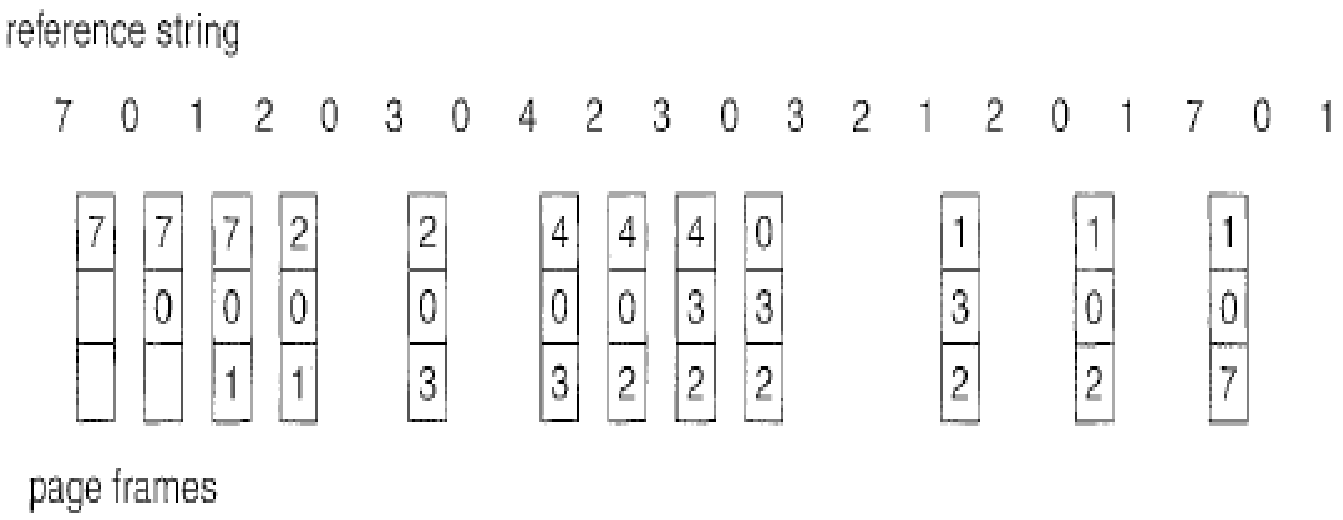
Q Assume that there are 3-page frames which are initially empty. If the page reference string is 1, 2, 3, 4, 2, 1, 5, 3, 2, 4, 6, the number of page faults using the optimal replacement policy is_____. **(GATE-2014) (2 Marks)**
Answer: 7

Q A process has been allocated 3-page frames. Assume that none of the pages of the process are available in the memory initially. The process makes the following sequence of page references (reference string): 1, 2, 1, 3, 7, 4, 5, 6, 3, 1. If optimal page replacement policy is used, how many page faults occur for the above reference string? **(GATE-2007) (2 Marks)**
(A) 7 **(B)** 8 **(C)** 9 **(D)** 10
Answer: (A)

Q The optimal page replacement algorithm will select the page that(GATE-2002) (1 Marks)
(A) Has not been used for the longest time in the past.
(B) Will not be used for the longest time in the future.
(C) Has been used least number of times.
(D) Has been used most number of times.
Answer: (B)

Least Recently Used (LRU) Page Replacement Algorithm

- Replace the page that has not been used for the longest period of time.
- We can think of this strategy as the optimal page-replacement algorithm looking backward in time, rather than forward.



- LRU gives us 12-page faults.
- LRU is much better than FIFO replacement.
- The LRU policy is often used as a page-replacement algorithm and is considered to be good.
- LRU also does not suffer from Belady's Anomaly.
- The major problem is how to implement LRU replacement. An LRU page-replacement algorithm may require substantial hardware assistance.
- LRU and OPT belong to a class of page-replacement algorithms, called **stack algorithms** and can never exhibit Belady's anomaly.
- A stack algorithm is an algorithm 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.

Q Consider a virtual page reference string 1, 2, 3, 2, 4, 2, 5, 2, 3, 4. Suppose LRU page replacement algorithm is implemented with 3-page frames in main memory. Then the number of page faults are_____. **(NET-JULY-2018)**

(a) 5 **(b)** 7 **(c)** 9 **(d)** 10

Answer: (B)

Q Consider a virtual page reference string 7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 1, 2, 0, 1, 7, 0, 1 Suppose a demand paged virtual memory system running on a computer system such that the main memory has 3 page frames. Then _____ page replacement algorithm has minimum number of page faults. **(NET-NOV-2017)**

A) FIFO **b)** LIFO **c)** LRU **d)** Optimal

Answer: (D)

Q Recall that Belady's anomaly is that the page-fault rate may increase as the number of allocated frames increases. Now, consider the following statements:

S1: Random page replacement algorithm (where a page chosen at random is replaced) suffers from Belady's anomaly

S2: LRU page replacement algorithm suffers from Belady's anomaly

Which of the following is CORRECT? **(GATE-2017) (1 Marks)**

(1)S1 is true, S2 is true **(2)**S1 is true, S2 is false

(3)S1 is false, S2 is true **(4)**S1 is false, S2 is false

Answer: (B)

Q Consider the following page reference string : 1, 2, 3, 4, 2, 1, 5, 6, 2, 1, 2, 3, 7, 6, 3, 2, 1, 2, 3, 6 Which of the following options, gives the correct number of page faults related to LRU, FIFO, and optimal page replacement algorithms respectively, assuming 05 page frames and all frames are initially empty ? **(NET-AUG-2016)**

a) 10, 14, 8 **b)** 8, 10, 7 **c)** 7, 10, 8 **d)** 7, 10, 7

Answer: (B)

Q Suppose that the virtual Address space has eight pages and physical memory with four-page frames. If LRU page replacement algorithm is used, _____ number of page faults occur with the reference string.

0 2 1 3 5 4 6 3 7 4 7 3 3 5 5 3 1 1 1 7 2 3 4 1 **(NET-AUG-2016)**

(A) 13 **(B)** 12 **(C)** 11 **(D)** 10

Answer: (A)

Q Consider a main memory with five page frames and the following sequence of page references: 3, 8, 2, 3, 9, 1, 6, 3, 8, 9, 3, 6, 2, 1, 3. Which one of the following is true with respect to page replacement policies First-In-First Out (FIFO) and Least Recently Used (LRU)? **(GATE-2015) (2 Marks)**

(A) Both incur the same number of page faults

(B) FIFO incurs 2 more-page faults than LRU

(C) LRU incurs 2 more-page faults than FIFO

(D) FIFO incurs 1 more page faults than LRU

Answer: (A)

Q A LRU page replacement is used with four page frames and eight pages. How many page faults will occur with the reference string 0172327103 if the four frames are initially empty? **(NET-JUNE-2015)**

a) 6 **b)** 7 **c)** 8 **d)** 5

Answer: (B)

Q A system uses 3-page frames for storing process pages in main memory. It uses the Least Recently Used (LRU) page replacement policy. Assume that all the page frames are initially empty. What is the total number of page faults that will occur while processing the page reference string given below?

4, 7, 6, 1, 7, 6, 1, 2, 7, 2 (**GATE-2014) (2 Marks)**

Answer: 6

Q Consider a program that consists of 8 pages (from 0 to 7) and we have 4-page frames in the physical memory for the pages. The page reference string is: 1 2 3 2 5 6 3 4 6 3 7 3 1 5 3 6 3 4 2 4 3 4 5 1. The number of page faults in LRU and optimal page replacement algorithms are respectively (without including initial page faults to fill available page frames with pages): **(NET-JUNE-2014)**

(A) 9 and 6 **(B)** 10 and 7 **(C)** 9 and 7 **(D)** 10 and 6

Answer: (B)

Q A job has four pages A, B, C, D and the main memory has two-page frames only. The job needs to process its pages in following order: ABACABDBACD. Assuming that a page interrupt occurs when a new page is brought in the main memory, irrespective of whether the page is swapped out or not. The number of page interrupts in FIFO and LRU page replacement algorithms are **(NET-JUNE-2013)**

(A) 9 and 7 **(B)** 7 and 6 **(C)** 9 and 8 **(D)** 8 and 6

Answer: (C)

Q Consider the virtual page reference string

1, 2, 3, 2, 4, 1, 3, 2, 4, 1

On a demand paged virtual memory system running on a computer system that main memory size of 3 pages frames which are initially empty. Let LRU, FIFO and OPTIMAL denote the number of page faults under the corresponding page replacements policy. Then **(GATE-2012) (2 Marks)**

(A) OPTIMAL < LRU < FIFO **(B)** OPTIMAL < FIFO < LRU

(C) OPTIMAL = LRU **(D)** OPTIMAL = FIFO

Answer: (B)

Q A process, has been allocated 3-page frames. Assume that none of the pages of the process are available in the memory initially. The process makes the following sequence of page references (reference string): 1,2,1,3,7,4,5,6,3,1

Least Recently Used (LRU) page replacement policy is a practical approximation to optimal page replacement. For the above reference string, how many more page faults occur with LRU than with the optimal page replacement policy? **(GATE-2007) (2 Marks)**

(A) 0 **(B)** 1 **(C)** 2 **(D)** 3

Answer: (C)

Solution: Two implementations are feasible:

- **Counters**. In the simplest case, we associate with each page-table entry a time-of-use field and add to the CPU a logical clock or counter. The clock is incremented for every memory reference. Whenever a reference to a page is made, the contents of the clock registers are copied to the time-of-use field in the page-table entry for that page. In this way, we always have the "time" of the last reference to each page. We replace the page with the smallest time value.

- **Stack:** Another approach to implementing LRU replacement is to keep a stack of page numbers. Whenever a page is referenced, it is removed from the stack and put on the top. In this way, the most recently used page is always at the top of the stack and the least recently used page is always at the bottom.

Counting-Based Page Replacement

We keep a counter of the number of references that have been made to each page and develop the following two schemes.

- The **least frequently used (LFU)** page-replacement algorithm requires that the page with the smallest count be replaced.
- The reason for this selection is that an actively used page should have a large reference count.
- A problem arises, however, when a page is used heavily during the initial phase of a process but then is never used again. Since it was used heavily, it has a large count and remains in memory even though it is no longer needed.
- One solution is to shift the counts right by 1 bit at regular intervals, forming an exponentially decaying average usage count.
- **The most frequently used (MFU)** page-replacement algorithm is based on the argument that the page with the smallest count was probably just brought in and has yet to be used.
- Neither MFU nor LFU replacement is common. The implementation of these algorithms is expensive, and they do not approximate OPT replacement well.

Q Consider a computer system with ten physical page frames. The system is provided with an access sequence ($a_1,a_2,\ldots,a_{20},a_1,a_2,\ldots,a_{20}$), where each a_i is a distinct virtual page number. The difference in the number of page faults between the last-in-first-out page replacement policy and the optimal page replacement policy is _____. **(GATE-2016) (1 Marks)**
Answer: 1

Q Consider a main memory with 3 page frames for the following page reference string : 5, 4, 3, 2, 1, 4, 3, 5, 4, 3, 4, 1, 4. Assuming that the execution of process is initiated after loading page 5 in memory, the number of page faults in FIFO and second chance replacement respectively are **(NET-SEP-2013)**
(A) 8 and 9 **(B)** 10 and 11 **(C)** 7 and 9 **(D)** 9 and 8
Answer: (D)

Q A computer has twenty physical page frames which contain pages numbered 101 through 120. Now a program accesses the pages numbered 1, 2, …, 100 in that order, and repeats the access sequence THRICE. Which one of the following page replacement policies experiences the same number of page faults as the optimal page replacement policy for this program? **(GATE-2014) (2 Marks)**
(A) Least-recently-used **(B)** First-in-first-out

(C) Last-in-first-out
(D) Most-recently-used
Answer: (D)

Q Increasing the RAM of a computer typically improves performance because **(GATE-2005) (1 Marks) (ISRO-2015)**
(A) Virtual memory increases
(B) Larger RAMs are faster
(C) Fewer page faults occur
(D) Fewer segmentation faults occur
Answer: (C)

Frame Allocation Algorithms

Minimum Number of frames to be allocated to each process depends on Instruction Set Architecture, and the maximum number of allocation of frames depends on the size of the process.

Equal Allocation: Frames will be equally allocated to each process. Ex: if we have 30 frames and 3 processes, each will get 10 regardless of their size.

Weighted / Proportional Allocation: Frames will be allocated according to the weights of the process. The allocation of frames per process: **a(i)= s(i)/S x m.**

Where, a(i) is the number of allocated frames, s(i) is the process and S is the size of the virtual memory in general, **S = Σ s(i),** and m is the total number of frames available.

Example: If we have 3 processes of size 20 k, 30k and 40 k then frames will be allocated as:

P1 gets: 20/100*30 = 6 frames, similarly P2 and P3 will get 9 and 15 frames respectively.

Some points to remember

- If the multiprogramming level is increased, each process will lose some frames to provide the memory needed for the new process.
- Conversely, if the multiprogramming level decreases, the frames that were allocated to the departed process can be spread over the remaining processes.

Global vs Local Frame Allocation

- With multiple processes competing for frames, we can classify page-replacement algorithms into two broad categories:
    - **Global Replacement**
    - **Local Replacement**

Global Replacement: Global replacement allows a process to select a replacement frame from the set of all frames, even if that frame is currently allocated to some other process; that is, one process can take a frame from another.

- With global replacement, a process may happen to select only frames allocated to other processes, thus increasing the number of frames allocated to it.
- One problem with a global replacement algorithm is that a process cannot control its own page-fault rate. As The set of pages in memory for a process depends not only on the paging behaviour of that process but also on the paging behaviour of other processes.
- Global replacement generally results in greater system throughput and is therefore the more common method.

Local Replacement: Local replacement requires that each process select from only its own set of allocated frames.

- With a local replacement strategy, the number of frames allocated to a process does not change.
- Under local replacement, the set of pages in memory for a process is affected by the paging behaviour of only that process.
- Local replacement might hinder a process, however, by not making available to it other, less used pages of memory.

Thrashing

- A process is thrashing if it is spending more time paging than executing. High paging activity is called Thrashing.

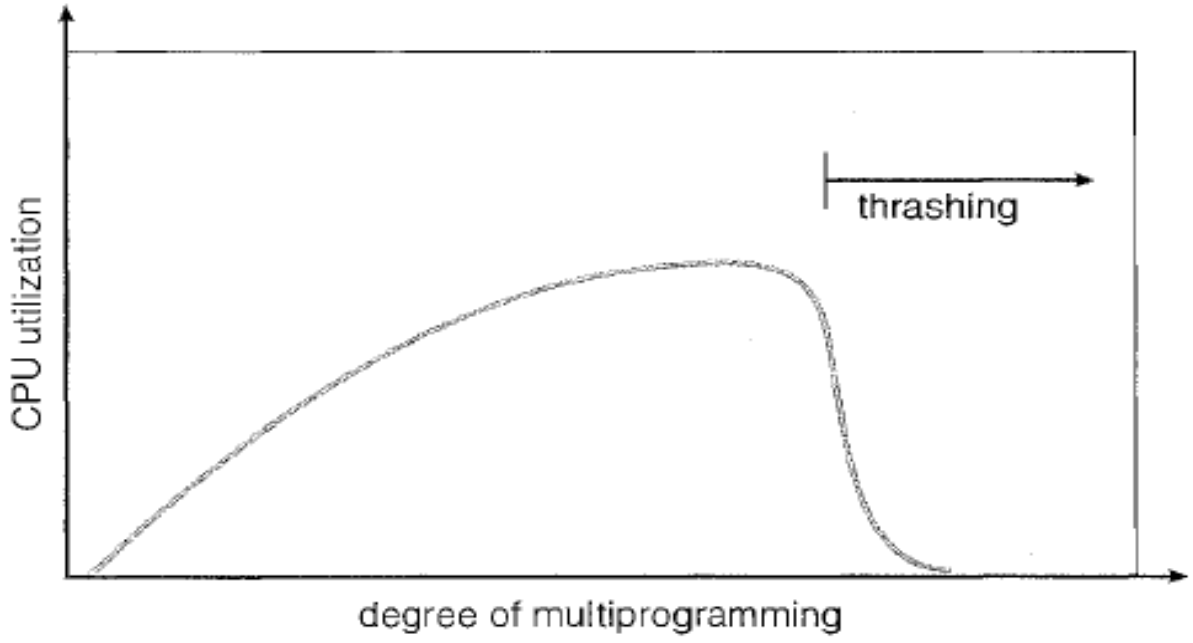


Figure: Thrashing

Causes of Thrashing

High Degree of Multiprogramming: If CPU utilization is too low, we increase the degree of multiprogramming by introducing a new process to the system.

A global page-replacement algorithm is used; it replaces pages without regard to the process to which they belong. Now if that a process enters a new phase in its execution and needs more frames. It will start faulting and taking frames away from other processes. These processes need those pages, however, and so they also fault, taking frames from other processes. These faulting processes must use the paging device to swap pages in and out. As they queue up for the paging device, the ready queue empties. As processes wait for the paging device, CPU utilization decreases.

The CPU scheduler sees the decreasing CPU utilization and increases the degree of multiprogramming as a result. The new process tries to get started by taking frames from running processes, causing more page faults and a longer queue for the paging device. As a result, CPU utilization drops even further, and the CPU scheduler tries to increase the degree of multiprogramming even more. Thrashing has occurred, and system throughput plunges. The page fault rate increases tremendously. As a result, the effective memory-access time increases. No work is getting done, because the processes are spending all their time paging.

Solution

- We can limit the effects of thrashing by using a **Local Replacement Algorithm.**
- With local replacement, if one process starts thrashing, it cannot steal frames from another process and cause the latter to thrash as well.
- To prevent thrashing, we must provide a process with as many frames as it needs.
- We decide how to allocate the number of frames to a process by **the working-set strategy.**

The Working Set Strategy

- This approach defines the **locality model** of process execution.
- The locality model states that, as a process executes, it moves from locality to locality. A locality is a set of pages that are actively used together
- A program is generally composed of several different localities, which may overlap.

Working Set Model

- This model uses a parameter Δ, to define the **working set window.**
- The set of pages in the most recent Δ page references is the working set.
- If a page is in active use, it will be in the working set. If it is no longer being used, it will drop from the working set Δ time units after its last reference.
- The working set is an approximation of the program's locality.
- The accuracy of the working set depends on the selection of Δ . If Δ is too small, it will not encompass the entire locality; if Δ is too large, it may overlap several localities.
- If we can calculate the Working-Set Size (WSS) for each process then: D = Σ WSSi, where D is D is the total demand for frames.
- If the total demand is greater than the total number of available frames (D > m), thrashing will occur, because some processes will not have enough frames.

Page-Fault Frequency

- It is a much more direct strategy to control **Page-Fault Frequency.**
- Thrashing has a high page-fault rate and thus we want to control the page-fault rate.
- When it is too high, we know that the process needs more frames. Conversely, if the page-fault rate is too low, then the process may have too many frames.
- **We establish upper and lower bounds on the desired page-fault rate.**
- If the actual page-fault rate exceeds the upper limit, we allocate the process another frame; if the page-fault rate falls below the lower limit, we remove a frame from the process. **With the working-set strategy, we may have to suspend a process. If the page-fault rate increases and no free frames are available, we must select some process and suspend it.**

Q Working set model is used in memory management to implement the concept of **(NET-JUNE-2013)**

(A) Swapping **(B)** Principal of Locality **(C)** Segmentation **(D)** Thrashing

Answer: (B)