Operating System
203105203

Prof. Riddhi Mehta & Prof. Rumana Shaikh,
Assistant Professor
Computer Science & Engineering Department

Parul® University

DIGITAL LEARNING CONTENT

CHAPTER-6

Virtual Memory

Topics to be Covered

- Basics of Virtual Memory
- Hardware and control structures
- Locality of reference
- Page fault
- Working Set
- Dirty page/Dirty bit
- Demand paging
- Page Replacement algorithms: Optimal
- First in First Out (FIFO), Second Chance (SC),
- Not recently used (NRU) and Least Recently used (LRU).

Parul® University

Topicsto be Covered

1. Concept of Paging
2. Concept of Demand Paging
3. Reviewing Paging and Page Replacement
4. Page Replacement Algorithams
5. Conclusion
6. References

Virtual Memory

- The goal of the virtual memory is to keep many processes in execution.
- It is a technique that allows the execution of processes those are not completely in main memory.
- In real time the size of programs are very large. So it is difficult to store whole program in main memory.
- To solve this problem Virtual Memory is used.

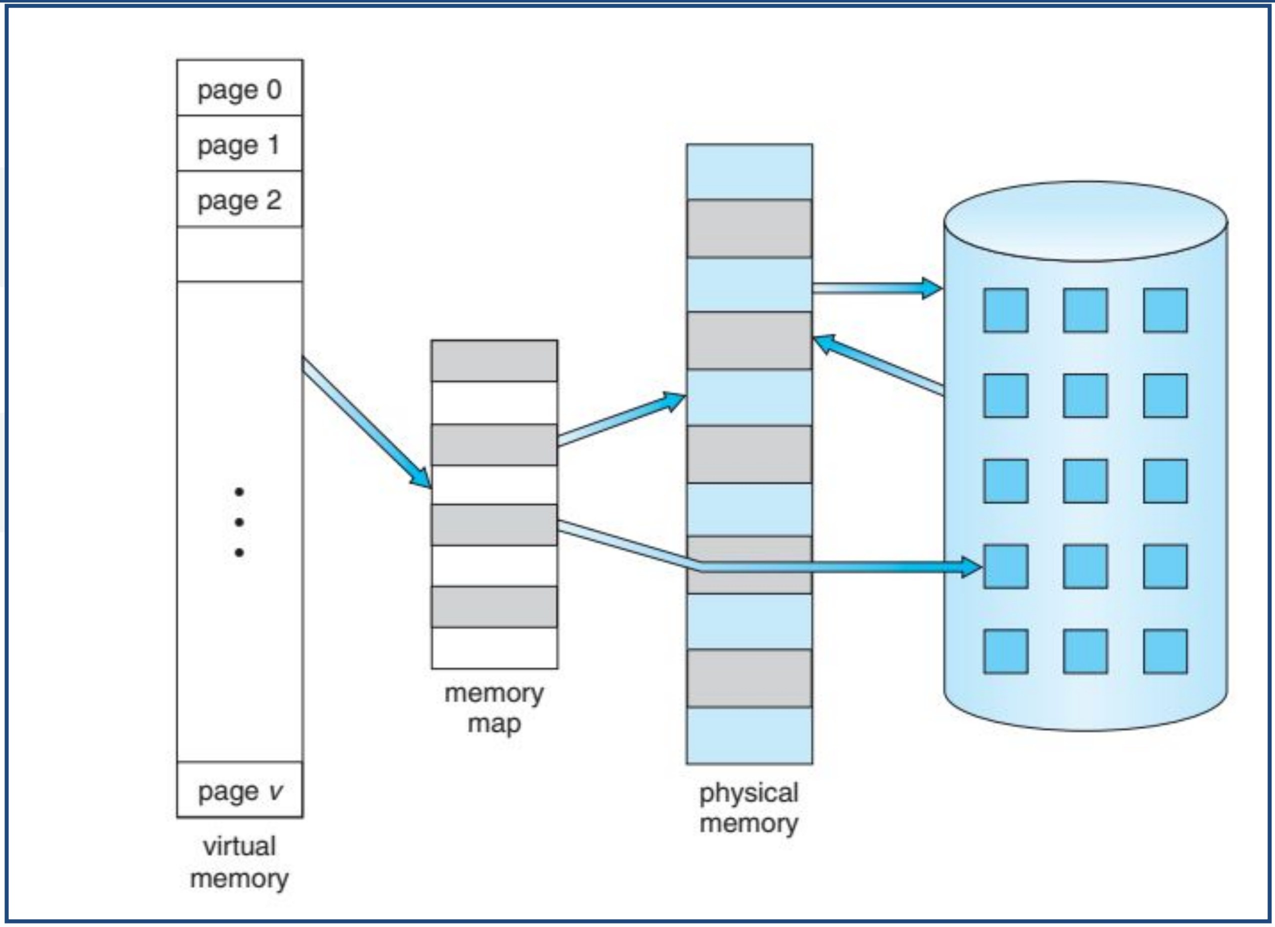
Virtual Memory

- Virtual Memory is provides space to store large programs in form of pages.
- When programs are in execution then only the required pages or part of processes are loaded into the main memory.
- This technique is useful when we have small physical memory and program size is very large.

Virtual Memory

Diagram showing virtual memory that is larger than physical memory

Animation Example



Parul® University

DIGITAL LEARNING CONTENT

Advantages of Virtual Memory

- Virtual memory will help to increase the speed when only a particular segment of the program is required for the execution of the program.
- It is very helpful in implementing a multiprogramming environment.
- It allows you to run more applications at once.
- It helps to fit many large programs into smaller programs.

Advantages of Virtual Memory

- Similar type of data or code may be shared between memory.
- The code can be stored anywhere in physical memory without requiring relocation.
- More processes should be maintained in the M.M., which increases the effective use of CPU.
- Each page is placed on a disk until it is required after that, it will be erased.

Disadvantages of Virtual Memory

- If the system is using virtual memory then applications running speed is slower down.
- It will takes more time to perform switching between applications.
- Hard drive space is reduced.
- It reduces system stability.

Disadvantages of Virtual Memory

- It allows larger applications to run in systems that don't offer enough physical RAM alone to run them.
- It doesn’t perform same as RAM.
- It negatively affects the overall performance of a system.
- Occupy the storage space, which may be used or else for long term data storage.

Locality of Reference

- Locality of reference refers to a computer program tends to access same set of memory locations for a specific time period.

- In other words, Locality of Reference refers to the tendency of the computer program to access instructions whose addresses are near one another.

- The property of LOR is mainly shown by subroutine calls and loops in a program.

Locality of Reference

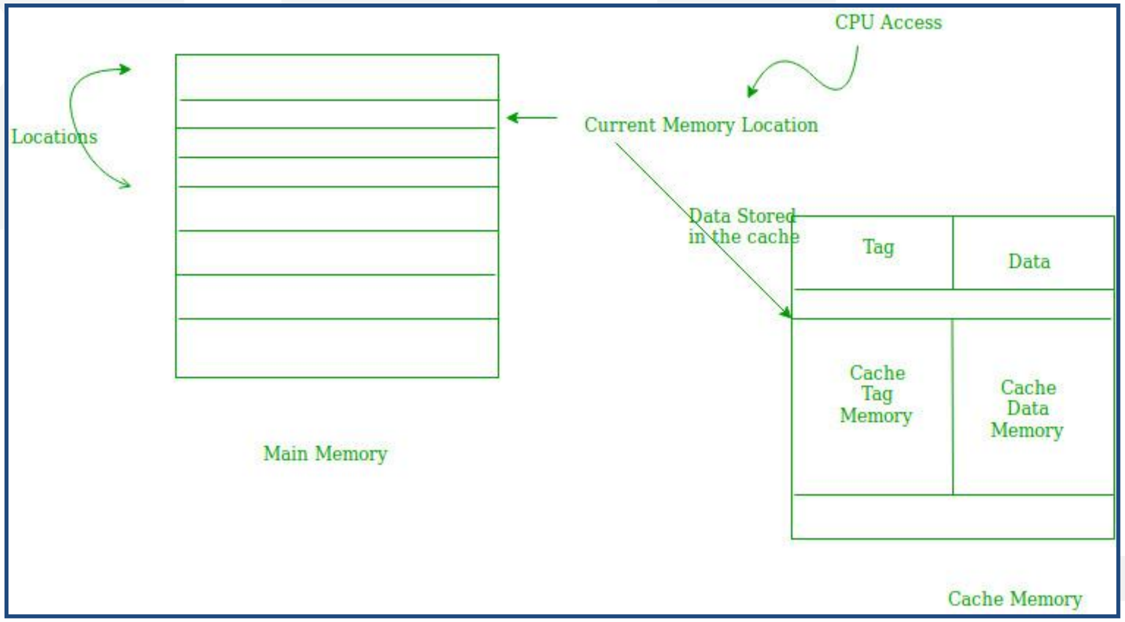
- It is based on the principle of locality of reference.
- There are two different ways using which data or instruction is fetched from M.M. and get stored in cache memory.
- These two ways are the following:
  1) Temporal Locality[6]
  2) Spatial Locality[6]

Parul® University

DIGITAL LEARNING CONTENT

1) Temporal Locality

- Temporal locality means current data or instruction that is being fetched may be required soon. So we should store that data or instruction in the cache memory so that we can avoid again searching in main memory for the same data.



Working of Temporal

Parul® University

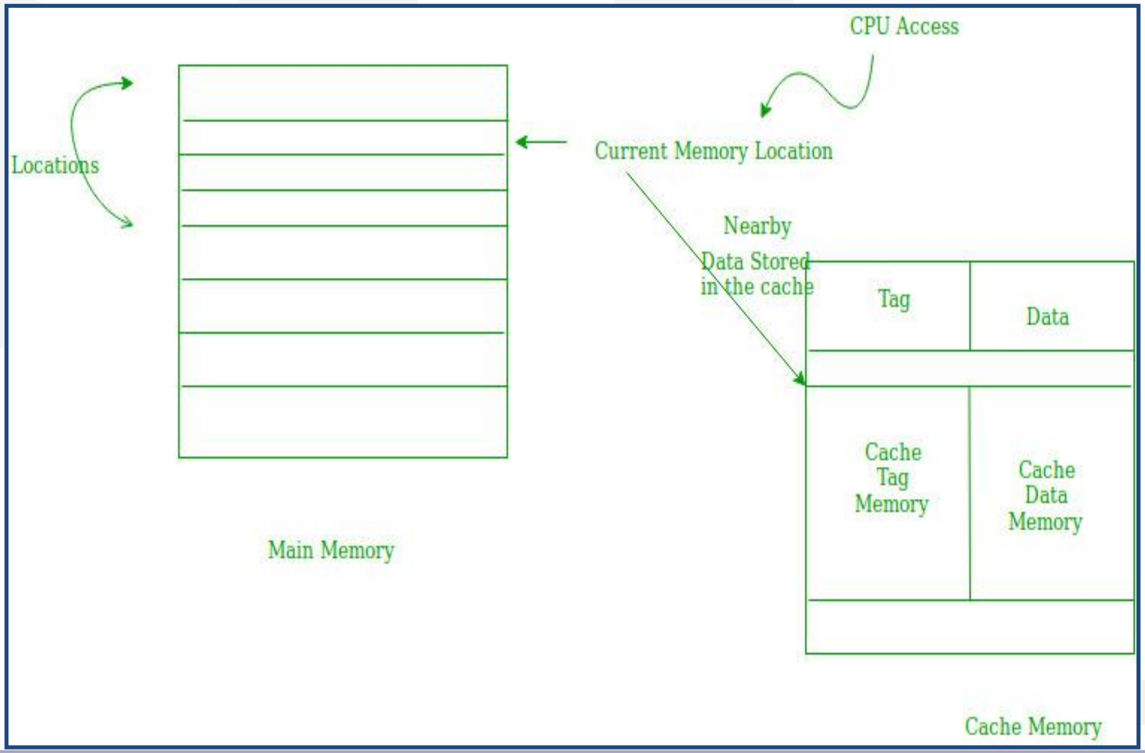
DIGITAL LEARNING CONTENT

2) Spatial Locality

- Spatial locality means instruction or data near to the current memory location that is being fetched, may be required soon in the near future.

- This is slightly different from the temporal locality. Here we are talking about nearly located memory locations while in temporal locality we were talking about the actual memory location that was being fetched.

2) Spatial Locality



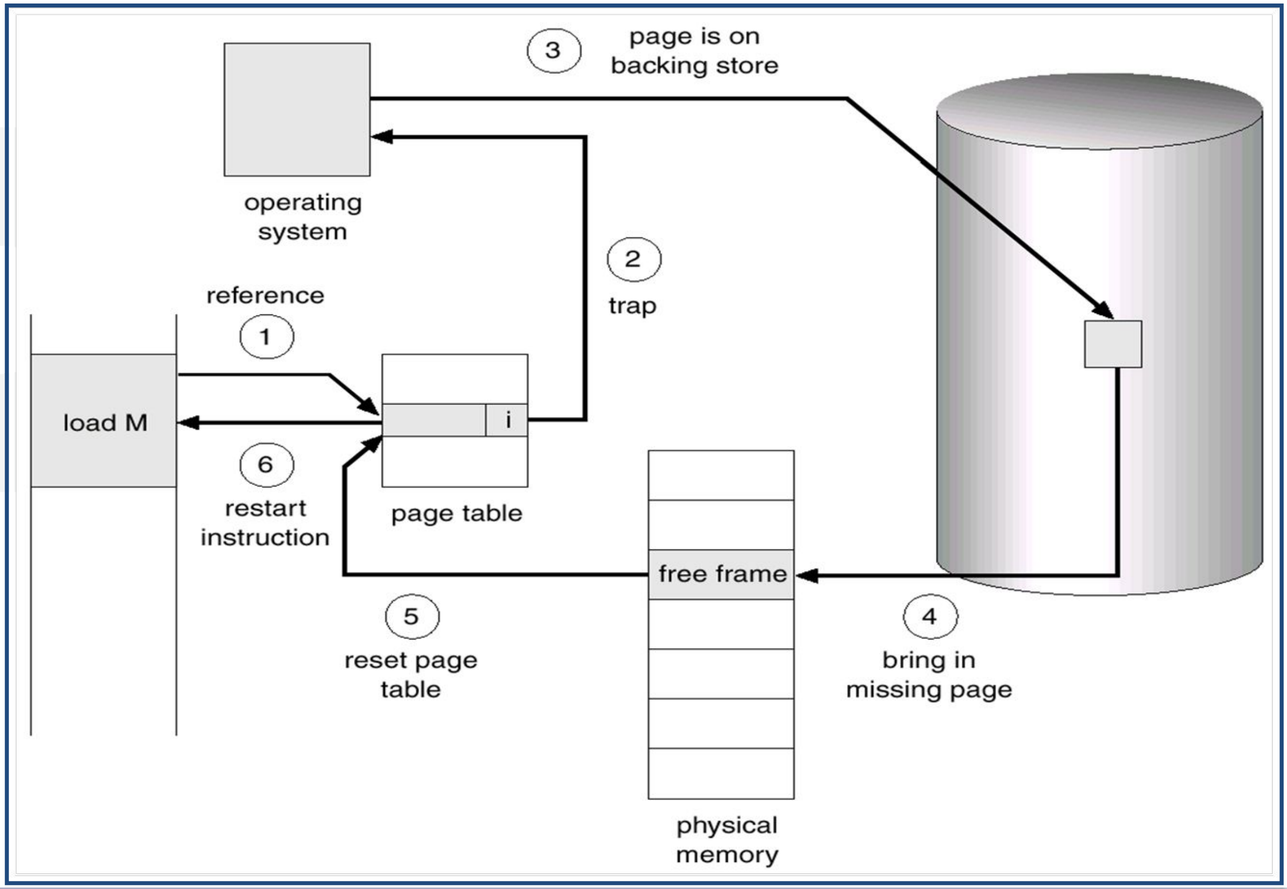
Working of Spatial

Page Fault

- When the process tries to access a page that was not available into memory, it causes a **Page Fault**.
- When process tries to get the page which valid/invalid bit is set to invalid, indicates the page is not loaded to memory, lazy pager generates the page fault to the operating system.
- So later on that page should be brought to the main memory.

PU

Steps in Handling a Page Fault



Animation Example

Steps to handling a

Steps in Handling a Page Fault

1) First we need to check an internal table for the process, to check whether the reference was a valid or invalid memory access.

2) If the reference was invalid, we need to terminate the process. But if in case it was valid, but we have not yet brought in that page, we now page it in.

3) We find a free frame.

Steps in Handling a Page Fault

4) We schedule a disk operation to read the desired page into the newly allocated frame.

5) When the disk reading task is complete, we modify the internal table kept with the process and the page table to indicate that page is now in memory.

6) We restart the instruction that was interrupted by the illegal address trap.

Dirty Pages

- Dirty pages are the pages in memory (page cache) that have been rationalized and consequently have changed from what is currently stored on disk.

- This usually happens when an existing file on the disk is altered or appended.

Dirty Bits / Modified Bits

- To reduce the page fault service time, a special bit called the dirty bit can be associated with each page.
- A dirty bit or modified bit is a bit that is related with a block of computer memory and indicates whether or not the corresponding block of memory has been modified.
- Dirty bits are used by the CPU cache.
- It can be also used in the page replacement algorithms of an operating system.

Dirty Bits / Modified Bits

- Whenever the page is modified (written into) the value of dirty bit is set to ‘1’ by the hardware.
- When we select a victim by using a page replacement algorithm, we needs to examine its dirty bit.
- If the dirty bit is set, that means the page has been modified since it was swapped in. In this scenario we have to write that page into the backing store.
- If the dirty bit is reset, that means the page has not been modified since it was swapped in, so we don’t have to write it into the backing store. The copy in the backing store is valid.

What is Paging?

- MemThe OS divides virtual memory and the main memory into

units, called “pages”.

- Each used page can be either in secondary memory or in a page frame in main memory.

- A frame does not have to comprise a single physically contiguous region in secondary storage.

Concept of Demand Paging

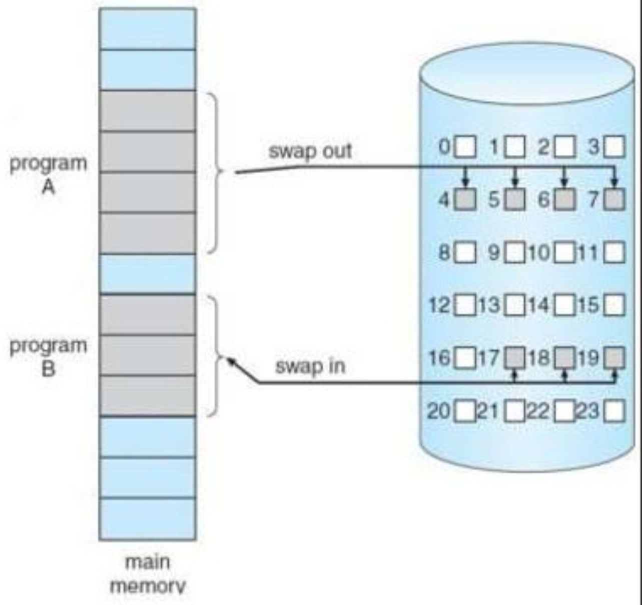
- According to the concept of Virtual Memory, in order to execute some process, only a part of the process needs to be present in the main memory which means that only a few pages will only be present in the main memory at any time.
- However, deciding, which pages need to be kept in the main memory and which need to be kept in the secondary memory, is going to be difficult because we cannot say in advance that a process will require a particular page at particular time.

Concept of Demand Paging Continue…

- Therefore, to overcome this problem, there is a concept called Demand Paging is introduced. It suggests keeping all pages of the frames in the secondary memory until they are required. In other words, it says that do not load any page in the main memory until it is required.
- Whenever any page is referred for the first time in the main memory, then that page will be found in the secondary memory.

Demand Page

- Bring a page into memory only when it is needed
  - Less I/O needed
  - Less memory needed
  - Faster response
  - More users
- Page is needed reference to it
  - Invalid reference abort
  - Not-in-memory bring to memory
- **Lazy swapper** -never swaps a page into memory unless page will be needed
  - Swapper that deals with pages is a **pager**

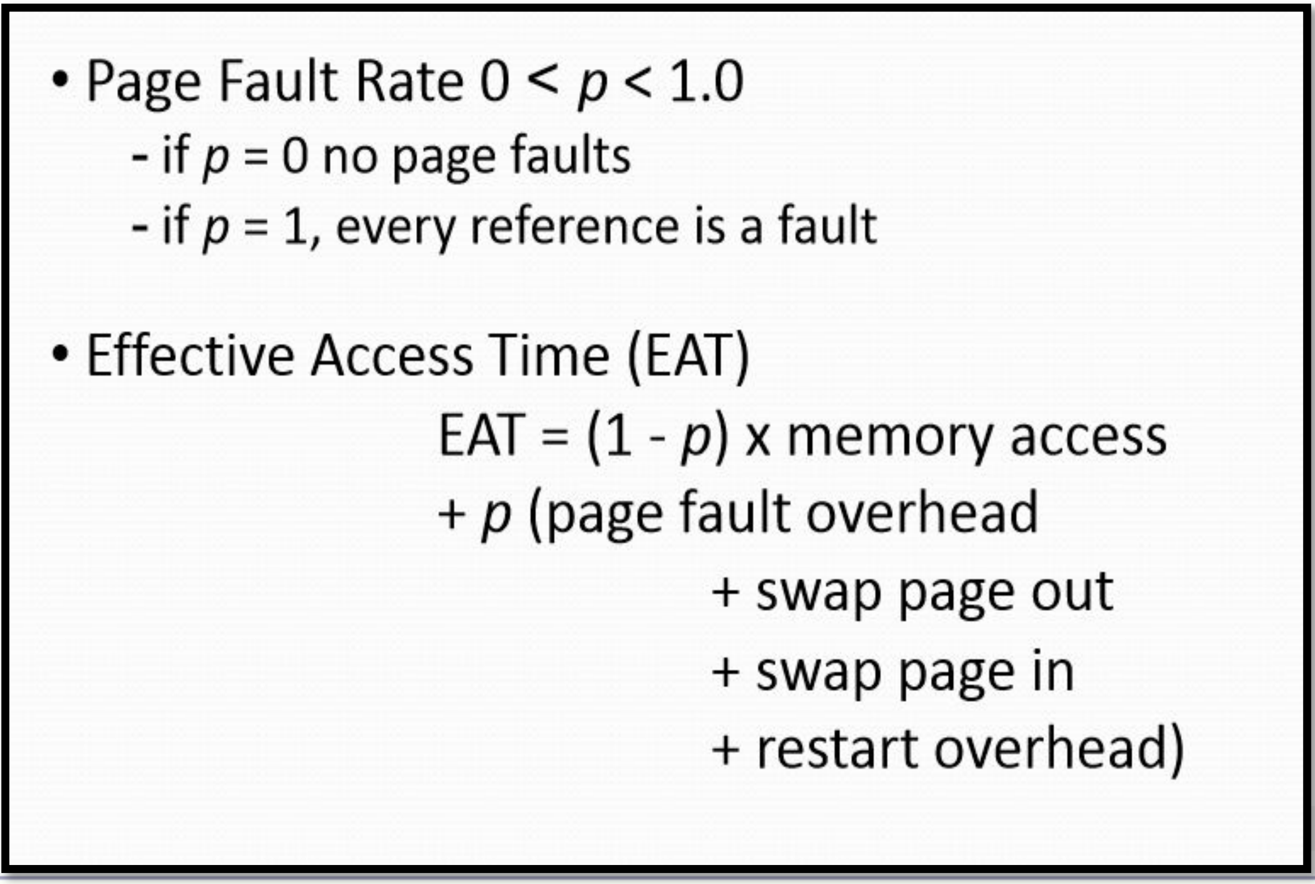


Activate W
Go to Settin

Page Fault

- If there is a reference to a page, first reference to that page will trap to operating system: page fault.
- If there is a reference to a page, first reference to that page will trap to operating system: page fault
- Get empty frame
- Swap page into frame
- Reset tables
- Set validation bit = v
- Restart the instruction that caused the page fault

Performance of Demand Paging



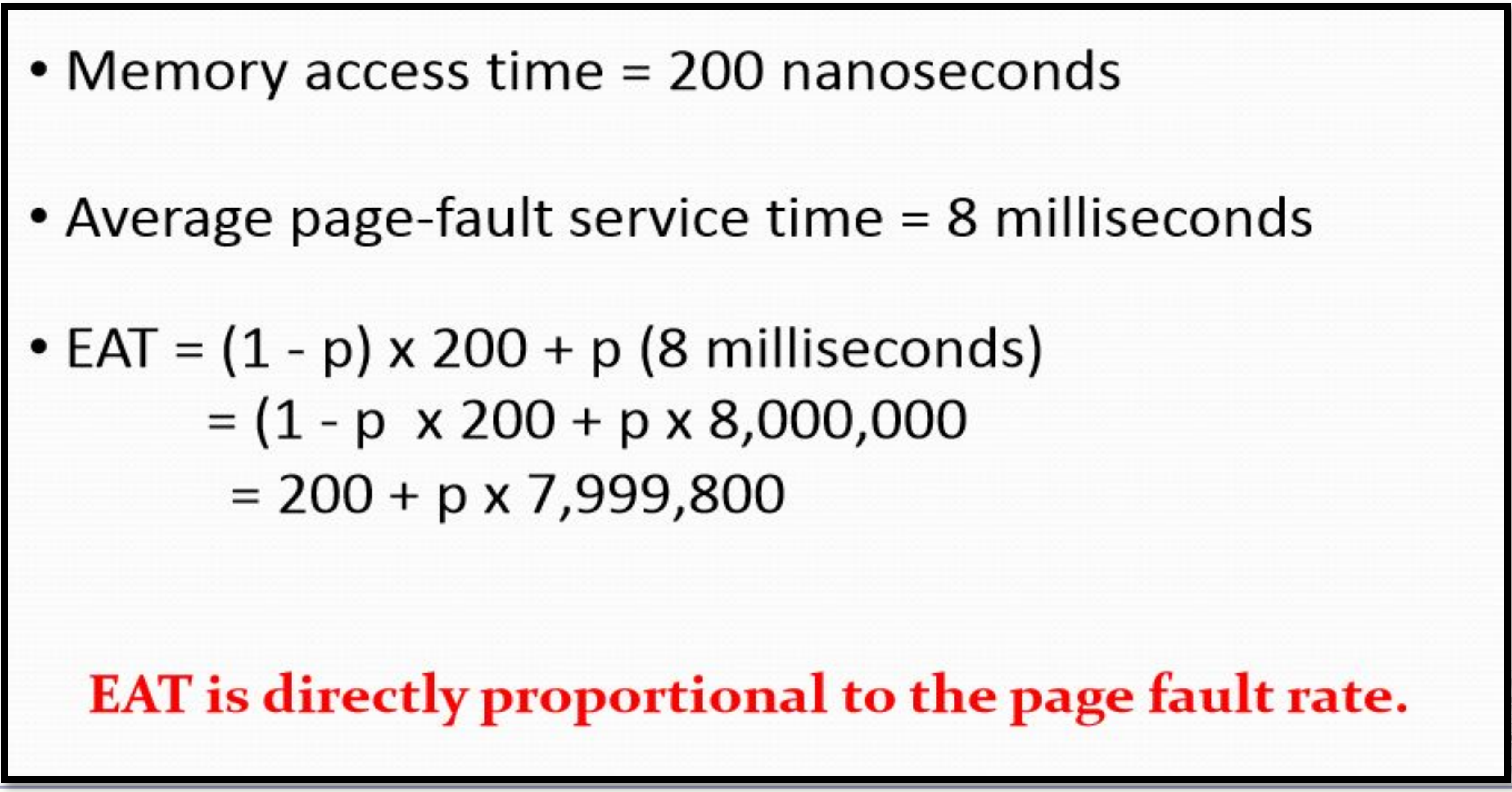
- Page Fault Rate 0 < *p* < 1.0
  - if *p* = 0 no page faults
  - if *p* = 1, every reference is a fault
- Effective Access Time (EAT)

EAT = (1 - *p*) x memory access
+ *p* (page fault overhead
+ swap page out
+ swap page in
+ restart overhead)

Parul® University

DIGITAL LEARNING CONTENT

Demand Paging Example



- Memory access time = 200 nanoseconds
- Average page-fault service time = 8 milliseconds
- EAT = (1 - p) x 200 + p (8 milliseconds)

  = (1 - p x 200 + p x 8,000,000

  = 200 + p x 7,999,800

EAT is directly proportional to the page fault rate.

HARDWARE SUPPORT

- The hardware to support demand paging is the same as the hardware forpaging and swapping:
- PAGE TABLE : This table has the ability to mark an entry invalid through a valid invalid bit or a special valueof protection bits.
- SECONDARY MEMORY : The memory holds those pages that are not present in the main memory. The secondary memory is usuallya high speed disk. It is known as Swap Device and the section of the disk used forthis purpose is known as the Swap Space.

Parul® University

DIGITAL LEARNING CONTENT

What happens if there is no free frame?

- Page replacement - find some page in memory, but not really in use, swap it out - Algorithm - Performance - want an algorithm which will result in minimum number of page faults.
- Same page may be brought into memory several times.

.

What is page replacement?

- When memory located in secondary memory is needed, it can be retrieved back to main memory.
- Process of storing data from main memory to secondary memory ->swapping out.
- Retrieving data back to main memory ->swapping in

Page Replacement

- Find the location of the desired page on the disk
- Find a free frame :

  If there is a free frame, use it.

  If there is no free frame, use a page replacement algorithm to Selecta victim frame.

  Write the victim frame to the disk, change the page and frame tables accordingly.
- Read the desired page into the newly freed frame, change the page and frame tables.
- Restart the user process.

Page Replacement

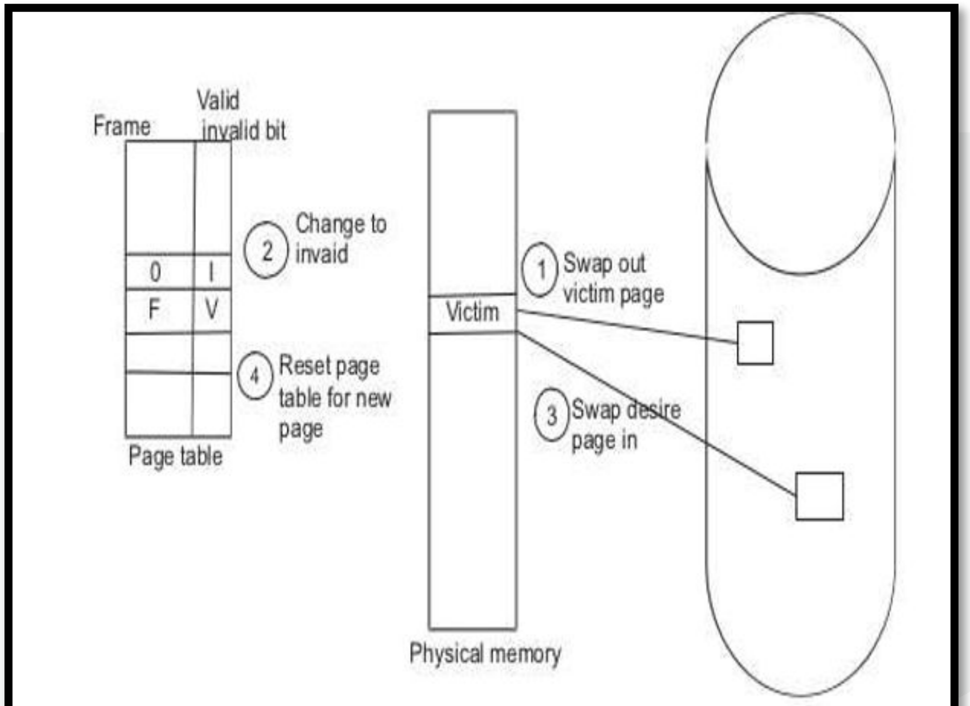


Fig: Page Replacement

STEPS IN PAGE REPLACEMENT

- Find the location of the desired page on the disk.
- Find a free frame :
  If there is a free frame, use it.
  If there is no free frame, use a page replacement algorithm to Selecta victim frame.
  Write the victim frame to the disk, change the page and frame tables accordingly.
- Read the desired page into the newly freed frame, change the page and frame tables.
- Restart theuserprocess.

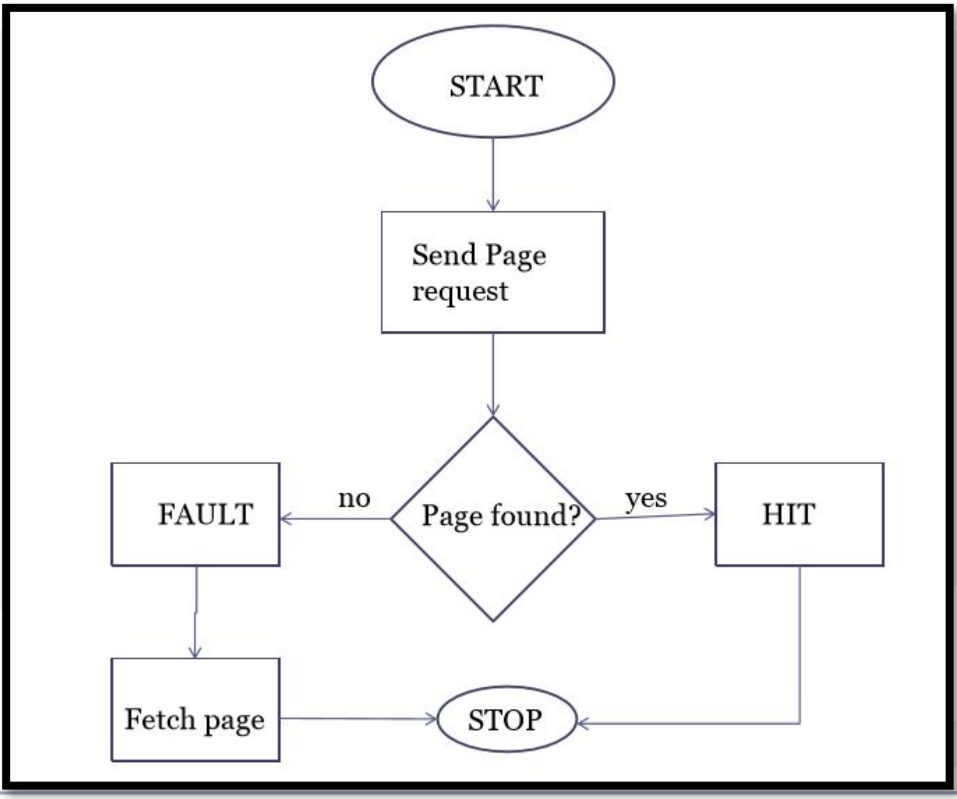
What are Page Replacement Algorithms?

- Deals with which pages need to be swapped out and which are the ones that need to be swapped in
- The efficiency lies in the least time that is wasted for a page to be paged in

Why we need a page replacement algorithm?

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

Why we need a page replacement algorithm?

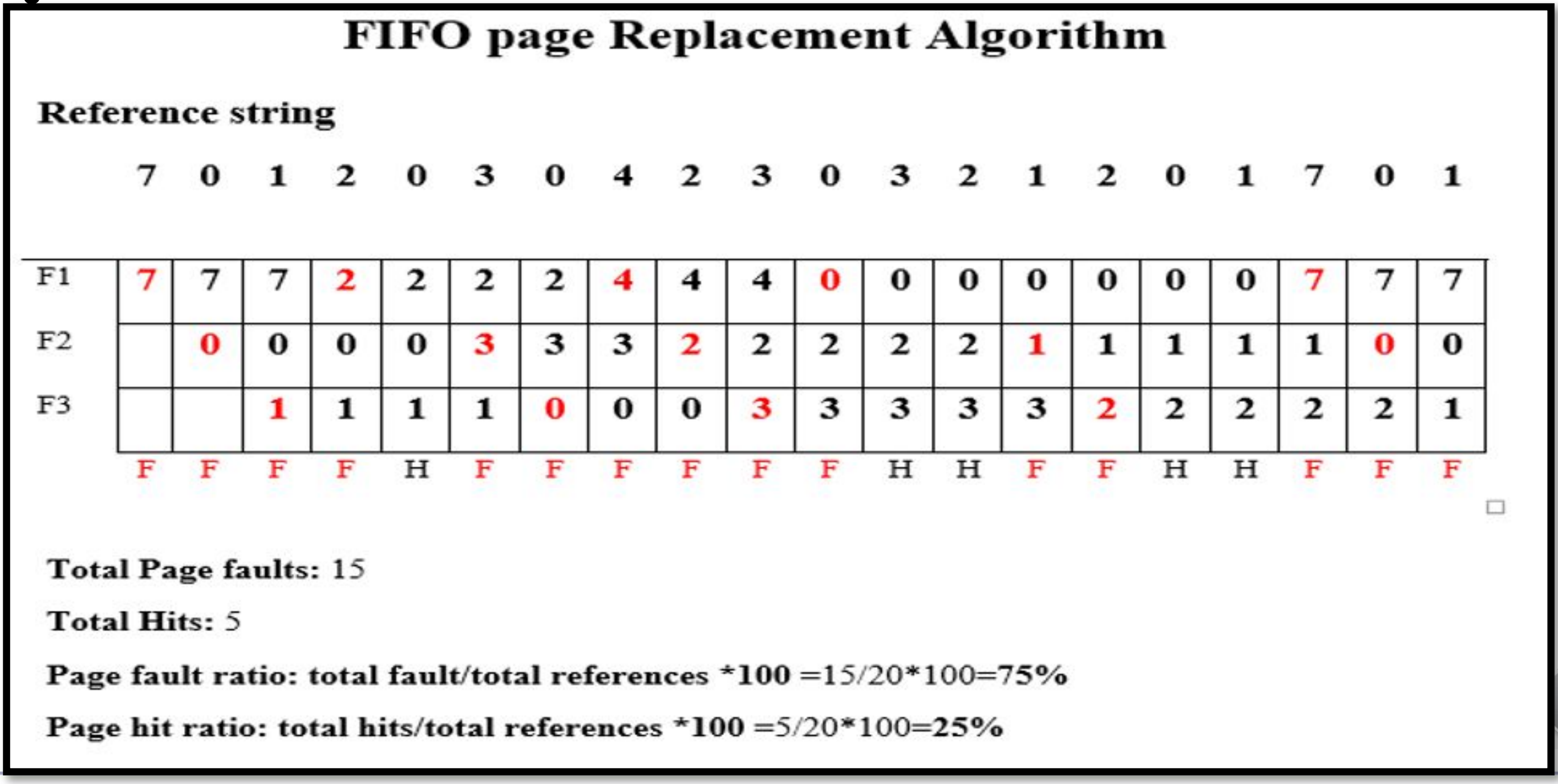


Page Replacement Algorithms

- Want lowest page-fault rate • Evaluate algorithm by running it on a particular string of memory references (reference string) and computing the number of page faults on that string
- In all our examples, the reference string is
- 7,0,1,2,0,3,0,4,2,3,0,3,2,1,2,0,1,7,0,1

FIFO page replacement

- Simplest page replacement algorithm
- FIFO: First In First Out
- Associate with each page the time when that page was brought into memory
- When page must be replaced oldest page is chosen.
- We can create a FIFO queue to hold all pages in memory.
- We replace page at the head of the queue.
- When page is brought into memory we replace insert it at the tail of the queue
- For our example of reference string we consider three frames
- These three frames are initially empty.
- FIFO page replacement works as below



FIFO page Replacement Algorithm

Reference string

	7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1	7	0	1
F1	7	7	7	2	2	2	2	4	4	4	0	0	0	0	0	0	0	7	7	7
F2		0	0	0	0	3	3	3	2	2	2	2	2	1	1	1	1	1	0	0
F3			1	1	1	1	0	0	0	3	3	3	3	3	2	2	2	2	2	1
	F	F	F	F	H	F	F	F	F	F	F	H	H	F	F	H	H	F	F	F

Total Page faults: 15

Total Hits: 5

Page fault ratio: total fault/total references *100 =15/20*100=**75%**

Page hit ratio: total hits/total references *100 =5/20*100=**25%**

Drawbacks - FIFO

- A page which is being accessed quite often may also get replaced because it arrived earlier than those present
- Ignores locality of reference. A page which was referenced last may also get replaced, although there is high probability that the same page may be needed again.

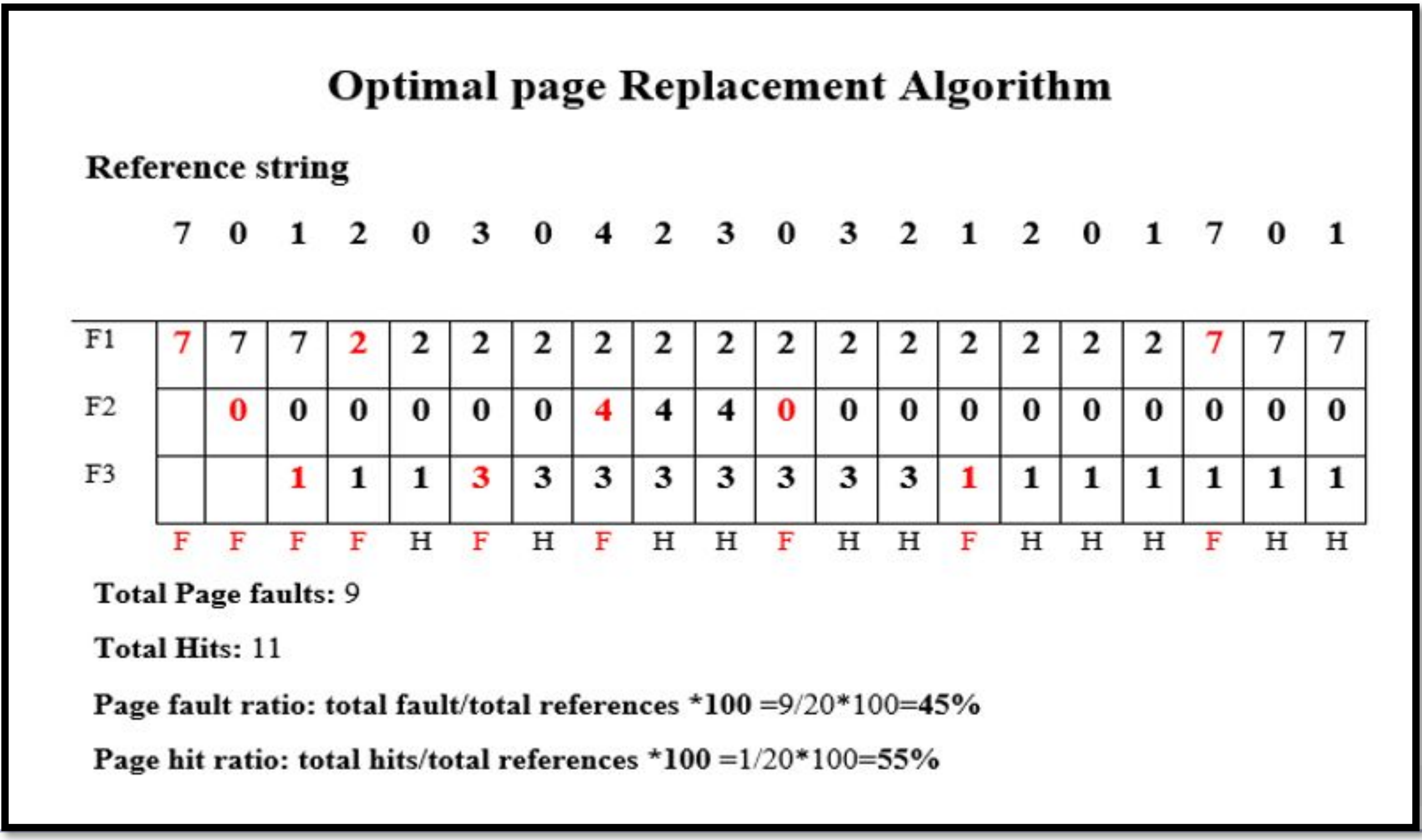
Optimal Page replacement

- Optimal page replacement has lowest page fault rate of all algorithms will never suffer from Baladys Anomaly.
- “replace the page that will not be used for longest period of time”
- Difficult to implement because it requires future knowledge of reference string.

Parul® University

DIGITAL LEARNING CONTENT

Optimal Page replacement



Optimal page Replacement Algorithm

Reference string

	7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1	7	0	1
F1	7	7	7	2	2	2	2	2	2	2	2	2	2	2	2	2	2	7	7	7
F2		0	0	0	0	0	0	4	4	4	0	0	0	0	0	0	0	0	0	0
F3			1	1	1	3	3	3	3	3	3	3	3	1	1	1	1	1	1	1
	F	F	F	F	H	F	H	F	H	H	F	H	H	F	H	H	H	F	H	H

Total Page faults: 9

Total Hits: 11

Page fault ratio: total fault/total references *100 =9/20*100=**45%**

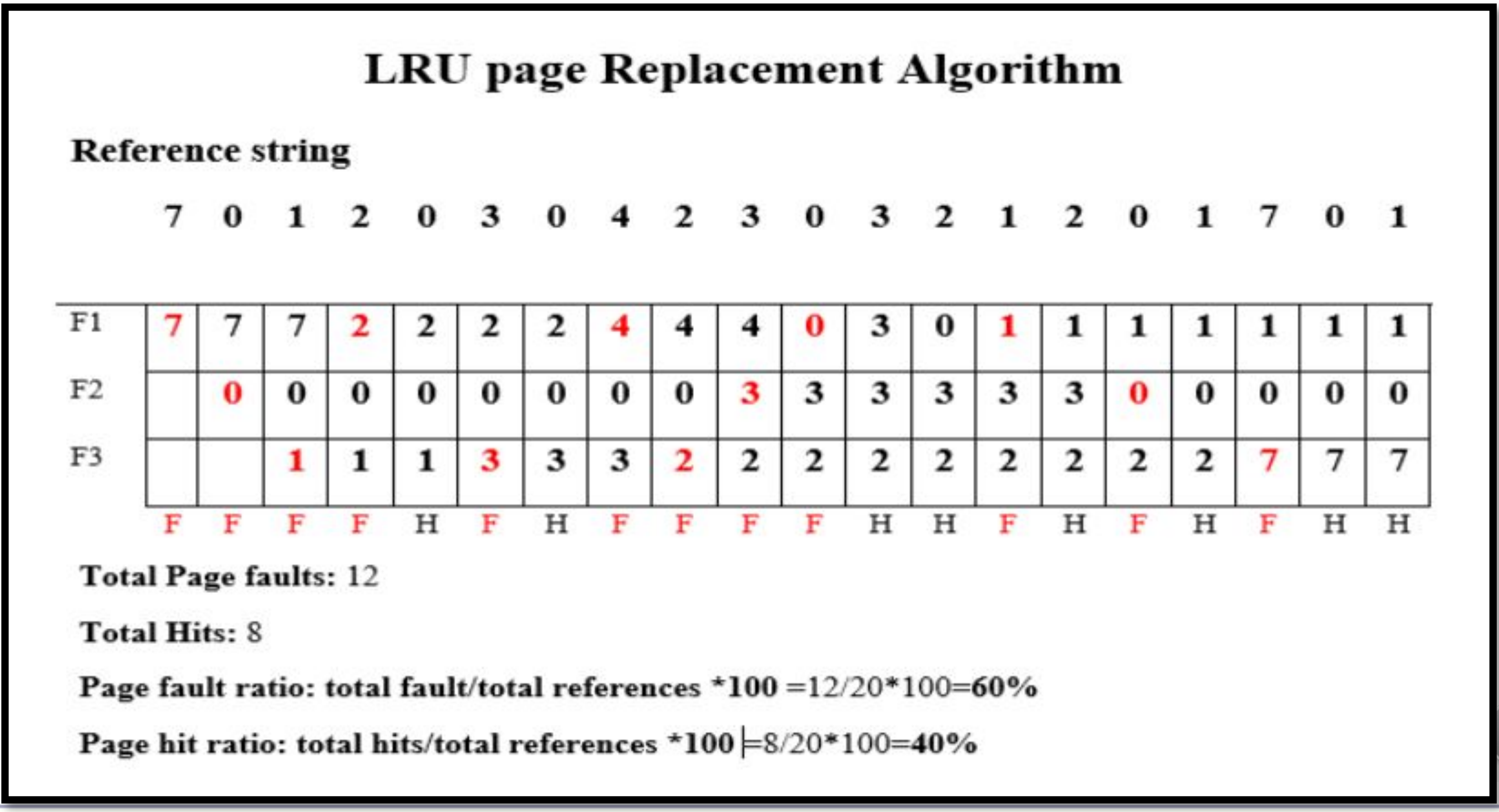
Page hit ratio: total hits/total references *100 =1/20*100=**55%**

LRU Page Replacement

- LRU: Least Recently Used
- FIFO: uses when page is brought into memory.
- Optimal: Uses time when a page to be used.
- Replace a page that has not been usedfrom longest period of time.
- When a page must be replaced, LRU choses a page that has not been used from longest period of time.
- Looking backward in time.

Parul® University

DIGITAL LEARNING CONTENT



LRU page Replacement Algorithm

Reference string

	7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1	7	0	1
F1	7	7	7	2	2	2	2	4	4	4	0	3	0	1	1	1	1	1	1	1
F2		0	0	0	0	0	0	0	0	3	3	3	3	3	3	0	0	0	0	0
F3			1	1	1	3	3	3	2	2	2	2	2	2	2	2	2	7	7	7
	F	F	F	F	H	F	H	F	F	F	F	H	H	F	H	F	H	F	H	H

Total Page faults: 12

Total Hits: 8

Page fault ratio: total fault/total references *100 =12/20*100=**60%**

Page hit ratio: total hits/total references *100 =8/20*100=**40%**

LRU Approximation

- Few computer system provides sfficient hardware support for LRU page replacement.
- Some system provide no hardware support and other page replacement algorithm must be used.
- Many system provide reference bits.
- Reference bit is set by hardware whenever page is referenced.
- reference bit is associated with each entry in page table.
- Initially bits are cleared (to0)by OS.
- A user process executes the bit associated with each page referenced is set (to 1) by hardware.
- After some time we determine which page have been used and have not been used by examining the reference bits.

Not Recently Used (NRU)

- It favours keeping pages in memory that have been recently used.
- The OS divides the pages into four classes based on usage during the last clock tick:
- 3. Referenced, modified
- 2. Referenced, not modified
- 1. Not referenced, modified
- 0. Not referenced, not modified
- Pick a random page from the lowest category for removal
- i.e. the not referenced, not modified pag

NRU EXAMPLE

NRU Example

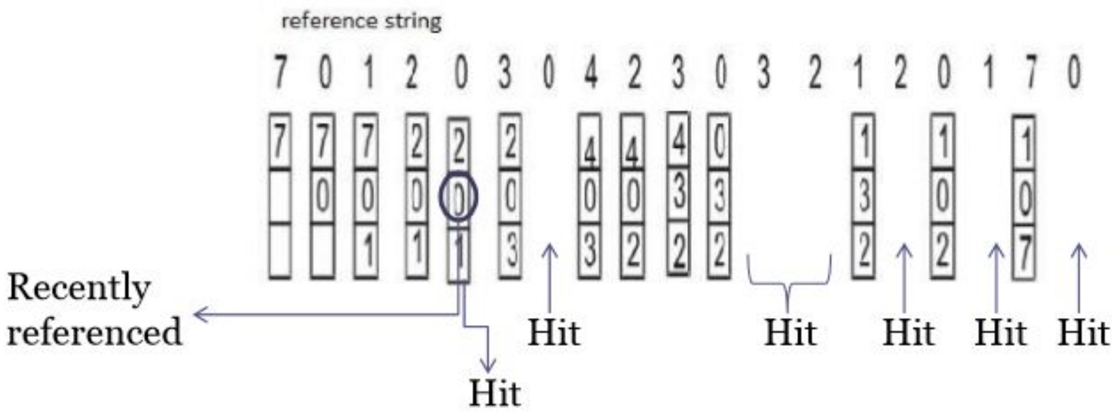
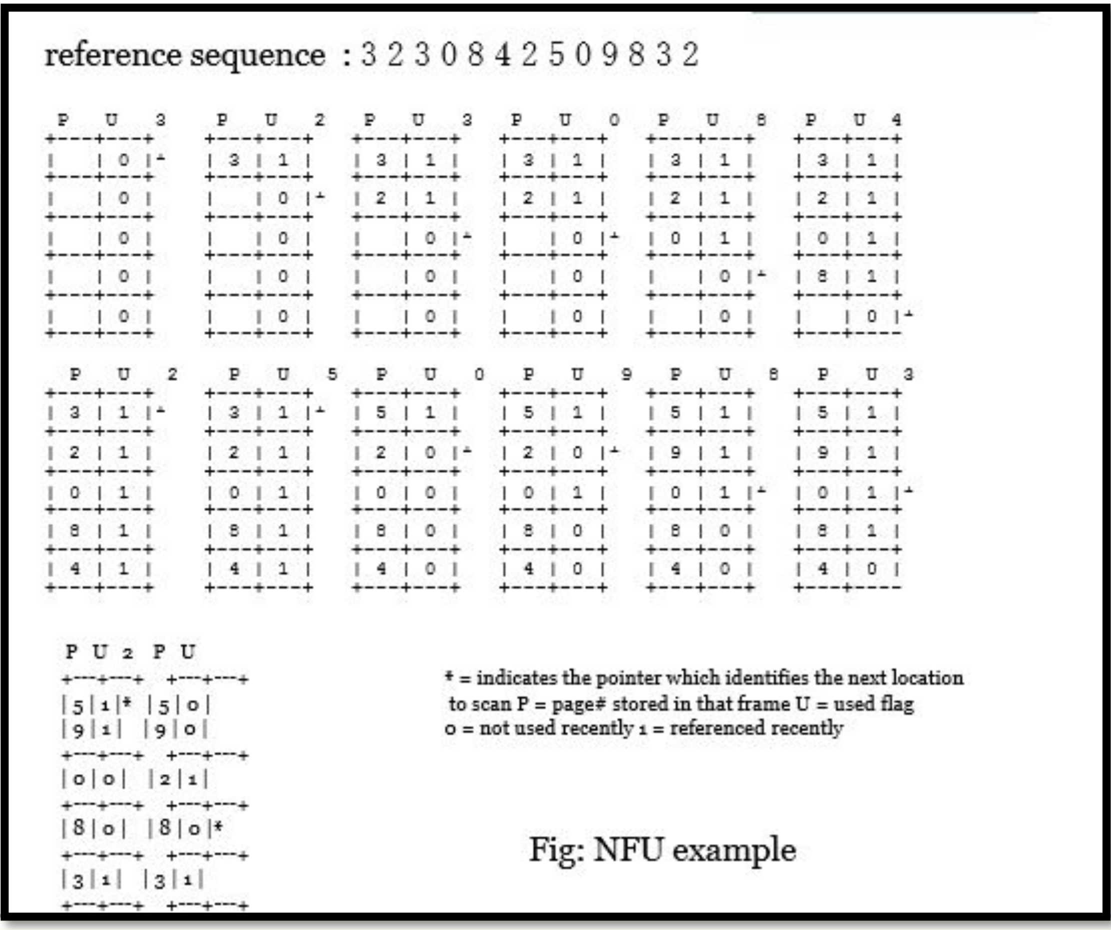


Fig: NRU example

Not frequently used (NFU)

- This page replacement algorithm requires a counter
- The counters keep track of how frequently a page has been used
- The page with the lowest counter can be swapped out

NFU EXAMPLE



reference sequence : 3 2 3 0 8 4 2 5 0 9 8 3 2

```
 P   U   3     P   U   2     P   U   3     P   U   0     P   U   8     P   U   4
+---+---+     +---+---+     +---+---+     +---+---+     +---+---+     +---+---+
|   | 0 |*    | 3 | 1 |     | 3 | 1 |     | 3 | 1 |     | 3 | 1 |     | 3 | 1 |
+---+---+     +---+---+     +---+---+     +---+---+     +---+---+     +---+---+
|   | 0 |     |   | 0 |*    | 2 | 1 |     | 2 | 1 |     | 2 | 1 |     | 2 | 1 |
+---+---+     +---+---+     +---+---+     +---+---+     +---+---+     +---+---+
|   | 0 |     |   | 0 |     |   | 0 |*    |   | 0 |*    | 0 | 1 |     | 0 | 1 |
+---+---+     +---+---+     +---+---+     +---+---+     +---+---+     +---+---+
|   | 0 |     |   | 0 |     |   | 0 |     |   | 0 |     |   | 0 |*    | 8 | 1 |
+---+---+     +---+---+     +---+---+     +---+---+     +---+---+     +---+---+
|   | 0 |     |   | 0 |     |   | 0 |     |   | 0 |     |   | 0 |     |   | 0 |*
+---+---+     +---+---+     +---+---+     +---+---+     +---+---+     +---+----

 P   U   2     P   U   5     P   U   0     P   U   9     P   U   8     P   U   3
+---+---+     +---+---+     +---+---+     +---+---+     +---+---+     +---+---+
| 3 | 1 |*    | 3 | 1 |*    | 5 | 1 |     | 5 | 1 |     | 5 | 1 |     | 5 | 1 |
+---+---+     +---+---+     +---+---+     +---+---+     +---+---+     +---+---+
| 2 | 1 |     | 2 | 1 |     | 2 | 0 |*    | 2 | 0 |*    | 9 | 1 |     | 9 | 1 |
+---+---+     +---+---+     +---+---+     +---+---+     +---+---+     +---+---+
| 0 | 1 |     | 0 | 1 |     | 0 | 0 |     | 0 | 1 |     | 0 | 1 |*    | 0 | 1 |*
+---+---+     +---+---+     +---+---+     +---+---+     +---+---+     +---+---+
| 8 | 1 |     | 8 | 1 |     | 8 | 0 |     | 8 | 0 |     | 8 | 0 |     | 8 | 1 |
+---+---+     +---+---+     +---+---+     +---+---+     +---+---+     +---+---+
| 4 | 1 |     | 4 | 1 |     | 4 | 0 |     | 4 | 0 |     | 4 | 0 |     | 4 | 0 |
+---+---+     +---+---+     +---+---+     +---+---+     +---+---+     +---+----

 P U 2 P U
+---+---+  +---+---+
|5|1|*  |5|0|
|9|1|  |9|0|
+---+---+  +---+---+
|0|0|  |2|1|
+---+---+  +---+---+
|8|0|  |8|0|*
+---+---+  +---+---+
|3|1|  |3|1|
+---+---+  +---+---+
```

* = indicates the pointer which identifies the next location to scan P = page# stored in that frame U = used flag
0 = not used recently 1 = referenced recently

Fig: NFU example

Conclusion

Algorithm	*Comment*
FIFO	Might throw out important pages
OPTIMAL	Not implementable
LRU	Excellent but difficult to implement
NRU	Crude approximation of LRU
NFU	Crude approximation of LRU

References

Web Links

- www.wikipedia.com
- www.youtube.com
- www.vbForum.com

Papers

- Operating System Page Replacement Algorithms by A. Frank C. Wersberg

Parul® University

DIGITAL LEARNING CONTENT

References

[1] Silberschatz, A., Galvin, P. B., & Gagne, G. (2005). Operating system concepts. Hoboken, NJ: J. Wiley & Sons.

[2] Stallings, W. (2018). Operating systems: Internals and design principles. Prentice-Hall

[3] Tanenbaum, A. (2014). Modern operating systems. Harlow: Pearson.

[4] Operating System Generations. Tutorialspoint. https://www.tutorialspoint.com/operating-system-generations

[5] Image source www.google.com

[6] Locality of Reference. Geeksforgeeks https://www.geeksforgeeks.org/locality-of-reference-and-cache-operation-in-cache-memory/

DIGITAL LEARNING CONTENT

Parul® University

www.paruluniversity.ac.in