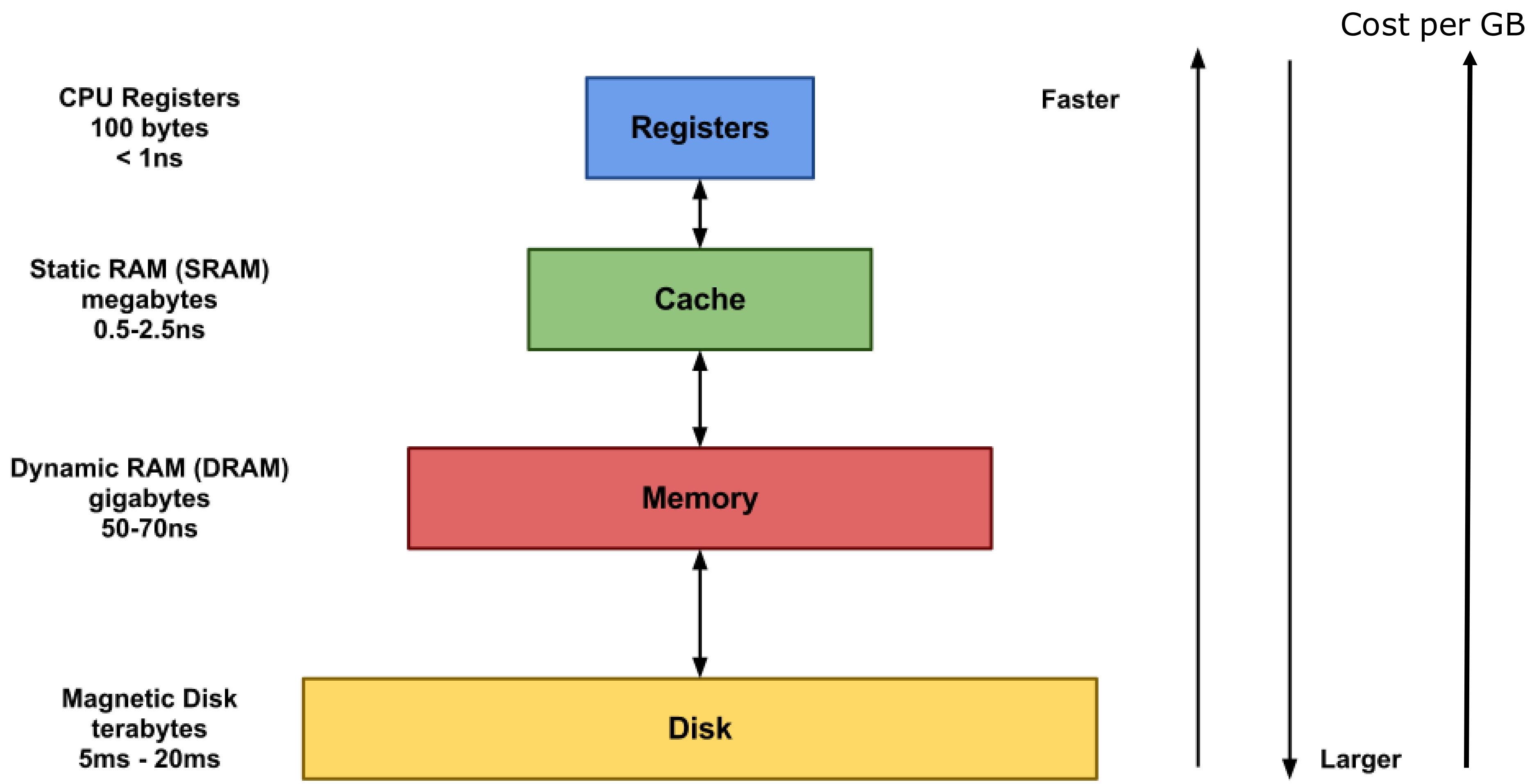
Memory Hierarchy



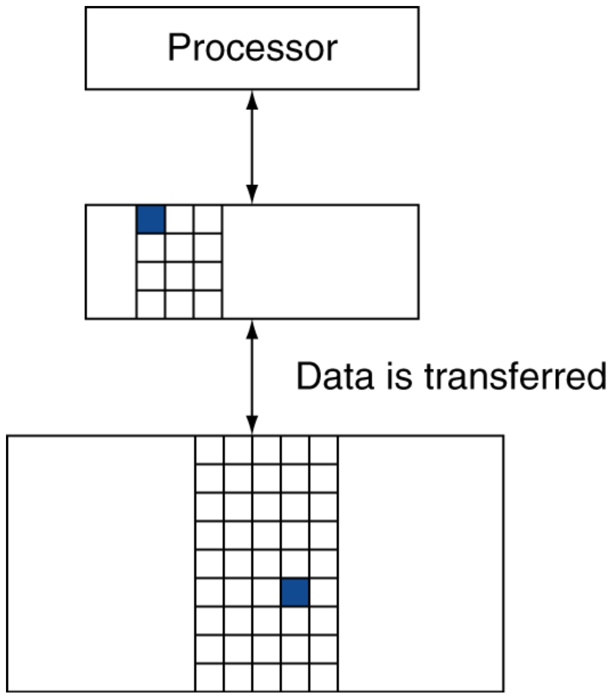
Memory Hierarchy Levels

- Idea:
  - Store everything on disk
  - Copy recently accessed (and nearby) items from disk to smaller DRAM memory
    - Main memory
  - Copy more recently accessed (and nearby) items from DRAM to smaller SRAM memory
    - Cache memory attached to CPU

Memory Hierarchy Levels

- A memory hierarchy can consist of multiple levels, but data is copied between only two adjacent levels at a time.
  - The upper level—the one closer to the processor—is smaller and faster than the lower level, since the upper level uses technology that is more expensive.

Memory Hierarchy Levels



- Block (aka line)
  - Unit of copying,
  - The minimum unit of information that can be either present or not present in a cache.
  - Usually one word, but can also be multiple words
- Hit - accessed data is present in upper level
- Miss - accessed data is absent

Memory Hierarchy Levels

- In the case of hit
  - E.g. cache hit: data will be transferred from the cache to registers.
  - Hit rate/ratio = no of hits / no of total accesses
  - Hit time – the time to access the upper level of the memory hierarchy, which includes the time needed to determine whether the access is a hit or a miss
- In the case of miss
  - blocks will be copied from the lower level, then accessed data supplied to upper level
  - Miss-rate = no of misses / no of total accesses = 1 – hit rate
  - Miss penalty – the time to replace a block in the upper level with the corresponding block from the lower level, plus the time to deliver this block to the processor

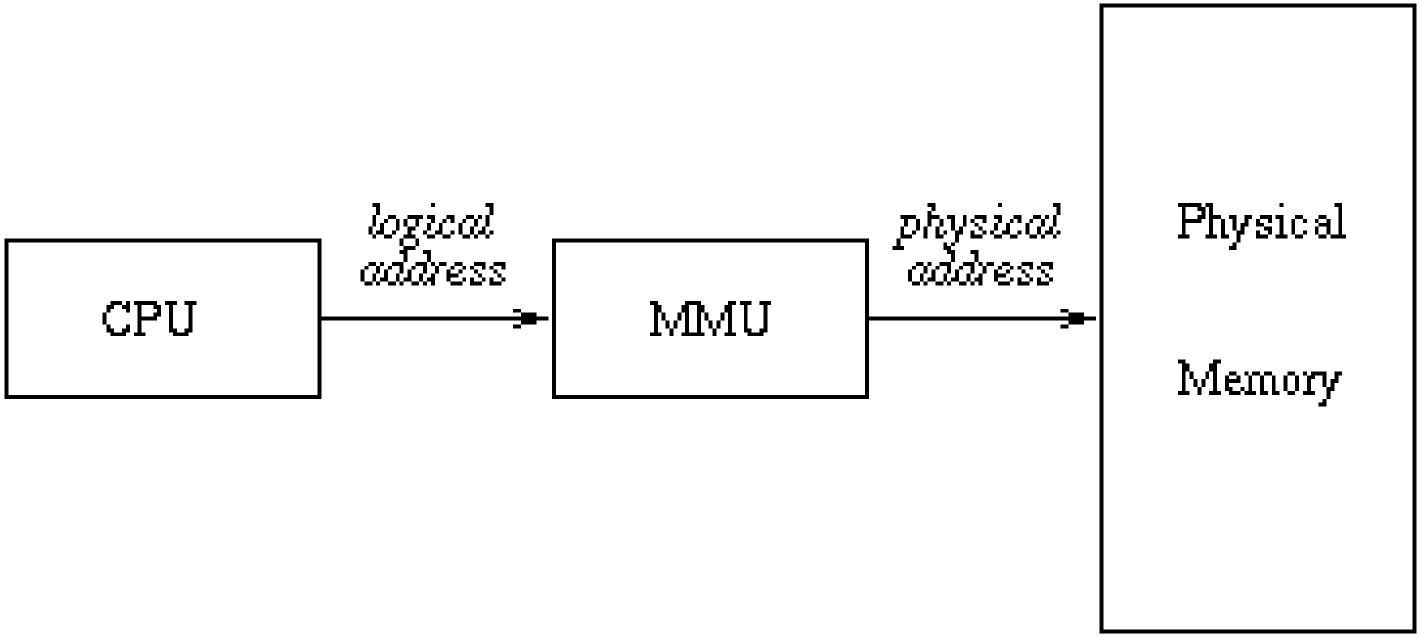
Virtual Memory

Virtual Memory

- Virtual or Logical addresses
- Virtual memory
- Page Replacement Algorithms

Virtual / Logical Addresses

- The addresses generated by the CPU
- The memory management unit will convert these addresses into physical addresses.

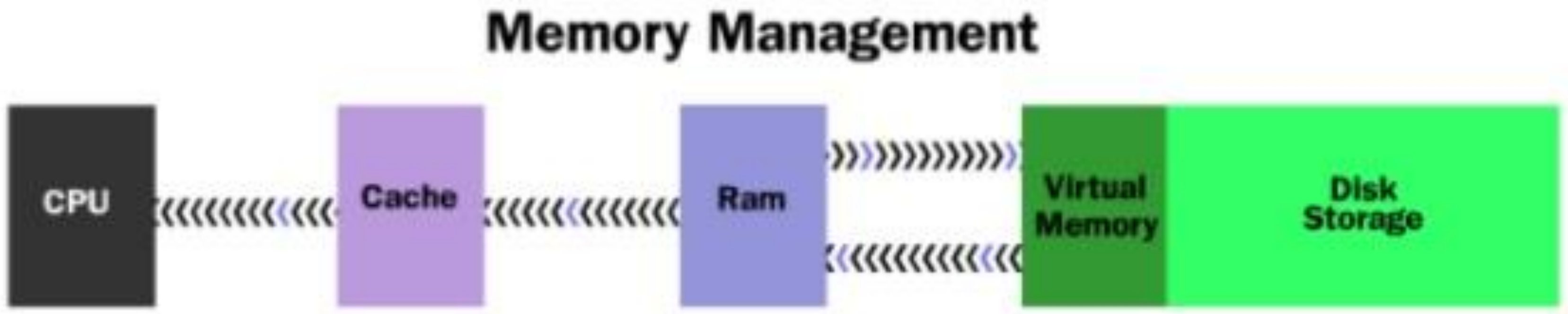


Virtual Memory

- Size of the virtual address space (the address space generated by programs) is much larger than the size of the physical address space (size of RAM).
- The main memory may not be enough to hold a large process.
- Virtual memory allows a computer to compensate for shortages of physical memory by temporarily transferring pages of data from RAM to disk storage.

How it works?

- With virtual memory, what the computer do is to look at RAM for areas that have not been used recently and copy them onto the hard disk. This frees up space in RAM to load the new application.
- The area of hard disk that stores the RAM image is called a **page file**. It holds **pages** of RAM on the hard disk, and the operating system moves data back and forth between the page file and RAM. On a Windows machine, page files have a .SWP extension.



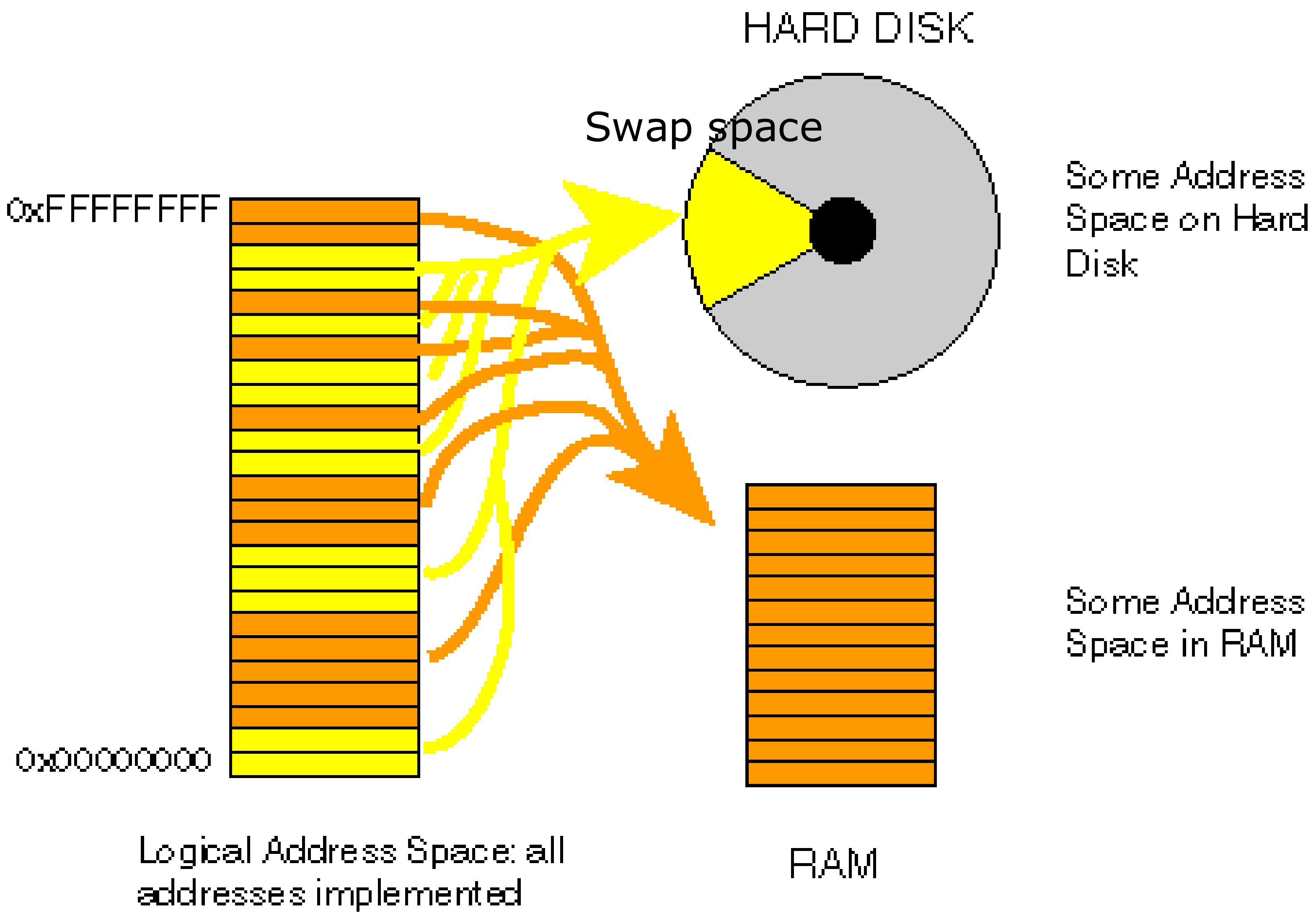
Virtual Memory

RAM

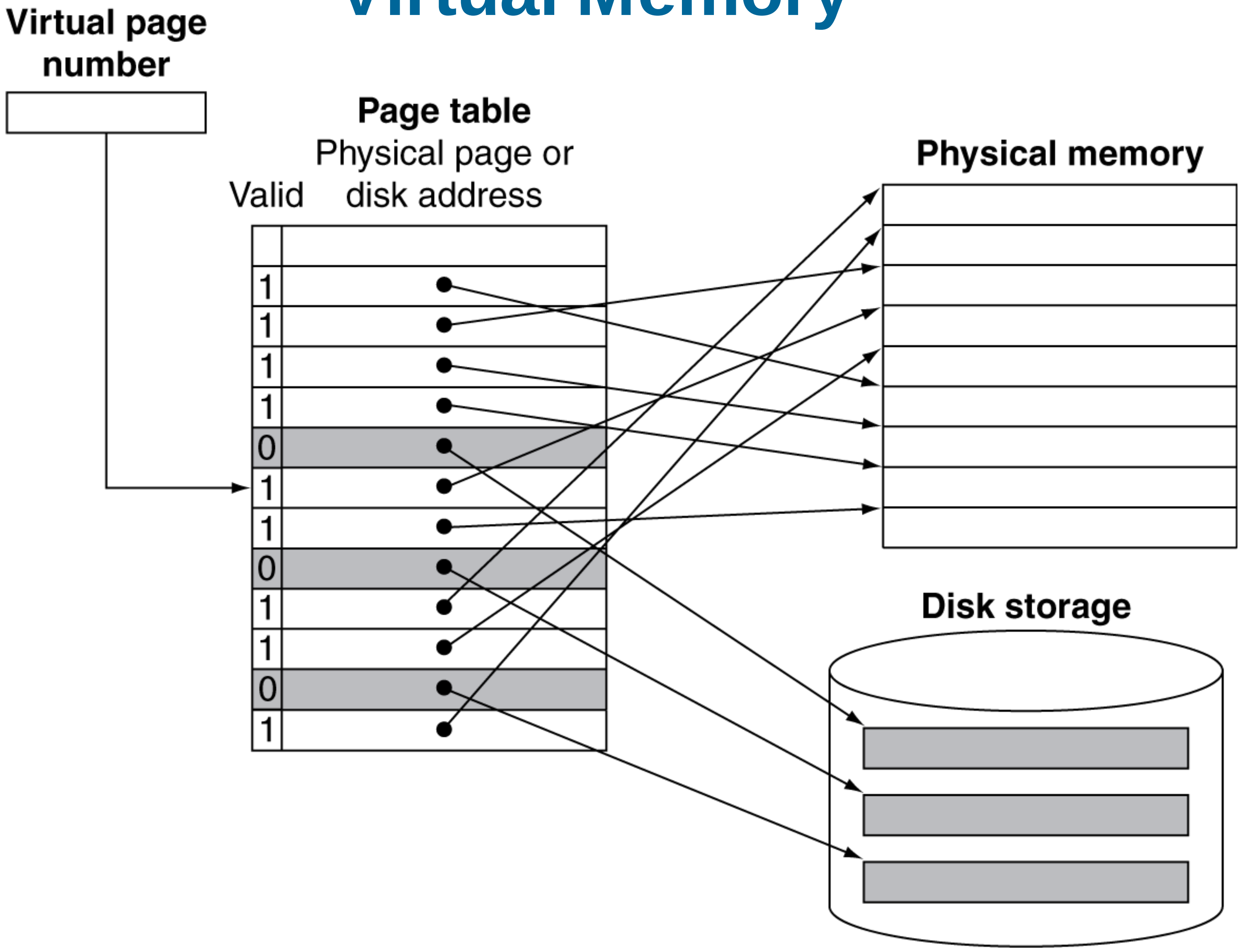
Swap Space

Hard Disk

Virtual Memory


HARD DISK
Swap space
Some Address Space on Hard Disk
0xFFFFFFFF
0x00000000
Some Address Space in RAM
Logical Address Space: all addresses implemented
RAM

Virtual Memory



Valid bit – whether the frame is in the main memory or not.

Virtual Memory

- **Virtual Memory** is a storage mechanism which offers user an illusion of having a very big main memory. It is done by treating a part of secondary memory as the main memory. In Virtual memory, the user can store processes with a bigger size than the available main memory.

- Advantages:
  - Separation of logical memory from physical memory. This allows the programmer to have large logical/virtual address space than the available physical address space.
  - Programmer does not need to worry about the available size of the main memory

Page Fault

- Page fault: The requested page is not in the main memory.

- Steps involved when a page fault occur
  1. Find a free frame in the main memory
  2. Swap the page into the frame via scheduled disk operation

- What if there is no free frame?
  - Use page replacement algorithms

Page and Frame Replacement Algorithms

- **Page-replacement algorithm**
  - Want lowest page-fault rate on both first access and re-access

- Evaluate algorithm by running it on a particular string of memory references (reference string) and computing the number of page faults on that string
  - String is just page numbers, not full addresses
  - Results depend on number of frames available
- In all our examples, the **reference string** of referenced page numbers is

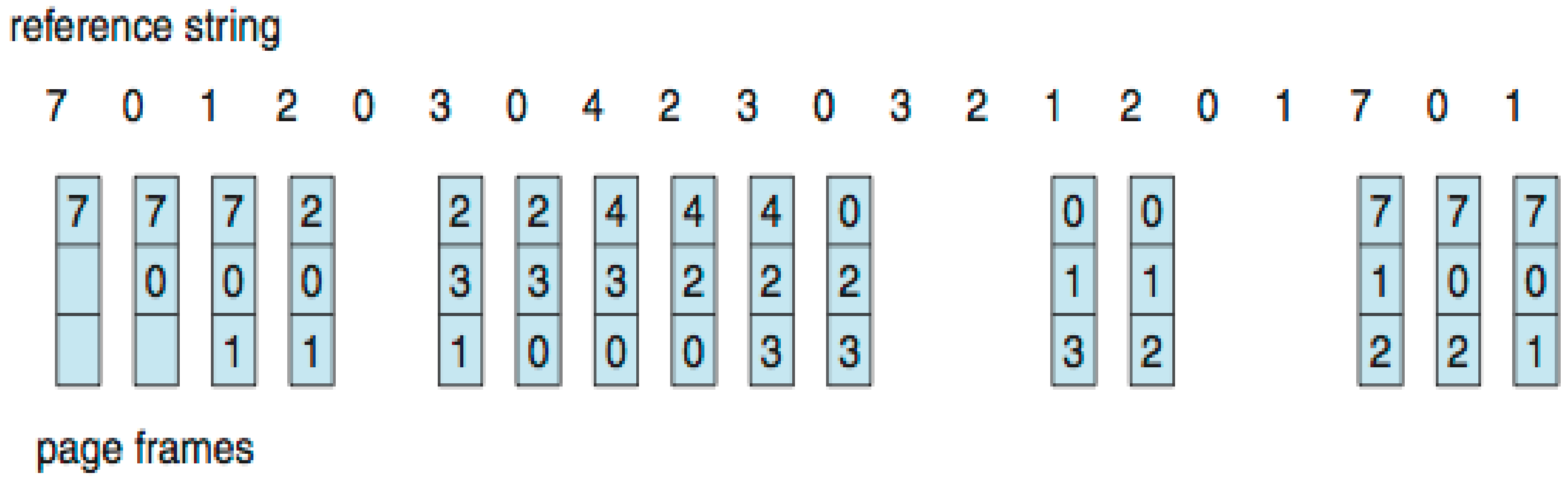
7,0,1,2,0,3,0,4,2,3,0,3,0,3,2,1,2,0,1,7,0,1

Page replacement algorithms

- First-In-First-Out (FIFO) Algorithm
- Optimal page replacement
- Least recently used
- Many more (Least frequently used, Most frequently used, etc.)

First-In-First-Out (FIFO) Algorithm

- Reference string: **7,0,1,2,0,3,0,4,2,3,0,3,2,1,2,0,1,7,0,1**
- 3 frames (3 pages can be in memory at a time per process)



15 page faults

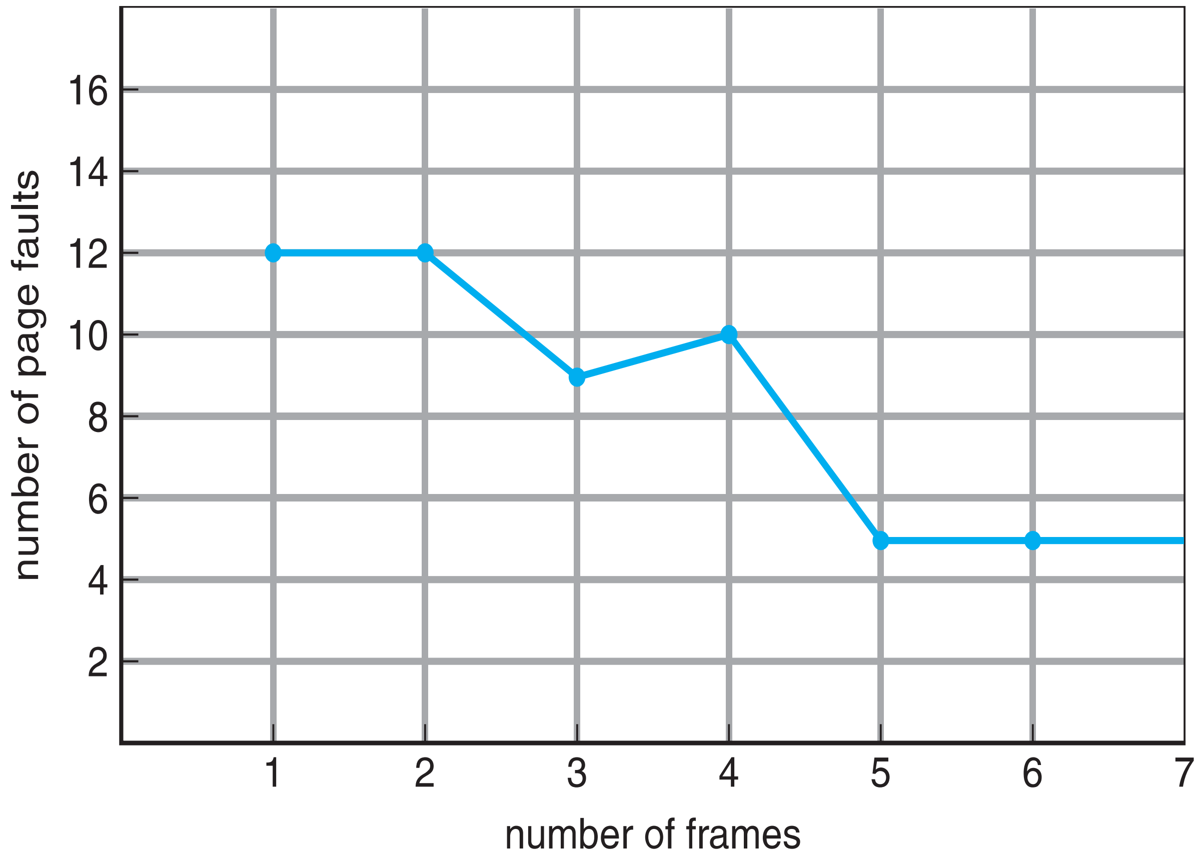
- How to track ages of pages?
  - Just use a FIFO queue

First-In-First-Out (FIFO) Algorithm

- Consider 1,2,3,4,1,2,5,1,2,3,4,5
  - Adding more frames can cause more page faults!
  - 3 frames → 9 faults, 4 frames → 10 faults

- **Belady's Anomaly –** for some page-replacement algorithms, the page fault rate may increase as the no of allocated frames increases.

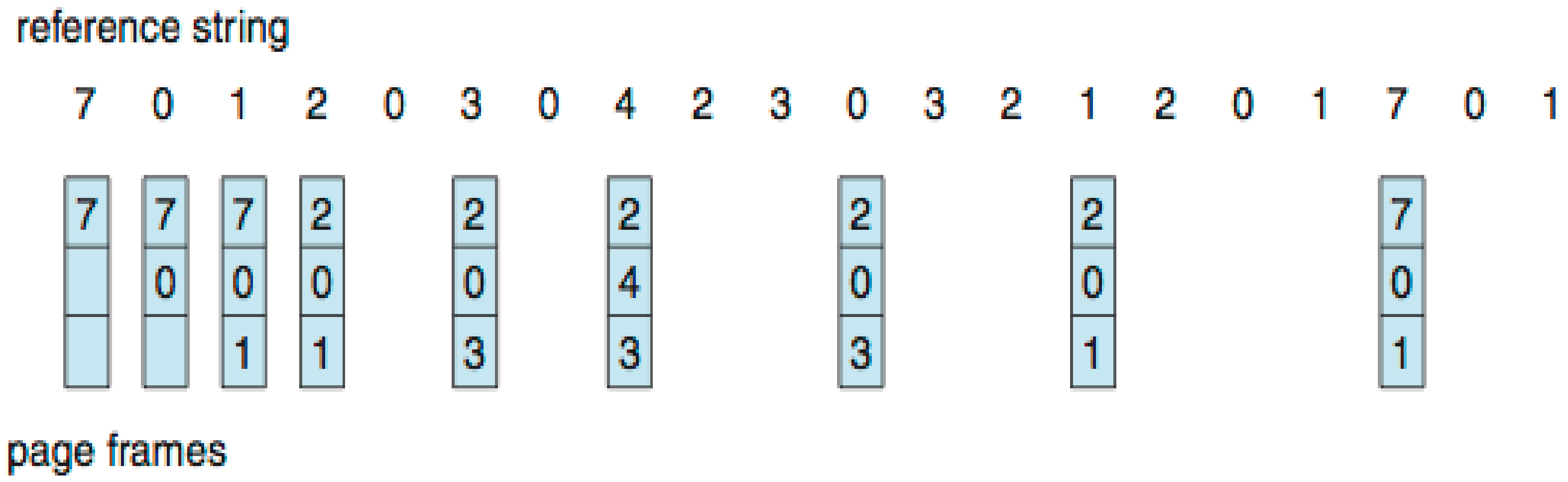
FIFO Illustrating Belady’ s Anomaly



The no of faults for 4 frames is greater than the no of faults for 3 frames – Belady’s anomaly

Optimal page replacement Algorithm

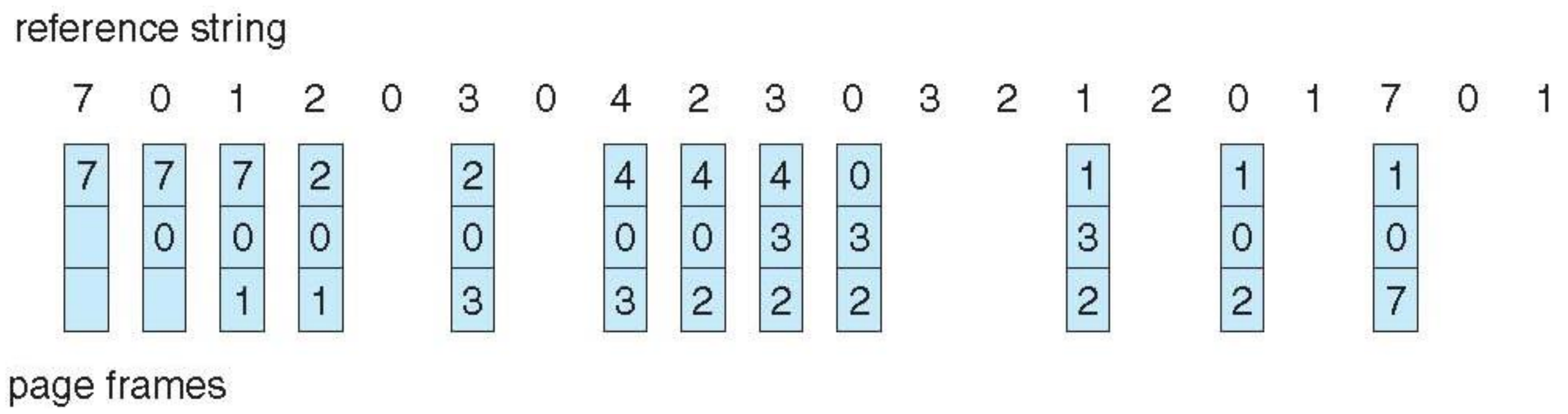
- Replace the page that **will not** be used for a longest period of time



- No. page faults = 9
- No replacement algorithm can process this string in three frames with fewer than nine faults.
- Difficult to implement as it requires future knowledge of the reference string.
- However can be used to compare different algorithms as this gives the minimum no of page faults.

Least Recently Used (LRU) Algorithm

- Use past knowledge rather than future
- Replace a page that has not been used in the most amount of time
- Associate time of last use with each page



- 12 faults – better than FIFO but worse than OPR
- Generally good algorithm and frequently used

Exercise - 1

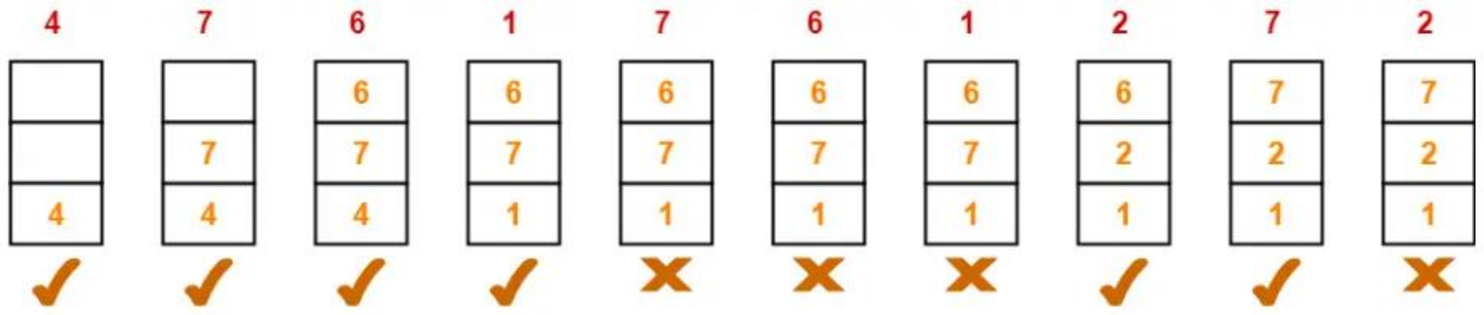
A system uses **3 page frames** for storing process pages in main memory. It uses the **First in First out (FIFO)** page replacement policy. Assume that all the page frames are initially empty. What is the total number of page faults that will occur while processing the page reference string given below?

4, 7, 6, 1, 7, 6, 1, 2, 7, 2

Also, calculate the hit ratio and miss ratio.

Solution

Total number of references = 10



Total number of page faults occurred = 6

Hit ratio = Total number of page hits / Total number of references
= (10-6) / 10 = 0.4 or 40%

Miss ratio = Total number of page misses / Total number of references
= 6 / 10 = 0.6 or 60%

Exercise - 2

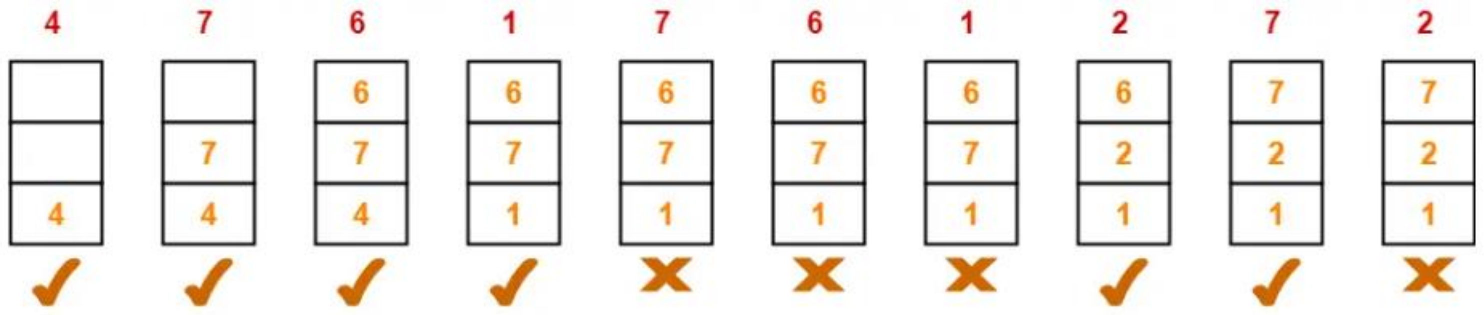
A system uses **3 page frames** for storing process pages in main memory. It uses the **Least Recently Used (LRU)** page replacement policy. Assume that all the page frames are initially empty. What is the total number of page faults that will occur while processing the page reference string given below?

4, 7, 6, 1, 7, 6, 1, 2, 7, 2

Also, calculate the hit ratio and miss ratio.

Solution

Total number of references = 10



Total number of page faults occurred = 6

Hit ratio = Total number of page hits / Total number of references
= (10-6) / 10 = 0.4 or 40%

Miss ratio = Total number of page misses / Total number of references
= 6 / 10 = 0.6 or 60%

Exercise - 3

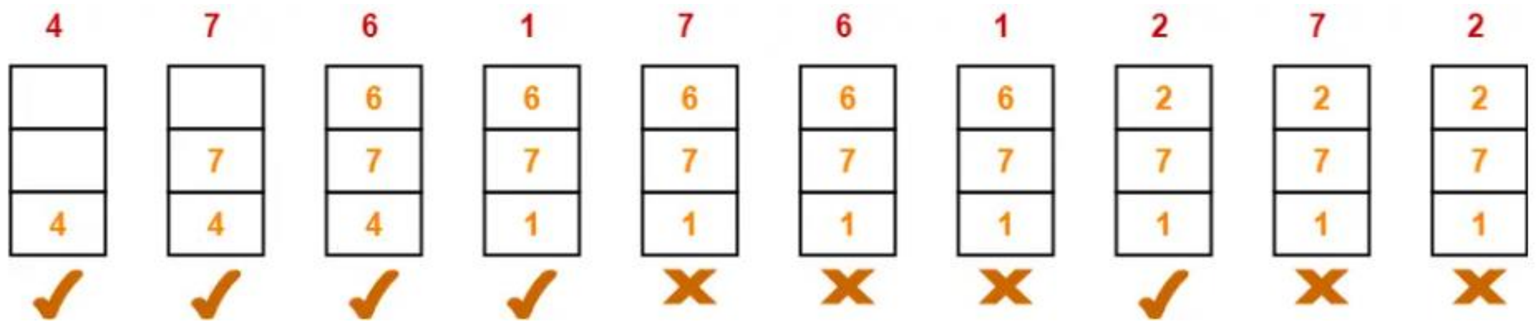
A system uses **3 page frames** for storing process pages in main memory. It uses the **Optimal** page replacement policy. Assume that all the page frames are initially empty. What is the total number of page faults that will occur while processing the page reference string given below?

4, 7, 6, 1, 7, 6, 1, 2, 7, 2

Also, calculate the hit ratio and miss ratio.

Solution

Total number of references = 10



Total number of page faults occurred = 5

Hit ratio = Total number of page hits / Total number of references
= (10-5) / 10 = 0.5 or 50%

Miss ratio = 1 – Hit ratio = 1 – 0.5 = 0.5 or 50%

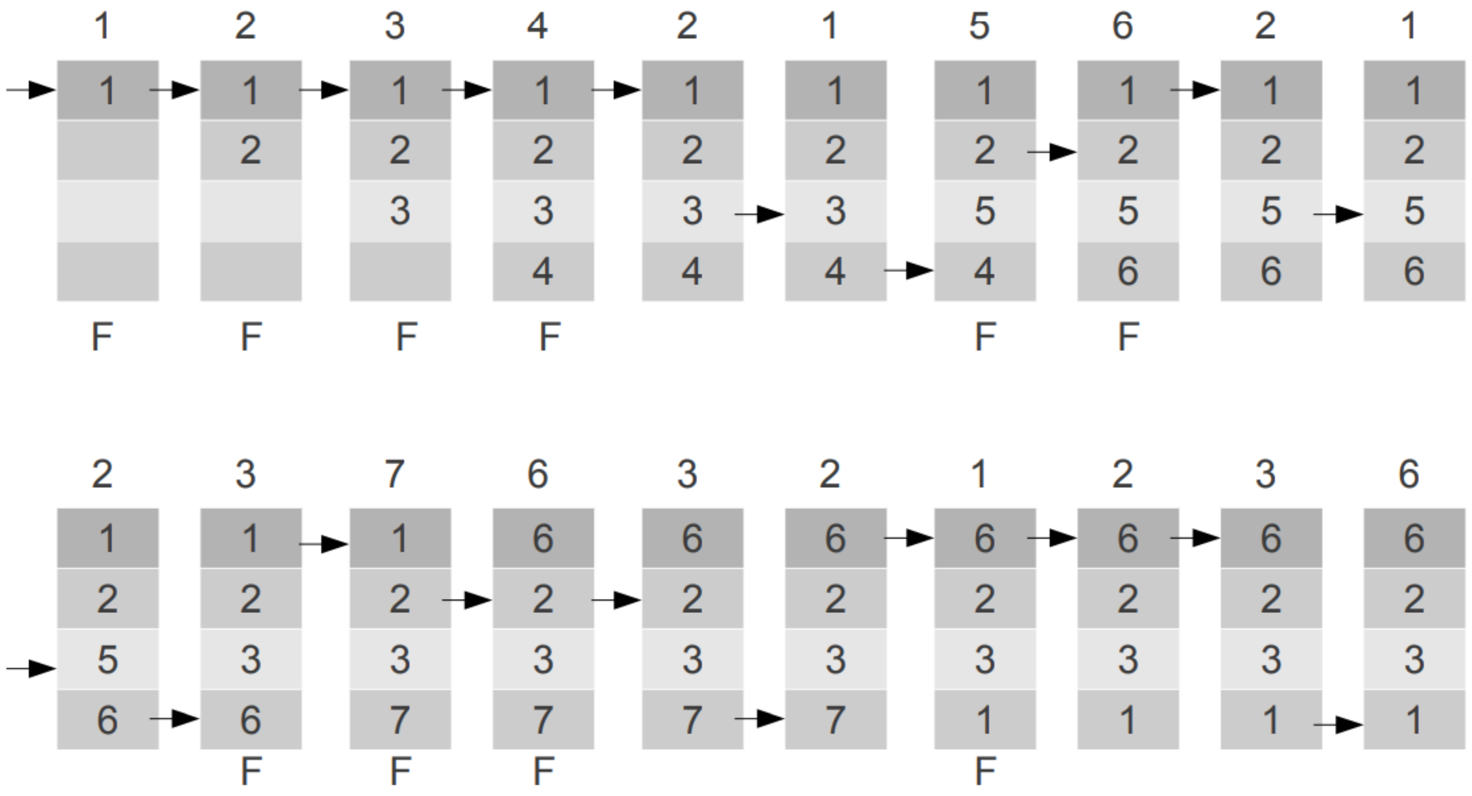
Exercise - 4

Given page reference string: 1,2,3,4,2,1,5,6,2,1,2,3,7,6,3,2,1,2,3,6

Compare the number of page faults for LRU, FIFO, and Optimal page replacement algorithm in 4 Frames.

Solution

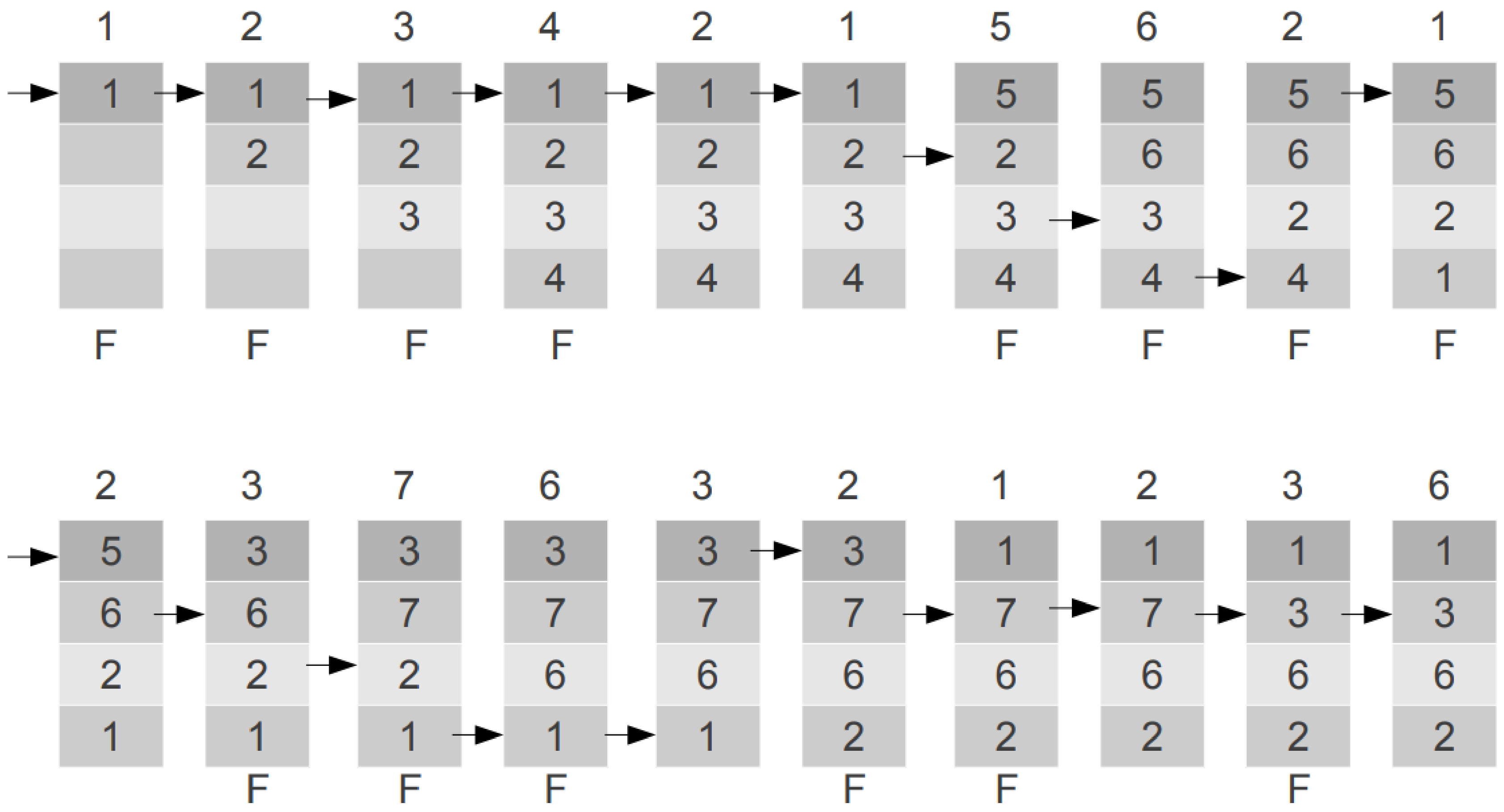
LRU



Total number of page faults occurred = 10

Solution

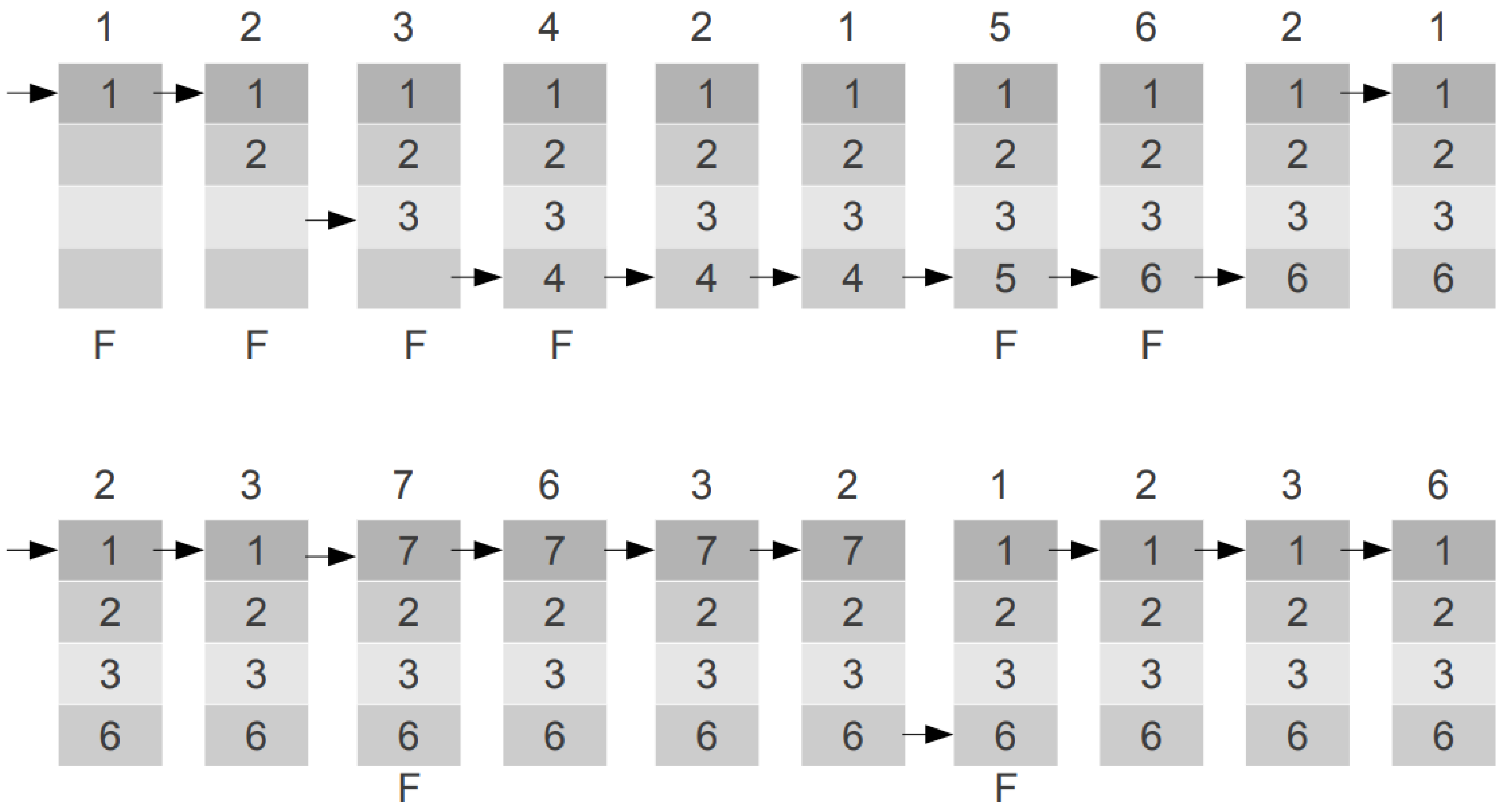
FIFO



Total number of page faults occurred = 14

Solution

Optimal



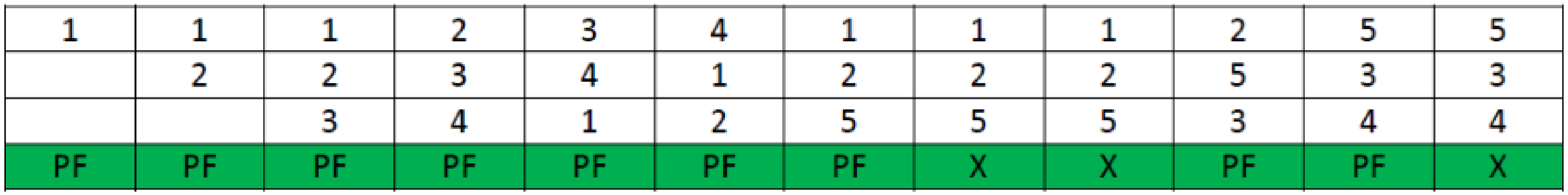
Total number of page faults occurred = 8

Exercise - 5

Assuming a system that has no pages loaded in the memory and uses the FIFO Page replacement algorithm. Consider the following reference string:

```
1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5
```

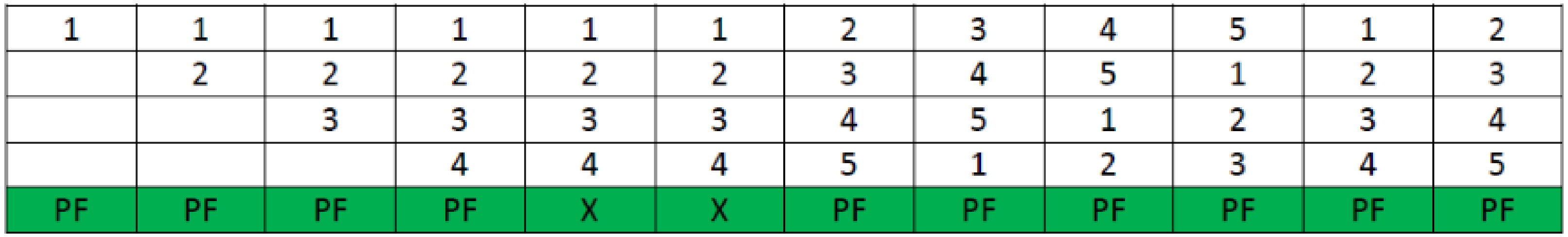
Case-1: If the system has 3 frames



1	1	1	2	3	4	1	1	1	2	5	5
	2	2	3	4	1	2	2	2	5	3	3
		3	4	1	2	5	5	5	3	4	4
PF	PF	PF	PF	PF	PF	PF	X	X	PF	PF	X

Total number of page faults occurred = 9

Case-2: If the system has 4 frames

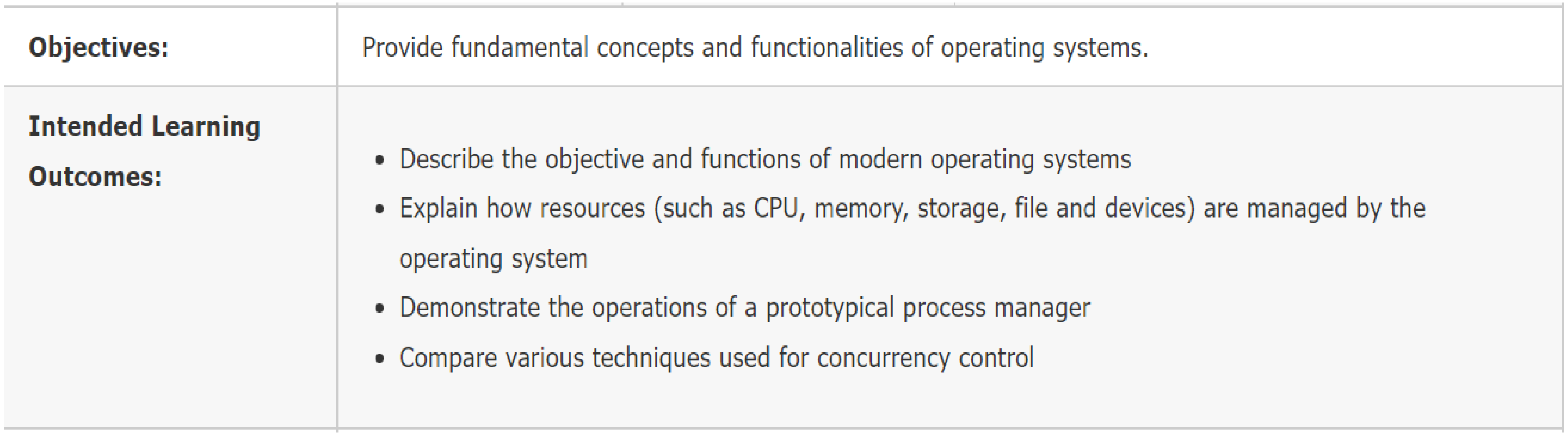


1	1	1	1	1	1	2	3	4	5	1	2
	2	2	2	2	2	3	4	5	1	2	3
		3	3	3	3	4	5	1	2	3	4
			4	4	4	5	1	2	3	4	5
PF	PF	PF	PF	X	X	PF	PF	PF	PF	PF	PF

Total number of page faults occurred = 10

End of Chapter

ILOs



Objectives:	Provide fundamental concepts and functionalities of operating systems.
Intended Learning Outcomes:	• Describe the objective and functions of modern operating systems • Explain how resources (such as CPU, memory, storage, file and devices) are managed by the operating system • Demonstrate the operations of a prototypical process manager • Compare various techniques used for concurrency control