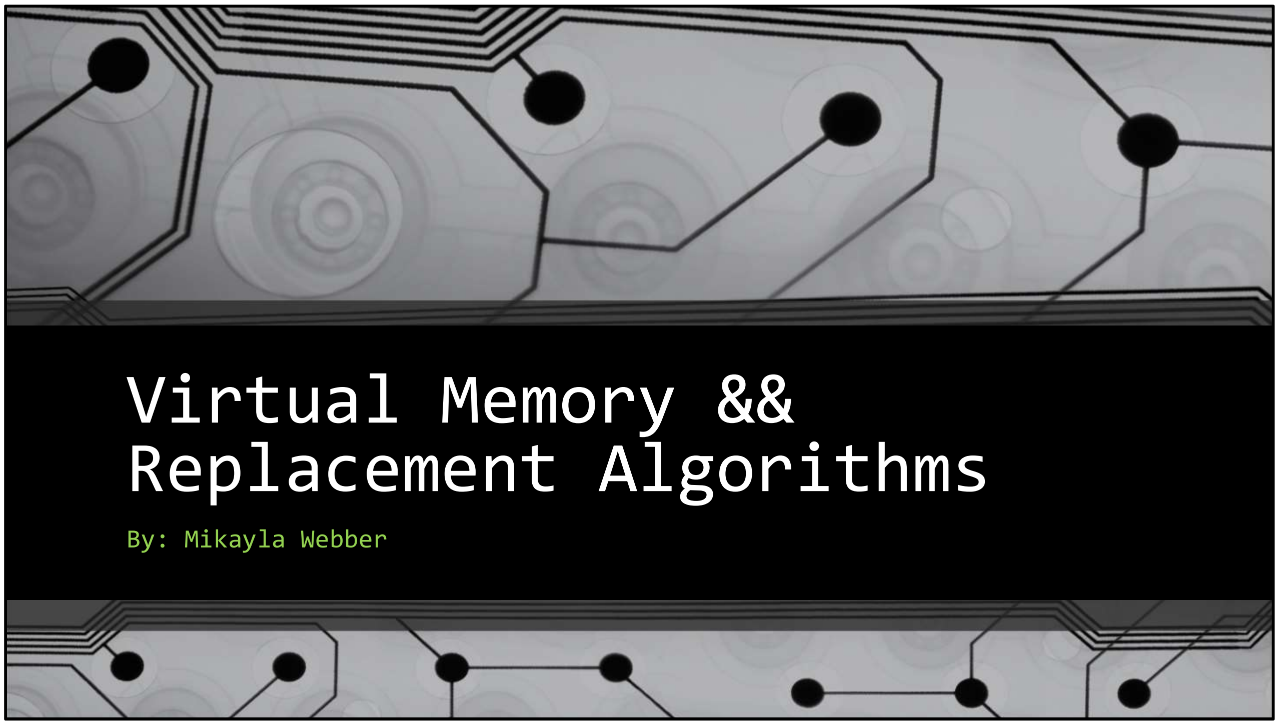
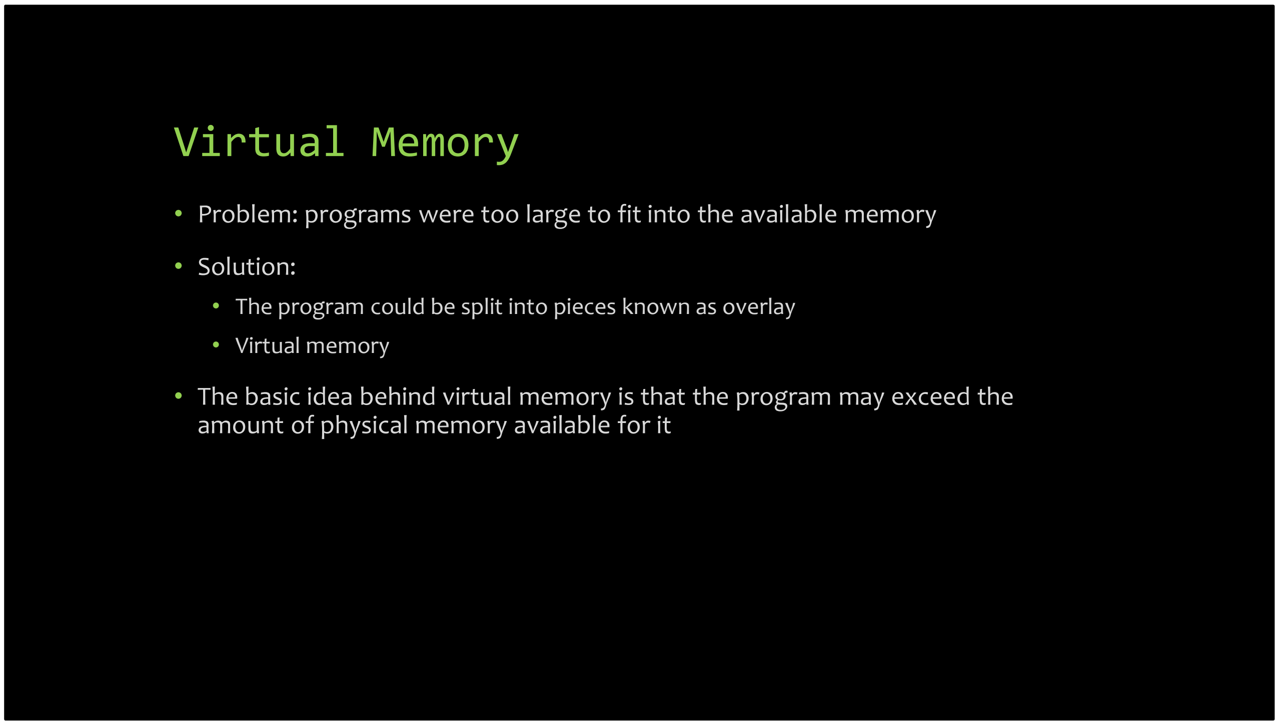


Virtual Memory && Replacement Algorithms

By: Mikayla Webber

Overview

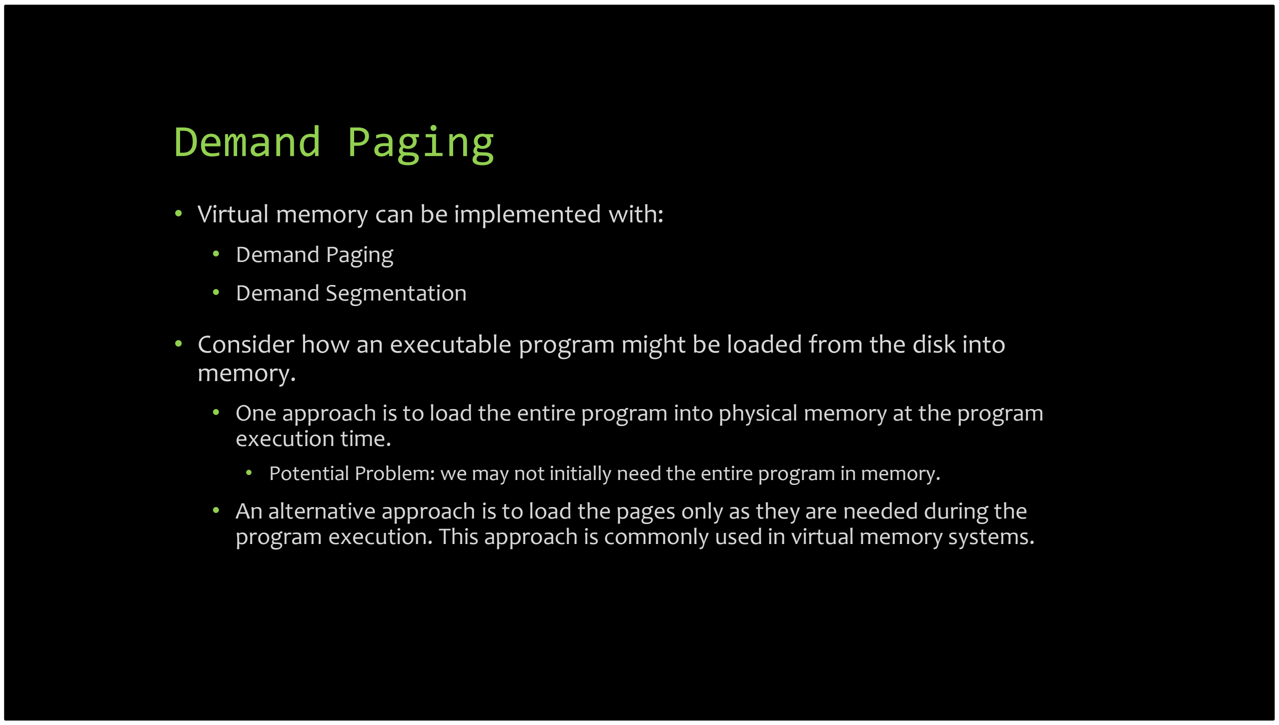
- Background
- Demand Paging
- Page Faults
- Page Replacement Algorithms
  - First-in, First-out (FIFO)
  - Optimal Page Replacement (OPT)
  - Least Recently Used (LRU)



Virtual Memory

- Problem: programs were too large to fit into the available memory
- Solution:
  - The program could be split into pieces known as overlay
  - Virtual memory
- The basic idea behind virtual memory is that the program may exceed the amount of physical memory available for it

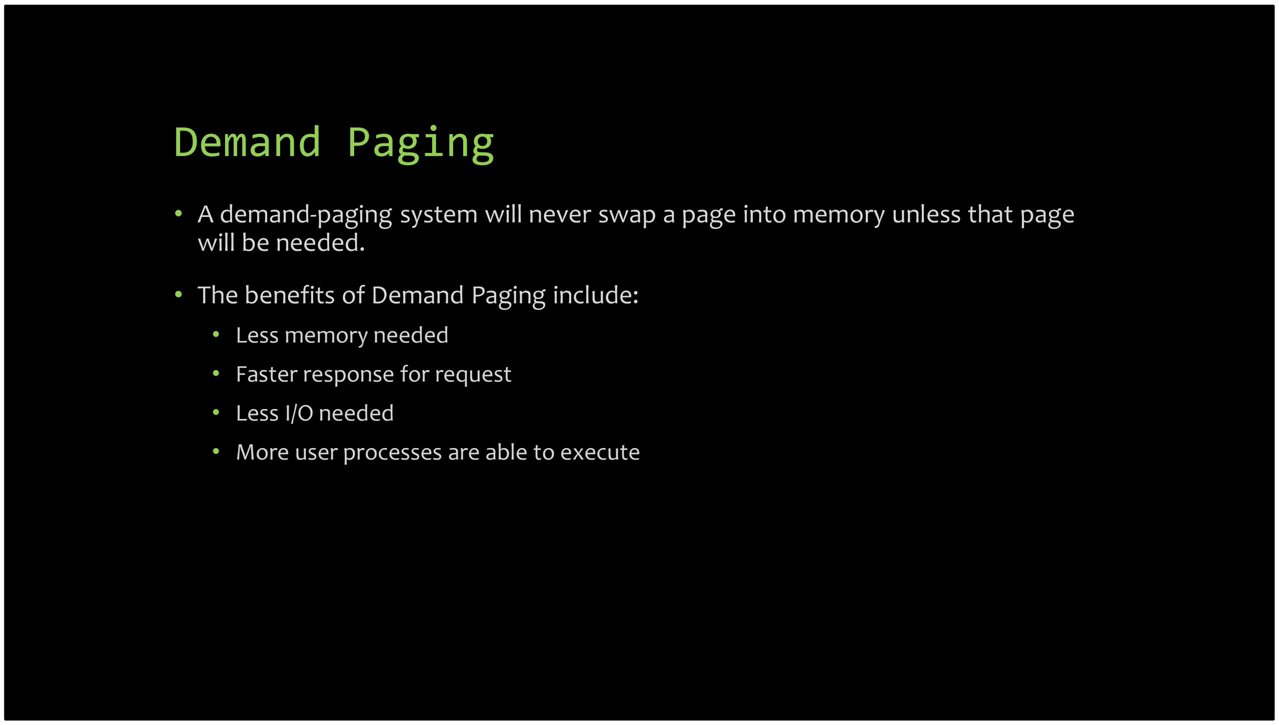
Demand Paging



Demand Paging

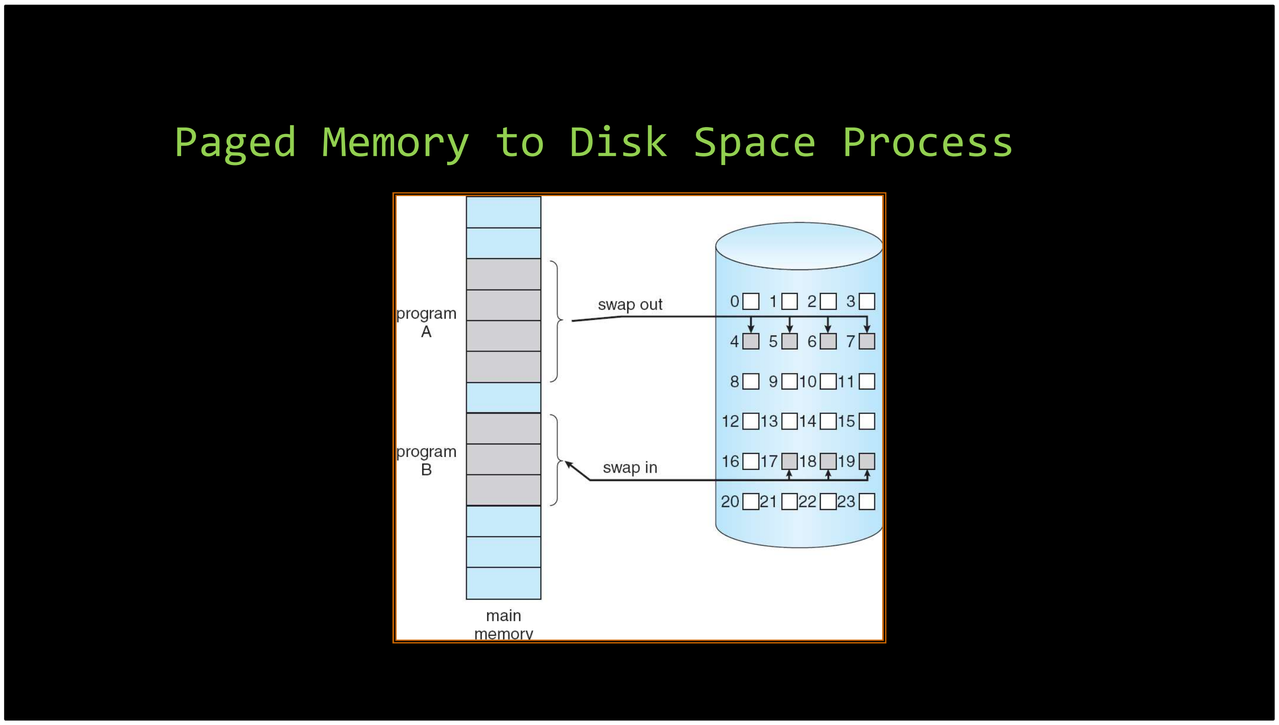
- Virtual memory can be implemented with:
  - Demand Paging
  - Demand Segmentation
- Consider how an executable program might be loaded from the disk into memory.
  - One approach is to load the entire program into physical memory at the program execution time.
    - Potential Problem: we may not initially need the entire program in memory.
  - An alternative approach is to load the pages only as they are needed during the program execution. This approach is commonly used in virtual memory systems.

Include example: If a program starts with a list of available options (like a menu) from which the user is to select. Loading the entire program into memory results in loading the executable code for all of the options, regardless of whether an option is ultimately selected by the user or not.

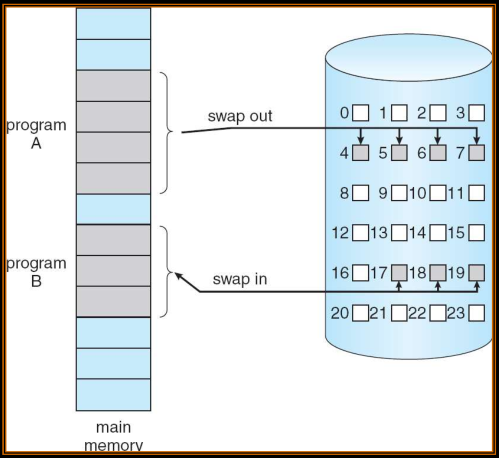


Demand Paging

- A demand-paging system will never swap a page into memory unless that page will be needed.
- The benefits of Demand Paging include:
  - Less memory needed
  - Faster response for request
  - Less I/O needed
  - More user processes are able to execute

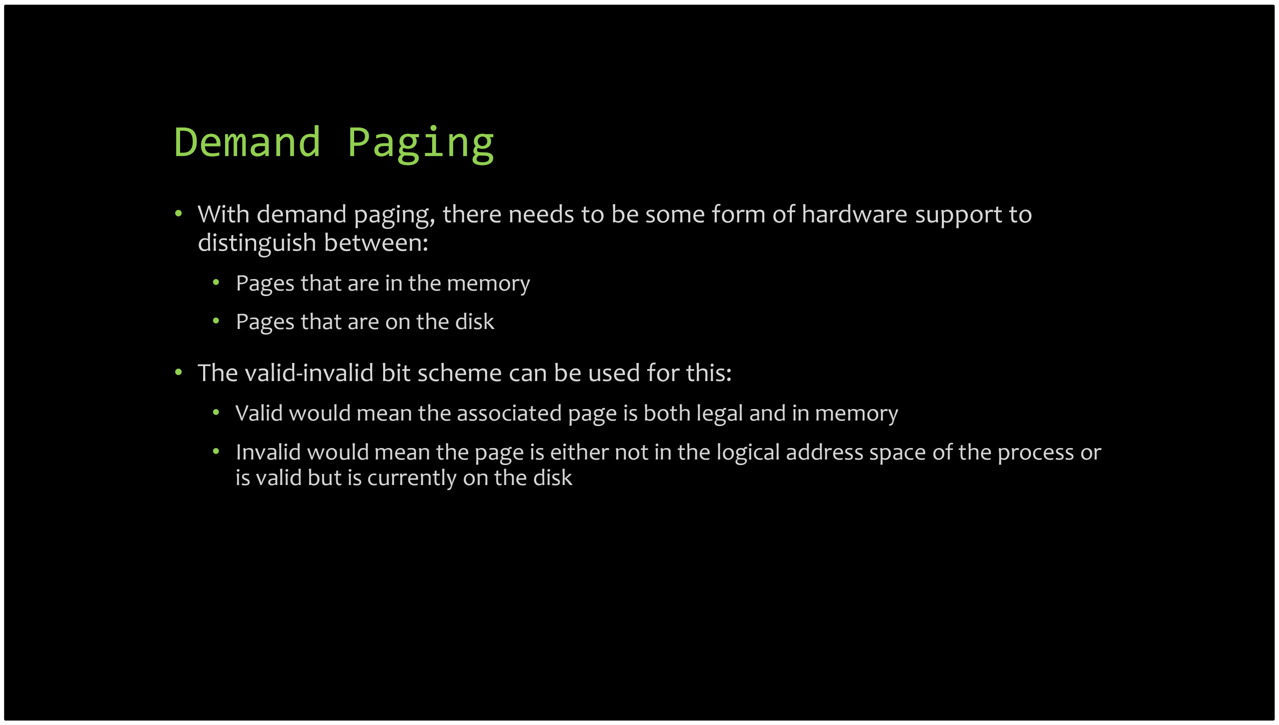


Paged Memory to Disk Space Process



contiguous disk space

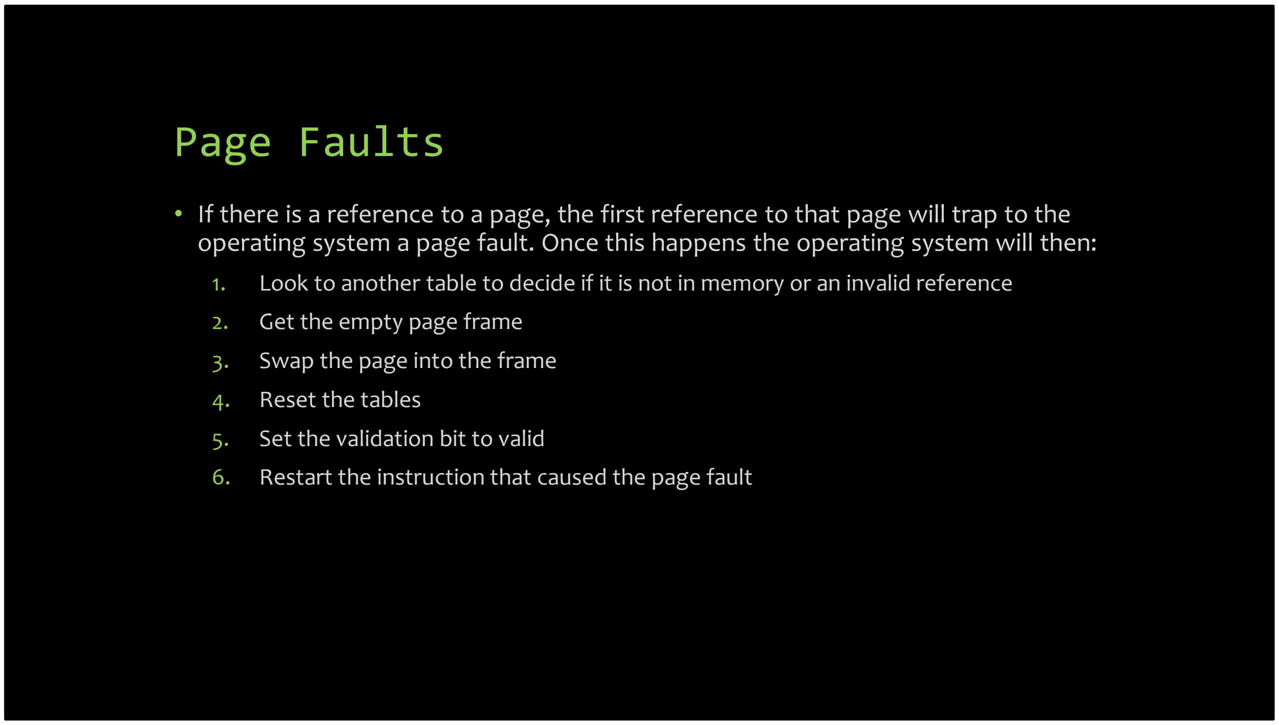
Page is needed -> reference to it
If invalid reference -> abort
Not-in-memory -> bring to memory



Demand Paging

- With demand paging, there needs to be some form of hardware support to distinguish between:
  - Pages that are in the memory
  - Pages that are on the disk
- The valid-invalid bit scheme can be used for this:
  - Valid would mean the associated page is both legal and in memory
  - Invalid would mean the page is either not in the logical address space of the process or is valid but is currently on the disk

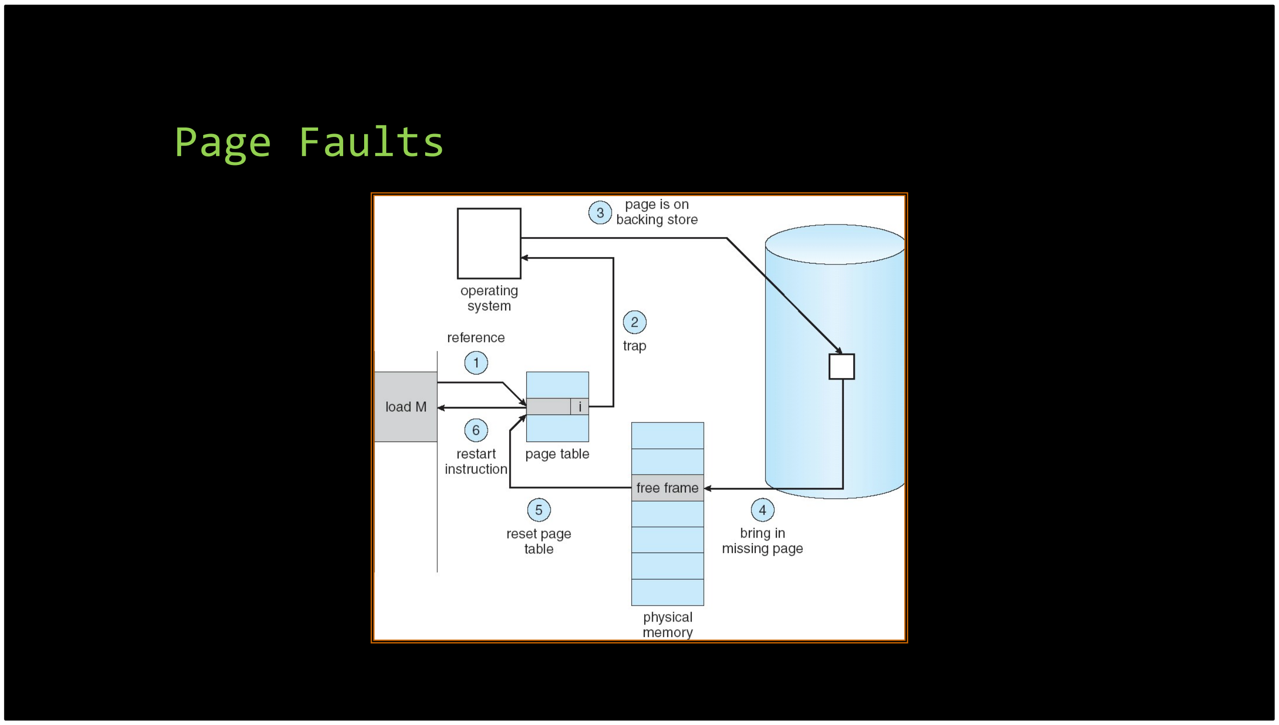
For a page table, initially the valid-invalid bit is set to invalid (not in memory) for all entries
Access to an invalid page -> page-fault trap



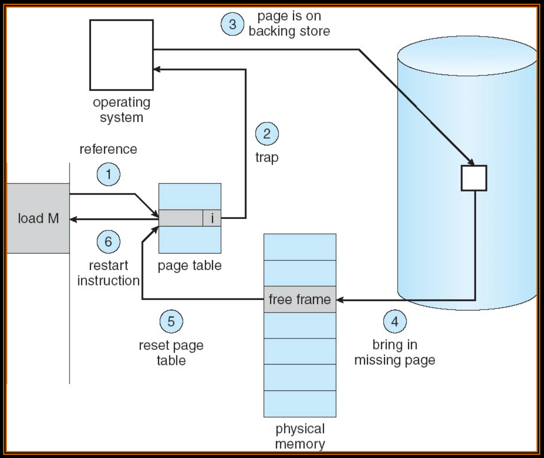
Page Faults

- If there is a reference to a page, the first reference to that page will trap to the operating system a page fault. Once this happens the operating system will then:
  1. Look to another table to decide if it is not in memory or an invalid reference
  2. Get the empty page frame
  3. Swap the page into the frame
  4. Reset the tables
  5. Set the validation bit to valid
  6. Restart the instruction that caused the page fault

- Definition: a page fault is a type of exception that is raised when a running program accesses a memory page that is not currently mapped into the virtual address space of the process.
- Not currently mapped by the MMU (memory management unit)
- For number one if it is an Invalid reference -> abort



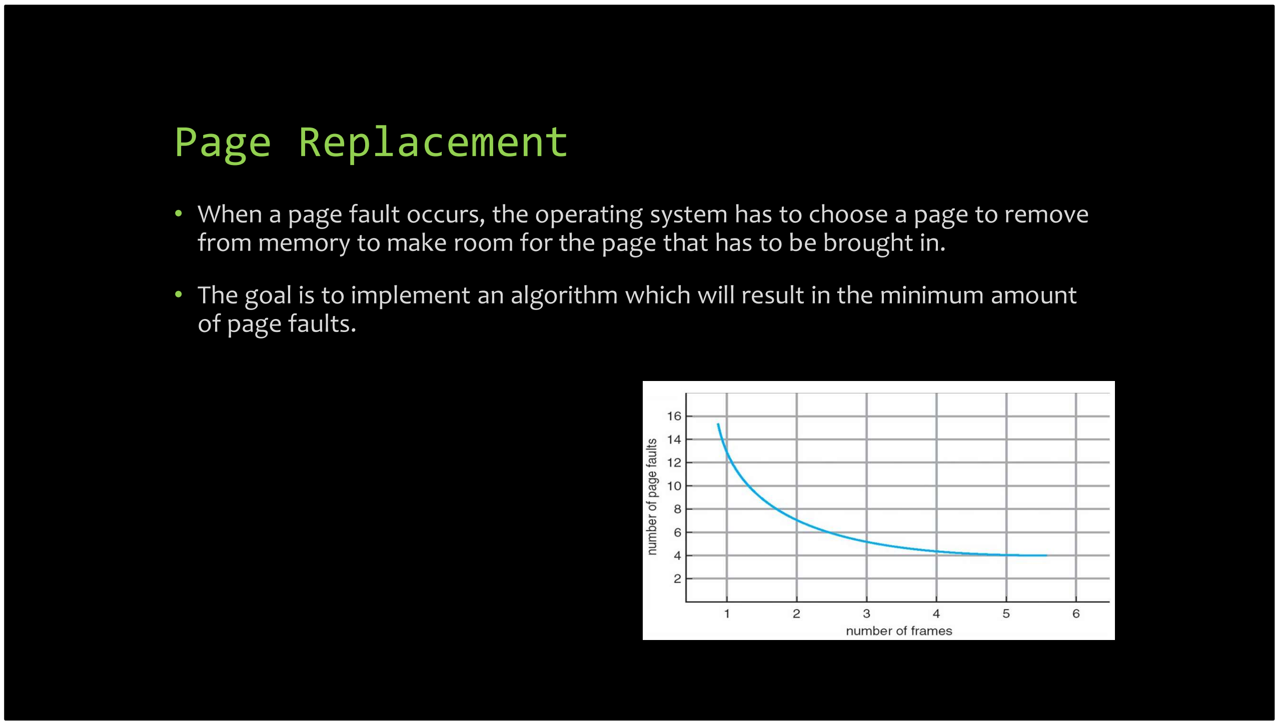
Page Faults



Repeat steps from previous slide

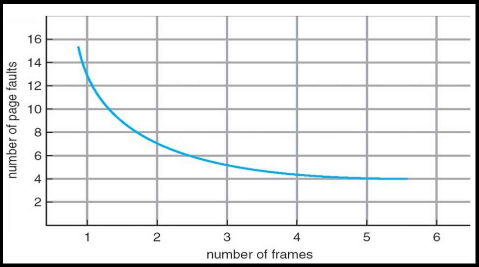
- Demand paging can significantly affect the performance of a computer system because if there are no page faults the effective access time is equal to the memory access time. However, if a page fault occurs we must first read the relevant page from disk and then access the desired word.
    1. Look to another table to decide if it is not in memory or an invalid reference
    2. Get the empty page frame
    3. Swap the page into the frame
    4. Reset the tables
    5. Set the validation bit to valid
    6. Restart the instruction that caused the page fault

Page Replacement Algorithms

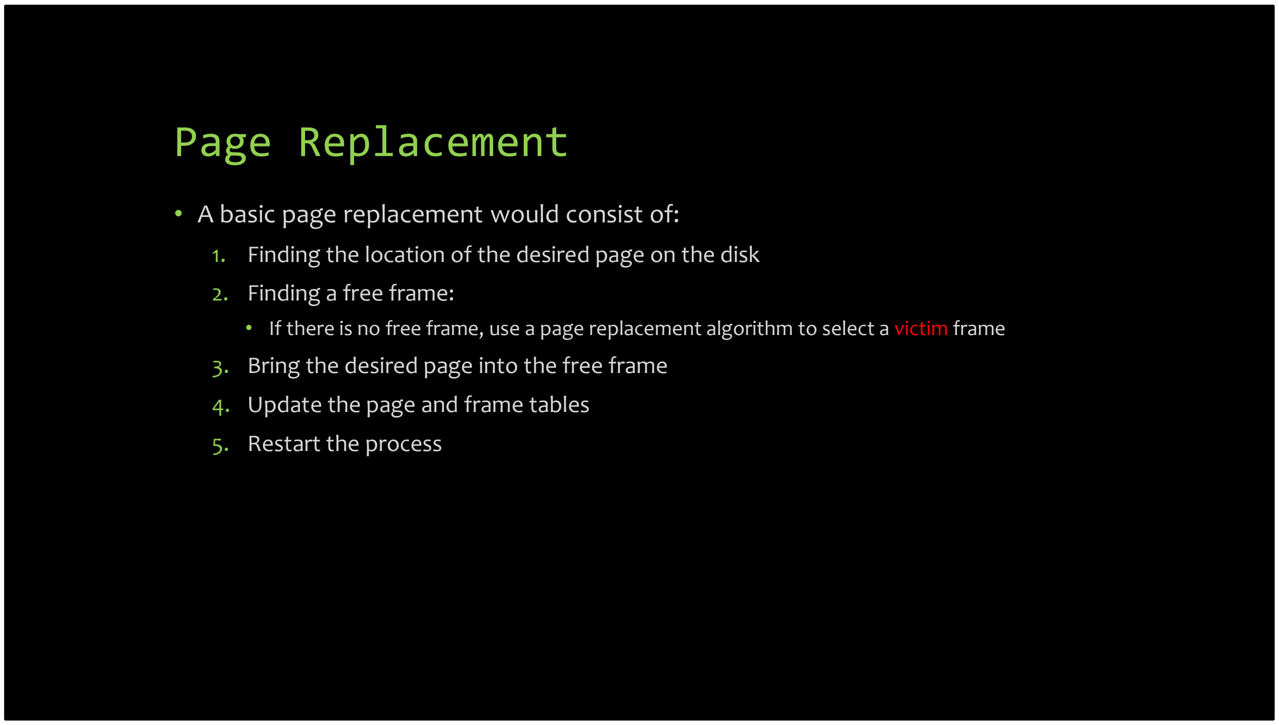


Page Replacement

- When a page fault occurs, the operating system has to choose a page to remove from memory to make room for the page that has to be brought in.
- The goal is to implement an algorithm which will result in the minimum amount of page faults.



As the number of frames increases, the number of page faults drops to some minimal level
adding physical memory increases the number of frames)

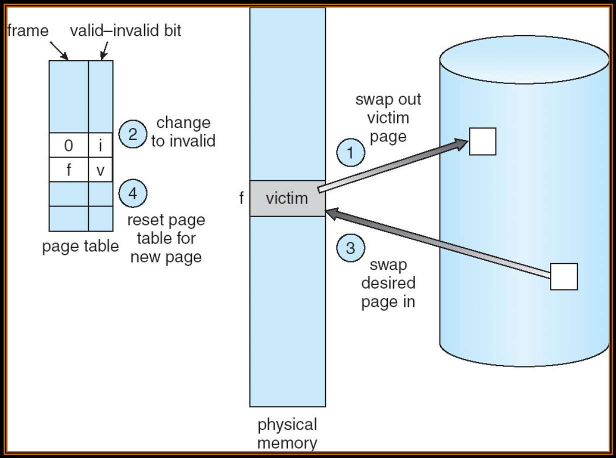


Page Replacement

- A basic page replacement would consist of:
  1. Finding the location of the desired page on the disk
  2. Finding a free frame:
     - If there is no free frame, use a page replacement algorithm to select a victim frame
  3. Bring the desired page into the free frame
  4. Update the page and frame tables
  5. Restart the process

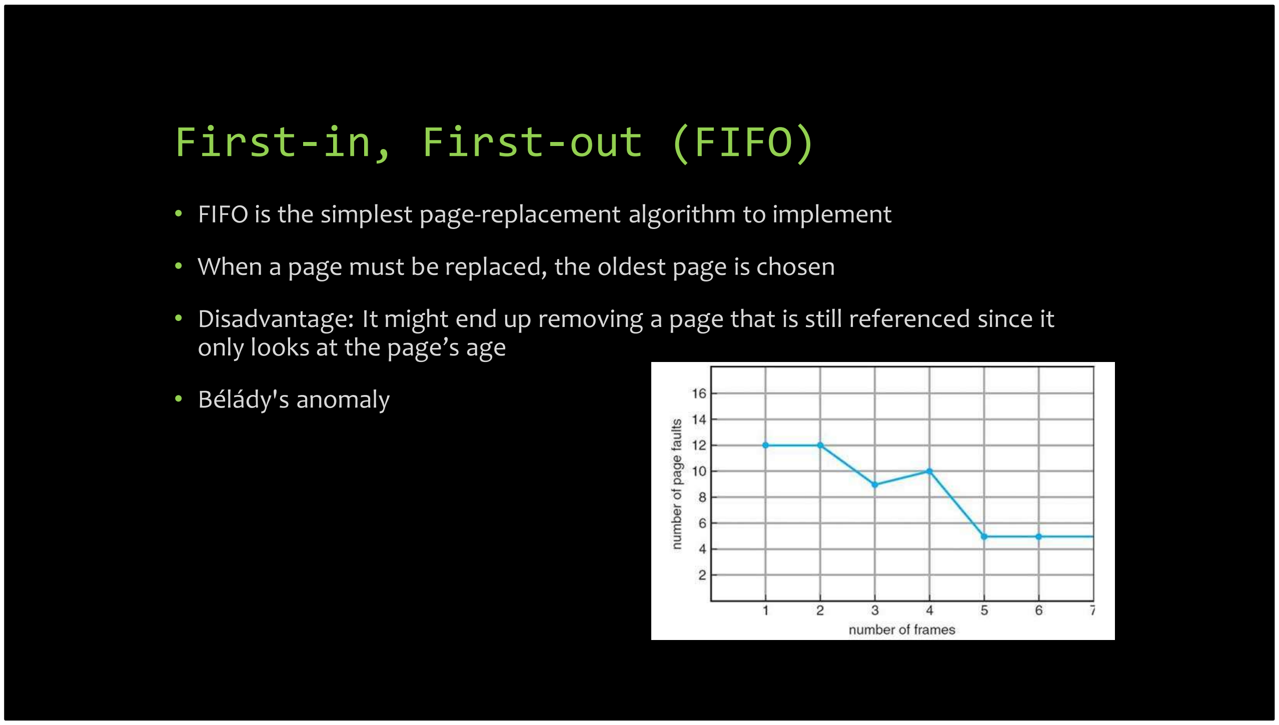
Discuss what a basic page replacement would entail.
- A frame that is replaced is considered a victim frame.

Page Replacement



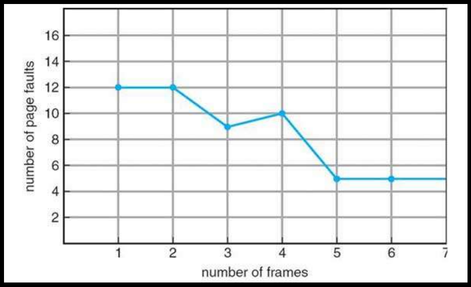
1. Finding the location of the desired page on the disk
2. Finding a free frame:

   If there is no free frame, use a page replacement algorithm to select a victim frame
3. Bring the desired page into the free frame
4. Update the page and frame tables
5. Restart the process

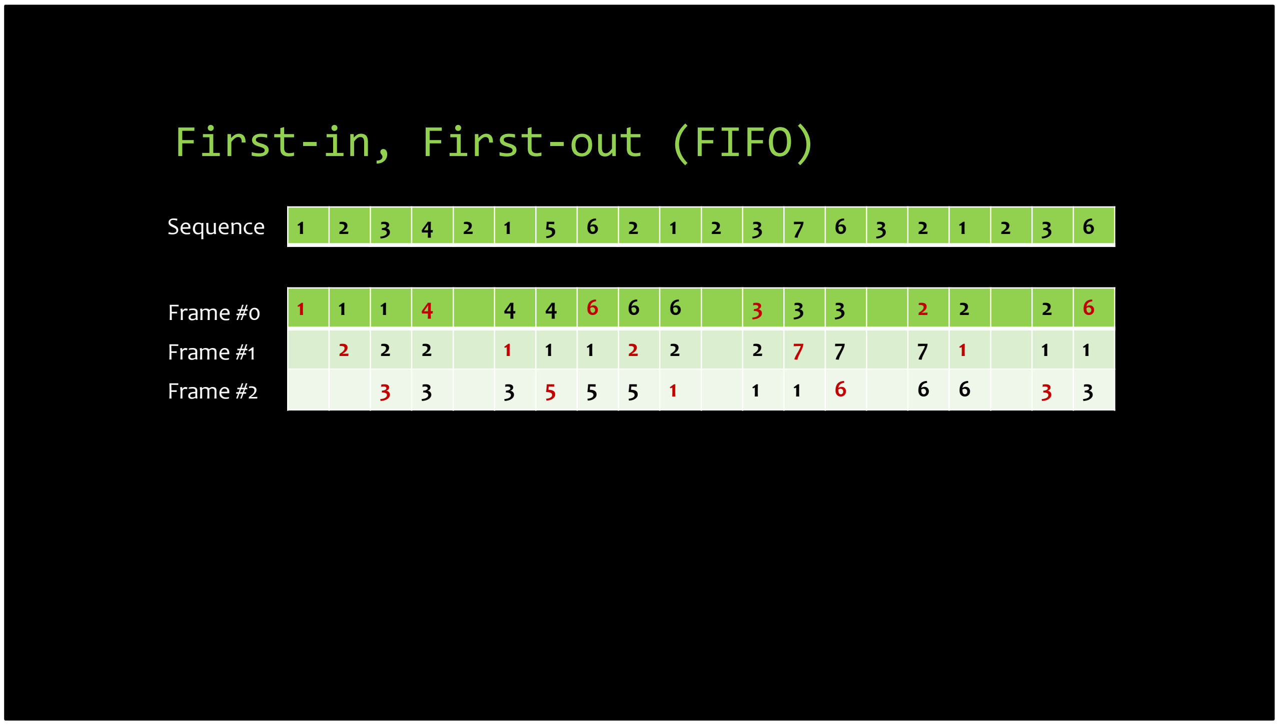


First-in, First-out (FIFO)

- FIFO is the simplest page-replacement algorithm to implement
- When a page must be replaced, the oldest page is chosen
- Disadvantage: It might end up removing a page that is still referenced since it only looks at the page’s age
- Bélády's anomaly



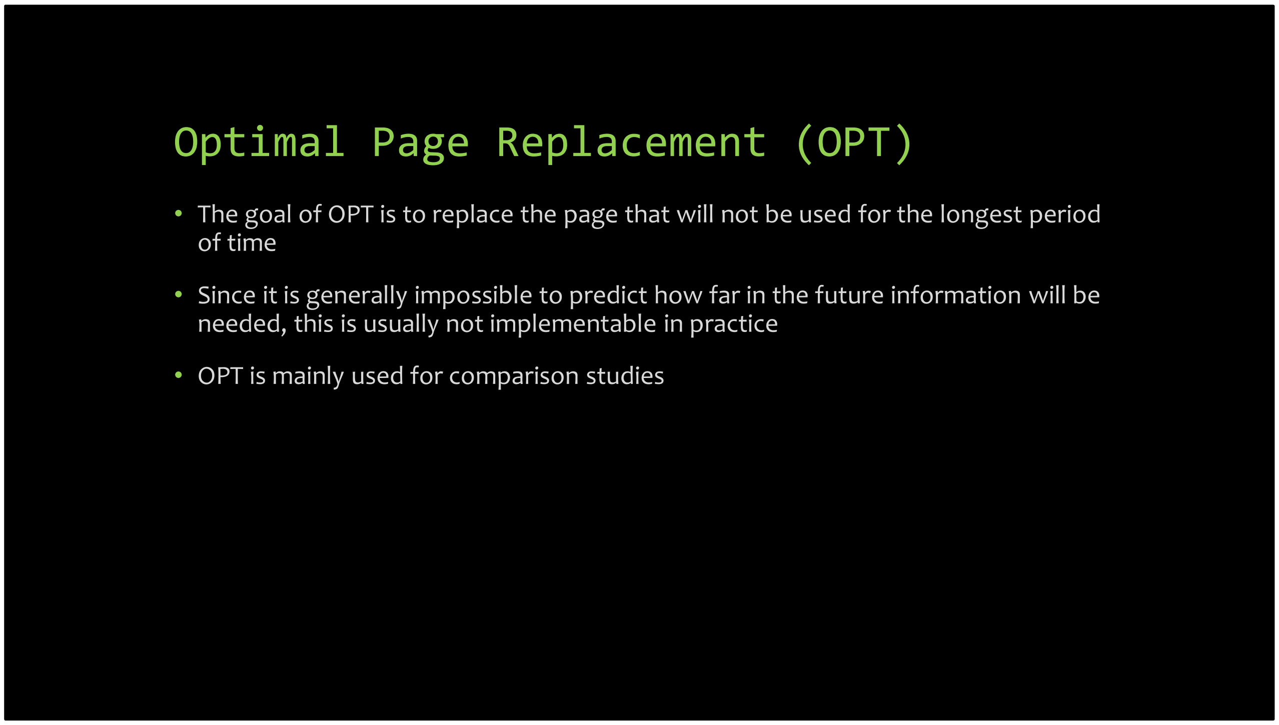
- It might throw out important pages
- Not particularly optimal
- Belady’s anomaly – when the size of the page table increases the page faults increase



First-in, First-out (FIFO)

Sequence	1	2	3	4	2	1	5	6	2	1	2	3	7	6	3	2	1	2	3	6
Frame #0	1	1	1	4		4	4	6	6	6		3	3	3		2	2		2	6
Frame #1		2	2	2		1	1	1	2	2		2	7	7		7	1		1	1
Frame #2			3	3		3	5	5	5	1		1	1	6		6	6		3	3

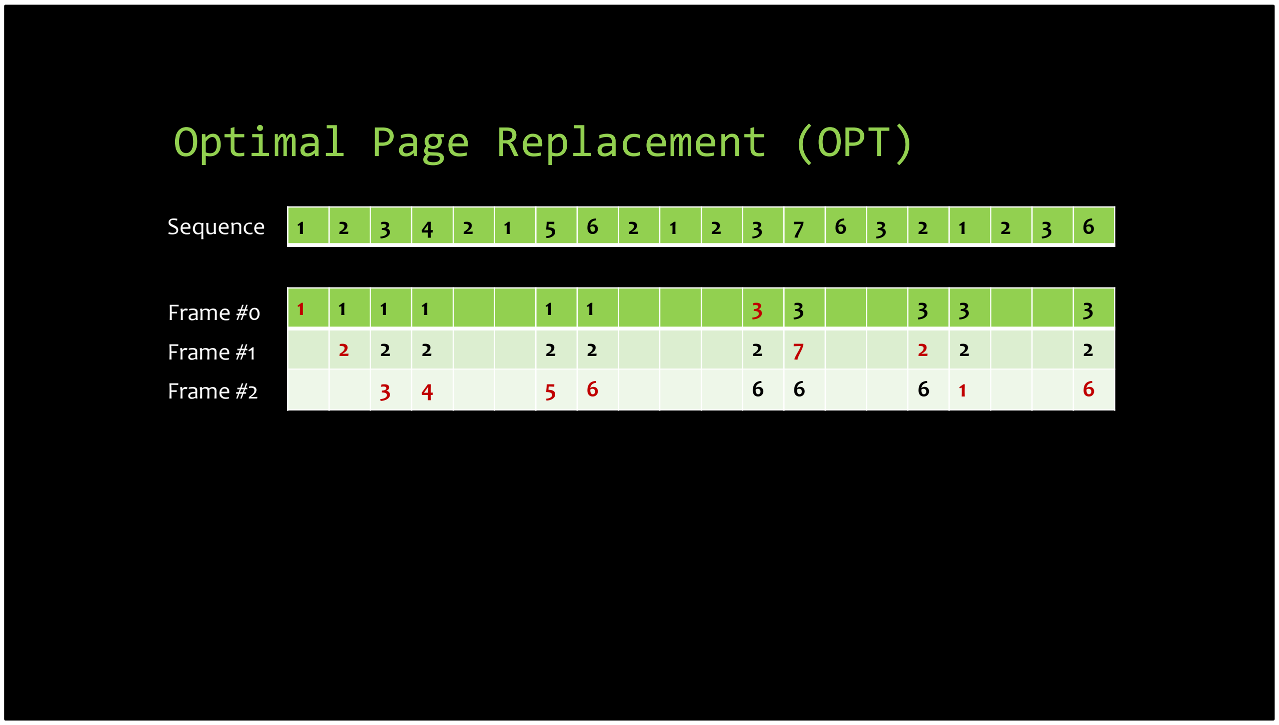
- Size of string 20, total page faults 16 (high over head cost)
- FIFO evicts the first block accessed first without any regard to how often or how many times it was accessed before.



Optimal Page Replacement (OPT)

- The goal of OPT is to replace the page that will not be used for the longest period of time
- Since it is generally impossible to predict how far in the future information will be needed, this is usually not implementable in practice
- OPT is mainly used for comparison studies

For example, an algorithm is good because it is 12% of optimal at worst and within 5% on average

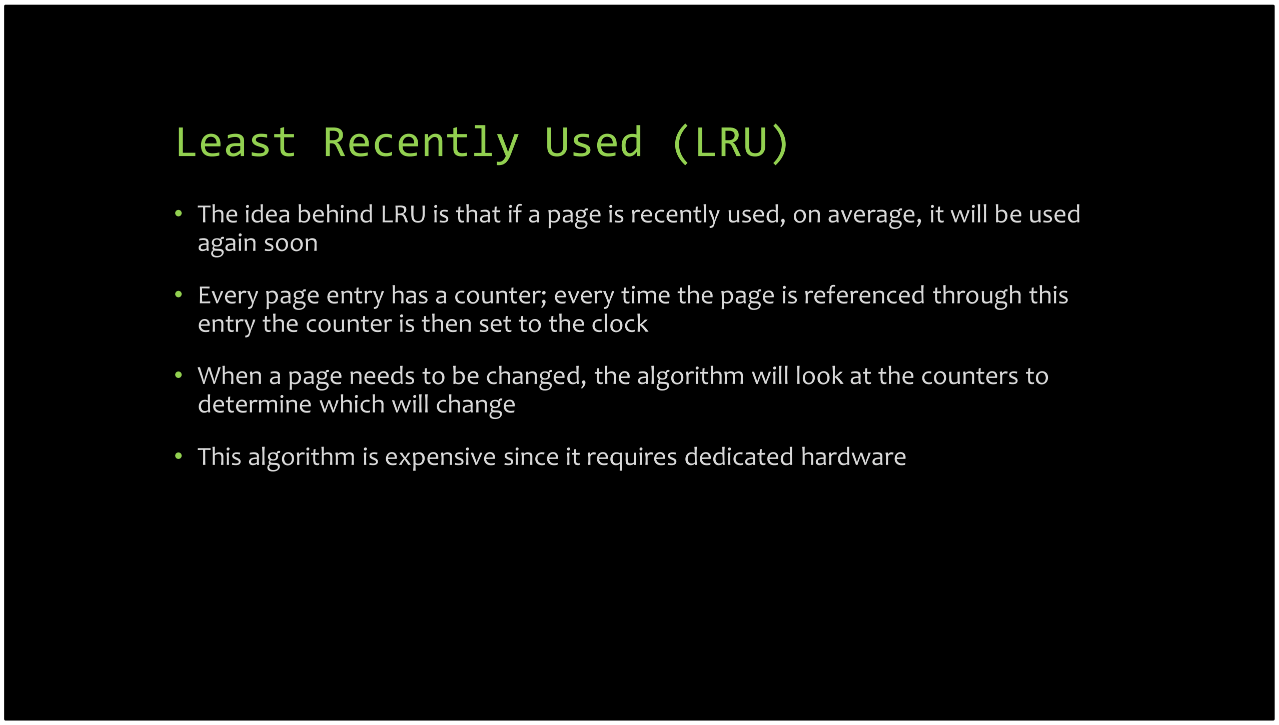


Optimal Page Replacement (OPT)

Sequence	1	2	3	4	2	1	5	6	2	1	2	3	7	6	3	2	1	2	3	6
Frame #0	1	1	1	1			1	1				3	3			3	3			3
Frame #1		2	2	2			2	2				2	7			2	2			2
Frame #2			3	4			5	6				6	6			6	1			6

- OPT look ahead to see which string reference will be used last. With the last column since there are no more references it uses FIFO. String size is 20, total page faults 11.

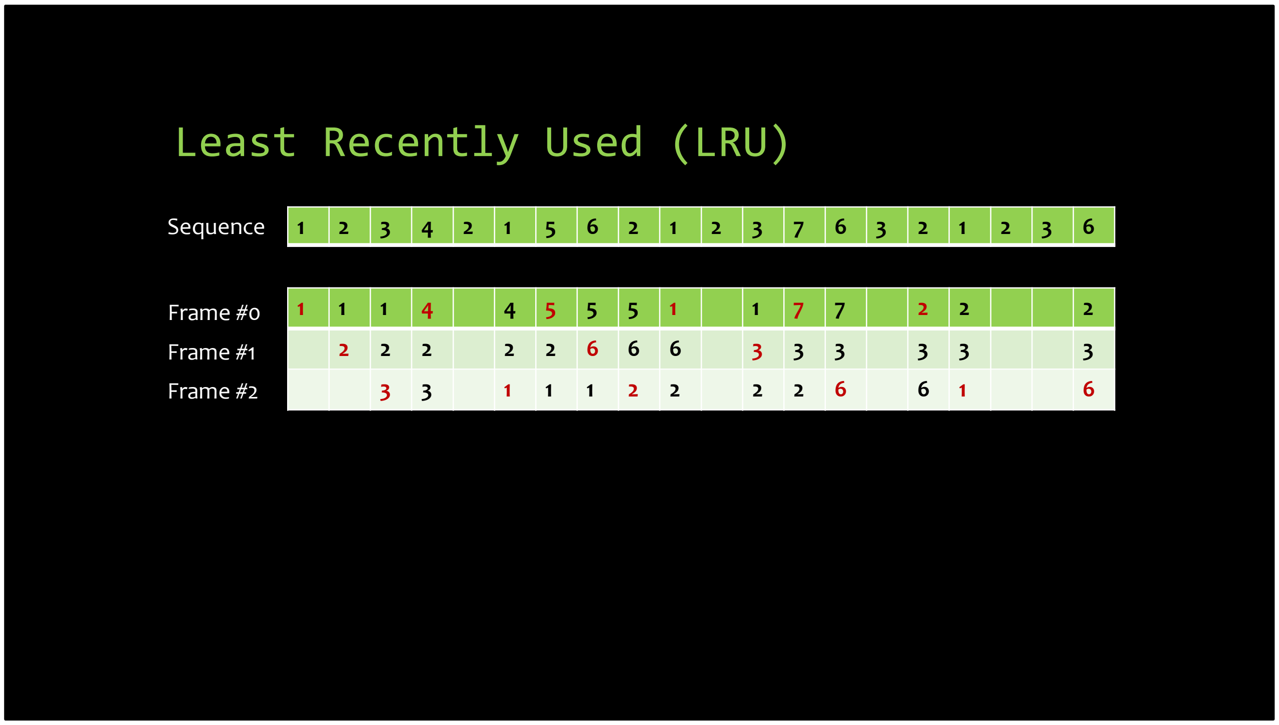
- Because real-life general purpose operating systems cannot actually predict when the next reference will be accessed, OPT cannot be implemented on such a system.



Least Recently Used (LRU)

- The idea behind LRU is that if a page is recently used, on average, it will be used again soon
- Every page entry has a counter; every time the page is referenced through this entry the counter is then set to the clock
- When a page needs to be changed, the algorithm will look at the counters to determine which will change
- This algorithm is expensive since it requires dedicated hardware

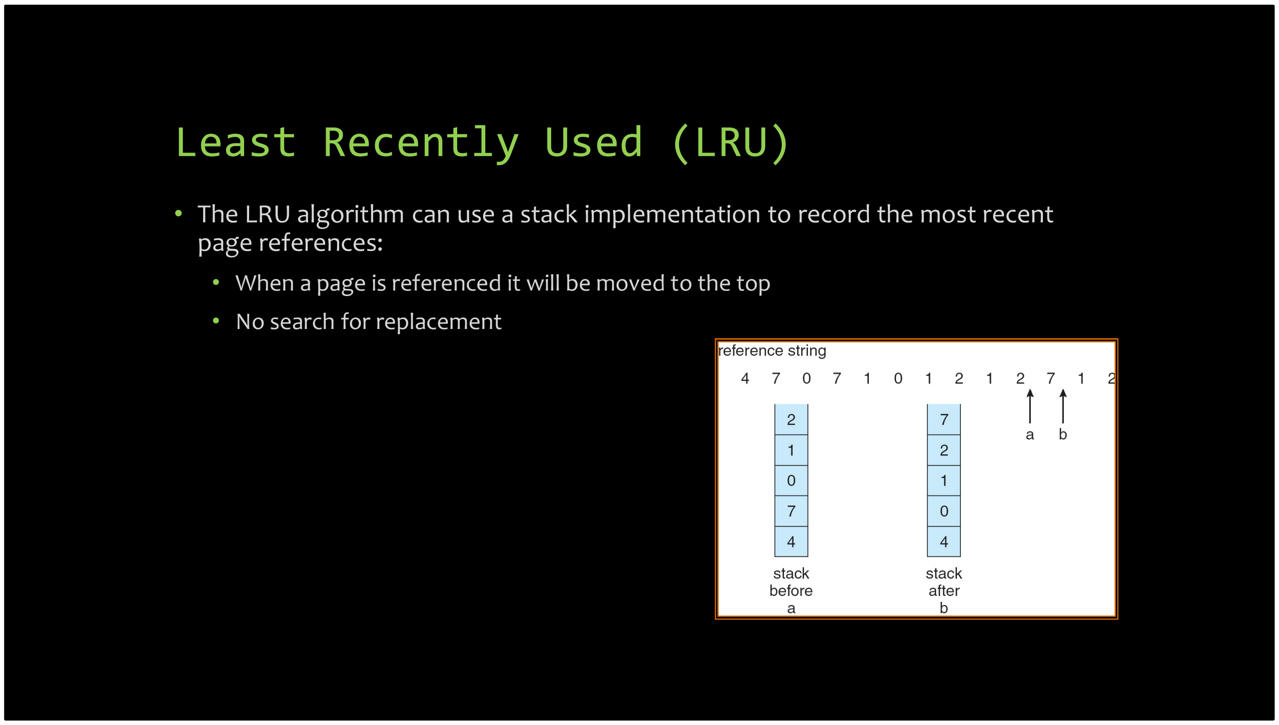
Count – time of last use



Least Recently Used (LRU)

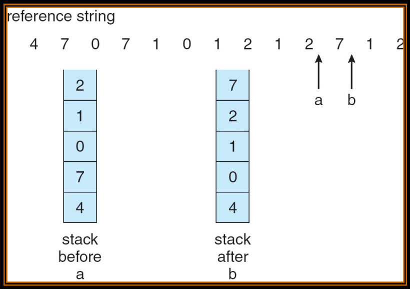
Sequence	1	2	3	4	2	1	5	6	2	1	2	3	7	6	3	2	1	2	3	6
Frame #0	1	1	1	4		4	5	5	5	1		1	7	7		2	2			2
Frame #1		2	2	2		2	2	6	6	6		3	3	3		3	3			3
Frame #2			3	3		1	1	1	2	2		2	2	6		6	1			6

- Size of string 20, total page faults 15.
- The counter is incremented after each instruction and stored into the page entry at each reference. Store the value of the counter in each entry of the page table (last access time to the page). When it is time to remove a page, find the lowest counter value.

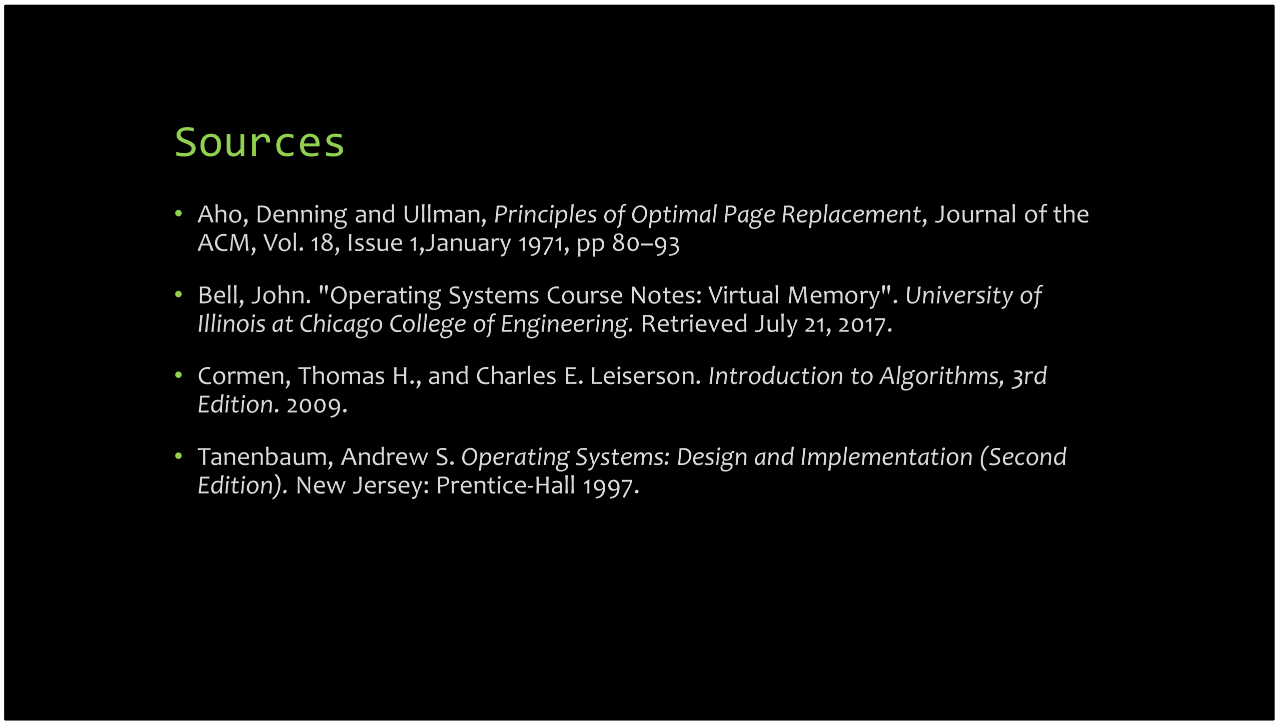


Least Recently Used (LRU)

- The LRU algorithm can use a stack implementation to record the most recent page references:
  - When a page is referenced it will be moved to the top
  - No search for replacement

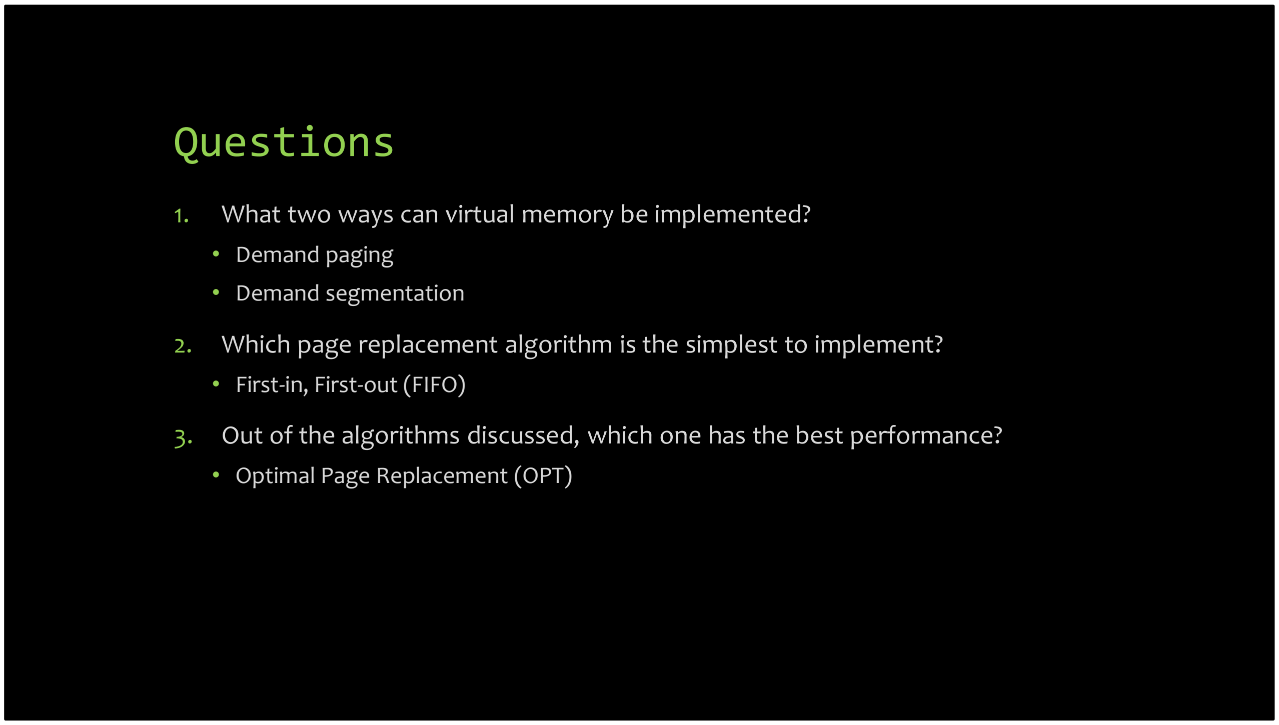


- Keeps a stack of page numbers in double link form



Sources

- Aho, Denning and Ullman, *Principles of Optimal Page Replacement*, Journal of the ACM, Vol. 18, Issue 1,January 1971, pp 80–93
- Bell, John. "Operating Systems Course Notes: Virtual Memory". *University of Illinois at Chicago College of Engineering.* Retrieved July 21, 2017.
- Cormen, Thomas H., and Charles E. Leiserson. *Introduction to Algorithms, 3rd Edition*. 2009.
- Tanenbaum, Andrew S. *Operating Systems: Design and Implementation (Second Edition).* New Jersey: Prentice-Hall 1997.



Questions

1. What two ways can virtual memory be implemented?
   - Demand paging
   - Demand segmentation
2. Which page replacement algorithm is the simplest to implement?
   - First-in, First-out (FIFO)
3. Out of the algorithms discussed, which one has the best performance?
   - Optimal Page Replacement (OPT)

Thank you!