CSCE 410/611 Operating Systems Spring 2023

Homework for Week 4

(Due Date: Check Canvas)

1. [1] Why are segmentation and paging sometimes combined into one scheme?

They are typically combined to minimize the overhead cost of large page tables by breaking down the segments and adding a single segment entry into the segment table. This gives the benefit of a large page table while saving memory using the segment table.

2. [1] Compare the segmented paging scheme with the hashed page table scheme for handling large address spaces. Under what circumstances is one scheme preferable to the other?

The segmented page table scheme allows for a large page table with the reference bits to the page table stored in the segment table. The hash page scheme uses a hash function to establish and check the reference bits. A hash scheme is very useful for large address spaces, the segmented page table is adept if there is low memory usage.

3. [1] Assume that a program has just referenced an address in virtual memory. For each of the following cases, describe a scenario. If no matching scenario exists, explain why.

(a) TLB miss with no page fault.

TLB reference to the page as already been clear and the page is now in memory

(b) TLB miss and page fault.

TLB reference to the page is not there, and is not in memory

(c) TLB hit and no page fault.

TLB has the page reference and the page is in memory, this is likely an already accessed page.

(d) TLB hit and page fault.

This is not possible because for a page to be referenced in the TLB it must be in memory as the TLB is like the cache of the page table.

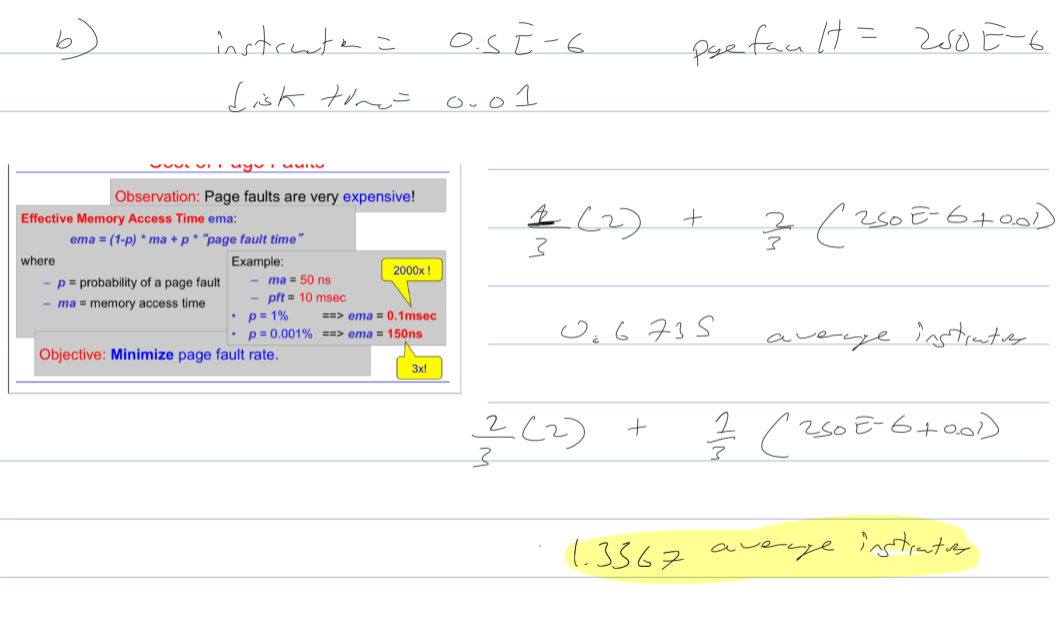
4. Suppose an instruction takes 1/2 microsecond to execute (on the average), and a page fault takes 250 microseconds of processor time to handle plus 10 milliseconds of disk time to read in the page.

(a) How many pages a second can the disk transfer?

One disc transfer takes 0.01 seconds so in one second 100 disk transfers can occur.

(b) Suppose that 1/3 of the pages are dirty. It takes two page transfers to replace a dirty page. Compute the average number of instructions between page fault that would cause the system to saturate the disk with page traffic, that is, for the disk to be busy all the time doing page transfers.

1.3367 average instructions



5. [4] A two-dimensional 512 x 512 array is stored *in row major order* in a paged virtual memory system with page size 512. Row-major order means the array is laid out in memory row after row, with the first row first. All elements of the array must be accessed sequentially. Initially, the entire array is on disk. There are 16 frames of physical memory available for processing the array.

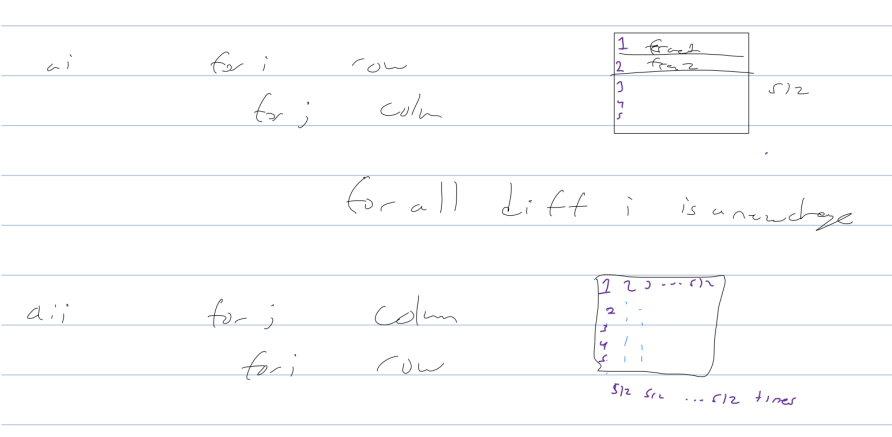
(a) For the FIFO page replacement discipline, how many page faults will be generated when accessing all elements of the array sequentially:

i. in row major order (i.e., row after row)

512

ii. in column major order (i.e., column after column)

512\*512 = 262144



(b) Would the number of page faults be different under the LRU policy?

No, as the LRU has the same access array elements that occurred in the previous iterations. This means that there will not be any differences for the FIFO.

6. [4] Assuming a physical memory of four page frames, give the number of page faults, the number of page read operations from disk, and the number of page write operations to disk, for the following reference string. Assume that all references to Page a are WRITE operations. (As a reminder of that, we mark references to Page a with an asterisk.)

a*∗* b g a*∗* d e a*∗* b a*∗* d e g d e

1

Repeat the assignment for each of the following policies. (Initially, all frames are empty.)

(a) Optimal

Page Fault = 6 Page Write = 1

(b) FIFO

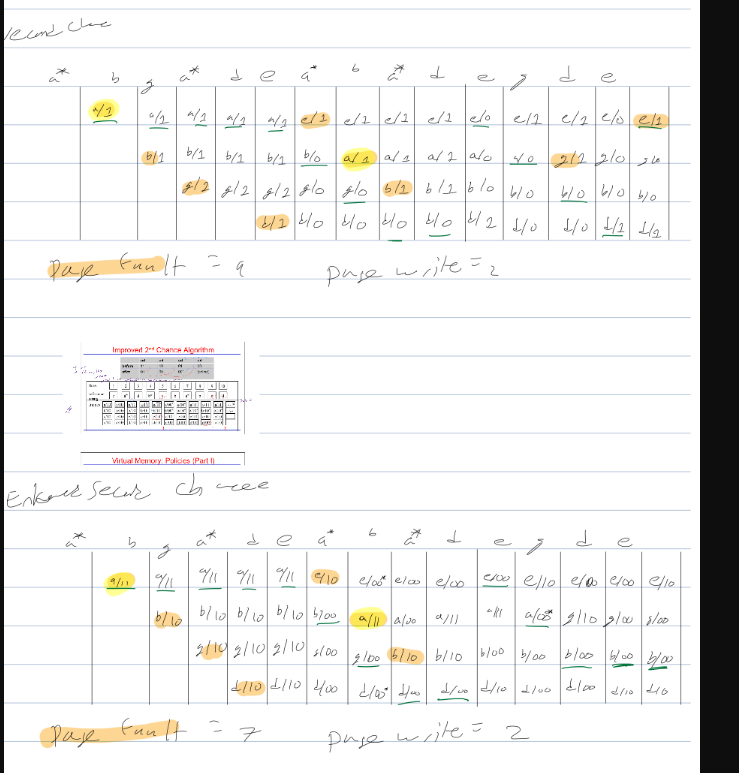
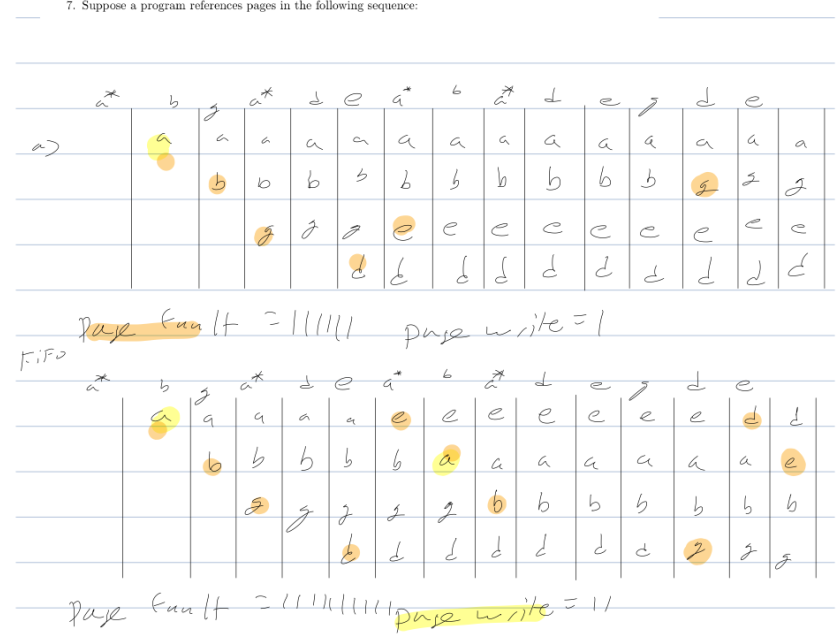
Page Fault = 10 Page Write = 2

(c) Second-chance algorithm

Page Fault = 9 Page Write = 2

(d) Enhanced second-chance algorithm

Page Fault = 7 Page Write = 2



7. Suppose a program references pages in the following sequence:

A C B\* D B\* A E F B\* F A G E F A

(Write references to pages are marked with an asterisk.) Assume the physical memory has four frames. For the following replacement algorithms, show how they would fault pages into the four frames of physical memory.

(a) LRU

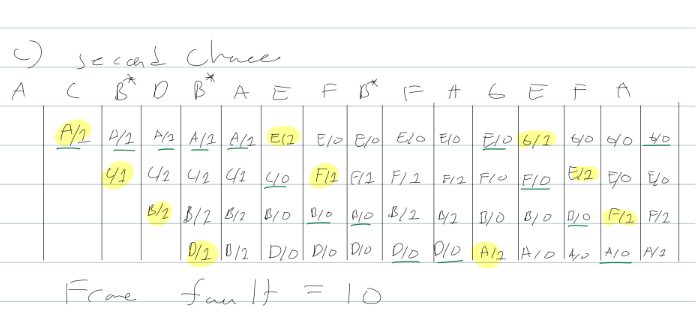
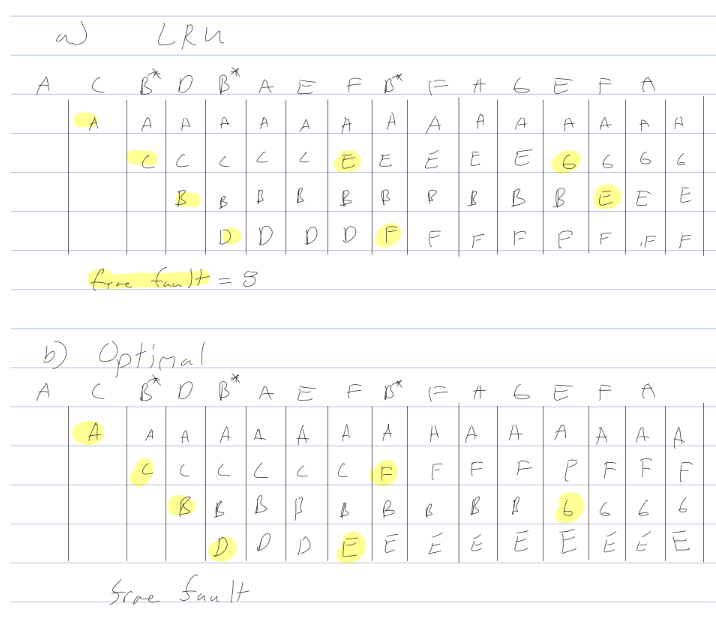
Page fault = 8

(b) Optimal

Page fault = 7

(c) Second-Chance Algorithm

Page fault = 10



8. Assume that you have a page reference string for a process. Let the page reference string have length *p* and *n* distinct page numbers occurring in it. Let *m* be the number of page frames that are allocated to the process (all the page frames are initially empty). Let *n > m*.

(a) What is the lower bound on the number of page faults?

There will be n compulsory misses as the table fills

(b) What is the upper bound on the number of page faults?

At the worst case there will be a fault for each page in the frame thus m.

Hint: Your answer should be independent of the page replacement scheme that you use.

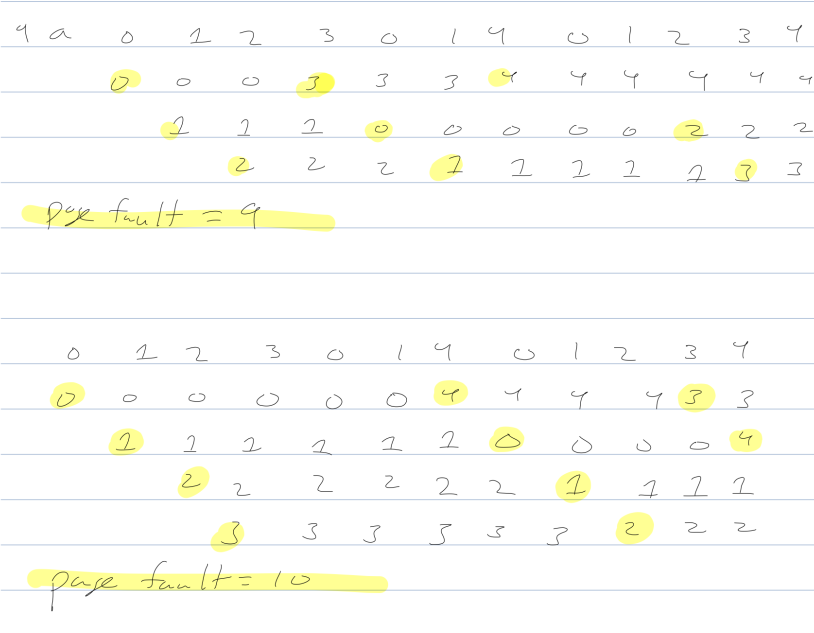
9. Belady’s anomaly: Intuitively, it seems that the more frames the memory has, the fewer page faults a program will get. This is not always the case, however. In 1969, Belady discovered an example in which FIFO page replacement causes more faults with four page frames than with three. This strange situation has become known as *Belady’s Anomaly*. To illustrate, a program with five pages numbered from 0 to 4 references it s pages in the order:

0 1 2 3 0 1 4 0 1 2 3 4

(a) Using FIFO, compute the number of page faults with 3 frames; repeat for 4 frames.

Fifo 3 frames, page faults = 9

Fifo 4 frames, page faults = 10



(b) Compute the number of page faults under LRU, the second-chance algorithm, and the optimal algorithm. Do these algorithms suffer from Belady’s anomaly for the same scenario?

LRU 3 frames page faults = 10

LRU 4 frames page faults = 10

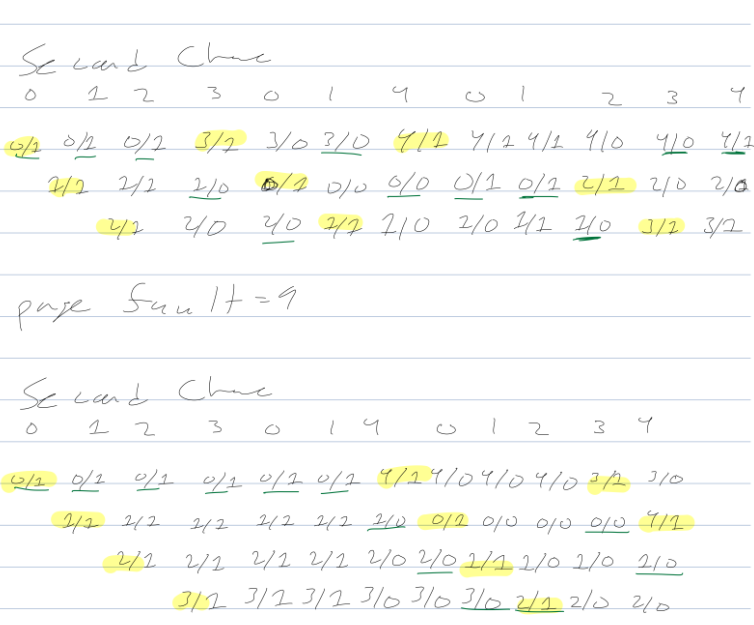
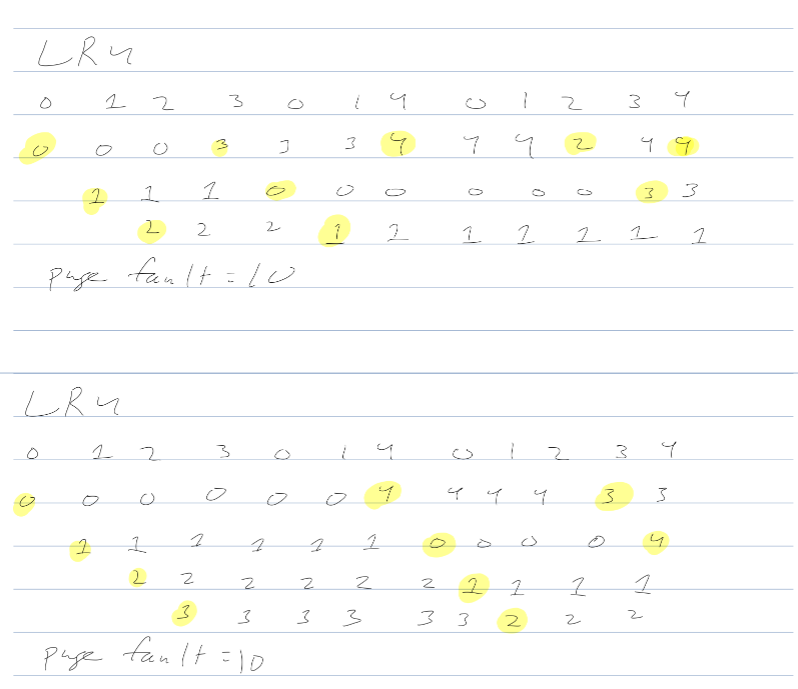
Second change 3 frames page faults = 9

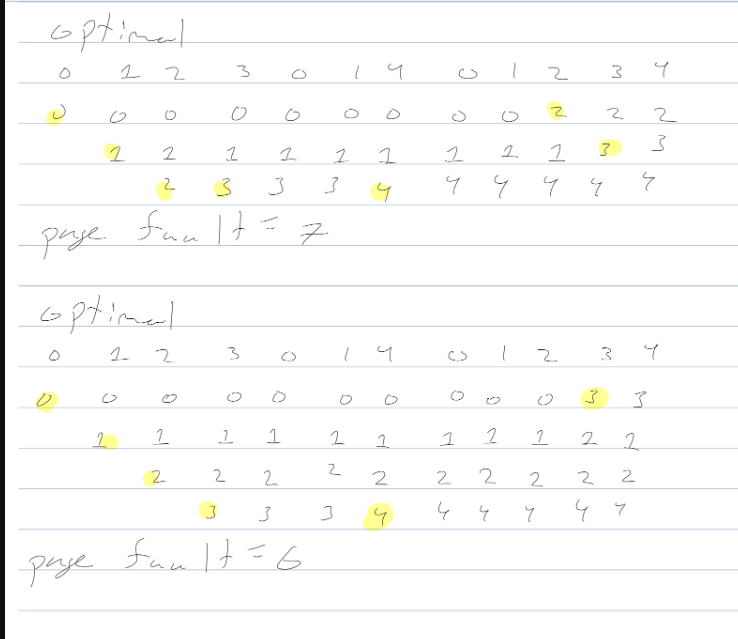
Second change 4 frames page faults = 10

Optimal 3 frames page faults = 7

Optimal 4 frames page faults = 6

Belady’s anomaly only holds for FIFO.





References

[1] A. Silberschatz, P. Galvin, and G. Gagne, *Applied Operating Systems Concepts*, John Wiley & Sons, Inc., New York, NY, 2000.

[2] Deitel, Deitel, and Choffnes, *Operating Systems*, Pearson / Prentice Hall, 2004. 2

[3] A. S. Tanenbaum, *Modern Operating Systems*, Pearson / Prentice Hall, 2008. [4] L. F. Bic, A. C. Shaw, *Operating Systems Principles*, Prentice Hall 2003. [5] C. Crowley, *Operating Systems, A Design-Oriented Approach*, Irwin 1997. [6] M. Herlihy, N. Shavit, *The Art of Multiprocessor Programming*, Elsevier, 2008

3