Homework for Week 4
(Due Date: Check Canvas)

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

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?

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.

   (b) TLB miss and page fault.

   (c) TLB hit and no page fault.

   (d) TLB hit and page fault.

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?

   (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.

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)

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

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

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`

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

(a) Optimal

(b) FIFO

(c) Second-chance algorithm

(d) Enhanced second-chance algorithm

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

(b) Optimal

(c) Second-Chance Algorithm

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?

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

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.

(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?

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.

[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