Caleb Faruki
Dr. Art Werschulz
CISC3595 Operating Systems
November 20, 2013

Homework 8

8.16

a. Higher page-fault rate.
b. Lower page-fault rate.
c. Swapping and scheduling. Free frames for new pages by replacing pages from memory.

8.20 (the numbers given in the problem description are in decimal

The system establishes the corresponding physical location of the process by first translating the virtual address into binary:

11123456 → 0001 0001 0001 0010 0011 0100 0101 0110

The page size is 12 bits (4kB → 2^12) and the page table size is 20 bits. The first 12 bits in the virtual address are used as the displacement in the page while the other 20 bits of the virtual address are used as the displacement in the page table. The offset bits are then concatenated to the resulting page number to create the corresponding physical address.

8.21

$$Where\ P = page - fault\ rate,$$

$$0.2\mu s = [(1-P)\cdot 0.1\,\mu s] + (0.3P\cdot 8000\mu s) + (0.7P * 20000\mu s)$$
$$0.1 = -0.1P + 2400P + 14000P$$
$$0.1 \quad 16399.9P$$
$$P = \boldsymbol{approx.}\ \mathbf{0.000006}$$

8.23

9EF → 0EF
111 → 211
700 → D00
0FF → EFF

8.25

The LFU page-replacement algorithm replaces a page if it's not being used frequently. In some cases, a page can be used frequently in the beginning and not at all toward the end. That page will retain its high usage count even though it's barely used. In situations where this happens often, LRU may be more desirable for reducing page faults.

Conversely, LFU might be more desirable in a scenario where the least frequently used pages are the most recently used pages. Pages are requested in an evenly distributed manner, meaning that

8.26

MFU is desirable when a page is never accessed twice in a row. It assumes that the most frequent page won't be requested again for some time.

LRU is desirable when older pages are accessed least. This would occur when newer pages are used repeatedly, leaving older pages to be ignored and eventually swapped out. MFU in this scenario would cause many more page faults because it assumes naively that the most recent page won't be accessed for a long time.

8.28

a. No. In fact, it will reduce CPU utilization since the same amount of work will be accomplished by the CPU faster.

b. A bigger paging disk may not improve CPU utilization. We don't know if the old paging disk has a high usage resulting from too much data or from slow operation.
c. No. The paging disk is already busy. This may reduce CPU utilization since more processes are competing for the paging disk.
d. Yes. It will reduce thrashing and optimize CPU utilization.
e. Yes. More memory will reduce paging activity unless we increase the degree of multiprogramming.
f. Maybe not. I/O devices aren't busy in the initial setup, which indicates that they are not the cause of the bottleneck.
g. Yes. This will reduce page faults and, therefore, increase CPU utilization.
h. No. Increasing page size will exacerbate thrashing by consuming more memory.

8.31 (show work for b & c)

a. i. Initial counter value: 0
   ii. Counters are increased whenever a new page is associated with that frame.
   iii. Whenever a page associated with that frame is no longer required, counters are decreased.
   iv. Find a frame with the smallest counter. FIFO as a tie-breaker.
b. 13 page faults. See 1st graph in attached photo.
c. 11 page faults. See 2nd graph in attached photo.

8.32

$$\text{Where } E = \text{effective memory access time},$$

$$E = (0.8 * 1\mu s) + (0.18 * 2\mu s) + (0.02 * 20002\mu s)$$
$$E = 0.8\mu s + 0.36\mu s + 400.04\mu s$$
$$E = \mathbf{401.2\mu s}$$

8.35

When Δ is small, then the set of resident pages for a process might be underestimated, allowing a process to be scheduled even though all of its required pages are not resident. This could increase page faults.

When Δ is large, then the set of resident pages for a process is overestimated. But once a process is scheduled, it probably won't cause page faults since the resident set is overestimated.