# CSC 415 OPERATING SYSTEM PRINCIPLES

# Homework 2

1. **(15 Points) Consider the following segment table:**

| Segment | Base | Length |
|---|---|---|
| 0 | 219 | 600 |
| 1 | 2300 | 14 |
| 2 | 90 | 100 |
| 3 | 1327 | 580 |
| 4 | 1952 | 96 |

   **What are the physical addresses for the following logical addresses?**
   a. **0,430**
   b. **1,10**
   c. **2,500**
   d. **3,400**
   e. **4,112**

*a) 0,430 => 219 + 430 = 649*
*b) 1,10 => 2300 + 10 = 2310*
*c) 2,500 will trigger a trap, since the offset is greater than the length of segment 2*
*d) 3,400 => 1327 + 400 = 1727*
*e) 4,112 will trigger a trap, since the offset is greater than the length of segment 4*

2. **(10 Points) What is internal fragmentation? Give an example of when it occurs.**
   *Internal fragmentation is the existence of empty space within frames. Paging often has internal fragmentation since processes will be split across multiple frames, which tends to leave the last frame containing empty space.*

4. **(10 Points) Assuming a byte-addressed system with 24-bit logical addresses and 8 frames in the physical memory. The size of one page is 2 MB (i.e., $2^{21}$ entries in one page). Translate the following logical addresses into physical addresses using the provided page table:**

| Page | Frame |
|---|---|
| 0 | 1 |
| 1 | 4 |
| 2 | 7 |
| 3 | 3 |
| 4 | 2 |
| 5 | 5 |
| 6 | 6 |
| 7 | 0 |

   a. **0x234800**
   b. **0xBB4400**

a) *0x234800 = 0010 0011 0100 1000 0000 0000*
*001:0 0011 0100 1000 0000 0000*
*page: 1 => frame: 4*
*offset: 0 0011 0100 1000 0000 0000*
*base address = 2^21 * 4 - 1 = 2^23 - 1 = 0111 1111 1111 1111 1111 1110*
*add base to offset:*

```
    0111 1111 1111 1111 1111 1110
+   0000 0011 0100 1000 0000 0000
---------------------------------
=   1000 0011 0100 0111 1111 1110
```

*=> 0x8347FE*

b) *0xBB4400 = 1011 1011 0100 0100 0000 0000*
*101:1 1011 0100 0100 0000 0000*
*page: 5 => frame: 5*
*offset: 1 1011 0100 0100 0000 0000*
*base address = 2^21 * 5 - 1 = (2^21 * 4) + 2^21 - 1*
*= 2^23 + 2^21 - 1*

```
= 0111 1111 1111 1111 1111 1111
+ 0001 1111 1111 1111 1111 1110
---------------------------------
= 1001 1111 1111 1111 1111 1101 = base address
```

*add base to offset:*

```
    1001 1111 1111 1111 1111 1101
+   0001 1011 0100 0100 0000 0000
---------------------------------
=   1011 1011 0100 0011 1111 1101
```

*=> 0xBB43FD*

**5. (18 Points) Consider the following page reference string: 1, 3, 2, 4, 3, 5, 6, 7, 2, 3, 2, 1, 7, 6, 5, 4, 7, 2, 5, 6. How many page faults would occur for the page replacement algorithms FIFO, Optimal, and LRU, assuming one, four, and seven free frames at the beginning?**

*<u>1 free frame</u>*
*FIFO:*
*no repeated numbers, so faults = # of references in string, which => page faults = 20*
*Optimal:*
*same as FIFO since there's no repeated numbers, so page faults = 20*
*LRU:*
*same as FIFO since there's no repeated numbers, so page faults = 20*

*<u>4 free frames</u>*
*FIFO:*
*1, 1-3, 1-3-2, 1-3-2-4, skip, 5-3-2-4, 5-6-2-4, 5-6-7-4, 5-6-7-2, 3-6-7-2, skip, 3-1-7-2, skip, 3-1-6-2, 3-1-6-5, 4-1-6-5, 4-7-6-5, 4-7-2-5, skip, 4-7-2-6*
*20 - 4 skips = 16 page faults*

*Optimal:*

*1, 1-3, 1-3-2, 1-3-2-4, skip, 1-3-2-5, 1-3-2-6, 1-3-2-7, skip, skip, skip, skip, skip, 6-3-2-7, 6-5-2-7, 4-5-2-7, skip, skip, skip, 6-5-2-7*

*20 - 9 skips = 11 page faults*

*LRU:*

*1, 1-3, 1-3-2, 1-3-2-4, skip, 5-3-2-4, 5-3-6-4, 5-3-6-7, 5-2-6-7, 3-2-6-7, skip, 3-2-1-7, skip, 6-2-1-7, 6-5-1-7, 6-5-4-7, skip, 2-5-4-7, skip, 2-5-6-7*

*20 - 5 skips = 15 page faults*

*<u>7 free frames</u>*

*FIFO:*

*1, 1-3, 1-3-2, 1-3-2-4, skip, 1-3-2-4-5, 1-3-2-4-5-6, 1-3-2-4-5-6-7, skip 12x*

*20 - 13 skips = 7 page breaks*

*Optimal:*

*since no replacement occurs with 7 free frames, Optimal is same as FIFO:*

*20 - 13 skips = 7 page breaks*

*LRU:*

*since no replacement occurs with 7 free frames, LRU is same as FIFO:*

*20 - 13 skips = 7 page breaks*

6. **(14 Points) What is the cause of thrashing? Once thrashing is detected, what can the system do to eliminate this problem?**

*Thrashing happens when an active process spends more time paging than executing. usually when processes don't have enough frames during execution. EIther Working-Set or Page Fault Frequency models can be used to correct for thrashing. In Working-Set, if the demand for pages exceeds the total number of available frames, a process gets swapped out and the frames it is utilizing are released to be distributed to the remaining processes. In Page Fault Frequency, there are upper and lower bounds on page faulting for each process. In the event of thrashing, the upper bound is being exceeded, and the offending processes will be assigned additional frames until the number of page faults drops below the upper bound.*

7. **(18 Points) Suppose that a disk drive has 5,000 cylinders, numbered 0 to 4999, with the read-write head having just finished a request at cylinder 1056. The queue of pending requests, in order of arrival, is:**

**333, 1200, 922, 4545, 3786, 3605**

**Starting from the current head position, what is the order in which the requests will be serviced for each of the following disk-scheduling algorithms (assuming no other requests arrive and the head's initial movement is toward higher numbered cylinders for c-f)?**

**a. FCFS b. SSTF c. SCAN d. LOOK e. C-SCAN f. C-LOOK**

a) *FCFS: 333, 1200, 922, 4545, 3786, 3605*

*b) SSTF: 922, 1200, 333, 3605, 3786, 4545*
*c) SCAN: 1200, 3605, 3786, 4545, 4999 (reaches end and reverses), 922, 333*
*d) LOOK: 1200, 3605, 3786, 4545, 922, 333*
*e) C-SCAN: 1200, 3605, 3786, 4545, 4999 (reaches end), 0 (continues on from beginning), 333, 922*
*f) C-LOOK: 1200, 3605, 3786, 4545, 333, 922*

**8. (15 Points) Describe the procedure for handling the page fault when there is always a free frame in demand paging.**

*If a page fault occurs and there is a free frame available, then it simply reads the page into the free frame, updates the page table to indicate that the page is now in memory, then adds a new frame to the process (to keep a frame always free), then restart the instruction that caused the page fault to continue regular execution.*