CSCI 3753 **Liam Kolber**
Problem Set 3 **Spring 2018, CU-Boulder**

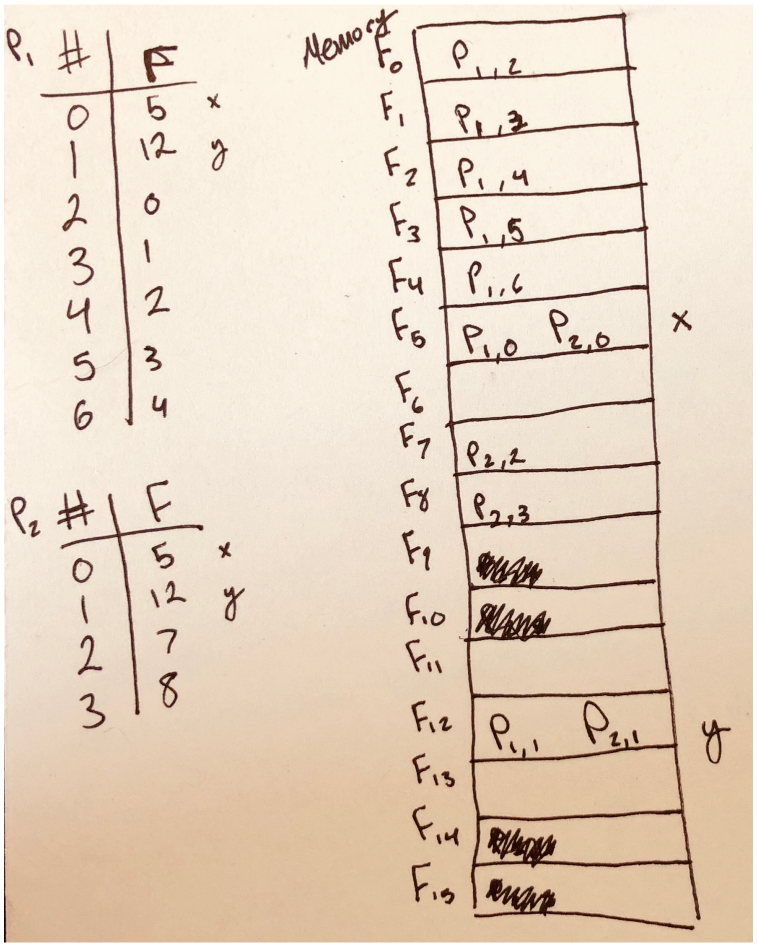
RECITATION: 104
NUMBER OF HOURS TO COMPLETE: 6 hours

1. *(20 pts) Describe at least 3 general approaches in memory management than can help solve the external fragmentation problem.:*

   - **Compaction:** This method attempts to free up contiguous space by shuffling all allocated memory towards one side of the physical memory allowing for a large block of free memory for the processes that request it. Unfortunately this isn't always a viable options as it is only possible to perform this procedure during execution time and not assmebly or load time.
   - **Segmentation:** This method works around the issue by allocating memory in chunks. The logical address space segments are no longer required to be contiguous. By doing so, the memory required for a process can be spread throughout physical memory filling in any open space that may exist rather than waiting for a large enough space for the whole process to be available.
   - **Paging:** Paging ensures that open chunks of memory have a guaranteed minimum size (since paging breaks up the physical memory into fixed sizes). With predetermined segment sizes and fixed sizes of "chunks" in physical memory due to paging, the memory required for a process/task can be easily distributed to any available slots within physical memory when combined with segmentation.

2. *(20 pts) Suppose two processes need to be mapped into main memory using pages. Process P1 consists of 7 pages, and process P2 consists of 4 pages. Assume main memory consists of 16 frames, a logical page is the same size as a physical frame, and that 4 entries in a page table fills up a frame of memory. Assume also that within the process' allocated address spaces, there are two pages of shared code X and Y common to both address spaces. Design a memory management system that can store these two processes and their page tables in RAM. Identify which frames you have chosen to assign to which process pages and page tables in main memory/RAM. Also show possible page tables for P1 and P2 (e.g. page table for P1 should have 7 entries):*

It takes 4 entries to fille a frame, but with paging, we don't need contiguous allocation. Because of this, I was able to give each page a unique frame in memory except for the four that were paired to represent X and Y.



P_1 #	F	
0	5	x
1	12	y
2	0	
3	1	
4	2	
5	3	
6	4	

P_2 #	F	
0	5	x
1	12	y
2	7	
3	8	

Memory		
F_0	$P_{1,2}$	
F_1	$P_{1,3}$	
F_2	$P_{1,4}$	
F_3	$P_{1,5}$	
F_4	$P_{1,6}$	
F_5	$P_{1,0}$ $P_{2,0}$	x
F_6		
F_7	$P_{2,2}$	
F_8	$P_{2,3}$	
F_9		
F_{10}		
F_{11}		
F_{12}	$P_{1,1}$ $P_{2,1}$	y
F_{13}		
F_{14}		
F_{15}		

3. *(20 pts) Suppose on-demand paging is employed in addition to TLB caching. The time for a TLB hit is T = 1 ns, a memory read M = 10 ns, and a disk read D = 10 ms. Let p_TLB = the probability of a TLB hit, and p = the probability of a page fault given a TLB miss. What is a general formula for the average memory access time expressed as a function of T, M, D, p, and p_TLB? Once parameter values are substituted, and assuming assuming p = .001 and p_TLB = 90%, what is the calculated average memory access time?:*

$$TLBHit + TLBMiss\&PageHit + TLBMiss\&PageFault$$

$$(T * p\_TLB) + (1 - p\_TLB)(1 - p) * M + (1 - p\_TLB)(p) * D$$

$$(1 * 0.9) + (1 - 0.9)(1 - 0.001) * 10 + (1 - 0.9)(0.001) * 10^7$$

$$1001.899[ns]$$

4. *(20 pts) The Least Recently Used (LRU) page replacement policy does not suffer from Belady's Anomaly. Explain intuitively why this is the case. Construct an example page fault sequence to illustrate your point:*

LRU is a specific type of paging algorithm known as a stacking algorithm that is ordered entirely by the most recently used elements. The first P elements in a (P+n) sized stack is the same first P elements in P sized stack. This means that the number of page faults that would occur either remains consistent or decreases with an increase in the number of frames.

FIFO

1	2	3	4	1	2	5	1	2	3	4	5
$\underline{1}$	1	1	$\underline{4}$	4	4	$\underline{5}$	5	5	5	5	5
-	$\underline{2}$	2	2	$\underline{1}$	1	1	1	1	$\underline{3}$	3	3
-	-	$\underline{3}$	3	3	$\underline{2}$	2	2	2	2	$\underline{4}$	4

Page Faults: 9

1	2	3	4	1	2	5	1	2	3	4	5
$\underline{1}$	1	1	1	1	1	$\underline{5}$	5	5	5	$\underline{4}$	4
-	$\underline{2}$	2	2	2	2	2	$\underline{1}$	1	1	1	$\underline{5}$
-	-	$\underline{3}$	3	3	3	3	3	$\underline{2}$	2	2	2
-	-	-	$\underline{4}$	4	4	4	4	4	$\underline{3}$	3	3

Page Faults: 10 ← **Anomaly**

LRU

1	2	3	4	1	2	5	1	2	3	4	5
$\underline{1}$	1	1	$\underline{4}$	4	4	$\underline{5}$	5	5	$\underline{3}$	3	3
-	$\underline{2}$	2	2	$\underline{1}$	1	1	1	1	1	$\underline{4}$	-
-	-	$\underline{3}$	3	3	$\underline{2}$	2	2	2	2	2	$\underline{5}$

Page Faults: 10

1	2	3	4	1	2	5	1	2	3	4	5
$\underline{1}$	1	1	1	1	1	1	1	1	1	1	$\underline{5}$
-	$\underline{2}$	2	2	2	2	2	2	2	2	2	2
-	-	$\underline{3}$	3	3	3	$\underline{5}$	5	5	5	$\underline{4}$	4
-	-	-	$\underline{4}$	4	4	4	4	4	$\underline{3}$	3	3

Page Faults: 8 ← **No Anomaly**

5. *(20 pts) Given a frame allocation of 3, and the following sequence of page references*

3 2 4 3 4 2 2 3 4 5 6 7 7 6 5 4 5 6 7 2 1

and assuming main memory is initially unloaded, show the page faulting behavior using the following page replacement policies. How many page faults are generated by each page replacement algorithm? Which generates the fewest page faults?

(a) FIFO

3	2	4	3	4	2	2	3	4	5	6	7	7	6	5	4	5	6	7	2	1
$\underline{3}$	3	3	3	3	3	3	3	3	$\underline{5}$	5	5	5	5	5	$\underline{4}$	4	4	$\underline{7}$	7	7
-	$\underline{2}$	2	2	2	2	2	2	2	2	$\underline{6}$	6	6	6	6	6	$\underline{5}$	5	5	$\underline{2}$	2
-	-	$\underline{4}$	4	4	4	4	4	4	4	4	$\underline{7}$	7	7	7	7	7	$\underline{6}$	6	6	$\underline{1}$

12 page faults

(b) OPT

3	2	4	3	4	2	2	3	4	5	6	7	7	6	5	4	5	6	7	2	1
$\underline{3}$	3	3	3	3	3	3	3	3	$\underline{5}$	5	5	5	5	5	5	5	5	$\underline{7}$	7	7
-	$\underline{2}$	2	2	2	2	2	2	2	2	$\underline{6}$	6	6	6	6	6	6	6	6	$\underline{2}$	2
-	-	$\underline{4}$	4	4	4	4	4	4	4	4	$\underline{7}$	7	7	7	$\underline{4}$	4	4	4	4	$\underline{1}$

10 page faults

(c) LRU

3	2	4	3	4	2	2	3	4	5	6	7	7	6	5	4	5	6	7	2	1
$\underline{3}$	3	3	3	3	3	3	3	3	3	$\underline{6}$	6	6	6	6	6	6	6	6	6	$\underline{1}$
-	$\underline{2}$	2	2	2	2	2	2	2	$\underline{5}$	5	5	5	5	5	5	5	5	5	$\underline{2}$	2
-	-	$\underline{4}$	4	4	4	4	4	4	4	4	$\underline{7}$	7	7	7	$\underline{4}$	4	4	$\underline{7}$	7	7

10 page faults

Both OPT and LRU produce the fewst page faults (10).

6. *(Extra Credit: 5 pts) A memory manager for a variable-sized region strategy has a free list of blocks of size 600, 400, 1000, 2200, 1600, and 1050 bytes. What block will be selected to honor a request for:*

   Best-Fit picks the optimal block size.

   (a) 1603 bytes using a best-fit policy? **2200**

   (b) 949 bytes using a best-fit policy? **1000**

   Worst-Fit picks the the least optimal (i.e largest) block size.

   (c) 1603 bytes using a worst-fit policy? **2200**

   (d) 349 bytes using a worst-fit policy? **2200**

   First-Fit picks the first block size in list that would fit.

   (e) 1603 bytes using a first-fit policy? **2200**

   (f) 1049 bytes uusing a first-fit policy? **2200**

7. *(Extra Credit: 15 pts) Assume the same sequence of page references as in problem #6, and assume memory is initially unloaded, but now assume that a dynamic paging working-set algorithm is applied to the same sequence of page references, with a window size of 6. Draw the page faulting behavior. Your solution chart should show the frame allocation at any given time to the process.*

Working sets at each process (starting from 6th given the window size). Underlined are the windows and current working sets are in braces.

$3\ 2\ 4\ 3\ 4\ \underline{2\ 2\ 3\ 4\ 5\ 6\ 7\ 7\ 6\ 5\ 4\ 5\ 6\ 7\ 2\ 1}$

2: $\underline{3\ 2\ 4\ 3\ 4\ 2}$ $\{2\ 3\ 4\}$
2: $\underline{2\ 4\ 3\ 4\ 2\ 2}$ $\{2\ 3\ 4\}$
3: $\underline{4\ 3\ 4\ 2\ 2\ 3}$ $\{2\ 3\ 4\}$
4: $\underline{3\ 4\ 2\ 2\ 3\ 4}$ $\{2\ 3\ 4\}$
5: $\underline{4\ 2\ 2\ 3\ 4\ 5}$ $\{2\ 3\ 4\ 5\}$
6: $\underline{2\ 2\ 3\ 4\ 5\ 6}$ $\{2\ 3\ 4\ 5\ 6\}$
7: $\underline{2\ 3\ 4\ 5\ 6\ 7}$ $\{2\ 3\ 4\ 5\ 6\ 7\}$
7: $\underline{3\ 4\ 5\ 6\ 7\ 7}$ $\{3\ 4\ 5\ 6\ 7\}$

6: $\underline{4\ 5\ 6\ 7\ 7\ 6}$ $\{4\ 5\ 6\ 7\}$
5: $\underline{5\ 6\ 7\ 7\ 6\ 5}$ $\{5\ 6\ 7\}$
4: $\underline{6\ 7\ 7\ 6\ 5\ 4}$ $\{4\ 5\ 6\ 7\}$
5: $\underline{7\ 7\ 6\ 5\ 4\ 5}$ $\{4\ 5\ 6\ 7\}$
6: $\underline{7\ 6\ 5\ 4\ 5\ 6}$ $\{4\ 5\ 6\ 7\}$
7: $\underline{6\ 5\ 4\ 5\ 6\ 7}$ $\{4\ 5\ 6\ 7\}$
2: $\underline{5\ 4\ 5\ 6\ 7\ 2}$ $\{4\ 5\ 6\ 7\ 2\}$
1: $\underline{4\ 5\ 6\ 7\ 2\ 1}$ $\{4\ 5\ 6\ 7\ 2\ 1\}$