CSCI3753: PS4 Ahmed Almutawa

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

   Compaction - with full compaction, we move all the process to one part of memory which allows us to release a big segment of memory. With partial compaction, we move allocated memory and free up their memory segments until we're able to deal with memory requests.

   Garbage Collection – collects all inaccessible and unused allocated memory and treats it as if it was free.

   Paging – Divides up memory into fixed size pages then selectively allocates pages to frames. It manages those pages in memory with pointers.

2. **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:**

   a. 1603 bytes using a best-fit policy?
   2200

   b. 949 bytes using a best-fit policy?
   1000

   c. 1603 bytes using a worst-fit policy?
   2200

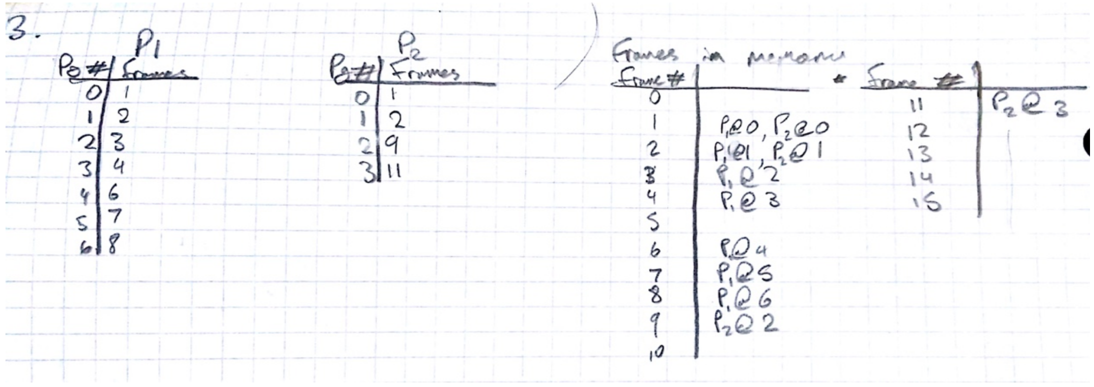
   d. 349 bytes using a worst-fit policy?
   2200

   e. 1603 bytes using a first-fit policy? (assume the free list is ordered as listed above)
   2200

   f. 1049 using a first-fit policy?
   2200

3. **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).**

   P1 : 7 pages, P2 : 4 pages, 16 Frames, 2 shared pages x and y.
   I chose to assign x and y to frames 1 and 2 respectively, the rest can be deduced from the following chart.



3.

P_1

Pg #	Frames
0	1
1	2
2	3
3	4
4	6
5	7
6	8

P_2

Pg #	Frames
0	1
1	2
2	9
3	11

Frames in memory

Frame #	
0	
1	P_1@0, P_2@0
2	P_1@1, P_2@1
3	P_1@2
4	P_1@3
5	
6	P_1@4
7	P_1@5
8	P_1@6
9	P_2@2
10	

Frame #	
11	P_2@3
12	
13	
14	
15	

4. 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 p = .001 and p_TLB = 90%, what is the calculated average memory access time?

T = 1 ns, M = 10 ns, D = 10ms, p=.001, P_TLB = 0.9

Avg. memory access time = $T\ *\ P\_TLB\ +\ (1-P\_TLB)(1-P)*M\ +\ (1-P\_TLB)*P*D$

So, given our parameters:

$$= 1*.9+(1-0.9)(1-.001)*10+(1-0.9)*.001*10000000 = 1001.899\ ns$$

5. 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.

Frequency is a better indicator for page loading compared to age. FIFO assumes the oldest pages are safest to replace, when in reality they're usually the one's used most often. LRU dispenses with the least recently used items, rather than the oldest. Below is an example of such a page fault sequence.

1	2	3	2	1	5	2	1	6
1	1	1	1	3	2	1	5	2
	2	2	3	2	1	5	2	1
		3	2	1	5	2	1	6

Red: Page Faults

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

12 page faults (in red)

b. OPT

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

10 page faults (in red)

c. LRU

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

12 page faults (in red)

7. **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.**

W(t1, z)								W(t3, z)												
3	2	4	3	4	2	2	3	4	5	6	7	7	6	5	4	5	6	7	2	1
				W(t2, z)											W(t4, z)					

W(t1, z) = {2, 3, 4}
W(t2, z) = {2, 3, 4, 5}
W(t3, z) = {4, 5, 6, 7}
W(t4, z) = {1, 2, 4, 5, 6, 7}