Exercises from old exams

Memory Management 1 (exercises on ch 10)

1. *Q.* Let's consider the following string of memory references, for a process with a virtual address space of 1000 words: 261, 409, 985, 311, 584, 746, 632, 323, 470, 915, 858.
   Compute the string of page references, under the assumption that a page has size 200 words. Use a ***second-chance*** page replacement algorithm, with a limit of 3 available frames. Assume that the reference string is an intermediate one (the program has already started, so it is in the middle of an execution): the 3 frames are already allocated to pages n. 4, 3, and 1, and the FIFO queue contains (head first) $4_0,3_1,1_0$ (subscript represents the reference bit). Represent the resident set after each reference, and compute which and how many page faults (accesses to pages not in the resident set) are generated by the string. Assume that the reference bit of a page is initialized to 0 after a page fault.

Here the page is a multiple of ten, in order to identify page number and displacement you simply have to split. Division by power a power of ten would mean splitting digits. 261 is page 1. 0 to 199 is page 0. 200 to 399 is page 1, 400 to 599 is page 2.

A:

The reference string is: **1,2,4,1,2,3,3,1,2,4,4**
(Beware, page numbering starts at 0, not 1: ex. 261/200 = 1)

	1	*2*	*4*	*1*	*2*	*3*	*3*	*1*	*2*	*4*	*4*
4₀	***4₀***	2_0	4_0	4_0	4_0	3_0	3_1	3_1	3_1	3_0	3_0
3_1	3_1	***3₁***	***3₀***	***3₀***	2_0	***2₀***	***2₀***	***2₀***	***2₁***	4_0	4_1
1_0	1_1	1_1	1_0	1_1	***1₁***	1_0	1_0	1_1	1_1	***1₀***	***1₀***
		PF	*PF*		*PF*	*PF*				*PF*	

We have to assume page 4 is one frame, page 3 in 1 frame.

A total number of 5 Page Faults are generated.

any time we have a ref to a page which is not in the frame is a page fault

Page 1 is not a fault cuz 1 already in memory, p2 is a page fault cuz not in memory. Let's look for victim head and takil is 4 ref bit is 0 so it is the victim. Simultaneously head and tail! Basically we replace 4 with 2. So the next added tail is 3. We need another victim which in page three ref bit is clear so becomes zero. page 1 is next candidate ref bit 1 which is we clear, move to page 2 whihc has ref bit 0 so that is victim and we repace 2 and move the next fifo head on 3(0).
So fifo head is 3 ref is zero it means this is the victim and replace the 3 and move fifo head to 1(1)

2. *(5/7/2019)* not necessary to know if r3f5 is read or write

   Let's consider the following string of memory references, for a given process. For each reference (Byte addressing, with addresses expressed in hexadecimal code) the read(R)/write(W) operation is also reported: **R 3F5**, R 364, **W 4D3**, W 47E, R 4C8, W 2D1, R 465, W 2A0, **R 3BA**, W 4E6, R 480, R 294, R 0B8, R 14E.
   Assume that physical and logical addresses are on 12 bits, page size is 128 Bytes, and C10 is the maximum address usable by the program (the address space top limit). this means process is not exploiting fully the 12 bits. It is given as input that maximum possible address is c10.

   Q. Compute the size of the address space (expressed as number of pages), and the internal fragmentation.

   In order to understand how many pages we need to c10 is located in which page. C =12 (8+4) 1 = 0001 and

   A. *Total number of pages in the address space (including the ones not in the reference scring): C10 = 1100 0, 001 0000. The maximum page index is 2*C (hex) = 2*12(dec) = 24 => So the address space has 25 pages. Alternative method:* $C10_H + 1 = 3089_{10}$, *NP* = $\lceil 3089/128 \rceil = 25$

The seperation between p and d is not accross two hesadecimal but incide one of the digits. In the end we can that the page where c10 is located is 11000.

   Internal fragmentation: The last page is used until the Byte at offset 0010000 (16) => Int.fr. = 128-17 = 111 Bytes. Alternative method: 128 – 3089%128 = 128-17 = 111 (Bytes)

   Q. Compute the string of page references *(we suggest going from hexadecimal to binary encoding, in order to easily identify the page number and, if necessary, the displacement/offset)*. Use an LRU (Least Recently Used) page replacement algorithm, assuming that 3 frames are available, at the following (hexadecimal) addresses: 780, A00, B00. Represent the resident set (physical frames containing logical pages) after each memory reference.
   Show page faults (references to pages outside the resident set) and compute their overall count. Finally compute the physical addresses corresponding to the (logical) references (among the previously listed ones) R 3F5, W 4D3, R 3BA

A. *Logical addresses (p,d) (page,displacement). A page contains 128 Bytes, so the displacement (d) needs 7 bits, whereas p needs 5 bits. References expressed in binary: R (0011 1,111 0101), R (0011 0,110 0011), W (0100 1,101 0011). Reference string (p is enough, d not needed): 7 (2*3+1), 6 (2*3+0), 9, 8, 9, 5, 8, 5, 7, 9, 9, 5, 1, 2 (notice that p, given by the 5 most significant bits of the logical address, can be quickly computed as the double of the first hexadecimal digit + the MSB (most significant bit) of the second hexadecimal digit).*

References		7	6	9	8	9	5	8	5	7	9	9	5	1	2
Resident Set	780	7	7	7	8		8			8	9			9	2
	A00		6	6	6		5			5	5			5	5
	B00			9	9		9			7	7			1	1
Page Faults		*	*	*	*		*			*	*			*	*

Fill the first row with memory references (represented either in hexadecimal or decimal encoding), the next three rows (representing the 3 frames in the resident set), with the logical pages mapped with the frames. The free list is initially given by: 780, A00, B00. Fill the last row with an indication on the presence of a Page Fault

Total number of Page Faults: 9

Logical addr.: R 3F5: (**0011 1**,111 0101) W 4D3: (**0100 1**,101 0011) R 3BA: (**0011 1**, 011 1010)
Physical addr.: R (0111 1,111 0101) = 7F5, W (1011 0,101 0011) = B53, R (1011 0, 011 1010) = B3A

Warning: Logical page 7 resides in two different frames at two different times of the reference string

3. *(11/9/2018)* Consider the following string of memory references, for a process with a virtual address space of 4K words. References are expressed by hexadecimal logical addresses, preceded by R for read and W for write: W 3A1, R 3F5, R A64, W BD3, W 57E, R A08, R B85, W 3A0, R A1A, W A36, R B20, R 734, R AB8, R C4E, W B64. Compute the page reference string, assuming a page size of 512 words. Use for page replacement an Enhanced Second-Chance algorithm, exploiting two bits: reference (to be initialized with 0 at the first access after a page fault), and modify. Assume a page is always modified by a write, 3 frames are available, and the algorithm works adopting the following strategy: given the pointer (index) to the current page (FIFO head), perform a first round (iteration), without resetting the reference bit, in order to find the victim, (the priority order is: (reference,modify): (0,0), (0,1), (1,0), (1,1)); once the victim has been found, perform a second iteration in order to clear/reset the reference bit of "saved" pages (included between the starting position and the victim).

   Show page faults (references to pages outside the resident set) and compute their overall count. Show the resident set (pages mapped to frames) after each memory access, and explicitly indicate, for each frame, the reference and modify bits. Number pages starting from 0.

A. Notice that an hexadecimal digit corresponds to 4 bits Two digits correspond to 8 bits. Division by 512 (2^9) can be easily performed by removing the two least significant digits, then further dividing by 2: overall, it ie enough to divide the first hexadecimal digit by 2

References	1	1	5	5	2	5	5	1	5	5	5	3	5	6	5
Read/write	W	R	R	W	W	R	R	W	R	W	R	R	R	R	W
Resident Set	1_{01}	1_{11}	1_{11}	1_{11}	1_{11}	1_{11}	1_{11}	1_{11}	1_{11}	1_{11}	1_{11}	1_{01}	1_{01}	1_{01}	1_{01}
			5_{00}	5_{11}	5_{11}	5_{11}	5_{11}	5_{11}	5_{11}	5_{11}	5_{11}	5_{01}	5_{11}	5_{01}	5_{11}
					2_{01}	2_{01}	2_{01}	2_{01}	2_{01}	2_{01}	2_{01}	3_{00}	3_{00}	6_{00}	6_{00}
Page Fault	X		X		X							X		X	

Total number of page faults: ……5…….