Operating Systems: Solutions to exercises on memory management & virtual memory

1. Assume that a program has just referenced an address in virtual memory. Describe a scenario in which each of the following can occur. (If no such scenario can occur, explain why.)

   (a) TLB miss with no page fault:

   Sol: No page fault means that the page is already in the page table but has not be referenced yet, therefor the TLB does not contain this reference. This may also occur if the entry in the TLB has been removed while the page is still in memory

   (b) TLB miss with page fault:

   Sol: This will occur when the page is not in the page table

   (c) TLB hit with no page fault:

   Sol: Having a TLB hit means the page is already in the page table, therefore there can't be a page fault.

   (d) TLB hit with page fault:

   Sol: If there is a TLB hit, then there is no page fault. The TLB is a cache of the page table, if there is a TLB hit it means the page is in the page table already

2. Assume the states of thread are ready, running, and blocked, where a thread is either ready and waiting to be scheduled, is running on the processor, or is blocked (for example, waiting for I/O). Assume further that the thread is in the running state, answer the following questions, and explain your answers:

   (a) Will the thread change state if it incurs a page fault? If so, to what state will it change?

   Sol. Yes, it changes from running to blocked. As we have seen, when a page fault occurs, the OS has to do a lot of things to address the page fault such as checks whether the page is really invalid or just on disk, finds a free memory frame, schedules a disk access to load the page into the frame,updates the page table with the new logical-physical mapping, updates the valid bit of that entry,and eventually restarts the process by change its state from Blocked to Ready

   (b) Will the thread change state if it generates a TLB miss that is resolved in the page table? If so, to what state will it change?

   Sol: No. If a page table entry is not found in the TLB (TLB miss), the page number is used to index and process the page table. If the page is already in main memory, then TLB is updated to include the new page entry, while the process execution continues since there is no I/O operation needed.

(c) Will the thread change state if an address reference is resolved in the page table? If so, to what state will it change?

Sol: No, because no I/O operation is needed is the address reference is resolved in the page table, which indicates the page needed is loaded in the main memory already

3. Consider a system with four pages frames and a program that uses eight pages. Consider the reference string 0172327103 and assume that all four page frames are initially empty. How many page faults occur assuming

   (a) FIFO replacement strategy. Sol: 6 page faults

   (b) LRU replacement. Sol: 7 page faults

4. Assume we have a demand paging memory. It takes 8 milliseconds to service a page fault if an empty page is available or the replaced page is not modified and 20 milliseconds if the replaced page is modified. Memory access time is 100 nanoseconds. Assume that the page to be replaced is modified 70 percent of the time. What is the maximum acceptable page-fault rate for an effective access time (EAT) of no more than 200 nanoseconds?

   Sol: Let p be page-fault rate.

   - The average access time for pages that are in main memory is $(1-p) \times 100ns$.
   - The average access time for page faults if empty frames are available or the replaced frames are not modified is $(p \times .3 \times 8,000,000ns)$
   - The average access time for page faults where replaced frames is not modified is $(p \times .7 \times 20,000,000ns)$

   We want the total EAT to be no more than 200 nanoseconds, i.e.

$$(1-p) \times 100ns + (p \times .3 \times 8,000,000ns) + (p \times .7 \times 20,000,000ns) \leq 200ns$$

$$(p \times 2,400,000ns) + (p \times 14,000,000ns) \leq 100ns$$

$$16,400,000ns \times p \leq 100$$

$$p \leq 0.0000061$$

5. Consider a demand-paging system with a paging disk that has an average access and transfer time of 20 milliseconds. Addresses are translated through a page table in main memory, with an access time of 1 microsecond per memory access. Thus, each memory reference through the page table takes two accesses. To improve this time, we have added an associative memory (TLB) that reduces access time to one memory reference if the page-table entry is in the associative memory. Assume

that 80 percent of the accesses are in the associative memory and that, of those remaining, 10 percent (or 2 percent of the total) cause page faults. What is the EAT, the effective memory access time?

Sol: 80% of the memory access are in the TLB, which means only one memory access is executed, costing 1 μs. The average access time for 80% of the memory accesses is is 0.8 μs.

The average access time for the 18% not in the TLB is $0.18 \times 2\mu s = 0.36\mu s$

The average access time for the 2% of memory accesses that cause page faults is $0.02 \times (2000\mu s + 2\mu s) = 400.02\mu s$

The EAT $= 0.8\mu s + 0.36\mu s + 400.02\mu s = 401.02\mu s$

6. Consider the page table shown below for a system with 12-bit virtual and physical addresses and with 256-byte pages. The list of free frames is D, E, F and they are to be allocated in the same order. A dash for a page frame indicates that the page is not in memory. Convert the following virtual addresses to their equivalent physical addresses in hexadecimal. In case of a page fault, you must use one of the free frames to update the page table and resolve the logical address to its corresponding physical address. All numbers are given in hexadecimal.

   - 9EF (binary 100111101111)
   - 111 (binary 000100010001)
   - 700 (binary 011100000000)
   - 0FF (binary 000011111111)

   Sol: 9EF 0EF; 111 211; 700 D00; 0FF EFF

Page	Frame
0	-
1	2
2	C
3	A
4	-
5	4
6	3
7	-
8	B
9	0

7. Same problem as previous one.

   - 2A1 (binary 001010100001)
   - 4E6 (binary 010011100110)

- 94A (binary 100101001010)
- 316 (binary 001100010110)

Page	Frame
0	4
1	B
2	A
3	-
4	-
5	2
6	-
7	0
8	C
9	1

8. Assuming a 1-KB page size, what are the page numbers and offsets for the following address references (provided as decimal numbers):

   (a) 3085

   (b) 42095

   (c) 215201

   (d) 650000

   (e) 2000001

   sol: Convert each decimal into binary. Split binary into part, entry in the page table and offset, convert binary part into decimal

references	page number	offset
3085	3	13
42095	41	111
215201	210	161
650000	634	784
2000001	1953	129

9. 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 addressed for the following logical addresses?
(a) 0,430
(b) 1,10
```

(c) 2,500
(d) 3,400
(e) 4,112

Sol:

(a) 219 + 430 = 649

(b) 2300 + 10 = 2310

(c) illegal reference; traps to operating system

(d) 1327 + 400 = 1727

(e) illegal reference; traps to operating system

10. Virtual memory

(a) Assume a process is only allocated 2 frames for its execution.

i. Suppose the process makes 12 references to the same logical page during a CPU burst, how many page faults will occur?
sol 1

ii. Suppose the process makes references to 12 different logical pages during a CPU burst, how many page faults will occur?
sol 12

iii. Suppose the process makes 12 references alternatively to the logical pages 1 and 13 (page 1, page 13, page 1, page 13, etc) during a CPU burst, how many page faults will occur?
sol 2

iv. Suppose the process makes 24 references alternatively to the logical pages 1, 2, 3 and 4 (page 1, page 2, page 3, page 4, page 1, page 2, page 3, page 4, etc) using the LRU replacement algorithm, how many page faults will occur?
sol 24

(b) For the following reference string, assuming 3 frames. Determine the number of page faults that will occur using each of the following page replacement algorithms. Show your work by filling the tables below.

a) FIFO (longest time a page has been in memory). Number of page faults is:

sol 8

	4	4	7	0	6	3	4	6	3	3	6	7	4	4	1
frame 1															
frame 2															
frame 3															

	4	4	7	0	6	3	4	6	3	3	6	7	4	4	1
frame 1	4				6	6	6					7			7
frame 2			7		7	3	3					3			1
frame 3				0	0	0	4					4			4

b) LRU (least recently referenced to). Number of page faults is: sol 9

	4	4	7	0	6	3	4	6	3	3	6	7	4	4	1
frame 1															
frame 2															
frame 3															

	4	4	7	0	6	3	4	6	3	3	6	7	4	4	1
frame 1	4				6	6	6					6	6		1
frame 2			7		7	3	3					3	4		4
frame 3				0	0	0	4					7	7		7

c) Optimal (latest page to be referenced). Number of page faults is:

sol 7

	4	4	7	0	6	3	4	6	3	3	6	7	4	4	1
frame 1															
frame 2															
frame 3															

	4	4	7	0	6	3	4	6	3	3	6	7	4	4	1
frame 1	4				4	4						4			4
frame 2			7		7	3						7			7
frame 3				0	6	6						6			1

(a) FIFO: 13 (b) LRU: 11 (c) Optimal: 10

11. You have a 6 pages virtual memory, $p_0, p_1, p_2, p_3, p_4, p_5$ and a 16 frames physical memory. Page p_0 is mapped to frame 0, page p_1 to frame 3, page p_2 to frame 11, page p_3 to frame 4, page p_4 to frame 15 and p_5 to frame 12.

(a) Fill the page table, the third row hold the valid-invalid bit:

page								
frame								
v-i								

(b) Assuming each page is 4 bytes, what is the size in bytes of the physical memory?

sol 64

(c) How the address 00111 is interpreted in a paging system, i.e.

i. What are the bits that refer to an entry in the page table?
sol 001

ii. What are the bits that refer to an offset in a frame?
so 11

iii. Is 00111 a valid address?
sol yes

(d) What is the physical address of the logical address 00101 (5)?
sol 5

(e) Assume pages p_0, p_1, p_2 of the above virtual memory form segment 0 and p_3, p_4, p_5 form segment 1, each segment is assigned physical memory through segmentation hardware. The physical memory has 64 bytes

i. What will be the value of the limit register of each segment?
sol 12

ii. The segment table has 4 entries. What is the interpretation of the logical address 11101010, i.e. which bit(s) refer to a segment table entry and which one refer to an offset in the physical memory?
sol 11 101010

iii. Will the logical address 01101010 provokes a segment fault?
sol yes

iv. Assume the base register of segment 1 is 6 while the base register of segment 2 is 24. What are the physical address of the logical addresses 00000001 and 01000100?
sol 7 and 27