Operating Systems: Exercises on memory management and 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:

   (b) TLB miss with page fault:

   (c) TLB hit with no page fault:

   (d) TLB hit with page fault:

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?

   (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?

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

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.

   (b) LRU replacement.

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?

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?

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)

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
9. 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?
      ii. Suppose the process makes references to 12 different logical pages during a CPU burst, how many page faults will occur?
      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?
      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?
   (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:

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

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

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

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

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

10. 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?

    (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?

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

    iii. Is 00111 a valid address?

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