COMP 346 – Fall 2017
Theory Assignment 3

Answer all questions

Due Date	**By 11:59pm Friday, November 10, 2017**
Format	Assignments must be typed and submitted online to Moodle system. **Scanned hand-written assignments will be discarded**.
Late Submission:	none accepted
Purpose:	The purpose of this assignment is to help you learn the overview of computer operating systems and Input and output mechanisms.
CEAB/CIPS Attributes:	Design/Problem analysis/Communication Skills

Question # 1

In a 64-bit computer system that uses pure paging with 16KB page size, if each page table entry is 4 bytes long then:

i. What is the maximum size of a page table?
ii. What is the maximum size of user-space main memory that can be supported by this scheme?
iii. If hierarchical paging is used then what is the total level of hierarchies required? Assume that 2-level of hierarchy corresponds to an outer page table and an inner page table. Show all relevant calculations.
iv. If the inverted paging scheme is used instead of paging, with a 16KB page/frame size, then what is the maximum size of the inverted page table considering the same memory size calculated in (ii) above? Assume that each entry of the inverted page table is 4 bytes long.

Question # 2

Answer the following questions:

i. (a) What are relocatable programs? (b) What makes a program relocatable? (c) From the OS memory management context, why programs (processes) need to be relocatable?
ii. What is (are) the advantage(s) and/or disadvantage(s) of small versus big page sizes?
iii. What is (are) the advantage(s) of paging over segmentation?
iv. What is (are) the advantage(s) of segmentation over paging?

Explain your answers.

Question # 3

Which of the following programming techniques and data structures (in a user-level program) are good for a demand-paged environment, and which are bad? Explain your answer.

i. Sequential search through a linked list
ii. Sequential search through an array
iii. Binary tree search
iv. Hashing with linear probing
v. Queue implemented using a circular array.

Question # 4

Consider a demand-paged system where the page table for each process resides in main memory. In addition, there is a fast-associative memory (also known as TLB which stands for Translation Look-aside Buffer) to speed up the translation process. Each single memory access takes 1 microsecond while each TLB access takes 0.2 microseconds. Assume that 2% of the page requests lead to page faults, while 98% are hits. On the average, page fault time is 20 milliseconds (includes everything: TLB/memory/disc access time and transfer, and any context switch overhead). Out of the 98% page hits, 80 % of the accesses are found in the TLB and the rest, 20%, are TLB misses. Calculate the effective memory access time for the system.

Question # 5

Consider the two-dimensional integer array A:

```
int A[][] = new int[100][100];
```

Assume that page size of the system = 200 bytes, and each integer occupies 1 byte. Also assume that A[0][0] is located at location 200 (i.e., page 1) in the paged memory system, and the array is stored in memory in row-major order (i.e., row by row). A small process that manipulates the array resides at page 0 (locations 0 to 199). Thus every instruction fetch is from page 0. The process is assigned 3 memory frames: frame 0 is already loaded with page 0 (i.e., process code), and the other two frames are initially empty.

a) Consider the following code fragment for initializing the array:

```
for (int i = 0; i < 100; i++)
        for (int j = 0; j < 100; j++)
            A[i][j] = 0;
```

If the LRU (Least Recently Used) page replacement scheme is used then what is the total number of page faults in executing the previous initialization code?

b) Now, instead, consider the following code fragment for initializing the array:

```
for (int j = 0; j < 100; j++)
        for (int i = 0; i < 100; i++)
            A[i][j] = 0;
```

If the LRU page replacement scheme is used then what is the total number of page faults in executing the above initialization code?

Question #6

Discuss situations in which the least frequently used (LFU) page replacement algorithm can generate fewer page faults than the least recently used (LRU) page replacement algorithm. Also discuss under what situations the opposite holds.