LAB 8
Virtual Memory Management and Performance

Labs must be submitted to the Assignments section of D2L by the due date indicated for full credit. Only one group member should submit this document. Late submissions will not be accepted.

Group Member Names: *Add row as needed if there is more than 4 members.*
1) Kingston Keobounphan
2)Cole hemiuk
3)Mlevin chew
4)

Virtual memory systems impose significant performance penalties on a computer system. It is important to understand the behaviour of virtual memory systems to make the most efficient use of memory resources.

If demand paging is used, pages are only brought in as needed. Therefore, it is simple to *over-allocate* memory such that the total memory requirements of all processes exceed available physical memory. If no frames are available when a page fault occurs, the system must replace a page in physical memory by finding a victim frame using a page replacement algorithm.

To speed up memory access, we can use a special, small, fast-lookup hardware cache called the Translation Look-aside Buffer (TLB). The TLB holds a part of the page table, usually the most recently used page table entries. Each entry is the TLB consist a key and a value just like a hash table. The search is fast because the item is compared with all the keys simultaneously, but the hardware to support TLB is expensive.

CMPS 369 Operating Systems for Software Developers – Lab 8

1. A process accesses pages in the following order (Assume all frames are initially empty – Pure Demand Paging):

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

   Determine the number of page faults that will occur for the following algorithms: (Show your work!)

   a. First In First Out replacement algorithm with 3 frames available to the process.

1	2	3	4	2	1	5	6	2	1	2	3	7	6	3	2	1	2	3	6
		3	3	3	3	5	5	5	1	1	1	1	6	6	6	6	6	3	3
	2	2	2	2	1	1	1	2	2	2	2	7	7	7	7	1	1	1	1
1	1	1	4	4	4	4	6	6	6	6	3	3	3	3	2	2	2	2	6
Miss	Miss	Miss	Miss	Hit	Miss	Miss	Miss	Miss	Miss	Hit	Miss	Miss	Miss	Hit	Miss	Miss	Hit	Miss	Miss

Total Page Faults=16

   b. Least Recently Used replacement algorithm with 3 frames available to the process.

LRU - 3 Frames

1	2	3	4	2	1	5	6	2	1	2	3	7	6	3	2	1	2	3	6
		3	3	3	1	1	1	2	2	2	2	2	6	6	6	1	1	1	5
	2	2	2	2	2	2	6	6	6	6	3	3	3	3	3	3	3	3	3
1	1	1	4	4	4	5	5	5	1	1	1	7	7	7	2	2	2	2	2
f	f	f	f	h	f	f	f	f	f	h	f	f	f	h	f	f	h	h	f

Total Page Faults = 15

   c. Least Recently Used replacement algorithm with 5 frames available to the process.

LRU - 5 frames

1	2	3	4	2	1	5	6	2	1	2	3	7	6	3	2	1	2	3	6
						5	5	5	5	5	5	7	7	7	7	7	7	7	7
			4	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3
		3	3	3	3	3	6	6	6	6	6	6	6	6	6	6	6	6	6
	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
f	f	f	f	h	h	f	f	h	h	h	f	f	h	h	h	h	h	h	h

Total Page Faults = 8

d. What is the minimum number of page faults that can occur for this process, regardless of algorithm or number of available frames? Why?

The minimum amount of page faults would be 7 regardless of algorithm or number of available frames because there are only 7 different pages that will replace a frame which will subsequently cause a page fault every time the page replacement process occurs.

2. A demand paging system provides a TLB (20 ns access time), cache memory (30 ns access time), and main memory (100 ns access time, **including** the cache "miss"). The page table is found in the TLB 60% of the time, and not in the TLB but in the cache 10% of the time. A process memory location is found in the cache 80% of the time. *Calculate the effective access time. (Show your work!)*

   Hint: you will need to consider all 6 of the possible combinations of where the page table is found and where the process memory location is found.

TLB: 20ns,
Cache: 30ns
RAM: 100ns
(1-P) where P is hit ratio
(1-0.60) = .40
(1-0.80) = .20

PAGE TABLE	DATA	
TLB	cache	[TLB: 20 ns; CACHE: 30 ns] ---> 0.40(20 ns + 30 ns) ---> **20**
TLB	memory	[TLB: 20 ns; RAM: 100 ns] ---> 0.20(20 ns + 100 ns) ---> **24**
CACHE	cache	[CACHE: 20 ns + 30 ns; CACHE:

		30 ns] ---> 0.40(20 ns + 30 ns + 30 ns) ---> **28**
CACHE	memory	[CACHE: 20 ns + 30 ns; RAM: 100 ns] ---> 0.20(20 ns + 30 ns + 100 ns) ---> **30**
MEMORY	cache	[RAM: 20 ns + 100 ns; CACHE: 30 ns] ---> 0.40(20 ns + 100 ns + 30 ns) ---> **60**
MEMORY	memory	[RAM: 20 ns + 100 ns; RAM: 100 ns] ---> 0.20(20 ns + 100 ns + 100 ns) ---> **44**

Next Step:
TLB ---> 60% ---> 0.60(20 + 24) ---> 26.4
CACHE ---> 10% ---> 0.10(28 + 30) ---> 5.8
RAM ---> 30% (Difference of 100% - [TLB + CACHE]) ---> 0.30(60 + 44) ---> 31.2

Final Step:
EAT = 26.2 + 5.8 + 31.2 = 63.2

BONUS ASSIGNMENT #2!