

## Operating Systems Design - 19CS2106S, 19CS2106A

### Home Assignment - CO-III

Mode of submission: Post Handwritten scanned documents in LMS

Submission date: on or before 30<sup>th</sup> October 2020

1. Assume that memory contains only three frames. Initially all three frames are empty. The page reference string is 7,0,1,2,0,3,0,4,2,3,0,3,2,1,2,0,1,7,0,1. Find how many page faults occurs using FIFO, **Optimal Page replacement**, and **Least Recently Used** algorithms. Pictorially show which pages are replaced.
2. Consider the following page reference string:  
1, 2, 3, 4, 2, 1, 5, 6, 2, 1, 2, 3, 7, 6, 3, 2, 1, 2, 3, 6.  
How many page faults would occur for the following replacement algorithms, assuming one, two, three, four, five, six, or seven frames? Remember all frames are initially empty, so your first unique pages will all cost one fault each.
  - LRU replacement
  - FIFO replacement
  - Optimal replacement
  - Clock replacement
3. Consider a disk queue with requests for I/O to blocks on cylinders 98, 183, 41, 122, 14, 124, 65, 67. The FCFS scheduling algorithm is used. The head is initially at cylinder number 53. The cylinders are numbered from 0 to 199. The total head movement (in number of cylinders) incurred while servicing these requests is \_\_\_\_\_.
4. SCAN is an algorithm that will service requests to the nearest end. Usually tracks start at 0 and the end track is given.  
For example, if the head is currently at 180 - it will go to 199 and service all requests in that direction.  
queue = 98, 183, 37, 122, 14, 124, 65, 67  
start = 53  
end = 199  
For the given data, calculate the total head movement
5. C-SCAN algorithm is very similar to SCAN in the sense that it will still go and hit the closest end but it “jumps” to the other end and services going in the opposite direction.  
For example, if the head starts at 100 - it will service all requests down to 0. But then “jump” up to 199 and service all requests going back down to 0. You should not include the distance of the “jump” in the total head movements.  
queue = 98, 183, 37, 122, 14, 124, 65, 67  
start = 53  
end = 199  
For the given data, calculate the total head movement
6. The LOOK algorithm is similar to SCAN but allows you to “look ahead” and not require you to end the end or beginning. This will save you head movements.  
queue = 98, 183, 37, 122, 14, 124, 65, 67  
start = 53  
end = 199  
For the given data, calculate the total head movement
7. The C-LOOK algorithm works similar to C-SCAN where it “jumps” but just like in LOOK it will not go to the end. It will “look” ahead to determine where it needs to go and stop.  
queue = 98, 183, 37, 122, 14, 124, 65, 67  
start = 53  
end = 199  
For the given data, calculate the total head movement
8. What if you wanted to run a program that needs more memory than you have? Consider a machine with 64 MB physical memory and a 32-bit virtual address space. If the page size is 4KB, what is the approximate size of the page table?

9. Consider a simple system running a single process. The size of physical frames and logical pages is 16 bytes. The RAM can hold 3 physical frames. The virtual addresses of the process are 6 bits in size. The program generates the following 20 virtual address references as it runs on the CPU: 0, 1, 20, 2, 20, 21, 32, 31, 0, 60, 0, 0, 16, 1, 17, 18, 32, 31, 0, 61. (Note: the 6-bit addresses are shown in decimal here.) Assume that the physical frames in RAM are initially empty and do not map to any logical page.
  - (a) Translate the virtual addresses above to logical page numbers referenced by the process. That is, write down the reference string of 20 page numbers corresponding to the virtual address accesses above. Assume pages are numbered starting from 0, 1, ...
  - (b) Calculate the number of page faults generated by the accesses above, assuming a FIFO page replacement algorithm. You must also correctly point out which page accesses in the reference string shown by you in part (a) are responsible for the page faults.
  - (c) Repeat (b) above for the LRU page replacement algorithm.
  - (d) What would be the lowest number of page faults achievable in this example, assuming an optimal page replacement algorithm were to be used? Repeat (b) above for the optimal algorithm.
10. Consider a system with only virtual addresses, but no concept of virtual memory or demand paging. Define total memory access time as the time to access code/data from an address in physical memory, including the time to resolve the address (via the TLB or page tables) and the actual physical memory access itself. When a virtual address is resolved by the TLB, experiments on a machine have empirically observed the total memory access time to be (an approximately constant value of)  $t_h$ . Similarly, when the virtual address is not in the TLB, the total memory access time is observed to be  $t_m$ . If the average total memory access time of the system (averaged across all memory accesses, including TLB hits as well as misses) is observed to be  $t_x$ , calculate what fraction of memory addresses are resolved by the TLB. In other words, derive an expression for the TLB hit rate in terms of  $t_h$ ,  $t_m$ , and  $t_x$ . You may assume  $t_m > t_h$ .
11. Consider a system with a 6 bit virtual address space, and 16 byte pages/frames. The mapping from virtual page numbers to physical frame numbers of a process is (0,8), (1,3), (2,11), and 4 (3,1). Translate the following virtual addresses to physical addresses. Note that all addresses are in decimal. You may write your answer in decimal or binary.
12. Consider a system with several running processes. The system is running a modern OS that uses virtual addresses and demand paging. It has been empirically observed that the memory access times in the system under various conditions are:  $t_1$  when the logical memory address is found in TLB cache,  $t_2$  when the address is not in TLB but does not cause a page fault, and  $t_3$  when the address results in a page fault. This memory access time includes all overheads like page fault servicing and logical-to-physical address translation. It has been observed that, on an average, 10% of the logical address accesses result in a page fault. Further, of the remaining virtual address accesses, two-thirds of them can be translated using the TLB cache, while one-third require walking the page tables. Using the information provided above, calculate the average expected memory access time in the system in terms of  $t_1$ ,  $t_2$ , and  $t_3$ .
13. Consider an operating system that uses 48-bit virtual addresses and 16KB pages. The system uses a hierarchical page table design to store all the page table entries of a process, and each page table entry is 4 bytes in size. What is the total number of pages that are required to store the page table entries of a process, across all levels of the hierarchical page table?
14. Consider a 64-bit system running an OS that uses hierarchical page tables to manage virtual memory. Assume that logical and physical pages are of size 4KB and each page table entry is 4 bytes in size.
15. Consider a system with 16 bit virtual addresses, 256 byte pages, and 4 byte page table entries. The OS builds a multi-level page table for each process. Calculate the maximum number of pages required to store all levels of the page table of a process in this system.
16. Consider a virtual memory system with physical memory of 8GB, a page size of 8KB and 46 bit virtual address. Assume every page table exactly fits into a single page. If page table entry size is 4B then how many levels of page tables would be required.

17. A certain computer system has the segmented paging architecture for virtual memory. The memory is byte addressable. Both virtual and physical address spaces contain  $2^{16}$  bytes each. The virtual address space is divided into 8 non-overlapping equal size segments. The memory management unit (MMU) has a hardware segment table, each entry of which contains the physical address of the page table for the segment. Page tables are stored in the main memory and consists of 2 byte page table entries. Assume that each page table entry contains (besides other information) 1 valid bit, 3 bits for page protection and 1 dirty bit. How many bits are available in page table entry for storing the aging information for the page? Assume that page size is 512 bytes.
18. A certain computer system has the segmented paging architecture for virtual memory. The memory is byte addressable. Both virtual and physical address spaces contain  $2^{16}$  bytes each. The virtual address space is divided into 8 non-overlapping equal size segments. The memory management unit (MMU) has a hardware segment table, each entry of which contains the physical address of the page table for the segment. Page tables are stored in the main memory and consists of 2 byte page table entries. What is the minimum page size in bytes so that the page table for a segment requires at most one page to store it? give the division of virtual address.