Homework #5

Partial Solutions

Question 2 (20 points) Consider a system where the virtual memory page size is 1KB (1024 bytes), and main memory consists of 4 page frames, which are empty initially. Now consider a process which requires 8 pages of storage. At some point during its execution, the page table is as shown below:

Virtual page #	Physical page #	Valid Flag
0		No
1		No
2	2	Yes
3	3	Yes
4		No
5		No
6	0	Yes
7	1	Yes

1. List the virtual address ranges that will result in a page fault.

   Virtual address range for a page fault includes [1, 2K-1], [4K, 6K-1].

2. Give the following **ordered** references to the virtual addresses (i) 4500, (ii) 8000, (iii) 3000, (iv) 1100, please calculate the main memory (physical) addresses. If there is a page fault, please use LRU based page replacement to replace the page. How which page will be affected and compute the physical addresses after the page fault. We assume the reference string is …2 4 7 3 0 4 3 0 7 5 0 7 6 0 2 3 6 4 7 6 3 2 before the new reference and the page table above is the actual page table before the new references.

   Virtual address 4500 = 4*1024 + 404, page # is 4 and offset is 404
   Virtual address 8000 = 7*1024 + 832, page # is 7 and offset is 832
   Virtual address 3000 = 2*1024 + 952, page # is 2 and offset is 952
   Virtual address 1100 = 1*1024 + 76, page # is 1 and offset is 76

   From the section 1 above, we know reference (1) will cause a page fault since the reference falls in the page fault address range of [4K, 6K-1]. Given the reference string (… 7 6 3 2), according to the LRU page replacement algorithm, we know that the page current resides in the physical frame 1 will be replaced (which is page 7). After the replacement, the page table will be as follows.

Virtual page #	Physical page #	Valid Flag
0		No

1		No
2	2	Yes
3	3	Yes
4	1	Yes
5		No
6	0	Yes
7		No

So now the reference to virtual address 4500, will be directed to physical frame 1 and the physical address is 1*1024+404 = 1428.

Second reference to virtual address 8000 will also cause a page fault, similarly according to LRU and reference string (… 6 3 2 4), page 6 will be replaced. So the new page table will be as follows and the physical address is 0*1024+832 = 832.

Virtual page #	Physical page #	Valid Flag
0		No
1		No
2	2	Yes
3	3	Yes
4	1	Yes
5		No
6		No
7	0	Yes

Third reference to virtual address 3000 will not cause any page fault, according to the page table, we know the reference is in frame 2, so the physical address is the same as the virtual address, which is 3000.

Fourth reference to virtual address 1100 will cause a page fault; according to LRU and reference string (… 6 3 2 4 7 2), page 3 will be replaced. Then the physical address is 3*1024 + 76 = 3148.

Question 3 (30 points) Given a computer system with the following paging based addressing for virtual addresses. Please answer the following questions:

2 bits	5 bits	5 bits	5 bits	7 bits

1. What is the size of the virtual address space?

   The size of the virtual address space is 2^{24}Bytes = 16MB.

2. What is the page size?

   The page size is determined by the offset length (7 bits), so the page size is 2^7=128Bytes.

3. What is the maximum number of pages for a process?

   The maximum number of pages for a process is determined by the maximum process size (= maximum virtual address size) and the page size. In this specified system, the maximum number of pages = $2^{24}/2^{7} = 2^{17}$ = 128K.

4. Given the system has a TLB hit ratio of 99% and page fault rate of 1%. Please formulate the effective memory access time.

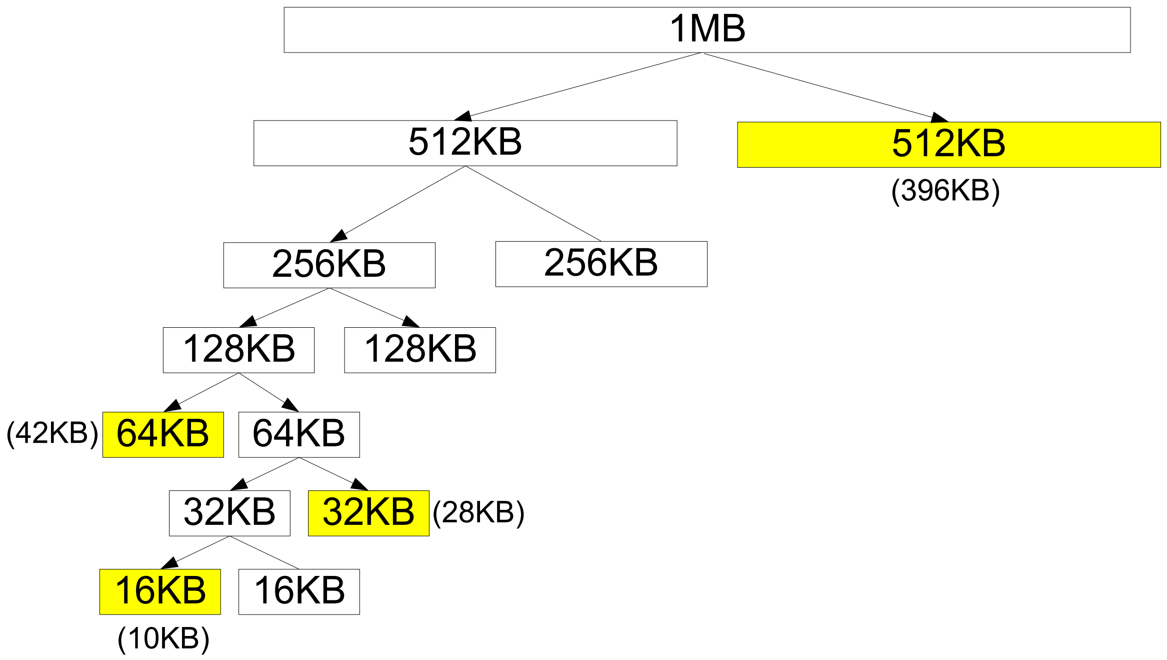
   Let TLB access time be *a*, memory access time *b* and page fault handling time *c*. the effective memory access time is equal to

   99%(no page fault access time) + 1%(page fault scenario memory access time) = 99%[99%(TLB hit scenario memory access time)+1%(TLB miss memory access time)] + 1%(page fault scenario memory access time)

   = 99%[99%(a+b)+1%(a+5b)]+1%(a+4b+c)

Question 4. (20 points) Consider a system with 1MB of available memory and requests for 42KB, 396KB, 10KB, and 28KB. The system is using Buddy Allocation Algorithm.

a). (10 points) Show the amount of memory allocated for each request and the state of memory after each request. Assume there is no memory release.



b). (5 points) Why does internal fragmentation occur with buddy allocation? How much internal fragmentation exists in this scenario?

Buddy memory allocation algorithm allocates memory in powers of 2, i.e., 2^x, where x is an integer. If the request is within (2^x, 2^{x+1}], then 2^{x+1} memory will be allocated. Thus the internal fragmentation occurs. In this scenario, the total internal fragmentation is (64KB-42KB)+(16KB-10KB)+(32KB-28KB)+(512KB-396KB) = 22KB + 6KB + 4KB + 116KB = 148KB.

c). (5 points) Why does external fragmentation occur with buddy allocation? How much external fragmentation exists in this scenario?

The buddy memory allocation technique inherits part of characteristics of contiguous allocation and dynamic allocation, so there will be external fragmentation as well. In this scenario, the system will have 16KB + 128KB + 256KB = 400KB external fragmentation, which means any request size between (256KB, 400KB) will not be granted.