**Homework Assignment #3**

***Due Date: 3/11 (hard deadline), please submit a soft copy via Blackboard. Please note that this is a hard deadline as we plan to post the solution after the deadline to help prepare for mid-term exam.***

***Please name your submission file starting with “LastName\_FirstName\_HW3”.***

**Q1.** In this problem you are asked to compare the storage needed to keep track of free memory using a bitmap versus using a linked list. Assume the 128-MB memory is allocated in units of *n* bytes. For the linked list, assume that memory consists of an alternating sequence of segments and holes, each 64 KB. Also assume that each node in the linked list needs a 32-bit memory address, a 16-bit length, and a 16-bit next-node field. How many bytes of storage is required for each method? Which one is better?

Since the total amount of 128MB is 2^27 bits and 2^24 bytes.

So, for the linked list, due to 4+2+2 bytes, which is 2^16 bits, thus, the linked list needs 2^27 / 2^16 = 2^11 nodes. The linked list will need 2^14 bytes space.

Therefore, for the bitmap, 2^24 / n = 2^14, and n = 2^10.

Conclusion, for n<1KB, the linked list is better, otherwise, the bitmap will be better because when n increase the storage space needed decreases.

**Q2. (Chapter 3, Problem 4)** Consider a swapping system in which memory consists of the following hole sizes in memory order: 10MB, 4MB, 20MB, 18MB, 7MB, 9MB, 12MB, and 15MB. Which hole is taken for successive segment requests of

(a) 12 MB

(b) 10 MB

(c) 9 MB

for first fit? Now repeat the question for best fit, worst fit, and next fit.

First fit takes 20MB, 10MB, 18MB

Best fit takes 12MB, 10MB, 9MB

Worst fit takes 20MB, 18MB, 15MB

Next fit takes 20MB, 18MB, 9MB

**Q3. (Chapter 3, Problem 20)** A computer has 32-bit virtual addresses and 4-KB pages. The program and data together fit in the lowest page (0-4095). The stack fits in the highest page. How many entries are needed in the page table if traditional (one-level) paging is used? How many page table entries are needed for two-level paging, with 10 bits in each part?

For one-level page table there are 1M entries, and for two-level page table there are 1K entries. Based on that, only three page tables are needed, one in top level and one in each of the lower level tables.

**Q4. (Chapter 3, Problem 22)** Supposea computer keeps its page table in memory. The overhead required for reading an entry from the page table is 5 nsec. To reduce this overhead, the computer has a TLB and can do a look up in 1 nsec. What hit rate is needed to reduce the mean overhead to 2 nsec?

Based on the 1h+5(1-h) = 2, it can give us h = 0.75. So h must greater than 0.75

**Q5. (Chapter 3, Problem 32)** In the WSClock algorithm of Fig. 3-20(c), the hand points to a page with R = 0. If , will this page be removed? What about if ?

The age of the page is 2204 – 1213 = 991. Based on that, if it will be evicted, but for , it is safe.

**Q6. (Chapter 3, Problem 36)** A computer has four page frames. The time of loading, time of last access, and the *R* and *M* bits for each page are as shown below (the times are in clock ticks):

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Page | Loaded | Last ref. | R | M |
| 0 | 126 | 280 | 1 | 0 |
| 1 | 230 | 265 | 0 | 1 |
| 2 | 140 | 270 | 0 | 0 |
| 3 | 110 | 285 | 1 | 1 |

1. Which page will NRU replace?

Page 2  
(b) Which page will FIFO replace?

Page 3  
(c) Which page will LRU replace?

Page 1  
(d) Which page will second chance replace?

Page 2

**Q7. (Chapter 3, Problem 41)** A computer provides each process with 65,536 bytes of address space divided into pages of 4096 bytes. A particular program has a text size of 32,768 bytes, a data size of 16,386 bytes, and a stack size of 15,870 bytes. Will this program fit in the address space? If the page size were 512 bytes, would it fit? Each page must contain either text, data, or stack, not a mixture of two or three of them.

Text is 8 pages and data is 5 pages, where the stack is 4 pages. The program doesn’t fit, but with 512 bytes page it fit. So with the small page size it works but not the large one.

**Q8.** Can a page be in two working sets at the same time? Please explain.

The answer is yes if the users are sharing the memory at the same time.

**Q9. (Chapter 3, Problem 46)** When segmentation and paging are both being used, as in MULTICS, first the segment descriptor must be looked up, then the page descriptor. Does the TLB also work this way, with two levels of lookup?

No. the exact page can be found in a single match.

**Q10. (Chapter 3, Problem 47)** We consider a program which has the two segments shown below consisting of instructions in segment 0, and read/write data in segment 1. Segment 0 has read/execute protection (only read/execute privilege), and segment 1 has read/write protection (only read/write privilege). The memory system is a demand-paged virtual memory system with virtual addresses that have a 4-bit page number, and a 10-bit offset. The page tables and protection are as follows (all numbers in the table are in decimal):

|  |  |  |  |
| --- | --- | --- | --- |
| Segment 0 | | Segment 1 | |
| Read/Execute | | Read/Write | |
| Virtual Page # | Page frame # | Virtual Page # | Page frame # |
| 0 | 2 | 0 | On Disk |
| 1 | On Disk | 1 | 14 |
| 2 | 11 | 2 | 9 |
| 3 | 5 | 3 | 6 |
| 4 | On Disk | 4 | On Disk |
| 5 | On Disk | 5 | 13 |
| 6 | 4 | 6 | 8 |
| 7 | 3 | 7 | 12 |

For each of the following cases, either give the real (actual) memory address which results from dynamic address translation or identify the type of fault which occurs (either page or protection fault).

1. Fetch from segment 1, page 1, offset 3

Address(14,3) No fault  
(b) Store into segment 0, page 0, offset 16

Address(NA) protection fault  
(c) Fetch from segment 1, page 4, offset 28

Address(NA) page fault

THE END.