# Homework Assignment #3 Solution

**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?

***ANSWER:*** *The bitmap needs 1 bit per allocation unit. With $2^{27}$ /n allocation units, this is $2^{24}$ /n bytes. The linked list has $2^{27}$ /$2^{16}$ or $2^{11}$ nodes, each of 8 bytes, for a total of $2^{14}$ bytes. For small n, the linked list is better. For large n, the bitmap is better. The crossover point can be calculated by equating these two formulas and solving for n. The result is 1 KB. For n smaller than 1 KB, a linked list is better. For n larger than 1 KB, a bitmap is better.*

**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.

***ANSWER:*** *First fit takes 20MB,10MB,18MB.*

*Best fit takes 12MB,10MB, and 9MB.*

*Worst fit takes 20 MB, 18 MB, and 15 MB.*

*Next fit takes 20 MB, 18 MB, and 9 MB.*

**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?

***ANSWER:*** *For a one-level page table, there are $2^{32}$ /$2^{12}$ or 1M pages needed. Thus, the page table must have 1M entries.*

*For two-level paging, the main page table has 1K entries, each of which points to a second page table. Only two of these are used. Thus in total only three page tables are needed, one in the top-level table and one in each of the lower-level tables, with 3K entries in total.*

**Q4. (Chapter 3, Problem 22) Suppose** a 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?

***ANSWER:*** *The effective instruction time is $1h + 5(1 - h)$, where h is the hit rate. If we equate this formula with 2 and solve for h, we find that h must be at least 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 $\tau = 400$, will this page be removed? What about if $\tau = 1000$?

***ANSWER:*** *The age of the page is 2204 − 1213 = 991. If $\tau$=400, it is definitely out of the working set and was not recently referenced so it will be evicted. The $\tau$ = 1000 situation is different. Now the page falls within the working set (barely), so it is not removed.*

**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 |

(a) Which page will NRU replace?
(b) Which page will FIFO replace?
(c) Which page will LRU replace?
(d) Which page will second chance replace?

***ANSWER:*** *NRU replaces page 2. FIFO replaces page 3. LRU replaces page 1. Second chance replaces 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.

***ANSWER:*** *The text is eight pages, the data are five pages, and the stack is four pages. The program does not fit because it needs 17 4096-byte pages. With a 512-byte page, the situation is different. Here the text is 64 pages, the data are 33 pages, and the stack is 31 pages, for a total of 128 512-byte pages, which fits. With the small page size it is OK, but not with the large one.*

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

***ANSWER:*** *If pages can be shared, yes. For example, if two users of a timesharing system are running the same editor at the same time, and the program text is shared rather than copied, some of those pages may be in each user's working set 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?

***ANSWER:*** *No. The search key uses both the segment number and the virtual page number, so 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).
(a) Fetch from segment 1, page 1, offset 3
(b) Store into segment 0, page 0, offset 16
(c) Fetch from segment 1, page 4, offset 28

***ANSWER:***

| | *Address* | *Fault ?* |
|---|---|---|
| *(a)* | *(14,3)* | *No* |
| *(b)* | *NA* | *Protection fault: Write to read/execute segment* |
| *(c)* | *NA* | *Page fault* |

THE END.