Chapter 8

8.3 Given memory partitions of 100K, 500K, 200K, 300K, and 600K (in order), how would each of **First-Fit, Best-fit, and Worst-fit a**lgorithms place processes of 212K, 417K, 112K, and 426K (in order)? Which algorithm makes the most efficient use of memory?

Answer:

a. First-fit:
    a. 212K put in 500K partition
    b. 417K put in 600K partition
    c. 112K put in 288K partition (new partition 500K-212K=288K)
    d. 426K must wait

b. Best-fit
    a. 212K put in 300K partition
    b. 417K put in 500K partition
    c. 112K put in 200K partition
    d. 426K put in 600K partition

c. Worst-fit
    a. 212K put in 600K partition
    b. 417K put in 500K partition
    c. 112K put in 388K partition (600K – 212K = 388K)
    d. 426K must wait

In this example, Best-fit is turns out to be the best.

8.5 Compare the main memory organization schemes of contiguous-memory allocation, pure segmentation, and pure paging with respect to the following issues:
a. external fragmentation
b. internal fragmentation
c. ability to share code across processes

Answer:

contiguous memory allocation scheme suffers from external fragmentation as address spaces are allocated contiguously and holes develop as old processes dies and new processes are initiated. It also does not allow processes to share code, since a process’s virtual memory segment is not broken into non-contiguous finegrained segments.

Pure segmentation also suffers fromexternal fragmentation as a segment of a process is laid out contiguously in physical memory and fragmentation would occur as segments of dead processes are replaced by segments of new processes.

Segmentation, however, enables processes to share code; for instance, two different processes could share a code segment but have distinct data segments. Pure paging does not suffer from external fragmentation, but instead suffers frominternal fragmentation. Processes are allocated in

page granularity and if a page is not completely utilized, it results in internal fragmentation and a corresponding wastage of space.
Paging also enables processes to share code at the granularity of pages.

8.9 Consider a paging system with the page table stored in memory.
a. If a memory reference takes 200 nanoseconds, how long does a paged memory reference take?
b. If we add associative registers, and 75 percent of all page-table references are found in the associative registers, what is the effective memory reference time? (Assume that finding a page-table entry in the associative registers takes zero time, if the entry is there.)
Answer:
a. 400 nanoseconds; 200 nanoseconds to access the page table and 200 nanoseconds to access the word in memory.
b. Effective access time = 0.75 $\times$ (200 nanoseconds) + 0.25 $\times$ (400nanoseconds) = 250 nanoseconds.

8.12: Considering the segment table, what are the physical addresses for the following logical addresses?

Segment	Base	Length
0	219	600
1	2300	14
2	90	100
3	1327	580
4	1952	96

What are the physical addresses for the following logical addresses?
a. 0,430
b. 1,10
c. 2,500
d. 3,400
e. 4,112

Answer: Physical addresses = Segment Base Address + Offset

a. (219+430) = **649**

b. (2300+10) = **2310**

c. (90+500)=590 However since the length of segment 2 is 100, the offset of 500 is out of the segment’s permissible memory bounds. Hence this access is **invalid.**

d. (1327+400)= **1727**

e. (1952+112)=2064 Since the length of the segment 4 is 96, the offset of 112 is out of the segment’s permissible memory bounds. Hence this access in **invalid**.