Chapter 8 Exercises: 8.9, 8.11, 8.13, 8.18, 8.19, 8.23, 8.24, 8.25, 8.28, 8.29

Problem 8.9

a. Internal fragmentation is the area in a region or a page that is not used by the process occupying that region or page. This space is unavailable for use by other processes until that process is finished and the page or region is released.
b. External fragmentation is unused space between allocated regions of memory. Typically external fragmentation results in memory regions that are too small to satisfy a memory request, but if we were to combine all the regions of external fragmentation, we would have enough memory to satisfy a memory request.

Problem 8.11

1. First-fit:
   a. 115 KB is put in 300 KB partition, leaving (185 KB, 600 KB, 350 KB, 200 KB, 750 KB, 125 KB)
   b. 500 KB is put in 600 KB partition, leaving (185 KB, 100 KB, 350 KB, 200 KB, 750 KB, 125 KB)
   c. 358 KB is put in 750 KB partition, leaving (185 KB, 100 KB, 350 KB, 200 KB, 392 KB, 125 KB)
   d. 200 KB is put in 350 KB partition, leaving (185 KB, 100 KB, 150 KB, 200 KB, 392 KB, 125 KB)
   e. 375 KB is put in 392 KB partition, leaving (185 KB, 100 KB, 150 KB, 200 KB, 17 KB, 125 KB)

2. Best-fit:
   a. 115 KB is put in 125 KB partition, leaving (300 KB, 600 KB, 350 KB, 200 KB, 750 KB, 10 KB)
   b. 500 KB is put in 600 KB partition, leaving (300 KB, 100 KB, 350 KB, 200 KB, 750 KB, 10 KB)
   c. 358 KB is put in 750 KB partition, leaving (300 KB, 100 KB, 350 KB, 200 KB, 392 KB, 10 KB)
   d. 200 KB is put in 200 KB partition, leaving (300 KB, 100 KB, 350 KB, 0 KB, 392 KB, 10 KB)
   e. 375 KB is put in 392 KB partition, leaving (300 KB, 100 KB, 350 KB, 0 KB, 17 KB, 10 KB)

3. Worst-fit:
   a. 115 KB is put in 750 KB partition, leaving (300 KB, 600 KB, 350 KB, 200 KB, 635 KB, 125 KB)
   b. 500 KB is put in 635 KB partition, leaving (300 KB, 600 KB, 350 KB, 200 KB, 135 KB, 125 KB)
   c. 358 KB is put in 600 KB partition, leaving (300 KB, 242 KB, 350 KB, 200 KB, 135 KB, 125 KB)
   d. 200 KB is put in 350 KB partition, leaving (300 KB, 242 KB, 150 KB, 200 KB, 135 KB, 125 KB)
   e. 375 KB must wait

Thus, only worst-fit does not allow a request to be satisfied. An argument could be made that best-fit is most efficient as it leaves the largest holes after allocation. However, best-fit runs at time $O(n)$ and first-fit runs in constant time $O(1)$.

Problem 8.13

The contiguous memory allocation scheme suffers from external fragmentation as address spaces are allocated contiguously and holes develop as old processes die 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 noncontiguous fine grained segments.

Pure segmentation also suffers from external 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 from internal 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.
Problem 8.18

ASIDs provide address space protection in the TLB as well as supporting TLB entries for several different processes at the same time.

Problem 8.19

1) Contiguous-memory allocation requires the operating system to allocate the entire extent of the virtual address space to the program when it starts executing. This could be much larger than the actual memory requirements of the process.
2) Pure segmentation gives the operating system flexibility to assign a small extent to each segment at program startup time and extend the segment if required.
3) Pure paging does not require the operating system to allocate the maximum extent of the virtual address space to a process at startup time, but it still requires the operating system to allocate a large page table spanning all of the program’s virtual address space. When a program needs to extend the stack or the heap, it needs to allocate a new page but the corresponding page table entry is preallocated.

Problem 8.23

a. 12 + 8 = 20 bits.
b. 12 + 6 = 18 bits.

Problem 8.25

a. 100 nanoseconds: 50 nanoseconds to access the page table and 50 nanoseconds to access the word in memory.
b. Effective access time = 0.75 × (50 nanoseconds) + 0.25 × (100 nanoseconds) = 62.5 nanoseconds. (more accurate: 0.75 × (52 nanoseconds) + 0.25 × (100 nanoseconds) = 64 nanoseconds).

Problem 8.28

a. 219 + 430 = 649
b. 2300 + 10 = 2310
c. illegal reference, trap to operating system
d. 1327 + 400 = 1727
e. illegal reference, trap to operating system

Problem 8.29

In certain situations the page tables could become large enough that by paging the page tables, one could simplify the memory allocation problem (by ensuring that everything is allocated as fixed-size pages as opposed to variable-sized chunks) and also enable the swapping of portions of page table that are not currently used.