HW Chapter 7

7.4 Consider a logical address space of 64 pages with 1024 words per page; mapped onto a physical memory of 32 frames.

a. How many bits are required for the logical address?

b. How many bits are required for the physical address?

7.11 There are six holes in memory of sizes 300 KB, 600 KB, 350 KB, 200 KB, 750 KB, and 125 KB (in that address order). The system uses the variable partition memory management protocol. How would each of the first-fit, best-fit, and worst-fit algorithms place processes of size 115 KB, 500 KB, 358 KB, 200 KB, and 375 KB (arriving in that order) into memory?

First-fit:

Best-fit:

Worst-fit:

))

7.14 On a system with paging, it is not possible for a process to access memory that it does not own.

a. Why is it not possible?

*It is not possible because on a system with paging, memory addresses are stored as a logical page number stored in a page table and an offset. If a process wants to read outside of the memory it does not own, it is not in the page table so the process can’t read or write to it.*

b. How could the operating system support access to other memory?

*The operating system could simply add non process memory to the process’ page table and then it can use those memory addresses.*

c. Why might that be beneficial (give an example)?

*If a process wanted to use shared memory, this allows for easy access to the same physical memory addresses.*

7.25 Consider a paging system with the page table stored in memory.

a. If a memory reference takes 50 nanoseconds, how long does a paged memory reference take?

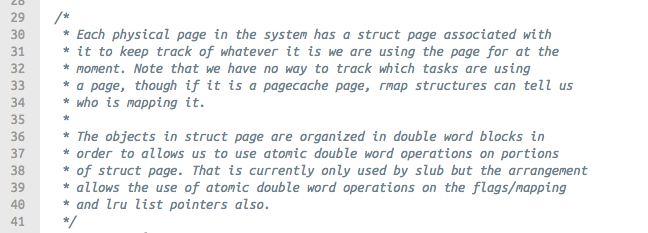
50 for the page table, and 50 for access in memory

b. If we add associative registers such that 75 percent of all page-table references are found in the TLB (associative registers), what is the effective memory reference time? (Assume that finding a page-table entry in the associative registers takes 2 nanoseconds if the entry is present.)

Practical: Give the location (path/file) and provide some evidence (e.g. the first few fields) that you have found the definition of the data structure used to represent frames of RAM, referred to in Linux as "physical pages" (hint: recall that definitions are typically placed in include files).

*Data Structure: Linux uses a struct in the pgd of the memory descriptor.*

*Found in: v4.15.10 /include/linux/mm\_types.h*

**