Kevin Jiang

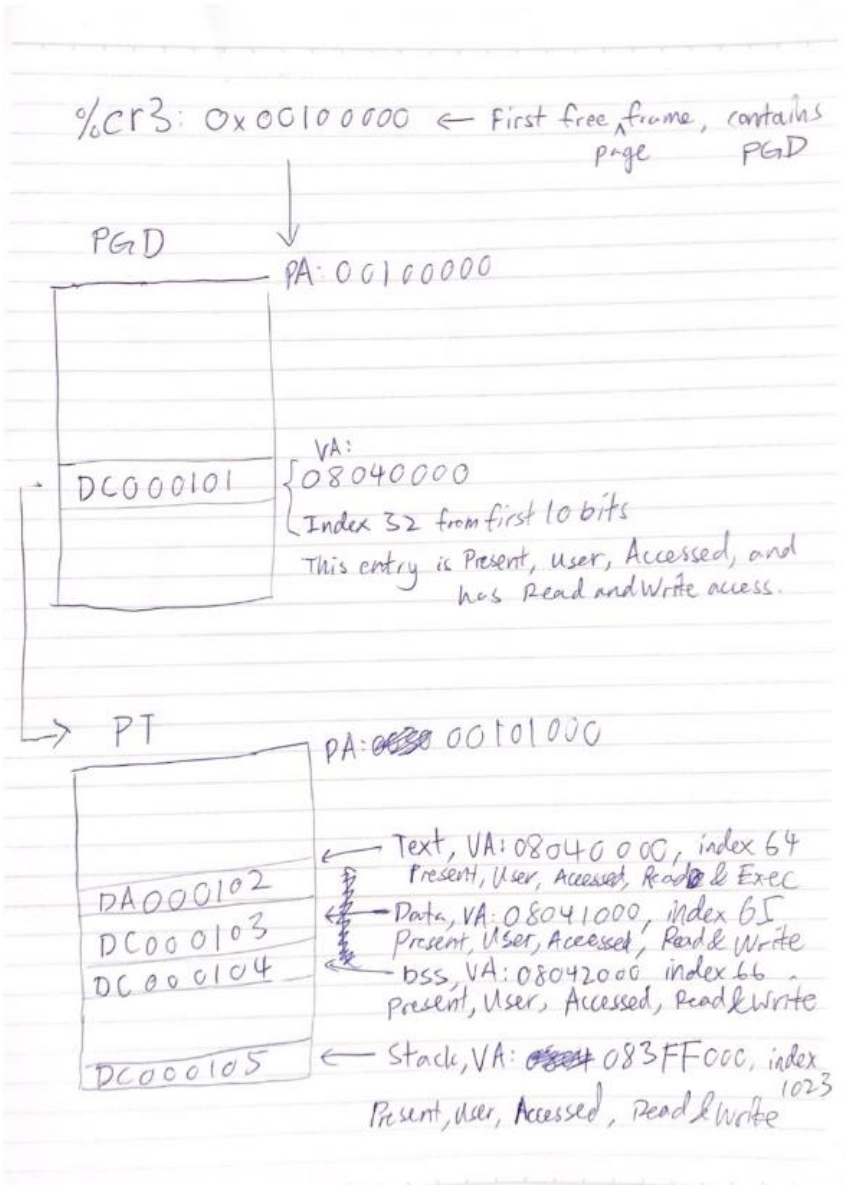
ECE357 OS

November 11th, 2020

Problem Set 5 Problems 1 and 2

Problem 1:

At the point of the program specified, f1() will have been called and d[3] will have been read and written into b[0]. That means the data, bss, and stack regions have all been accessed. The text region has already been accessed when running the program. The resulting program virtual address space looks as shown below. The PDE/PTE entries are shown in hexadecimal:



Problem 2:

A. Technically, it does not achieve “true” LRU behavior. The Accessed bit in the PTEs say nothing about when the page was accessed, only that it was accessed. Therefore, when PFRA moves two pages to the inactive list based on Accessed bit and PG_reference bit, the order in which they go into the list may not match the true order in which they were last accessed. PFRA still comes pretty close though, because inactive frames are arranged in FIFO order, so the pages which have been set as “inactive” for the longest will be sent to the free pages list first. The free pages list is also in FIFO order, further enforcing the notion of Least Recently Used.

B. The radix-64 tree is a data structure that can provide the struct page corresponding to the offset of a file, which is given in multiples of page sizes. Each struct page maps to a physical page. Minor and major faults require use of this data structure, which is when programs need to establish a PTE that maps to a valid physical page frame which contains the correct contents for the virtual page that is trying to be accessed. To get this valid physical page frame, the struct page is needed, which is when the radix-64 tree comes into play.

C. The vm_area_struct data structure is consulted. The linked list formed by many vm_data_structs are looked at to see if the faulting virtual address is outside of the user process’s virtual memory region. If it is outside, SIGSEGV is sent, otherwise it is handled as described in the notes.