Chapter 9

Project 4: Virtual Memory

9.1 Introduction

The purpose of this project is to add paging to the GeekOS kernel. This will require many small, but difficult changes to your project. More than any previous project, it will be important to implement one thing, test it and then move to the next one.

9.2 Changing the Project to Use Page Tables

The first step is to modify your project to use page tables and segmentation rather than just segments to provide memory protection. To enable using page tables, every region of memory your access (both kernel and data segment) must have an entry in a page table. The way this will work is that there will be a single page table for all kernel only threads, and a page table for each user process. In addition, the page tables for user mode processes will also contain entries to address the kernel mode memory. The memory layout for this is shown in Figure 9.1.

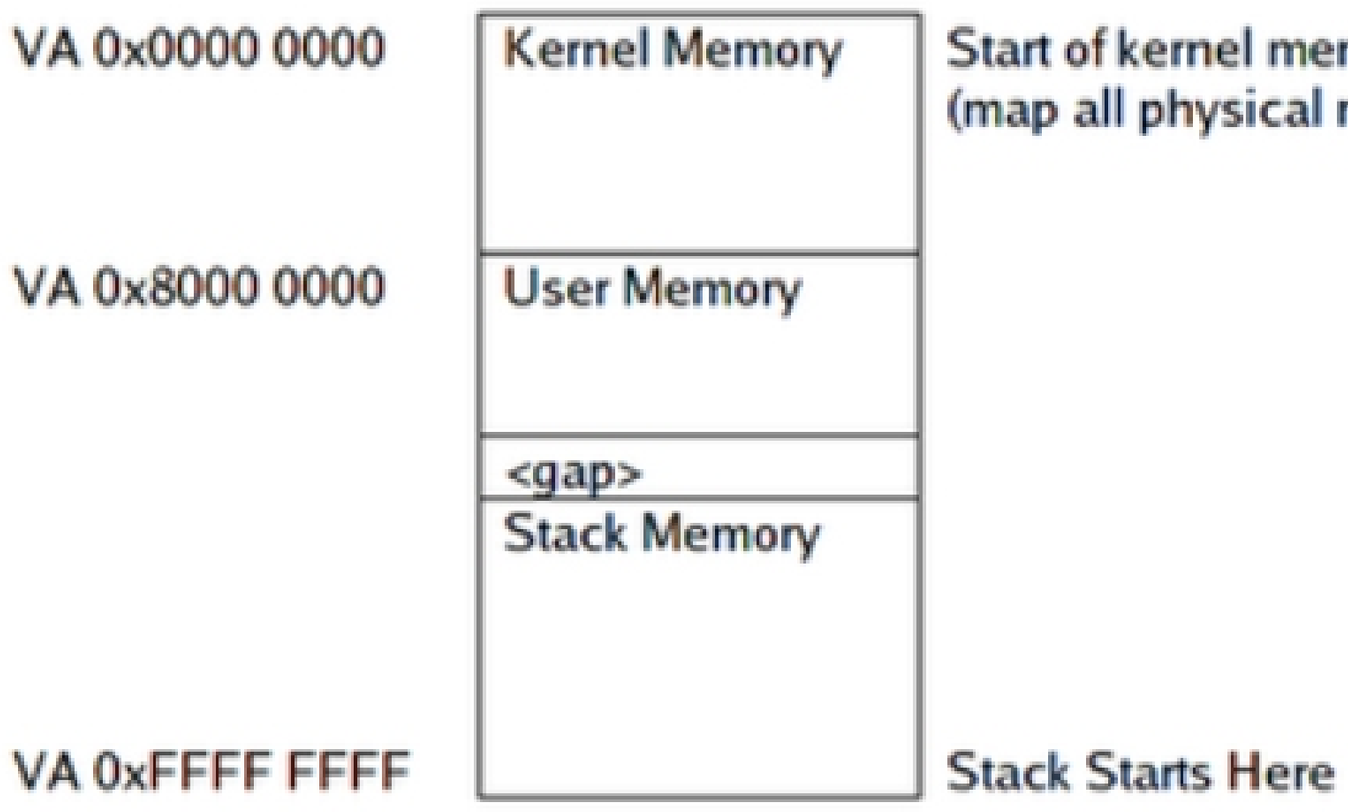

Figure 9.1: Virtual memory layout in GeekOS.

The kernel memory should be a one to one mapping of all of the physical memory in the processor (this limits the physical memory of the processor to 2GB, but this is not a critical limit for this project). The page table entries for this memory should be marked so that this memory is only accessible from kernel mode

(i.e. the

Cause	Indication	Action
Stack growing to new page	Fault is within one page of the current stack limit	Allocate a new page and continue.
Fault for paged out page	Bits in page table indicate page is on disk	Read page from paging device (sector indicated in PTE) and continue.
Fault for invalid address	None of the other conditions apply	Terminate user process

Figure 9.2: Actions to be taken when a page fault occurs.

implement a version of pseudo-LRU. Use the reference bit in the page tables to keep track of how frequently pages are accessed. To do this, add a