**CSE 421/521**

**OPERATING SYSTEMS**

**DESIGN DOCUMENT**

**AUTHORS: SUDHARCHITH SONTY (50169912)**

**SAURAV MOHAPATRA**

**(50101106)**

**ASSIGNMENT 3: VIRTUAL MEMORY**

**PART 1: CORE MAP INITIALIZATION**

The virtual memory subsystem is a Memory Management Technique that maps virtual addresses used by a user process into physical address in computer memory.

**DEFINITION:**

* A Core Map is a data structure that keeps track of the pages that you have.
* All information we need to keep about a page is stored in the Core Map
* The Core Map is of a fixed size
* The Core Map has to be initialized before the other data structures are initialized.
* The number of entries in the Core Map should include the number of physical pages in memory.
* Memory allocation has to be page aligned.

**HOW TO IMPLEMENT**

* The existing call to vm\_bootstrap is very late and leaks memory.
* We need to call vm\_bootstrap() before proc\_bootstrap() and after ram\_bootstrap() is called.
* We have to compute the Core Map size. This can be done by multiplying the size of each entry with the total number of pages.
* Total number of pages can be calculated by dividing the size of the RAM by the individual size of the page.
* Then we need to take the first free address as the start of the core map and the last free address as the end of the core map.
* Since, we are working with only two states of the pages thus far, all the pages in the Core Map must be either FIXED or FREE.
* The allocation has to be dynamic and not static. Fixing the size of the Core Map before the boot is not correct.
* We can use locks as the synchronization primitive for this part.

**FUNCTIONS TO IMPLEMENT:**

**Vm\_bootstrap():**

* In this function we initialize the Core Map .
* We declare the ram size, total number of pages, number of fixed pages, first free address as the start index of the Core Map and the last free address as the end of the core map.
* Size = use ram\_getsize();
* Total number of pages = size/PAGE\_SZIZE:
* First free address = use ram\_getfirstfree()
* Last free address = use ram\_bootstrap()
* Then we initialize the Core Map entries with pages between 0 and first free address being marked as FIXED for the exception handling and the kernel and the pages between free address and the last address as FREE.
* Keep a global flag to check if the VM is bootstrapped and initialize the lock

**ALLOC\_KPAGES():**

This is a function to acquire the pages and allocate them

First we check if the VM is bootstrapped and if it is then we allocate the pages otherwise we return 0;

**FREE\_KPAGES():**

The free page function will only get the address of the first page and will not get how many pages to free.

We have to calculate how many pages to free and mark their state as free from fixed.

**Core Map\_used\_bytes():**

We have to use this function to retrieve how many bytes the Core Map uses. Basically all the pages with FIXED State have to be counted and the number has to be retrieved.