OPERATING SYSTEMS
CS 611, FALL 2021
Virtual Memory Management and Memory Allocation

SIRI NAMBURI
331007601

In this machine problem, we make necessary changes to previous mp3 to upgrade into a full fledged virtual memory allocator. First change to be done is the recursive page table lookup in x86. As we are moving from kernel memory to process memory, this is implemented. Logic is to recursively map page table and directory, i.e, for example, the address of the page directory is stored in the last entry of the page the page directory is in, so that it can access the page fault address in a correct manner.

Different functions to implement Page Table:

init_paging:

kernel and process memory pool ,shared memory are set here.

Constructor:

Here we setup page directory from process memory pool. The implementation is similar to mp3, except everything is setup from process memory pool instead of kernel memory pool. Here we implement the directory by recursive page table lookup

Load:

Loads page directory address to cr3 register, sets current page table pointer to current object of page table.

enable_paging:

Conceptually this starts paging in the machine

Handle_fault

First we check if the page fault address is legitimate. This can be done by seeing if that address corresponds to any of the virtual memory pools present.And it handles page faults. Basically page fault occurs when we are trying to access a page which is not mapped to any frame. This does so by checking if the fault is at page directory level or the page table level. If the fault is at page directory level we assign a new frame to a new page table and then make the page fault address in the new table as present. Similar steps were taken if the fault is found at page table level.

Register_pool

Registers new vm pool to the page table

free_page

Page table frees the pages by this functionality.

Implementation of virtual memory pools

These are somewhat similar to the frame pools we implemented in mp2

Constructor:

sets different variables belonging to the virtual memory pool class like base address, size, frame pool, page table and initialises list to handle allocated regions

Allocate:

This basically allocates required regions of memory in this virtual memory pool. This allocates memory in multiples of pages only. Allocated memory regions are kept track of by a list of struct variables, where each variable consists of base address of allocated memory region and size of the allocated memory region

Release:

This function releases the memory allocated to a region of memory. This is done by searching for that region of memory in the allocated list , freeing the corresponding pages and removing the region from allocated list. At the end page table is loaded again so that tlb will be flushed.

Is_legitimate:

Checks if the input address belongs to this virtual memory pool or not.

Note: if the make clean is run, the ContFramePool object file solution should be copy pasted again to complete the code.