**Project #4 Design**

**Physical memory layout**

Draw the physical memory layout, indicating the areas of memory devoted to code (text area), data, page directories and page tables, BSS, etc. For each memory area, indicate its start address and its size. Note that different teams may use different layouts.

**Initialization of page directories and page tables**

Indicate how page directories and page tables should be initialized. You don’t need to show the exact content of page directories and page tables (i.e., the content of each entry). But, you need to indicate how many page directory/table entries you need for each memory area that should be mapped to the processes’ virtual space. If different types of processes require different mappings, show them all.

In designing the memory layout and defining the initial content of page directories and page tables, you need to consider the following.

* Before paging is enabled, the OS uses only physical addresses. However, after paging is enabled, all memory accesses are through the virtual address space. You need to map the static segments (TEXT, DATA, etc.) in the virtual address space of all processes.
* User processes cannot map the whole FFS area: they can map only the portion of it that they use (that is, FFS area will be mapped only when vmalloc is invoked)
* **If you are taking the course at the graduate level**, in your implementation user processes cannot map the area devoted to page directories and page tables.

**System Initialization**

Where is paging enabled and how? (see hints)

**Process Creation**

How do you need to modify process creation to support paging?

**Process Termination**

How do you need to modify process termination to support paging?

**Context Switch**

What should be done at context switch to support paging?

**Heap allocation, deallocation and access**

What should be done at heap allocation, deallocation and when the heap is accessed? Remember that you need to implement lazy allocation in your code.

**Page Fault Handler Design**

1. In which circumstances will the hardware raise a page fault?
2. What operations should be performed by the page fault handler depending on the circumstances under which it is invoked?