**Design Document**

**NOTE:**

**Only first 2 phases are done but tried to implement phase 3.**

**Please consider this code for 2 phases only.**

**Phase 1:**

In this phase the total number of frames in Main Memory was calculated. It is based in Total memory capacity of Main memory and Page Size. I created a **Free List** of frames. I also calculated frames needed by OS and Idle process and allocated those pages respectively.

Whenever a process is submitted, number of frames required by the process will be calculated. In the free list, first few frames are allocated to the **Page Table** of any process depending on the number of frames required by the process.

I calculated physical address from offset obtained. According to that I copied contents of the process onto main memory and started execution.

Also, dump memory function is implemented according to the paging scheme.

**Phase 2:**

In this phase, I implemented Demand Paging. Here swap space also come into picture. When a process is submitted all of its required frames are loaded onto swap space. Only 1 frame is allocated into Main Memory upon a process submission. And the other frames are allocated on demand by the process.

When a process is submitted, I created a page table for it depending upon number of frames required by that process. But only 1st frame is allocated. When the offset is obtained, I calculated an index and new offset. This index represents the index we have to search for a frame number in the Page table. So, if that entry is 0, it means a page fault.

When a page fault occurs, a new frame from free list is allocated. If its Dirty bit is 1 then, I copied the contents of it into swap space.

Also, there is a periodic interrupt called memory age scan. During this, every frame’s age vector is updated.

I implemented a dump\_swap function, which will dump all the swap related data.

**Phase-3**

In this phase, init\_pagefault\_handler() function will take care of copying contents back and forth of swap space and memory.

I created an IO device thread, which will be waiting on a ready list. When there is a IO related operation (either copying content from Swap to Memory or vice versa), Io thread will take the task of that operation. IO device comes into picture when there is a page fault. When there is a page fault, and if a frame is allocated to that process, I checked whether the dirty bit of it is zero or not. If it is zero, I place required data into a IO queue and active the IO thread. It will take all the data present in the head of the list and it will perform required task.