**Architectural Document**

1. **WHAT is included in my design**

In Project 2, I have finished Test21-Test26 and Test28. I modified the Fault Handler to deal with page faults, allocate frames for a process which tries to access to memory, and do page replacement if needed. I also implemented some system call functions to enable the operating system to define shared area in memory, send a message and receive a message in a process.

Here are the system call functions I have implemented in Project 2:

1. DEFINE\_SHARED\_AREA(StartingAddress, PagesInSharedArea, AreaTag, &NumberPreviousSharers, &ErrorReturned);
2. SEND\_MESSAGE( ProcessID, MessageBuffer, MessageSendLength, &ErrorReturned);
3. RECEIVE\_MESSAGE(SourcePID, MessageBuffer, MessageReceiveLength , &MessageSendLength, &MessageSenderPid , &ErrorReturned);
4. **High Level Design and Justification**
5. Initialization to handle memory access

**Justification:** When a process is created, for each one, we create a **PageTable** and a **PageStatusTable**. The PageTable contains the base address of each page in physical memory. The PageStatusTable contains detailed information about each page of the process. The detailed information includes whether the page has a copy in disk, if so where the copy is(its diskID and sector number), and whether it is a shared page. We store the pointers of both PageTable and PageStatusTable in the PCB of this process.

Also, when the operating system is started, we create a FrameTable to store detailed information about each frame in the physical memory. The information includes, for each frame, whether it is free or not, what is the PID of process if the frame is owned by a process, what is the page number correspond to this frame in PageTable, and whether it is a shared frame.

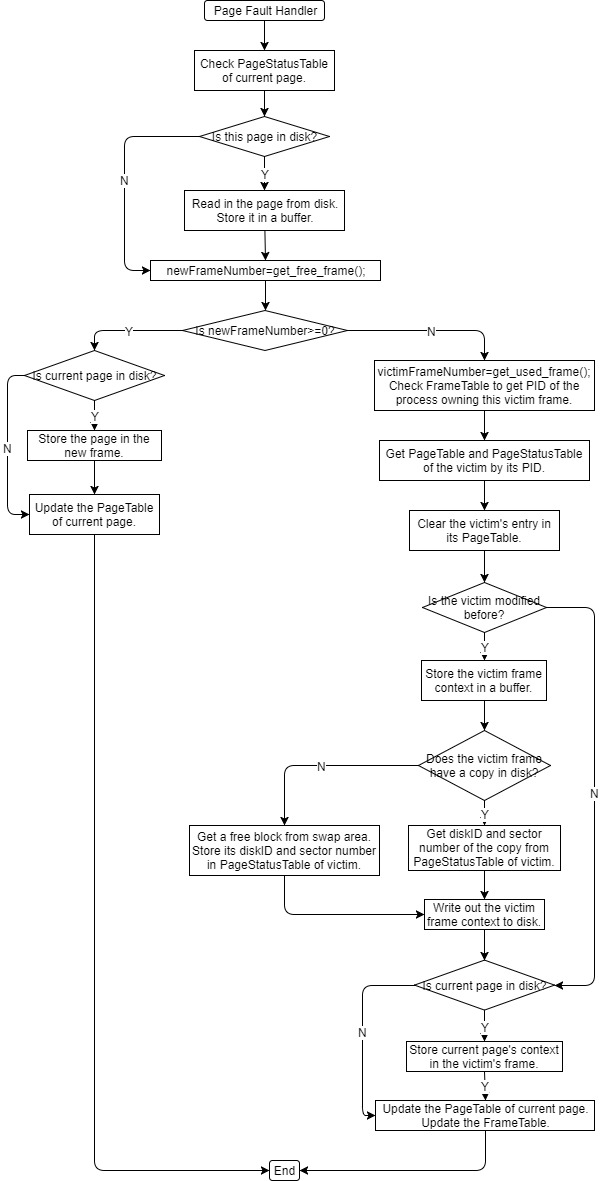
1. Page Fault Handler

**Justification:** Each time when the process tries to access to memory through PageTable but finds a page fault, the Page Fault Handler is triggered to handle it.

The flow chart below shows the procedure how Page Fault Handler works.

Since there are two reasons why a page fault could happen. One is that the process tries to access an address which is not visited before -- a totally new address, so a new void page needs to be created. The other is that the process wants to visit an old page which happens to be not in physical memory but in disk, so we need to retrieve this page from the disk and put it in the memory. In both situations, we need to find some space in memory to store this page whether. The operating system checks the FrameTable to find whether there is a free frame. If so, we store the page retrieved from disk in the free frame, and update the PageTable to record the frame number of this frame.

What should we do if no free frame is available? We need to do a page replacement. According to a page replacement algorithm, we find a victim in physical memory and



swap it out to the disk if it was modified before or this is the first time it is swapped out.Also, we clear the victim’s entry in its PageTable to declare its absence in physical memory. We allocate a free block in disk for the victim, or store it in its copy’s area if the victim was once swapped out. At last, we use the frame of victim to store the page which is accessed by the process, and update the PageTable of process.

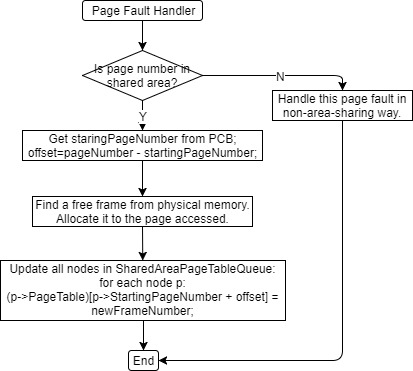
1. Shared Memory Usage
2. Define Shared Area:

**Justification:** To realize shared area, when the operating system is started, it maintains a **SharedAreaPageTableQueue**.

Each time when the process makes a system call to define shared area, the operating system inserts the pointer of the PageTable of the process in the SharedAreaPageTableQueue with other information like startingPageNumber (the starting page number at which the shared area should begin).

Also, the operating system stores the startingPageNumber and pagesInSharedArea (the number of pages that are to be in the shared area) in the PCB of the process.

1. Page Fault Handler for a page fault happening in shared area

****

**Justification:** The flow chart above shows the procedure how page fault handler deals with a page fault happening in shared area.

First, the page number of the page fault is checked whether it falls in a shared area. We can do this by using staringPageNumber and pagesInSharedArea stored in the PCB. If the page number is outside the shared area, the fault handler will treat this page fault just in a non-page-sharing way. Otherwise, the fault handler compute the offset between staringPageNumber and the page number of this page fault. Then it allocates a free frame from physical memory to this page. The fault handler updates all PageTables in SharedAreaPageTableQueue to store the new frame number in all of them. So when another process which also owns this page can find it easily by looking at its PageTable.

1. Send and receive message

**Justification:** Whey a process is going to send a message to another process, it puts the message in a message queue with the PID of source process and target process, and the message length.

When a process is going to receive a message, it checks the message queue to find whether there is a message that is sent to it. If it finds one, it will copies the message to its buffer. Otherwise, the operating system will do a process-switch to suspend the current process (put in the Ready Queue), and at the next time when the process is rerun, it checks the message queue again to find a message sent to it. The process may be suspended repeatedly until it can find its message.

1. **What is UNIQUE about my project**
2. Page Replacement Algorithm

In my design, I use a second-chance FIFO replacement algorithm to find a victim. When a page has been selected, its reference bit is inspected. If the value is 0, the page is going to be replaced. If the reference bit is set to 1, the page can have a second chance, but the reference bit must be cleared, and the algorithm moves on to select the next FIFO page. So the page that get a second chance will not be replaced until all other pages have be replaced (or given second chances).