Scaletta Gioele ID: 811106170

CSCI4730/6730 – Operating Systems

Project #3

General structure of the program

In the submitted program, the two fundamentals functions to implement are the *MMU* function and the *replacement* function. In addition to that also the *is_page_hit* function has been implemented. This part of the program should handle the memory request and the possible page faults that can happen, also calling the replacement function in case of full main memory. This function will call a page replacement algorithm to decide which frame will be the victim frame.

is_page_hit function

This function exploits the *pagetable* structure (2 dimensional array of structure) to check whether the page is present in the main memory and if its bit is valid (it is present) it returns the frame in which it is contained. To check the valid bit it will be enough to use *pid* and *pageNo* as indexes. Moreover in this function we need to check if the request is a write request. In this case it is necessary to set the dirty bit to true (I've chosen to set it as true when the content has to be updated) because that process will write and so the content will change and will need to be updated also in the swap/disk.

MMU function

In this function, in case *is_page_hit* returns a frame, we will just update stats and return the physical address associated with that frame. If there is no hit (so there is a page fault) then we should associate this page with a free frame (if present). The number of free frames available are recorded using a counter and when this counter (*freeFrame*) is not smaller than *NUM_FRAMES* anymore then the function should handle the situation calling the page replacement function. The page replacement function will return a victim frame. In the last part of the function we need to update the pagetable. Swap back to memory the content of the victim frame if this frame content has changed (we will know this by checking the dirty bit). Then swap to main memory the new content of the frame (the page of the request). Other updates include setting the new frame in the page table, setting *pid* and *pageNo* in the frame structure (we need this structure to identify to which *pid* and *pageNo* the frame returned by *page replacement* was associated with) and update the dirty bit to false, if it is a read request, or to true, if it is a write request.

replacement function

This function is just a wrapper for the two function implementing the replacement algorithms. I have decided to implement those algorithms creating the list in the pagetable.c file so that the list (to choose the victim frame we use a double liked list) can be updated in the MMU function. This update will happen in case of a page hit (needed only for lru) and in case of insertion of content in a still available frame (a frame has to be added to the double linked list in this case). This is the reason why in my implementation it is needed to pass a pointer to the list as an argument of the replacement function and then as an argument of fifo and lru functions.

fifo and lru function

The *fifo* and *lru* function will be very similar in my implementation and their task will be to remove the first frame in the list and return it. Then update the list by inserting this frame in the tail of the list. The difference between those two functions will consist in the modification of the list that will occur for page hit (and only in case of *lru*). In this case the algorithm will remove from the list the hit frame and add it again on the head of the list (it will be the most recently used).

Conclusion

The above described implementation of the project allows us to translate the virtual memory address into a physical address. So that it will be possible to store more memory than the physical memory in the virtual memory, improving performances. To do this it is necessary to handle page faults also using page replaments algorithms.