Michael Weaver

CST – 221

John Zupan

2/11/18

GitHub Link: <https://github.com/battousairurik/CST-221>

**Memory Management**

1. The MMU organizes virtual memory into logical address space and physical address into physical address space. The MMU uses paging techniques, which utilize a base register, to dynamically link virtual memory to physical memory based on individual processes. Paging systems such as; demand paging, segmentation systems, or demand segmentation, utilize a reference string to record the address of a memory reference. When a process calls on a page for reference, the TLB is checked for a page reference, then main memory is checked for the page reference, and finally the page is retrieved from the disk if all else fails.

Pseudocode

*Read in process page request, retrieve base register, determine physical memory address*

*Check TLB, then main memory, if both miss trigger page fault*

*Determine if invalid reference of genuine page fault*

*Check other tables to see if reference exists*

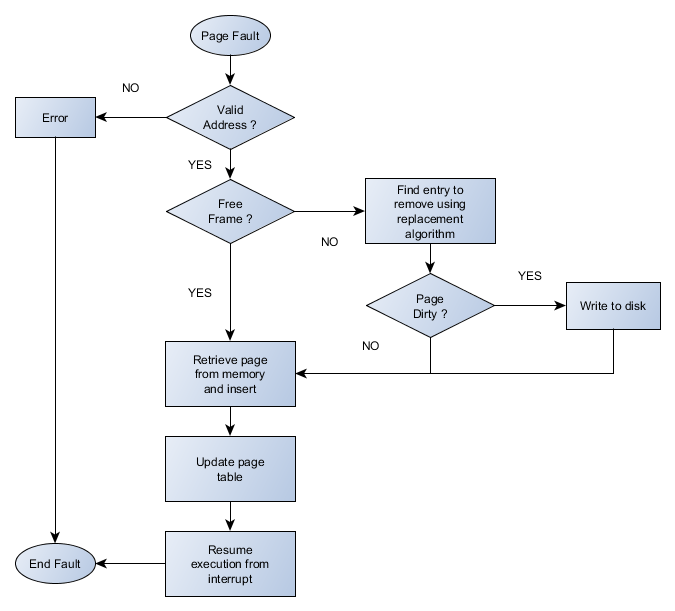
*Secure empty frame, swap out old page if necessary*

*Swap in new frame, reset tables*

*While OS is retrieving from disk, CPU can be allocated to other programs*

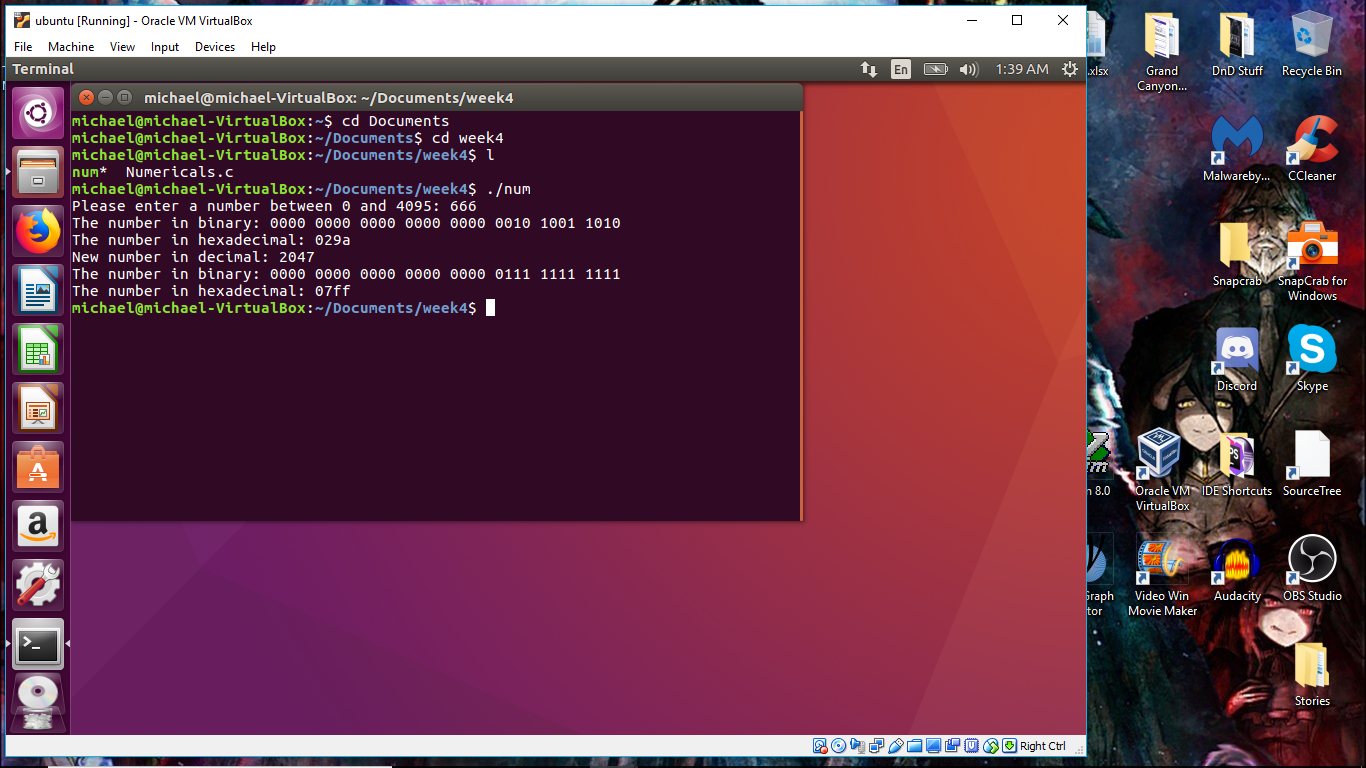
*Interrupt, save program 2, Restart process 1 that caused fault*

Flowchart



1. Page fault handling is used when a process references a page that is not in main memory and it must be fetched from disk. The MMU determines if there is a free frame, clears a frame page determined by its replacement algorithm and then fetches the intended page from disk. A few examples of replacement algorithms are FIFO, Optimal Page, and Least Recently Used. The new page is then inserted into the frame, the table and TLB updated, then the process restarted from the point it threw the page fault error. To save time and processing power, while the OS retrieves from disk other programs may be run.
2. The separation of Policy and Mechanism is known as Abstraction in broader terms. Virtual memory only exists because of this abstraction. When a virtual memory address is referenced, the exact physical address of the data is not at all known. The reference string is passed to the MMU as a parameter (most likely) and the process knows absolutely nothing about how the MMU will use the reference string. The MMU then references the page to its own tables to determine if it is in main memory, a page fault occurs when it is not. The Page fault handler then passes the page reference to the Operating system in a similar manner, most likely as a parameter, given that the Page Fault Handler has absolutely no idea how the External Pager Function will handle the page fault. The external pager function then retrieves the page from disk and passes the page back to the PFH to be placed in main memory, though the MMU has no idea where the page is stored or what is contained on the page.

Program Execution



Analysis

The program was much more straightforward than I would have thought. It took a bit of digging to uncover the unsigned int fields and the bitwise operators, but they streamlined the most difficult part of the assignment. Everything went rather smoothly this time around.

I did notice that there is no output log required, if this is not the case then please message me on the individual forum so I can adjust the program and generate an output log so I do not loose points.

References

8BitAvenue. (2016). *Policy vs mechanism in operating system*. Retrieved from: <http://www.8bitavenue.com/2016/12/policy-vs-mechanism-in-operating-system/>

Tutorialspoint. (2018). *Operating System - Virtual Memory*. Retrieved from: <https://www.tutorialspoint.com/operating_system/os_virtual_memory.htm>

Tutorialspoint. (2018). Operating System - Memory Management. Retrieved from <https://www.tutorialspoint.com/operating_system/os_memory_management.htm>

123. (2011). *Clear lower 16 bits*. Retrieved from <https://stackoverflow.com/questions/6867489/clear-lower-16-bits>

Tutorialspoint. (2018). *Bitwise Operators in C*. Retrieved from <https://www.tutorialspoint.com/cprogramming/c_bitwise_operators.htm>

Includehelp.com. (2018). *Working with Hexadecimal values in C programming language*. Retrieved from <http://www.includehelp.com/c/working-with-hexadecimal-values-in-c-programming-language.aspx>