Chad Weirick

CST-221

GCU

John Zupan

6/14/19

**MMU Overview**

The MMU acts as a table which maps the physical memory in a machine to the virtual memory space allocated for a program. In this way when a program calls memory it views as a single contiguous block, it is being routed memory from various physical locations. The physical locations may not necessarily be contiguous based on memory needs and allocations and the order in which they happened. For example, program A requests 10 pages of memory and gets it, then program B requests 10 pages and gets the next 10 physical pages. Program A makes a request for 10 more, and gets the 10 after B, but it does not see B. From the program’s perspective, it has 20 contiguous blocks of memory.

An internal table kept by the MMU helps determine what physical memory is allocated to which process. This includes page files and other types of system swaps. This is covered further in this document in regards to MMU policy and mechanism.

**MMU Psuedo-Code**

Program A requests 10 pages.

MMO returns the pointers to the start of the memory page blocks.

Program loads 10 pages of memory into memory.

As each page is loaded, it is sent to the MMU’s memory map.

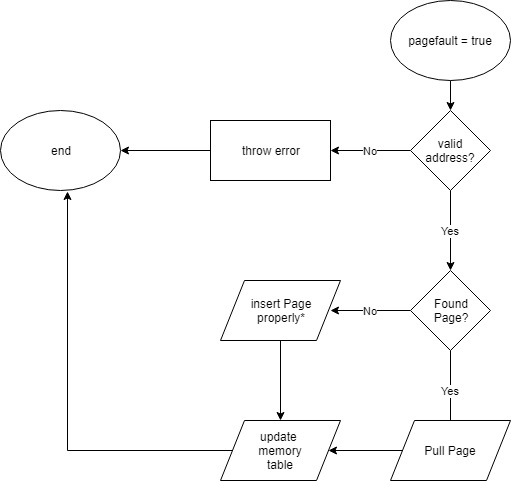
The MMU takes the write request and routes it to the correct physical memory page.

A request for one of those pages is made

TLB check is made, if the TLB is hit then it pulls from TLB; if TLB is a miss, then it goes to the MMU. If for some reason the memory has already swapped out, then a page fault occurs and swap file is engaged for the read.

In the event of a swap file access, the MMU needs to be engaged to determine if another program needs to/can be swapped out to disk.

**Page Fault Flowchart**



The process begins with a page fault happening. The first thing that needs to be checked is whether or not the address is valid. If the address is invalid, the process should throw an error to a log/or another appropriate source and then end abruptly.

If the address is valid, the next thing is to determine if the page is found in the caching mechanism of the MMU. If not, then the page needs to be inserted. This will include checking the memory to ensure it is not already dirty/tainted. If the page is already in memory, retrieve it.

Regardless of the previous action, the memory table should be updated to reflect any changes that need to be noted. When done, the page fault can complete. This will result in a resume from the interrupted state in most modern hardware.

**VMM Policy and Mechanism Separation**

The mechanism is how it is executed in terms of authorization/enforcement while the policy is a mechanism to determine the best course of action. The abstraction between the policy and the actual execution mechanism is essential for allowing modern computing devices to handle multiple very large programs. In fact, the entire concept of a swap file relies on this abstraction because it is possible to allow for a larger virtual memory pool than physical memory, so long as there is some other medium such as a hard disk or solid-state drive to acts as a lower-level pool to read and write from.

**C Program**

The C Program for this assignment can be found at the following GitHub Link: <https://github.com/chadatgcu/cst221.git> and then access the week 4 folder.

**Notes**

Note that the class is a class in C, but the Rubric specifies that this should be written in JAVA.

Also, I know there is a bug with the shifted decimal form not showing. It seems to be a logic error, but I am not seeing where it originates from at this time.