Memory Management: Page Fault Algorithm

Daniel R. Cender

Grand Canyon University: CST-221-O500

February 1, 2020

**Handling Page Faults**

**Diagram**

*Figure 1.* Diagram of how the OS kernel handles a page fault while translating virtual addresses.

A close up of a map

Description automatically generated

**Explanation**

When a memory address passed to the MMU (Memory Management Unit) is not found in the TLB (Translation Lookaside Buffer) or the page table, a page fault is generated and the MMU traps to the kernel. An assembly-language subroutine saves general registers and other volatile information to the disk, so the operating system doesn’t overwrite it (Tanenbaum and Bos, 2015). An initial check ensures the address maps to a real space in addressable memory. If it’s an invalid address, the kernel generates a *segmentation fault* to kill the calling program. At that point, it’s safe to assume that there is an error in the program itself, at no fault of the hardware or operating system.

Once the operating system knows the address is valid and won’t break protection clauses, it checks for any free page frames. If there are none, a page frame replacement algorithm is run to select a page to replace. If the selected page is dirty, the kernel schedules to write it to the disk and the faulting process is suspended, so another may run while the page transfer runs. Once the page is “clean,” having either been free all along or just finished its transfer, the kernel loads it into the page table (Tanenbaum and Bos, 2015).

Now that the frame is fully loaded, the disk creates an interrupt signal and the page table is updated. Once the last-run instruction is backed up to its state before the fault, the process is scheduled for execution. The kernel returns to the assembly routine, which loads back all the registers and volatile data from the disk. The process can now run its last instruction again as though the page fault had never occurred.

References

Tanenbaum, A.S. & Bos, H. (2015).*Modern Operating Systems.*Chapter 3.