CS241 - Virtual Memory

This week we are going to be dealing with the MMU and TLB. We are going to explore what Address Translation looks like and some pitfalls.

Translate Address - Drawing

Assume that we have the following binary address 100011. We have a six bit system with two level page tables (That mean two page tables). Tell me what the first level page table is. What about the second level?

1	0	0	0	1	1

What is the offset?

How much space does the entire thing take up assuming 1 byte entries?

Translate Address - Bit Shifting

Let's walk through an involved example – assuming that we have frame sizes of 4096 bytes on a 64 Bit machine with a quadruple level page table (Yes, Intel now supports those).

How many bits are in the VPNs? What about the Offset?

How big do the page tables need to be assuming 8 Byte Entries?

What is the Offset and VPN3 of 0xFABDECAFBEADDAB2?

Translate Address - Page Faults

What are the types of page faults? What do each of them generate in the operating system?

Food for thought

It is completely valid for two processes to have the same virtual addresses, how could the MMU ensure that the TLB has valid cache for a single core machine (So that when P2 requests a memory address, it doesn't get P1's value - that would be a break of security!)

We know that in the age of multi-core machines that is not possible, how do we do it now? (Hint look up the x86 instruction INVLPG) What does a kernel do when context switching?