Chapter 19 Paging: Faster Translations

- Using paging as the core mechanism to support virtual memory can lead to high performance overheads.
    - Paging requires a large amount of mapping information.
- Translation Lookaside Buffer.
    - Used to speed up address translation.
    - Part of the chips MMU.
    - A hardware cache of popular virtual to physical address translations.
    - Consulted before a page table to see if the address is found.

19.1 TLB Basic Algorithm

- First extract the virtual page number from the virtual address.
    - Check if the TLB holds the translation for the VPN
        - If yes then TLB hit.
        - If no then TLB miss.
- In a miss access the page table to find the translation.
    - Assuming the virtual memory accessed is valid and accessible update the TLB.

19.2 Example: Accessing an Array.

- Assume an array of 10 4-byte integers in memory.
    - start at virtual address 100.
- Assume 8-bit address space with 16 pages.
- Consider a single loop that accesses each array element.
    - The loop will hit consecutive entries that are on the same memory page with the TLB.

    - When the next element is in a different page the TLB will miss.
    - The TLB improves performance due to spatial locality.
    - If we accessed the array a second time Temporal locality would increase the TLB hits to all of the entries.

Handling TLB Misses

- There are two options for what handles the TLB misses.
    - Hardware or software.
- Historically the hardware had complex instruction sets CISC
    - The hardware would handle the TLB miss entirely.
    - The hardware had to know exactly where the page tables are located in meemory
        - done via a page-table base register.
    - Also had to know their exact format.
    - On a miss the hardware would walk the page table and find the correct page table entry.
        - Update the TLB and retry the instruction.
    - Intel x86 architecture uses a fixed multi-level page table.
- Hardware with a RISC instruction set use software managed TLB
    - On a TLB the hardware raises an exception
    - pauses the instruction stream.
    - Raises privilege level to kernel
    - Jumps to a trap handler.
        - The return from trap instruction is a little different for system calls in this instance.
        - The return to execution is at the instruction that caused the miss not after.
    - The OS has to be careful about causing an infinite sequence of TLB misses to occur.
        - TLB miss handles could be kept in physical memory.

            - they would be unmapped and not subject to translation.
        - Reserve entries in the TLB for permanently valid translations.
            - Called wired translations and always hit.
    - A primary advantage of software-managed approach is flexibility.
        - The OS can use any data structure it wants to manage the page table.
    - A second advantage is simplicity.
        - The hardware doesn't have to do much on a miss.

19.4 TLB Contents

- Contents of the hardware TLB
    - A typical TLB may have 32, 64, or 128 entries.
    - Called fully associative
        - Any given translation can be anywhere in the TLB
        - The hardware will search the TLB in parallel to find the desired translation.

19.5 Context Switches

- A new issues come up when context switching with TLB's
    - The TLB entries are only valid for the currently running process.
    - The hardware or OS must ensure the new process doesn't use the previous process TLB
- Solutions:
    - One solution is to flush the TLB on context switches.
        - On software-based systems done with a hardware instruction.

        - On hardware based systems enacted when the page-table base register is switched.
        - By flushing the TLB a cost is incurred.
            - the first instruction will always be a TLB miss.
            - If context switching occurs frequently this may be a high cost.
        - To reduce the cost some systems add hardware support.
            - enable sharing of the TLB across context switches.
            - Provide an address space identifier field.
                - similar to a process id but with fewer bits.

19.6 Replacement Policy

- Cache replacement.
    - When we are installing a new entry in the TLB an old one has to be replaced.
- Common approaches
    - leased-recently-used LRU
        - Tries to take advantage of locality in the memory-reference stream.
        - Assumes an entry that hasn't been recently used is a good candidate for removal.
    - Randoom policy
        - Picks an entry at random to remove.
            - Useful to to simplicity.
            - avoids corner case behaviors.