Lecture Topics: 11/19

- Paging
  - Page tables
  - Memory protection, validation
  - Multilevel paging

Address Translation

- The user process has one view of memory: its virtual address space
- In reality, the physical addresses of the process are entirely different
  - maybe not contiguous
  - maybe out of order
- The address translation hardware (the MMU) must reconcile these two views at execution time

Per-process Translation

- Each process has its own virtual address space
- The hardware must know how to translate addresses for *this process*
- Context switches must now involve
  - register contents
  - program counter
  - memory management

Data Structures for Paging

- Page tables are too large to be kept on the chip (millions of entries)
- Instead, the page tables are kept in memory
- A special register, the *page table base register (PTBR)*, points to the beginning of the page table for this process
- The contents of this register must be changed during a context switch

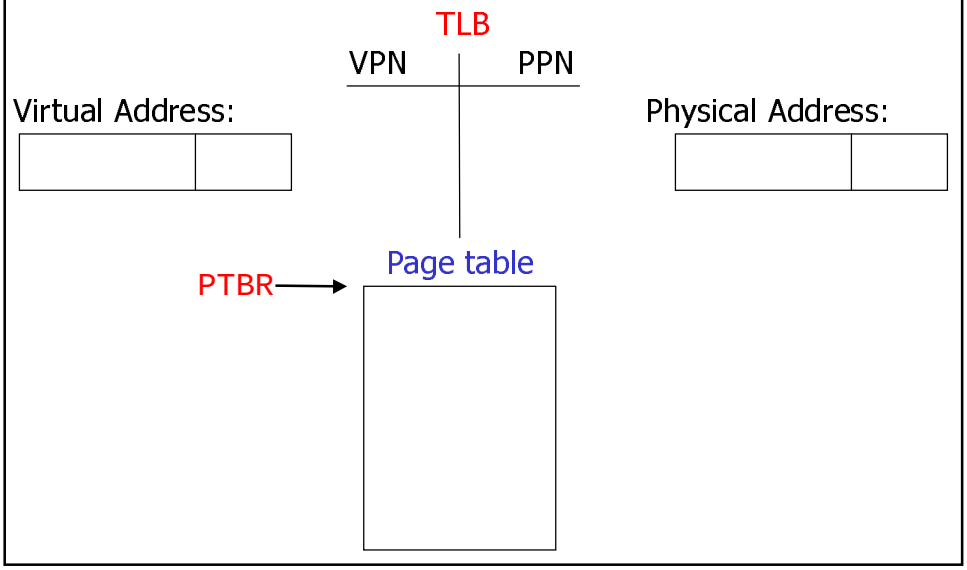
Memory Access Time

- Our new procedure for accessing memory from a user process:
  - Send the virtual address of the memory location to the MMU
  - Translate the virtual page number to a physical page number by looking it up in the page table
  - Send the physical address (physical page number + offset) to the memory system
- Twice the access time!

Translation Lookaside Buffer

- The problem: translating addresses by looking in memory is slow
- The solution: cache translations
- The implementation: a small hardware cache called a TLB (*Translation Lookaside Buffer*)
- Note that this is different from caching the memory contents themselves

TLB Example



Page Table Entries

- The main purpose of a page table entry is to map a virtual page to a physical page frame
- We can piggyback other information:
  - Is this page read-only, read-write, none?
  - Is this page currently valid?
- This info propagates to the TLB as well
- Now, relatively flexible memory protection can be defined, but delivered at hardware speeds