Virtual Memory

• It's just another level in the hierarchy

Virtual memory is the name of the technique that allows us to view main memory as a part of a larger memory space (on disk).

Virtual Memory

• is just cacheing, but uses different terminology
  cache VM
  block
  cache miss page fault
  address
  index physical address (sort of)

Virtual Memory

• What happens if another program in the processor uses the same addresses that yours does?

• What happens if your program uses addresses that don’t exist in the machine?

• What happens to “...” in the address space your program uses?

• So, virtual memory provides
  – protection
  – ease of programming/compilation
  – use of memory
Virtual Memory

• is just a mapping function from memory addresses to memory locations, which allows caching of virtual pages in physical memory.

What makes VM different than memory caches

• MUCH higher miss penalty (millions of cycles)!
• Therefore
  – large pages [equivalent of cache line] (4 KB to MBs)
  – associative mapping of pages (typically fully associative)
  – software handling of misses (but not hits!!)
  – write-through not an option, only write-back

Virtual Memory mapping

Address translation via the page table

• all page mappings are in the page table, so hit/miss is determined solely by the valid bit (i.e., no tag)
• so why is this fully associative???
Making Address Translation Fast

- A translation lookaside buffer for address translations: translation lookaside buffer

Virtual Memory Key Points

- How does virtual memory provide:
  - protection?
  - sharing?
  - performance?
  - illusion of large main memory?

- Virtual Memory requires twice as many memory accesses, so we cache page table entries in the TLB.

- Three things can go wrong on a memory access: cache miss, TLB miss, page fault.