CSE 490/590 Computer Architecture

Virtual Memory I

Steve Ko
Computer Sciences and Engineering
University at Buffalo

Last time…

- Dynamic address translation
  - – Base and bound registers
  - – Memory fragmentation problem
- Paged memory
  - – Pages form an entire program
  - – Uses page tables
- Demand paging
  - – Hardware-assisted page swapping

Modern Virtual Memory Systems

Illusion of a large, private, uniform store

Protection & Privacy

several users, each with their private address space and one or more shared address spaces

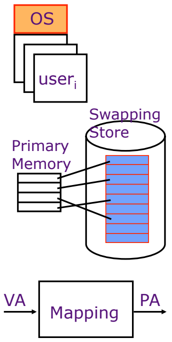
page table ≡ name space

Demand Paging

Provides the ability to run programs larger than the primary memory

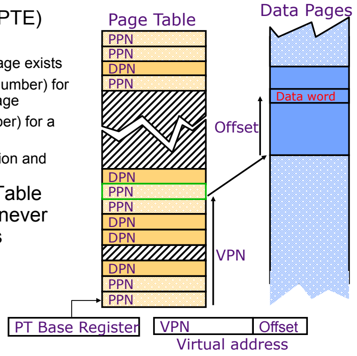
Hides differences in machine configurations

The price is address translation on each memory reference



Linear Page Table

- Page Table Entry (PTE) contains:
  - – A bit to indicate if a page exists
  - – PPN (physical page number) for a memory-resident page
  - – DPN (disk page number) for a page on the disk
  - – Status bits for protection and usage
- OS sets the Page Table Base Register whenever active user process changes



Size of Linear Page Table

With 32-bit addresses, 4-KB pages & 4-byte PTEs:

⇒ 2^{20} PTEs, i.e, 4 MB page table per user

⇒ 4 GB of swap needed to back up full virtual address space

Larger pages?

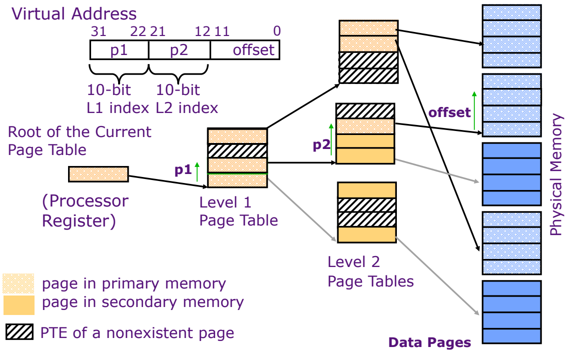
- Internal fragmentation (Not all memory in page is used)
- Larger page fault penalty (more time to read from disk)

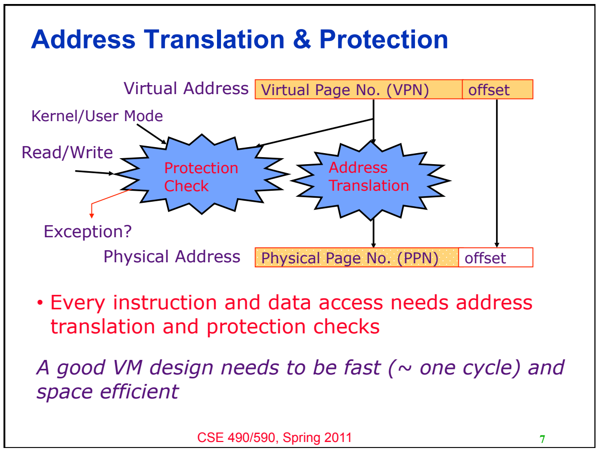
What about 64-bit virtual address space???

- Even 1MB pages would require 2^{44} 8-byte PTEs (35 TB!)

What is the "saving grace" ?

Hierarchical Page Table

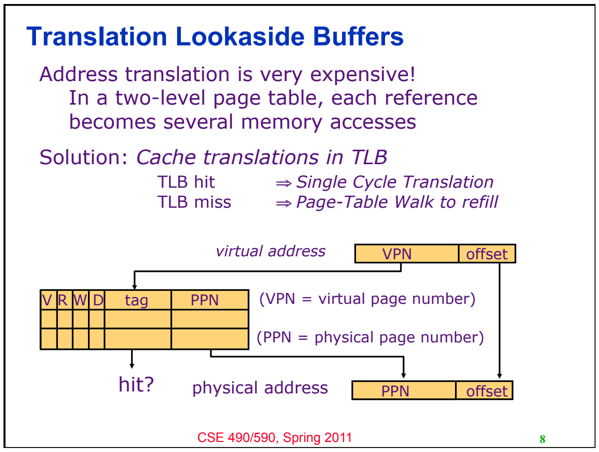




Address Translation & Protection

- Every instruction and data access needs address translation and protection checks

A good VM design needs to be fast (~ one cycle) and space efficient

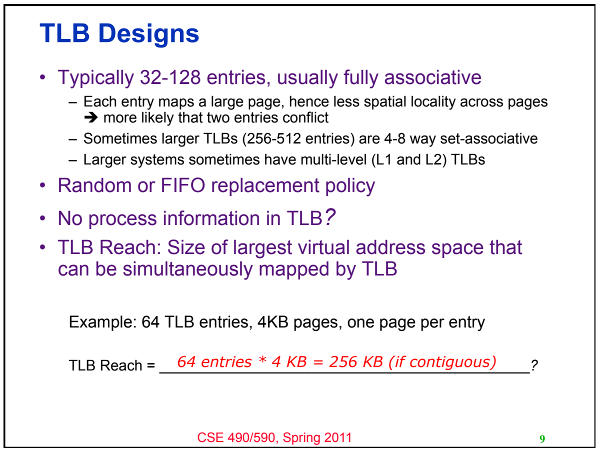


Translation Lookaside Buffers

Address translation is very expensive!
In a two-level page table, each reference becomes several memory accesses

Solution: *Cache translations in TLB*

TLB hit ⇒ *Single Cycle Translation*
TLB miss ⇒ *Page-Table Walk to refill*

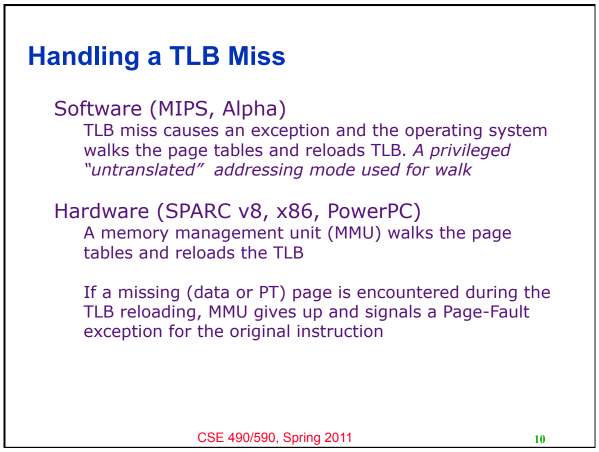


TLB Designs

- Typically 32-128 entries, usually fully associative
  - Each entry maps a large page, hence less spatial locality across pages ➔ more likely that two entries conflict
  - Sometimes larger TLBs (256-512 entries) are 4-8 way set-associative
  - Larger systems sometimes have multi-level (L1 and L2) TLBs
- Random or FIFO replacement policy
- No process information in TLB?
- TLB Reach: Size of largest virtual address space that can be simultaneously mapped by TLB

Example: 64 TLB entries, 4KB pages, one page per entry

TLB Reach = *64 entries * 4 KB = 256 KB (if contiguous)*?

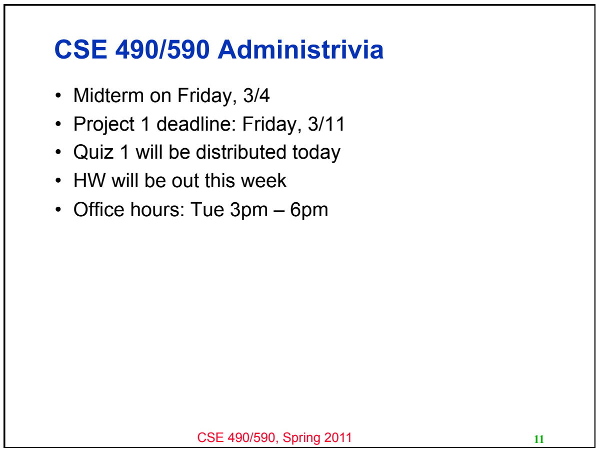


Handling a TLB Miss

Software (MIPS, Alpha)
TLB miss causes an exception and the operating system walks the page tables and reloads TLB. *A privileged "untranslated" addressing mode used for walk*

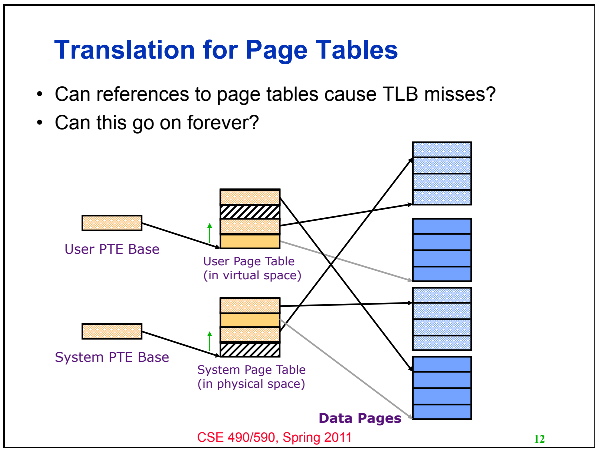
Hardware (SPARC v8, x86, PowerPC)
A memory management unit (MMU) walks the page tables and reloads the TLB

If a missing (data or PT) page is encountered during the TLB reloading, MMU gives up and signals a Page-Fault exception for the original instruction



CSE 490/590 Administrivia

- Midterm on Friday, 3/4
- Project 1 deadline: Friday, 3/11
- Quiz 1 will be distributed today
- HW will be out this week
- Office hours: Tue 3pm – 6pm

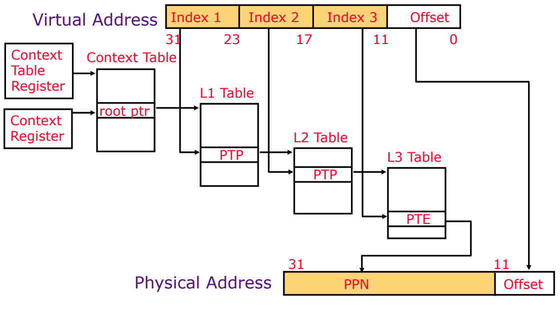


Translation for Page Tables

- Can references to page tables cause TLB misses?
- Can this go on forever?

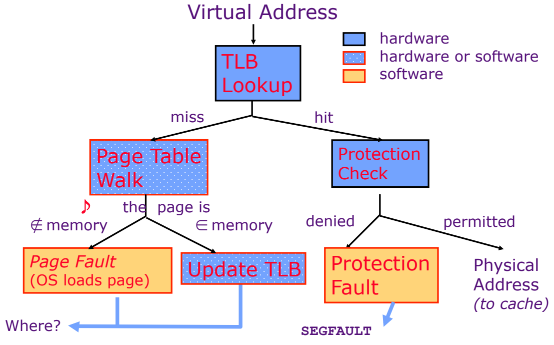
User PTE Base
User Page Table (in virtual space)
System PTE Base
System Page Table (in physical space)
Data Pages

Hierarchical Page Table Walk: SPARC v8

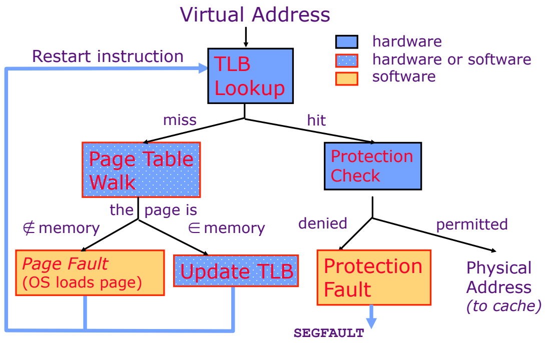


MMU does this table walk in hardware on a TLB miss

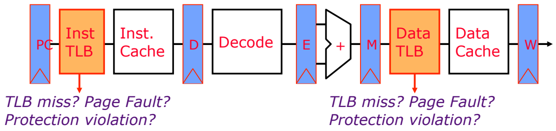
Address Translation: *putting it all together*



Address Translation: *putting it all together*



Address Translation in CPU Pipeline


PC
Inst TLB
Inst. Cache
D
Decode
E
+
M
Data TLB
Data Cache
W
TLB miss? Page Fault? Protection violation?
TLB miss? Page Fault? Protection violation?

- Software handlers need *restartable* exception on page fault or protection violation
- Handling a TLB miss needs a *hardware* or *software* mechanism to refill TLB
- Need mechanisms to cope with the additional latency of a TLB:
  - slow down the clock
  - pipeline the TLB and cache access
  - virtual address caches
  - parallel TLB/cache access

Acknowledgements

- These slides heavily contain material developed and copyright by
  - Krste Asanovic (MIT/UCB)
  - David Patterson (UCB)
- And also by:
  - Arvind (MIT)
  - Joel Emer (Intel/MIT)
  - James Hoe (CMU)
  - John Kubiatowicz (UCB)
- MIT material derived from course 6.823
- UCB material derived from course CS252