inst.eecs.berkeley.edu/~cs61c

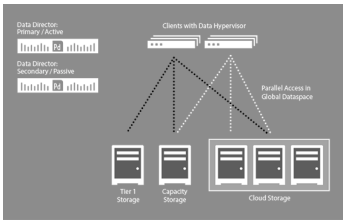
UCB CS61C : Machine Structures

Lecture 34 – Virtual Memory II

11/19/2014:
Primary Data – A new startup came out of stealth mode. The Los Altos company raised about $63 million in funding to help in their plan to untrap data from traditional storage, much like server virtualization decoupled computing from servers in the data center. It does this through an appliance placed in the data center and software deployed in a customer's IT infrastructure.

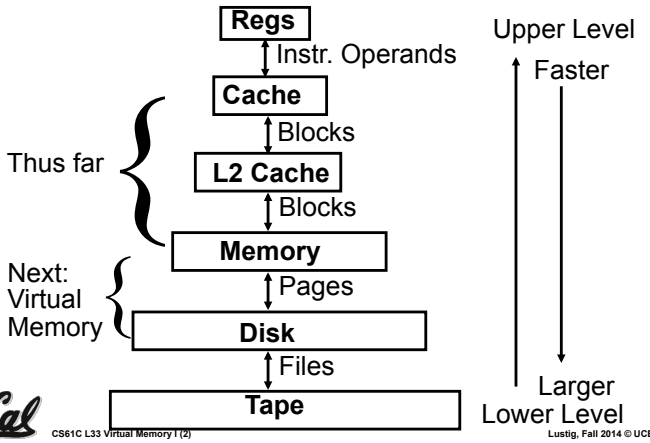
http://www.zdnet.com/can-wozniak-reshape-the-data-center-7000036018/

DATA VIRTUALIZATION



A powerful metadata server provides a logical abstraction of physical storage, while the policy engine intelligently automates data movement and placement to enable the vision of the software-defined datacenter: Simple, cost-effective, scalable performance.

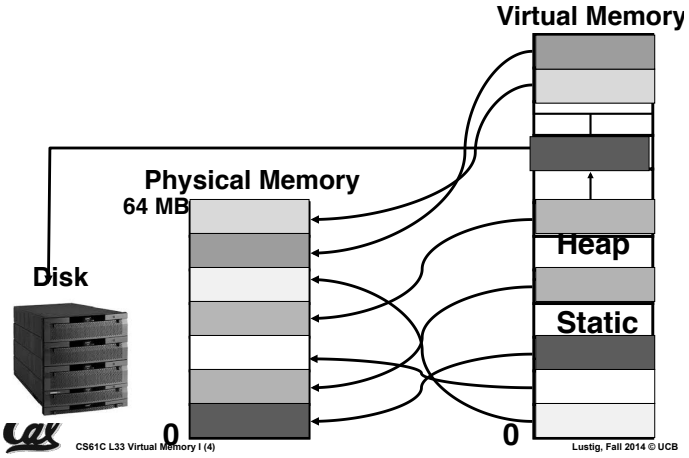
Another View of the Memory Hierarchy



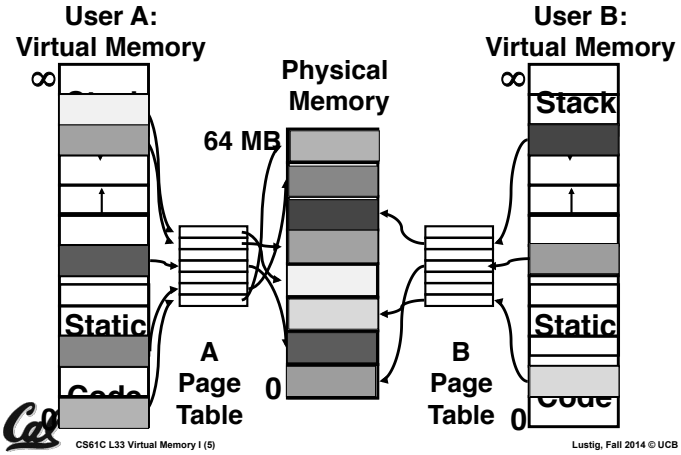
Virtual Memory

- Fully associative Cache for the disk
  - Provides program with illusion of a very large main memory:
  - Working set of "pages" reside in main memory - others reside on disk.
- Enables OS to share memory, protect programs from each other
- Today, more important for protection vs. just another level of memory hierarchy
- Each process thinks it has all the memory to itself

Mapping Virtual Memory to Physical Memory



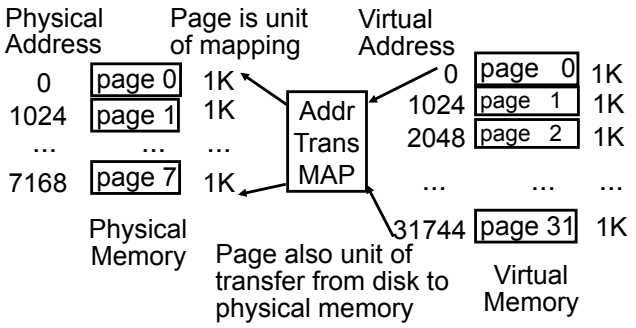
Paging/Virtual Memory Multiple Processes



Review: Paging Terminology

- Programs use *virtual addresses (VAs)*
  - Space of all virtual addresses called *virtual memory (VM)*
  - Divided into pages indexed by *virtual page number (VPN)*
- Main memory indexed by *physical addresses (PAs)*
  - Space of all physical addresses called *physical memory (PM)*
  - Divided into pages indexed by *physical page number (PPN)*

Paging Organization (assume 1 KB pages)



Virtual Memory Mapping Function

- How large is main memory? Disk?
  - Don't know! Designed to be interchangeable components
  - Need a system that works regardless of sizes
- Use lookup table (*page table*) to deal with arbitrary mapping
  - Index lookup table by # of pages in VM (not all entries will be used/valid)
  - Size of PM will affect size of stored translation

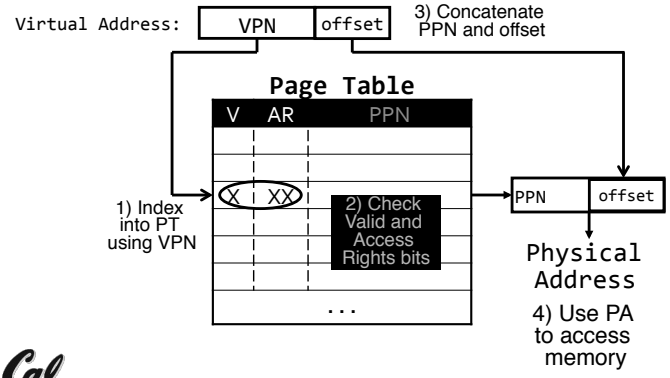
Address Mapping: Page Table

- **Page Table functionality:**
  - Incoming request is Virtual Address (VA), want Physical Address (PA)
  - Physical Offset = Virtual Offset (page-aligned)
  - So just swap Virtual Page Number (VPN) for Physical Page Number (PPN)

Physical Page #	Virtual Page #	Page offset

- **Implementation?**
  - Use VPN as index into PT
  - Store PPN and management bits (Valid, Access Rights)
  - Does NOT store actual data (the data sits in PM)

Page Table Layout

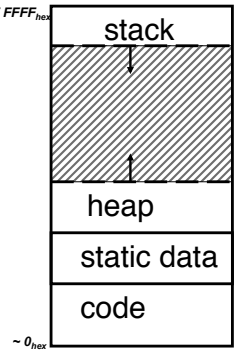


Notes on Page Table

- Solves Fragmentation problem: all chunks same size, so all holes can be used
- OS must reserve "Swap Space" on disk for each process
- To grow a process, ask Operating System
  - If unused pages, OS uses them first
  - If not, OS swaps some old pages to disk
  - (Least Recently Used to pick pages to swap)
- Each process has own Page Table
- Will add details, but Page Table is essence of Virtual Memory

Why would a process need to "grow"?

- A program's *address space* contains 4 regions:
  - stack: local variables, grows downward
  - heap: space requested for pointers via `malloc()`; resizes dynamically, grows upward
  - static data: variables declared outside main, does not grow or shrink
  - code: loaded when program starts, does not change



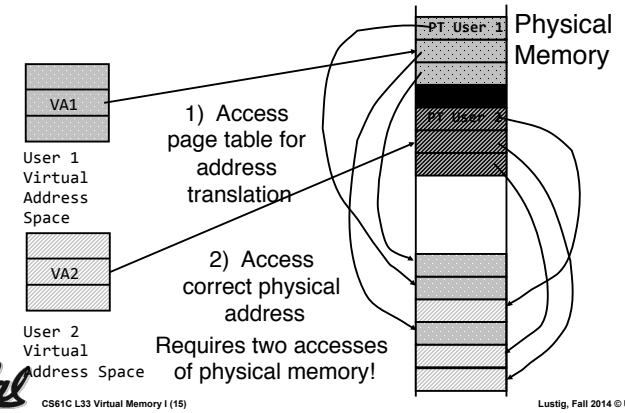
For now, OS somehow prevents accesses between stack and heap (gray hash lines).

Question: How many bits wide are the following fields?

- 16 KiB pages
- 40-bit virtual addresses
- 64 GiB physical memory

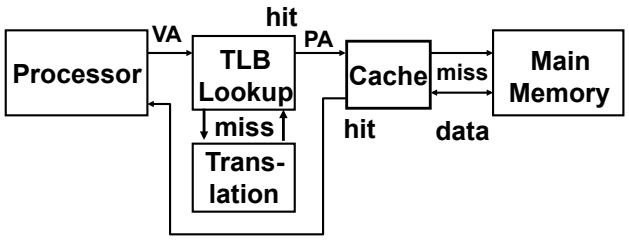
	VPN	PPN
A)	26	26
B)	24	20
C)	22	22
D)	26	22

Retrieving Data from Memory



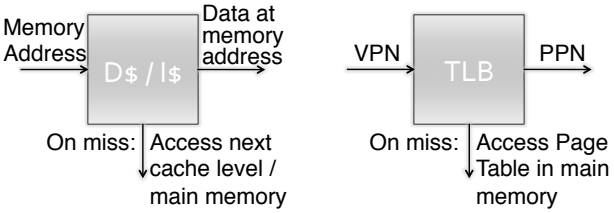
Translation Look-Aside Buffers (TLBs)

- TLBs usually small, typically 128 - 256 entries
- Like any other cache, the TLB can be direct mapped, set associative, or fully associative



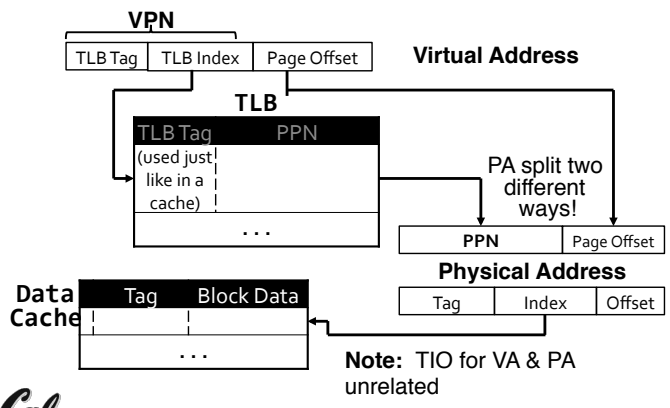
Can cache hold requested data if corresponding page is not in physical memory? No!

TLBs vs. Caches



- TLBs usually small, typically 16 – 512 entries
- TLB access time comparable to cache (« main memory)
- TLBs can have associativity
  - Usually fully/highly associative

Address Translation Using TLB



Note: TIO for VA & PA unrelated

Question: How many bits wide are the following?

- 16 KiB pages
- 40-bit virtual addresses
- 64 GiB physical memory
- 2-way set associative TLB with 512 entries

Valid	Dirty	Ref	Access Rights	TLB Tag	PPN
X	X	X	XX		

	TLB Tag	TLB Index	TLB Entry
A)	12	14	38
B)	18	8	45
C)	14	12	40
D)	17	9	43

Fetching Data on a Memory Read

1) Check TLB (input: VPN, output: PPN)
   - *TLB Hit:* Fetch translation, return PPN
   - *TLB Miss:* Check page table (in memory)
     - *Page Table Hit:* Load page table entry into TLB
     - *Page Table Miss (Page Fault):* Fetch page from disk to memory, update corresponding page table entry, then load entry into TLB

2) Check cache (input: PPN, output: data)
   - *Cache Hit:* Return data value to processor
   - *Cache Miss:* Fetch data value from memory, store it in cache, return it to processor

Page Faults

- Load the page off the disk into a free page of memory
  - Switch to some other process while we wait
- Interrupt thrown when page loaded and the process' page table is updated
  - When we switch back to the task, the desired data will be in memory
- If memory full, replace page (LRU), writing back if necessary, and update *both* page table entries
  - Continuous swapping between disk and memory called “thrashing”

Virtual Memory Summary

- User program view:
  - Contiguous memory
  - Start from some set VA
  - “Infinitely” large
  - Is the only running program
- Reality:
  - Non-contiguous memory
  - Start wherever available memory is
  - Finite size
  - Many programs running simultaneously
- Virtual memory provides:
  - Illusion of contiguous memory
  - All programs starting at same set address
  - Illusion of ~ infinite memory (2^{32} or 2^{64} bytes)
  - Protection , Sharing
- Implementation:
  - Divide memory into chunks (pages)
  - OS controls page table that maps virtual into physical addresses
  - memory as a cache for disk
  - TLB is a cache for the page table