



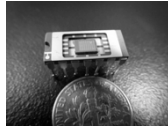
Lecturer SOE
Dan Garcia

inst.eecs.berkeley.edu/~cs61c
UCB CS61C : Machine Structures

Lecture 34 – Virtual Memory II 2010-04-16

OPTICAL COMPUTING REALIZED

Researchers at Stanford have developed “nanoscale single-mode LED”, which can transmit chip-to-chip data at 10 Gbs (10x what is currently used) at 1/1000th the energy. Pretty cool! (get it?) ☺



<http://news.stanford.edu/news/2011/november/data-transmission-breakthrough-111511.html>

Review

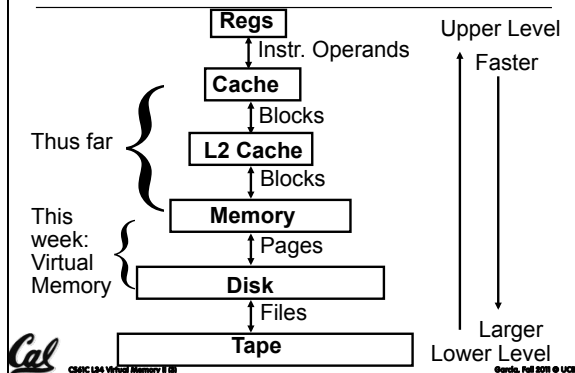
- Next level in the memory hierarchy:
 - Provides program with illusion of a very large main memory:
 - Working set of “pages” reside in main memory - others reside on disk.
- Also allows OS to share memory, protect programs from each other
- Today, more important for protection vs. just another level of memory hierarchy
- Each process thinks it has all the memory to itself
- (Historically, it predates caches)



CS61C L34 Virtual Memory II (2)

Garcia, Fall 2011 © UCB

Review: View of the Memory Hierarchy

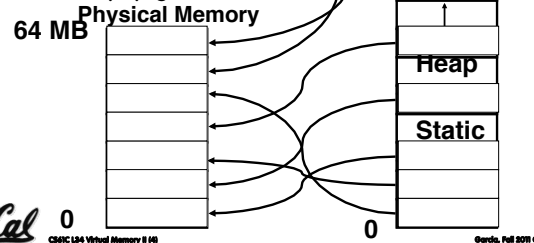


CS61C L34 Virtual Memory II (3)

Garcia, Fall 2011 © UCB

Mapping Virtual Memory to Physical Memory

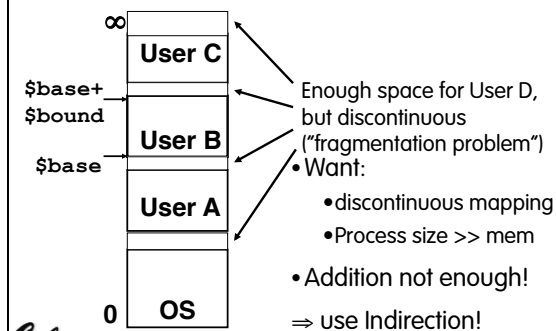
- Divide into equal sized chunks (about 4 KB - 8 KB)
- Any chunk of Virtual Memory assigned to any chunk of Physical Memory (“page”)



CS61C L34 Virtual Memory II (4)

Garcia, Fall 2011 © UCB

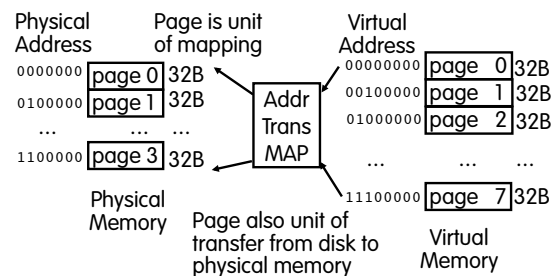
Another Model: Base and Bound Reg



CS61C L34 Virtual Memory II (5)

Garcia, Fall 2011 © UCB

Paging Organization (assume 32B pages)



CS61C L34 Virtual Memory II (6)

Garcia, Fall 2011 © UCB

Virtual Memory Mapping Function

- Cannot have simple function to predict arbitrary mapping
- Use table lookup of mappings

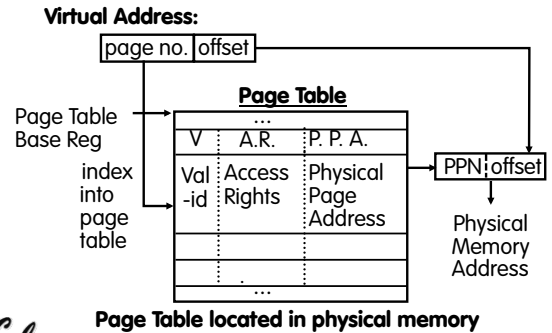
Page Number	Offset
-------------	--------
- Use table lookup ("Page Table") for mappings:
Page number is index
- Virtual Memory Mapping Function
 - Physical Offset = Virtual Offset
 - Physical Page Number = PageTable[Virtual Page Number] (P.P.N. also called "Page Frame")



CS63C L34 Virtual Memory II (7)

© 2012, Fall 2011 © UCS

Address Mapping: Page Table



CS63C L34 Virtual Memory II (8)

© 2012, Fall 2011 © UCS

Page Table

- A page table is an operating system structure which contains the mapping of virtual addresses to physical locations
 - There are several different ways, all up to the operating system, to keep this data around
- Each process running in the operating system has its own page table
 - "State" of process is PC, all registers, plus page table
 - OS changes page tables by changing contents of Page Table Base Register



CS63C L34 Virtual Memory II (9)

© 2012, Fall 2011 © UCS

Requirements revisited

- Remember the motivation for VM:
- Sharing memory with protection
 - Different physical pages can be allocated to different processes (sharing)
 - A process can only touch pages in its own page table (protection)
- Separate address spaces
 - Since programs work only with virtual addresses, different programs can have different data/code at the same address!
- What about the memory hierarchy?

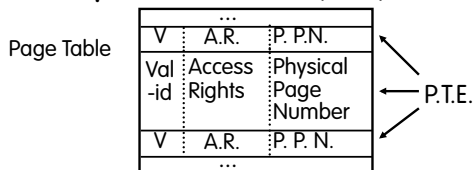


CS63C L34 Virtual Memory II (10)

© 2012, Fall 2011 © UCS

Page Table Entry (PTE) Format

- Contains either Physical Page Number or indication not in Main Memory
- OS maps to disk if Not Valid (V = 0)



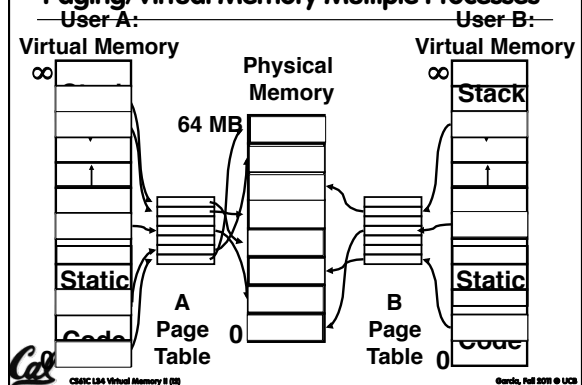
- If valid, also check if have permission to use page: Access Rights (A.R.) may be Read Only, Read/Write, Executable



CS63C L34 Virtual Memory II (11)

© 2012, Fall 2011 © UCS

Paging/Virtual Memory Multiple Processes



CS63C L34 Virtual Memory II (12)

© 2012, Fall 2011 © UCS

Comparing the 2 levels of hierarchy

Cache version	Virtual Memory vers.
Block or Line	Page
Miss	Page Fault
Block Size: 32-64B	Page Size: 4K-8KB
Placement:	Fully Associative
Direct Mapped, N-way Set Associative	
Replacement:	Least Recently Used
LRU or Random	(LRU)
Write Thru or Back	Write Back



CS50C L34 Virtual Memory I (8)

Garido, Fall 2011 © UCSB

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



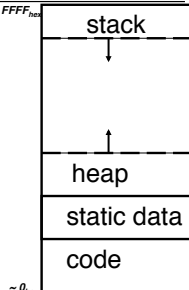
CS50C L34 Virtual Memory I (9)

Garido, Fall 2011 © UCSB

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).



CS50C L34 Virtual Memory I (10)

Garido, Fall 2011 © UCSB

Virtual Memory Problem #1

- Map every address \Rightarrow 1 indirection via Page Table in memory per virtual address \Rightarrow 1 virtual memory accesses = 2 physical memory accesses \Rightarrow SLOW!
- Observation: since locality in pages of data, there must be locality in virtual address translations of those pages
- Since small is fast, why not use a small cache of virtual to physical address translations to make translation fast?
- For historical reasons, cache is called a Translation Lookaside Buffer, or TLB

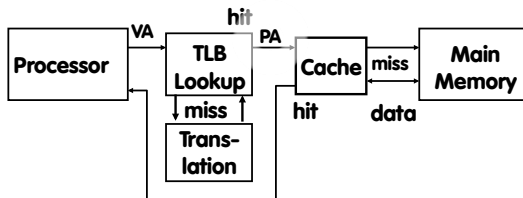


CS50C L34 Virtual Memory I (11)

Garido, Fall 2011 © UCSB

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



On TLB miss, get page table entry from main memory



CS50C L34 Virtual Memory I (12)

Garido, Fall 2011 © UCSB

Another Analogy

- Book title like virtual address
- Library of Congress call number like physical address
- Card catalogue like page table, mapping from book title to call #
- On card for book, in local library vs. in another branch like valid bit indicating in main memory vs. on disk
- On card, available for 2-hour in library use (vs. 2-week checkout) like access rights



CS50C L34 Virtual Memory I (13)

Garido, Fall 2011 © UCSB

Peer Instruction

- 1) Locality is important yet different for cache and virtual memory (VM): temporal locality for caches but spatial locality for VM
- 2) VM helps both with security and cost

	12
a)	FF
b)	FT
c)	TF
d)	TT



CS63C L34 Virtual Memory I (19)

©Gerd, Fall 2019 © UCSB

Peer Instruction Answer

- 1) Locality is important yet different for cache and virtual memory (VM): temporal locality for caches but spatial locality for VM
FALSE
 1. No. Both for VM and cache
- 2) VM helps both with security and cost
TRUE
 2. Yes. Protection and a bit smaller memory

	12
a)	FF
b)	FT
c)	TF
d)	TT



CS63C L34 Virtual Memory I (20)

©Gerd, Fall 2019 © UCSB

And in conclusion...

- **Manage memory to disk? Treat as cache**
 - Included protection as bonus, now critical
 - Use Page Table of mappings for each user vs. tag/data in cache
 - TLB is cache of Virtual \Rightarrow Physical addr trans
- **Virtual Memory allows protected sharing of memory between processes**
- **Spatial Locality means Working Set of Pages is all that must be in memory for process to run fairly well**



CS63C L34 Virtual Memory I (21)

©Gerd, Fall 2019 © UCSB