CS 61C:
Great Ideas in Computer Architecture
Virtual Machines/Programming Contest

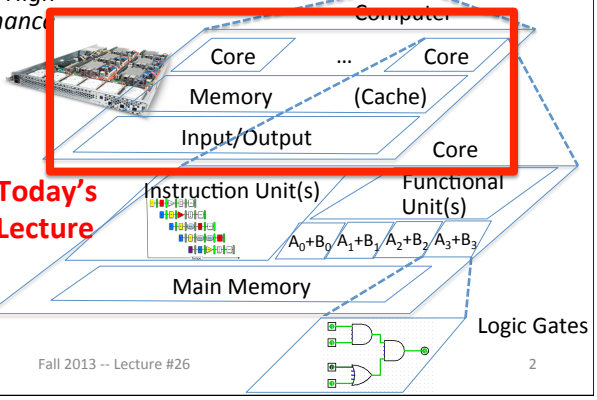
Instructor:
Randy H. Katz
http://inst.eecs.Berkeley.edu/~cs61c/fa13

Part II: You Are Here!

Software *Hardware*

- Parallel Requests
  Assigned to computer
  e.g., Search "Katz"
- Parallel Threads
  Assigned to core
  e.g., Lookup, Ads
- Parallel Instructions
  >1 instruction @ one time
  e.g., 5 pipelined instructions
- Parallel Data
  >1 data item @ one time
  e.g., Add of 4 pairs of words
- Hardware descriptions
  All gates @ one time
- Programming Languages

Harness Parallelism & Achieve High Performanc



Today's Lecture

Agenda

- Virtual Memory Review
- Virtual Machines
- Administrivia
- Programming Contest
- And, in Conclusion …

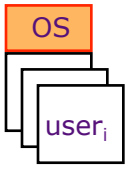
Agenda

- Virtual Memory Review
- Virtual Machines
- Administrivia
- Programming Contest
- And, in Conclusion …

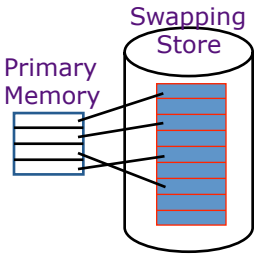
Modern Virtual Memory Systems

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

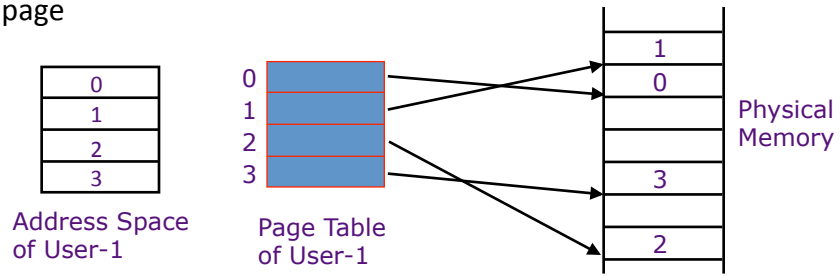


The price is address translation on each memory reference

Paged Memory Systems

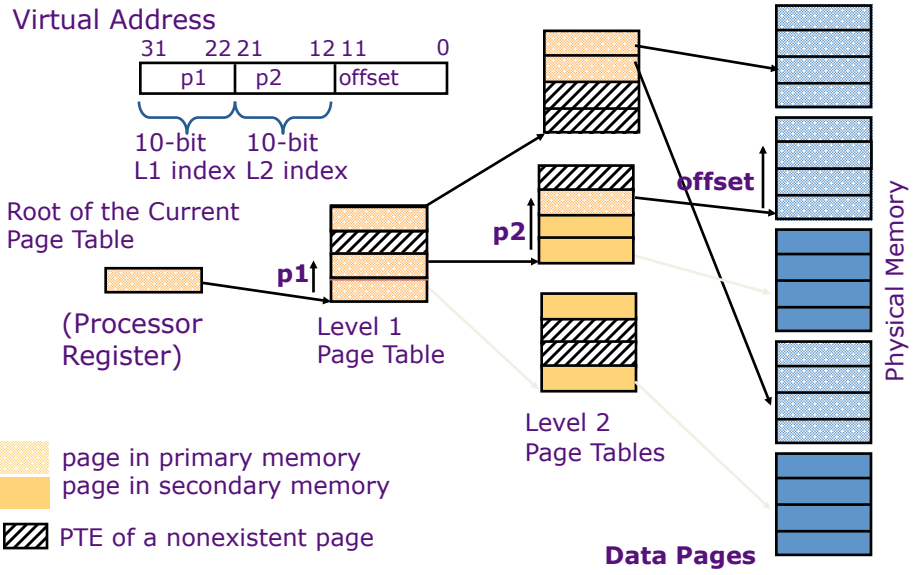
- Processor-generated address can be split into:

- A page table contains the physical address of the base of each page

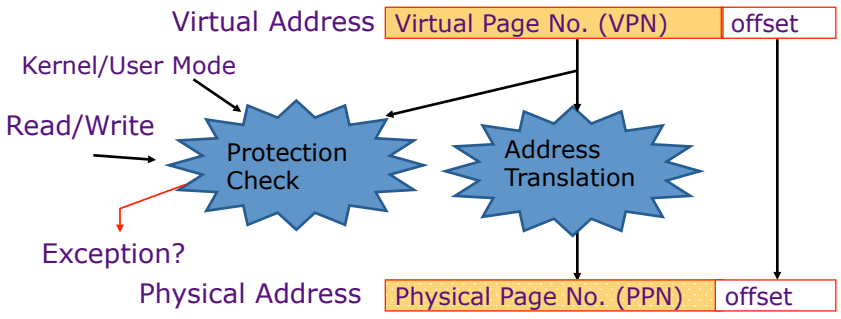


Page tables make it possible to store the pages of a program non-contiguously.

Hierarchical Page Table



Address Translation & Protection



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

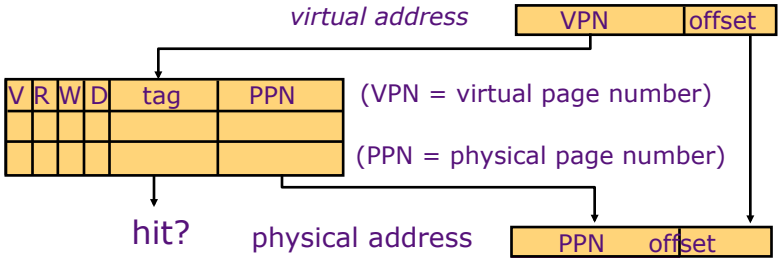
A good VM design needs to be fast (~ one cycle) and space efficient

Translation Lookaside Buffers (TLB)

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

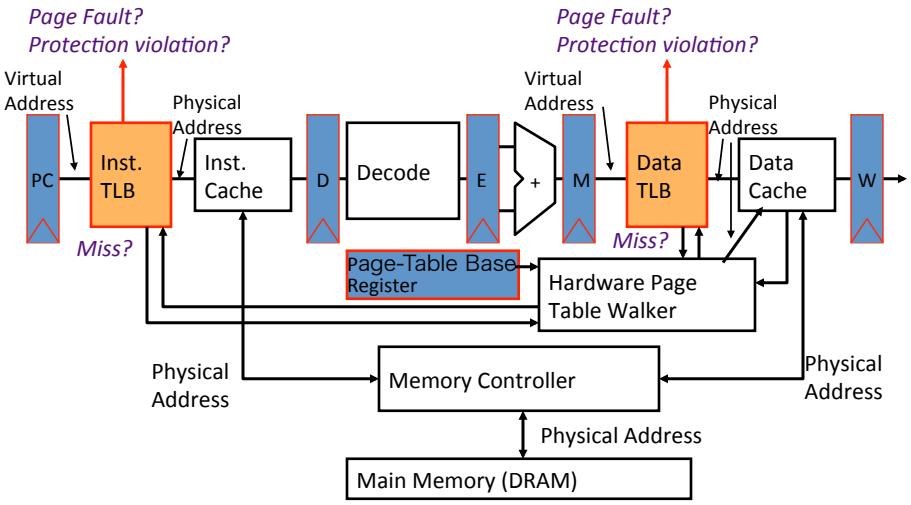
Solution: *Cache translations in TLB*

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



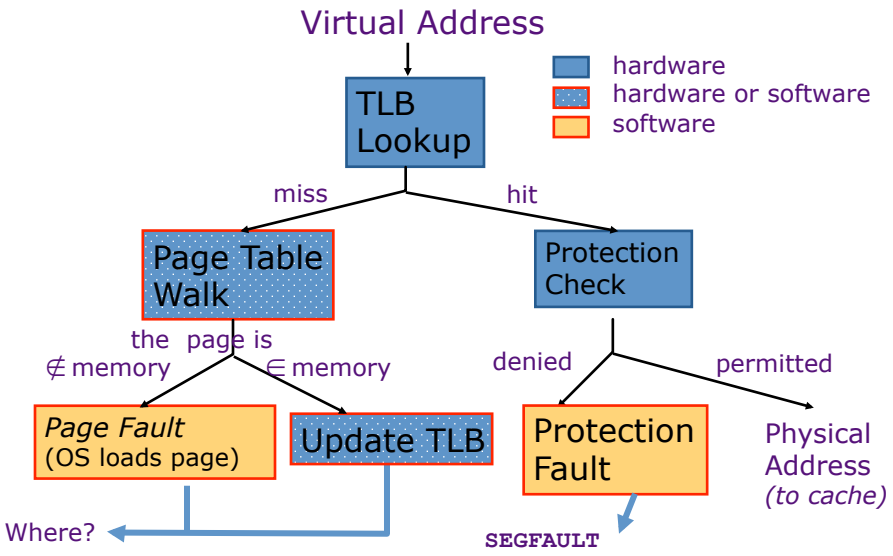
Page-Based Virtual-Memory Machine

(Hardware Page-Table Walk)



- Assumes page tables held in untranslated physical memory

Address Translation: Putting it All Together



Agenda

- Virtual Memory Review
- Virtual Machines
- Administrivia
- Programming Contest
- And, in Conclusion …

Safely Sharing a Machine

- Amazon Web Services allows independent tasks run on same computer
  - Can sell each "instance"
- Can a "small" operating system (~10,000 LOC) simulate the hardware of some machine, so that
  - Another operating system can run in that simulated hardware?
  - More than one instance of that operating system run on the same hardware at the same time?
  - More than one *different* operating system can share the same hardware at the same time?
  - And none can access each others' data?
- Answer: *Yes*

Solution – *Virtual Machine*

- A virtual machine provides interface *identical* to underlying bare hardware
  - I.e., all devices, interrupts, memory, etc.
- Examples
  - IBM VM/370 (1970s technology!)
  - VMWare (founded by Mendel & Diane Rosenblum)
  - Xen (used by AWS)
  - Microsoft Virtual PC
- Called "System Virtual Machines" vs. language interpreters (e.g., Java Virtual Machine)
- Virtualization has some performance impact
  - Feasible with modern high-performance computers

Virtual Machines

- *Host Operating System:*
  - OS actually running on the hardware
  - Together with *virtualization layer*, it simulates environment for …
- *Guest Operating System:*
  - OS running in the simulated environment
  - Runs identical as if on native hardware (except performance)
  - Cannot change access of real system resources
  - Guest OS code runs in native machine ISA
- The resources of the physical computer are shared to create the virtual machines
- Processor scheduling by OS can create the appearance that each user has own processor

Virtual Machine Monitor (a.k.a. Hypervisor)

- Maps virtual resources to physical resources
  - Memory, I/O devices, processors
- VMM handles real I/O devices
  - Emulates generic virtual I/O devices for guest OS
- Host OS must intercept attempts by Guest OS to access real I/O devices, allocate resources

Why Virtual Machines Popular (Again)?

- Increased importance of isolation and security
- Failures in security and reliability of modern OS's
- Sharing of single computer between many unrelated users
  - E.g., Cloud computing
- Dramatic increase in performance of processors makes VM overhead acceptable

5 Reasons Amazon Web Services uses Virtual Machines

- (Uses x86 ISA, Linux Host OS, and Xen VMM)

1. Allow AWS protect users from each other
2. Simplified SW distribution within WSC
   - Customers install image, AWS distributes to all instances
3. Can reliably "kill" a VM => control resource usage
4. VMs hide identity of HW => can keep selling old HW AND can introduce new more efficient HW
   - VM Performance not need be integer multiple of real HW
5. VM limiting rate of processing, network, and disk space => AWS offers many price points

Peer Instruction: True or False

Which statements is *True* about Virtual Machines?

I. Multiple Virtual Machines can run on one computer

II. Multiple Virtual Machine Monitors can run on one computer

III. The Guest OS must be the same as the Host OS

A) I only

B) II only

C) III only

D) I and II

Virtual Machine/Memory Instruction Set Support

- 2 modes in hardware: User and System modes
  - Some instruction only run in System mode
- Page Table Base Register (PTBR): Page Table addr
- Privileged instructions only available in system mode
  - Trap to system (and VMM) if executed in user mode
- All physical resources only accessible using privileged instructions
  - Including interrupt controls, I/O registers
- Renaissance of virtualization support in ISAs
  - Current ISAs (e.g., x86) adapting, following IBM's path

Example: Timer Virtualization

- In native machine (no VMM), on timer interrupt
  - OS suspends current process, handles interrupt, selects and resumes next process
- With Virtual Machine Monitor
  - VMM suspends current VM, handles interrupt, selects and resumes next VM
- If a VM requires timer interrupts
  - VMM emulates a virtual timer
  - Emulates interrupt for VM when physical timer interrupt occurs

Virtual Machine Monitor (a.k.a. Hypervisor)

- Maps virtual resources to physical resources
  - Memory, I/O devices, processors
- Guest OS code runs on native machine ISA in user mode
  - Traps to VMM on privileged instructions and access to protected resources
- Guest OS may be different from host OS
- VMM handles real I/O devices
  - Emulates generic virtual I/O devices for guest

Performance Impact of Virtual Machines?

- No impact on computation bound program, since they spend ≈0 time in OS
  - E.g., matrix multiply
- Big impact on I/O-intensive programs, since spend lots of time in OS and execute many systems calls and many privileged instructions
  - Although if I/O-bound => spend most time waiting for I/O device, then can hide VM overhead

Virtual Machines and Cores

- Host OS can also limit amount of *time* a virtual machine uses a processor (core)
- Hence, at cost of swapping registers, it can run multiple virtual machines on a *single* core
- AWS cheapest VM was originally 2 VMs per core
- Now, with faster processors, can install more VMs per core and deliver same performance or have multiple speeds of cores

Agenda

- Virtual Memory Review
- Virtual Machines
- Administrivia
- Programming Contest
- And, in Conclusion …

Administrivia

- Verify all HW, Lab, Midterm, and Project Grades by Friday, 12/6!
- Final Exam
  - Friday, December 20, 8:00-11:00 RSF Fieldhouse!
  - Short answer, fill in the blank, multiple choice, mix and match: 100 points/minutes
  - Comprehensive, but concentrated on material since midterm examination
  - Closed book/note, open crib sheet as before, MIPS Green Card provided
  - Review Session, Monday, 12/9, 1-4 PM, Room 155 Dwinelle
  - *Special consideration students, please contact*

Administrivia

- Topics for Final Exam
  - Cache Aware Programming/Cache Blocking
  - Data Level Parallelism: Intel SIMD SSE instructions and programming
  - Thread Parallelism: Cache Coherency + Synchronization concepts
  - OpenMP Programming
  - Hardware: Transistors to Gates
  - Hardware: Truth Tables to Boolean Algebra
  - Hardware: Synchronous System Timing and Timing Diagrams
  - Finite State Machines: State Diagrams and Implementation
  - CPU Design: Data Path Design (ALUs, Shifters, Register Files, Muxes)
  - CPU Design: Controller Design (FSMs for processor implementation)
  - Instruction Level Parallelism/Instruction Pipelining
  - Set Associative Caches
  - Dependability: ECC + RAID
  - Virtual Memory
  - X-semester issues: Great Ideas in Computer Architecture

Agenda

- Virtual Memory Review
- Virtual Machines
- Administrivia
- Programming Contest
- And, in Conclusion …

Programming Contest

Agenda

- Virtual Memory
- Virtual Machines
- Administrivia
- Programming Contest
- And, in Conclusion …

Agenda

- Virtual Memory
- Administrivia
- Virtual Machines
- And, in Conclusion …

And in Conclusion, …

- Virtual Memory, Paging for Isolation and Protection, help Virtual Machines share memory
  - Can think of as another level of memory hierarchy, but not really used like caches are today
  - Not really routinely paging to disk today
- Virtual Machines as even greater level of protection to allow greater level of sharing
  - Enables fine control, allocation, software distribution, multiple price points for Cloud Computing