

CS352H: Computer Systems Architecture

Topic 12: Memory Hierarchy - Virtual Memory and Virtual Machines

October 27, 2009



Memory Technology

- Static RAM (SRAM)
 - 0.5ns – 2.5ns, \$2000 – \$5000 per GB
- Dynamic RAM (DRAM)
 - 50ns – 70ns, \$20 – \$75 per GB
- Magnetic disk
 - 5ms – 20ms, \$0.20 – \$2 per GB
- Ideal memory
 - Access time of SRAM
 - Capacity and cost/GB of disk



Principle of Locality

- Programs access a small proportion of their address space at any time
- Temporal locality
 - Items accessed recently are likely to be accessed again soon
 - e.g., instructions in a loop, induction variables
- Spatial locality
 - Items near those accessed recently are likely to be accessed soon
 - E.g., sequential instruction access, array data

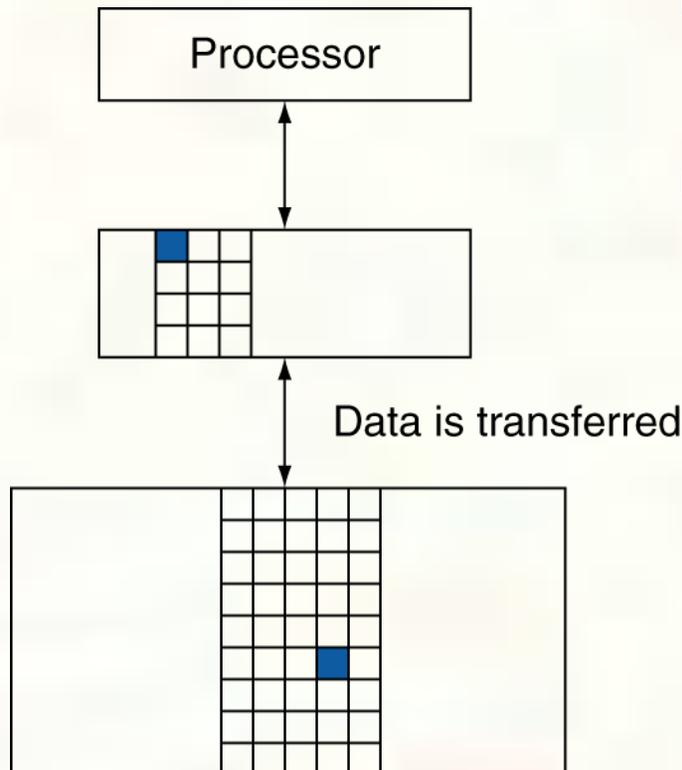


Taking Advantage of Locality

- Memory hierarchy
- Store everything on disk
- Copy recently accessed (and nearby) items from disk to smaller DRAM memory
 - Main memory
- Copy more recently accessed (and nearby) items from DRAM to smaller SRAM memory
 - Cache memory attached to CPU



Memory Hierarchy Levels



- Block (aka line): unit of copying
 - May be multiple words
- If accessed data is present in upper level
 - Hit: access satisfied by upper level
 - Hit ratio: hits/accesses
- If accessed data is absent
 - Miss: block copied from lower level
 - Time taken: miss penalty
 - Miss ratio: misses/accesses
 $= 1 - \text{hit ratio}$
 - Then accessed data supplied from upper level



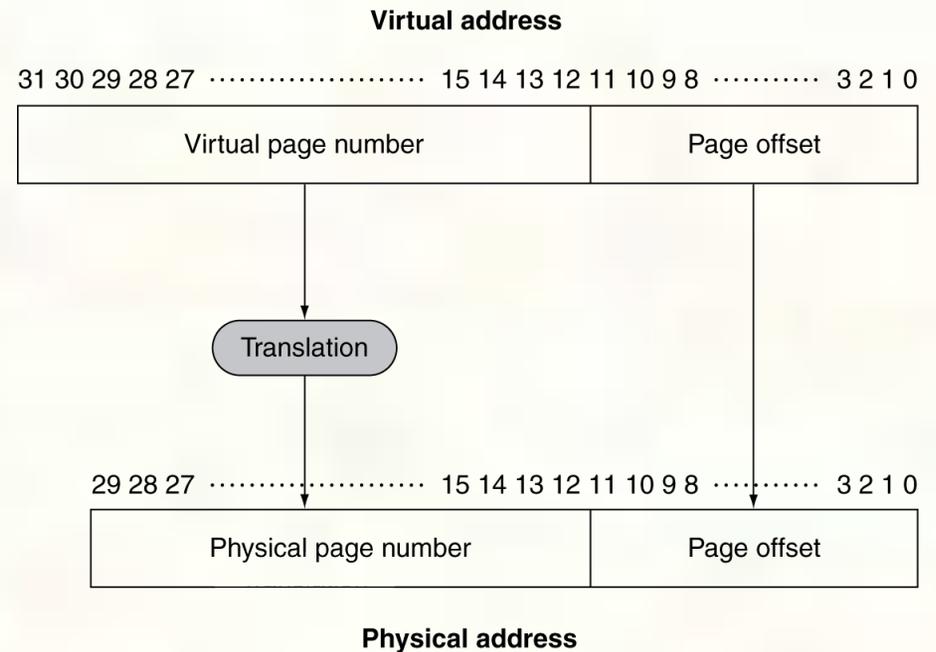
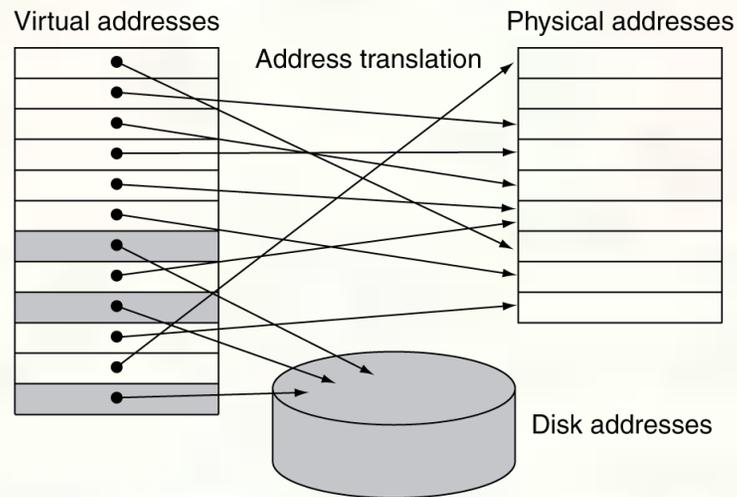
Virtual Memory

- Use main memory as a “cache” for secondary (disk) storage
 - Managed jointly by CPU hardware and the operating system (OS)
- Programs share main memory
 - Each gets a private virtual address space holding its frequently used code and data
 - Protected from other programs
- CPU and OS translate virtual addresses to physical addresses
 - VM “block” is called a page
 - VM translation “miss” is called a page fault



Address Translation

■ Fixed-size pages (e.g., 4K)





Page Fault Penalty

- On page fault, the page must be fetched from disk
 - Takes millions of clock cycles
 - Handled by OS code
- Try to minimize page fault rate
 - Fully associative placement
 - Smart replacement algorithms

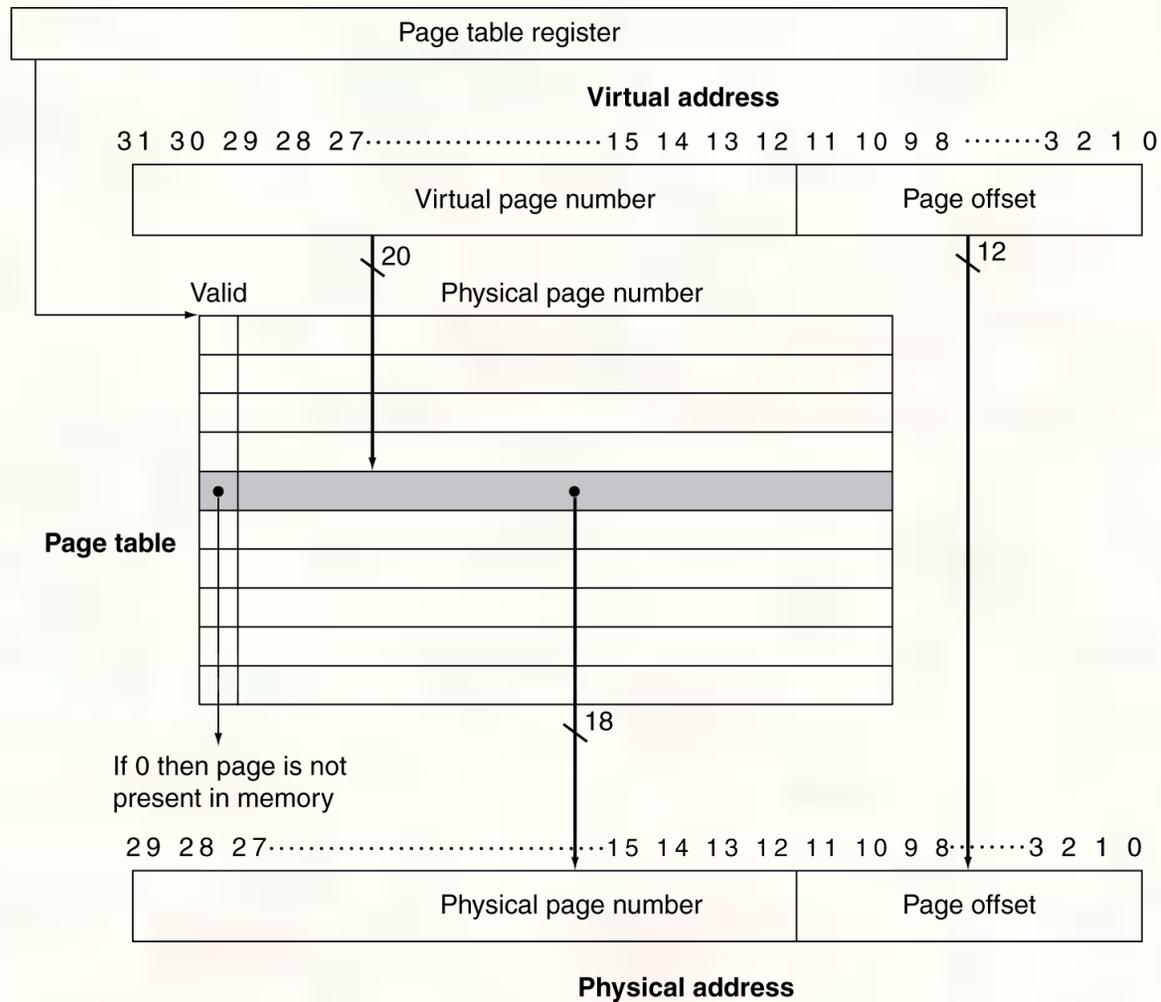


Page Tables

- Stores placement information
 - Array of page table entries, indexed by virtual page number
 - Page table register in CPU points to page table in physical memory
- If page is present in memory
 - PTE stores the physical page number
 - Plus other status bits (referenced, dirty, ...)
- If page is not present
 - PTE can refer to location in swap space on disk

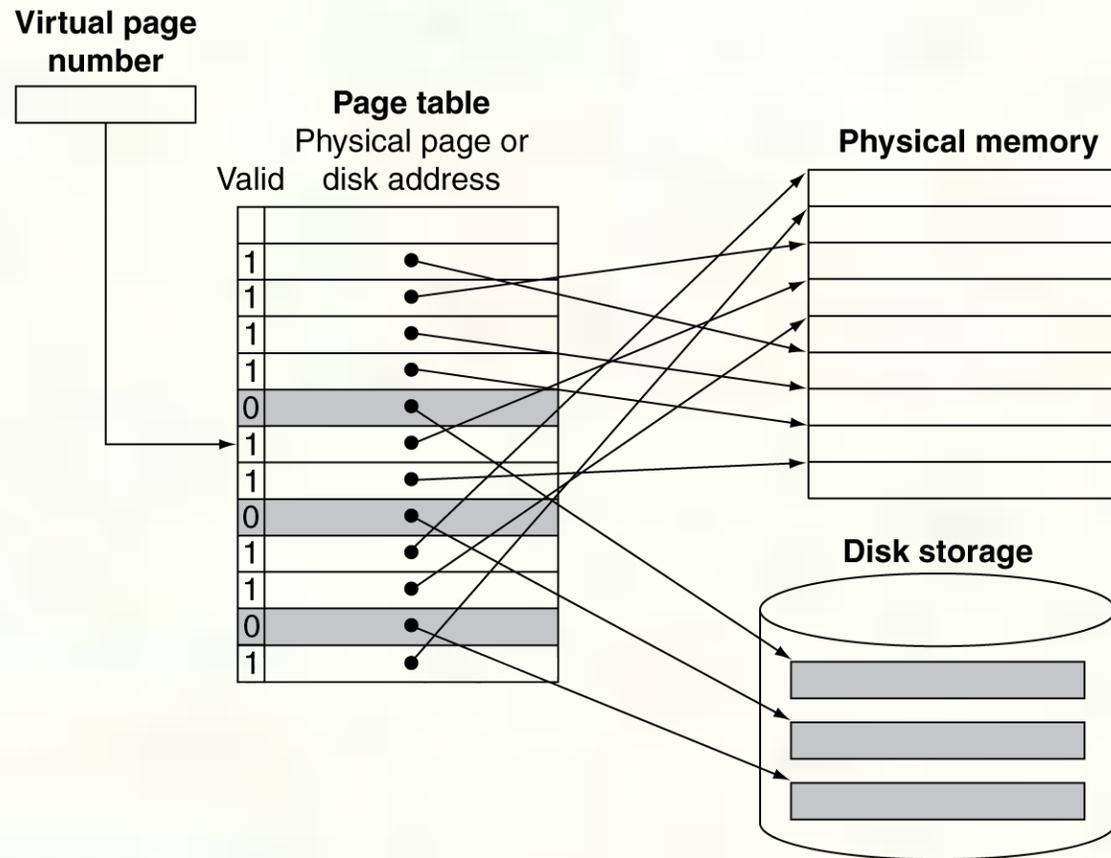


Translation Using a Page Table





Mapping Pages to Storage





Replacement and Writes

- To reduce page fault rate, prefer least-recently used (LRU) replacement
 - Reference bit (aka use bit) in PTE set to 1 on access to page
 - Periodically cleared to 0 by OS
 - A page with reference bit = 0 has not been used recently
- Disk writes take millions of cycles
 - Block at once, not individual locations
 - Write through is impractical
 - Use write-back
 - Dirty bit in PTE set when page is written

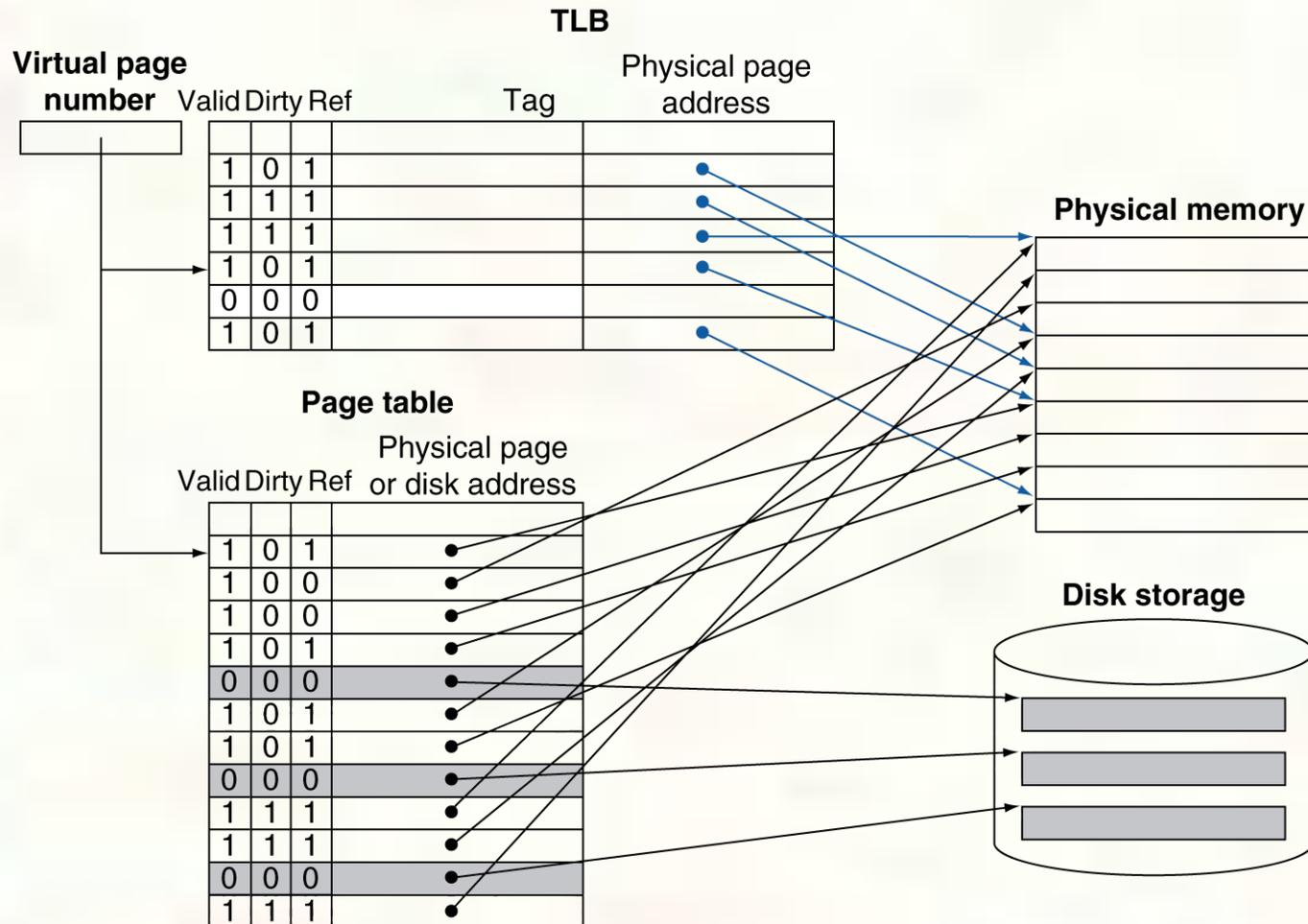


Fast Translation Using a TLB

- Address translation would appear to require extra memory references
 - One to access the PTE
 - Then the actual memory access
- But access to page tables has good locality
 - So use a fast cache of PTEs within the CPU
 - Called a Translation Look-aside Buffer (TLB)
 - Typical: 16–512 PTEs, 0.5–1 cycle for hit, 10–100 cycles for miss, 0.01%–1% miss rate
 - Misses could be handled by hardware or software



Fast Translation Using a TLB





TLB Misses

- If page is in memory
 - Load the PTE from memory and retry
 - Could be handled in hardware
 - Can get complex for more complicated page table structures
 - Or in software
 - Raise a special exception, with optimized handler
- If page is not in memory (page fault)
 - OS handles fetching the page and updating the page table
 - Then restart the faulting instruction



TLB Miss Handler

- TLB miss indicates
 - Page present, but PTE not in TLB
 - Page not present
- Must recognize TLB miss before destination register overwritten
 - Raise exception
- Handler copies PTE from memory to TLB
 - Then restarts instruction
 - If page not present, page fault will occur

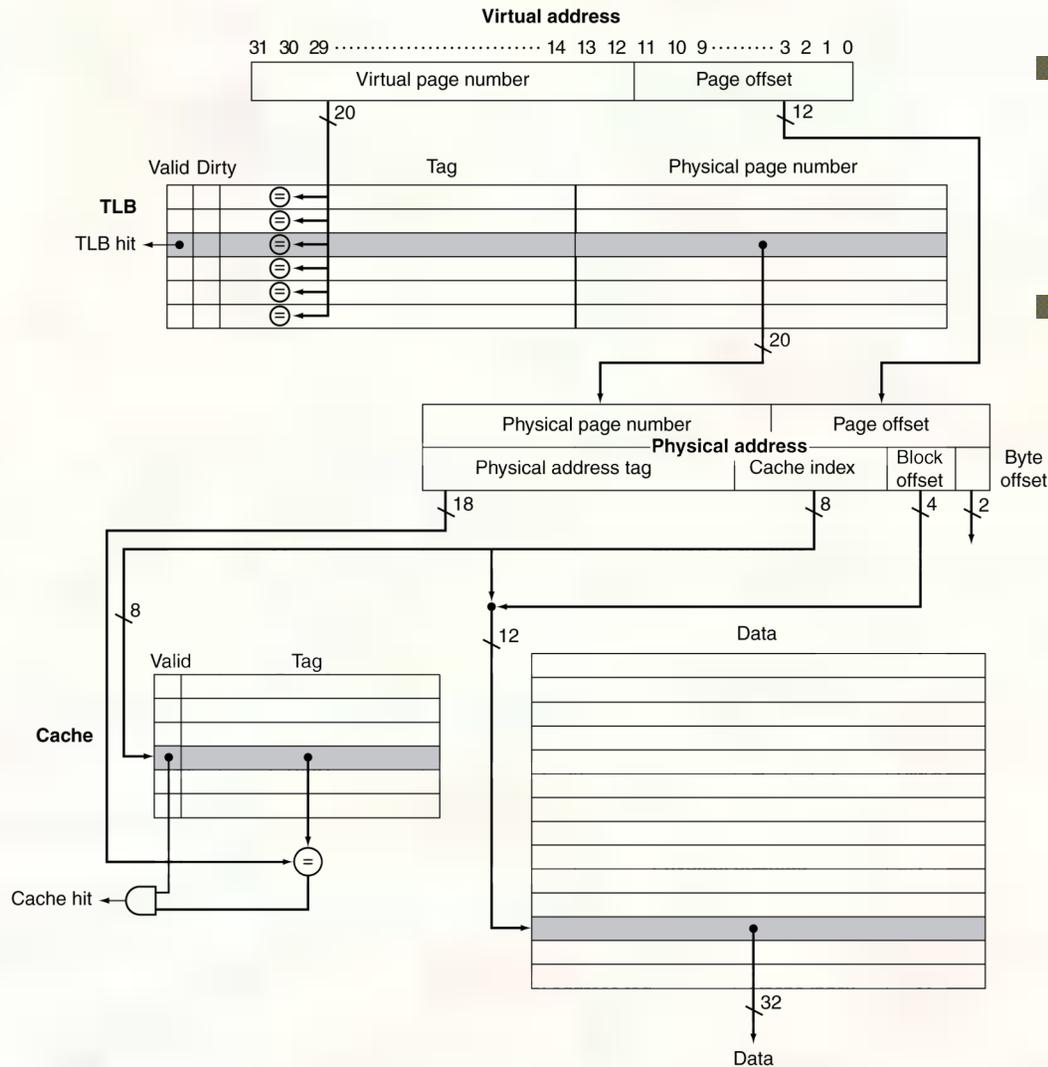


Page Fault Handler

- Use faulting virtual address to find PTE
- Locate page on disk
- Choose page to replace
 - If dirty, write to disk first
- Read page into memory and update page table
- Make process runnable again
 - Restart from faulting instruction



TLB and Cache Interaction



- If cache tag uses physical address
 - Need to translate before cache lookup
- Alternative: use virtual address tag
 - Complications due to aliasing
 - Different virtual addresses for shared physical address



Memory Protection

- Different tasks can share parts of their virtual address spaces
 - But need to protect against errant access
 - Requires OS assistance
- Hardware support for OS protection
 - Privileged supervisor mode (aka kernel mode)
 - Privileged instructions
 - Page tables and other state information only accessible in supervisor mode
 - System call exception (e.g., syscall in MIPS)



Virtual Machines

- Host computer emulates guest operating system and machine resources
 - Improved isolation of multiple guests
 - Avoids security and reliability problems
 - Aids sharing of resources
- Virtualization has some performance impact
 - Feasible with modern high-performance computers
- Examples
 - IBM VM/370 (1970s technology!)
 - VMWare
 - Microsoft Virtual PC



Virtual Machine Monitor

- Maps virtual resources to physical resources
 - Memory, I/O devices, CPUs
- Guest code runs on native machine 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



Example: Timer Virtualization

- In native machine, 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



Instruction Set Support

- User and System modes
- Privileged instructions only available in system mode
 - Trap to system if executed in user mode
- All physical resources only accessible using privileged instructions
 - Including page tables, interrupt controls, I/O registers
- Renaissance of virtualization support
 - Current ISAs (e.g., x86) adapting



Pitfalls

- Extending address range using segments
 - E.g., Intel 80286
 - But a segment is not always big enough
 - Makes address arithmetic complicated
- Implementing a VMM on an ISA not designed for virtualization
 - E.g., non-privileged instructions accessing hardware resources
 - Either extend ISA, or require guest OS not to use problematic instructions



Concluding Remarks

- Fast memories are small, large memories are slow
 - We really want fast, large memories ☹
 - Caching gives this illusion ☺
- Principle of locality
 - Programs use a small part of their memory space frequently
- Memory hierarchy
 - L1 cache ↔ L2 cache ↔ ... ↔ DRAM memory
↔ disk
- Memory system design is critical for multiprocessors