# CS323 Operating Systems
# Memory Management III

Yuanyuan Zhou
Lecture 15
2/24/2003

## Content of this lecture

- Administrative announcements
- Page Table and TLB
- Multi-level Paging
- Inverted Page Table
- Protection
- Summary



## Administrative

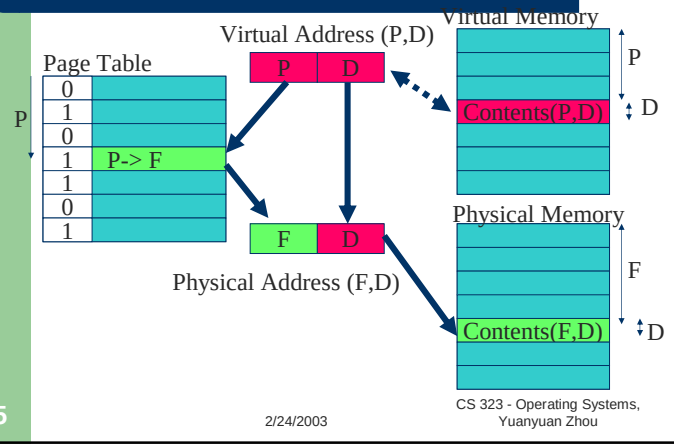
- Quiz2 starts
- MP2



## Review

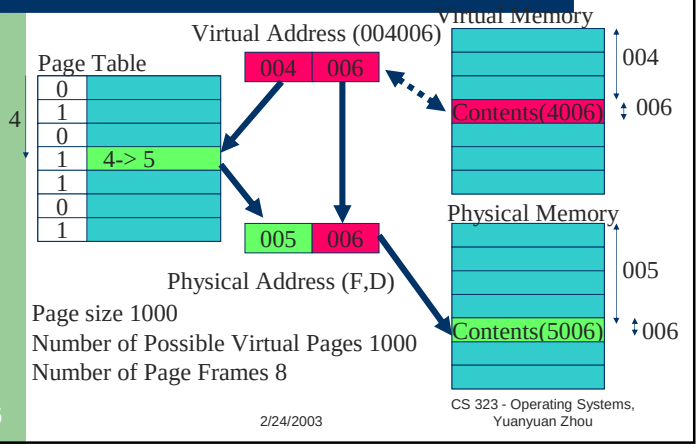
- Storage management
  - Bitmap or link list
  - Compaction
  - Best fit, quick fit, first fit, next fit, worst fit
- Virtual Memory
- Paging
  - Page fault

## Page Mapping Hardware (page hit)






## Page Mapping Hardware






## Page Fault

- Access a virtual page that is not mapped into any physical page
  - A fault is triggered by hardware
- Page fault handler (by VM Software, a part of OS)
  - Find if there is any free physical page available
    - If no, evict some resident page to disk (swapping space)
  - Allocate a free physical page
  - Load the faulted virtual page to the prepared physical page
  - Modify the page table



## Paging Issues

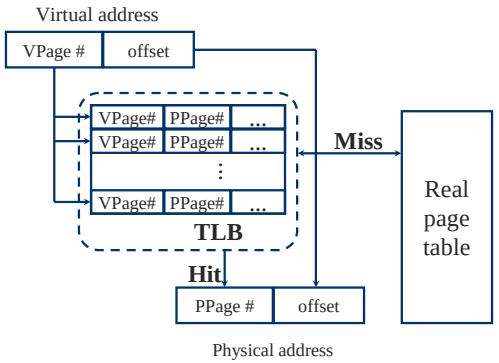
- Page size is $2^n$
  - usually 512, 1k, 2k, 4k, or 8k
  - E.g. 32 bit VM address may have $2^{20}$ (1MB) pages with 4k ($2^{12}$ ) bytes per page
- Page table:
  - $2^{20}$ page entries take $2^{22}$ bytes (4MB)
  - page frames must map into real memory
  - Page Table base register must be changed for context switch
- NO External fragmentation, Internal fragmentation on last page ONLY

## Virtual-To-Physical Lookups

- Programs only know virtual addresses
  - The page table can be extremely large
- Each virtual address must be translated
  - May involve walking hierarchical page table
  - Page table stored in memory
  - So, each program memory access requires several actual memory accesses
- Solution: cache “active” part of page table
  - TLB also called “associative memory”



## Translation Look-aside Buffer (TLB)



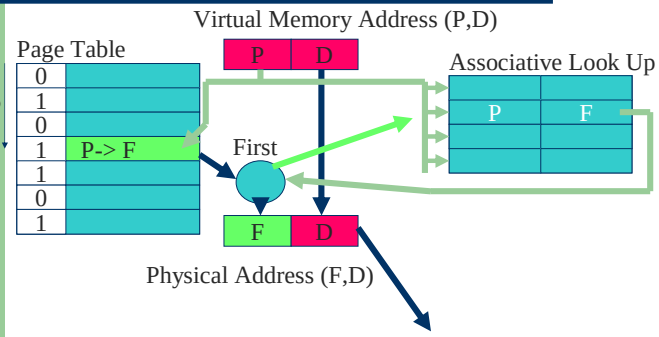



## TLB Function

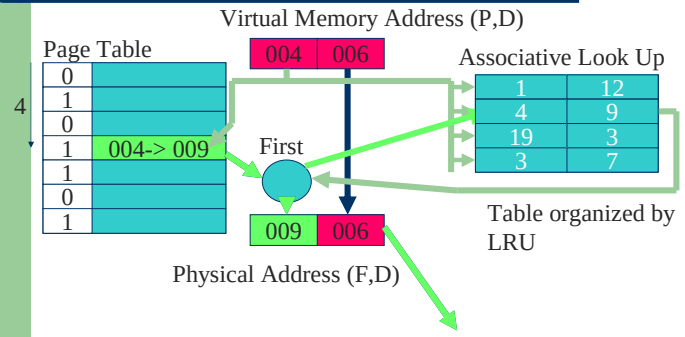
- If a virtual address is presented to MMU, the hardware checks TLB by comparing all entries simultaneously (in parallel).
- If match is valid, the page is taken from TLB without going through page table.
- If match is not valid
  - MMU detects miss and does an ordinary page table lookup.
  - It then evicts one page out of TLB and replaces it with the new entry, so that next time that page is found in TLB.



## Page Mapping Hardware

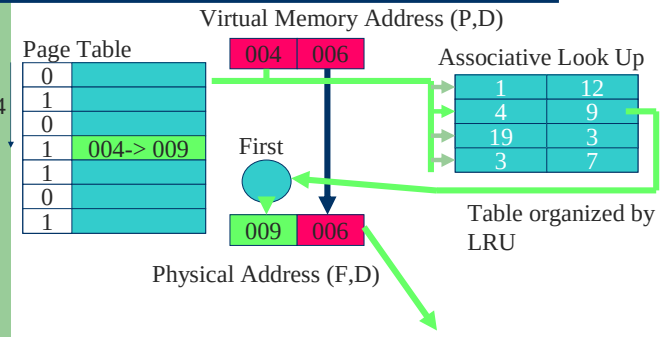


## Page Mapping Example






## Page Mapping Example: next reference






## Bits in a TLB Entry

- Common (necessary) bits
  - Virtual page number: match with the virtual address
  - Physical page number: translated address
  - Valid
  - Access bits: kernel and user (nil, read, write)
- Optional (useful) bits
  - Process tag
  - Reference
  - Modify
  - Cacheable



## Paging Implementation Issues

- TLB can be implemented using
  - Associative registers
  - Look-aside memory
  - Content-addressable memory
- TLB hit ratio (Page address cache hit ratio)
  - Percentage of time page found in associative memory

## Hardware-Controlled TLB

- On a TLB miss (different from page fault)
  - Hardware loads the PTE into the TLB
    - Need to write back if there is no free entry
  - Generate a fault if the page containing the PTE is invalid
  - VM software performs fault handling
  - Restart the CPU
- On a TLB hit, hardware checks the valid bit
  - If valid, pointer to page frame in memory
  - If invalid, the hardware generates a page fault
    - Perform page fault handling
    - Restart the faulting instruction



## Software-Controlled TLB

- On a miss in TLB, VM software
  - Write back if there is no free entry
  - Check if the page containing the PTE is in memory
  - If no, perform page fault handling
  - Load the PTE into the TLB
  - Restart the faulting instruction
- On a hit in TLB, the hardware checks valid bit
  - If valid, pointer to page frame in memory
  - If invalid, the hardware generates a page fault
    - Perform page fault handling
    - Restart the faulting instruction
- Example: RISC including SPARC, Alpha, MIPS, HP PA



## Hardware vs. Software Controlled

- Hardware approach
  - Efficient
  - Inflexible
  - Need more space for page table
- Software approach
  - Flexible
  - Software can do mappings by hashing
    - PP# $\rightarrow$ (Pid, VP#)
    - (Pid, VP#) $\rightarrow$ PP#
  - Can deal with large virtual address space



## Group Discussion

- Similarity between Cache and TLB
- Difference between Cache and TLB

## Cache vs. TLBs

- Similarities
  - Both cache a portion of memory
  - Both write back on a miss
- Combine L1 cache with TLB
  - Virtually addressed cache
  - Why wouldn't everyone use virtually addressed caches?
- Differences
  - Associativity
    - TLB is usually fully set-associative
    - Cache can be direct-mapped
  - Consistency
    - TLB does not deal with consistency with memory
    - TLB can be controlled by software



## Issues

- What TLB entry to be replaced?
  - Random
  - Pseudo Least Recently Used (LRU)
- What happens on a context switch?
  - Process tag: change TLB registers and process register
  - No process tag: Invalidate the entire TLB contents
- What happens when changing a page table entry?
  - Change the entry in memory
  - Invalidate the TLB entry



## Effective Access Time

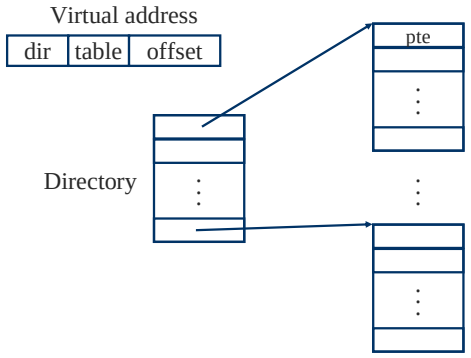
- TLB lookup time = $\varepsilon$ time unit
- Memory cycle -- m microsecond
- TLB Hit ratio -- $\alpha$
- Effective access time
  - Eat = $(1m+\varepsilon)\alpha+(2m+\varepsilon)(1-\alpha)$
  - Eat = $2m+\varepsilon-\alpha$



## Multilevel Page Tables

- Since the page table can be very large, one solution is to page the page table
- Divide the page number into
  - An index into a page table of second level page tables
  - A page within a second level page table
- Advantage
  - No need to keeping all the page tables in memory all the time
  - Only recently accessed memory's mapping need to be kept in memory, the rest can be fetched on demand

## Multiple-Level Page Tables

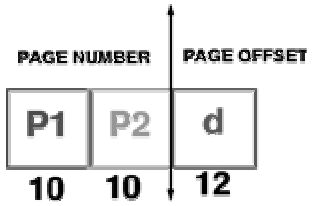



What does this buy us? Sparse address spaces and easier paging



## Example Addressing on a Multilevel Page Table System

- A logical address (on 32 bit machine with 4k page size) is divided into
  - A page number consisting of 20 bits
  - A page offset consisting of 12 bits
- Divide the page number into
  - A 10-bit page number
  - A 10-bit page offset



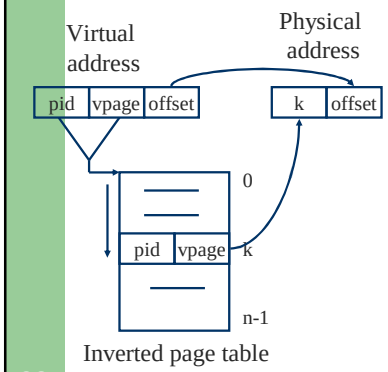



## Multilevel Paging and Performance

- Since each level is stored as a separate table in memory, converting a logical address to a physical one in a four-level paging may take five memory accesses. Why?

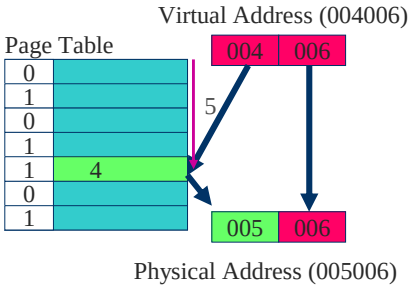


## Inverted Page Tables




- Main idea
  - One PTE for each physical page frame
  - Hash (Vpage, pid) to Ppage#
  - Trade off space for time
- Pros
  - Small page table for large address space
- Cons
  - Lookup is difficult
  - Overhead of managing hash chains, etc

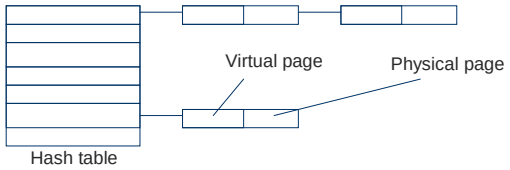
## Inverted Page Table






## Inverted Page Table Implementation

- TLB is same as before
- TLB miss is handled by software
- In-memory page table is managed using a hash table
  - number of entries = number of physical frames
  - Not found: page fault



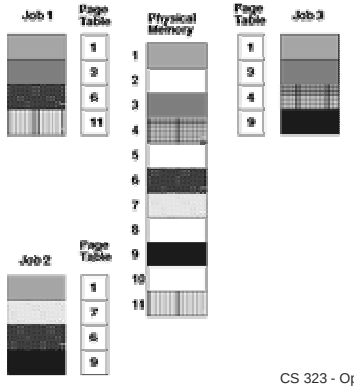



## Sharing Pages

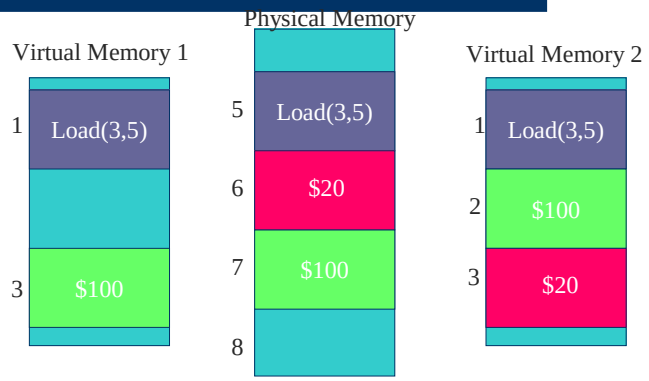
- Code and data can be shared among processes
  - mapping them into pages with common page frame mappings
- Code and data must be position independent if VM mappings for the shared data are different
- Code and data cannot store VM addresses for functions and variables not in shared pages
- Shared code and data usually are not position independent resulting in certain regions of memory being reserved for shared libraries and the operating system



## Shared Pages



## Incorrect Sharing






## Protection

- Can add read, write, execute protection bits to page table to protect memory
- Check is done by hardware during access
- shared memory location
  - different protections from different processes
  - Solution:
    - associate protection lock with page frame. Each process has its own key. If the key fits the lock, the process may access the page frame

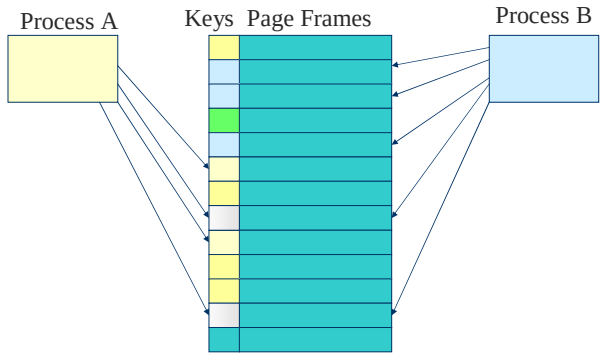


## Protection

- Typically many different keys can fit a lock using a priority numbering scheme
- (E.G. Key 3 fits all locks 3, 7, 15
- Key 4 fits 5,6,7,12,13,14,15.)



## Keys and Locks



## Reminder

- MP2