CS 316: Virtual Memory

Kavita Bala

Fall 2007

Computer Science

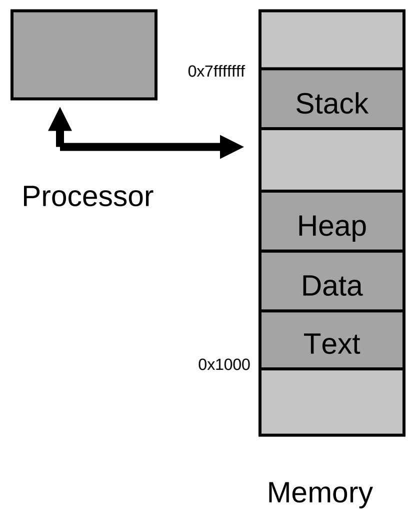
Cornell University

Announcements

- HW 2 is due on Friday
- PA 4 is out on Friday

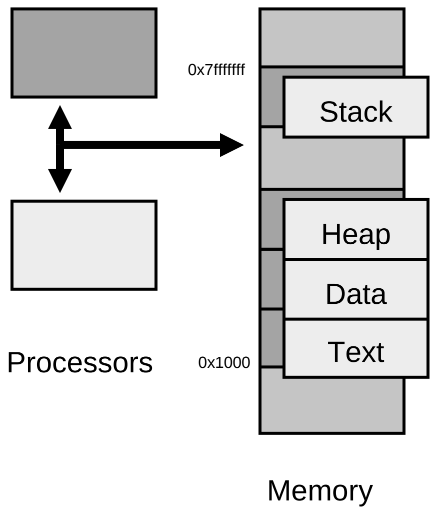
Processor & Memory

- Currently, the processor’s address lines are directly routed via the system bus to the memory banks
  – Simple, fast
- What happens when the program issues a load or store to an invalid location?
  – e.g. 0x000000000 ?
  – uninitialized pointer



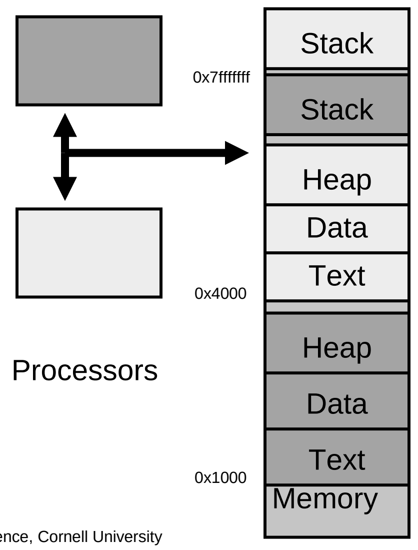
Physical Addressing Problems

- What happens when another program is executed concurrently on another processor?
  – The addresses will conflict



Physical Addressing Problems

- What happens when another program is executed concurrently on another processor?
  - The addresses will conflict
- We could try to relocate the second program to another location
  - Assuming there is one
  - Introduces more problems!

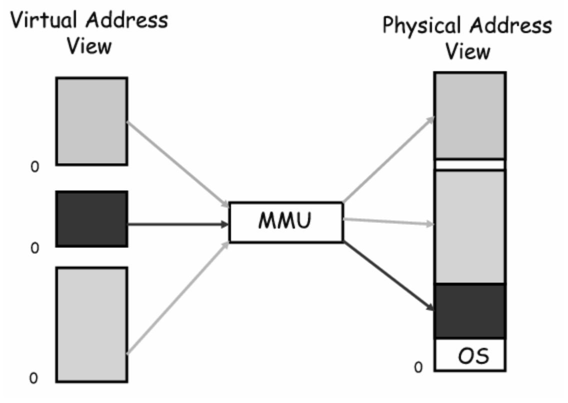


Virtual Memory

- Each process has its own view of memory
  - Called virtual address space
  - So can conceptually put your code, data in the place you want it
- On-the-fly at runtime: need translation from virtual address space to physical address space of machine
  - Relocate loads and stores to actual memory
- Interface:
  - Programs load/store to virtual addresses
  - Actual memory uses physical addresses

Address Space

- Memory Management Unit (MMU)
  - Combination of hardware and software



Virtual Memory Advantages

- Easy relocation: simplifies loading a program for execution by providing for code relocation (i.e., the code can be loaded anywhere in main memory)

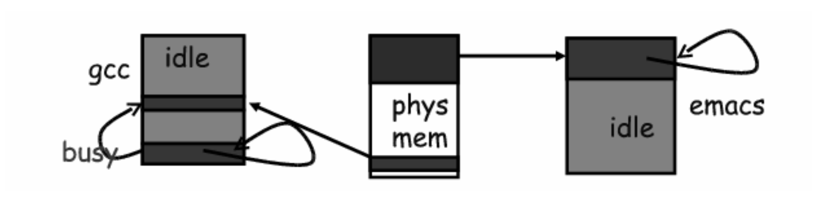
- Easy sharing: allows efficient and safe sharing of memory among multiple programs

Virtual Memory Advantages

- Larger than physical memory: use main memory as a “cache” for secondary memory
  - Provides ability to run programs larger than the size of physical memory
  - Again, the Principle of Locality
    - The 80/20 rule
    - A program is likely to access a relatively small portion of its address space during any period of time

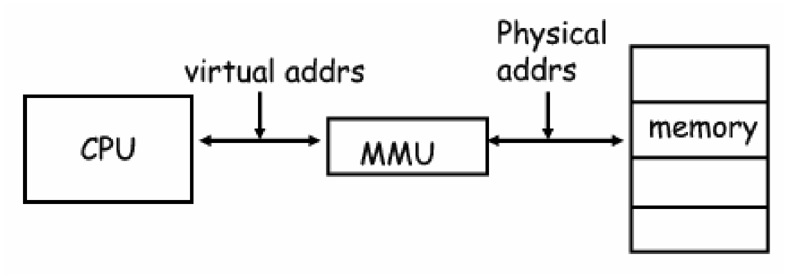
Virtual Memory Advantages

- Can relocate program while running
- Virtualization
  - In CPU: if process is not doing anything, switch
  - In memory: when not using it, somebody else can use it



How to make it work?

- Challenge: Virtual Memory can be slow!
- At run-time: virtual address must be translated to a physical address
- MMU (combination of hardware and software)



Address Translation

- How to translate addresses?
  - Per word? Per block?
- Costs dictate granularity of translation
  - Cost to disk is very large
  - Block size has to be large too
- A program’s address space is divided into pages (all one fixed size)
  - Typical: 4KB to 16KB
  - Example: virtual address space: 2^32 = 4GB, physical address space: 1 GB, page size: 4KB

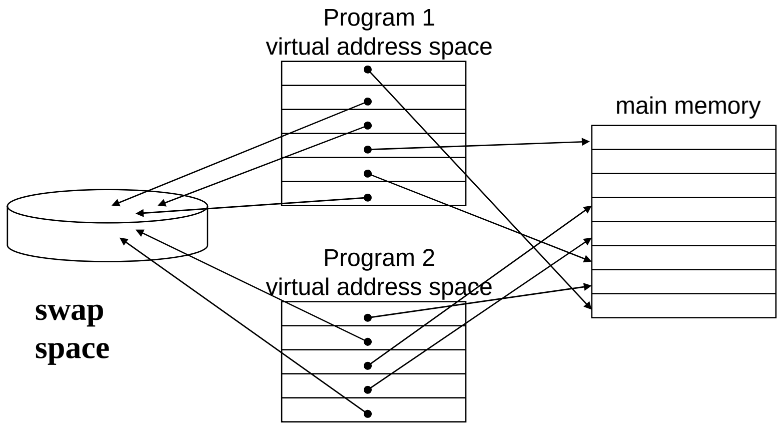
Paging

- Divide memory into small pages
- Each process has separate mapping of virtual to physical pages

- OS gets control on certain operations
  - Read-only pages trap to OS on write
  - Invalid pages trap to OS on read/write

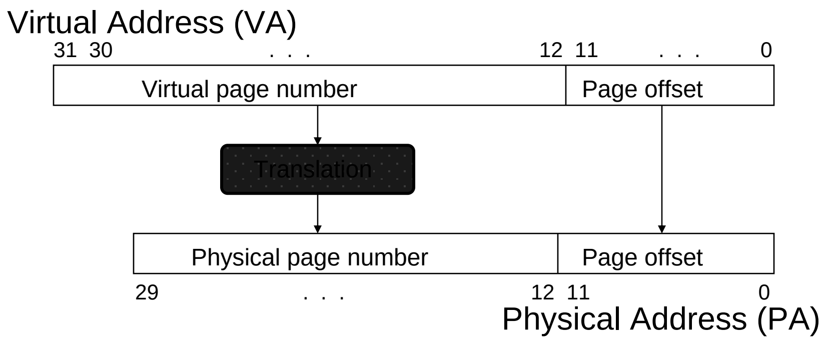
Two Programs Sharing Physical Memory

- The starting location of each page (either in main memory or in secondary memory) is contained in the program's page table

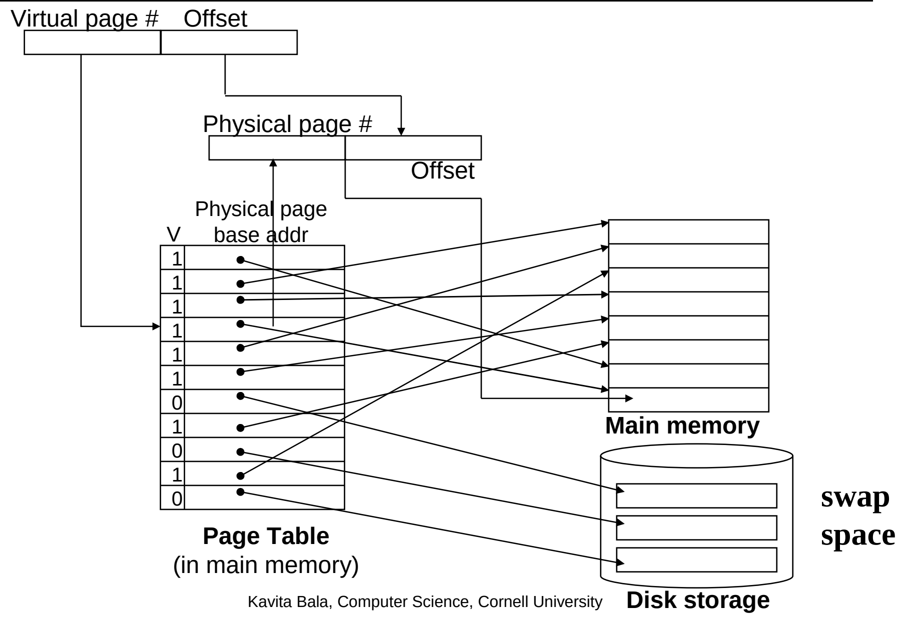


Address Translation

- So each memory request *first* requires an address translation from the virtual space to the physical space

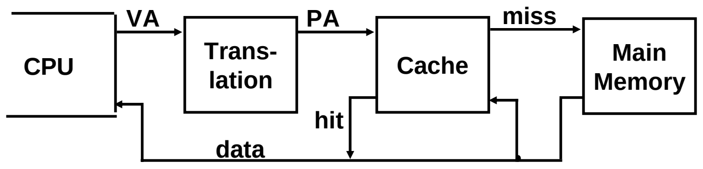


Page Table for Translation



Virtual Addressing with a Cache

- Thus it takes an *extra* memory access to translate a VA to a PA

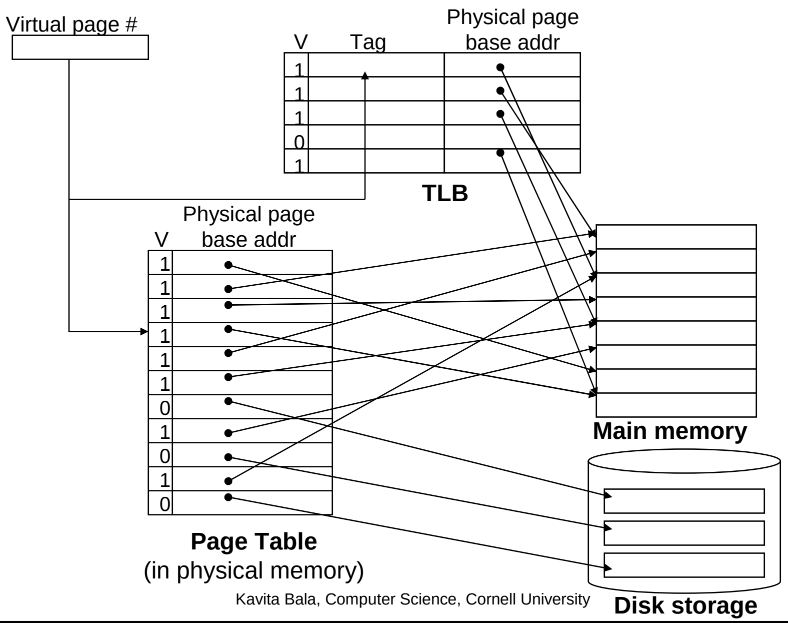


- This makes memory (cache) accesses very expensive (if every access was really *two* accesses)

Translation Lookaside Buffer (TLB)

- Hardware: TLB
- A small cache of recently used address mappings
  - – TLB hit: avoid a page table lookup

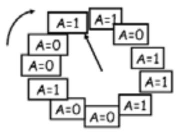
Making Address Translation Fast



Translation Lookaside Buffers (TLBs)

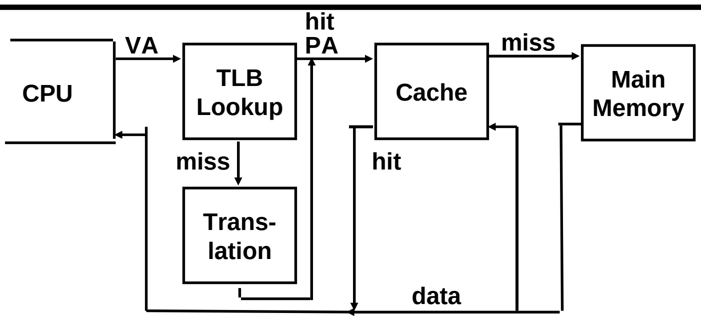
- Just like any other cache, the TLB can be organized as fully associative, set associative, or direct mapped

V	Virtual Page #	Physical Page #	Dirty	Ref



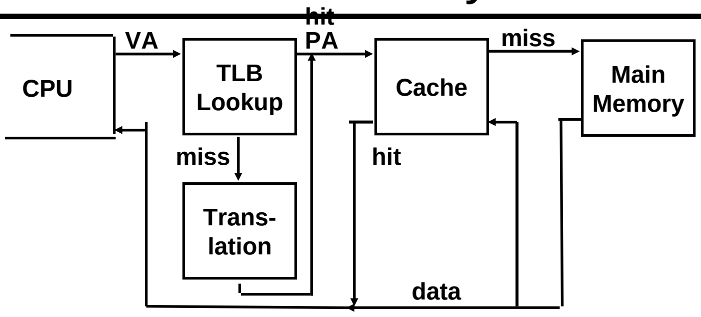
- TLB access time is typically smaller than cache access time (because TLBs are much smaller than caches)
  - Typically not more than 64 to 256 entries
  - CPU pipeline speed: small/fast

A TLB in the Memory Hierarchy



- A TLB miss:
  - If page in main memory, TLB miss can be handled (in hardware or software)
    - Load from the page table into the TLB
    - Takes 10’s of cycles

A TLB in the Memory Hierarchy



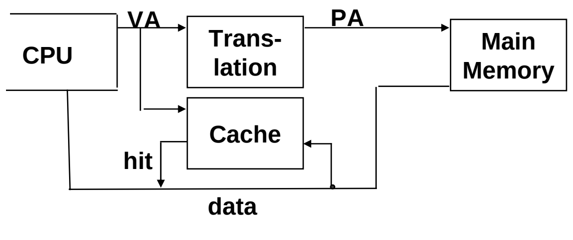
- A TLB miss:
  - If the page is not in main memory, then it’s a true page fault
    - Takes 1,000,000’s of cycles to service a page fault
- TLB misses are much more frequent than true page faults

TLB Validity

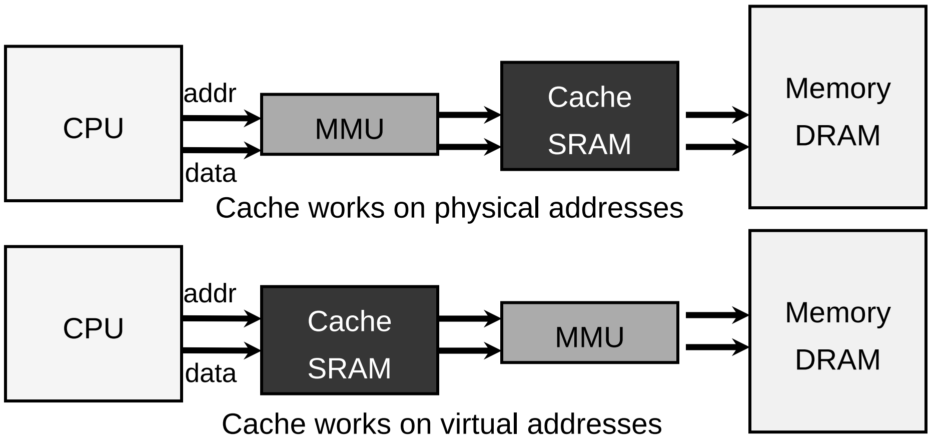
- Keep TLB valid on context switch
  - Flush TLB on context switch (x86)
  - Store process id (MIPs)
- OS has to keep TLB valid

Remove TLB from critical path?

- A virtually addressed cache would only require address translation on cache misses



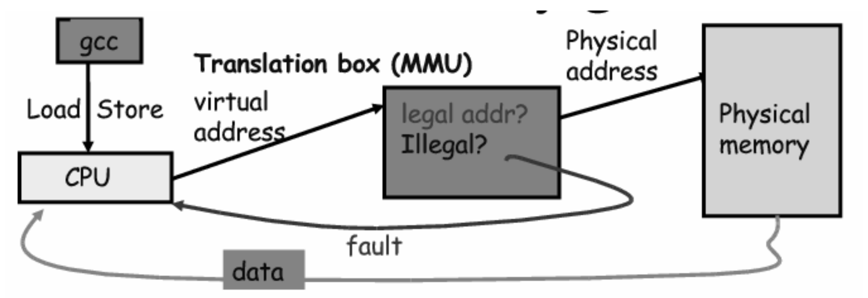
Virtual vs. Physical Caches



- L1 (on-chip) caches are typically virtual
- L2 (off-chip) caches are typically physical

Address Translation

- Translation is done through the page table
  - A virtual memory miss (i.e., when the page is not in physical memory) is called a page fault



Hardware/Software Boundary

- Virtual to physical address translation is assisted by hardware?
  - Translation Lookaside Buffer (TLB) that caches the recent translations
    - TLB access time is part of the cache hit time
    - May allot an extra stage in the pipeline for TLB access
  - TLB miss
    - Can be in software (kernel handler) or hardware

Hardware/Software Boundary

- Virtual to physical address translation is assisted by hardware?
  - Page table storage, fault detection and updating
    - Page faults result in interrupts (precise) that are then handled by the OS
    - Hardware must support (i.e., update appropriately) Dirty and Reference bits (e.g., ~LRU) in the Page Tables

Summary

- Caches, TLBs, Virtual Memory all understood by examining how they deal with the four questions
  1. Where can block be placed?
  2. How is block found?
     - TLB: 4-way set associative or fully associative
  3. What block is replaced on miss?
  4. How are writes handled?
- Page tables map virtual address to physical address
  - TLBs are important for fast translation

Kavita Bala, Computer Science, Cornell University