EE 3613: Computer Organization

Chapter 5: Large and Fast: Exploiting Memory Hierarchy - 2

Avinash Karanth
Department of Electrical Engineering & Computer Science
Ohio University, Athens, Ohio 45701
E-mail: Karanth@ohio.edu
Website: http://oucsace.cs.ohiou.edu/~avinashk/ee461a.htm
Acknowledgement: Mary J. Irwin, PSU; Srinivasan Ramasubramanian, UofA

1

Course Administration

- Homework 6 is posted – 11/20/2020
  - Due Dec 2 by 11:59 PM
- Final exam is scheduled for Monday **December 7th** (asynchronous) – exam will be available from 6 AM to 11:30 PM
  - Bypass Proctortrack; exam will be for 2.5 hours via blackboard
  - Same rules as before (open book, scratch paper upload, etc)
  - All topics included; more emphasis on topics after exam-2

2

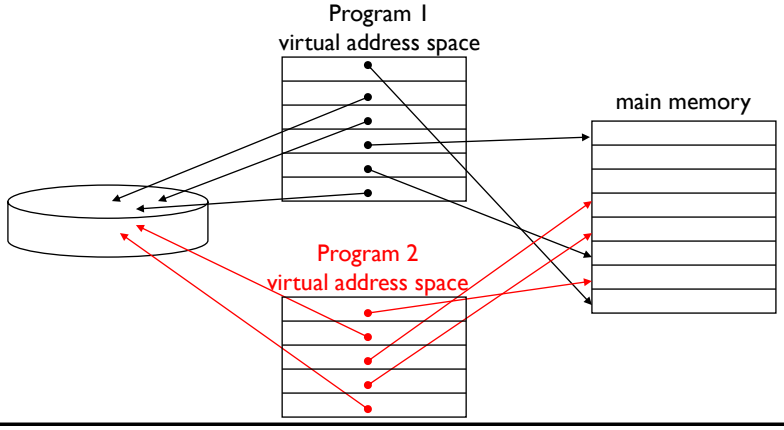
Virtual Memory

- Use main memory as a "cache" for secondary memory
  - Allows efficient and safe sharing of memory among multiple programs
  - Provides the ability to easily run programs larger than the size of physical memory
  - Simplifies loading a program for execution by providing for code relocation (i.e., the code can be loaded anywhere in main memory)
- What makes it work? – again the Principle of Locality
  - A program is likely to access a relatively small portion of its address space during any period of time
- Each program is compiled into its own address space – a "virtual" address space
  - During run-time each virtual address must be translated to a physical address (an address in main memory)

3

Two Programs Sharing Physical Memory

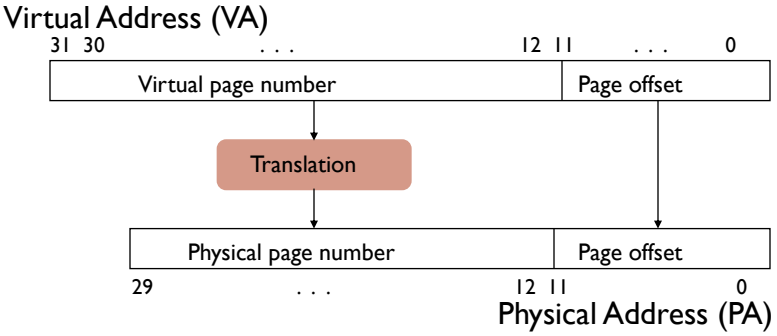
- A program's address space is divided into pages (all one fixed size) or segments (variable sizes)
  - The starting location of each page (either in main memory or in secondary memory) is contained in the program's page table



4

Address Translation

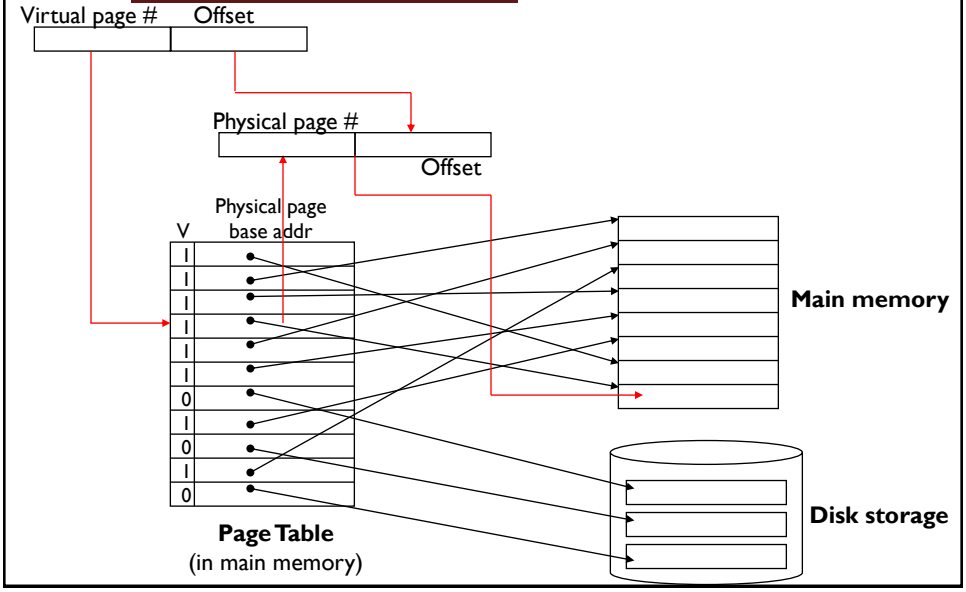
- A virtual address is translated to a physical address by a combination of hardware and software



- So each memory request *first* requires an address translation from the virtual space to the physical space
  - A virtual memory miss (i.e., when the page is not in physical memory) is called a page fault

5

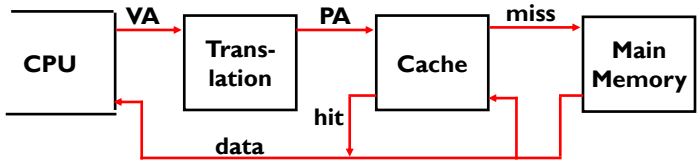
Address Translation Mechanisms



6

Virtual Addressing with a Cache

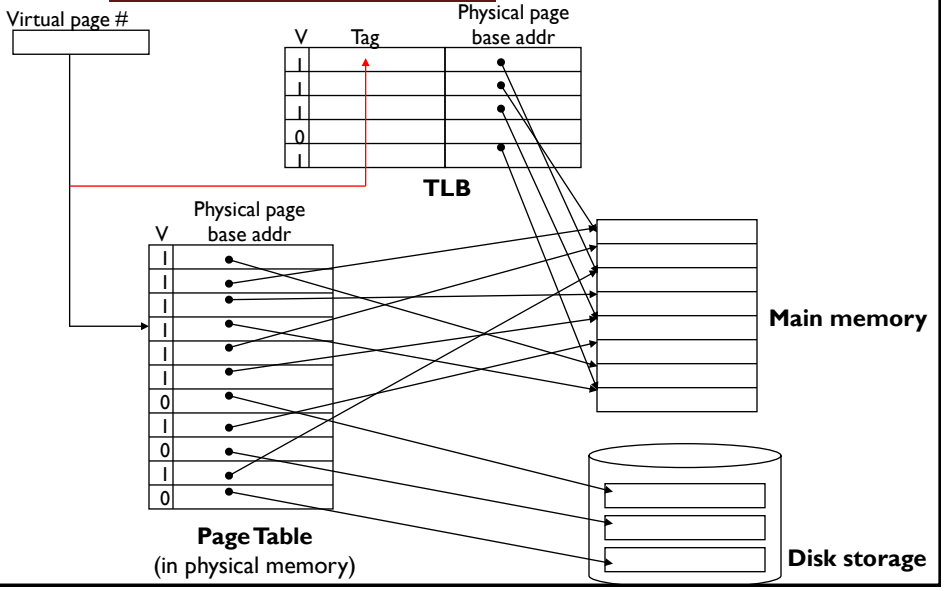
- Thus it takes an *extra* memory access to translate a VA to a PA



- This makes memory (cache) accesses very expensive (if every access was really *two* accesses)
- The hardware fix is to use a Translation Lookaside Buffer (TLB) – a small cache that keeps track of recently used address mappings to avoid having to do a page table lookup

7

Making Address Translation Fast



8

Translation Lookaside Buffers (TLBs)

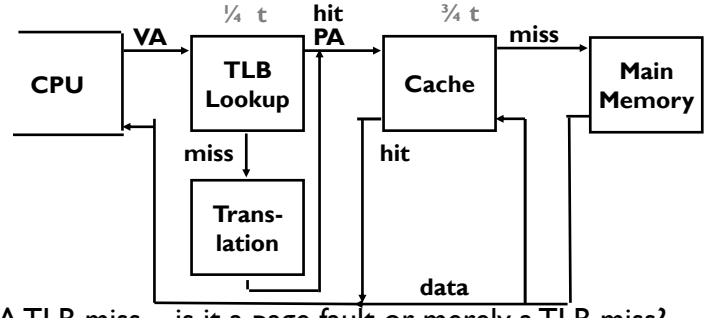
- Just like any other cache, the TLB can be organized as fully associative, set associative, or direct mapped

V	Virtual Page #	Physical Page #	Dirty	Ref	Access

- TLB access time is typically smaller than cache access time (because TLBs are much smaller than caches)
  - TLBs are typically not more than 128 to 256 entries even on high end machines

9

A TLB in the Memory Hierarchy



- A TLB miss – is it a page fault or merely a TLB miss?
  - If the page is loaded into main memory, then the TLB miss can be handled (in hardware or software) by loading the translation information from the page table into the TLB
    - Takes 10's of cycles to find and load the translation info into the TLB
  - 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

10

Some Virtual Memory Design Parameters

	Paged VM	TLBs
Total size	16,000 to 250,000 words	16 to 512 entries
Total size (KB)	250,000 to 1,000,000,000	0.25 to 16
Block size (B)	4000 to 64,000	4 to 32
Miss penalty (clocks)	10,000,000 to 100,000,000	10 to 1000
Miss rates	0.00001% to 0.0001%	0.01% to 2%

11

Two Machines' Cache Parameters

	Intel P4	**AMD Opteron**
TLB organization	1 TLB for instructions and 1TLB for data Both 4-way set associative Both use ~LRU replacement Both have 128 entries TLB misses handled in hardware	2 TLBs for instructions and 2 TLBs for data Both L1 TLBs fully associative with ~LRU replacement Both L2 TLBs are 4-way set associative with round-robin LRU Both L1 TLBs have 40 entries Both L2 TLBs have 512 entries TBL misses handled in hardware

12

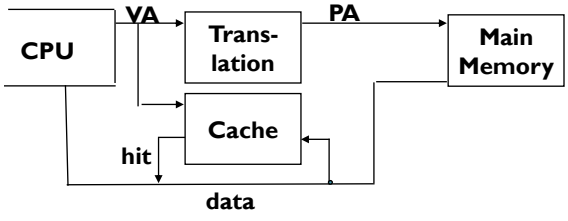
TLB Event Combinations

TLB	Page Table	Cache	Possible? Under what circumstances?
Hit	Hit	Hit	Yes – what we want!
Hit	Hit	Miss	Yes – although the page table is not checked if the TLB hits
Miss	Hit	Hit	Yes – TLB miss, PA in page table
Miss	Hit	Miss	Yes – TLB miss, PA in page table, but data not in cache
Miss	Miss	Miss	Yes – page fault
Hit	Miss	Miss/ Hit	Impossible – TLB translation not possible if page is not present in memory
Miss	Miss	Hit	Impossible – data not allowed in cache if page is not in memory

14

Why Not a Virtually Addressed Cache?

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



but

- Two different virtual addresses can map to the same physical address (when processes are sharing data), i.e., two different cache entries hold data for the same physical address – synonyms
  - Must update all cache entries with the same physical address or the memory becomes inconsistent

15

The Hardware/Software Boundary

- What parts of the virtual to physical address translation is done by or assisted by the 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
  - 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
  - Disk placement
    - Bootstrap (e.g., out of disk sector 0) so the system can service a limited number of page faults before the OS is even loaded

16

Summary

- The Principle of Locality:
  - Program likely to access a relatively small portion of the address space at any instant of time.
    - Temporal Locality: Locality in Time
    - Spatial Locality: Locality in Space
- 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?
  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

17