Lecture 8
Virtual Memory

Fall 2016

Pejman Lotfi-Kamran

Adapted from slides originally developed by Profs. Falsafi, Hill, Hoe, Lipasti, Shen, Smith, Sohi, and Vijaykumar of Carnegie Mellon University, EPFL, Purdue University, and University of Wisconsin.

Where Are We?

Fr	Sa	Su	Mo	Tu
	27-Shahrivar		29-Shahrivar	
	3-Mehr		5-Mehr	
	10-Mehr		12-Mehr	
	17-Mehr		19-Mehr	
	24-Mehr		26-Mehr	
	1-Aban		3-Aban	
	8-Aban		10-Aban	
	15-Aban		17-Aban	
	22-Aban		24-Aban	
	29-Aban		1-Azar	
	6-Azar		8-Azar	
	13-Azar		15-Azar	
	20-Azar		22-Azar	
	27-Azar		29-Azar	
	4-Dey		6-Dey	

This Lecture
- Virtual Memory

Next Lecture:
- DRAM

2 Parts to Modern VM

In a multi-tasking system, VM provides each process with the illusion of a large, private, uniform memory

Part A: Protection
- each process sees a large, contiguous memory segment without holes
- each process's memory space is private, i.e. protected from access by other processes

Part B: Demand Paging
- capacity of secondary memory (swap space on disk)
- at the speed of primary memory (DRAM)

Based on a common HW mechanism: address translation
- user process operates on "virtual" or "effective" addresses
- HW translates from virtual to physical on each reference
  - controls which physical locations can be named by a process
  - allows dynamic relocation of physical backing store (DRAM vs. HD)
- VM HW and memory management policies controlled by the OS

Evolution of Protection Mechanisms

Earliest machines had no concept of protection and address translation
- no need---single process, single user
- automatically "private and uniform" *(but not very large)*
- programs operated on physical addresses directly

no multitasking protection, no dynamic relocation

(at least not very easily)

base and bound registers

In a multi-tasking system

Each process is given a non-overlapping, contiguous physical memory region, *everything belonging to a process must fit in that region*

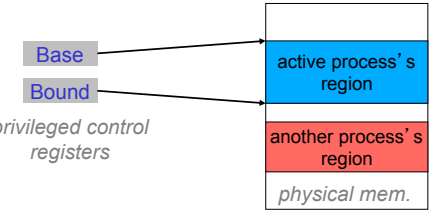
When a process is swapped in, OS sets base to the start of the process's memory region and bound to the end of the region

HW translation and protection check *(on each memory reference)*

PA = EA + base

provided (PA < bound), else violations

⇒ *Each process sees a private and uniform address space (0 .. max)*



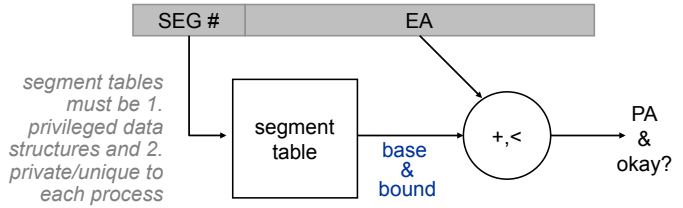
Bound can also be formulated as a range

Segmented Address Space

segment == a base and bound pair

segmented addressing gives each process multiple segments

- initially, separate code and data segments
  - *2 sets of base-and-bound reg's for inst and data fetch*
  - *allowed sharing code segments*
- became more and more elaborate: *code, data, stack, etc.*
- also (ab)used as a way for an ISA with a small EA space to address a larger physical memory space



Paged Address Space

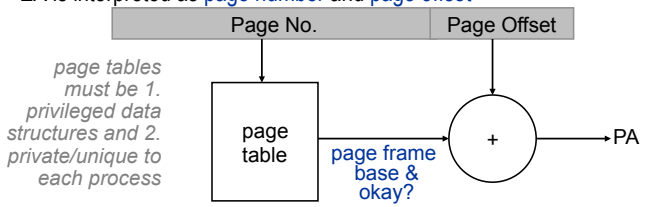
Segmented addressing creates fragmentation problems,

- a system may have plenty of unallocated memory locations
- they are useless if they do not form a contiguous region of a sufficient size

In a Paged Memory System:

PA space is divided into fixed size segments (e.g. 4kbyte),
more commonly known as "page frames"

EA is interpreted as page number and page offset



Demand Paging

Main memory and Disk as automatically managed levels in the memory hierarchies

analogous to cache vs. main memory

Drastically different size and time scales

⇒ very different design decisions

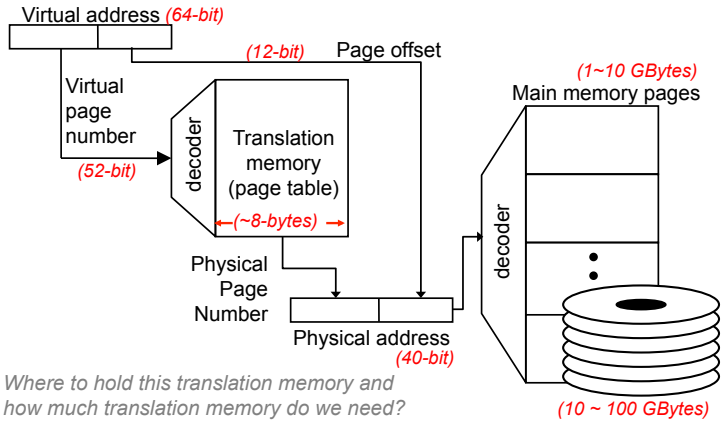
Early attempts

- von Neumann already described manual memory hierarchies
- Brookner's interpretive coding, 1960
  - a software interpreter that managed paging between a 40kb main memory and a 640Kb drum
- Atlas, 1962
  - hardware demand paging between a 32-page (512 word/page) main memory and 192 pages on drums
  - user program believes it has 192 pages

Demand Paging vs. Caching

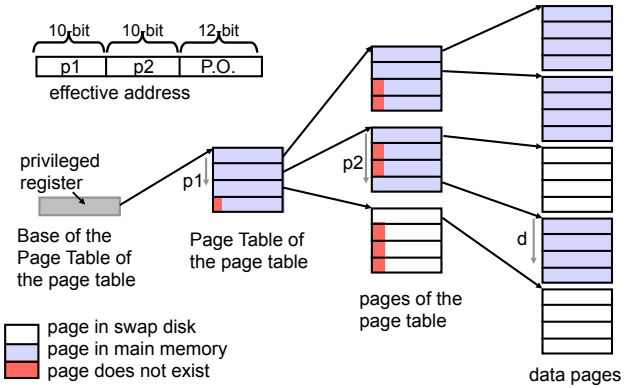
	L1 Cache	Demand Paging
capacity	4KB~64KB	??
block size	16~128 Byte	4K to 64K Byte
hit time	1~3 cyc	50-150 cyc
miss penalty	5~150 cycles	1M to 10M cycles
miss rate	0.1~10%	0.00001~0.001%
hit handling	hw	hw
miss handling	hw	sw

Page-Based Virtual Memory



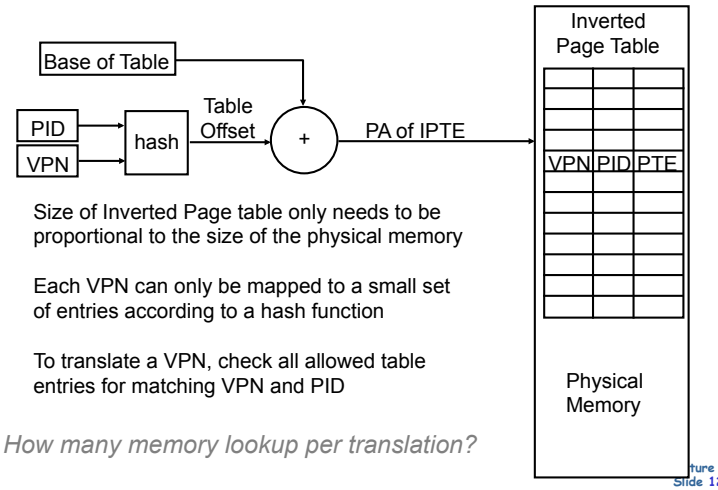
Where to hold this translation memory and how much translation memory do we need?

Hierarchical Page Table



Storage of overhead of translation should be proportional to the size of physical memory and not the virtual address space

Inverted or Hashed Page Tables



Size of Inverted Page table only needs to be proportional to the size of the physical memory

Each VPN can only be mapped to a small set of entries according to a hash function

To translate a VPN, check all allowed table entries for matching VPN and PID

How many memory lookup per translation?

Translation Look-aside Buffer (TLB)

Essentially a cache of

recent address translations

avoids going to the page table

on every reference

indexed by lower bits of

VPN (virtual page #)

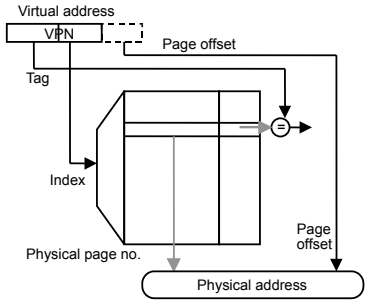
tag = unused bits of VPN +

process ID

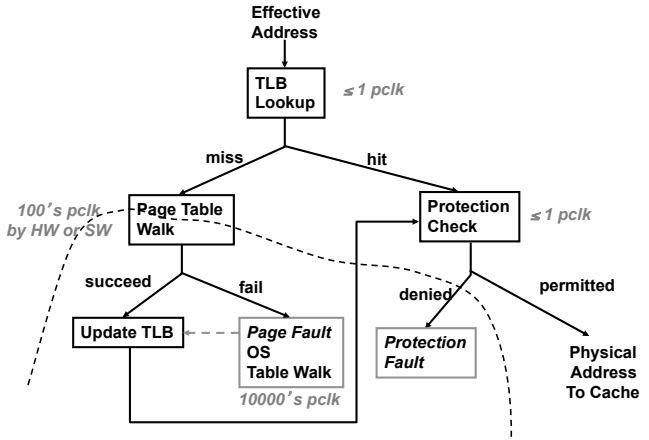
data = a page-table entry

i.e. PPN (physical page #) and

access permission

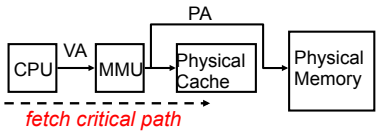


Virtual to Physical Address Translation



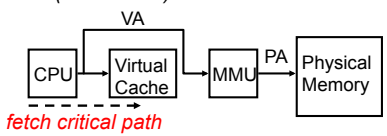
Cache Placement and Address Translation

Physical Cache (Most Systems)



longer hit time

Virtual Cache (SPARC2's)

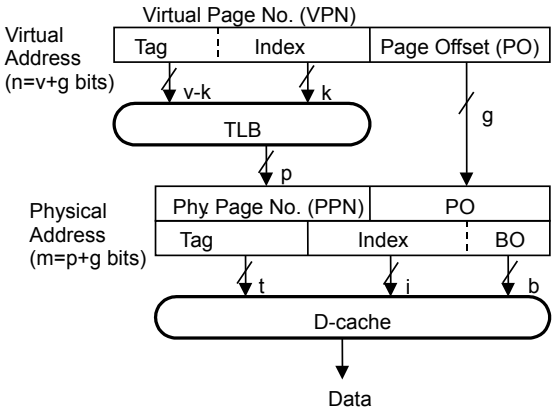


aliasing problem

cold start after context switch

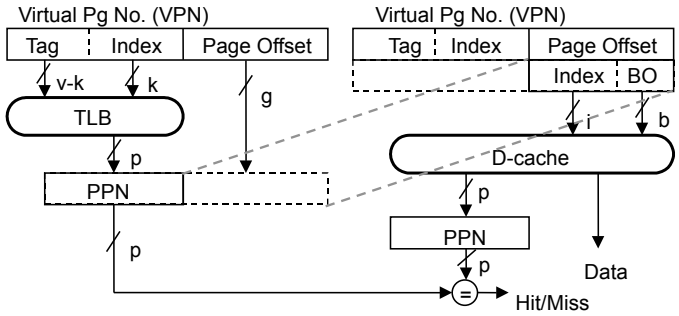
Virtual caches are not popular anymore because MMU and CPU can be integrated on one chip

Physically Indexed Cache



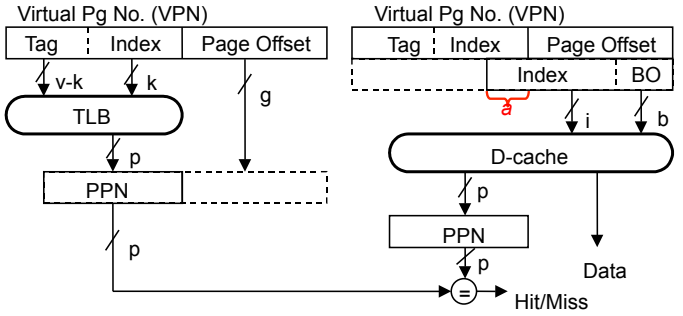
Virtually Indexed Cache

Parallel Access to TLB and Cache arrays



How large can a virtually indexed cache get?

Large Virtually Indexed Cache

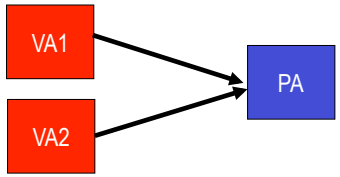


If two VPNs differs in *a*, but both map to the same PPN then there is an aliasing problem

Virtual Address Synonyms

Two virtual pages that map to the same physical page

- within the same virtual address space
- across address spaces



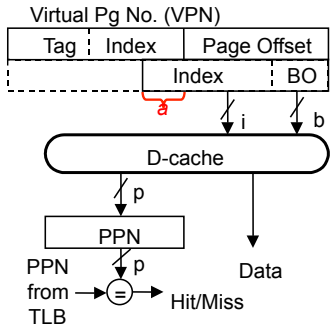
Using VA bits as IDX, PA data may reside in different sets in cache!!

Synonym (or Aliasing)

When VPN bits are used in indexing, two virtual addresses that map to the same physical address can end up sitting in two cache lines

In other words, two copies of the same physical memory location may exist in the cache

⇒ *modification to one copy won't be visible in the other*



*If the two VPNs do not differ in **a** then there is no aliasing problem*

Synonym Solutions

Limit cache size to page size times associativity
- get index from page offset

Search all sets in parallel
- 64K 4-way cache, 4K pages, search 4 sets (16 entries)
- Slow!

Restrict page placement in OS
- make sure index(VA) = index(PA)