EECS 370

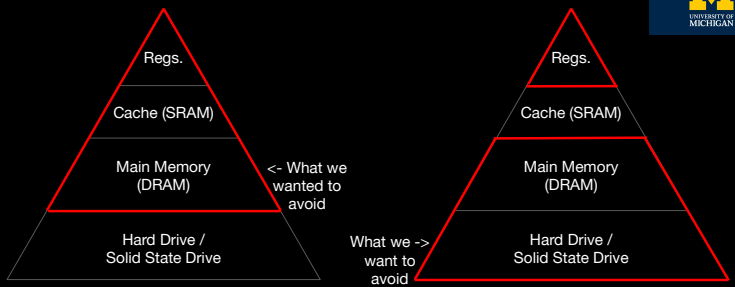
Lab 12:
Cache & Virtual Memory Performance

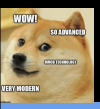
Upcoming Assignments

Lab 12 Assignment Due Wednesday 12/4

Project 4 Due Thursday 12/5

Final Exam Wednesday 12/11 @10:30 a.m.





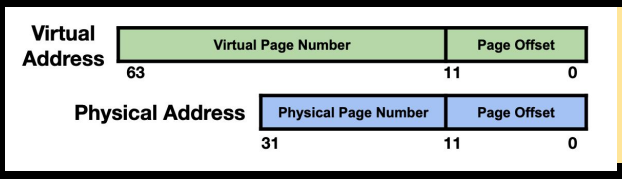
Cache -> Virtual Memory Analogies

Cache	VM
Block	Page
Cache	Phys Mem
Phys Mem	Virtual Address Space/Disk
Tag	VPN
Block Offset	Page Offset
"Cache Block #"	PPN
Cache Miss	Page Fault

VM Has a Key Difference from a Cache: Must Maintain a Mapping

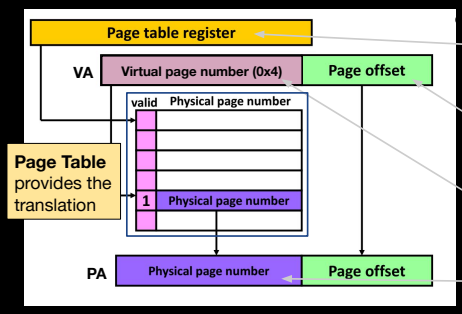
Programs think they can access all 64- or 32-bits of addressable memory

Must provide a mapping/translation of ***virtual addresses*** (where the programs *think* they can access) to a ***physical address*** (real addresses in memory)



Page offset is used to index into a page, much like a *block offset*

Page Tables Provide the Mapping as an Array



Page Table provides the translation

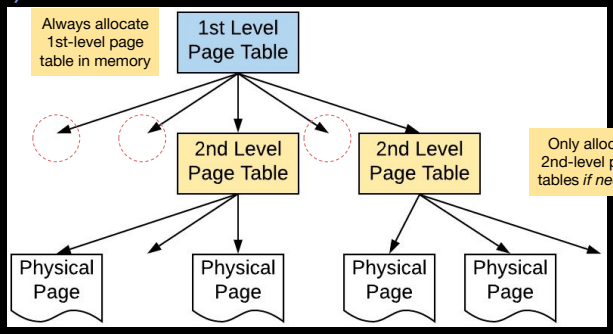
Page Table Register tells the OS where to find the page table

Page Offset is the same for VA and PA (easy translation)

Virtual Page Number comes from the program's mem. addr.

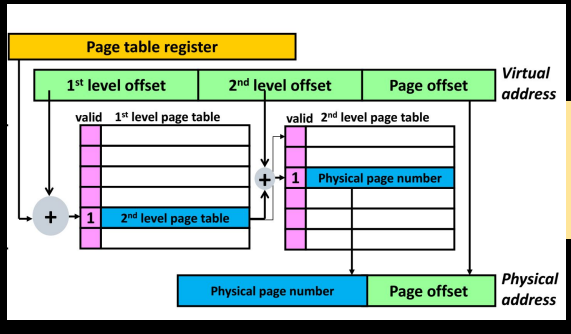
Physical Page Number is the actual page location in memory

Multi-Level PTs Save Space (in the Common Case)



Only allocate 2nd-level page tables *if needed*

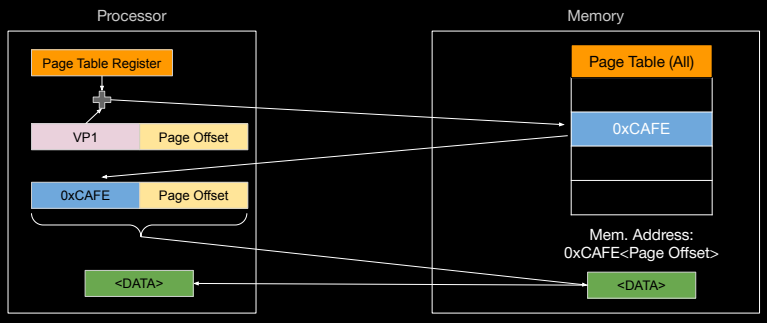
How to Index into a Multi-Level Page Table



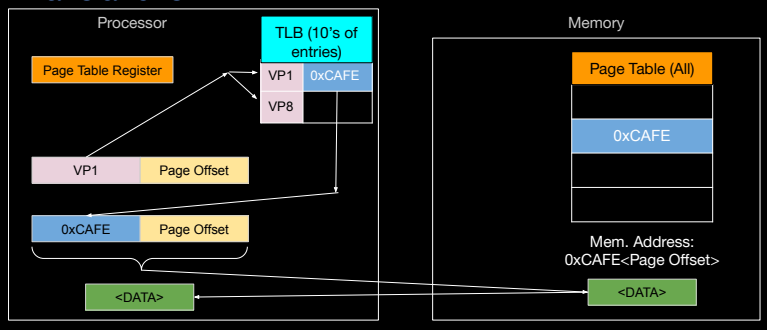
Same process as single-level, but with additional steps

Think of our page tables like a tree with pointers to PT/Pages

TLB is a Cache for Virtual-to-Physical Translations



TLB is a Cache for Virtual-to-Physical Translations



Translations Also Cause a Change in our Cache

The VM system gives us two memory addresses:
a virtual memory address and a physical memory address

We therefore have two types of caches:

Virtually Addressed Cache: Cache *before* Virtual Memory translation

Physically Addressed Cache: Cache *after* Virtual Memory translation

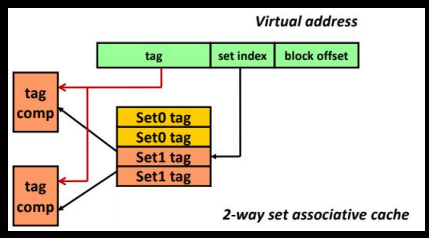
VA Caches Cause TLB Lookup After Comparison

Use bits from **virtual** address for set index and tag

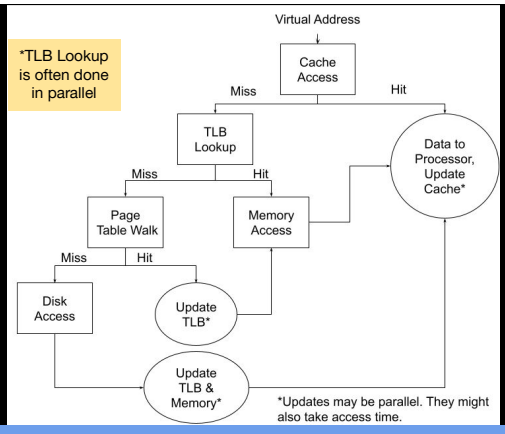
Two processes may refer to the same physical address with two different virt. addrs.

Cache is invalidated when switching processes

Much less common than physically addressed caches



Virtually Addressed Cache Access Tree

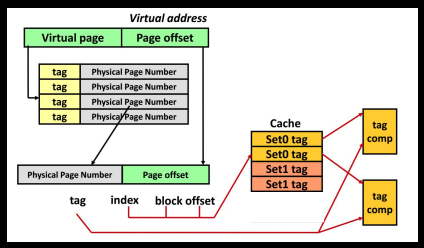


PA Caches Cause TLB Lookup Before Comparison

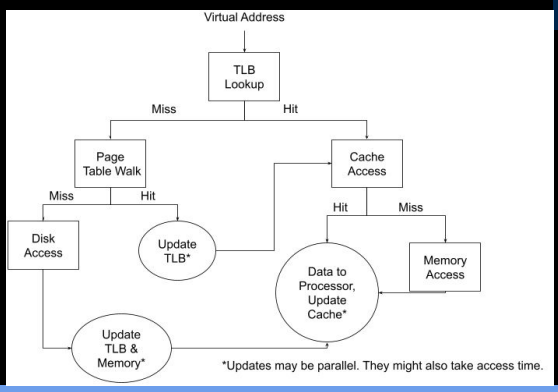
Use bits from **physical** address for set index and tag

Slower access than virtually addressed caches due to TLB lookup

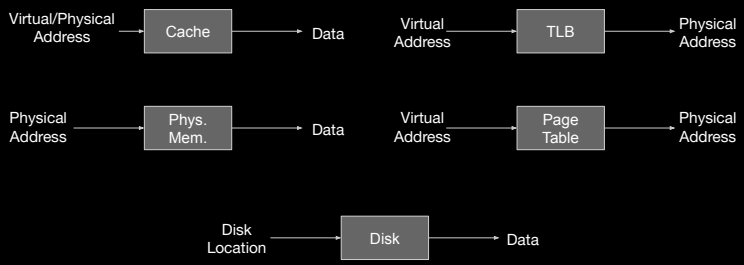
Cache is not invalidated when switching processes



Physically Addressed Cache Access Tree



Memory System Components



Virtual Memory Terminology

Extra Slides: For your reference

- **Disk** - REALLY SLOW MEMORY, but also really big
- **Virtual Address Space** - All of a program's addressable memory
- **Virtual Address** - Address a program uses to access its data. A "fake"address. Translated into a Physical Address
- **Physical Address** - Address that the OS uses to index into Physical Memory to get a program's data. The "real" address. Translated from Virtual Address. ($\log_2$(Physical Memory Size))
- **Page** - A chunk of memory
- **Virtual Page** - A page within a program's virtual address space. Always exists on the Disk*, sometimes in Physical Memory. Number of virtual pages = $2^{\text{VPN Bits}}$
- **Physical Page** - A page within physical memory. "Container for Virtual Pages"
- **Physical Memory Size:** Physical Page Size * #Physical Pages
- **Virtual Page Number** - The ID/label/"name"/number for each Virtual Page
  - VPN Bits: Virtual Address size - page offset bits
- **Physical Page Number** - The ID/label/"name"/number for each Physical Page
  - PPN bits: Physical Page Address Size - page offset bits
- **Page Offset** - Index/offset within a Virtual or Physical Page. (*Log_2(Page Size)*)
- **Page Table** - Stores all translations for Virtual Page Numbers to Physical Page Numbers
- **Page Fault** - The virtual page a program tried to access isn't in Physical Memory/doesn't have a valid entry in the Page Table. We need to bring the page from Disk into Physical Memory.

Formulas!

cheat sheet!

- **Page Offset Bits** = $\log_2$(Page Size)
- **VPN Bits** = Virtual Address Size - Page Offset Bits
- **PPN Bits** = Physical Address Size - Page Offset Bits
- **Physical Address Size** = $\log_2$(Physical Memory Size)
- **#Virtual Pages** = $2^{\text{VPN Bits}}$
- **Physical Memory Size** = Physical Page Size * #Physical Pages
- **#Physical Pages** = $2^{\text{PPN Bits}}$
- **#Page Table Mappings** = #Virtual Pages

Formulas! (cont.)

cheat sheet!

- **Size of Page Table** = #Entries * Size of an Entry

Single Level Page Table:

- **#Entries** = #Mappings = #Virtual Pages
- **Size of Page Table Entry** = Size of Control Bits + PPN bits

Multi Level Page Table:

- **Total #Entries in Leaf Level** = #Mappings = #Virtual Pages
- **Size of Leaf Level Page Table Entry** = Size of Control Bits + PPN bits
- **Total #Entries in Intermediate Level** = #Page Tables in Next Level Down
- **Size of Intermediate Level Page Table Entry** = Size of Control Bits + PPN bits/Physical Address Size

EECS 370

Lab 12:
Cache & Virtual Memory Performance