CS356 Unit 9

Virtual Memory & Address Translation

Indirection

- Indirection means using one entity to _____
- Examples:
  - A variable name vs. it's _____
  - _____ vs. cell tower location/phone ID
  - Titles like "CEO" or "head coach" are virtual titles that can be applied to different people at different times
- The benefits are we can change one without changing the other
  - We can change the underlying implementation without changing the higher level task. For example, a job description would read "The CEO shall perform this duty or that." and it need not be changed if the company replaces John Doe with Jane Doe.
- "All problems in computer science can be solved by another level of indirection" – attributed to David Wheeler

Virtual Memory & Address Translation

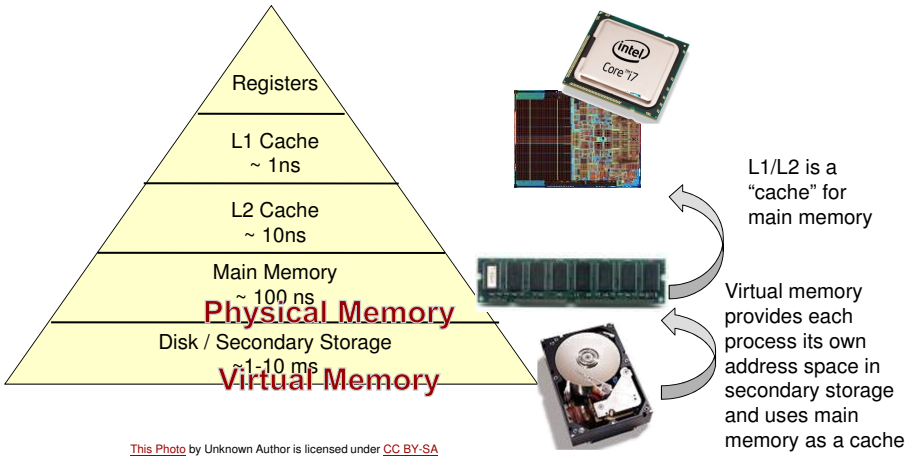
- We are going to indirect the addresses used by computer programs
- Primary Idea = Compile the program with _____addresses and have a _____ convert these to physical addresses as the program runs (this is Address Translation)
  - Efficiently _____ the physical memory between several running programs/processes and provide _____ from accessing each others' information
- Secondary Idea = Use main memory (MM) as a "_____" for multiple programs' data as they run, using _____ as the home location (this is Virtual Memory)
  - Remove the need of the programmer to know how much memory is physically present and/or give the illusion of _____ physical memory than is present
- These ideas are often used interchangeably

Benefits of Address Translation

- What is enabled through virtual memory and address translation?
  - Illusion of more or less memory than physically present
  - Isolation
  - *Controlled sharing of code or data
  - *Efficient I/O (memory-mapped files)
  - *Dynamic allocation (Heap / Stack growth)
  - *Process migration
- *Will be discussed in a subsequent unit or Operating Systems class

Memory Hierarchy & Caching

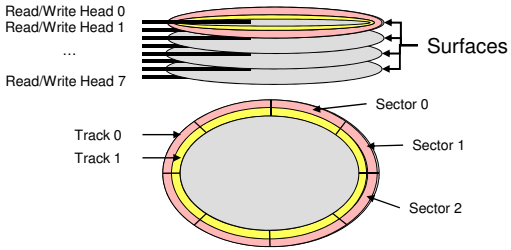
- Lower levels act as a cache for upper levels



Secondary Storage: Magnetic Disks

- Magnetic hard drive consists of
  - Double sided surfaces/platters (with R/W head)
  - Each platter is divided into concentric tracks of small sectors that each store several thousand bits
- Performance is slow primarily due to moving _____

• **Seek Time: Time needed to position the read-head above the proper track**	_____
• **Rotational delay: Time needed to bring the right sector under the read-head**	_____
◦ **Depends on rotation speed (e.g. 5400 RPM)**	
• **Transfer Time:**	**0.1 ms**
• **Disk Controller Overhead:**	**+ 2.0 ms**
	~20 ms



Secondary Storage: Flash

- Flash (_____) drives store bits using special transistors that retain their values even when power is turned off
- Performance is higher than magnetic disks but still slower comparted to main memory
  - Better sequential read throughput
    - HD (Magnetic): 122-204 MB/s
    - SSD: 210-270 MB/s
  - MUCH better random read
    - Max latency for single read/write: 75us
    - When many requests present we can overlap and achieve latency of around 26us (1/38500)
- Flash drives "_____" after some number of writes/erasures

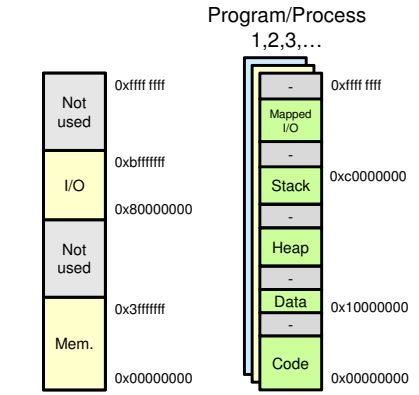
Size	
Capacity	300 GB
Page Size	4 KB
Performance	
Bandwidth (Sequential Reads)	270 MB/s
Bandwidth (Sequential Writes)	210 MB/s
Read/Write Latency	75 µs
Random Reads Per Second	38,500
Random Writes Per Second	2,000 2,400 with 20% space reserve
Interface	SATA 3 Gb/s
Endurance	
Endurance	1.1 PB 1.5 PB with 20% space reserve
Power	
Power Consumption Active/Idle	3.7 W / 0.7 W

OS:PP 2nd Ed. Fig. 12.6

Intel 710 SSD specs.

Address Spaces

- Physical address spaces corresponds to the actual system address _____ (based on the _____ of the address bus) of the processor and how much main memory is physically present
- Each process/program runs in its own private "virtual" address space
  - Virtual address space can be larger (or smaller) than physical memory
  - Virtual address spaces are _____ from each other

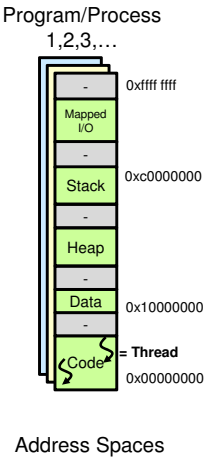


32-bit Physical Address Space w/ only 1 GB of Mem

32-bit Fictitious Virtual Address Spaces (> 1GB Mem)

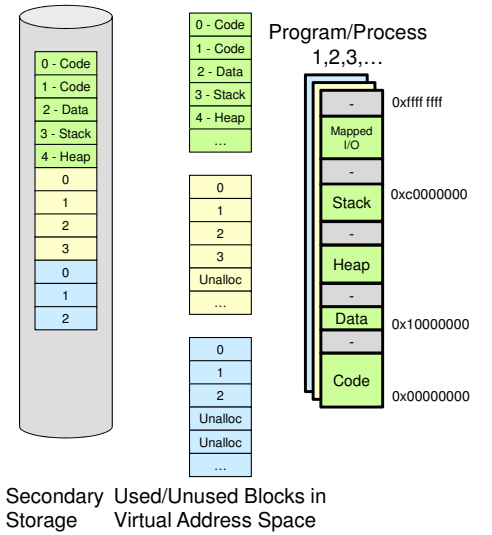
Processes

- **Process**
  - **(def 1.)** _____
    - (Virtual) Address Space = Protected view of memory
    - 1 or more threads
  - **(def 2.) : Running** _____ **that has** _____
    - Memory is protected: Address translation (VM) ensures no access to any other processes' memory
    - I/O is protected: Processes execute in user-mode (not kernel mode) which generally means direct I/O access is disallowed instead requiring **system calls** into the kernel
- OS Kernel is not considered a "process"
  - Has access to all resources and much of its code is invoked under the execution of a user process thread



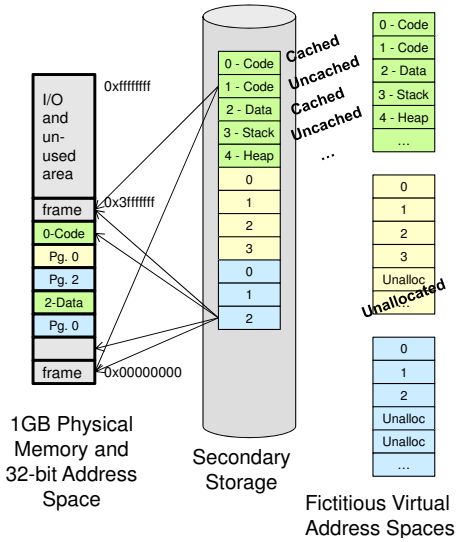
Virtual Address Spaces (VAS)

- Virtual address spaces (VASs) are broken into blocks called "_____"
- Depending on the program, much of the virtual address space will be _____
- Pages can be allocated "_____" (i.e. when the stack grows, etc.)
- All allocated pages can be stored in secondary storage (hard drive)



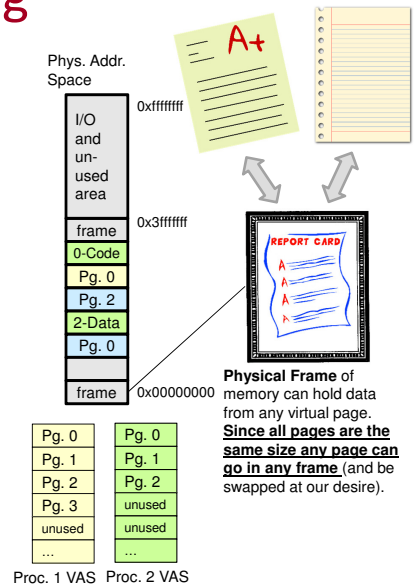
Physical Address Space (PAS)

- Physical memory is broken into page-size blocks called "_____"
- Multiple programs can be running and their pages can _____ the physical memory
- Physical memory acts as a _____ for pages with secondary storage acting as the backing store (next lower level in the hierarchy)
- A page can be:
  - _____ (not needed yet…stack/heap)
  - Allocated and residing in secondary storage (_____)
  - Allocated and residing in main memory (_____)



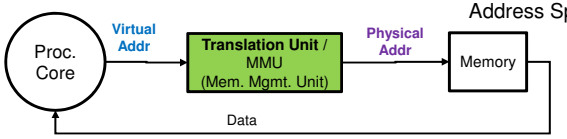
Paging

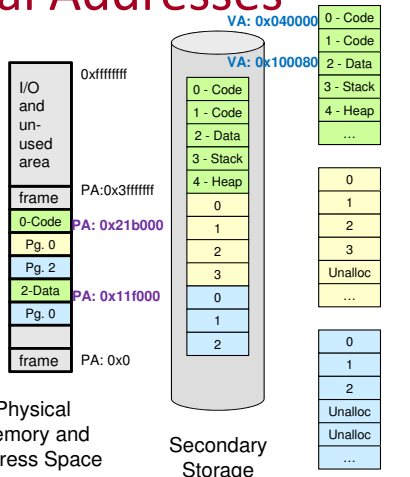
- Virtual address space is divided into equal size "pages" (often around 4KB)
- Physical memory is broken into page frames (which can hold any page of virtual memory and then be swapped for another page)
- Virtual address spaces can be _____ while physical layout is not



Virtual vs. Physical Addresses

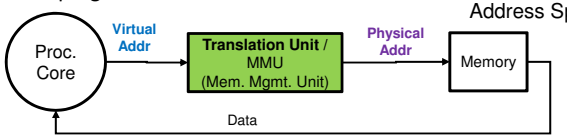
- **Key**: Programs are written using **virtual addresses**
- HW & the OS will _____ the virtual addresses used by the program to the **physical address** where that page resides
- If an attempt is made to access a page that is not in physical memory, ____ generates a "_____ exception" and the OS is invoked to bring in the page to physical memory (possibly evicting another page)
- Notice: Virtual addresses are not _____
  - Each program/process has VA: 0x00000000

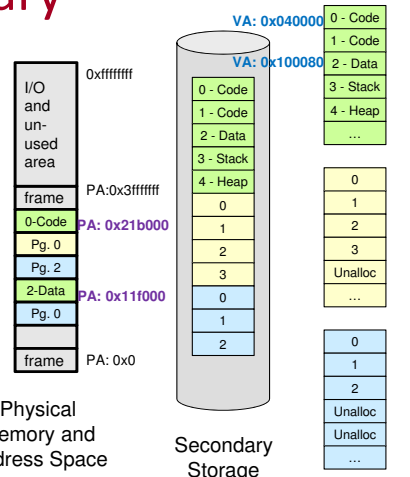




Summary

- Program takes an abstract (virtual) view of memory and uses virtual addresses and necessary data is broken into large chunks called pages
- HW and OS work together to bring pages into main memory acting as a cache and allowing sharing
- HW and OS work together to perform translation between:
  - **Virtual address**: Address used by the process (programmer)
  - **Physical address**: Physical memory location of the desired data
- Translation allows protection against other programs





VM Design Implications

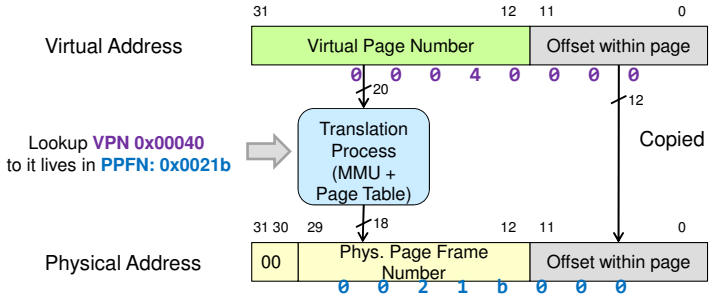
- SLOW secondary storage access on page faults (100us - 10ms)
  - Implies page size should be fairly _____ (i.e. once we've taken the time to find data on disk, make it worthwhile by accessing a reasonably large amount of data)
  - Implies the placement of pages in main memory should be _____ to reduce conflicts and maximize page hit rates
  - Implies a "page fault" is going to take so much time to even access the data that we can handle them in _____ (via an exception) rather than using _____ like typical cache misses
  - Implies we should use a _____ policy for pages (since _____ would be too expensive)

Page Tables

ADDRESS TRANSLATION

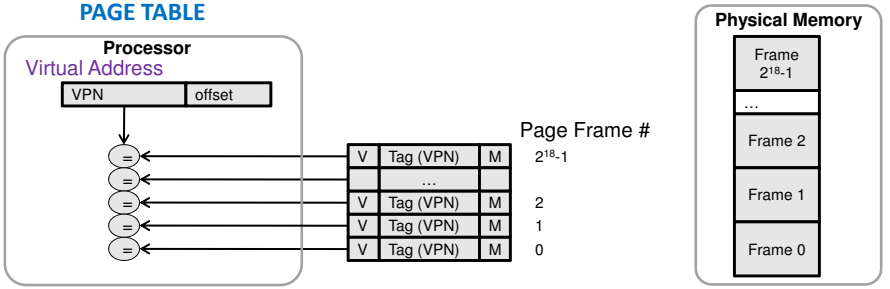
Page Size and Address Translation

- Since pages are usually retrieved from disk, we size them to be fairly large (several KB) to amortize the large access time
- Virtual page number to physical page frame translation performed by HW unit = _____ (Mem. Management Unit)
- _____ is an **in-memory** data structure that the HW MMU will use to look up translations from VPN to PPFN



Address Translation Issues

- We want to take advantage of all the physical memory so page placement should be fully associative
  - For 1GB of physical memory, a 4KB page can be anywhere in the _____ page frames
- We could potentially track the contents of physical memory using similar techniques to cache
  - TAG = VPN that is currently stored in the frame
  - This would be _____ tags to check
- Instead, most systems implement full associativity using a **look-up table** = **PAGE TABLE**



Analogy for Page Tables

- Suppose we want to build a caller-ID mechanism for your contacts on your cell phone
  - Let us assume 1000 contacts represented by a 3-digit integer (0-999) in the cell phone (this ID can be used to look up their names)
  - We want to use a simple array (or Look-Up Table (LUT)) to translate phone numbers to contact ID's, how shall we organize/index our LUT

1. LUT indexed w/ contact ID

000	213-745-9823
001	626-454-9985
002	818-329-1980
…	…
999	323-823-7104

O(__) - _____ Work
We are given phone # and need to translate to ID
(_____ accesses)

2. Sorted LUT indexed w/ used phone #'s

213-730-2198	436
213-745-9823	000
323-823-7104	999
…	…
818-329-1980	002

O(_____) - _____ Work
Since its in sorted order we could use a binary search
(_____ accesses)

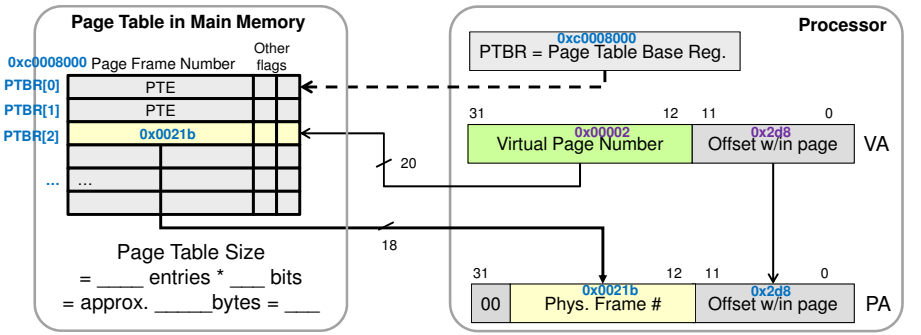
3. LUT indexed w/ all possible phone #'s

000-000-0000	null
..	
213-745-9823	000
…	…
999-999-9999	null

O(_) - _____ Work
Easy to index & find but
_____
(__ access)

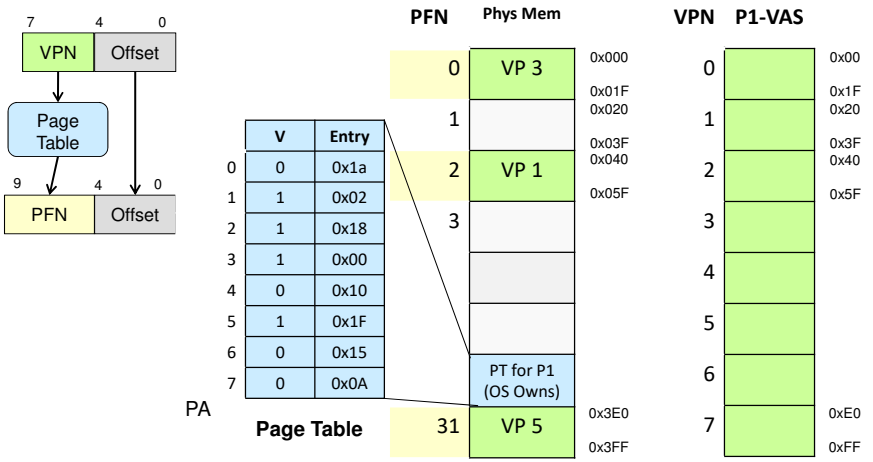
Page Tables

- VA is broken into:
  - VPN (upper bits) + Page offset: Based on page size (i.e. 0 to 4K-1 for 4KB page)
- MMU uses VPN & _____ to access the page table in memory and lookup physical frame (i.e. like an array access where VPN is the index: PTBR[VPN])
  - Each entry is referred to as a _____ (PTE) and holds the physical frame number & bookkeeping info
- Physical frame is combined with offset to form physical address
- For 20-bit VPN, how big is the page table? (See below)



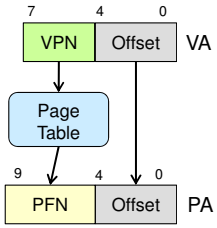
Page Table Example

- Suppose a system with 8-bit VAs, 10-bit PAs, and 32-byte pages.



Page Table Exercise

- Suppose a system with 8-bit VAs, 10-bit PAs, and 32-byte pages.
- Fill in the table below showing the corresponding physical or virtual address based on the other. If no translation can be made, indicate "INVALID"



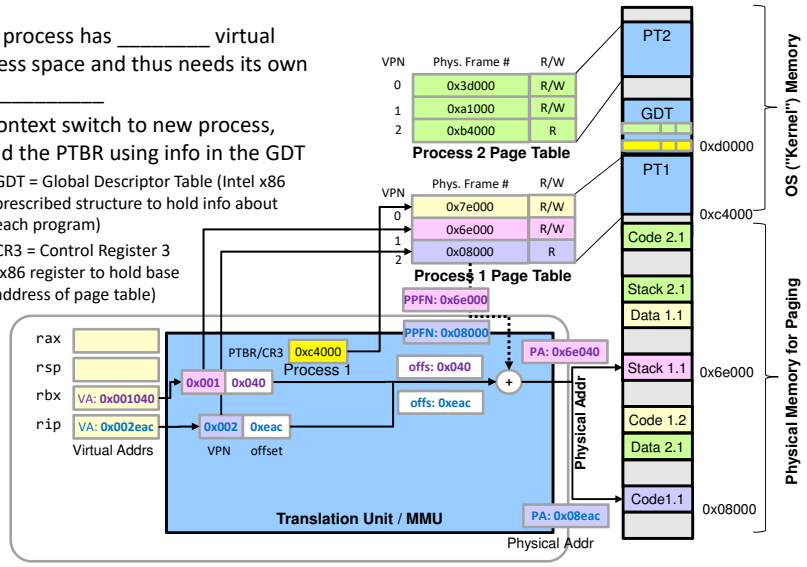
VA	PA
0x2D = 0010 1101	
	0x0DA=0011011010
0xEF = 1110 1111	
0xA8 = 1010 1000	

	V	Entry
0	0	0x0E
1	1	0x1E
2	1	0x16
3	1	0x06
4	0	0x0B
5	1	0x1F
6	0	0x15
7	0	0x0A

Page Table

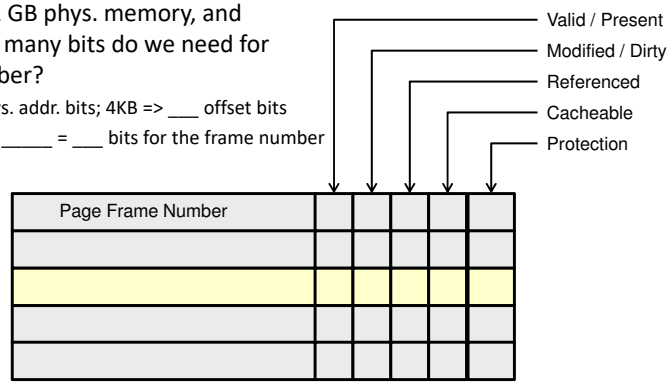
Paging

- Each process has _____ virtual address space and thus needs its own _____
- On context switch to new process, reload the PTBR using info in the GDT
  - GDT = Global Descriptor Table (Intel x86 prescribed structure to hold info about each program)
  - CR3 = Control Register 3 (x86 register to hold base address of page table)



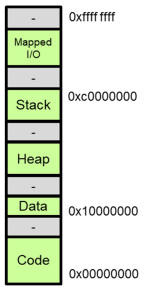
Page Table Entries (PTEs)

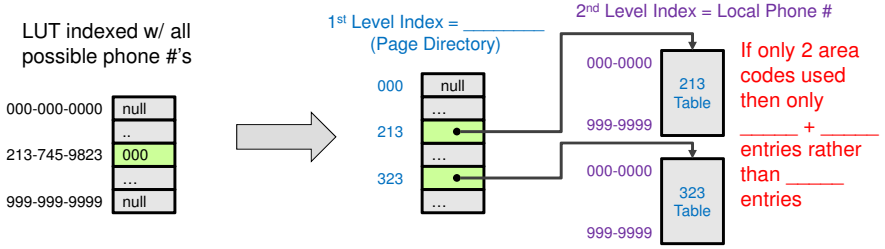
- Usually fits within a 32-bit (4-byte) or 64-bit (8-byte) value:
  - Valid bit (1 = desired page in memory / 0 = page not present / page fault)
  - Modified/Dirty
  - Referenced = To implement _____
  - Protection: _____
- For 32-bit VA, 1 GB phys. memory, and 4KB pages how many bits do we need for the frame number?
  - 1GB = ___ phys. addr. bits; 4KB => ___ offset bits
  - Thus we need _____ = ___ bits for the frame number



Multi-level Page Table Concept

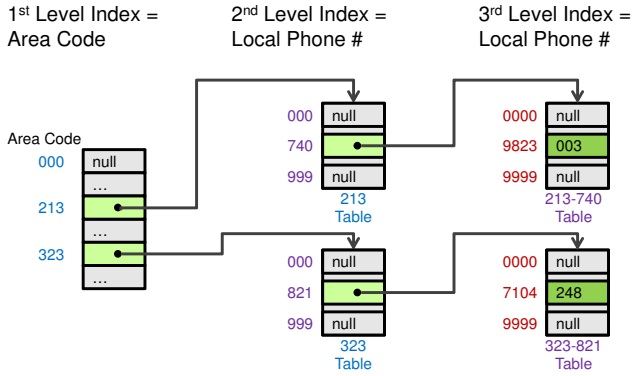
- Much of the VAS is often unused (gray areas in the image on the right) which implies many of the page table entries would be unused
- Can we reduce the page table size and still do a lookup in _____ time?
  - Do you have friends from every _____?
  - Likely contacts are clustered in only a few.
- Use a 2-level organization
  - 1st level LUT is indexed on _____ and contains pointers to 2nd level tables
  - 2nd level LUT's indexed on local phone numbers and contains contact ID entries
- The first level is often called the **page directory** and while the 2nd level is the called page tables
  - PDE's (Page Directory Entries) contain pointers to 2nd level Page Tables





Analogy for Page Tables

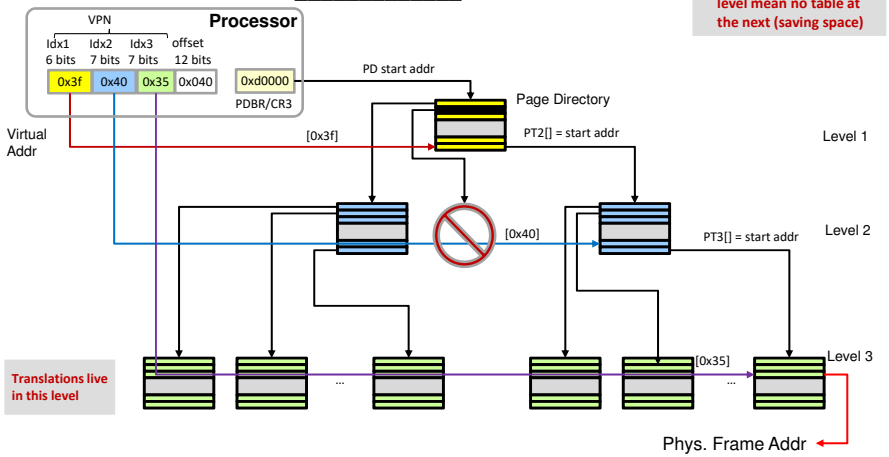
- Could extend to 3 levels if desired
  - 1st Level: Indices are area codes and values are pointers to 2nd level tables
  - 2nd Level: Indices are first 3-digits of local phone and values are pointers to 3rd level tables
  - 3rd Level: Indices are last 4-digits of local phone and values are contact ID's (i.e. Translations)



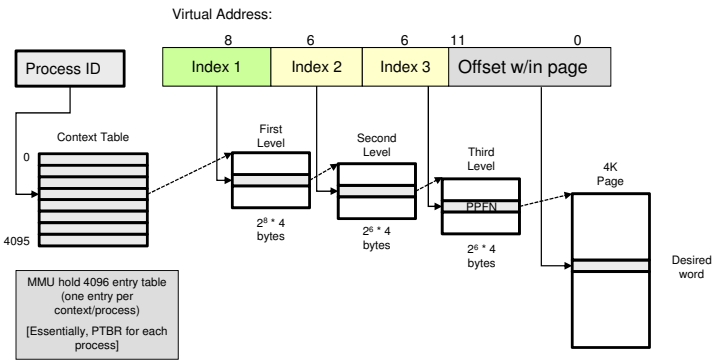
Multi-level Page Tables

- Think of a multi-level page table as a _____
  - Internal nodes contain _____ to other page tables
  - Leaves hold actual _____

- **Unused entries in one level mean no table at the next (saving space)**



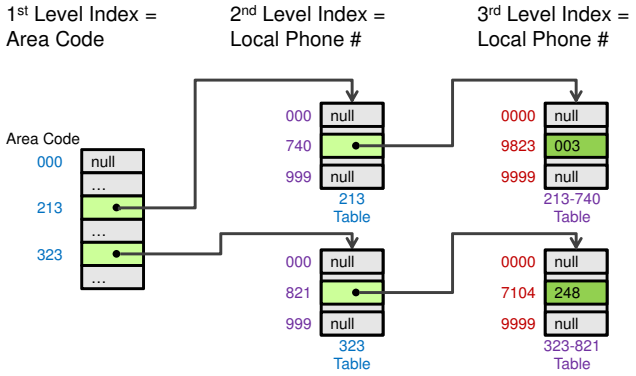
SPARC Processor VM Implementation



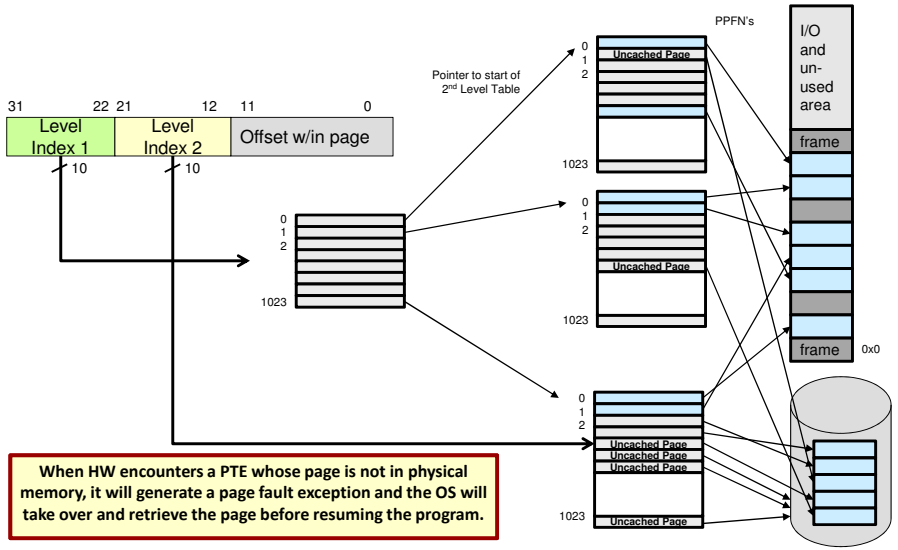
How many accesses to memory does it take to get the desired word that corresponds to the given virtual address? Would that change for a 1- or 2- level table?

Analogy for Page Tables

- If we add a friend from area code 408 we would have to add a second and third level table for just this entry.
- If we had 1 friend from every area code and every 3-digit local prefix, would this scheme save us any storage? No!



Page Faults



Page Fault Steps

- What happens when you reference a page that is not present?
- HW will…
  - Record the offending address and generate a page fault exception
- SW (the OS) will…
  - Pick an empty frame or _____
  - Writeback the evicted page if it has been _____
    - May block process while waiting and _____
  - Bring in the desired page and _____
    - May block process while waiting and _____
  - Restart the offending instruction
- Key Idea: Handler can bring in the page or do *anything appropriate* to handle the page fault
  - Allocate a new page, zero it out, retrieve from secondary storage, etc.

Page Replacement Policies

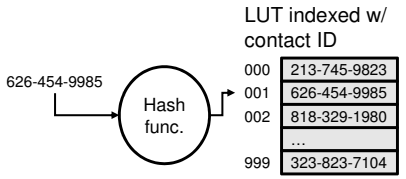
- Possible algorithms: LRU, FIFO, Random
- Since page misses are so costly (slow) we can afford to spend sometime keeping statistics to implement pseudo-LRU
- HW will implement simple mechanism that allows SW to implement a pseudo-LRU algorithm
  - HW will set the "_____" bit when a page is used
  - At certain intervals, SW will use these reference bits to keep statistics on which pages have been used in that interval and then _____ the reference bits
  - On replacement, these statistics can be used to find the pseudo-LRU page
- Other simpler replacement algorithms (e.g. variants of the **clock algorithm**) might also be used

Cache & VM Comparison

	Cache	Virtual Memory
Block Size	16-64B	4 KB – 64 MB
Mapping Schemes	_____	_____
Miss handling and replacement	_____	_____
Replacement Policy	Full LRU if low associativity / Random is also used	Pseudo-LRU can be implemented

Inverted Page Tables

- Page tables may seem expensive in terms of memory overhead
  - Though they really aren't that big
- One option to consider is an "inverted" page table
  - One entry per physical frame
  - Hash the virtual address and whatever results is where that page must reside
- What about collisions?
  - Becomes hard to maintain in hardware, but can be used by secondary software structures

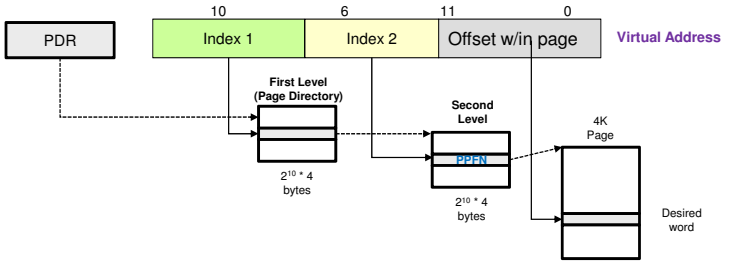


Achieving faster translations…

TLB (TRANSLATION LOOKASIDE BUFFERS)

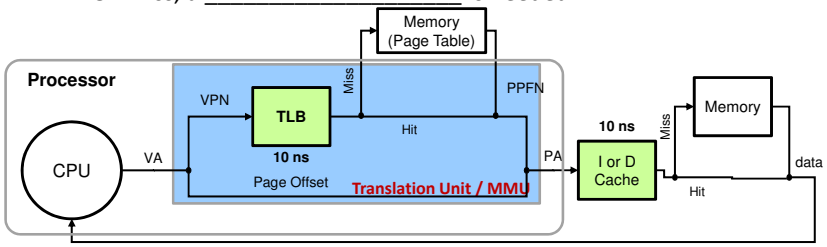
Page Table Performance

- How many accesses to memory does it take to get the desired word that corresponds to the given virtual address?
- So for each needed memory access, we need _____ additional?
  - That sounds BAD!
- Would that change for a 1- or 3- level table?
- M-level page table may require _____ memory accesses to find the translation…EXPENSIVE!!



Translation Lookaside Buffer (TLB)

- Solution: Let's create a _____ for translations = Translation Lookaside Buffer (TLB)
- Needs to be small (64-128 entries) so it can be _____, with high degree of associativity (at least 4-way and many times fully associative) to avoid conflicts
  - On hit, the PPFN is produced and concatenated with the offset
  - On miss, a _____ is needed

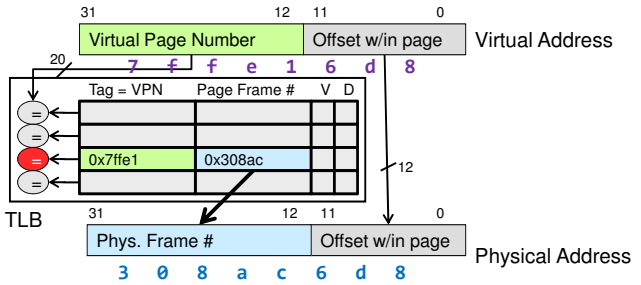


Translation Lookaside Buffer (TLB)

- T(Translation): T(TLB lookup) + (1-P(TLB hit)) * T(PT Walk)
- What is P(TLB hit)?
  - Suppose 4KB page size and that we are walking an array of integers in sequential order
  - What fraction of accesses will be misses in the TLB?
  - _____
- Below is a fully associative TLB diagram

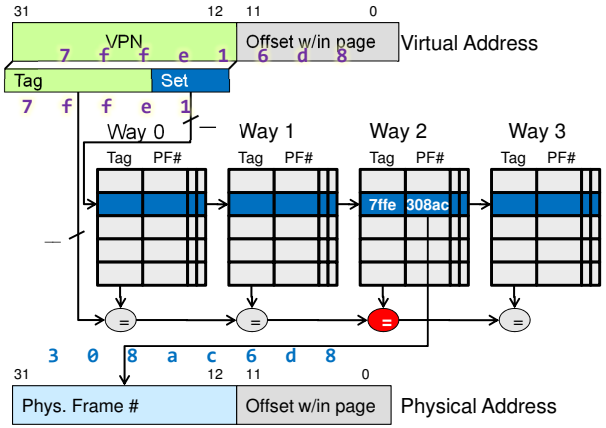
Fully Associative TLB

(Entry can be anywhere and thus we must check all locations in TLB for a hit)

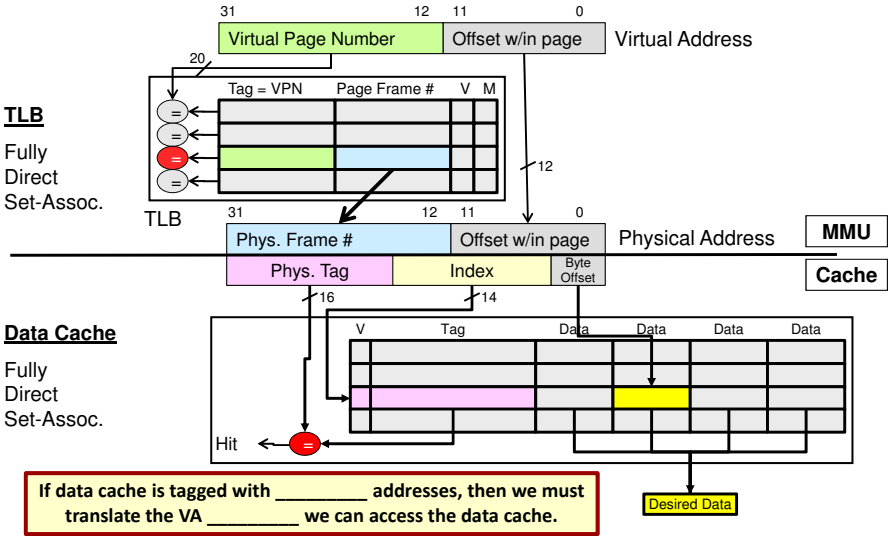


A 4-Way Set Associative TLB

- 64 entry 4-way SA TLB (set field indexes each "way")
  - On hit, page frame # supplied quickly w/o page table access

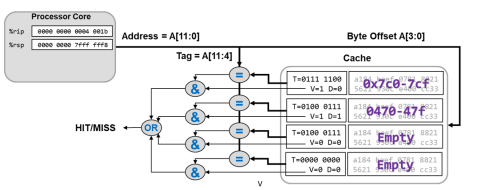


TLB + Data Cache

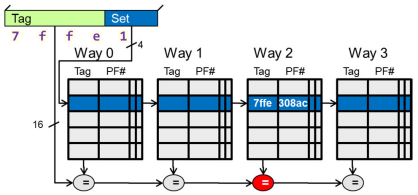


Differences of TLB & Data Cache

- Data cache
  - 1 tag (to identify the block) corresponds to _____
- TLB
  - 1 tag (VPN) corresponds to _____
- Main Point: TLBs are _____ than normal data caches and faster to access



Instruc./Data Cache
(Offset needed since one tag corresponds to many values)



TLB
(1 tag = 1 translation. No Offset needed)

TLB Exercise

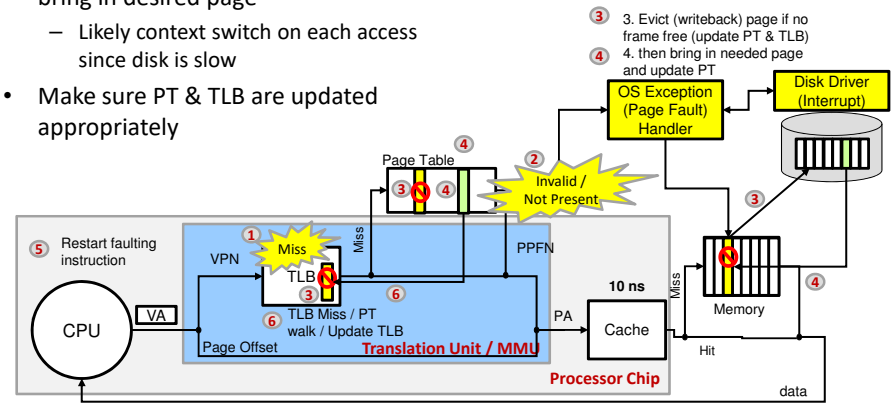
- This TLB is 2-way set associative, with 4 sets
- Page sizes are 256 bytes and 16-bit VAs and PAs
- What is the physical address of virtual address 0x7E85?
- What is the virtual address of physical address 0x3020?

Index	V	Tag	PPFN
0	0	0x13	0x30
	1	0x34	0x58
1	0	0x1F	0x80
	1	0x2A	0x30
2	1	0x1F	0x95
	1	0x20	0xAA
3	1	0x3F	0x20
	0	0x3E	0xFF

TLB

Page Fault Steps

- On page fault, handler will access disk to evict old page (if dirty) and to bring in desired page
  - Likely context switch on each access since disk is slow
- Make sure PT & TLB are updated appropriately



Page Eviction Bookkeeping

- When we want to remove a page from memory
  - Data/instruction cache
    - Writeback any modified blocks belonging to that page
    - Invalidate (set V=0) all blocks belonging to that page
  - TLB (check if a translation for that page is even in the TLB), if so…
    - If Modified/Dirty bit is set for that translation, set modified bit in the page table
    - Invalidate (V=0) the translation
  - Writeback page to disk if modified/dirty bit in Page Table entry is set
  - Update Page Table Entry to indicate the page is not present in memory anymore
  - Simple way to remember this…
    - Children (cache & TLB entries related to a page) must leave when the parent (the actual page) leaves
- Bring in new page and update page directory/page table

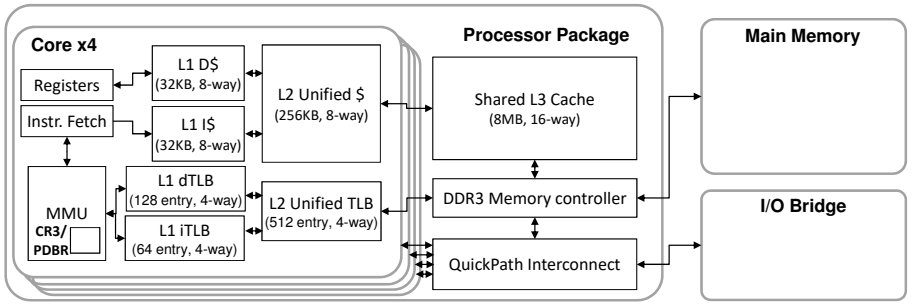
Cache, VM, and Main Memory

TLB	VM	Cache	Possible Y/N & Description
Hit	Hit	Hit	
Hit	Hit	Miss	
Miss	Hit	Hit	
Miss	Hit	Miss	
Miss	Miss	Miss	
Hit	Miss	Miss	
Hit	Miss	Hit	
Miss	Miss	Hit	

Taken from H & P, "Computer Organization" 3rd, Ed.

x86 HW Cache/VM Support

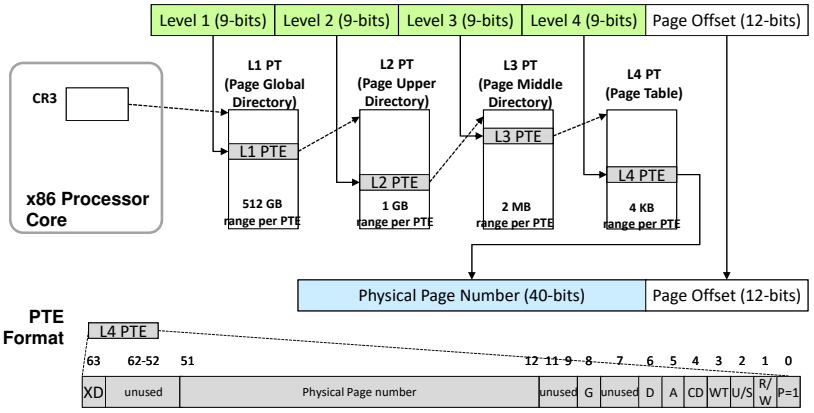
- Cache and TLB Configuration



Intel Core™ i7 Memory System

Core™ i7 Page Table & Entries Format

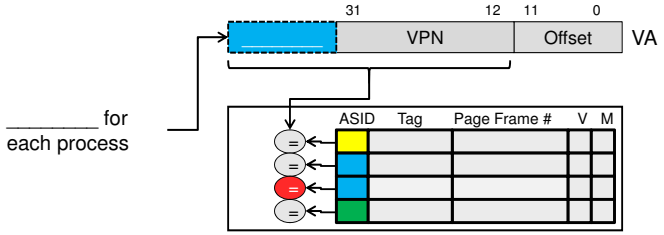
- Specs: 48-bit VA, 52-bit PA, 4KB pages, 4-level Page Table



XD (Execute Disable), G (Global Page), D (Dirty Bit), A (Referenced Bit), CD (Caching Disabled), WT (Write-Thru/WriteBack), U/S (User or Supervisor (Kernel) Mode Permission, R/W (Read-only or Read-write), P (Present/Valid). If P=0, all 63 other bits may be used as the OS desires to store information about the page (i.e. disk location, etc.)

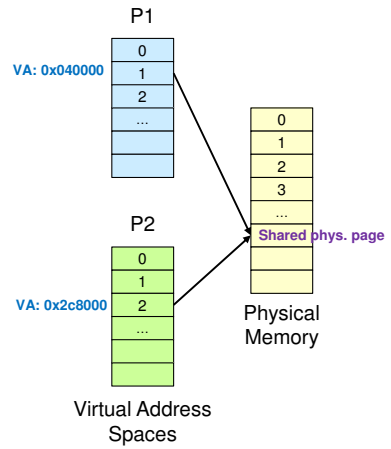
Multiple Processes

- On a process context switch can TLB keep its entries?
  - Can TLB share mappings from multiple processes? _____
- Recall each process has its _____ virtual address space, page table, and translations
  - Virtual addresses are _____ between processes
- How does TLB handle context switch
  - Can choose to only hold translations for current process and thus _____ all entries on _____
  - Can hold translations for multiple processes concurrently by concatenating a _____ to the VPN tag



Shared Memory

- In current system, all memory is _____ to each process
- To share memory between two processes, the OS can allocate an entry in each process' page table to point to the _____ physical page
- Can use different protection bits for each page table entry (e.g. P1 can be R/W while P2 can be read only)

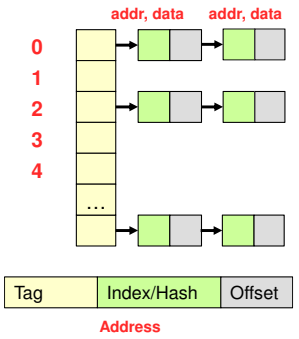


Overlapping TLB access with Data/Instruction Cache access

IF TIME PERMITS

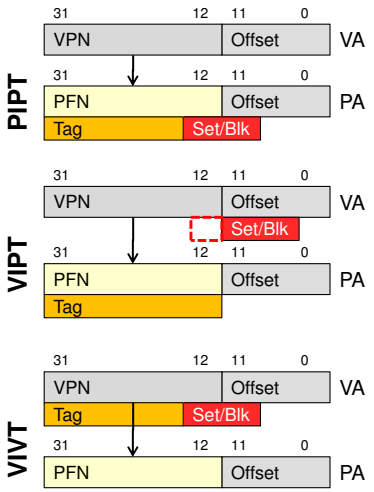
Cache Addressing with VM

- Review of cache
  - Store copies of data indexed based on the address they came from in MM
  - Simplified view: 2 steps to determine hit
    - Index: Hash portion of address to find "set" to look in
    - Tag match: Compare remaining address to all entries in set to determine hit
  - Sequential connection between indexing these two steps (index + tag match)
- Rather than waiting for address translation and then performing this two step hit process, can we overlap the translation and portions of the hit sequence?
  - Yes if we choose page size, block size, and set/direct mapping carefully



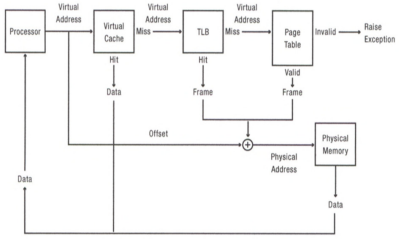
Virtual vs. Physical Addressed Cache

- Physically indexed, physically tagged (PIPT)
  - Wait for full address translation
  - Then use physical address for both indexing and tag comparison
- Virtually indexed, physically tagged (VIPT)
  - Use portion of the virtual address for indexing then wait for address translation and use physical address for tag comparisons
  - Easiest when index portion of virtual address w/in offset (page size) address bits, otherwise aliasing may occur
- Virtually indexed, virtually tagged (VIVT)
  - Use virtual address for both indexing and tagging…No TLB access unless cache miss
  - Requires invalidation of cache lines on context switch or use of process ID as part of tags

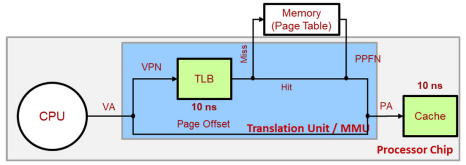


Virtual vs. Physical Addressed Cache

- Another view:



Virtually addressed Cache



Physically addressed Cache

In a modern system the L1 caches may be virtually addressed while L2 may be physically addressed.

EXERCISES