ECE M116C / CS 151B: Week 8 Section

Justin Feng

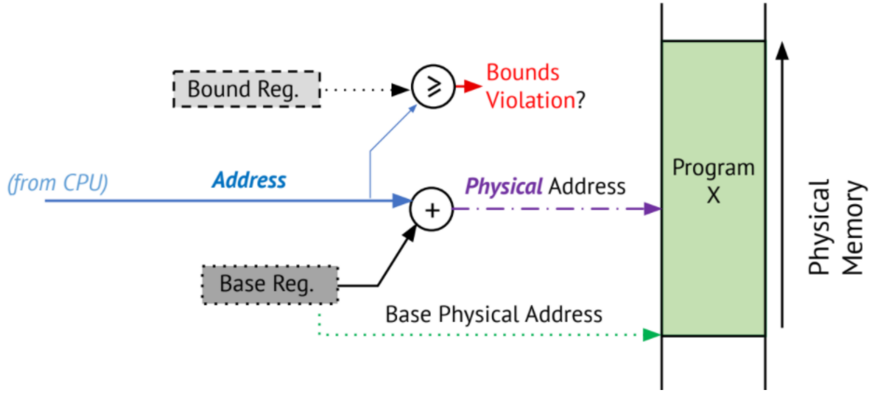
UCLA

Static Partitioning: Shared Memory

-**Ensure that independent programs can’t affect each other**

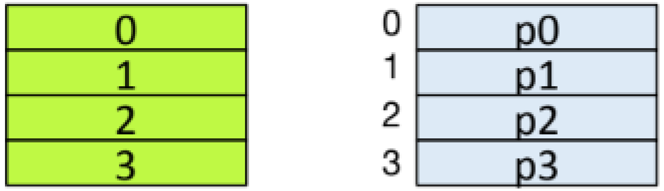
-Protection: check the bounds of the programs

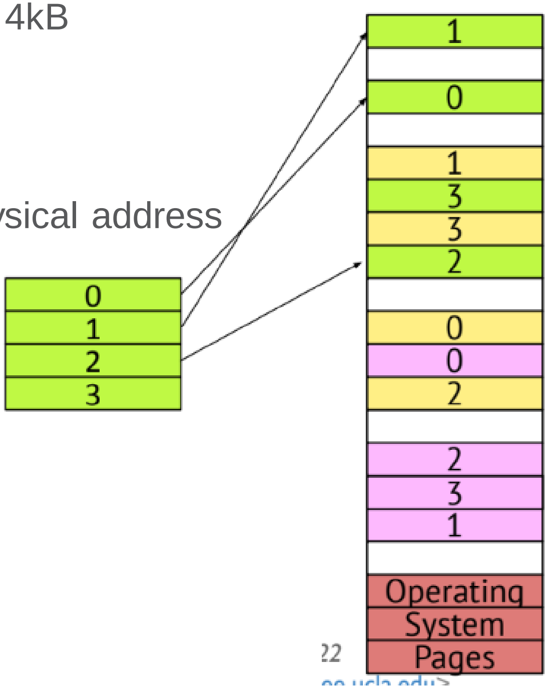
-Location-independence: use a pointer, as program could be moved in memory



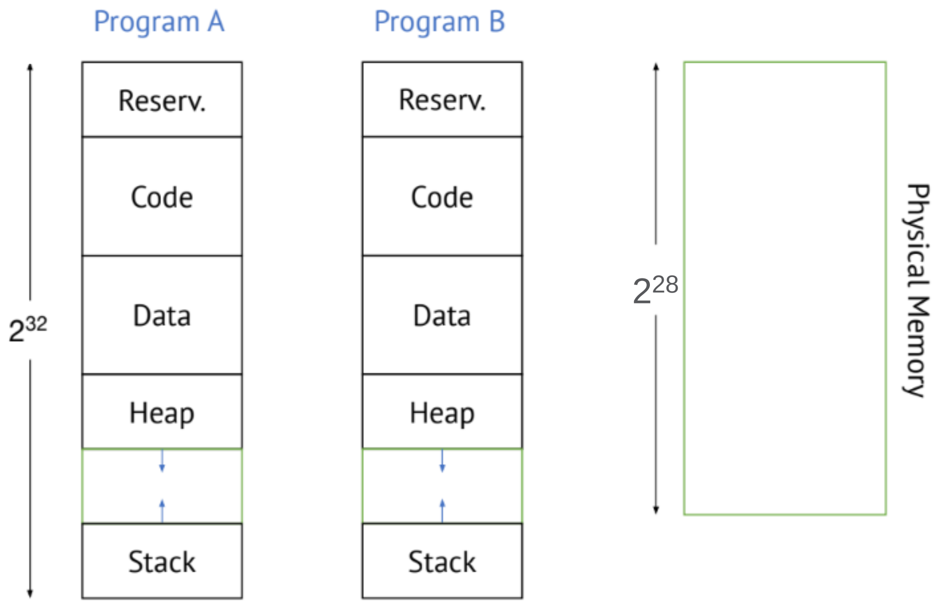
Dynamic Partitioning: Shared Memory

- **Enable addition of new program, changing of a program's memory space**
  - Divide memory into a page (fixed-size blocks) -> typically 4kB
  - A program consists of many pages, scattered in memory
  - How to keep track of page location: use a page table
    - Results in a virtual address, which maps to a physical address





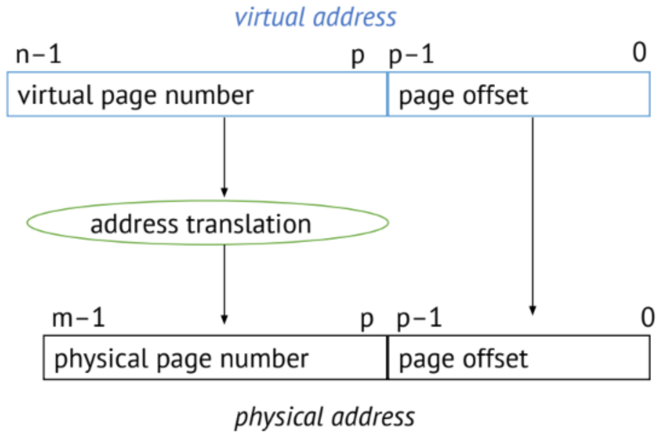
Page Tables



2^n: virtual address size

2^m: physical address size

2^p: page size



Page Tables

-Virtual address size: 32 bits (results in 4 GB of memory -> 2^{32})

-Physical address size: 28 bits (results in 256 MB of memory -> 2^{28})

-Page offset: 12 bits (results in 4 kB of memory per page -> 2^{12})

-# of virtual pages?

-# of physical pages?

Page Tables

-Virtual address size: 32 bits (results in 4 GB of memory -> 2^{32})

-Physical address size: 28 bits (results in 256 MB of memory -> 2^{28})

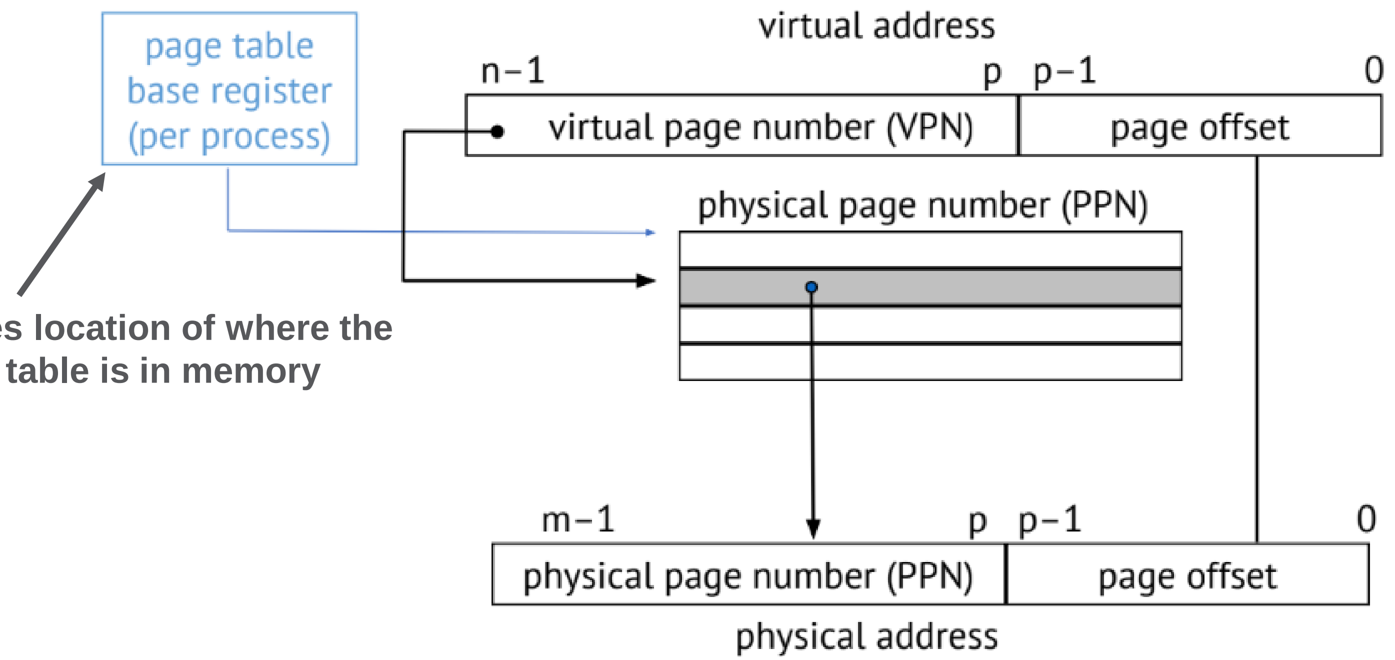
-Page offset: 12 bits (results in 4 kB of memory per page -> 2^{12})

-# of virtual pages? **2^32 / 2^12 = 2^20 pages**

-# of physical pages? **2^28 / 2^12 = 2^16 pages**

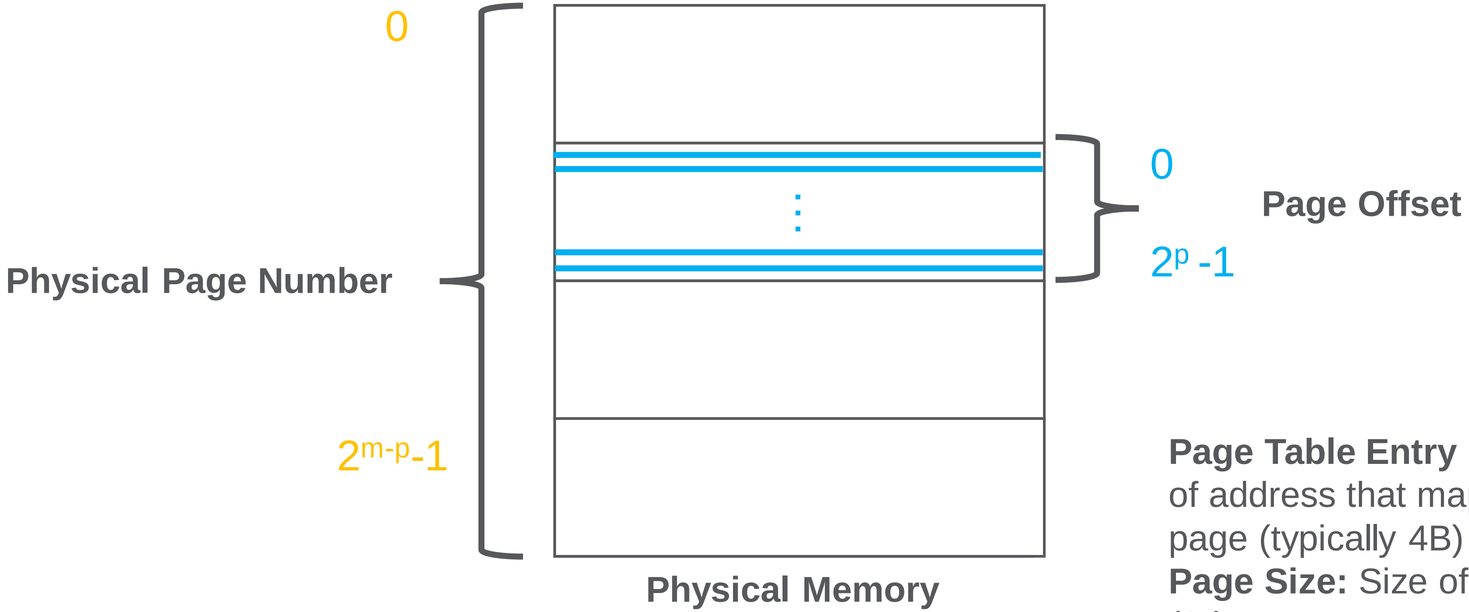
UCLA

Page Tables


page table base register (per process)
virtual address
n−1
p
p−1
0
virtual page number (VPN)
page offset
physical page number (PPN)
m−1
p
p−1
0
physical page number (PPN)
page offset
physical address

Stores location of where the page table is in memory

Page Tables

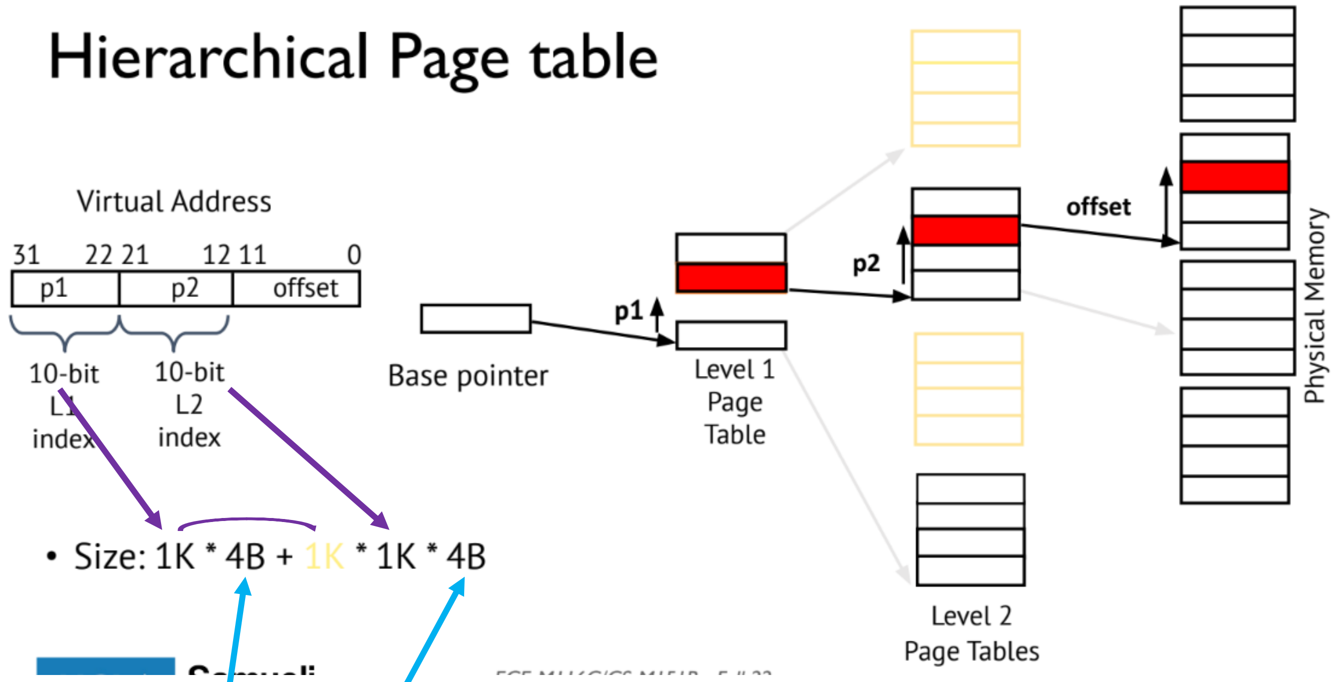


Page Table Entry Size: size of address that maps to the page (typically 4B)
Page Size: Size of the page (2^p)

Hierarchical Page Table

-Virtual addresses are sparse, thus, can make page table storage compact by leveraging this sparsity!

-Issue: more memory accesses



Hierarchical Page table

- Size: 1K * 4B + 1K * 1K * 4B

4B PT Entry Size

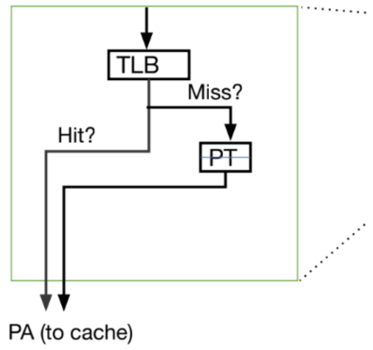
Translation Lookaside Buffer

-Each access now has multiple memory accesses (one per each level of the hierarchical page table).

-Use translation lookaside buffer to speed up memory accesses by caching past page table entries

-If have computed translation in buffer, directly grab data from physical memory

-Else, need to walk the page table



Example: Page Tables

EX) 4 GB Virtual Address Space. 4 kB Page Size. Page Table Entry Size of 4 B (aka 32 bits).

-Number of Virtual Pages?

-Size of a Flat Page Table?

-Size of a 2 Level Page Table, 10 bits for First Level?

UCLA

Example: Page Tables

EX) 4 GB Virtual Address Space. 4 kB Page Size. Page Table Entry Size of 4 B (aka 32 bits).

-Number of Virtual Pages?

-> # Virtual Pages = $\frac{Virtual\ Address\ Space}{Page\ Size}$ **=** $\frac{4\ GB}{4\ kB}$ **->** $\frac{2^{32}}{2^{12}}$ **=** 2^{20}

-Size of a Flat Page Table

-Size of a 2 Level Page Table, 10 bits for First Level?

Example: Page Tables

EX) 4 GB Virtual Address Space. 4 kB Page Size. Page Table Entry Size of 4 B (aka 32 bits).

-Number of Virtual Pages?

-> # Virtual Pages = $\frac{Virtual\ Address\ Space}{Page\ Size} = \frac{4\ GB}{4\ kB}$ -> $\frac{2^{32}}{2^{12}} = 2^{20}$

-Size of a Flat Page Table?

-> Page Table Size = # Virtual Pages x Entry Size = 2^{20} x 4 B = ~4 MB

-Size of a 2 Level Page Table, 10 bits for First Level?

UCLA

Example: Page Tables

EX) 4 GB Virtual Address Space. 4 kB Page Size. Page Table Entry Size of 4 B (aka 32 bits).

-Size of a 2 Level Page Table, 10 bits for First Level?

-> First Level Table Size: # Pages x Entry Size = 2^{10} x 4 B = ~4 kB

-> Second Level Table Size: $2^{\text{VPN Width} - \text{1st Level Width} - \text{Offset Bits}}$ x Entry Size

= $2^{32-10-12}$ x 4 B = ~4 kB

-> Total Size: First Level Size + Second Level Size x (1st Level number of entries)

= 4 kB + 4 kB x 2^{10} = ~4 MB

-Note that Virtual Memory is sparse! So size of second level will be less than appears

UCLA

DRAM vs Disk

UCLA

DRAM vs Disk

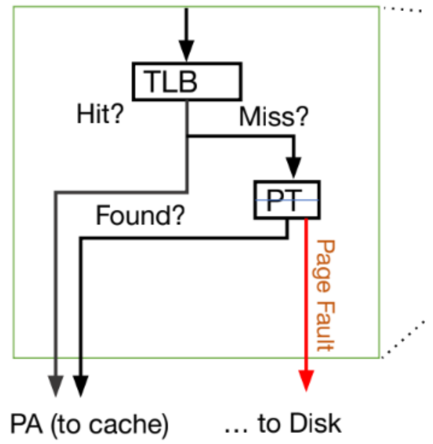
-DRAM is to cache as Disk is to DRAM

-Keep most frequently used bytes in DRAM

-Page Fault: when page is not in DRAM

-Demand Paging: how to know when to bring in pages, evict pages

-taken care of by OS



VIPT Cache

UCLA

VIPT

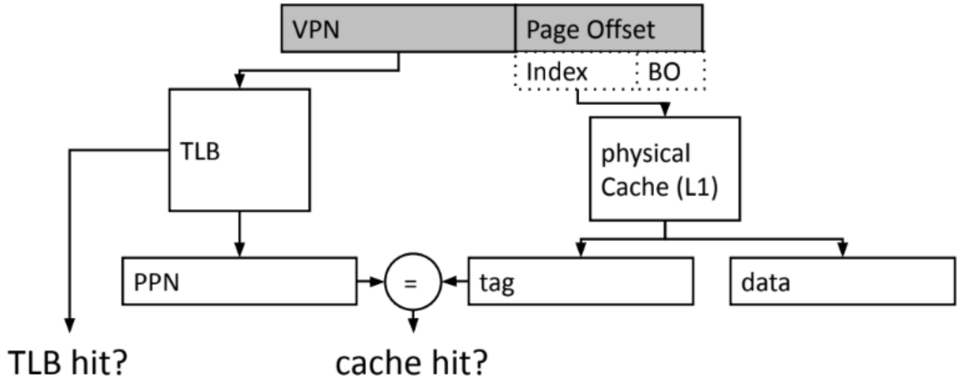
-VIPT: Virtually Indexed, Physically Tagged Cache

-Goal: access TLB and L1 cache in parallel (can’t access L1 first because of aliasing)

-If we can confirm the PPN = Tag, then we can access the data in the L1 cache

-Hit time: time to access L1 cache

-Limitation: cache is small

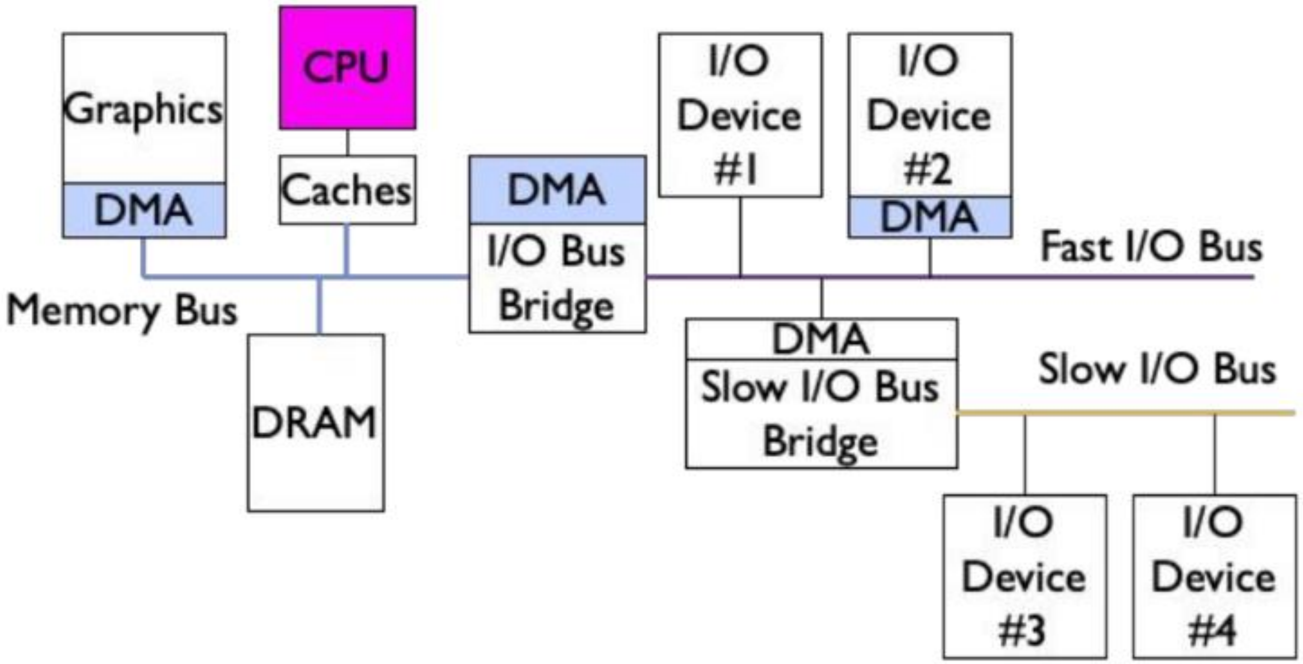


UCLA

I/O

UCLA

I/O


Graphics
DMA
CPU
Caches
Memory Bus
DRAM
DMA
I/O Bus Bridge
I/O Device #1
I/O Device #2
DMA
Fast I/O Bus
DMA
Slow I/O Bus Bridge
Slow I/O Bus
I/O Device #3
I/O Device #4

I/O

-Computer System may have several different kinds of I/O’s connected

-Need to read/write to these I/O’s

- -Write to registers specific to these I/O’s, using a channel (pin)

-Each I/O has a portion of memory assigned to it

- -Protocol: DMA (direct memory access)
    - -Allows I/O to access memory without help of CPU
    - -Use interrupts or polling to synchronize

UCLA

Interrupts vs Polling

-Interrupt: asynchronous unplanned context switch

- -External event
- -Causes interrupt exception to occur, which causes CPU to experience a trap
  - -Trap: transfer of control to a handler -> jump to a side routine
- -I/O's cause interrupts. Different interrupts have different priorities (affects when to trap)

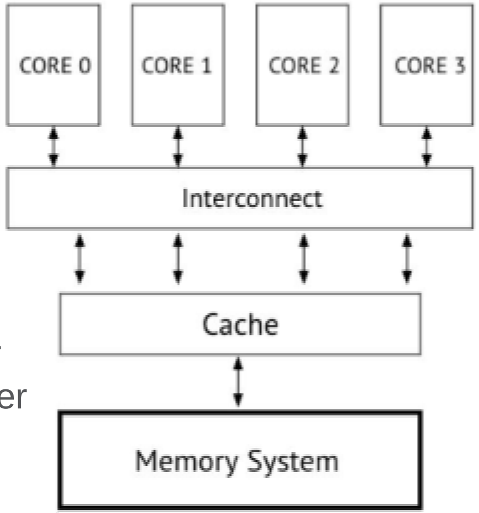
-Polling: synchronous planned context switch

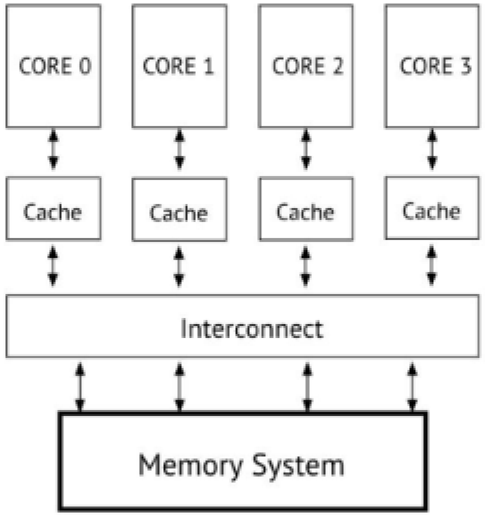
Multicore

Shared vs Private Caches

Shared

-Worse Hit Time
-Cache Friendly App: Lower Miss Rate (utilizing cache better)
-App that needs a lot of memory: higher miss rate (will evict other app's data)
-Slightly less miss time (doesn't need to go through Interconnect)
-Easier sharing (unified cache)



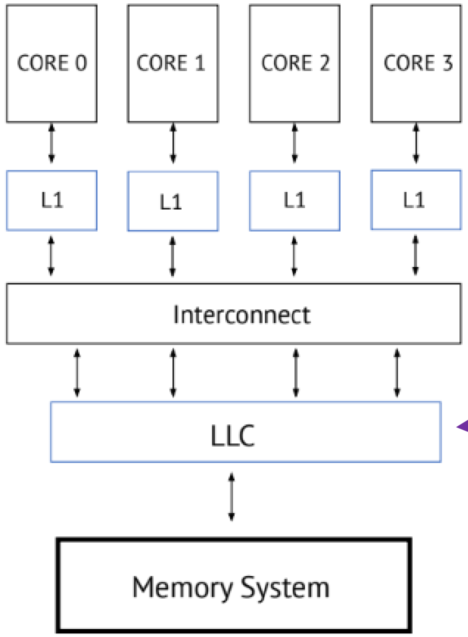


Private

UCLA

Hybrid System (Best of both worlds)

Private L1 Cache
-Best hit rate



Shared LLC Cache
-Easier sharing
-Cores can "fight" over this memory