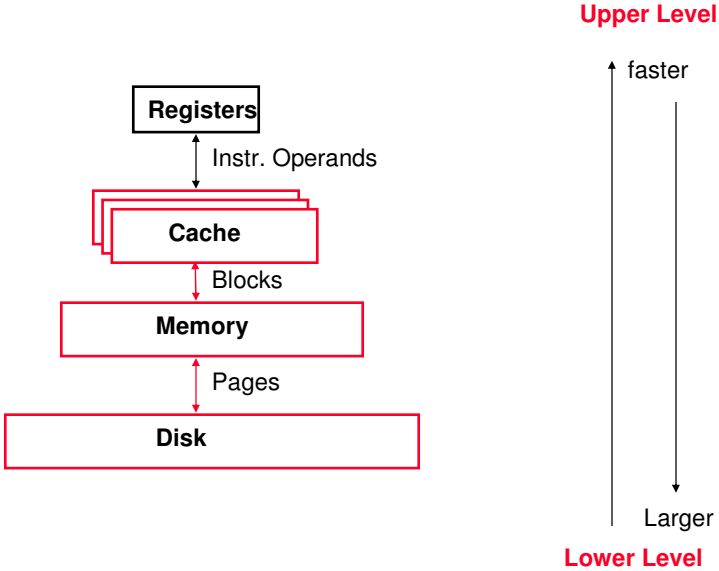
Appendix C
Memory Hierarchy

Adapted from Patterson and Hennessey
(Morgan Kauffman Pubs)

Levels of the Memory Hierarchy



Adapted from Patterson and Hennessey
(Morgan Kauffman Pubs)

Reminder: The Principle of Locality

- The Principle of Locality:
  - Program access a relatively small portion of the address space at any instant of time.
- Two Different Types of Locality:
  - Temporal Locality (Locality in Time): If an item is referenced, it will tend to be referenced again soon (e.g., loops, reuse)
  - Spatial Locality (Locality in Space): If an item is referenced, items whose addresses are close by tend to be referenced soon (e.g., straightline code, array access)

Adapted from Patterson and Hennessey (Morgan Kauffman Pubs)

CS613 s12 – Appendix C — 3

Reminder: Terminology

- Hit: data appears in some block in the upper level (example: Block X)
  - Hit Rate: the fraction of memory access found in the upper level
  - Hit Time: Time to access the upper level
  - Miss: data needs to be retrieved from a block in the lower level (Block Y)
  - Miss Rate = 1 - (Hit Rate)
  - Miss Penalty: Time to access the upper level (determine it's a miss)
    + Time to replace a block in the upper level
    + Time to deliver the block the processor
- Hit Time << Miss Penalty

Adapted from Patterson and Hennessey (Morgan Kauffman Pubs)

CS613 s12 – Appendix C — 4

Cache Measures

- *Hit rate*: fraction found in that level
  - So high that usually talk about *Miss rate*
  - Miss rate fallacy: only useful as a rule-of-thumb.
    as MIPS to CPU performance,
    miss rate to average memory access time in memory
- Average memory-access time
  = Hit time + Miss rate x Miss penalty (ns or clocks)
- *Miss penalty*: time to replace a block from lower level, including time to replace in CPU
  - *access time*: time to lower level
    = f(latency to lower level)
  - *transfer time*: time to transfer block
    =f(BW between upper & lower levels)

Adapted from Patterson and Hennessey
(Morgan Kauffman Pubs)

CS613 s12 – Appendix C — 5

4 Questions for Memory Hierarchy

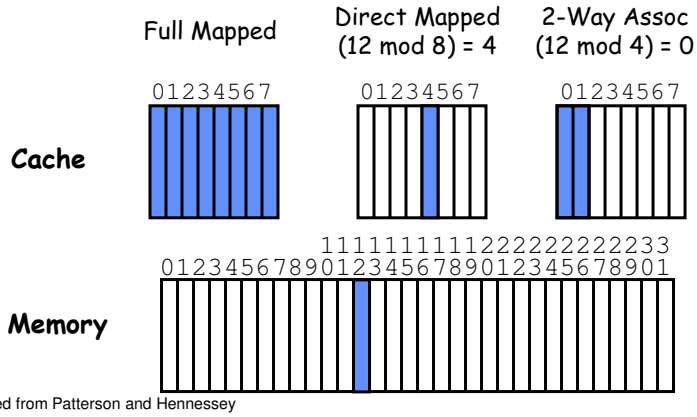
- Q1: Where can a block be placed in the upper level?
  (Block placement)
- Q2: How is a block found if it is in the upper level?
  (Block identification)
- Q3: Which block should be replaced on a miss?
  (Block replacement)
- Q4: What happens on a write?
  (Write strategy)

Adapted from Patterson and Hennessey
(Morgan Kauffman Pubs)

CS613 s12 – Appendix C — 6

Q1: Where can a block be placed in the upper level?

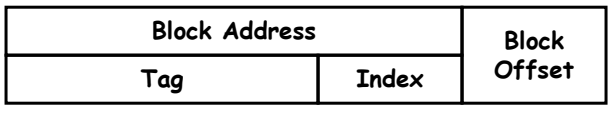
- **Block 12 placed in 8 block cache:**
  - **Fully associative, direct mapped, 2-way set associative**
  - **S.A. Mapping = Block Number Modulo Number Sets**



Adapted from Patterson and Hennessey
(Morgan Kauffman Pubs)

Q2: How is a block found if it is in the upper level?

- **Tag on each block**
  - **No need to check index or block offset**
- **Increasing associativity shrinks index, expands tag**



Adapted from Patterson and Hennessey
(Morgan Kauffman Pubs)

Q3: After a cache read miss, if there are no empty cache blocks, which block should be removed from the cache?

The Least Recently Used (LRU) block? Appealing, but hard to implement for high associativity

A randomly chosen block? Easy to implement, how well does it work?

Miss Rate for 2-way Set Associative Cache

Size	Random	LRU
16 KB	5.7%	5.2%
64 KB	2.0%	1.9%
256 KB	1.17%	1.15%

Also, try other LRU approx.

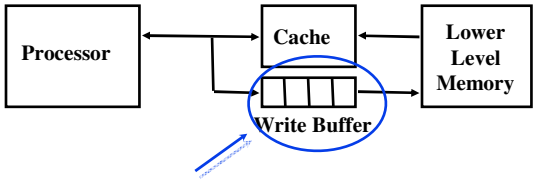
Adapted from Patterson and Hennessey (Morgan Kauffman Pubs)

Q4: What happens on a write?

	Write-Through	Write-Back
Policy	Data written to cache block also written to lower-level memory	Write data only to the cache Update lower level when a block falls out of the cache
Advantages	Lower level stays (fairly) up-to-date with new values	Repeated writes don't affect the lower level
Disadvantages	Write queues may delay updates	Lower level out-of-date with new values (problem in multi processors, DMA, etc)

Adapted from Patterson and Hennessey (Morgan Kauffman Pubs)

Write Buffers for Write-Through Caches



Holds data awaiting write-through to lower level memory

Q. Why a write buffer ? **A. So CPU doesn't stall**

Q. Why a buffer, why not just one register ? **A. Bursts of writes are common.**

Adapted from Patterson and Hennessey (Morgan Kauffman Pubs)

5 Basic Cache Optimizations

- **Reducing Miss Rate**

1. **Larger Block size**
2. **Larger Cache size**
3. **Higher Associativity**

- **Reducing Miss Penalty**

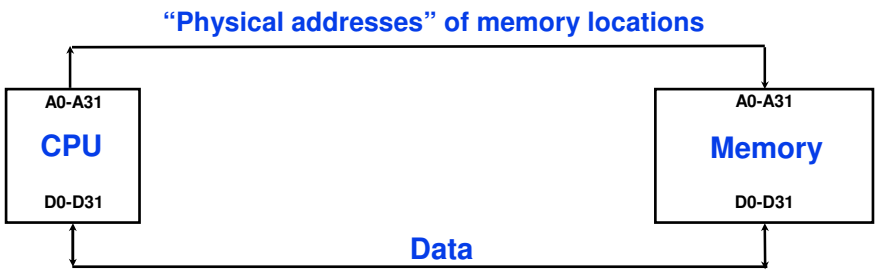
4. **Multilevel Caches**

- **Reducing hit time**

5. **Giving Reads Priority over Writes**
   - **E.g., Read complete before earlier writes in write buffer**

Adapted from Patterson and Hennessey (Morgan Kauffman Pubs)

The Limits of Physical Addressing



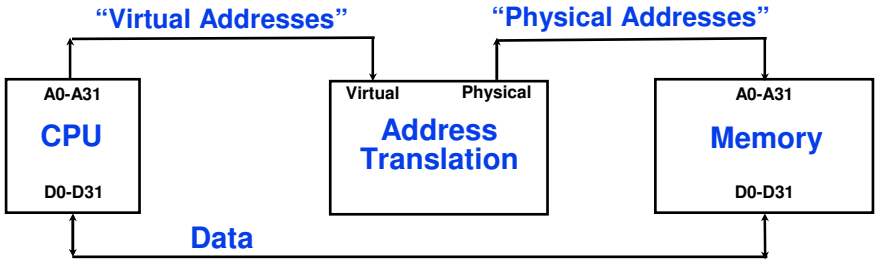
All programs share one address space: The physical address space

May desire larger memory than the physical space has

No way to prevent a program from accessing any machine resource

Adapted from Patterson and Hennessey (Morgan Kauffman Pubs)

Solution: Add a Layer of Indirection



User programs run in an standardized virtual address space

Address Translation hardware managed by the operating system (OS) maps virtual address to physical memory

Adapted from Patterson and Hennessey (Morgan Kauffman Pubs)

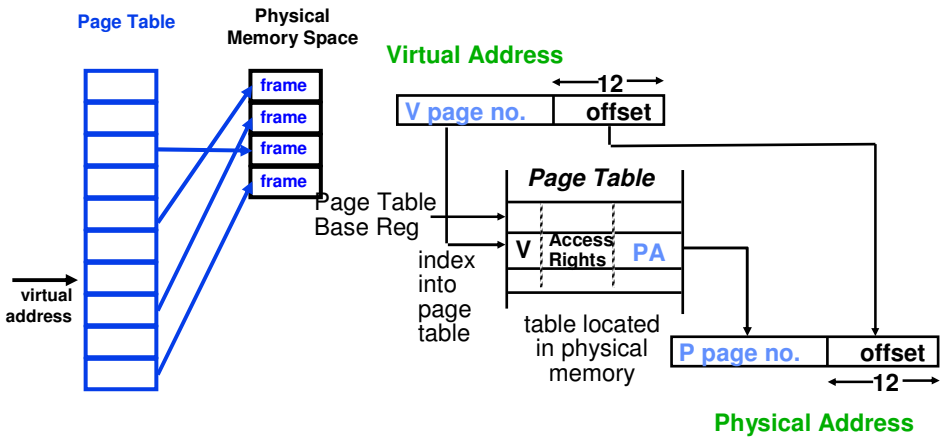
Three Advantages of Virtual Memory

- **Translation:**
  - Program can be given consistent view of memory, even though physical memory is scrambled
  - Only the most important part of program ("Working Set") must be in physical memory.
  - Contiguous structures (like stacks) use only as much physical memory as necessary yet still grow later.
- **Protection:**
  - Different threads (or processes) protected from each other.
  - Different pages can be given special behavior
    - » (Read Only, Invisible to user programs, etc).
  - Kernel data protected from User programs
  - Very important for protection from malicious programs
- **Sharing:**
  - Can map same physical page to multiple users ("Shared memory")

Adapted from Patterson and Hennessey (Morgan Kauffman Pubs)

CS613 s12 – Appendix C — 15

Details of Page Table

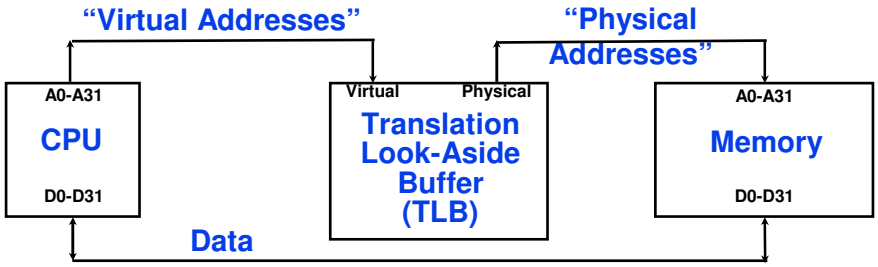


- **Page table maps virtual page numbers to physical frames ("PTE" = Page Table Entry)**
- **Virtual memory => treat memory ≈ cache for disk**

Adapted from Patterson and Hennessey (Morgan Kauffman Pubs)

CS613 s12 – Appendix C — 16

Speeding up the Translation process

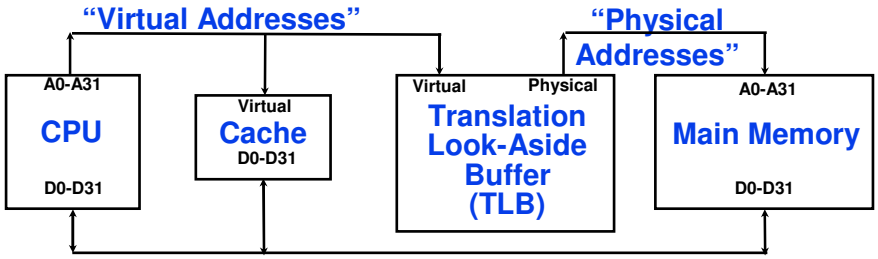


Translation Look-Aside Buffer (TLB)
A small fully-associative cache of mappings from virtual to physical addresses

TLB also contains protection bits for virtual address

Adapted from Patterson and Hennessey (Morgan Kauffman Pubs)

Use virtual addresses for cache?



Only use TLB on a cache miss !

Downside: Synonym problem. If two address spaces share a physical frame, data may be in cache twice. Maintaining consistency is a nightmare.

Adapted from Patterson and Hennessey (Morgan Kauffman Pubs)

Cache choices versus VM choices

- **Caches**
  - **The most important thing is speed**
  - **We choose:**
    - » **Direct-Mapped or small Set Associative**
    - » **Fast selection of block to be replaced (e.g., Random)**
    - » **Write-through**
- **Virtual memory**
  - **The most important thing is to minimize misses**
  - **We choose:**
    - » **Full Associative or large Set Associative**
    - » **The best (practical) algorithm for replacing blocks (e.g, LRU)**
    - » **Write-back**

Adapted from Patterson and Hennessey
(Morgan Kauffman Pubs)