inst.eecs.berkeley.edu/~cs61c

CS61C : Machine Structures

Lecture 37
VM III

2004-11-24

Lecturer PSOE Dan Garcia

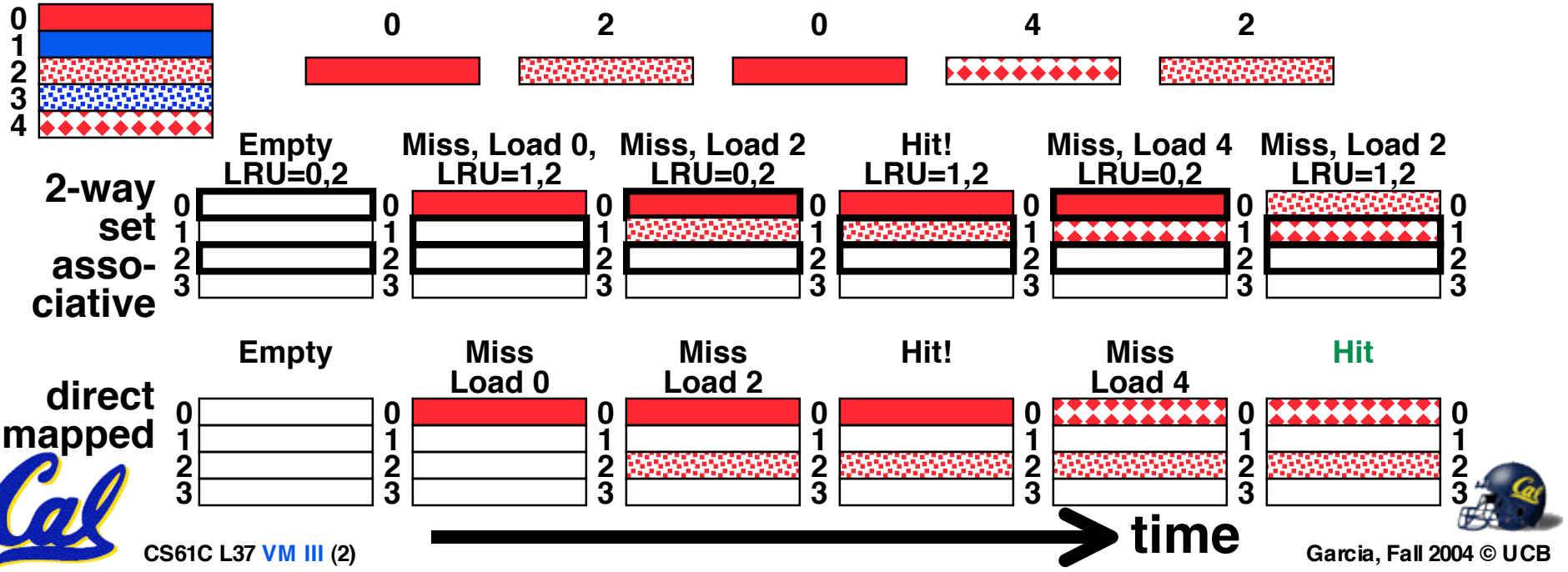
www.cs.berkeley.edu/~ddgarcia

This 230mph electric car accelerates faster than a Porsche 911 Turbo and goes 200 miles on a single charge! Wow!

eliica.com

Peer Instruction Correction

- A direct-mapped $ will never out-perform a 2-way set-associative $ of the same size.
  - I said "TRUE … increased associativity!"
  - Right Answer "FALSE … consider the following"
    - We have 4 byte cache, block size = 1 byte. Compare a 2-way set-associative cache (2 sets using LRU replacement) with a direct mapped cache (four rows).



Peer Instruction

1. Increasing at least one of {associativity, block size} always a win
2. Higher DRAM bandwidth translates to a lower miss rate
3. DRAM access time improves roughly as fast as density

	ABC
1:	FFF
2:	FFT
3:	FTF
4:	FTT
5:	TFF
6:	TFT
7:	TTF
8:	TTT

Peer Instruction Answers

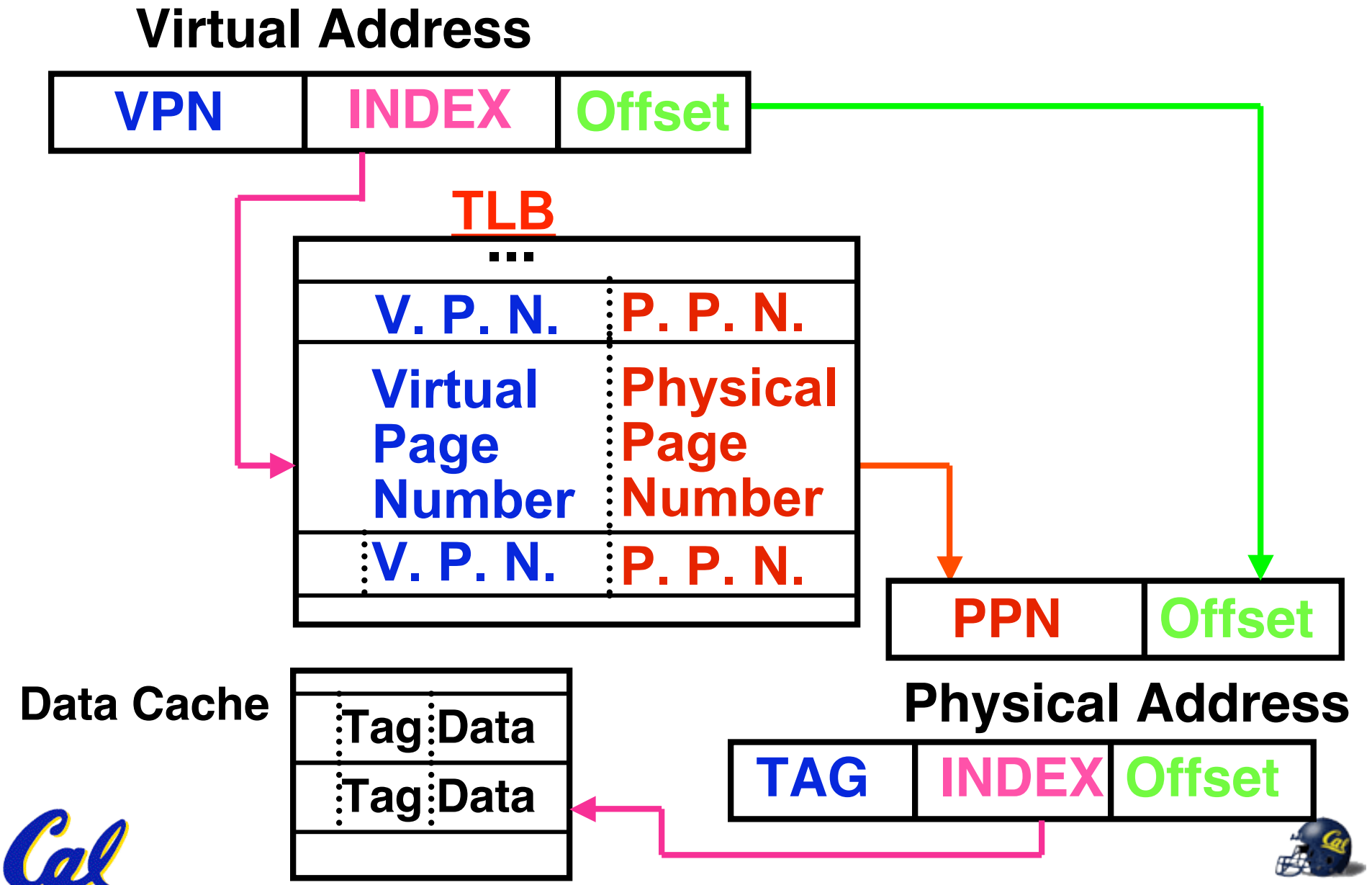
1. Increasing at least one of {associativity, block size} is always a win
2. Higher DRAM bandwidth translates to a lower miss rate
3. DRAM access time improves roughly as fast as density

FALSE FALSE FALSE

1. Assoc. may increase access time, block may increase miss penalty
2. No, a lower miss penalty
3. No, access = 9%/year, but density = 2x every 2 yrs!

	ABC
1:	FFF
2:	FFT
3:	FTF
4:	FTT
5:	TFF
6:	TFT
7:	TTF
8:	TTT

Address Translation & 3 Concept tests



Peer Instruction (1/3)

- 40-bit virtual address, 16 KB page

- 36-bit physical address

- Number of bits in Virtual Page Number/ Page offset, Physical Page Number/Page offset?

1: 22/18 (VPN/PO), 22/14 (PPN/PO)
2: 24/16, 20/16
3: 26/14, 22/14
4: 26/14, 26/10
5: 28/12, 24/12

Peer Instruction (1/3) Answer

- 40- bit virtual address, 16 KB (2^{14} B)

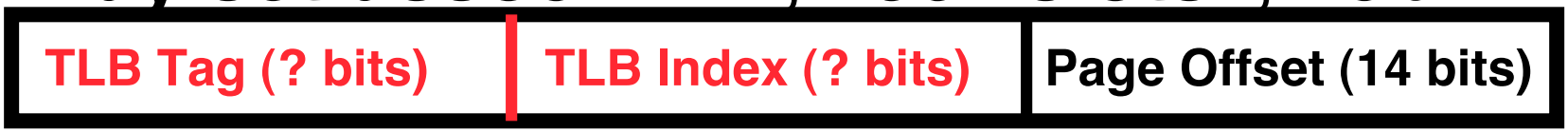
- 36- bit virtual address, 16 KB (2^{14} B)

- Number of bits in Virtual Page Number/ Page offset, Physical Page Number/Page offset?

1: 22/18 (VPN/PO), 22/14 (PPN/PO)
2: 24/16, 20/16
3: 26/14, 22/14
4: 26/14, 26/10
5: 28/12, 24/12

Peer Instruction (2/3): 40b VA, 36b PA

- 2-way set-assoc. TLB, 256 "slots", 40b VA:



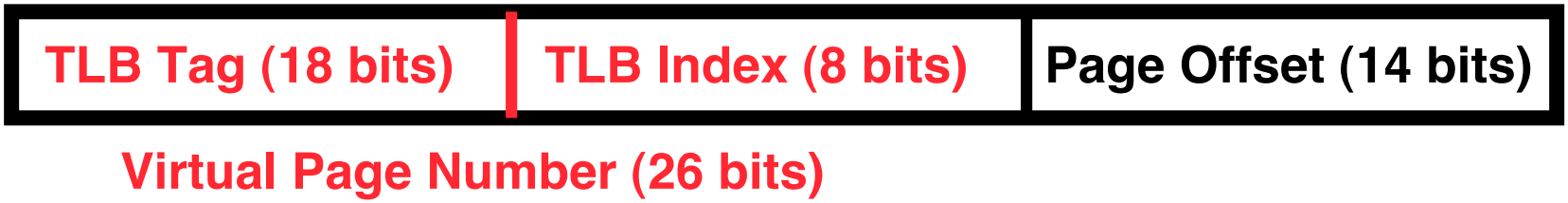
- TLB Entry: Valid bit, Dirty bit, Access Control (say 2 bits), Virtual Page Number, Physical Page Number

- Number of bits in TLB Tag / Index / Entry?

1: 12 / 14 / 38 (TLB Tag / Index / Entry)
2: 14 / 12 / 40
3: 18 / 8 / 44
4: 18 / 8 / 58

Peer Instruction (2/3) Answer

- 2-way set-assoc data cache, 256 (2^8) "slots", 2 TLB entries per slot => 8 bit index



- TLB Entry: Valid bit, Dirty bit, Access Control (2 bits), Virtual Page Number, Physical Page Number

1: 12 / 14 / 38 (TLB Tag / Index / Entry)
2: 14 / 12 / 40
3: 18 / 8 / 44
4: 18 / 8 / 58

Peer Instruction (3/3)

- 2-way set-assoc, 64KB data cache, 64B block

Physical Page Address (36 bits)

- Data Cache Entry: Valid bit, Dirty bit, Cache tag + ? bits of Data

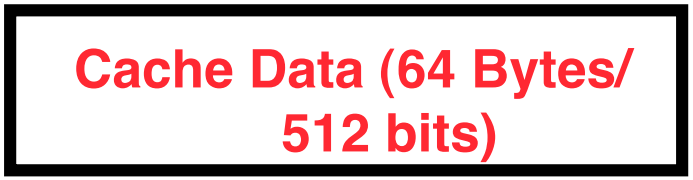
- Number of bits in Data cache Tag / Index / Offset / Entry?

1: 12 / 9 / 14 / 87 (Tag/Index/Offset/Entry)
2: 20 / 10 / 6 / 86
3: 20 / 10 / 6 / 534
4: 21 / 9 / 6 / 87
5: 21 / 9 / 6 / 535

Peer Instruction (3/3) Answer

- 2-way set-assoc data cache, 64K/1K (2^{10}) "slots", 2 entries per slot => 9 bit index

- Data Cache Entry: Valid bit, Dirty bit, Cache tag + 64 Bytes of Data



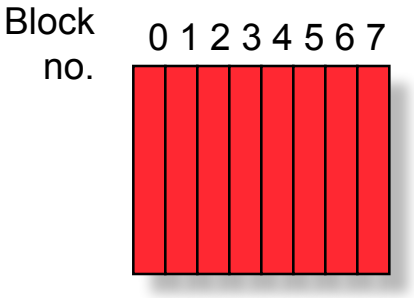
1: 12 / 9 / 14 / 87 (Tag/Index/Offset/Entry)
2: 20 / 10 / 6 / 86
3: 20 / 10 / 6 / 534
4: 21 / 9 / 6 / 87
5: 21 / 9 / 6 / 535

4 Qs for any Memory Hierarchy

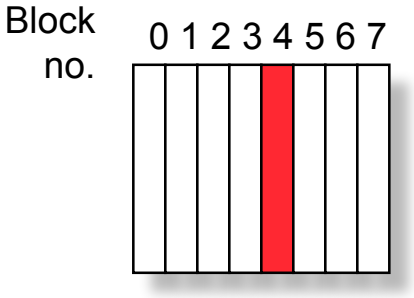
- Q1: Where can a block be placed?
  - One place (direct mapped)
  - A few places (set associative)
  - Any place (fully associative)
- Q2: How is a block found?
  - Indexing (as in a direct-mapped cache)
  - Limited search (as in a set-associative cache)
  - Full search (as in a fully associative cache)
  - Separate lookup table (as in a page table)
- Q3: Which block is replaced on a miss?
  - Least recently used (LRU)
  - Random
- Q4: How are writes handled?
  - Write through (Level never inconsistent w/lower)
  - Write back (Could be “dirty”, must have dirty bit)

Q1: Where block placed in upper level?

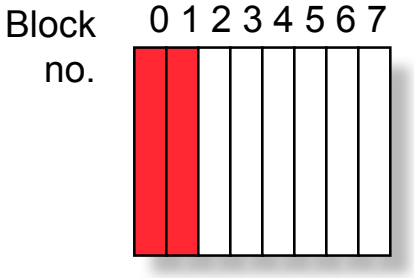
- Block 12 placed in 8 block cache:
  - Fully associative
  - Direct mapped
  - 2-way set associative
    - Set Associative Mapping = Block # Mod # of Sets



Fully associative: block 12 can go anywhere

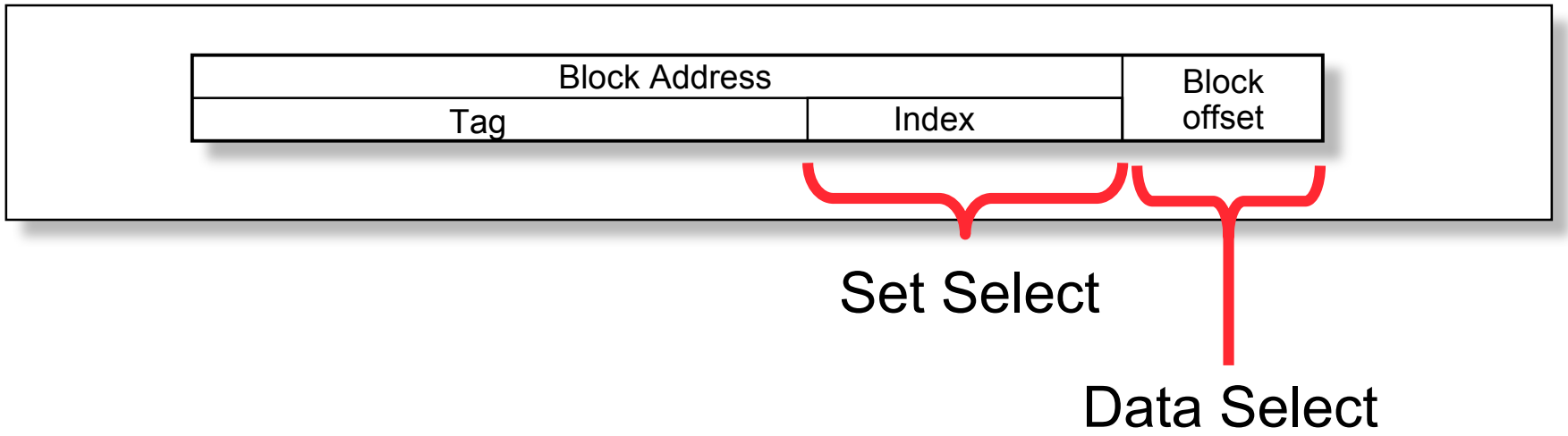


Direct mapped: block 12 can go only into block 4 (12 mod 8)



Set associative: block 12 can go anywhere in set 0 (12 mod 4)

Q2: How is a block found in upper level?



- Direct indexing (using index and block offset), tag compares, or combination
- Increasing associativity shrinks index, expands tag

Q3: Which block replaced on a miss?

- Easy for Direct Mapped
- Set Associative or Fully Associative:
  - Random
  - LRU (Least Recently Used)

Miss Rates

Associativity:	2-way		4-way		8-way	
Size	LRU	Ran	LRU	Ran	LRU	Ran
16 KB	5.2%	5.7%	4.7%	5.3%	4.4%	5.0%
64 KB	1.9%	2.0%	1.5%	1.7%	1.4%	1.5%
256 KB	1.15%	1.17%	1.13%	1.13%	1.12%	1.12%

Q4: What to do on a write hit?

- **Write-through**
  - update the word in cache block and corresponding word in memory
- **Write-back**
  - update word in cache block
  - allow memory word to be "stale"

  => add 'dirty' bit to each line indicating that memory be updated when block is replaced

  => OS flushes cache before I/O !!!

- Performance trade-offs?
  - WT: read misses cannot result in writes
  - WB: no writes of repeated writes

Three Advantages of Virtual Memory

1) Translation:

- Program can be given consistent view of memory, even though physical memory is scrambled
- Makes multiple processes reasonable
- 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

Three Advantages of Virtual Memory

2) Protection:

- Different 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 ⇒ Far more "viruses" under Microsoft Windows
- Special Mode in processor ("Kernel more") allows processor to change page table/TLB

3) Sharing:

- Can map same physical page to multiple users ("Shared memory")

Why Translation Lookaside Buffer (TLB)?

- Paging is most popular implementation of virtual memory (vs. base/bounds)
- Every paged virtual memory access must be checked against Entry of Page Table in memory to provide protection
- Cache of Page Table Entries (TLB) makes address translation possible without memory access in common case to make fast

Bonus slide: Virtual Memory Overview (1/4)

- User program view of memory:
  - Contiguous
  - Start from some set address
  - Infinitely large
  - Is the only running program
- Reality:
  - Non-contiguous
  - Start wherever available memory is
  - Finite size
  - Many programs running at a time

Bonus slide: Virtual Memory Overview (2/4)

- Virtual memory provides:
  - illusion of contiguous memory
  - all programs starting at same set address
  - illusion of ~ infinite memory (2^{32} or 2^{64} bytes)
  - protection

Bonus slide: Virtual Memory Overview (3/4)

- Implementation:
  - Divide memory into “chunks” (pages)
  - Operating system controls page table that maps virtual addresses into physical addresses
  - Think of memory as a cache for disk
  - TLB is a cache for the page table

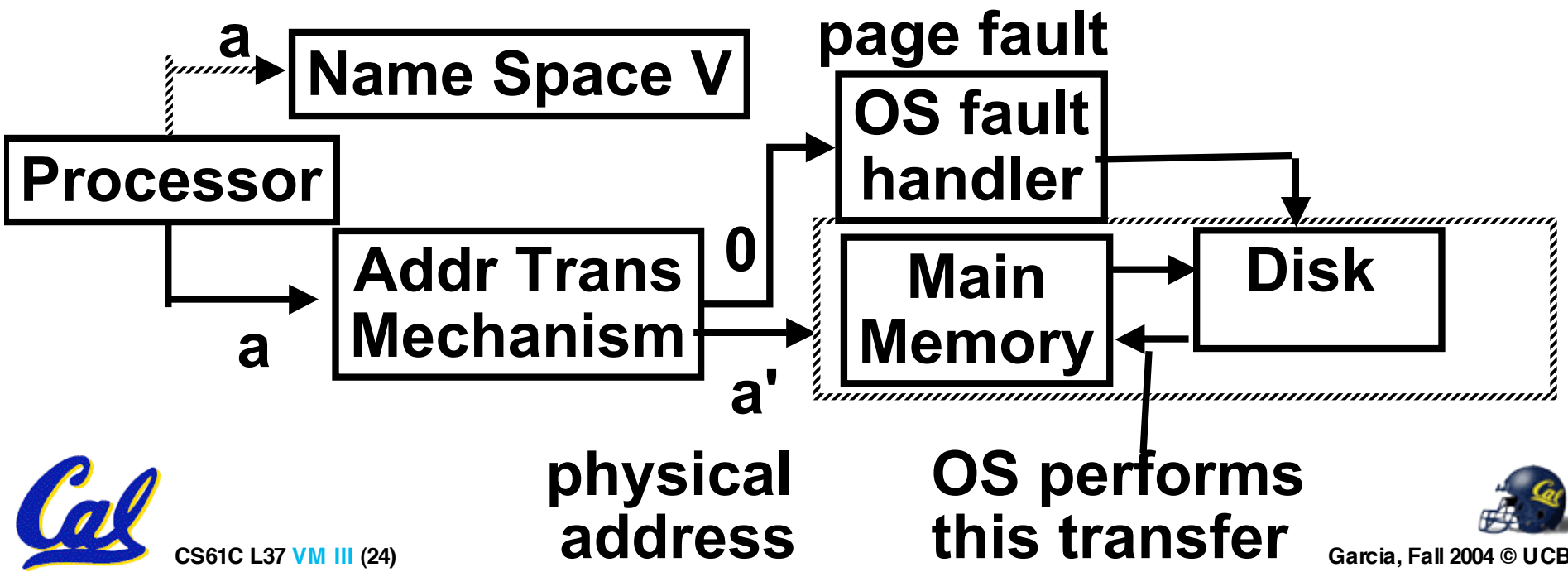
Bonus slide: Virtual Memory Overview (4/4)

- Let's say we're fetching some data:
  - Check TLB (input: VPN, output: PPN)
    - hit: fetch translation
    - miss: check page table (in memory)
      - Page table hit: fetch translation
      - Page table miss: page fault, fetch page from disk to memory, return translation to TLB
  - Check cache (input: PPN, output: data)
    - hit: return value
    - miss: fetch value from memory

Address Map, Mathematically

V = {0, 1, . . . , n - 1} virtual address space (n > m)
M = {0, 1, . . . , m - 1} physical address space
MAP: V --> M U {θ} address mapping function

MAP(a) = a' if data at virtual address a
is present in physical address a' and a' in M
= θ if data at virtual address a is not present in M



And in Conclusion…

- Virtual memory to Physical Memory Translation too slow?
  - Add a cache of Virtual to Physical Address Translations, called a TLB
- Spatial Locality means Working Set of Pages is all that must be in memory for process to run fairly well
- Virtual Memory allows protected sharing of memory between processes with less swapping to disk