Comp 590-184: Hardware Security and Side-Channels

Lecture 3: Caches

January 15, 2026
Andrew Kwong

THE UNIVERSITY
of NORTH CAROLINA
at CHAPEL HILL

Slides adapted from Roger Dannenberg and Greg Ganger

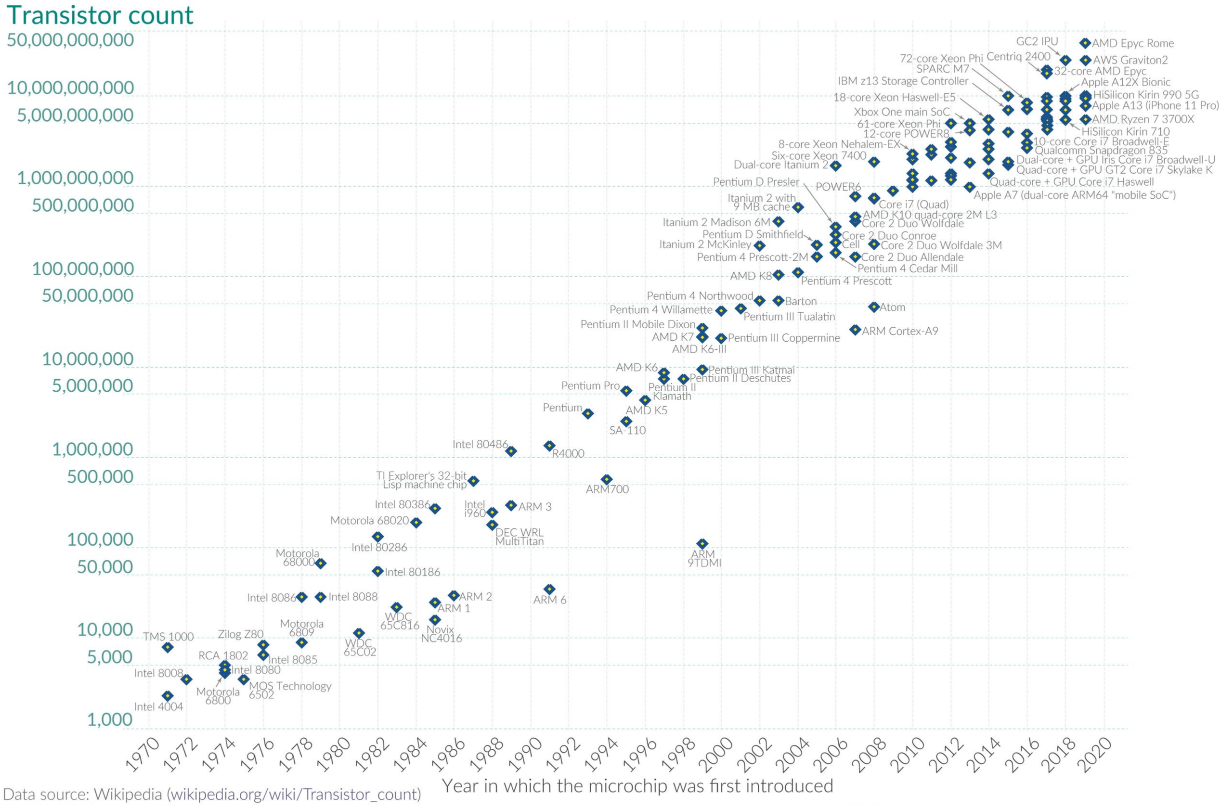
Today’s Class

- Computer Architecture Background
  - General background on caches
  - How caches can be used for side-channels
  - What cache side-channels can accomplish

Moore’s Law: The number of transistors on microchips doubles every two years

Our World in Data

Moore's law describes the empirical regularity that the number of transistors on integrated circuits doubles approximately every two years. This advancement is important for other aspects of technological progress in computing – such as processing speed or the price of computers.



Data source: Wikipedia (wikipedia.org/wiki/Transistor_count)

OurWorldinData.org – Research and data to make progress against the world’s largest problems.

Problem: Processor-Memory Bottleneck

Processor performance doubled about every 18 months

Bus bandwidth evolved much slower

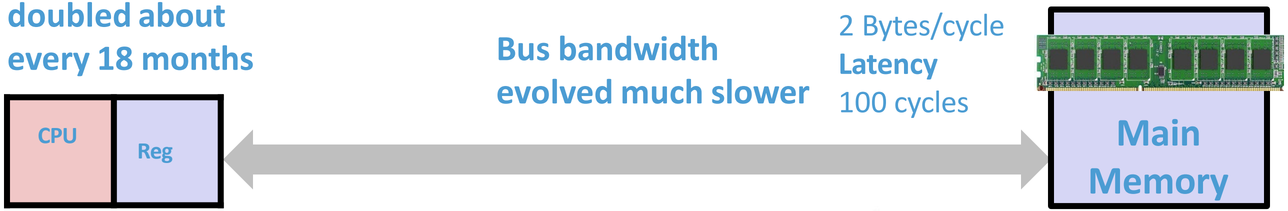
Core 2 Duo:

Bandwidth

2 Bytes/cycle

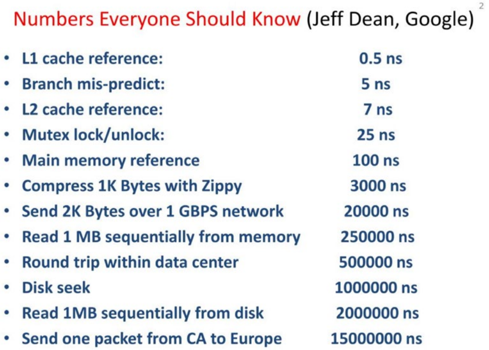
Latency

100 cycles



Can process at least

512 Bytes/cycle

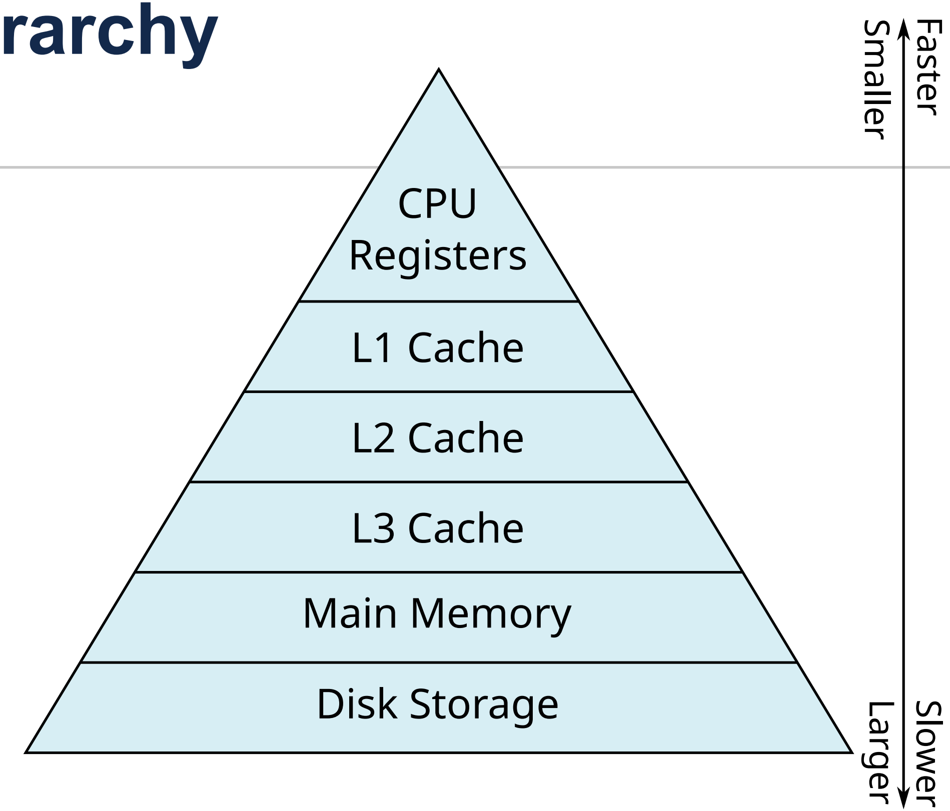


Numbers Everyone Should Know (Jeff Dean, Google) [2]

L1 cache reference:	0.5 ns
Branch mis-predict:	5 ns
L2 cache reference:	7 ns
Mutex lock/unlock:	25 ns
Main memory reference	100 ns
Compress 1K Bytes with Zippy	3000 ns
Send 2K Bytes over 1 GBPS network	20000 ns
Read 1 MB sequentially from memory	250000 ns
Round trip within data center	500000 ns
Disk seek	1000000 ns
Read 1MB sequentially from disk	2000000 ns
Send one packet from CA to Europe	15000000 ns

Solution: Caches

Memory Hierarchy

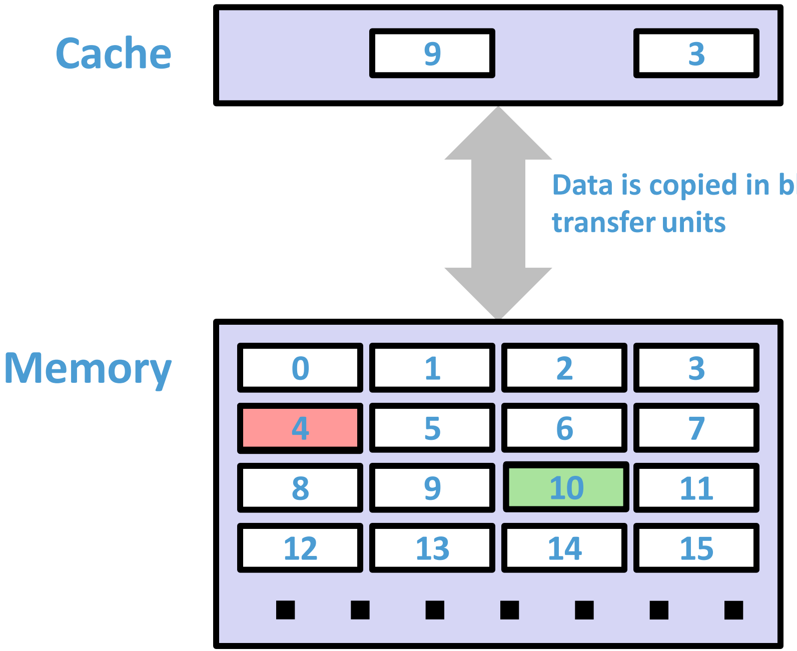


Solution: Caches

Cache

- **Definition:** Computer memory with short access time used for the storage of frequently or recently used instructions or data

General Cache Mechanics

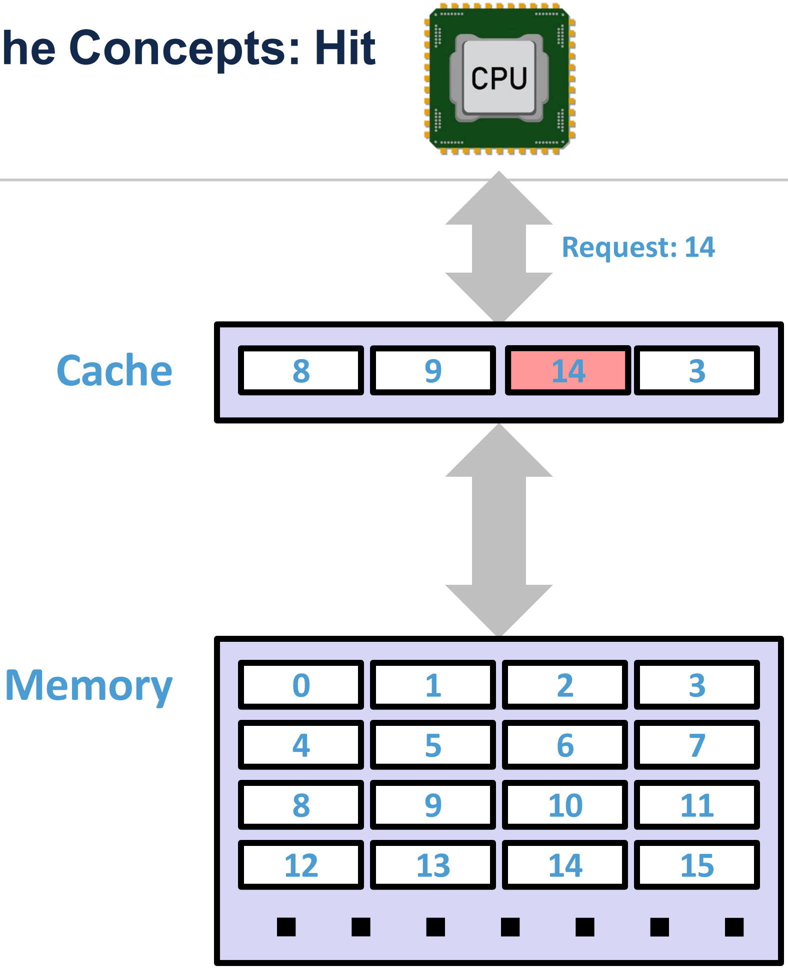


Smaller, faster, more expensive memory caches a subset of the blocks

Data is copied in block-sized transfer units

Larger, slower, cheaper memory viewed as partitioned into “blocks”

General Cache Concepts: Hit



Request: 14

Cache

8	9	14	3

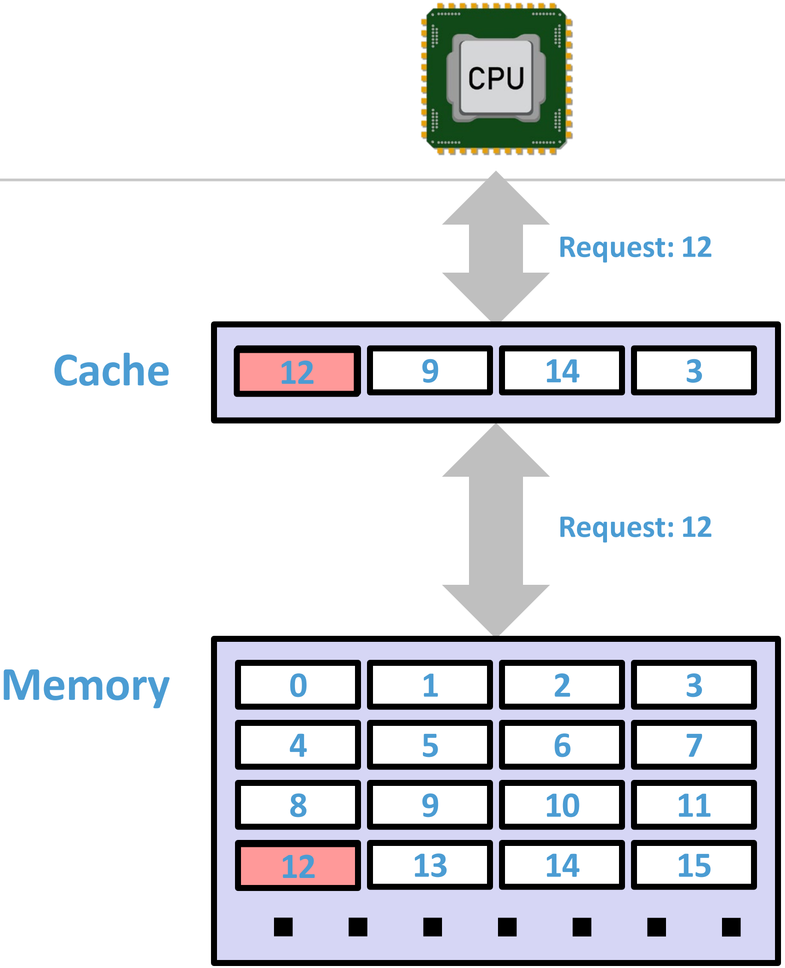
Memory

0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

Data in block b is needed

Block b is in cache:
Hit!

Cache Miss



12	9	14	3

0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

Data in block b is needed

Block b is not in cache:
Miss!

Block b is fetched from memory

Block b is stored in cache

- Placement policy: determines where b goes
- Replacement policy: determines which block gets evicted (victim)

Cache Performance Metrics

- **Miss Rate**
  - Fraction of memory references not found in cache (misses / accesses)
    - = 1 – hit rate
  - Typical numbers (in percentages):
    - 3-10% for L1
    - can be quite small (e.g., < 1%) for L2, depending on size, etc.
- **Hit Time**
  - Time to deliver a line in the cache to the processor
    - includes time to determine whether the line is in the cache
  - Typical numbers:
    - 1-2 clock cycle for L1
    - 5-20 clock cycles for L2
- **Miss Penalty**
  - Additional time required because of a miss
    - typically 50-200 cycles for main memory (Trend: increasing!)

Lets think about those numbers

- **Huge difference between a hit and a miss**
  - Could be 100x, if just L1 and main memory

- **How much better is 99% hit rate vs 97% hit rate?**
  - cache hit time of 1 cycle miss penalty of 100 cycles

  - Average access time:
    - 97% hits: 1*(0.97) cycle + 0.03 * 100 cycles = **3.97 cycles**
    - 99% hits: 1*(0.99) cycle + 0.01 * 100 cycles = **1.99 cycles**

- **This is why "miss rate" is used instead of "hit rate"**

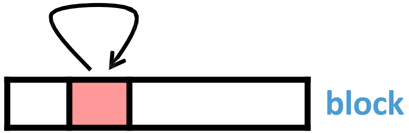
Types of Cache Misses

- **Cold (compulsory) miss**
  - Occurs on first access to a block
- **Conflict miss**
  - Most hardware caches limit blocks to a small subset (sometimes a singleton) of the available cache slots
    - e.g., block i must be placed in slot (i mod 4)
  - Conflict misses occur when the cache is large enough, but multiple data objects all map to the same slot
    - e.g., referencing blocks 0, 8, 0, 8, ... would miss every time
- **Capacity miss**
  - Occurs when the set of active cache blocks (working set) is larger than the cache

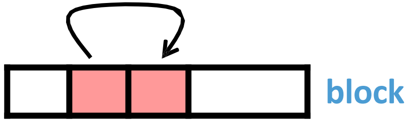
Why Caches Work

- **Locality: Programs tend to use data and instructions with addresses near or equal to those they have used recently**

- **Temporal locality:**
  - Recently referenced items are likely to be referenced again in the near future



- **Spatial locality:**
  - Items with nearby addresses tend to be referenced close together in time



Example: Locality?

```
sum = 0;
for (i = 0; i < n; i++)
          sum += a[i];
return sum;
```

- **Data:**
  - Temporal: **`sum`** referenced in each iteration
  - Spatial: array **`a[]`** accessed in stride-1 pattern
- **Instructions:**
  - Temporal: cycle through loop repeatedly
  - Spatial: reference instructions in sequence

- **Being able to assess the locality of code is a crucial skill for a programmer**

Locality Example

```
int sum_array_cols(int a[M][N])
{
    int i, j, sum = 0;

    for (j = 0; j < N; j++)
        for (i = 0; i < M; i++)
            sum += a[i][j];
    return sum;
}
```

```
int sum_array_rows(int a[M][N])
{
    int i, j, sum = 0;

    for (i = 0; i < M; i++)
        for (j = 0; j < N; j++)
            sum += a[i][j];
    return sum;
}
```

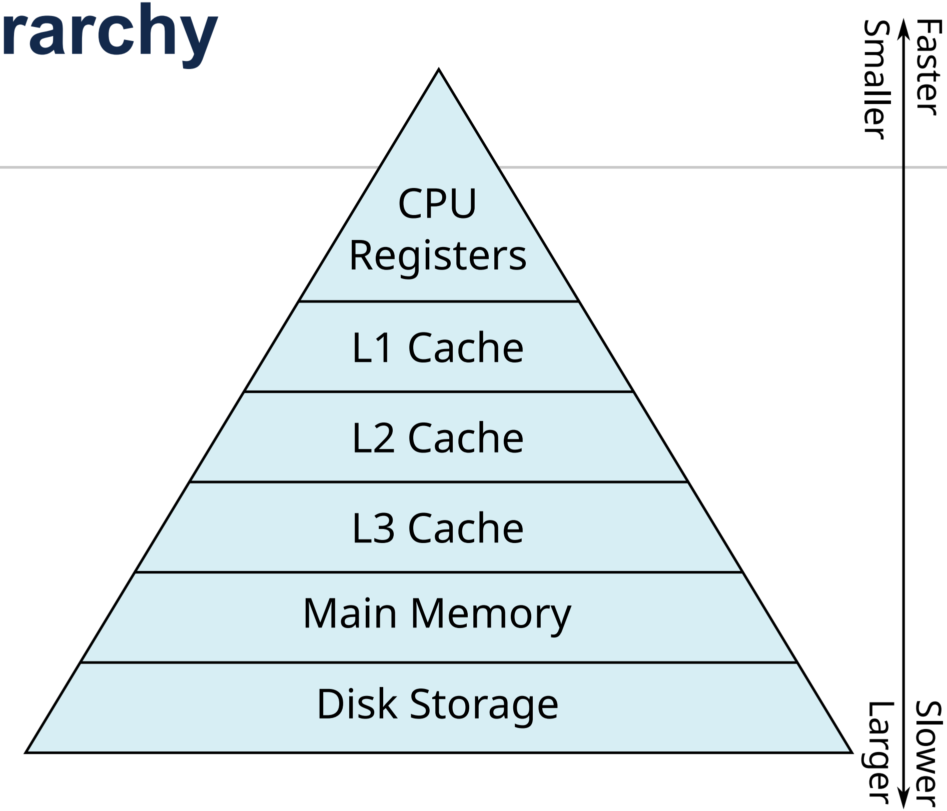
Memory Hierarchies

- **Some fundamental and enduring properties of hardware and software systems:**
  - Faster storage technologies almost always cost more per byte and have lower capacity
  - The gaps between memory technology speeds are widening
    - True of registers ↔ DRAM, DRAM ↔ disk, etc.
  - Well-written programs tend to exhibit good locality

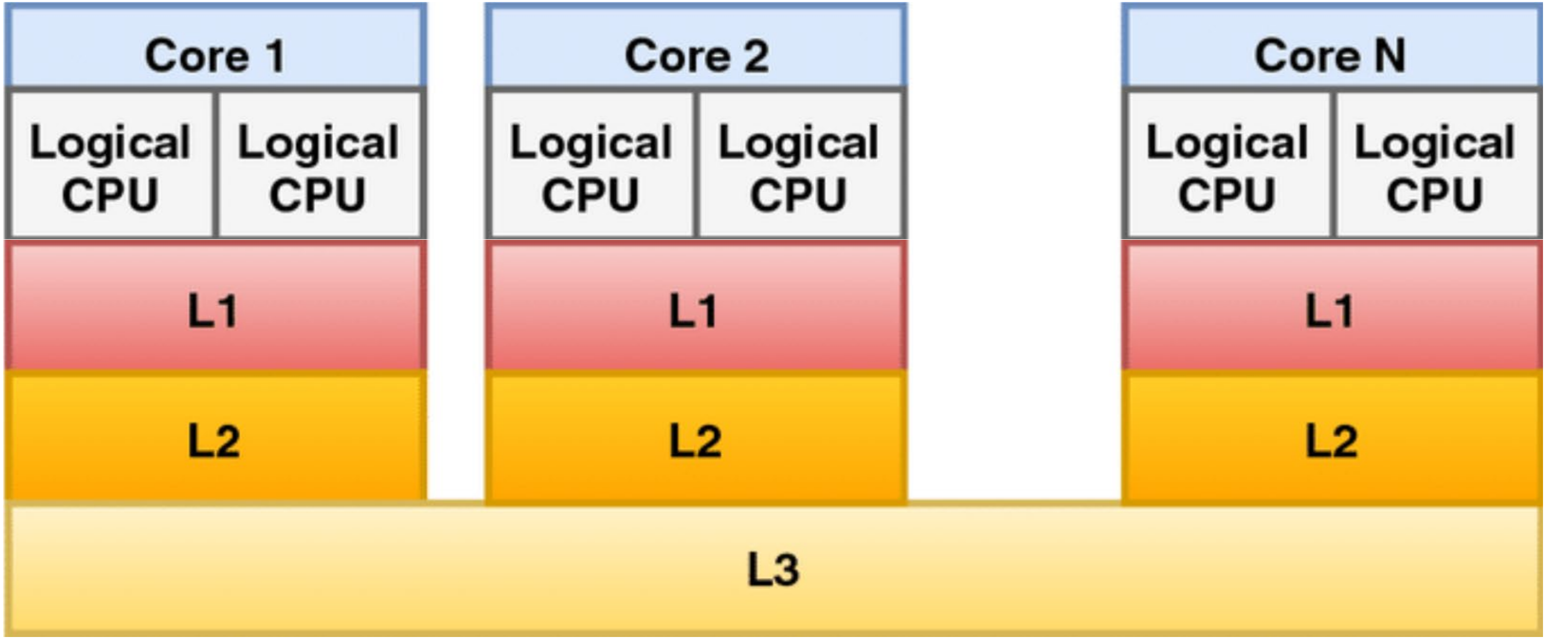
- **These properties complement each other beautifully**

- **They suggest an approach for organizing memory and storage systems known as a memory hierarchy**

Memory Hierarchy



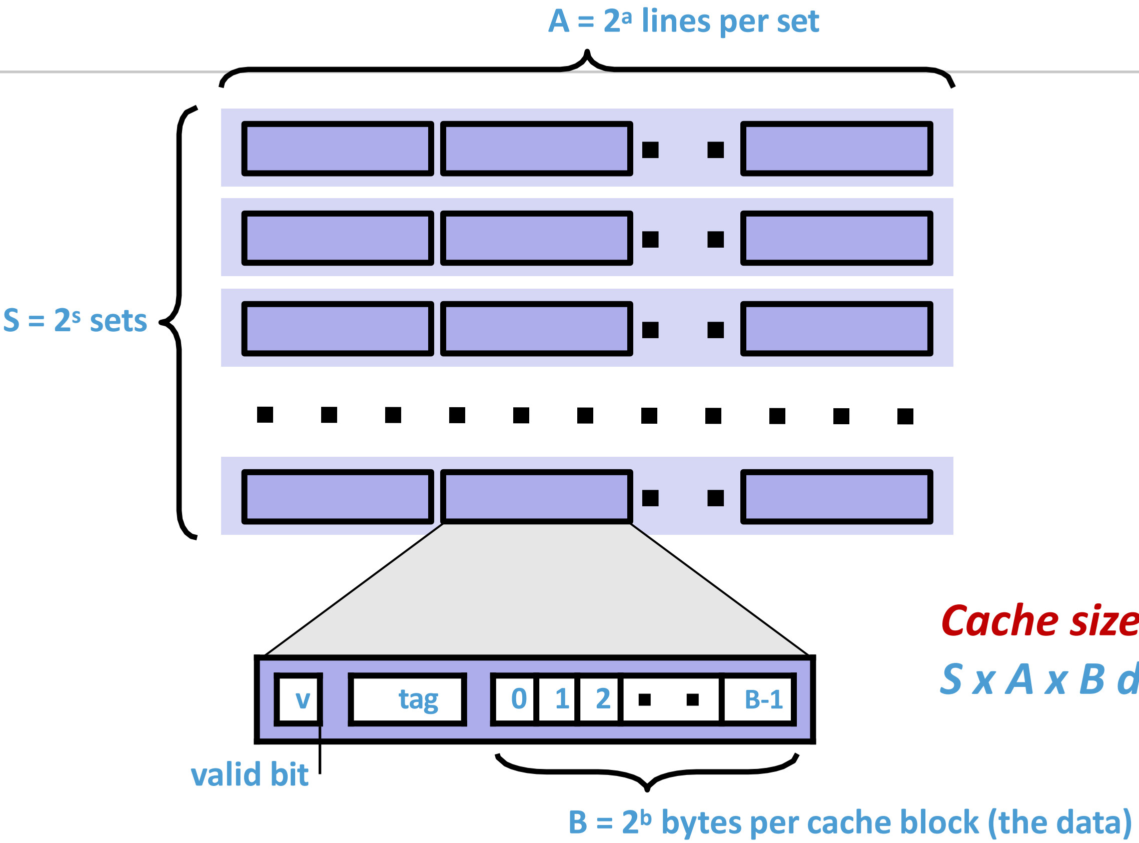
Hierarchy on a modern CPU



Examples of Caching in the Hierarchy

Cache Type	What is Cached?	Where is it Cached?	Latency (cycles)	Managed By
Registers	8-byte words	CPU core	0	Compiler
TLB	Address translations	On-Chip TLB	0	Hardware
L1 cache	64-bytes block	On-Chip L1	1	Hardware
L2 cache	64-bytes block	On-Chip L2	10	Hardware
Virtual Memory	4-KB page	Main memory	100	Hardware+OS
Page cache	Parts of files	Main memory	100	OS
Network buffer cache	Parts of files	Local disk	10,000,000	File system client
Browser cache	Web pages	Local disk	10,000,000	Web browser
CDN	Web pages	Remote server disks	1,000,000,000	CDN

General Cache Organization (S, A, B)

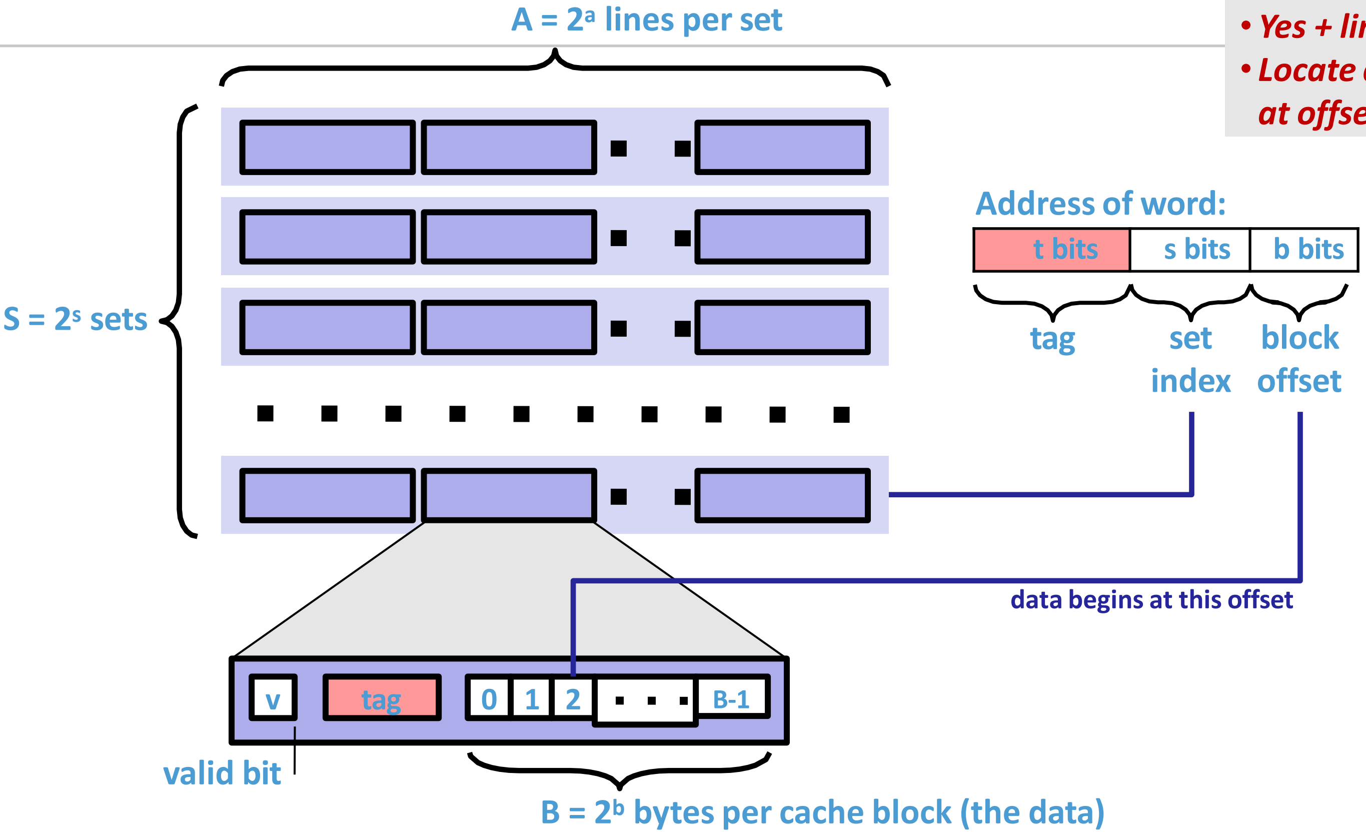


Cache size:

S x A x B data bytes

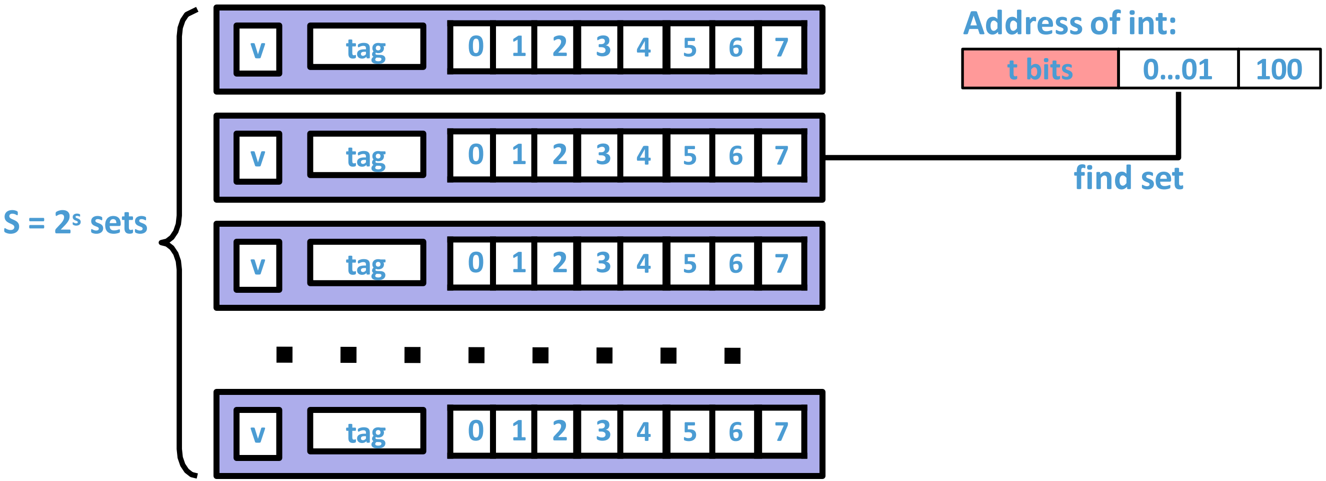
Cache Read

- *Locate set*
- *Check if any line in set has matching tag*
- *Yes + line valid: hit*
- *Locate data starting at offset*



Example: Direct Mapped Cache (A = 1)

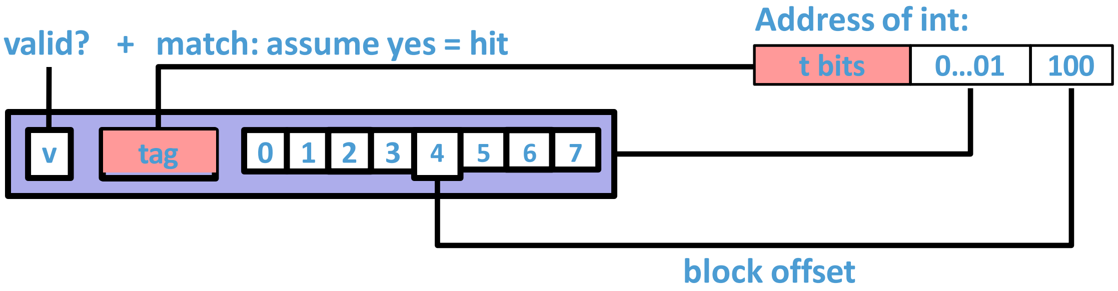
Direct mapped: One line per set
Assume: cache block size 8 bytes



Example: Direct Mapped Cache (A = 1)

Direct mapped: One line per set

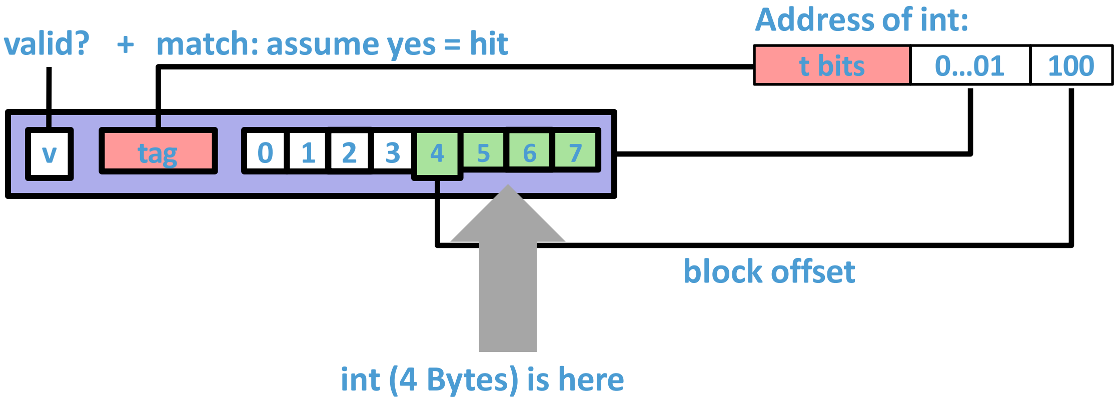
Assume: cache block size 8 bytes



Example: Direct Mapped Cache (A = 1)

Direct mapped: One line per set

Assume: cache block size 8 bytes

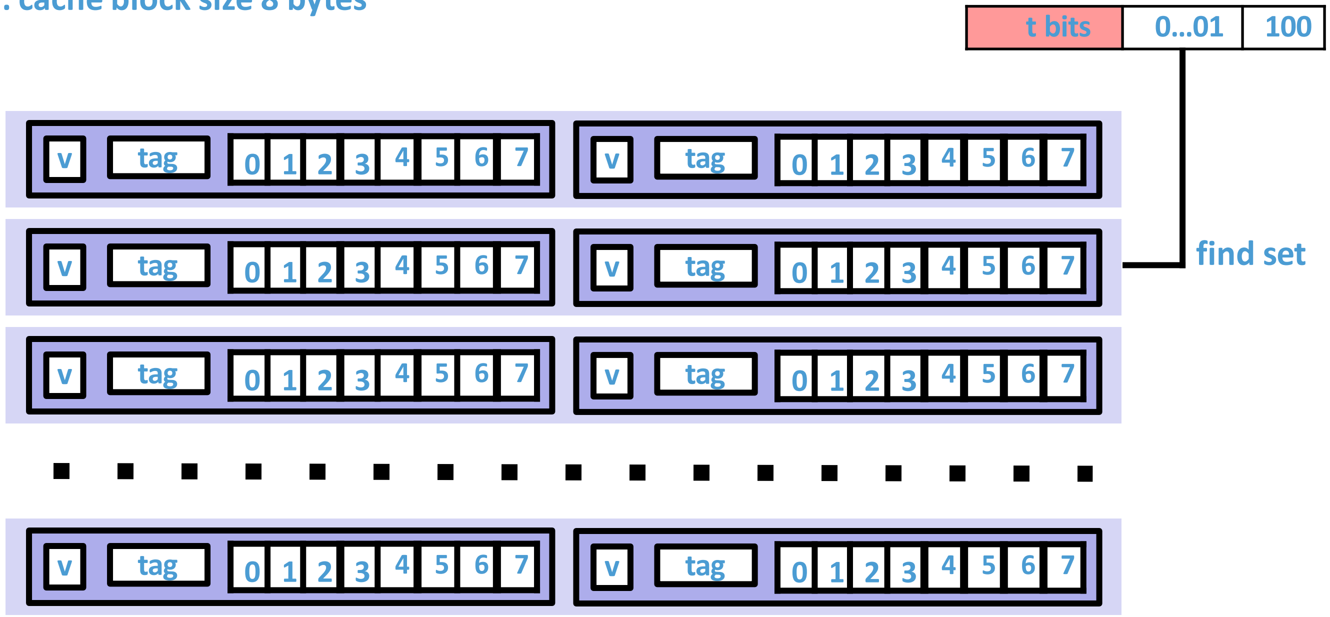


No match: old line is evicted and replaced

E-way Set Associative Cache (Here: A = 1)

E = 2: Two lines per set

Assume: cache block size 8 bytes

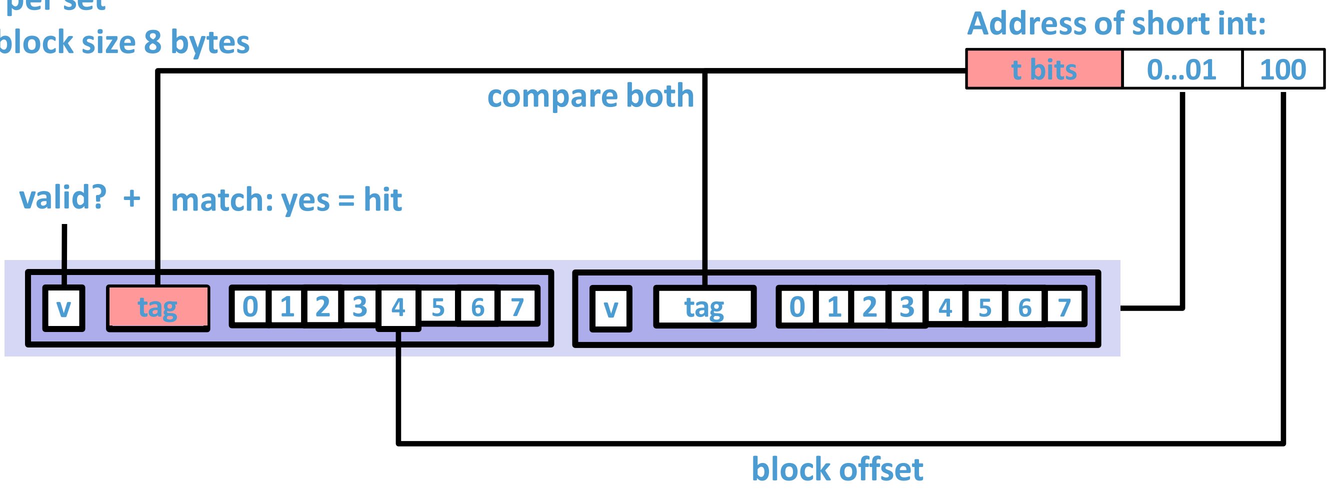


Address of short int:

A-way Set Associative Cache (Here: A = 2)

A = 2: Two lines per set

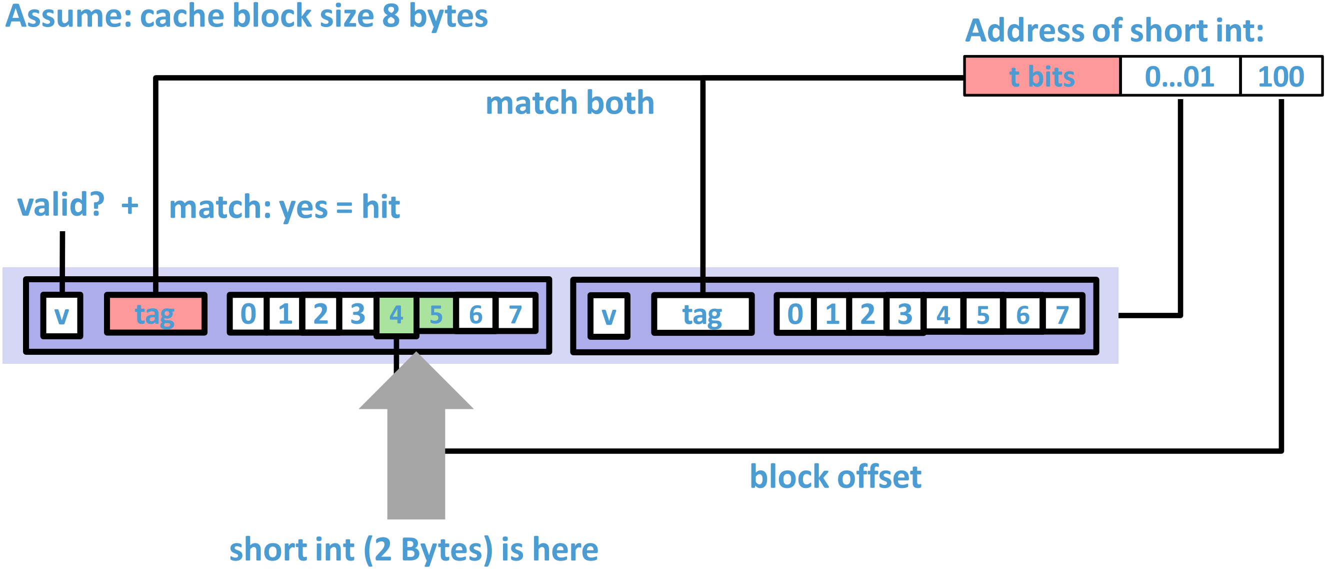
Assume: cache block size 8 bytes



A-way Set Associative Cache (Here: A = 2)

A = 2: Two lines per set

Assume: cache block size 8 bytes



No match:

- One line in set is selected for eviction and replacement

Replacement Policy

- Optimally: replace block that is accessed furthest in the future

- Locality argument
    - Hasn’t been used recently, less likely to be used in future
- LRU: replace line that was least recently used

What about writes?

- **Multiple copies of data exist:**
  - L1, L2, Main Memory, Disk
- **What to do on a write-hit?**
  - Write-through (write immediately to memory)
  - Write-back (defer write to memory until replacement of line)
    - Need a dirty bit (line different from memory or not)
- **What to do on a write-miss?**
  - Write-allocate (load into cache, update line in cache)
    - Good if more writes to the location follow
  - No-write-allocate (writes immediately to memory)

THE UNIVERSITY
of NORTH CAROLINA
at CHAPEL HILL