TEXAS TECH UNIVERSITY
Department *of* Computer Science

CS5375 Computer Systems Organization and Architecture

Lecture 8

Instructor: Yong Chen, Ph.D.
Department of Computer Science
Texas Tech University
Yong.Chen@ttu.edu, 806-834-0284

Announcements

- Learn to use HPCC systems for both programming projects
  - Please watch Part 1 and Part 2 videos and check out slides too, to learn how to use the system
  - https://www.depts.ttu.edu/hpcc/about/training.php#new_user_training
  - Key parts you must study for now:
    - Part 1, “Logging and Using the Cluster”
    - Part 2, “Interactive Session”
  - If you’re not familiar with Linux, then you must further study “Introduction to Linux”
  - https://www.depts.ttu.edu/hpcc/about/training.php#intro_linux
- Once your account is ready, you can try your account following the training videos

Review of Last Lecture

- Measuring Cache Performance
  - Write-through cache, write-back cache
  - Average memory access time (AMAT)
    - AMAT = Hit time + Miss rate × Miss penalty
- Associative Caches
  - Fully associative
  - n-way set associative
  - Spectrum of associativity
  - Examples
- Multilevel Caches
  - Focusing on reducing miss penalty
  - Examples

Outline

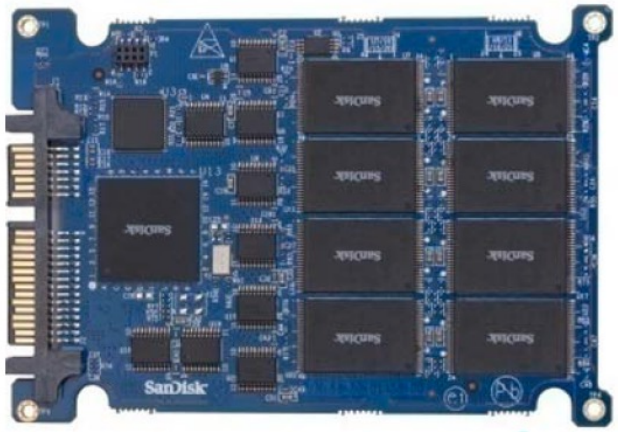
- Memory Technology and Optimizations
- Optimizations of Cache Performance

Memory Technology

- Static RAM (SRAM) (“cache”, integrated onto the processor chip)
  - Low latency, used for cache, as latency is concern of cache

- Dynamic RAM (DRAM)
  - Organize DRAM chips into many banks for high bandwidth

- Flash memory (USB flash drives, solid state drives)



- Magnetic disk (hard disk drives)

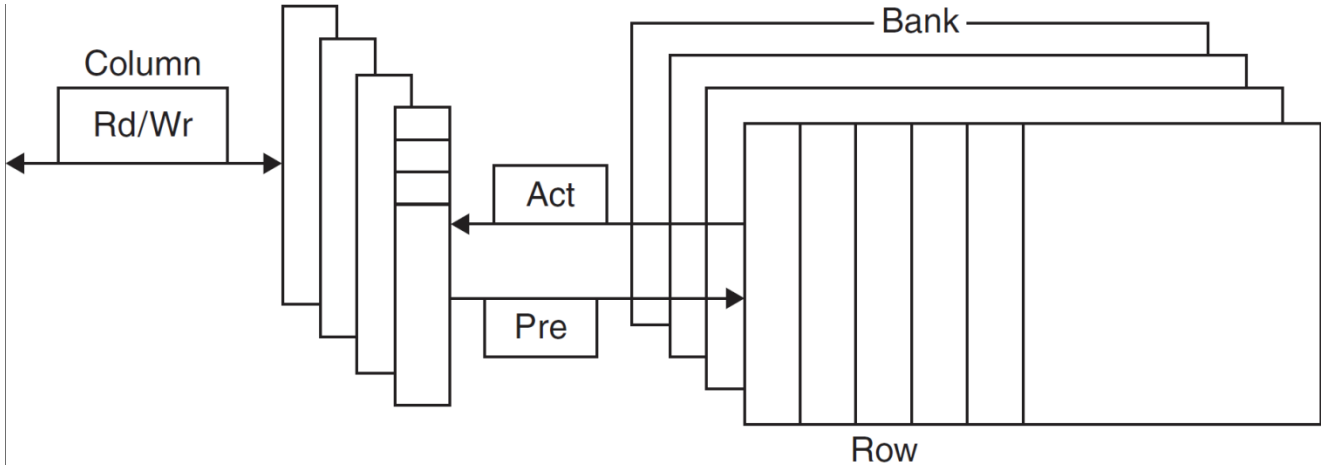
Memory Technology (cont.)

- Ideal memory
  - Access time of SRAM
  - Capacity and cost/GB of magnetic disk

Memory technology	Typical access time	$ per GiB in 2020
SRAM semiconductor memory	0.5–2.5 ns	$500–$1000
DRAM semiconductor memory	50–70 ns	$3–$6
Flash semiconductor memory	5,000–50,000 ns	$0.06–$0.12
Magnetic disk	5,000,000–20,000,000 ns	$0.01–$0.02

DRAM Technology

- Data stored as a charge in a capacitor
  - Single transistor used to access the charge
  - Must periodically be refreshed
    - Read contents and write back
    - Performed on a DRAM “row”
  - Double data rate (DDR) DRAM: data transfer on both rising and falling clock edges

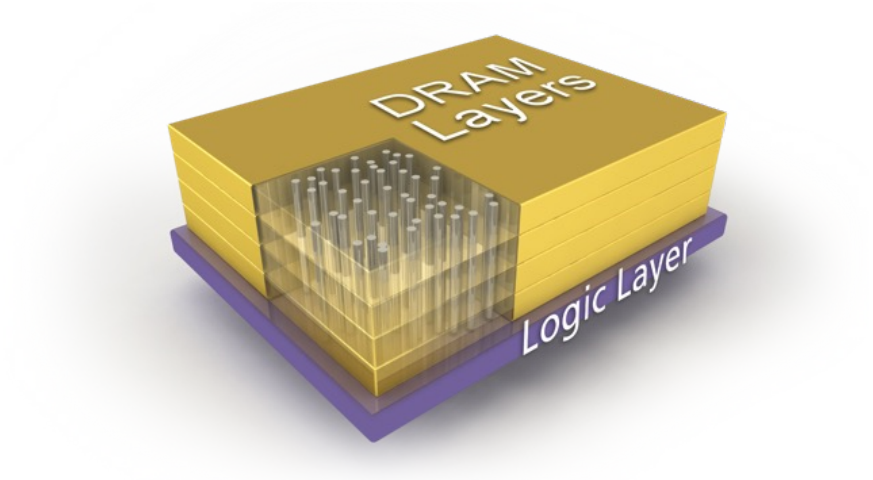


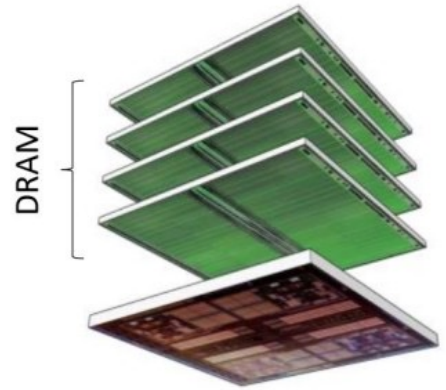
DRAM Generations

Year introduced	Chip size	$ per GiB	Total access time to a new row/column	Average column access time to existing row
1980	64 Kibibit	$1,500,000	250 ns	150 ns
1983	256 Kibibit	$500,000	185 ns	100 ns
1985	1 Mebibit	$200,000	135 ns	40 ns
1989	4 Mebibit	$50,000	110 ns	40 ns
1992	16 Mebibit	$15,000	90 ns	30 ns
1996	64 Mebibit	$10,000	60 ns	12 ns
1998	128 Mebibit	$4,000	60 ns	10 ns
2000	256 Mebibit	$1,000	55 ns	7 ns
2004	512 Mebibit	$250	50 ns	5 ns
2007	1 Gibibit	$50	45 ns	1.25 ns
2010	2 Gibibit	$30	40 ns	1 ns
2012	4 Gibibit	$1	35 ns	0.8 ns

3D-Stacked Memory

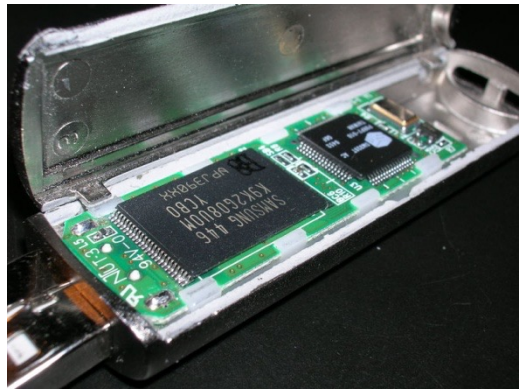
- Hybrid Memory Cube (HMC) by Micron, and High Bandwidth Memory (HBM) by Samsung, AMD and Hynix
- ~10x-15x improvements over DDR3
- Power consumption remains a limiting factor





Flash Memory

- Non-volatile semiconductor storage, flash drives, solid state drives (SSDs)
  - 100x – 1000x faster than magnetic disk
  - Smaller, lower power, more robust
  - But more $/GB (between disk and DRAM)



Flash Memory

- Stores data in array of floating gate transistors (no movable parts)
- Non-volatile: FG electrically isolated by an insulating layer -> no discharge in many years

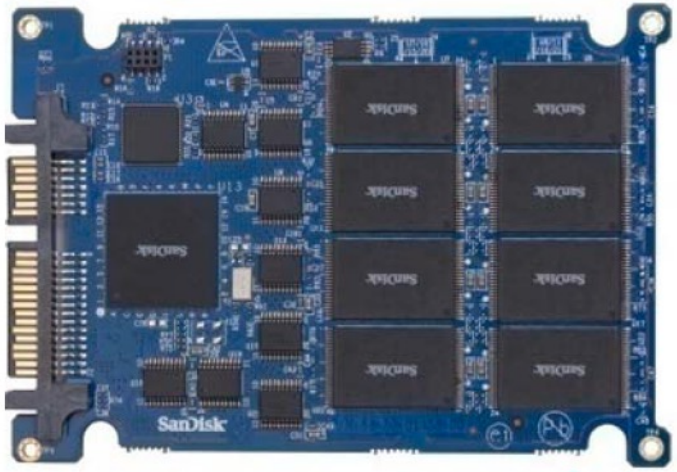
- Types: NOR, NAND

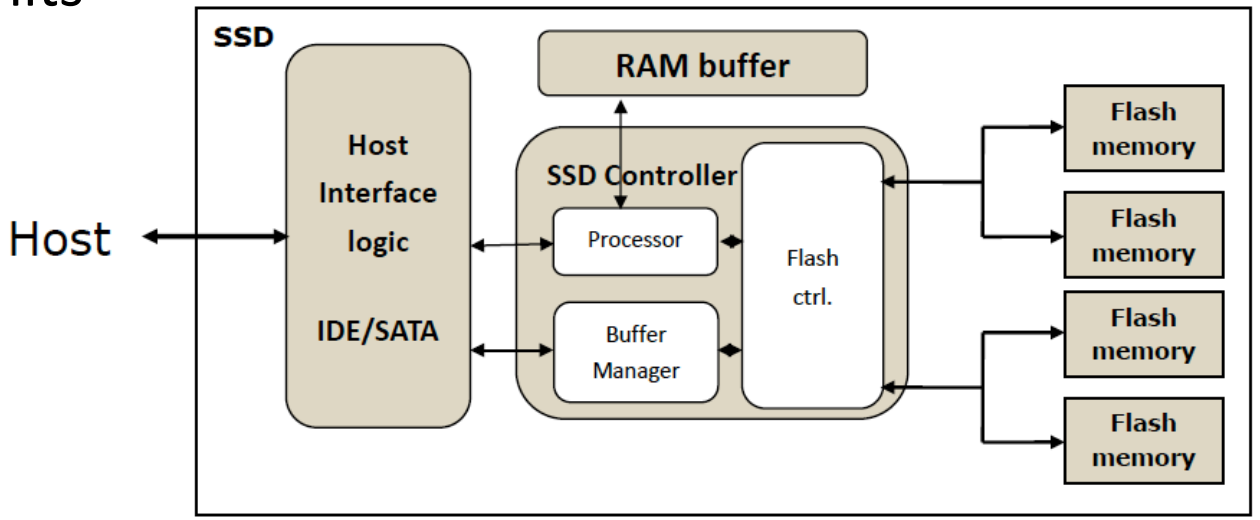
- **SLC: Single-Level Cell**
  - One bit in each cell
  - More reliable, more expensive, less capacity

- **MLC: Multi-Level Cell**
  - Multiple bits in one cell
  - Distinguishing the voltage
  - More capacity, less expensive, less reliable

Solid State Drives Internal

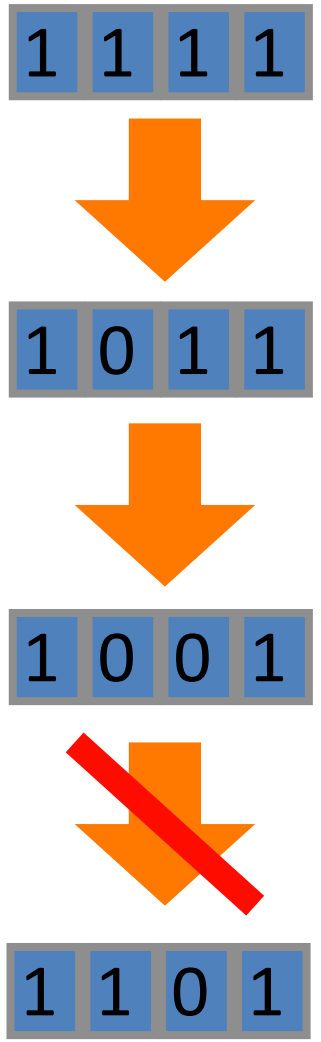
- Host interface logic
- SSD controller
  - ASIC/FPGA
  - Flash translation layer (FTL)
- RAM buffer
- Flash memory units





Limitations – Block Erasure

- Properties of NAND Flash
  - Cells can only be programmed (set to zero), i.e. bits can be set to zero individually, but not back to one
  - Changing bits from zero to one requiring erasing the block (i.e. block erasure, changing all bits set to one), then set rest bits to zero
- When block erasure needed (only occurs when 0->1 needed)
  - Instead of in-place writes that need to wait for block erasure, we can write to an empty block
  - Virtual block now maps to a new flash memory unit
  - Flash Translation Layer (FTL) manages such mapping on the fly
- Writes needing block erasure should be avoided where possible



Limitations - NAND Wear out

- NAND cells can only go through a limited number of program/erase (P/E) cycles
- Applications might want to overwrite the same blocks of data repeatedly
  - Critical for FTL to balance wearing out (wear leveling)

- P/E cycles are getting better, but still a concern
  - 10K-100K P/E cycles

- Active research ongoing

Outline

- Memory Technology and Optimizations
- Optimizations of Cache Performance

Cache Optimization Techniques

- Six basic cache optimizations:
  - Larger block size
    - Reduces compulsory misses
    - Increases capacity and conflict misses, increases miss penalty
  - Larger total cache capacity to reduce miss rate
    - Increases hit time, increases power consumption
  - Higher associativity
    - Reduces conflict misses
    - Increases hit time, increases power consumption

How Much Associativity

- Increased associativity decreases miss rate
  - But with diminishing returns
  - And with more hardware cost (more comparators needed)

- Simulation of a system with 64KB D-cache (data cache), 16-word blocks, SPEC CPU2000 benchmarks

Associativity	Data miss rate
1	10.3%
2	8.6%
4	8.3%
8	8.1%

Cache Optimization Techniques (cont.)

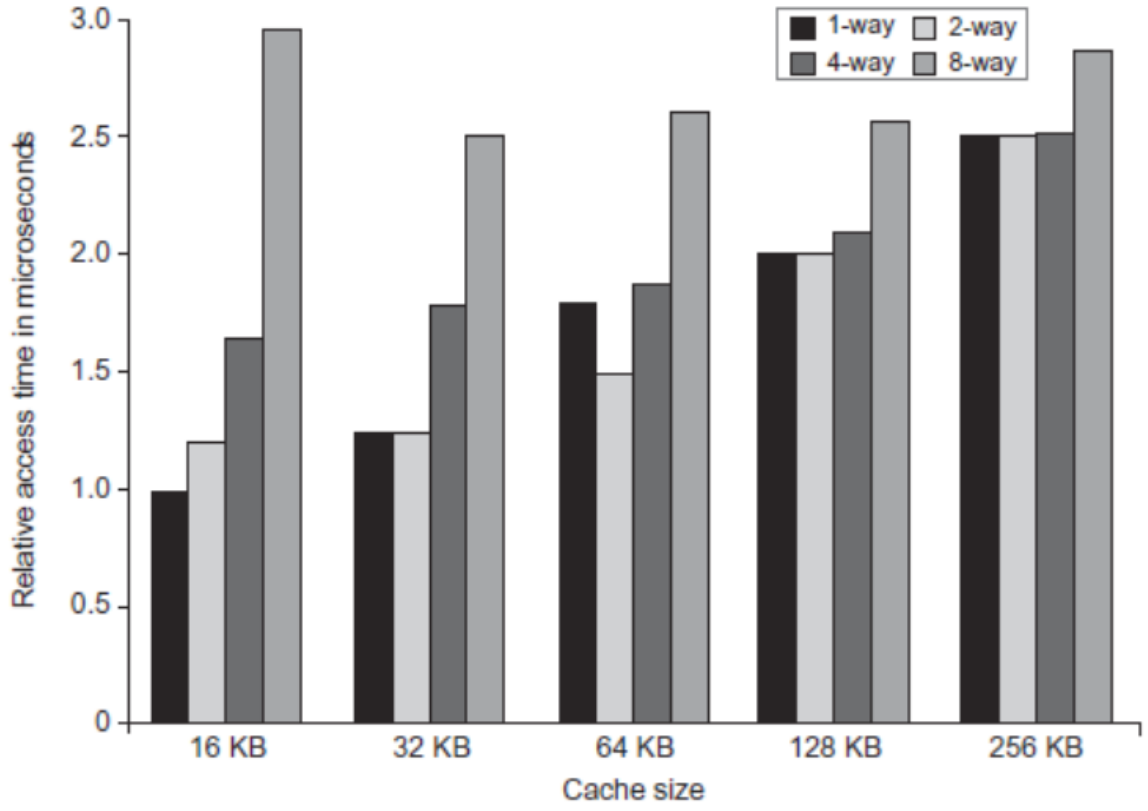
- Six basic cache optimizations (cont.):
  - Higher number of cache levels
    - Reduces overall memory access time
  - Giving priority to read misses over writes
    - Reduces miss penalty
  - Avoiding address translation in cache indexing
    - Reduces hit time

Advanced Optimizations

- Reduce hit time
  - Small and simple first-level caches
  - Way prediction
- Increase bandwidth
  - Non-blocking caches
- Reduce miss penalty
  - Critical word first, merging write buffers
- Reduce miss rate
  - Compiler optimizations

Reduce Hit Time

- Small and simple first-level caches to reduce hit time



L1 access time vs. size and associativity

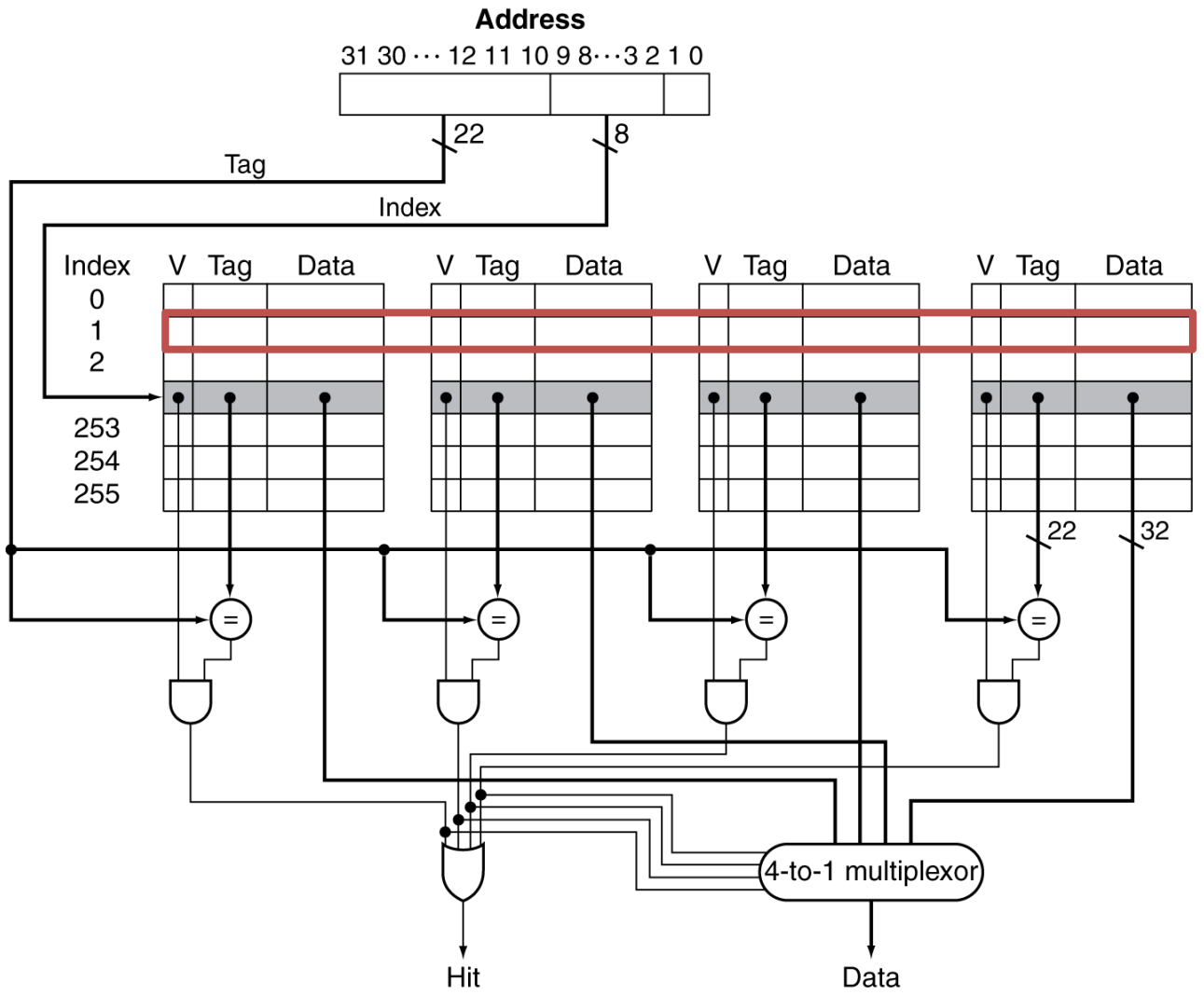
Reduce Hit Time (cont.)

- Way prediction to reduce hit time
  - To improve hit time, predict the way to of the next cache access and pre-set the multiplexor
  - Mis-prediction gives longer hit time

- Prediction accuracy
  - For two-way: > 90%
  - For four-way: > 80%

- I-cache has better accuracy than D-cache

Reduce Hit Time (cont.)



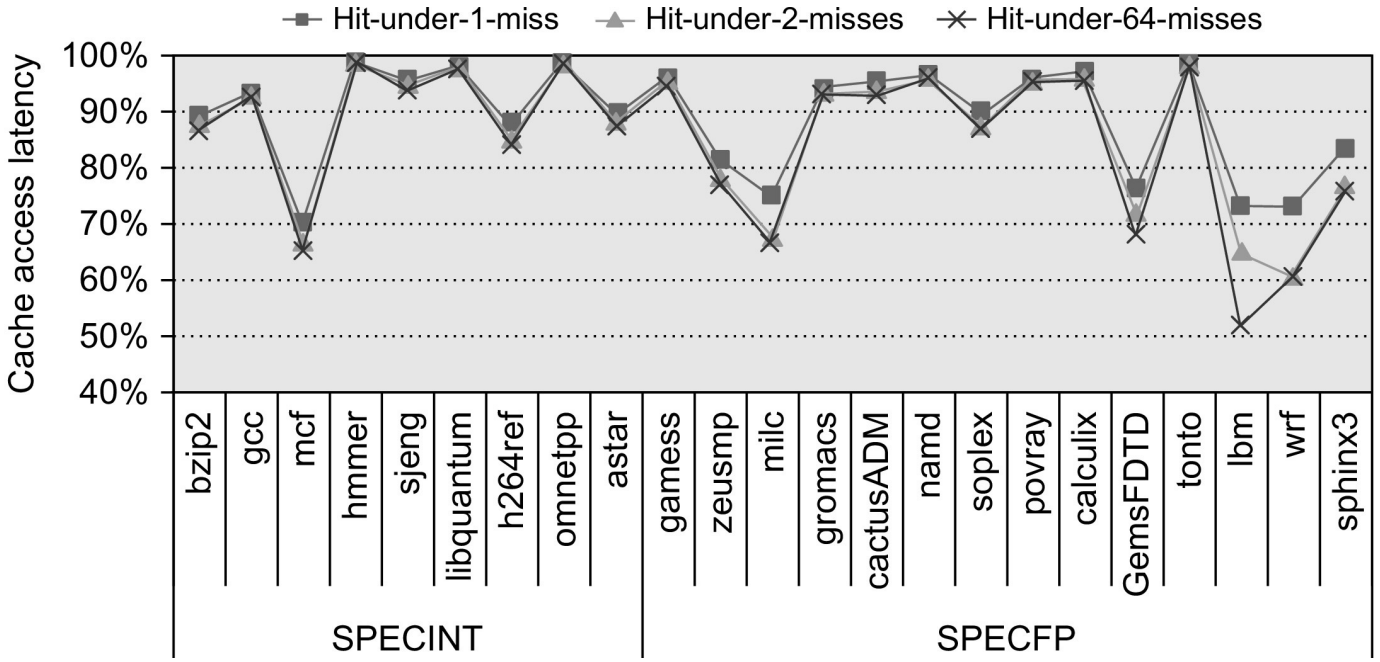
256 sets in total

One set, 4-way

A four-way set-associative cache requires four comparators and a 4-to-1 multiplexor

Increase Bandwidth: Nonblocking Caches

- Allow hits before previous misses complete
  - "Hit under miss"
  - "Hit under multiple miss"
- L2 must support this
- In general, processors can hide L1 miss penalty but not L2 miss penalty



Allowing one hit under miss reduces the miss penalty by 9% for the integer benchmarks and 12.5% for the floating point.

Allowing a second hit improves these results to 10% and 16%, and allowing 64 results in little additional improvement.

Reduce Miss Penalty: Critical Word First, Early Restart

- Critical word first
  - Request missed word from memory first
  - Send it to the processor as soon as it arrives

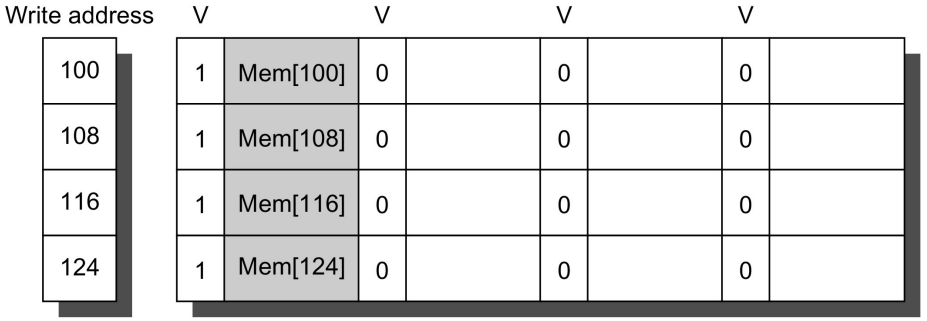
- Early restart
  - Request words in normal order
  - Send missed word to the processor as soon as it arrives

- Effectiveness of these strategies depends on block size and likelihood of another access to the portion of the block that has not yet been fetched

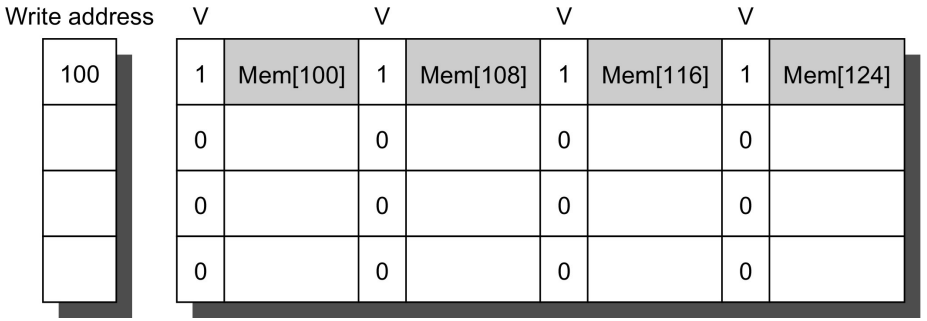
Reduce Miss Penalty: Merging Write Buffer

- When storing to a block that is already pending in the write buffer, update/merge write buffer if the address of new data matches the address of a valid entry
- Reduces stalls due to full write buffer

The buffer has four entries, and each entry holds four 64-bit words



No write buffer merging



Write buffer merging

Reduce Miss Rate: Compiler Optimizations

- Loop Interchange
  - Swap nested loops to access memory in sequential order

```
for (j = 0; j < 100; j = j + 1)
    for (i = 0; i < 5000; i = i + 1)
        x[i][j] = 2 * x[i][j];
```

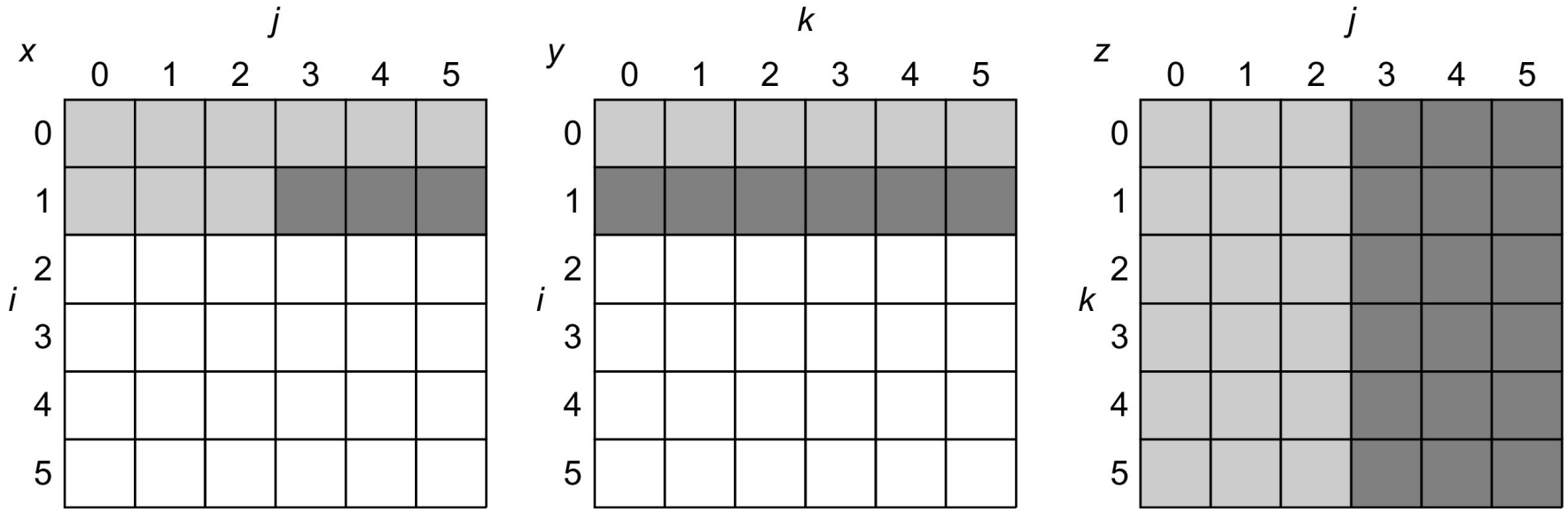
```
for (i = 0; i < 5000; i = i + 1)
    for (j = 0; j < 100; j = j + 1)
        x[i][j] = 2 * x[i][j];
```

The left-side code would skip through memory in strides of 100 words, while the right-side version accesses all the words in one cache block before going to the next block.

This optimization improves cache performance without affecting the number of instructions executed

Reduce Miss Rate: Compiler Optimizations (cont.)

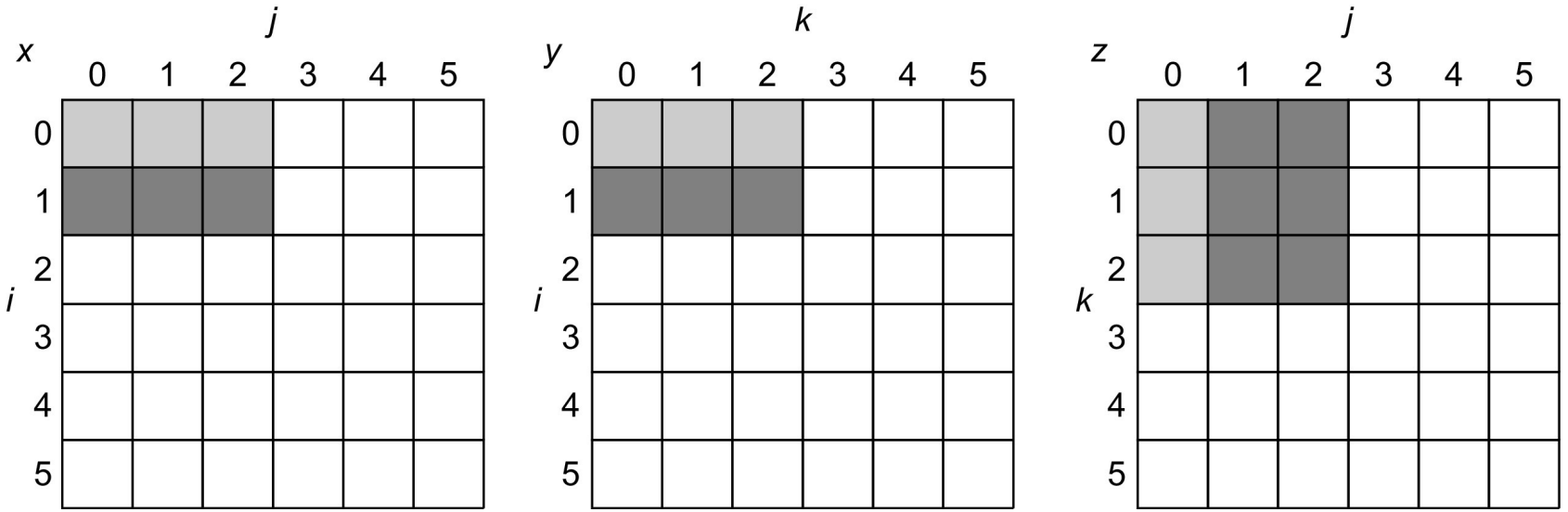
```
for (i = 0; i < N; i = i + 1)
  for (j = 0; j < N; j = j + 1)
  {
    r = 0;
    for (k = 0; k < N; k = k + 1)
      r = r + y[i][k]*z[k][j];
    x[i][j] = r;
};
```



Reduce Miss Rate: Compiler Optimizations (cont.)

```
for (jj = 0; jj < N; jj = jj + B)
  for (kk = 0; kk < N; kk = kk + B)
    for (i = 0; i < N; i = i + 1)
      for (j = jj; j < min(jj + B,N); j = j + 1)
      {
        r = 0;
        for (k = kk; k < min(kk + B,N); k = k + 1)
          r = r + y[i][k]*z[k][j];
        x[i][j] = x[i][j] + r;
};
```

- Blocking technique
- Instead of accessing entire rows or columns, subdivide matrices into blocks
- Improves locality of accesses



Readings

- Chapter 2, 2.2 - 2.3