CS/ECE 5381/7381 Computer Architecture Spring 2023

Dr. Manikas

Computer Science

Lecture 15: Mar. 23, 2023

Assignments

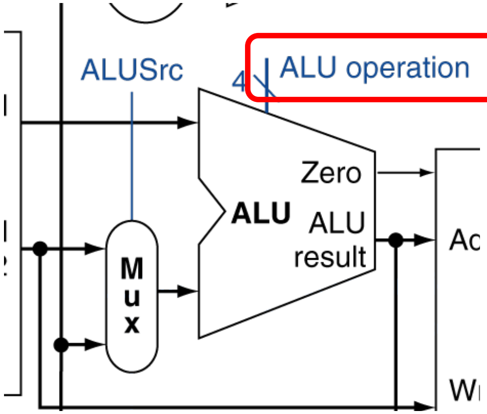
- Quiz 7 – due Sat., Mar. 25 (11:59 pm)
  - Covers concepts from Module 8 (this week)
- NEXT WEEK
  - Project 4 – due Tue., Mar. 28
  - Exam 2 – Mar. 30 – Apr. 1

Quiz 7 Details

- The quiz is open book and open notes.
- You are allowed 90 minutes to take this quiz.
- You are allowed 2 attempts to take this quiz - your highest score will be kept.
  - Note that some questions (e.g., fill in the blank) will need to be graded manually
- Quiz answers will be made available 24 hours after the quiz due date.

Project 4

- Due next **Tues., Mar. 28** (11:59 pm)
- Cadence Xcelium tool
  - Used to develop and test Verilog code
- Verilog
  - Hardware Description Language
  - Used to construct and simulate computer hardware
- Assignment:
  - Run tool on simple MIPS ALU design



Simple MIPS ALU – only does two functions.

Use Xcelium tool to test this ALU, using the provided test bench

ALU control	Function	Description
0001	OR	Bitwise OR
0111	set-on-less-than	True if A < B, false otherwise

Exam 2

- Exam will be administered using Lockdown Browser (same as for Exam 1)
- Exam format also same as Exam 1
  - 25 questions, 2 hours
  - The exam will be available from **Thursday, Mar. 30 at 12 am**
  - The exam must be completed and submitted by **Saturday, Apr 1 at 11:59 pm**

Exam 2

- **Exam 2 will cover the following materials:**
  - Modules: 5 – 8
  - Quizzes: 5 - 7
  - Text: Ch. 3.1 – 3.6, App. B.1 – B.4, Ch. 2.1 – 2.3
- **MATERIALS ALLOWED FOR EXAM:**
  - Open book and notes
  - Calculator

Memory Design Hierarchy

(Chapter 2, Hennessy and Patterson)

Note: some course slides adopted from publisher-provided material

Outline

- 2.1 Introduction
- 2.2 Memory Technology and Optimizations
- 2.3 Ten Advanced Optimizations of Cache Performance
- 2.4 Virtual Memory and Machines

Six Basic Cache Optimizations

1. Larger block size
   - Reduces compulsory misses
   - Increases capacity and conflict misses, increases miss penalty
2. Larger total cache capacity to reduce miss rate
   - Increases hit time, increases power consumption
3. Higher associativity
   - Reduces conflict misses
   - Increases hit time, increases power consumption

Six Basic Cache Optimizations

4. Higher number of cache levels
   - Reduces overall memory access time
5. Giving priority to read misses over writes
   - Reduces miss penalty
6. Avoiding address translation in cache indexing
   - Reduces hit time

Outline

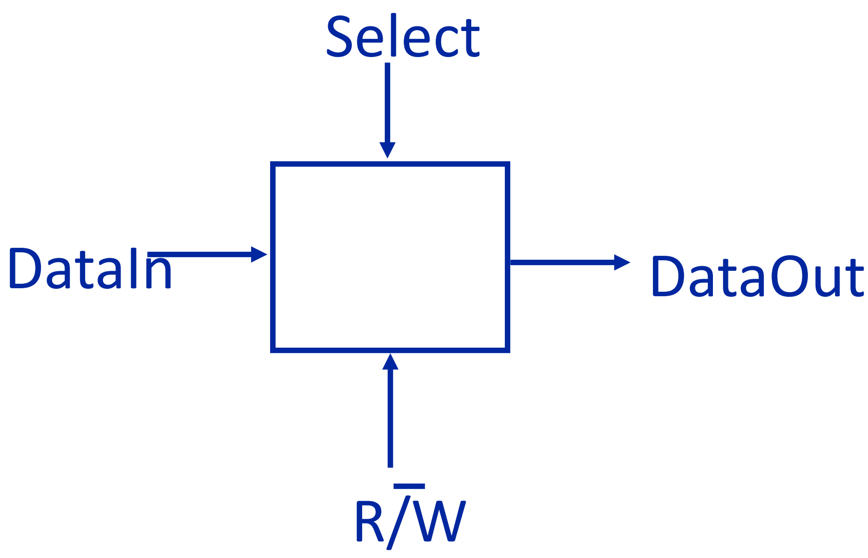
- 2.1 Introduction
- 2.2 Memory Technology and Optimizations
- 2.3 Ten Advanced Optimizations of Cache Performance
- 2.4 Virtual Memory and Machines

RAM (Random-Access Memory)

- Cache and main memory are RAM
- Types of memory access
  - Random-access: all cells can be accessed in equal time
  - Sequential: cells must be accessed in sequence (think of a tape)
  - Disk-access: time varies dependant on location of R/W head

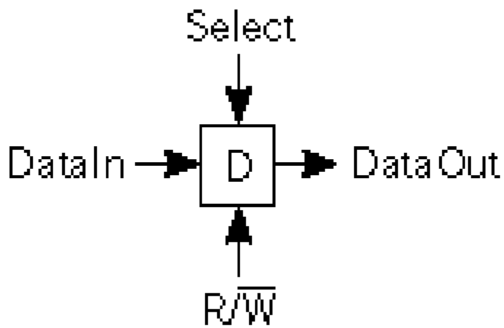
Memory Cells - a conceptual view

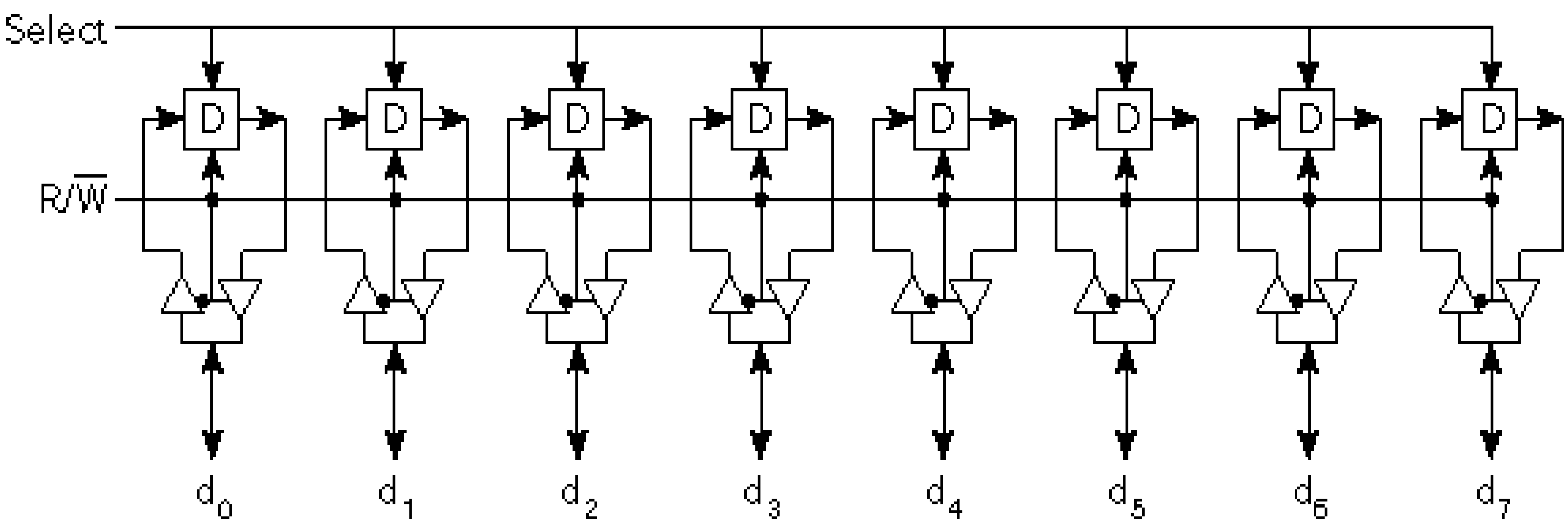
Regardless of the technology, all RAM memory cells must provide these four functions: Select, DataIn, DataOut, and R/W.



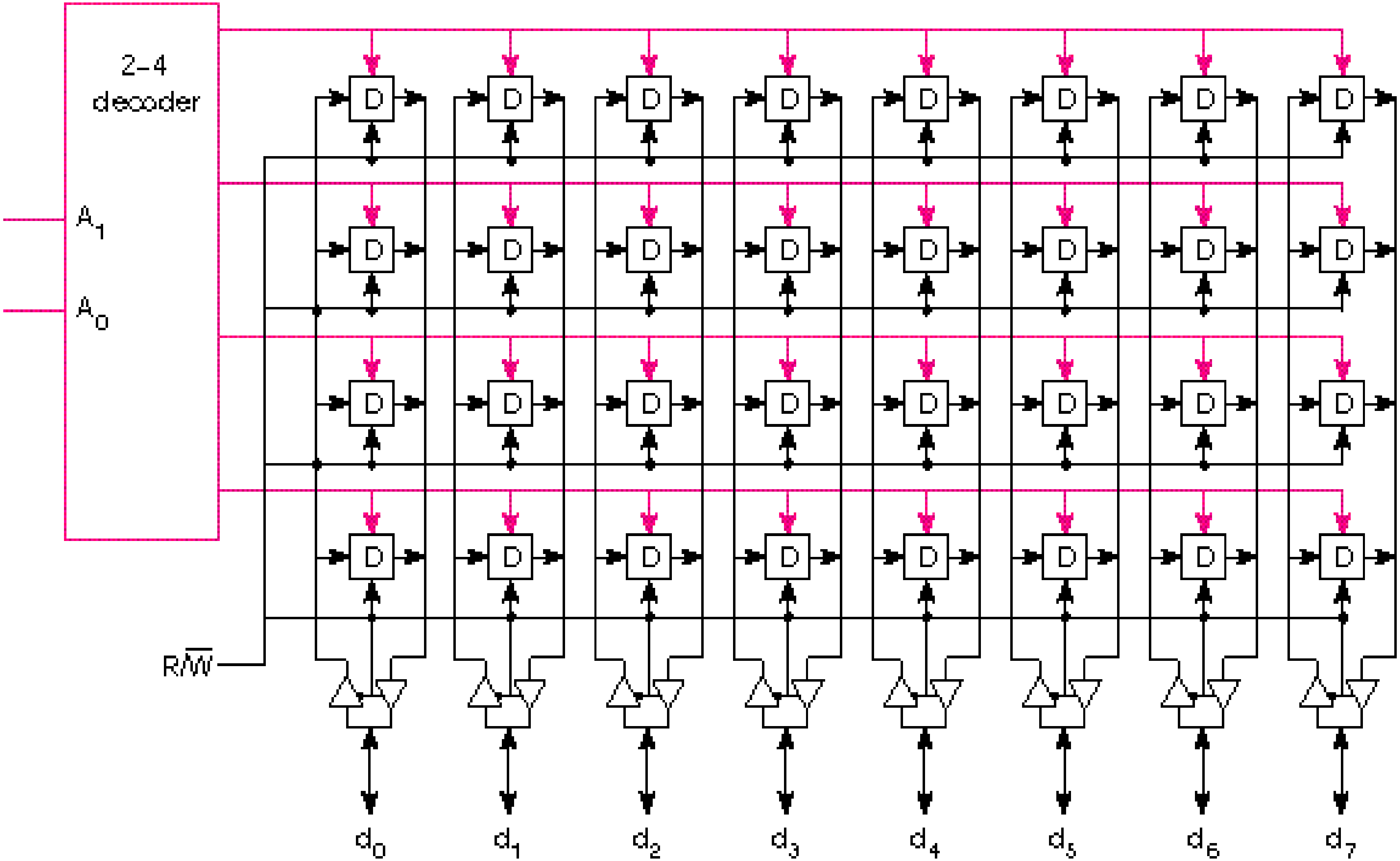
An 8-bit register as a 1D RAM array

The entire register is selected with one select line, and uses one





A 4x8 2D Memory Cell Array

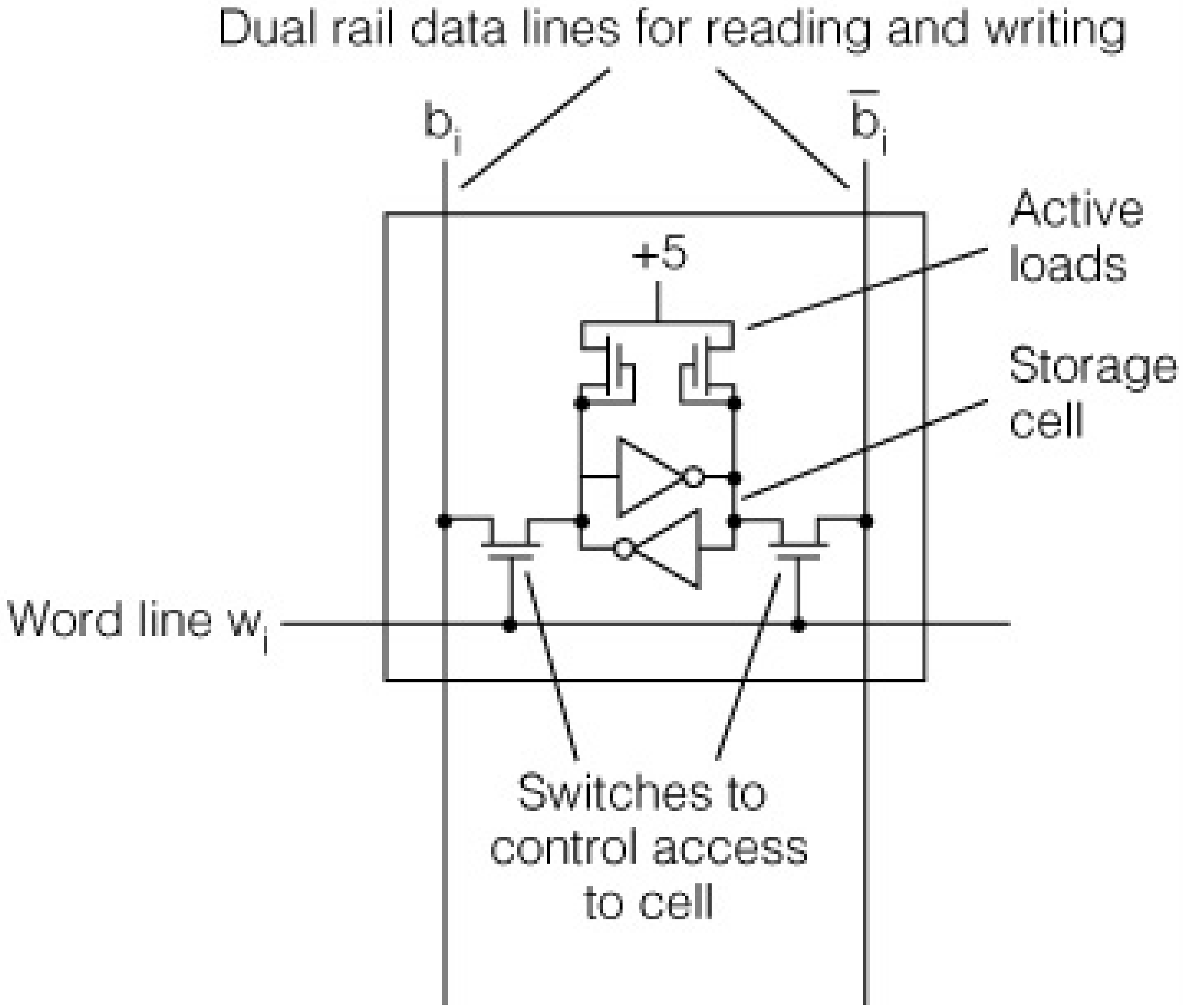


Memory Technology

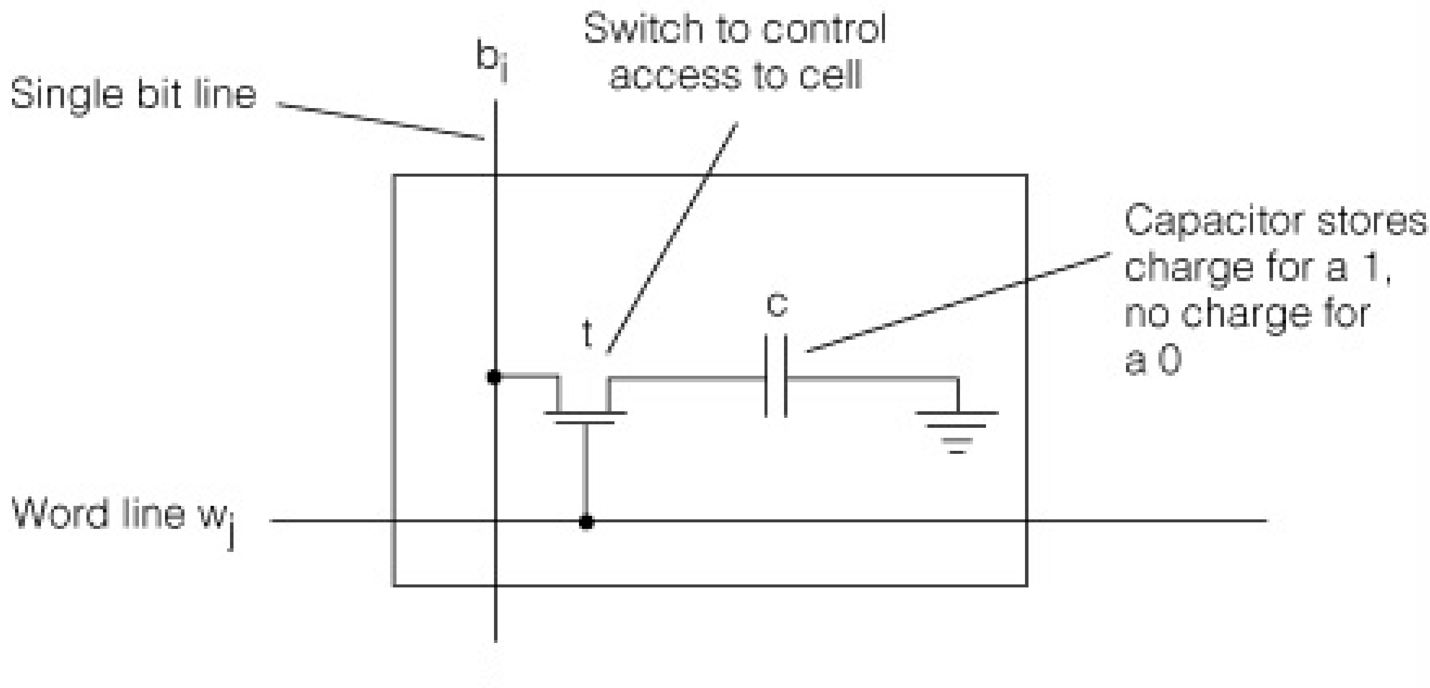
- Performance metrics
  - Latency is concern of cache
  - Bandwidth is concern of multiprocessors and I/O
  - Access time
    - Time between read request and when desired word arrives
  - Cycle time
    - Minimum time between unrelated requests to memory

- DRAM used for main memory, SRAM used for cache

SRAM (Static RAM) cell



DRAM (Dynamic RAM) cell



Memory Technology

- SRAM
  - Requires low power to retain bit
  - Requires 6 transistors/bit

- DRAM
  - Must be re-written after being read
  - Must also be periodically refreshed
    - Every ~ 8 ms
    - Each row can be refreshed simultaneously
  - One transistor/bit
  - Address lines are multiplexed:
    - Upper half of address: row access strobe (RAS)
    - Lower half of address: column access strobe (CAS)

Memory Cell Applications

- Main Memory is ***DRAM***
  - Dynamic since needs to be refreshed periodically (8 ms, 1% time)
- Cache uses ***SRAM***
  - No refresh (6 transistors/bit vs. 1 transistor)
    Size: DRAM/SRAM *4-8*,
    Cost/Cycle time: SRAM/DRAM *8-16*

Memory Technology

- Amdahl:
  - Memory capacity should grow linearly with processor speed
  - Unfortunately, memory capacity and speed has not kept pace with processors
- Some optimizations:
  - Multiple accesses to same row
  - Synchronous DRAM
    - Added clock to DRAM interface
    - Burst mode with critical word first
  - Wider interfaces
  - Double data rate (DDR)
  - Multiple banks on each DRAM device

Memory Optimizations

- DDR:
  - DDR2
    - Lower power (2.5 V -> 1.8 V)
    - Higher clock rates (266 MHz, 333 MHz, 400 MHz)
  - DDR3
    - 1.5 V
    - 800 MHz
  - DDR4
    - 1-1.2 V
    - 1333 MHz

- GDDR5 is graphics memory based on DDR3

Flash Memory

- Non volatile
  - Retains values when powered off
  - Cf: volatile: RAM
- Faster than disk drive (magnetic disks)
  - More expensive than disk drive per bit, but becoming more affordable
  - Most modern laptops has SSD (Solid-State Drive)
    - This is flash memory

Memory Dependability

- Memory is susceptible to cosmic rays
- *Soft errors*: dynamic errors
  - Detected and fixed by error correcting codes (ECC)
- *Hard errors*: permanent errors
  - Use sparse rows to replace defective rows

Outline

- 2.1 Introduction
- 2.2 Memory Technology and Optimizations
- 2.3 Ten Advanced Optimizations of Cache Performance
- 2.4 Virtual Memory and Machines

10 Advanced Cache Optimizations

- Reducing hit time

1. Small and simple caches
2. Way prediction

1. Fast Hit times via Small and Simple Caches

- Time-consuming part of cache hit
  - Use index part of address to read tag memory
  - Compare tag memory to block address
- Recall multilevel caches
  - L1 = small cache – cycle time similar to CPU
  - L2 = large cache – holds more blocks

1. Fast Hit times via Small and Simple Caches (cont)

- Small cache can help hit time since smaller memory takes less time to index
  - E.g., L1 caches same size for 3 generations of AMD microprocessors: K6, Athlon, and Opteron
  - Also L2 cache that is small enough to fit on chip with the processor avoids time penalty of going off chip
- Simple $\Rightarrow$ direct mapping
  - Can overlap tag check with data transmission since no choice
  - We know that data will be in row (group) X
  - Tag will verify that data is from column Y

Example 2.3-1

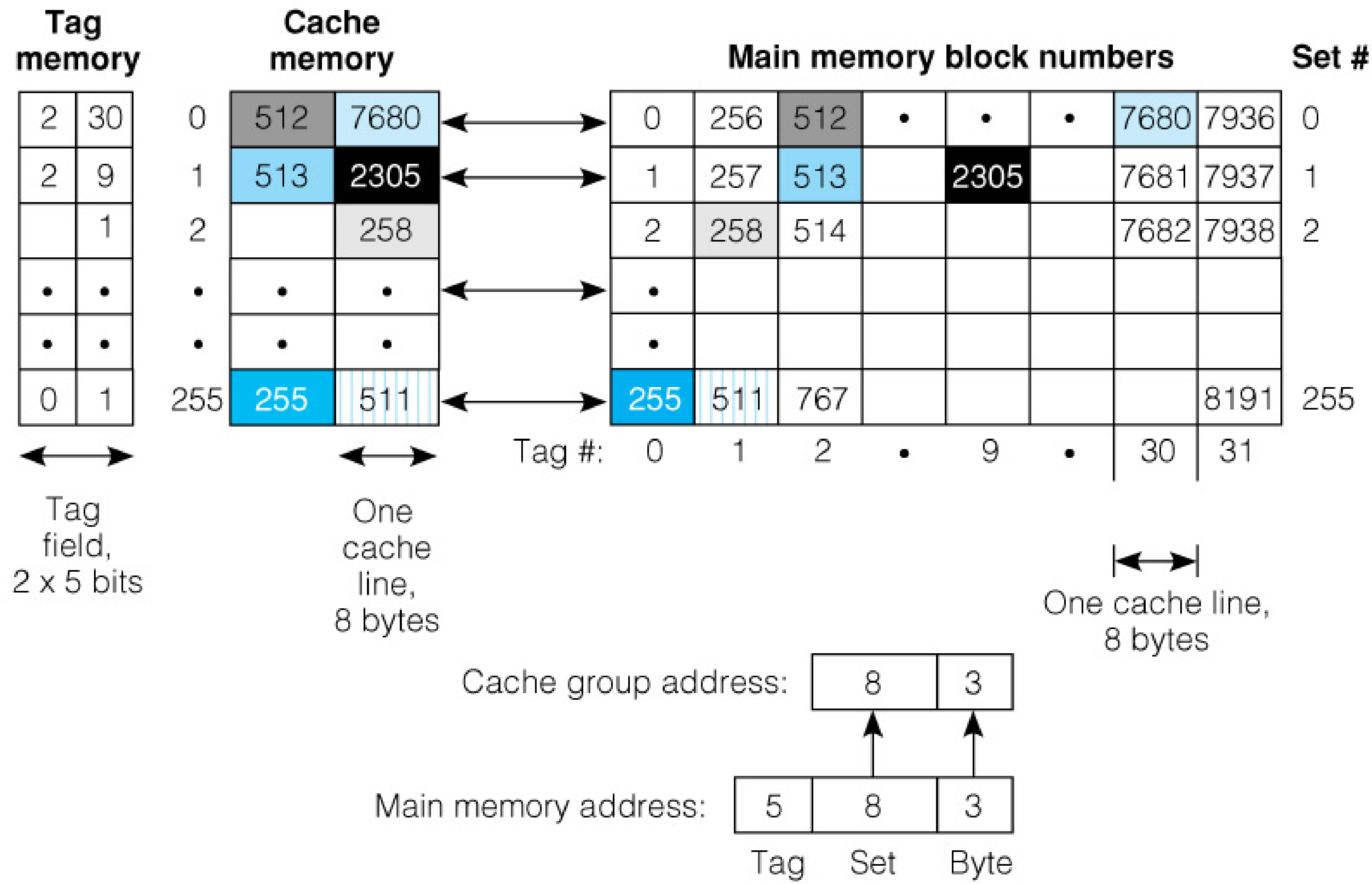
We have a 64 KB 2-way set-associative L2 cache that stores data as 64-byte blocks. To fetch a block from main memory, it takes 8 clock cycles for the first 8 bytes of the block (latency), then 1 clock cycle per 8 bytes for the remaining bytes of the block.

How many clock cycles are required to fetch one block from main memory into the cache (that is, what is the miss penalty for this cache)?

2. Fast Hit times via Way Prediction

- How to combine fast hit time of Direct Mapped and have the lower conflict misses of 2-way SA cache?
- Way prediction: keep extra bits in cache to predict the "way," or block within the set, of next cache access.
  - If correct, hit time is cache access latency
  - If incorrect, try other block and change way predictor to that block

2-Way Set Associative Cache



2. Fast Hit times via Way Prediction (cont)

- Accuracy ≈ 85%
- Drawback: CPU pipeline is hard if hit takes 1 or 2 cycles
  - Used for instruction caches vs. data caches

10 Advanced Cache Optimizations

- Increasing cache bandwidth

3. Pipelined caches
4. Multibanked caches
5. Nonblocking caches

3: Increasing Cache Bandwidth by Pipelining

- Pipeline cache access to maintain bandwidth, but higher latency
- Instruction cache access pipeline stages:

- $\Rightarrow$ greater penalty on mispredicted branches
- $\Rightarrow$ more clock cycles between the issue of the load and the use of the data

4: Increasing Cache Bandwidth via Multiple Banks

- Rather than treat the cache as a single monolithic block, divide into independent banks that can support simultaneous accesses
  - E.g.,T1 (“Niagara”) L2 has 4 banks
- Banking works best when accesses naturally spread themselves across banks $\Rightarrow$ mapping of addresses to banks affects behavior of memory system

4: Increasing Cache Bandwidth via Multiple Banks (cont)

- Simple mapping that works well is "sequential interleaving"
  - Spread block addresses sequentially across banks
  - E.g., if there 4 banks, Bank 0 has all blocks whose address modulo 4 is 0; bank 1 has all blocks whose address modulo 4 is 1; …

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

5. Increasing Cache Bandwidth: Non-Blocking Caches

- *Non-blocking cache* or *lockup-free cache* allow data cache to continue to supply cache hits during a miss
  - requires F/E bits on registers or out-of-order execution
  - requires multi-bank memories
- "*hit under miss*" reduces the effective miss penalty by working during miss vs. ignoring CPU requests

5. Increasing Cache Bandwidth: Non-Blocking Caches (cont)

- "*hit under multiple miss*" or "*miss under miss*" may further lower the effective miss penalty by overlapping multiple misses
  - Significantly increases the complexity of the cache controller as there can be multiple outstanding memory accesses
  - Requires multiple memory banks (otherwise cannot support)
  - Pentium Pro allows 4 outstanding memory misses

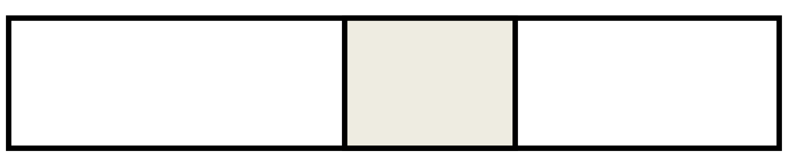
10 Advanced Cache Optimizations

- Reducing Miss Penalty

6. Critical word first
7. Merging write buffers

6. Reduce Miss Penalty: Early Restart

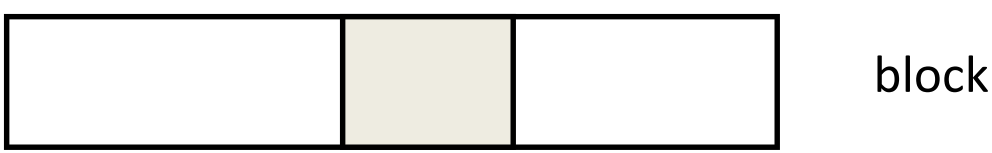
- Don’t wait for full block before restarting CPU
- ***Early restart***—As soon as the requested word of the block arrives, send it to the CPU and let the CPU continue execution
  - Spatial locality $\Rightarrow$ tend to want next sequential word, so not clear size of benefit of just early restart



block

6. Reduce Miss Penalty: Critical Word First

- *Critical Word First*—Request the missed word first from memory and send it to the CPU as soon as it arrives; let the CPU continue execution while filling the rest of the words in the block
  - Long blocks more popular today $\Rightarrow$ Critical Word 1st Widely used



7. Merging Write Buffer to Reduce Miss Penalty

- Write buffer to allow processor to continue while waiting to write to memory
- If buffer contains modified blocks, the addresses can be checked to see if address of new data matches the address of a valid write buffer entry
- If so, new data are combined with that entry

7. Merging Write Buffer to Reduce Miss Penalty (cont)

- Increases block size of write for write-through cache of writes to sequential words, bytes since multiword writes more efficient to memory

10 Advanced Cache Optimizations

- Reducing Miss Rate

8. Compiler optimizations