Memory Hierarchy

CS6133 - Computer Architecture I

Vikram Padman

Polytechnic Institute of New York University

vikram@poly.edu

Agenda

1. Introduction
2. Hierarchy
3. Cache
4. SDRAM
5. Activity

Simple CPU's Memory

- Let's consider the memories in Simple CPU
    1. How does instruction get into Instruction Memory (IM) ?
    2. How does data get in and out of Data Memory (DM) ?
    3. We said that IM and DM are L1 instruction and data cache in a modern CPU.
        1. What is a cache and what is its function?
        2. What is the size of the cache ?
        3. Why can't DDR3 RAM be IM and/or DM?

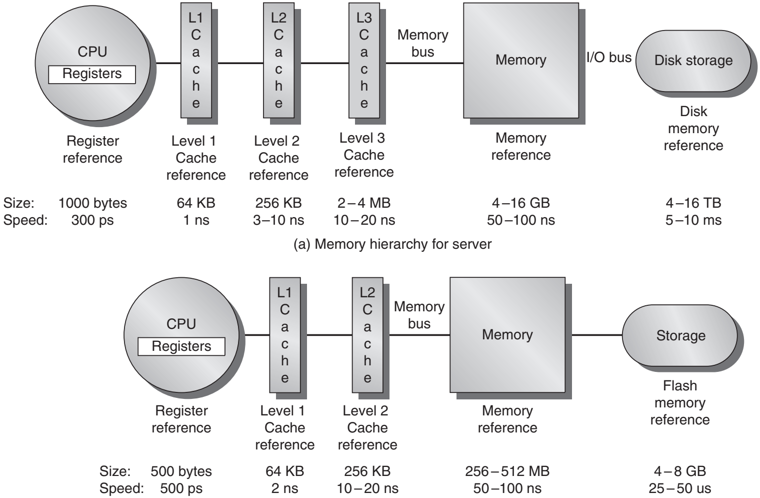
Memory-Cost

1. **CPU Registers** Flip Flops
   - 1 CPU clock cycle (100's ps), $2000-$5000/GB
2. **L1 Data/Instruction Cache** Static RAM
   - 10's CPU clock cycle (<1ns), $1000-$2000/GB
3. **L2 Cache** Static RAM
   - 25-50 CPU clock cycle (1-5 ns), $500-$1000/GB
4. **L3 Cache** Static RAM
   - 100's CPU clock cycle (< 10's ns), <$500/GB
5. **SDRAM** Dynamic RAM
   - 1,000's CPU clock cycle (50-100 ns), <$25/GB
6. **Magnetic / Flash Disk**
   - > 10,000's CPU clock cycle (5-20us), <$1/GB
7. **Google Disk**
   - > Million's CPU clock cycle (< 1s), <$.10/GB

Memory - Goals

- What are the goals of a memory system?
  1. Large
     - Less constrains on programs, more users, solve larger problems .. etc
  2. Fast
     - Run as fast as the CPU
  3. Cheap
     - User should be able to afford it
  4. Low Power
     - Can't afford a reactor!
- Solution: Memory Hierarchy
  - Use slowest/cheapest memory to support the entire address space
  - Use progressively smaller but faster memories each containing a subset of memory below it.

Memory Hierarchy



(a) Memory hierarchy for server

(b) Memory hierarchy for a personal mobile device

Memory Hierarchy

Theory

- **Principle of Locality**: A small subset of memory is accessed more frequently than others at any given moment.
    1. **Spatial Locality**: Memory locations near the current position have a high probability for being accessed soon
    2. **Temporal Locality**: Memory locations that were accessed recently have a high probability for being accessed again
- Note that memory hierarchy would have a much higher negative impact on access time if we always access memory in a random order.

Memory Hierarchy

Advantages

1. Store everything in hard drive or in some slow secondary storage
2. Load all data and instructions required to run a particular program into main memory (DDR3, DDR2 ..etc)
3. Copy recently accessed and nearby memory locations from main memory to smaller, but much faster SRAM cache
4. Provide data and instruction from the cache to the processor when it is needed

Memory Hierarchy

Terminology

- **Cache Line or Block**: Smallest unit of memory handled by cache, usually 32, 64 or 128 bytes in length.
- **Cache Hit**: Memory request by the CPU could be fulfilled by cache
- **Hit Rate**: Cache hits per unit of time, usually in seconds.
- **Cache Miss**: Memory request by the CPU could not be fulfilled by cache
- **Miss Rate**: Cache misses per unit of time, usually in seconds.

Cache Memory

How Does Cache Work?

Mem. Add.	
00000	🟥
00001	🟩
00010	🟦
00011	🟨
00100	🟥
00101	🟩
00110	🟦
00111	🟨
01000	🟥
01001	🟩
01010	🟦
01011	🟨
01100	🟥
01101	🟩
01110	🟦
01111	🟨
10000	🟥
10001	🟩
10010	🟦
10011	🟨
10100	🟥
10101	🟩
10110	🟦
10111	🟨
11000	🟥
11001	🟩
11010	🟦
11011	🟨
11100	🟥
11101	🟩
11110	🟦
11111	🟨

Cache

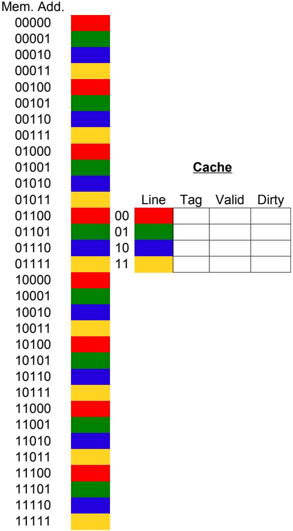
	Line	Tag	Valid	Dirty
00	🟥			
01	🟩			
10	🟦			
11	🟨			

Let's consider a **direct mapped** cache

- Every address in main memory is mapped to a specific cache line or block
- In **direct mapped** cache, an item in main memory could only go to one location in cache
- In this example, cache location = (Mem. Add.) mod 4

Cache Memory

How Does Cache Work?



Mem. Add.	
00000	🟥
00001	🟩
00010	🟦
00011	🟨
00100	🟥
00101	🟩
00110	🟦
00111	🟨
01000	🟥
01001	🟩
01010	🟦
01011	🟨
01100	🟥
01101	🟩
01110	🟦
01111	🟨
10000	🟥
10001	🟩
10010	🟦
10011	🟨
10100	🟥
10101	🟩
10110	🟦
10111	🟨
11000	🟥
11001	🟩
11010	🟦
11011	🟨
11100	🟥
11101	🟩
11110	🟦
11111	🟨

Cache

	Line	Tag	Valid	Dirty
00	🟥			
01	🟩			
10	🟦			
11	🟨			

- **Valid** bit indicates cache line's validity
  (1 = Valid, 0 = Not valid)
  Valid bit is set to 0 initially
- **Tag** Holds the high order bits of the memory address that is being cached.
  In this example, tag would hold 3 high order memory address bits
- **Dirty** bit is asserted when cache line is written
  (Used only in cache write back mode)

Cache Memory

Direct Mapped Cache Example

Mem. Add.	
00000	01F
00001	020
00010	021
00011	022
00100	023
00101	024
00110	025
00111	026
01000	027
01001	028
01010	029
01011	02A
01100	02B
01101	02C
01110	02D
01111	02E
10000	02F
10001	030
10010	031
10011	032
10100	033
10101	034
10110	035
10111	036
11000	037
11001	038
11010	039
11011	03A
11100	03B
11101	03C
11110	03D
11111	03E

CPU

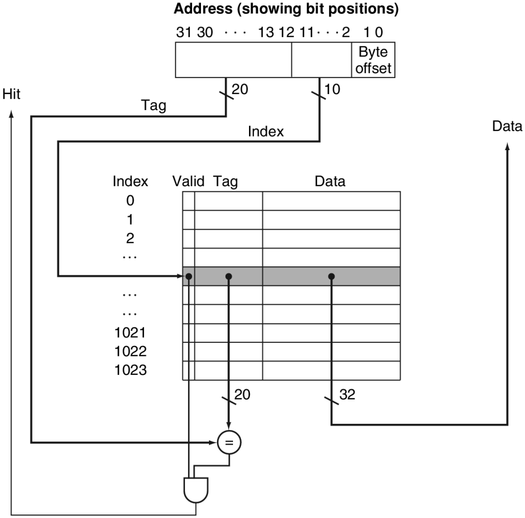
	Binary Add	Hit / Miss
1	00000	Miss
2	00001	Miss
3	00000	Hit
4	01011	Miss
5	11010	Miss

Cache

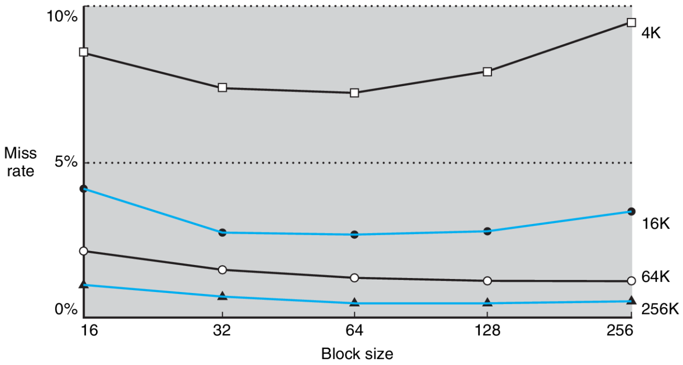
	Line	Tag	Valid	Dirty
00	01F	000	Y	N
01	020	000	Y	N
10	039	110	Y	N
11	02A	010	Y	N

Cache Memory

Direct Mapped - Address Subdivision



Block Size vs Cache Size



Copyright ©2009 Elsevier, Inc

- Miss rate depends on cache size and block size
  - Larger block size does reduce miss rate only when the cache size is much larger than the block
  - Large block ⇒ fewer blocks ⇒ more competition ⇒ higher miss rate

Handling Cache Miss

- CPU proceeds normally on a cache hit
- On a cache miss the CPU:
    - Stalls the pipeline
    - Fetches block from lower (slower) level memory, next in the hierarchy, to cache
    - L1 Instruction miss: Restart instruction fetch
    - L1 Data miss: Complete data access

Data Writes

- **Write-Through**: Write to cache and to main memory directly
    - Writes are slow, but memory and cache would be consistent
- **Write-Back**: Only write to cache and update memory when evicted
    - Writes are fast, but memory and cache contents would be inconsistent
    - Cache and memory inconsistency could be an issue in multi processor system

Cache Performance

- CPU_{tot} should includes time spent on cache hit and miss time
- $MEM_{stalls} = MEM_{acc}/Program * Miss_{rate} * Miss_{penalty}$
- $\Rightarrow Instructions/Program * Misses/Instruction * Miss_{penalty}$
- For Example:
  - L1 I-Cache miss rate = 2%, L1 D-Cache miss rate = 4%
  - $Miss_{penalty} = 100$ Cycles
  - $CPI_{base} = 2$
  - Load & store are 36% of instructions
  - Cache Misses = I-Cache: $.02 * 100 = 2$, D-Cache: $.36 * .04 * 100 = 1.44$
  - $CPI_{actual} = 2 + 2 + 1.44 - 5.44 \Rightarrow 2.72$ slower than CPI_{base}

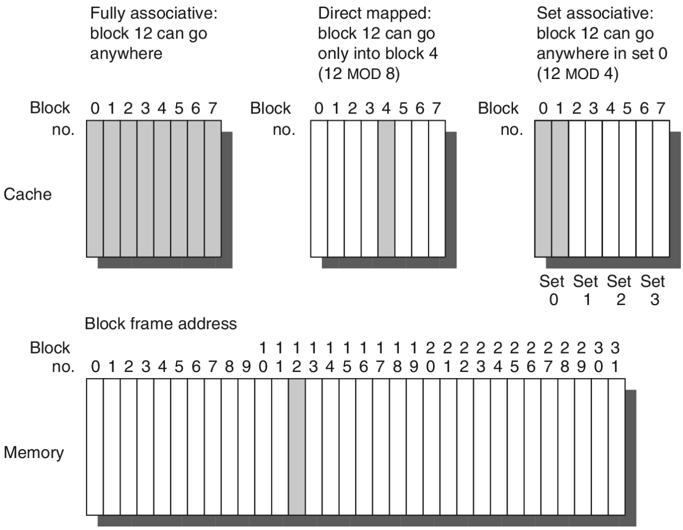
Cache Performance

- Hit time is also important for performance
- Average memory access time $(AMAT)$
  - $AMAT = HitTime + Miss_{rate} * Miss_{penalty}$
- Example
  - CPU with 1ns clock, hit time = 1 cycle, miss penalty = 20 cycles, I-cache miss rate = 5%
  - $AMAT = 1 + .05 * 20 = 2ns$

Associative Caches

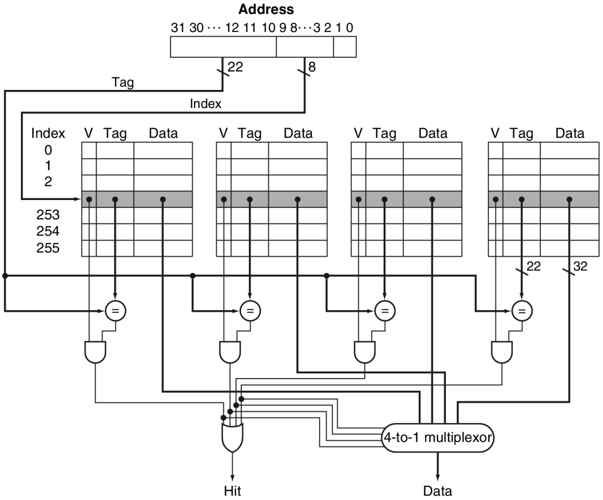
- Fully associative cache
  - Allow any memory block to any cache line
  - All cache line entries has to be searched on access
- *n*-way set associative
  - Each set contains *n* entries
  - Block number determines which set
    - (Block Num) MOD (Number of Sets)
  - All cache line entries within a given set should be searched on access

Associative Caches



Copyright ©2009 Elsevier, Inc

Associative Caches

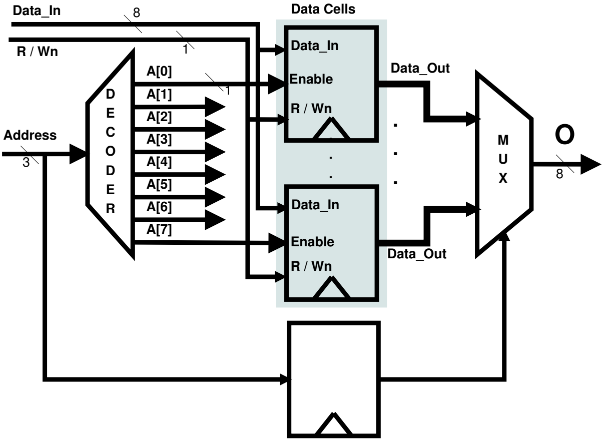


Copyright ©2009 Elsevier, Inc

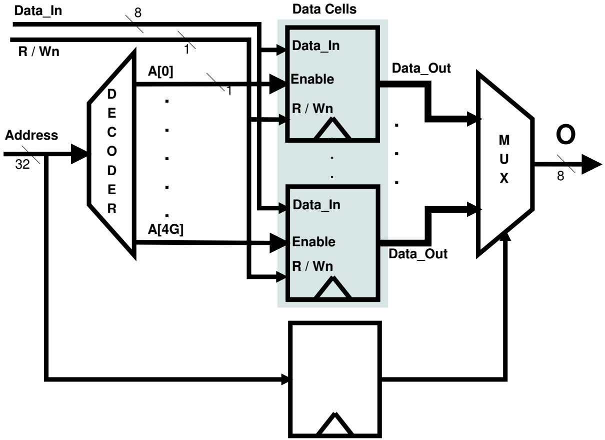
Replacement Policy

- **Direct Mapper**: Don't have much choice
- **Set Associative**:
  - **LRU** Least-Recently Used
    - Choose the cache line that is not used for the longest time.
  - **Random**
    - Randomly select a cache line for eviction.
    - Does provide comparable performance as LRU for high associative cache

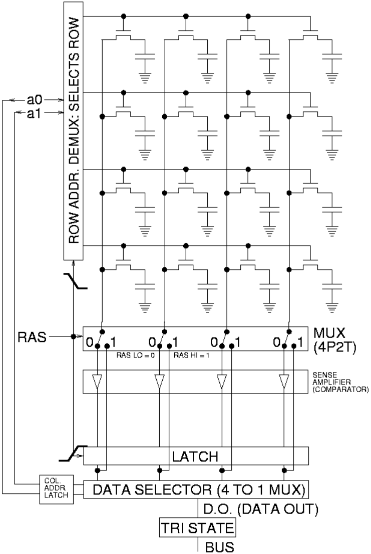
SDRAM – Very Simple Memory(SRAM)



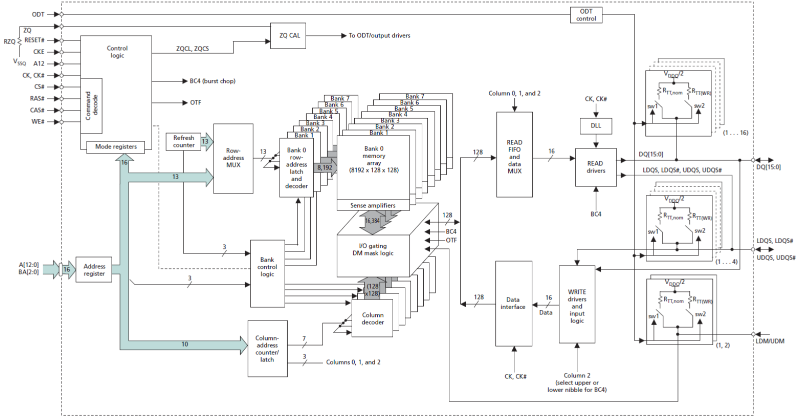
SDRAM – 4GB SRAM



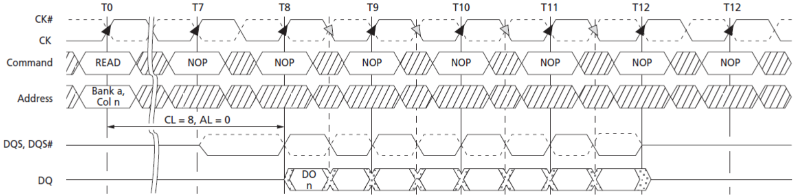
DRAM –Simple


ROW ADDR. DEMUX: SELECTS ROW
a0
a1
RAS
0 1
0 1
0 1
0 1
RAS LO = 0
RAS HI = 1
MUX
(4P2T)
SENSE
AMPLIFIER
(COMPARATOR)
LATCH
COL.
ADDR.
LATCH
DATA SELECTOR (4 TO 1 MUX)
D.O. (DATA OUT)
TRI STATE
BUS

SDRAM – 1Gb DDR3 SDRAM Component



SDRAM – Read Timing



Week-12: Activity 1

Read Appendix B in your text book and answer the following questions:

1. What is the difference between “n-way set associative” and “fully associative cache”? Describe the advantages and disadvantages.
2. How is virtual memory managed?
3. How does out-of-order execution effect cache memories performance?
4. Is it possible to calculate optimal cache and page size for a given CPU architecture?
5. Would it help application developers who develop high performance applications if they understand memory architecture? if yes, how?

Week-12: Activity 2

DDR4 SDRAM: Research about the upcoming DDR4 memory standard and answer the following questions:

1. Could DDR4 memory eliminate the need for cache memory?
2. What is the difference between DDR3 and DDR4 memory?

Week-12: Activity 3

Consider Direct mapped and 2-way set associative cache architectures with 4K cache size, 16 byte blocks and 16 memory address bits.

1. What is the tag size and index size for each architecture?
2. Which architecture, in principle, would have a high hit rate? Provide detailed proof with access patterns to support your answer.
3. Assuming LRU replacement policy, show a high miss rate in 2-way set associative cache compared to direct mapped cache. Provide detailed proof with access patterns to support your answer.