CS / EE 320
Computer Organization and Assembly Language
Spring 2025
Lecture 23

Shahid Masud

Topics: Address Decoding Examples, Cache and Memory Hierarchy, Direct Mapped Cache, Some Examples

Topics

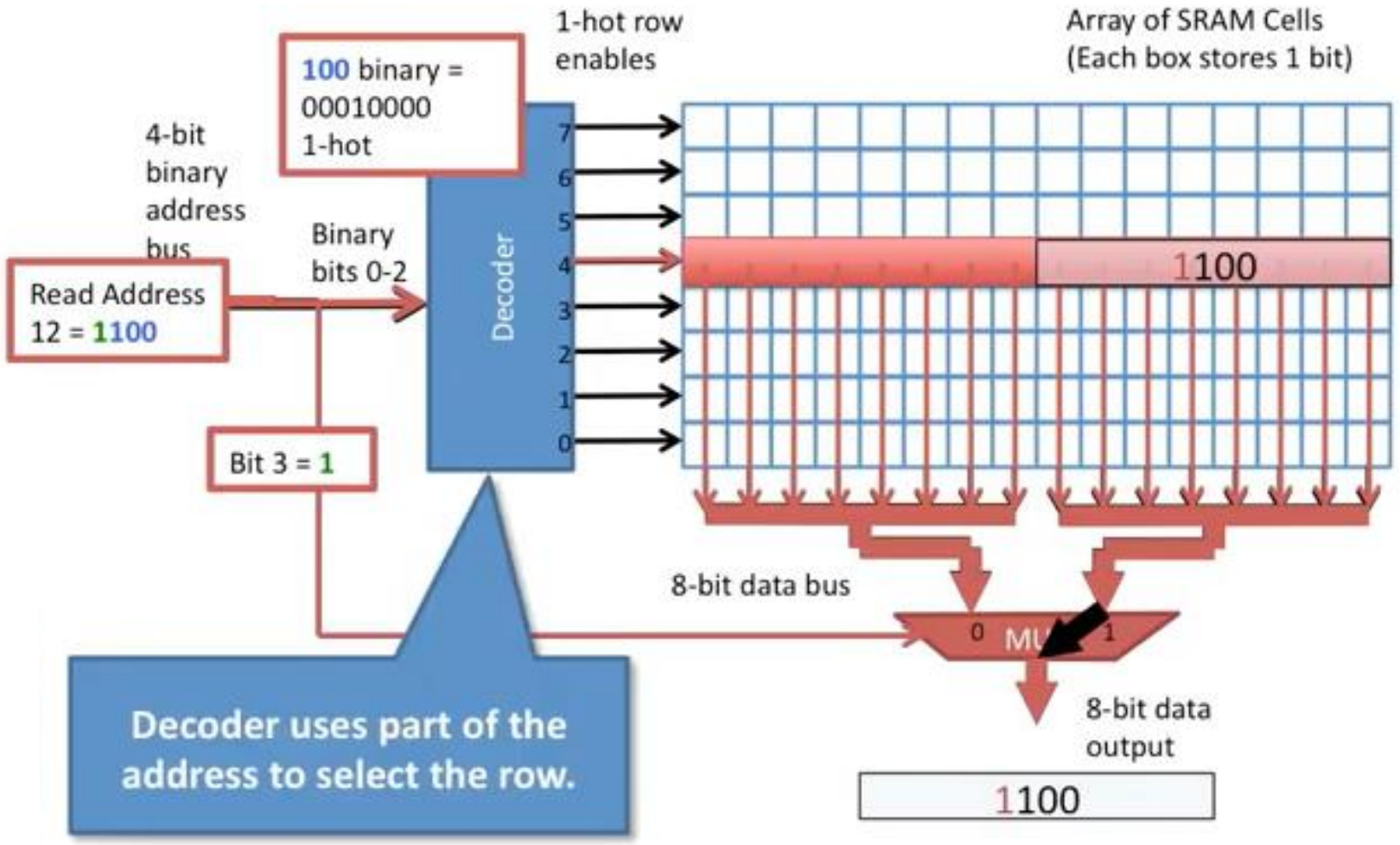
- Examples of Memory Address Decoding
- Caches and Memory Hierarchy
- Direct Mapped Caches
- Example of Direct Mapped Caches

Quiz in Next Week

Memory Organization and Address Decoding Examples

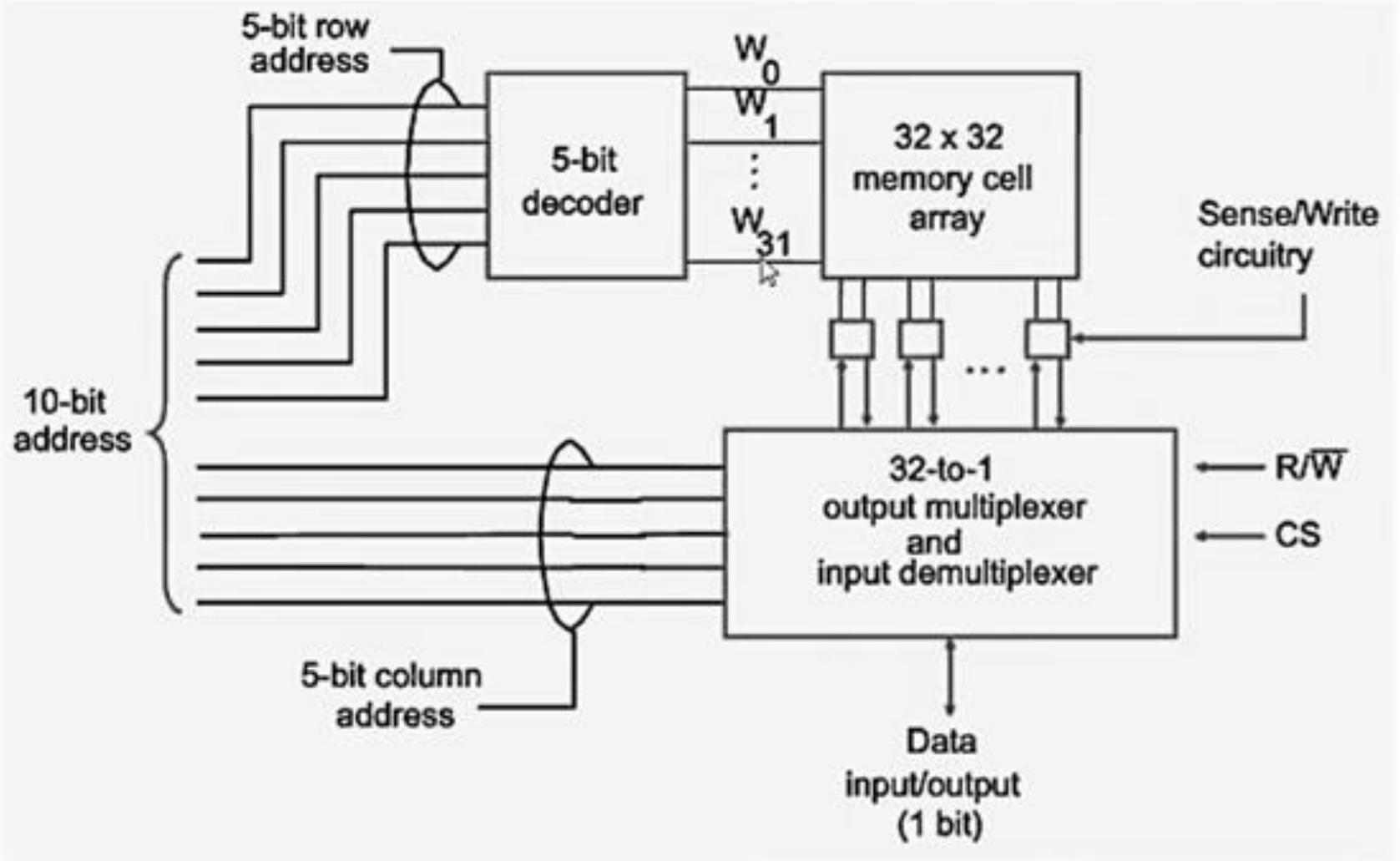
Typical Memory Organization

Question?
Is this
2D Memory organization
OR
2.5D Memory Organization



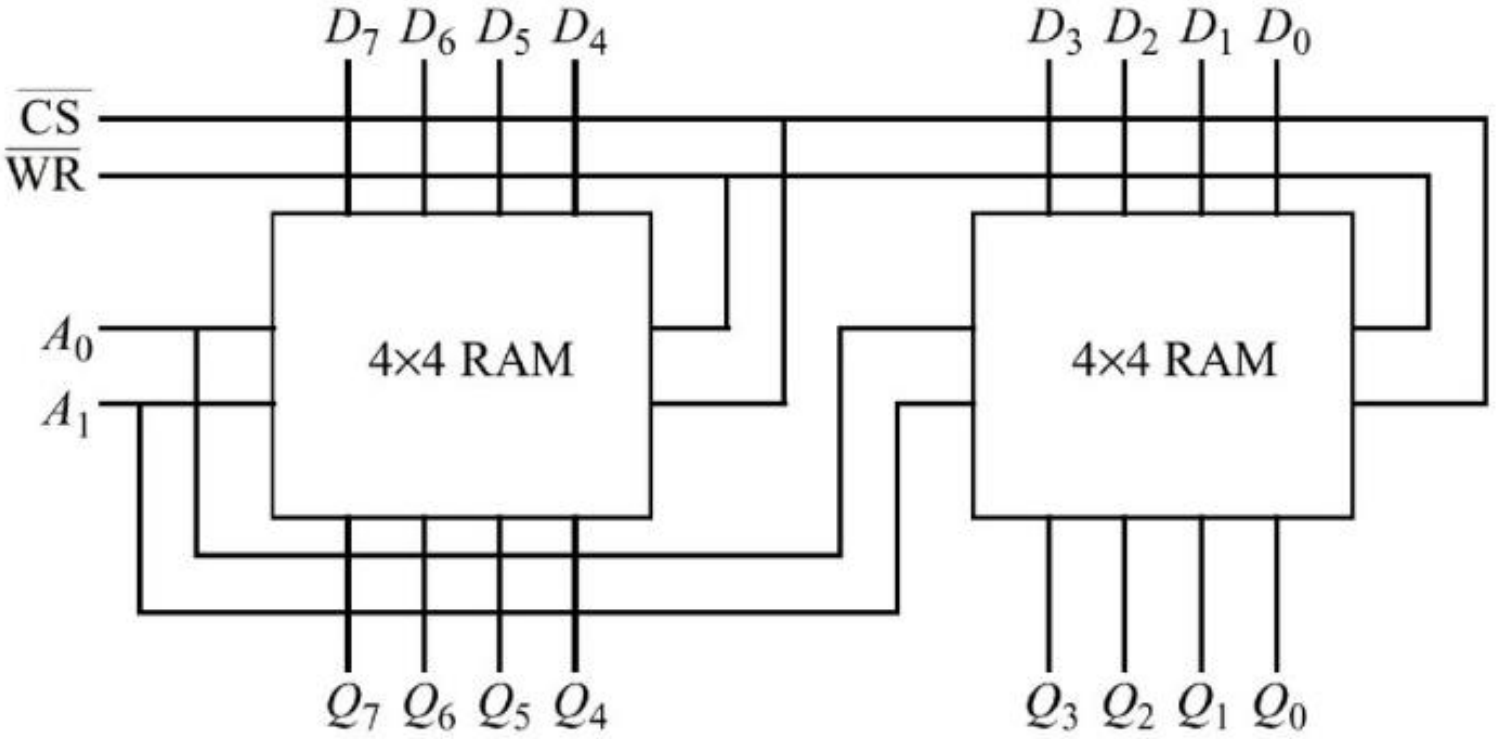
Organization of a 1Kx1 Memory Chip

2.5 D Memory Organization of a 1K x 1 Chip made from 32 x 32 Memory Cell Array

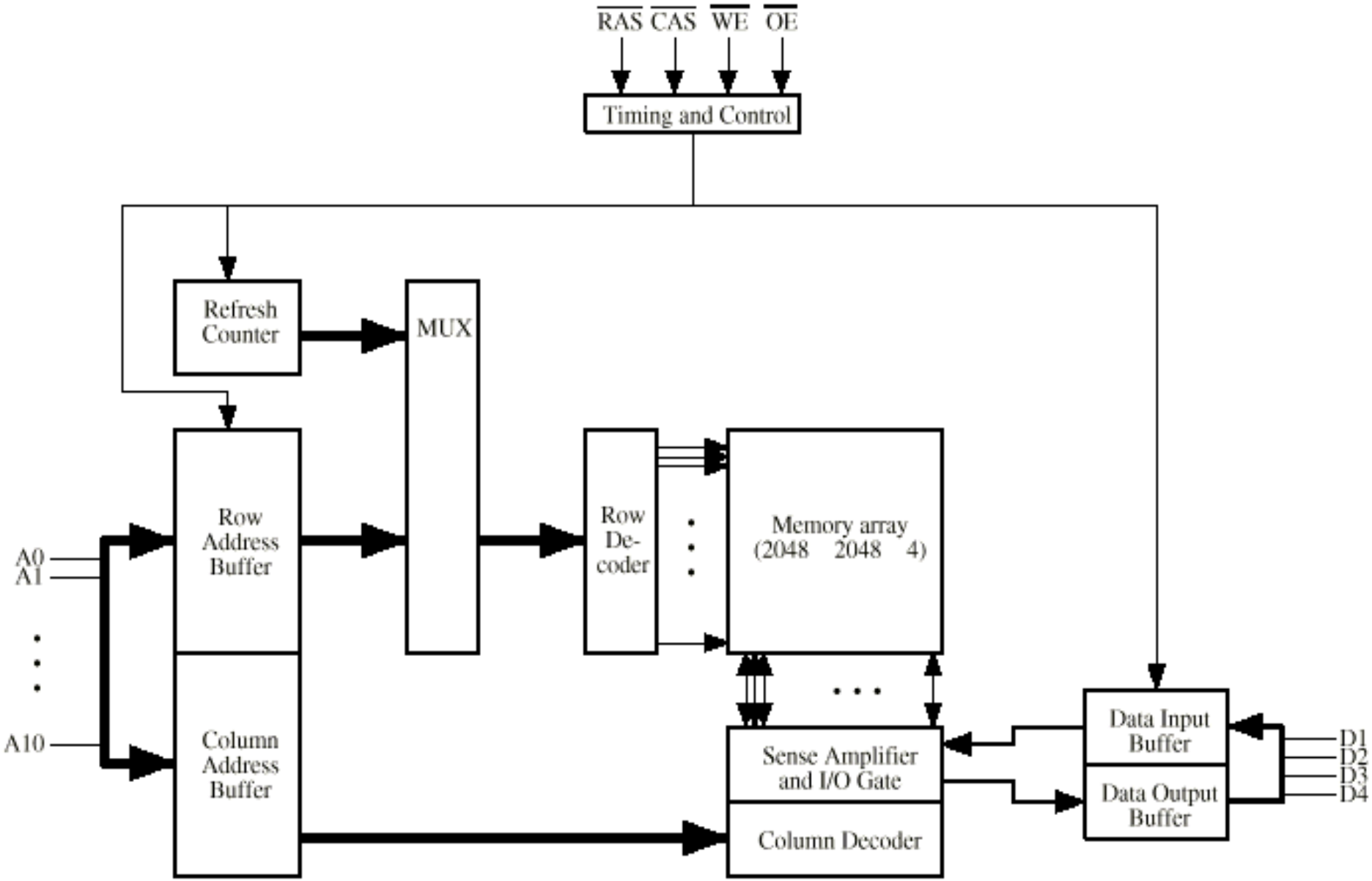


Example 2

Two Four-Word by Four-Bit RAMs are Used in Creating a Four-Word by Eight-Bit RAM



Typical 16 Mb DRAM (4M x 4)



Example 1 to Solve – Memory Organization

Design a 512 Bytes RAM Memory using blocks of size 128 x 8 bits

A. No of blocks needed?

B. Address Bits needed?

C. Size of Address Decoder?

D. Draw the Organization

Example 2 to Solve – Memory Organization

Design a 16 Kilo Bytes RAM memory using blocks of size 128 x 1 bit

A. No of blocks needed?

B. Address Bits needed?

C. Size of Address Decoder?

D. Draw the Organization

Locality of Reference in Memory Access

Locality of Reference

- During the course of the execution of a program, memory references tend to cluster
- e.g. loops

Principle of Locality

- Programs access a small proportion of their address space at any time
- **Temporal locality**
  - Items accessed recently are likely to be accessed again soon
  - e.g., instructions in a loop, induction variables
- **Spatial locality**
  - Items near those accessed recently are likely to be accessed soon
  - E.g., sequential instruction access, array data

Taking Advantage of Locality

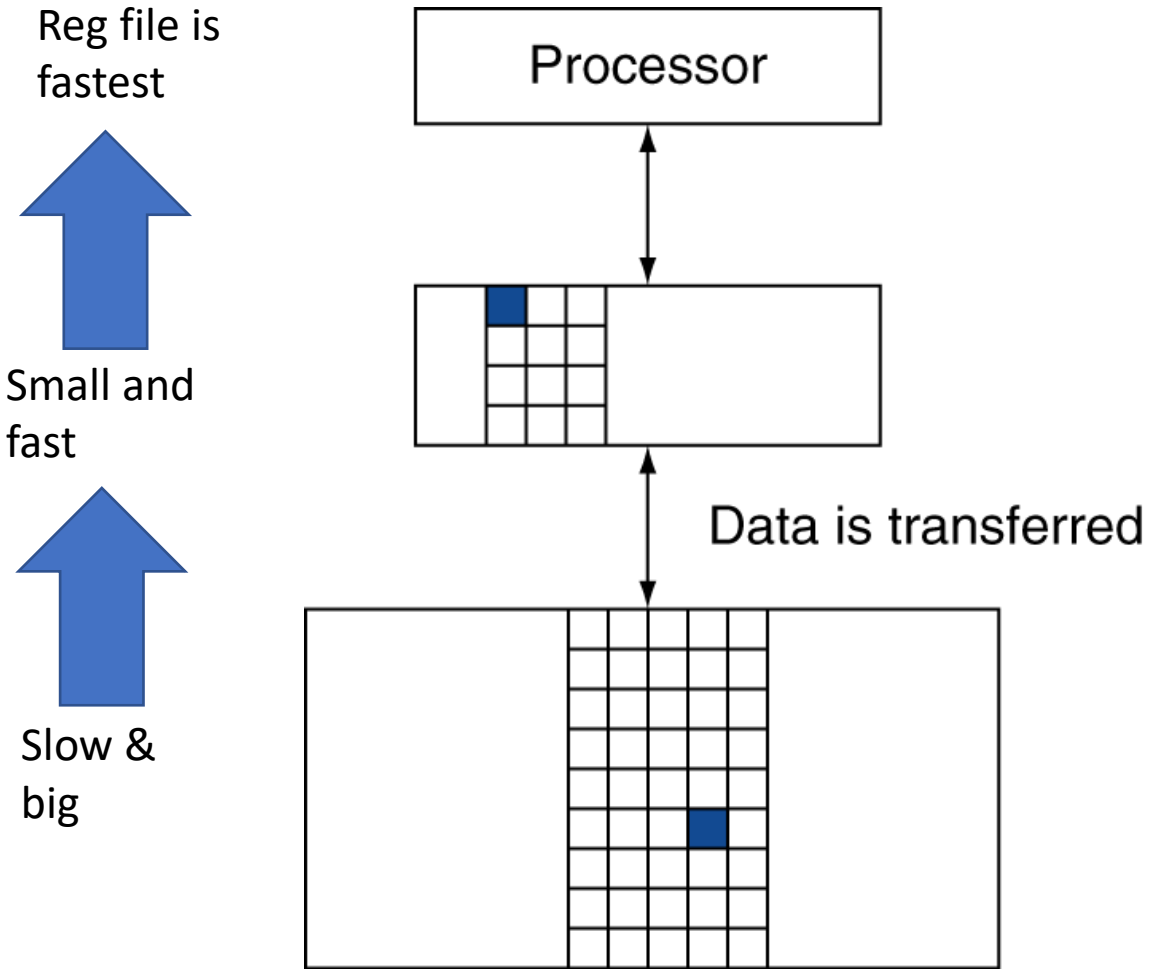
- Distribute total Memory in hierarchy
- Store everything on disk
- Copy recently accessed (and nearby) items from disk to smaller DRAM memory
  - Main memory
- Copy more recently accessed (and nearby) items from DRAM to smaller SRAM memory
  - Cache memory attached to CPU

The Memory Hierarchy

The BIG Picture

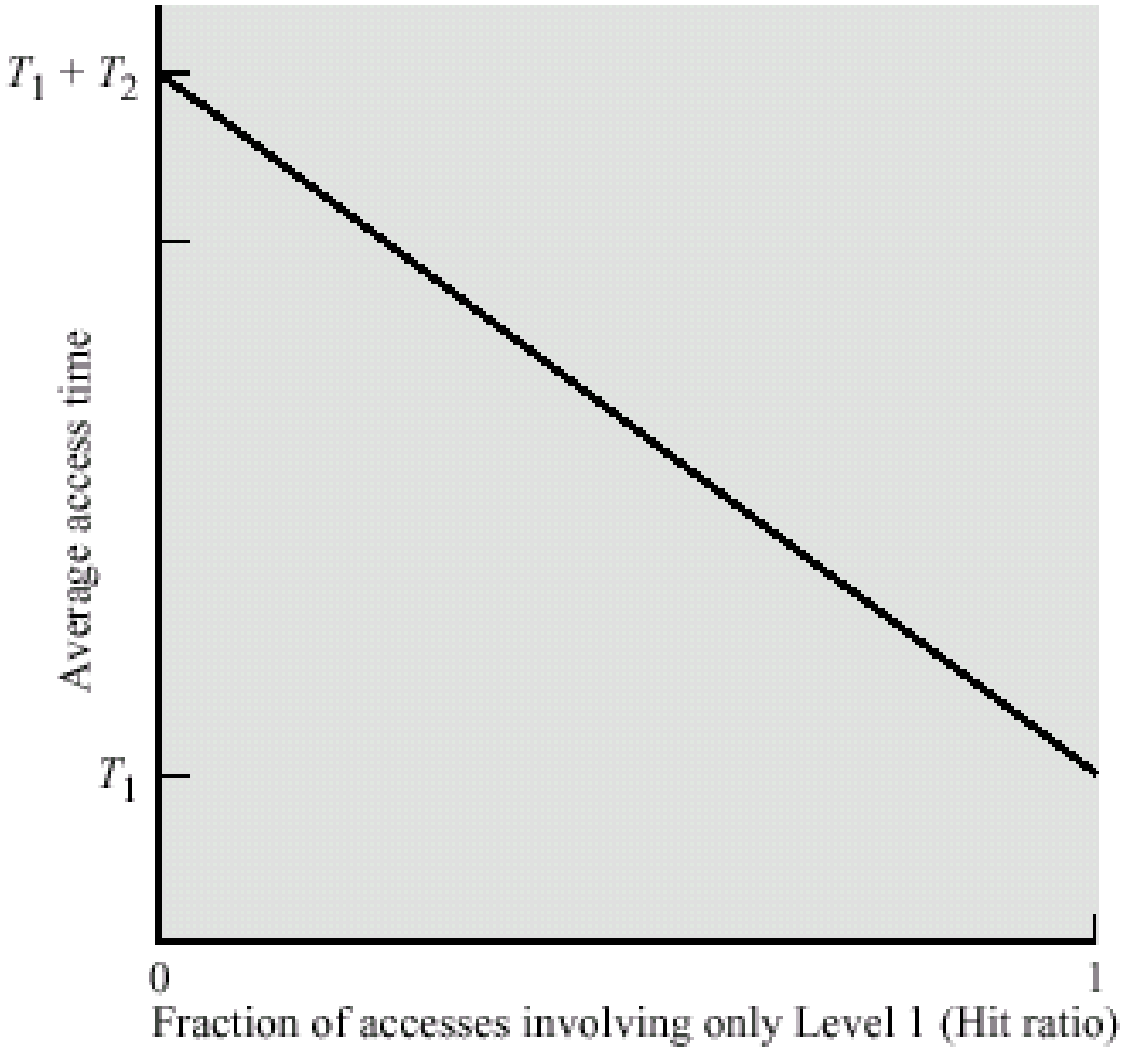
- Common principles apply at all levels of the memory hierarchy
  - Based on notions of caching
- At each level in the hierarchy
  - Block placement
  - Finding a block
  - Replacement on a miss
  - Write policy

Memory Hierarchy Levels



- **Block (or line): one unit of copying**
  - May be multiple words
- If accessed data is present in upper level
  - Hit: access satisfied by upper level
    - Hit ratio: hits/accesses
- If accessed data is absent
  - Miss: block copied from lower level
    - Time taken: miss penalty
    - Miss ratio: misses/accesses
      = 1 – hit ratio
  - Then accessed data supplied from upper level

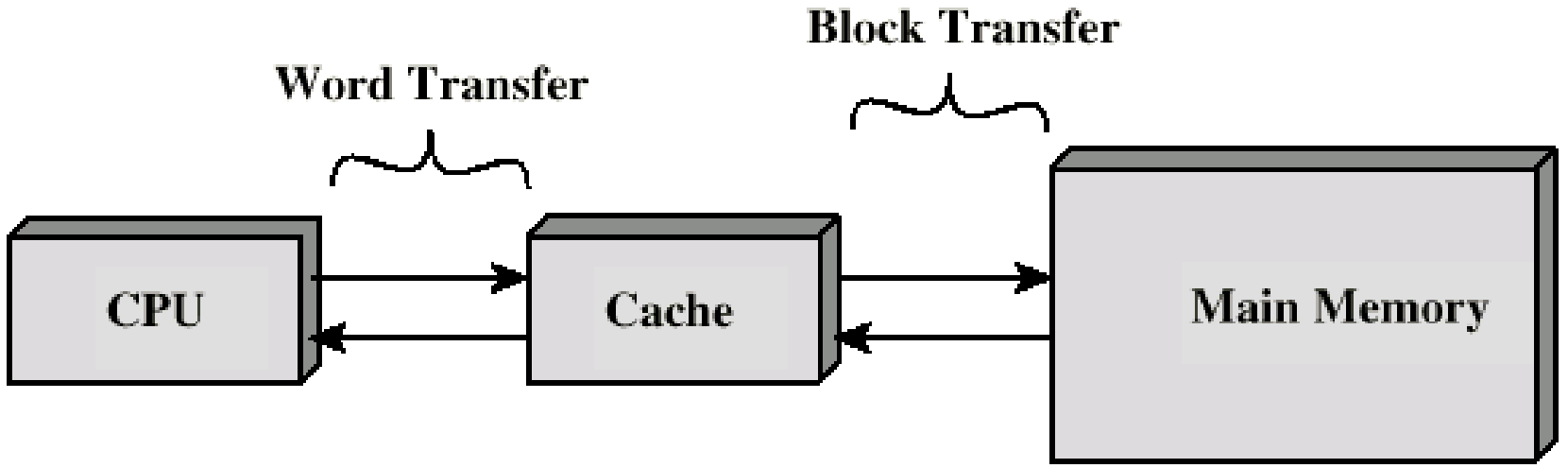
Average Access Time vs Hit Ratio



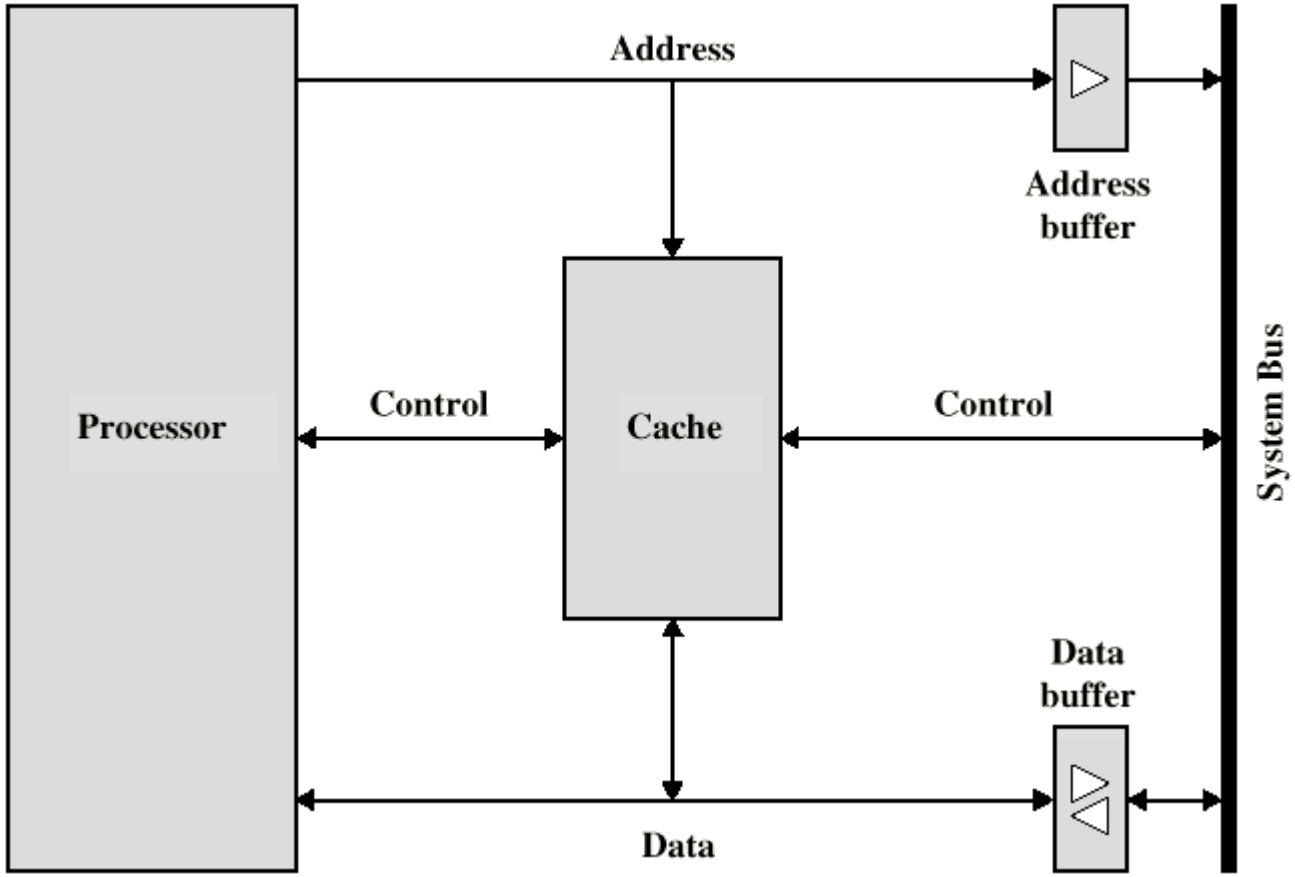
Cache Memory

Cache and Memory

- Small amount of fast memory
- Sits between normal main memory and CPU
- May be located on CPU chip or module

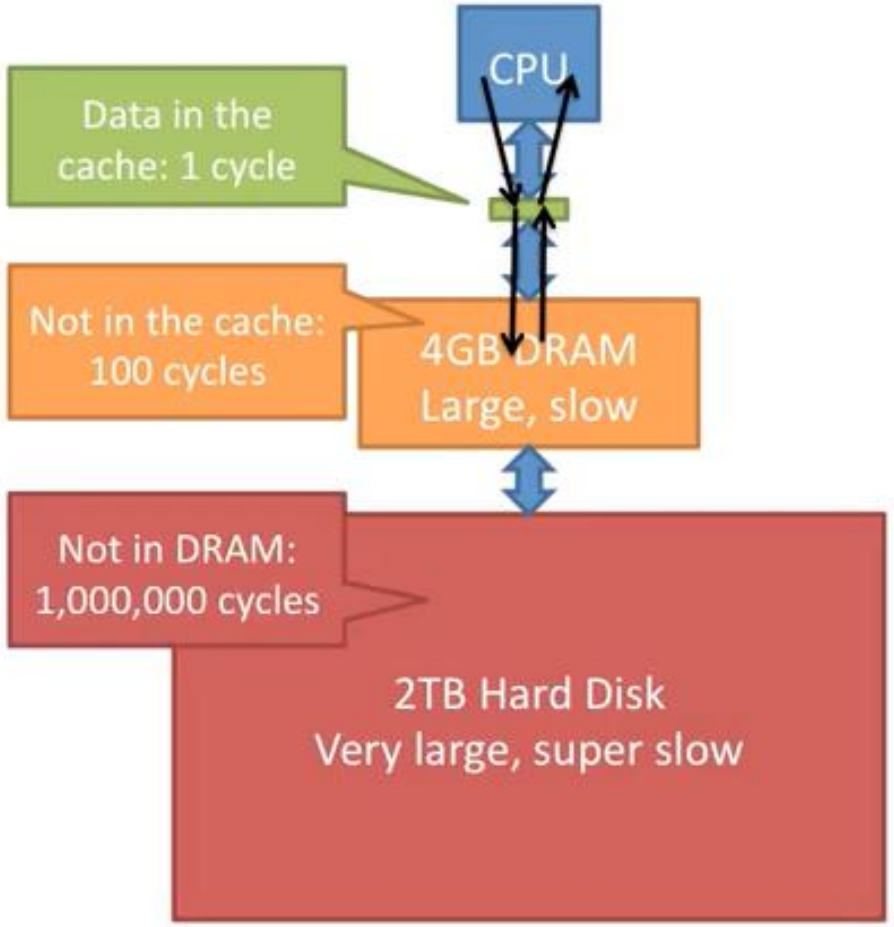


Typical Cache Organization



Idea of Cache Memory

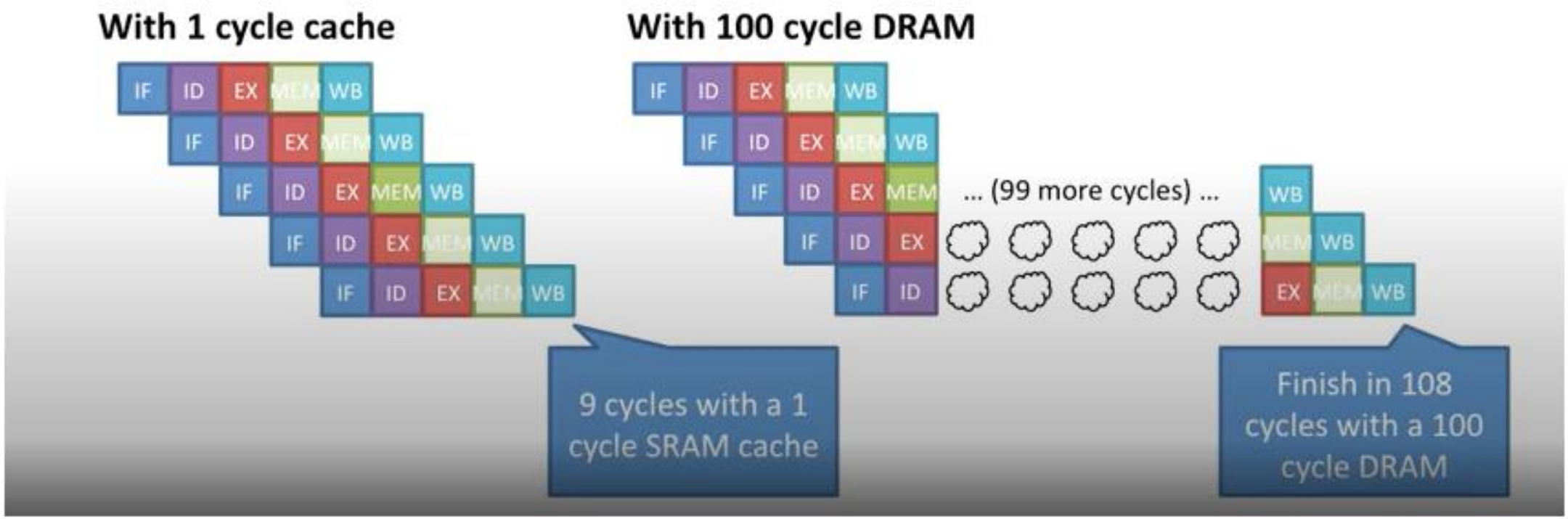
- Put the important data in a small fast memory (cache)
- If we access (load/store) that important data, we can do it quickly
- If we access (load/store) other data, we move it into the cache



Performance of Caches and Memory

- Accessing data in DRAM takes 100 cycles
- Accessing data in Cache (SRAM) takes 1 cycle
- 33% of instructions are load/stores

Ignore loading instructions for now (we'll add a second memory for this later)



Cache operation - Overview

- CPU requests contents of memory location
- Check cache for this data
- If present, get from cache (fast)
- If not present, read required block from main memory to cache
- Then deliver from cache to CPU
- Cache includes tags to identify which block of main memory is in each cache slot

Cache Design Parameters

- Size
- Mapping Function
- Replacement Algorithm
- Write Policy
- Block Size
- Number of Caches

Cache Size vs Speed

- Cost
  - More cache is expensive
- Speed
  - More cache is faster (up to a point)
  - Checking cache for data takes time

Cache Capacity Limits

- What do we do when we run out of space?

- **Replacement policy**
  - Need to choose some data in the cache to remove (evict)
  - Want to choose data that isn't going to be used soon

- Approaches:
  - **Direct-mapped:** only one block we can evict (because each block can only go in one place)
  - **Set/Fully-associative:**
    - Choose a **random** block to evict
    - Choose the **least recently used** (LRU) block to evict

Direct Mapped Cache

Direct Mapping

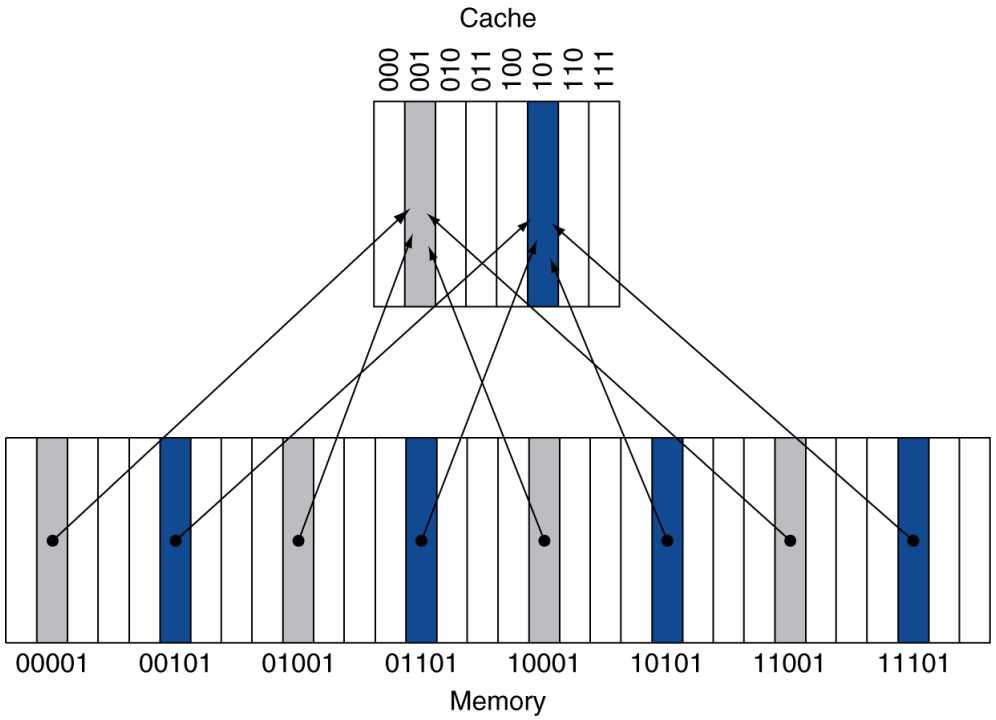
- Each block of main memory maps to only one cache line
  - i.e. if a block is in cache, it must be in one specific place
- Address is in two parts
- Least Significant w bits identify unique word
- Most Significant s bits specify one memory block
- The MSBs are split into a cache line field r and a tag of s-r (most significant)

Mapping Function

- Cache of 64kByte
- Cache block of 4 bytes
  - i.e. cache is 16k (2^{14}) lines of 4 bytes
- 16MBytes main memory
- 24 bit address
  - (2^{24}=16M)

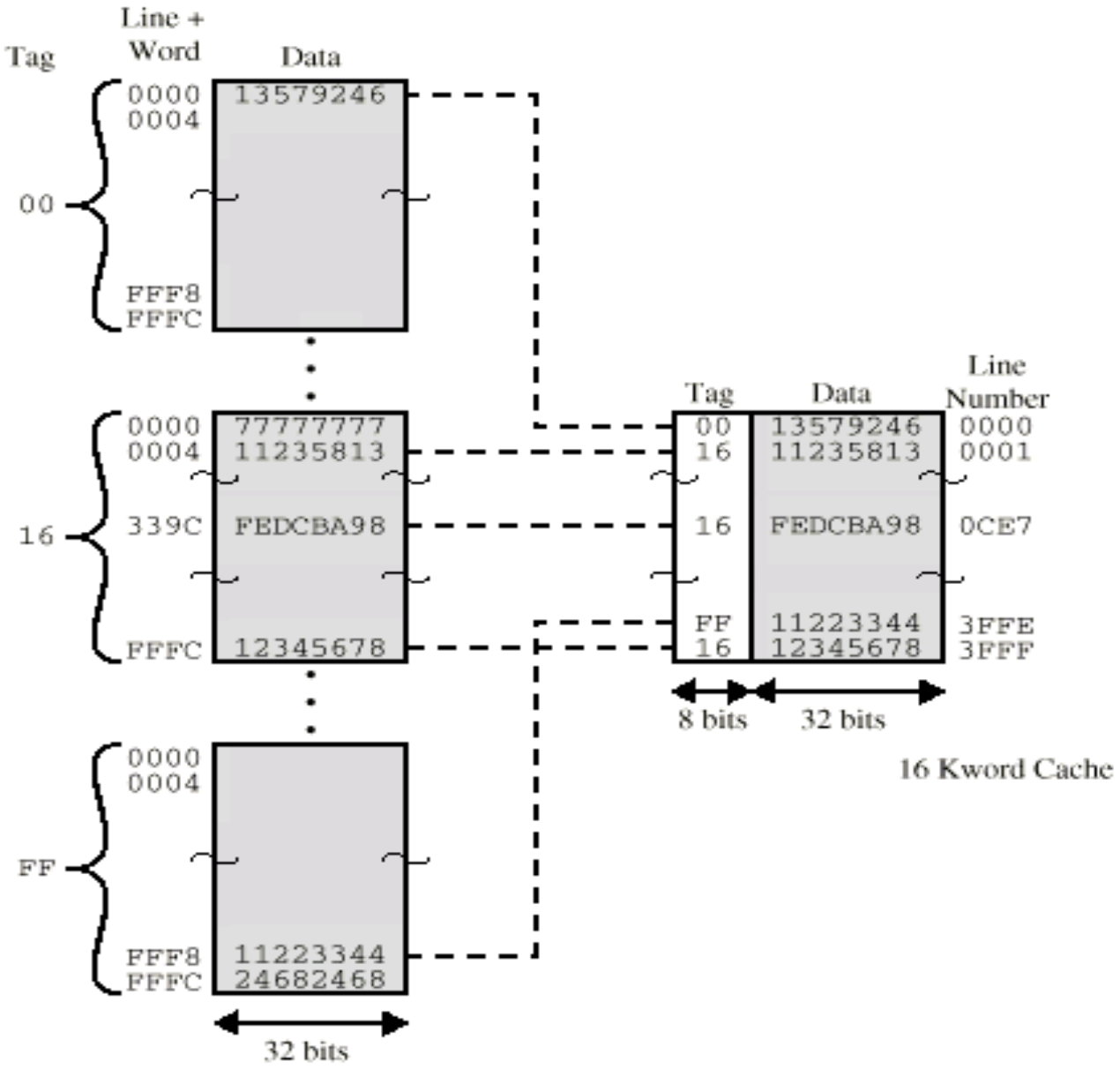
Direct Mapped Cache

- Location determined by address
- Direct mapped: only one choice
  - (Block address) modulo (No. of Blocks in cache)



- #Blocks is a power of 2
- Use low-order address bits

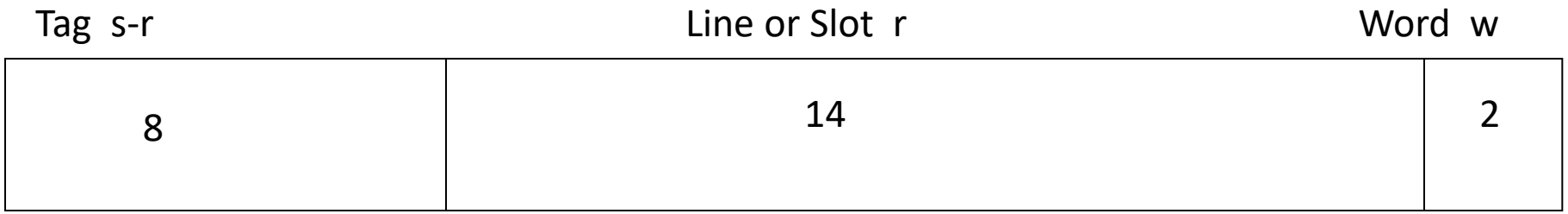
Direct Mapping Example



Direct Mapping Cache Line Table

Cache line	Main Memory blocks held
0	0, m, 2m, 3m…, 2^s-m
1	1,m+1, 2m+1…., 2^s-m+1
…….	
m-1	m-1, 2m-1, 3m-1, ….., 2^s-1

Direct Mapping Address Structure



- 24 bit address
- 2 bit word identifier (4 byte block)
- 22 bit block identifier
  - 8 bit tag (=22-14)
  - 14 bit slot or line
- No two blocks in the same line have the same Tag field
- Check contents of cache by finding line and checking Tag

Direct Mapping Cache Organization