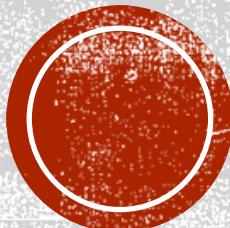


# **COMPUTER ORGANIZATION AND ARCHITECTURE**

**Course Code : CSE 2151**

**Credits : 04**



# MODULE 5: THE MEMORY SYSTEM

- Overview
  - Basic memory circuits
  - Organization of the main memory
  - Cache memory concept
  - Virtual memory mechanism
  - Secondary storage

# SOME BASIC CONCEPTS

## ▪ Traditional Architecture

- If MAR is  $k$  bits long and MDR is  $n$  bits long, then memory will contain up to  $2^k$  addressable locations and  $n$ -bits of data are transferred between the memory and processor.
- MFC: Memory Function Complete.

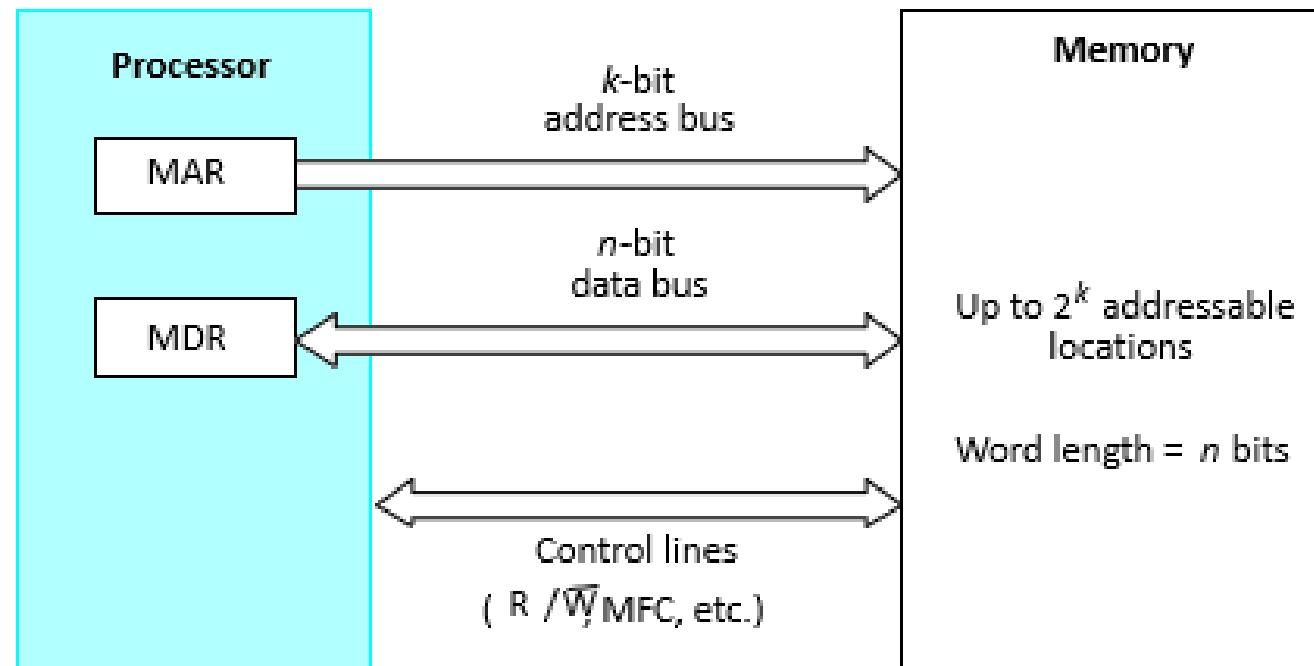


Figure 5.1. Connection of the memory to the processor.

# SOME BASIC CONCEPTS

- “Block transfer”:
  - bulk data transfer
- Memory access time:
  - time that elapses between the initiation of an operation to transfer a word of data and the completion of that operation.
- Memory cycle time:
  - the minimum time delay required between the initiation of two successive memory operations
- RAM:
  - any location can be accessed for a Read or Write operation in some fixed amount of time that is independent of the location’s address
- Cache memory
  - a small, fast memory inserted between the larger, slower main memory and the processor.
  - It holds the currently active portions of a program and their data.
- Virtual memory:
  - only the active portions of a program are stored in the main memory, and the remainder is stored on the much larger secondary storage device.
  - Sections of the program are transferred back and forth between the main memory and the secondary storage device in a manner that is transparent to the application program.
  - the application program sees a memory that is much larger than the computer’s physical main memory
  - memory management unit

# SEMICONDUCTOR RAM MEMORIES

- Internal Organization of Memory Chips

- 16 words of 8 bits each: 16x8 memory org.
- It has 16 external connections: addr. 4, data 8, control: 2, power/ground: 2
- 1K memory cells: 128x8 memory, external connections: ? 19(7+8+2+2)
- 1Kx1?:

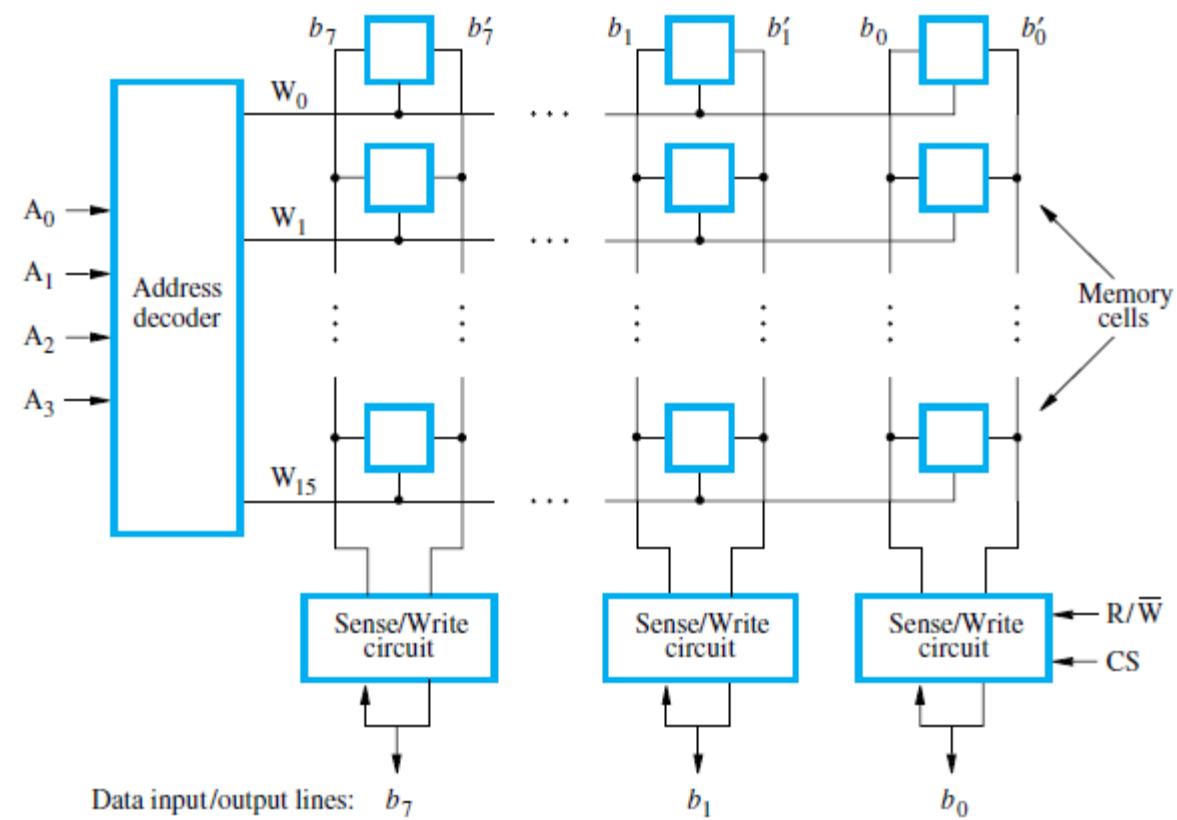
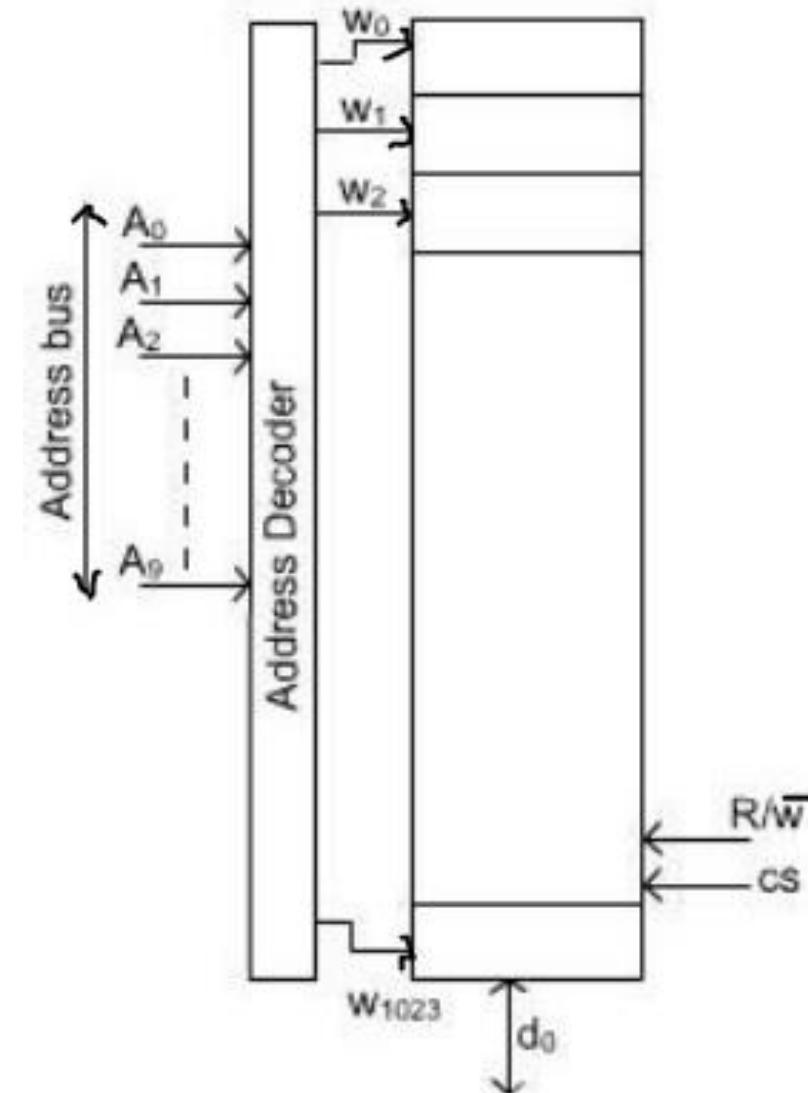


Figure 8.2 Organization of bit cells in a memory chip.

# SEMICONDUCTOR RAM MEMORIES

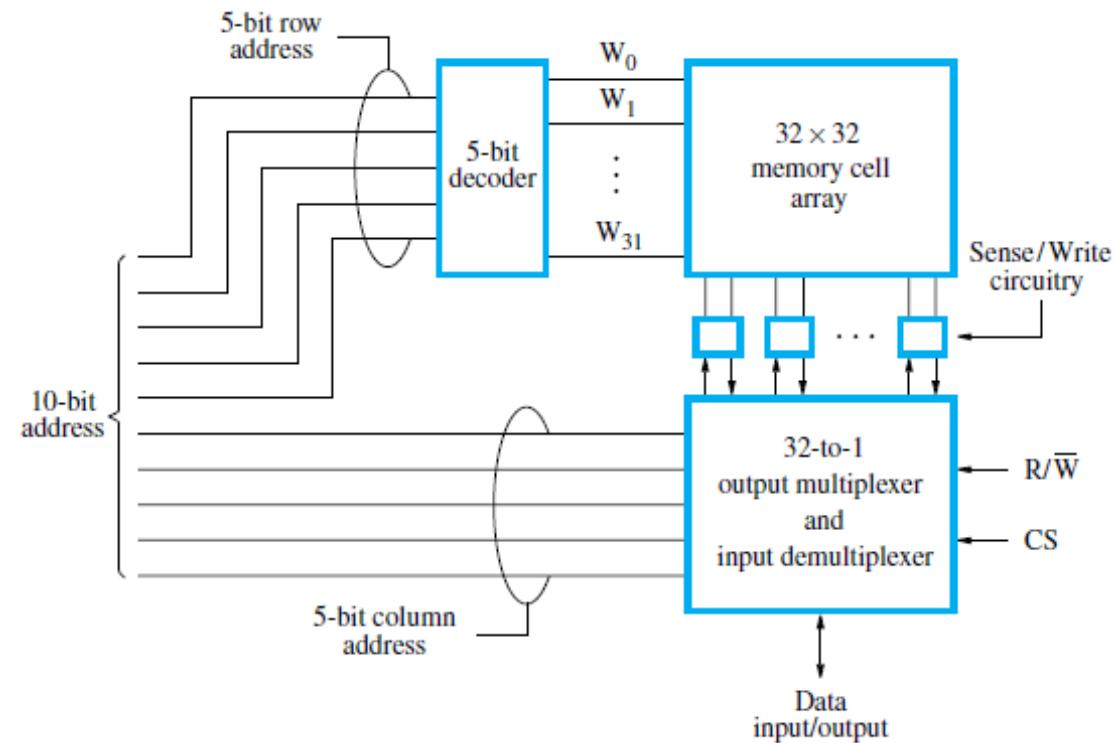
- Internal Organization of Memory Chips
  - 1Kx1: Meaning 1024 words X 1 bit organization.,.



# SEMICONDUCTOR RAM MEMORIES

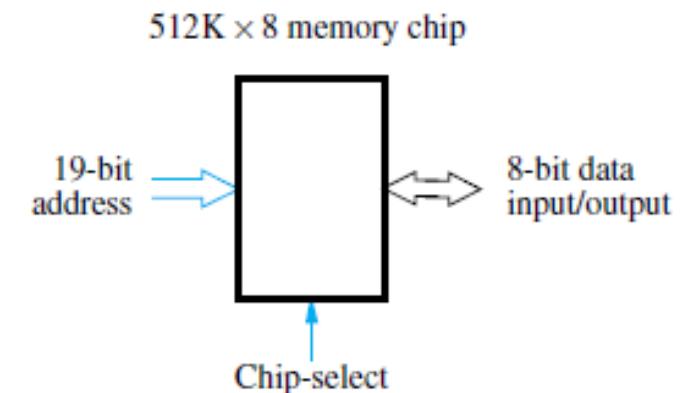
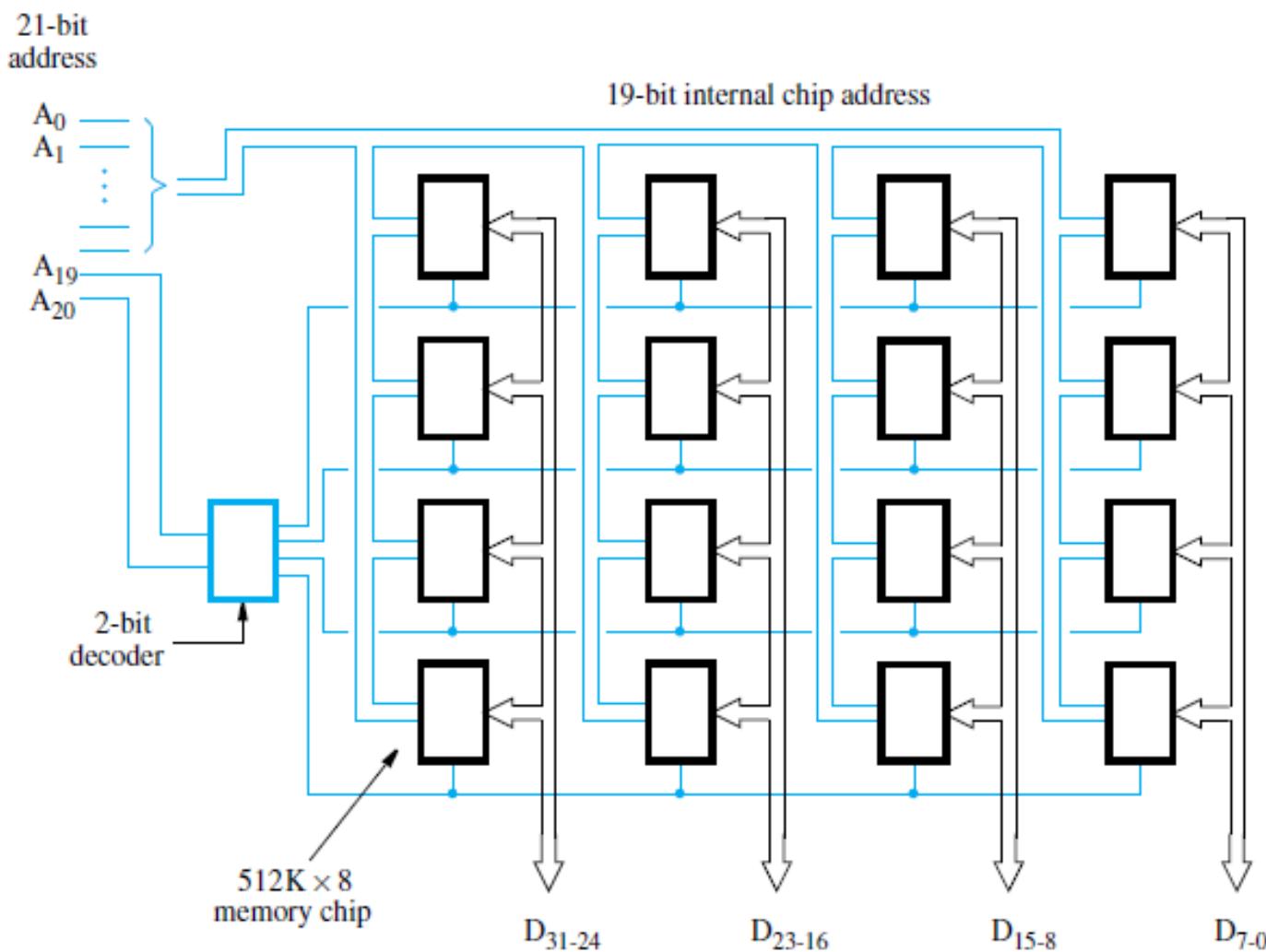
- **1Kx1:**

- If several memory words are organized in one row, then address bus is divided into two groups: one forms the row address, and the other forms the column address.
- The 10-bit address is divided into 5 bits each to form the row and column address of the cell array.
- A row address selects a row of 32 cells, all of which are accessed in parallel. However, according to the column address, only one of these cells is connected to the external data line via the input output multiplexers.
- 15 (10+1+2+2)



# STRUCTURE OF LARGER MEMORIES

- Organization of a  $2M \times 32$  memory module using  $512K \times 8$  static memory chips.



# MEMORY HIERARCHY: SPEED, SIZE, AND COST

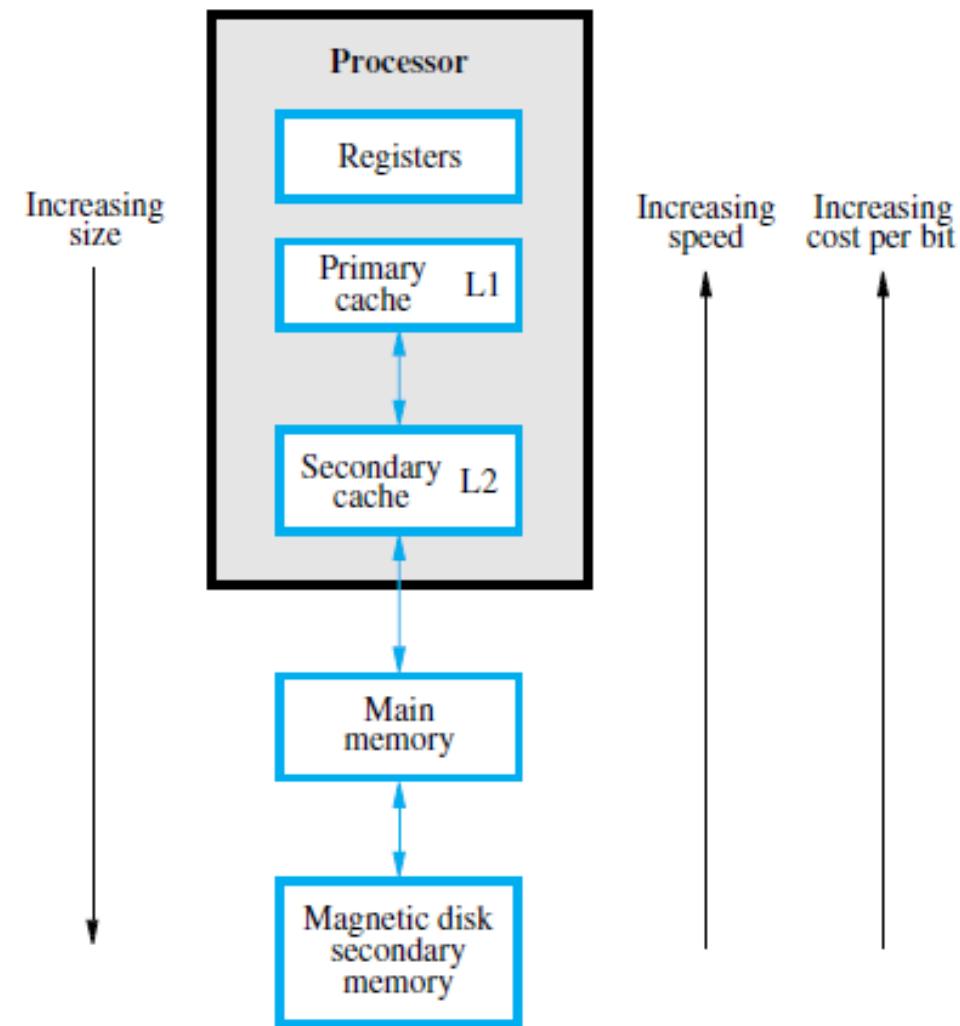


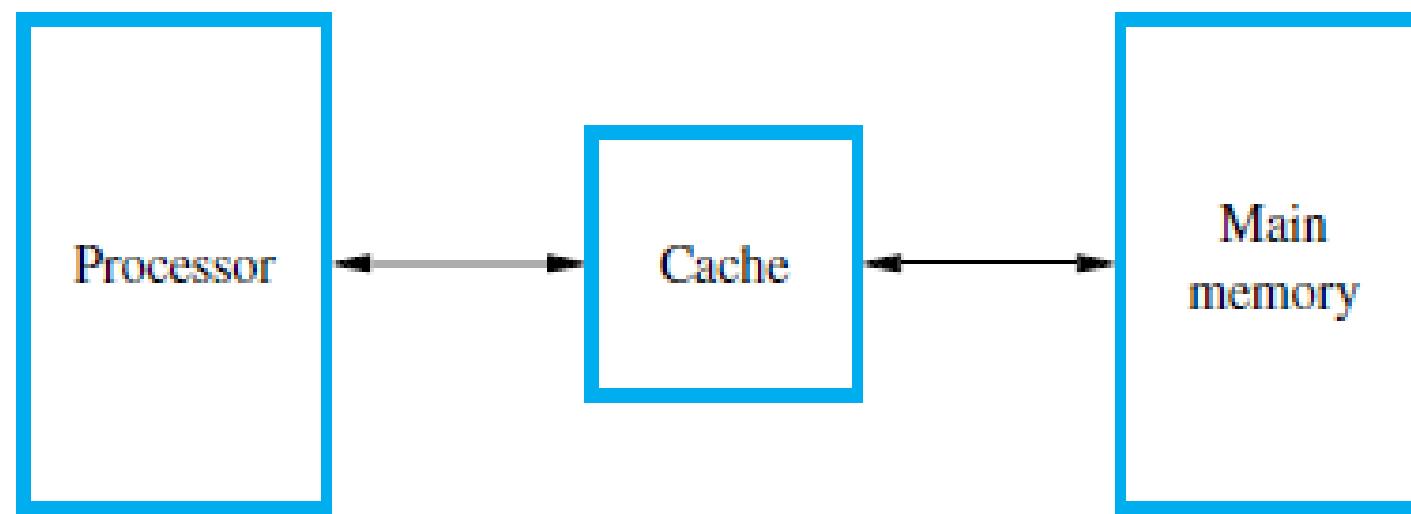
Figure 8.14 Memory hierarchy.

# CACHE MEMORIES

- What is cache?
- Why we need it?
- Locality of reference
  - temporal- a recently executed instruction is likely to be executed again very soon.
  - spatial- instructions close to a recently executed instruction are also likely to be executed soon
- Cache block – cache line
  - A set of contiguous address locations of some size

# CACHE MEMORIES

- Mapping function
- Replacement algorithm
- Hit / miss
- Write-through / Write-back
- Load through



**Figure 8.15** Use of a cache memory.

# CACHE MEMORIES

- The correspondence between the main memory blocks and those in the cache is specified by a mapping function.
- When the cache is full and a memory word (instruction or data) that is not in the cache is referenced, the cache control hardware must decide which block should be removed to create space for the new block that contains the referenced word.
- The collection of rules for making this decision constitutes the cache's replacement algorithm.

# CACHE HITS

- The cache control circuitry determines whether the requested word currently exists in the cache.
- If it does, the Read or Write operation is performed on the appropriate cache location. -a read or write hit.
  - main memory not involved when there is a cache hit in a Read operation
  - Write Operation –two ways
    - write-through protocol, both the cache location and the main memory location are updated.
    - write-back, or copy-back, protocol-update only the cache location and to mark the block containing it with an associated flag bit, often called the dirty or modified bit. The main memory location of the word is updated later

# WRITE-THROUGH V/S WRITE-BACK

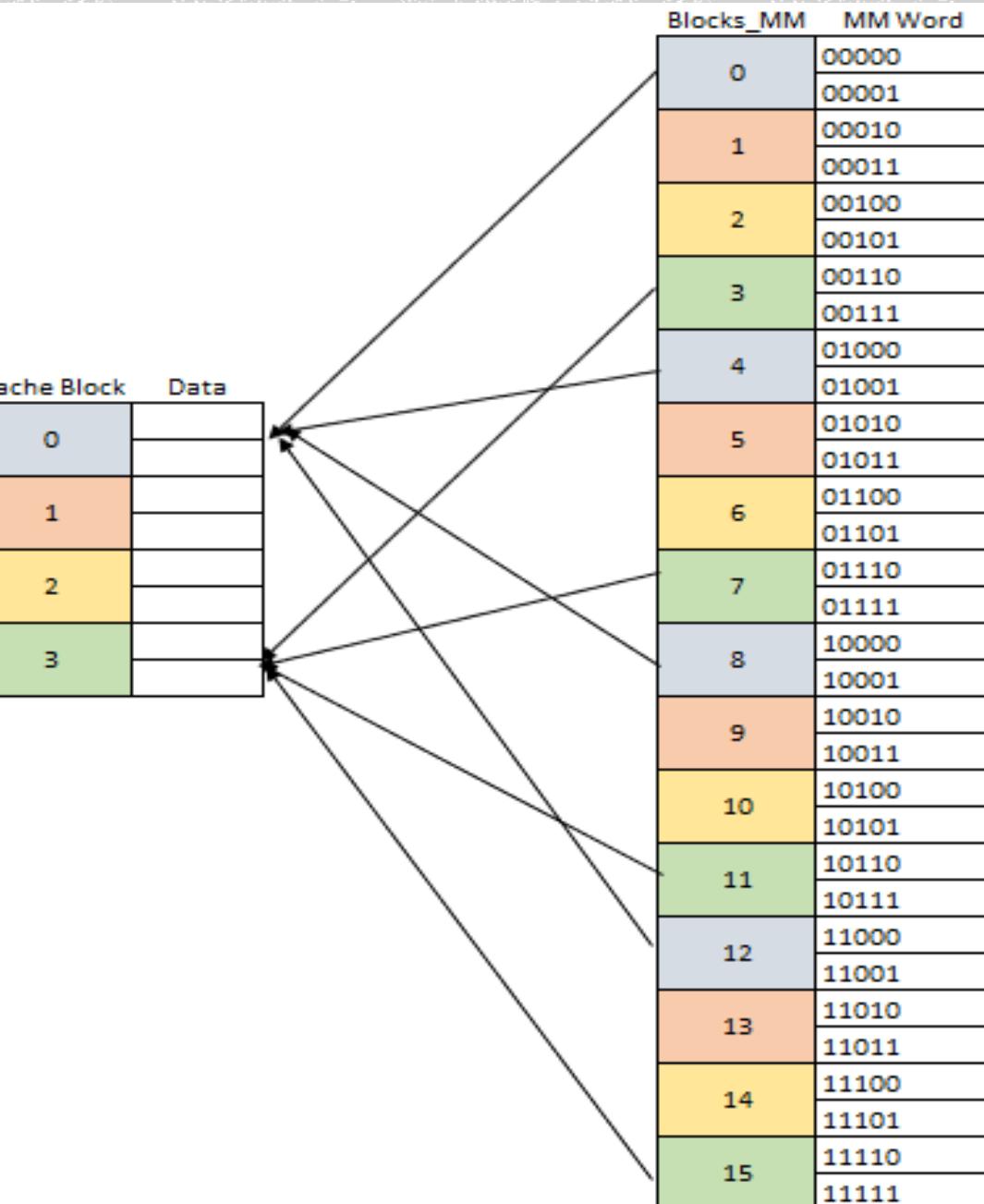
- The write-through protocol is simpler than the write-back protocol
  - But it results in unnecessary Write operations in the main memory when a given cache word is updated several times during its cache residency.
- The write-back protocol also involves unnecessary Write operations
  - Because all words of the block are eventually written back, even if only a single word has been changed while the block was in the cache.
- The write-back protocol is used most often, to take advantage of the high speed with which data blocks can be transferred to memory chips.

# CACHE MISSES

- Read operation for a word that is not in the cache constitutes a Read miss.
- causes the block of words containing the requested word to be copied from the main memory into the cache.
- After the entire block is loaded into the cache, the word requested is forwarded to the processor.
- Alternatively, this word may be sent to the processor as soon as it is read from the main memory. The latter approach, which is called load-through, or early restart, reduces the processor's waiting time somewhat, at the expense of more complex circuitry.
- When a Write miss occurs in a computer that uses the write-through protocol, the information is written directly into the main memory.
- For the write-back protocol, the block containing the addressed word is first brought into the cache, and then the desired word in the cache is overwritten with the new information.

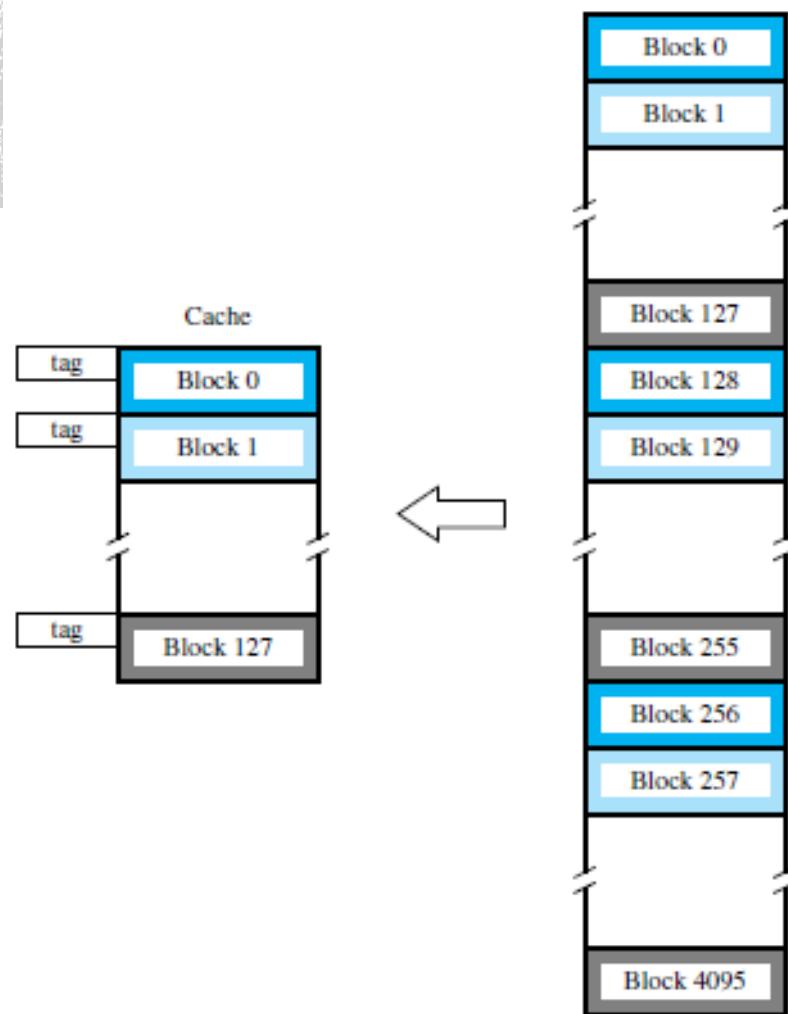
# DIRECT MAPPING

- Block j of main memory maps onto block j modulo 3 of the cache
- To select 1 of 2 word in each block: **1 bit**
- Identify the block among 4 blocks in the cache: **2 bits**
- Out of the 4 blocks in MM that maps to the cache memory, which one is currently residing in the cache?  
**2 bits** (Can also be calculated as  $16/4=4$ : 2 bits)
  
- Example: **11000**
- **Tag:** 11
- **Block:** 00=0, in the 0th block of the cache
- **Word:** 0=0, the 0th word of the 0th block in the cache



# DIRECT MAPPING

- Block j of main memory maps onto block j modulo 128 of the cache
- 4: to select one of 16 words. (each block has  $16=2^4$  words)
- 7: points to a particular block in the cache ( $128=2^7$ )
- 5: 5 tag bits are compared with the tag bits associated with its location in the cache. Identify which of the 32 blocks that are resident in the cache ( $4096/128$ ).
- Example: 11101,1111111,1100
- Tag: 11101
- Block: 1111111=127, in the 127th block of the cache
- Word:1100=12, the 12th word of the 127th block in the cache



Tag	Block	Word	Main memory address
5	7	4	

Figure 8.16 Direct-mapped cache.

# ASSOCIATIVE MAPPING

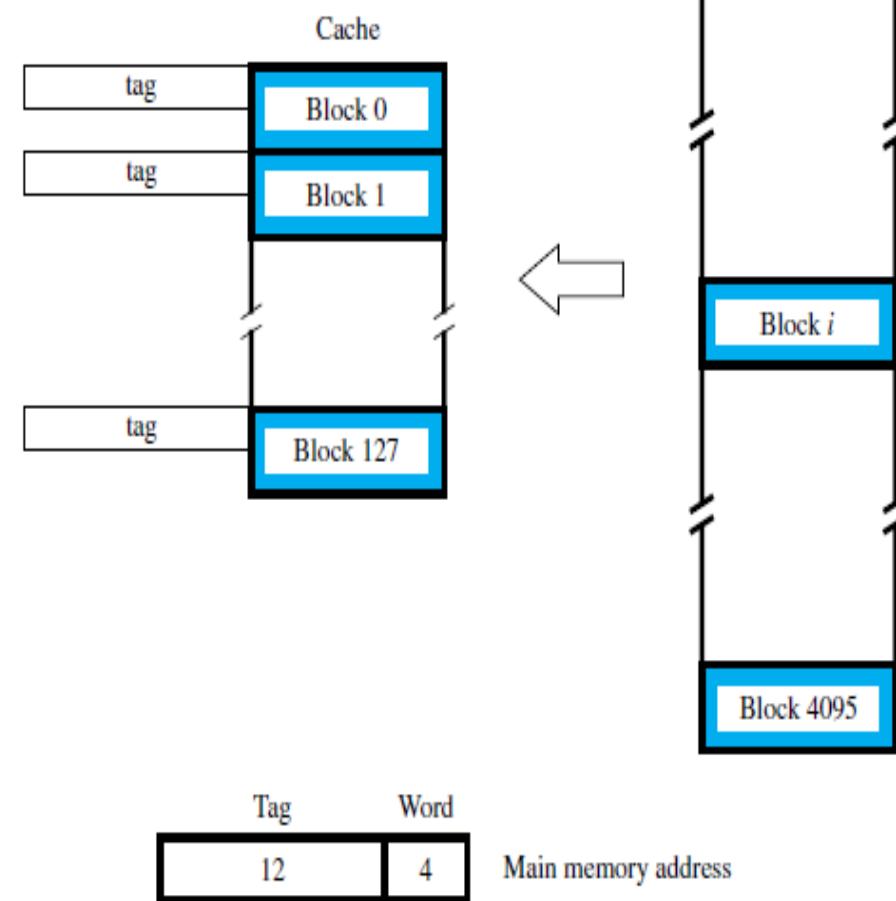
- To select 1 of 2 word in each block: **1 bit**
- Identify which of the 16 blocks that are resident in the cache : **4 bits**
- Example: **11000**
- **Tag:** 1100
- Word: 0=0, the 0th word of a block in the cache

Cache Block(0)	Data(1)
0	
1	
2	
3	

Blocks_MM	MM Word
0	00000
	00001
1	00010
	00011
2	00100
	00101
3	00110
	00111
4	01000
	01001
5	01010
	01011
6	01100
	01101
7	01110
	01111
8	10000
	10001
9	10010
	10011
10	10100
	10101
11	10110
	10111
12	11000
	11001
13	11010
	11011
14	11100
	11101
15	11110
	11111

# ASSOCIATIVE MAPPING

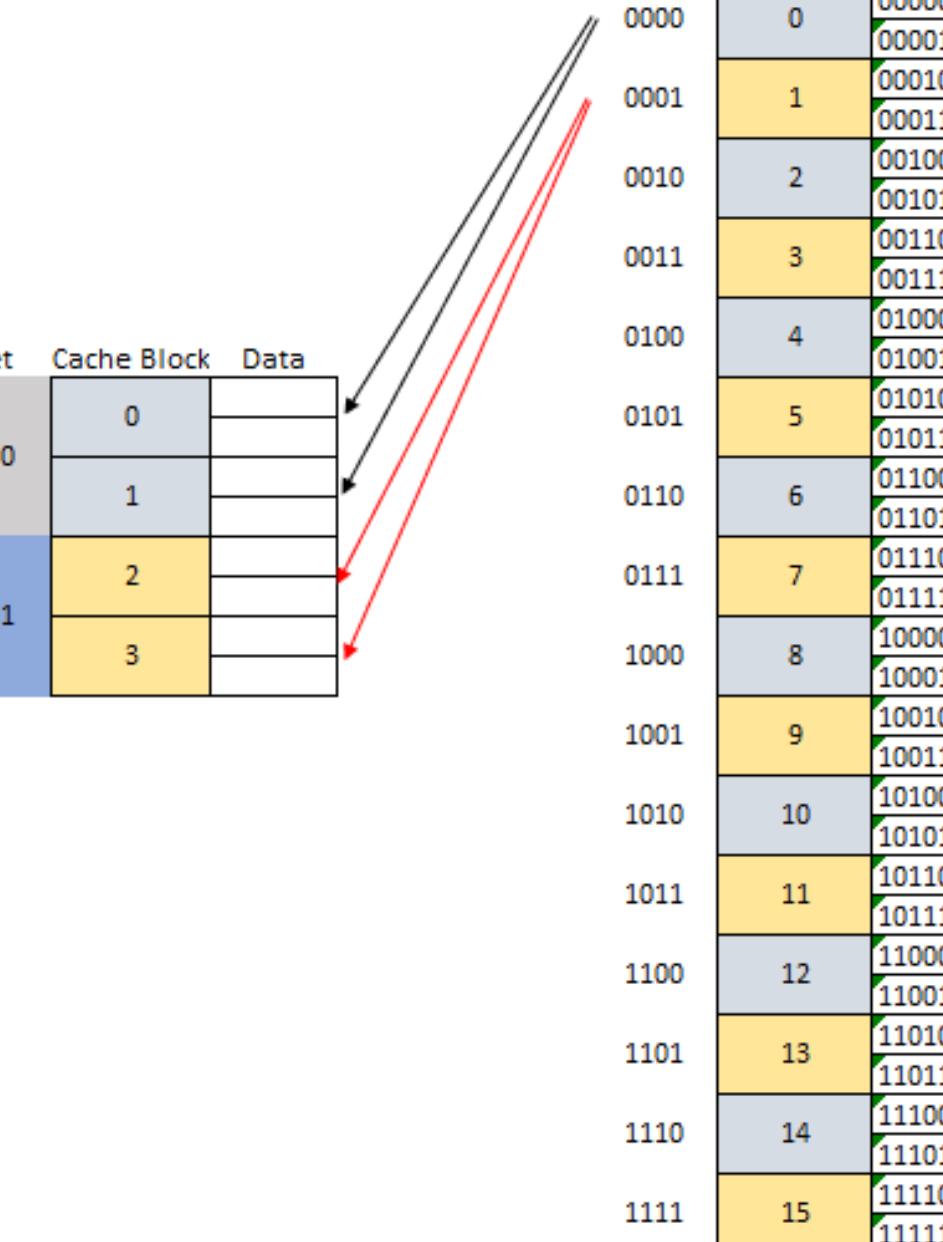
- 4: one of 16 words. (each block has  $16=2^4$  words)
- 12: 12 tag bits Identify which of the 4096 blocks that are resident in the cache  $4096=2^{12}$ .
- Example: 111011111111,1100
- Tag: 111011111111
- Word: 1100=12, the 12th word of a block in the cache



**Figure 8.17** Associative-mapped cache.

# SET ASSOCIATIVE MAPPING

- To select 1 of 2 word in each block: **1 bit**
- Identify the set among 2 sets in the cache: **1 bit**
- Out of the 8 blocks in MM that maps to the cache memory, which one is currently residing in the cache?  
**3 bits** (Can also be calculated as  $16/2=8$ : 3 bits)
- Example: **11000**
- Tag:** 110
- Set:** 0=0, in the 0th set of the cache
- Word: 0=0, the 0th word of the 0th set in the cache



# SET ASSOCIATIVE MAPPING

- 4: one of 16 words. (each block has  $16=2^4$  words)
- 6: points to a particular set in the cache ( $128/2=64=2^6$ )
- 6: 6 tag bits is used to check if the desired block is present ( $4096/64=64=2^6$ ).
- Example: 111011,111111,1100
- Tag: 111011
- Set: 111111=63, in the 63th set of the cache
- Word: 1100=12, the 12th word of the 63th set in the cache

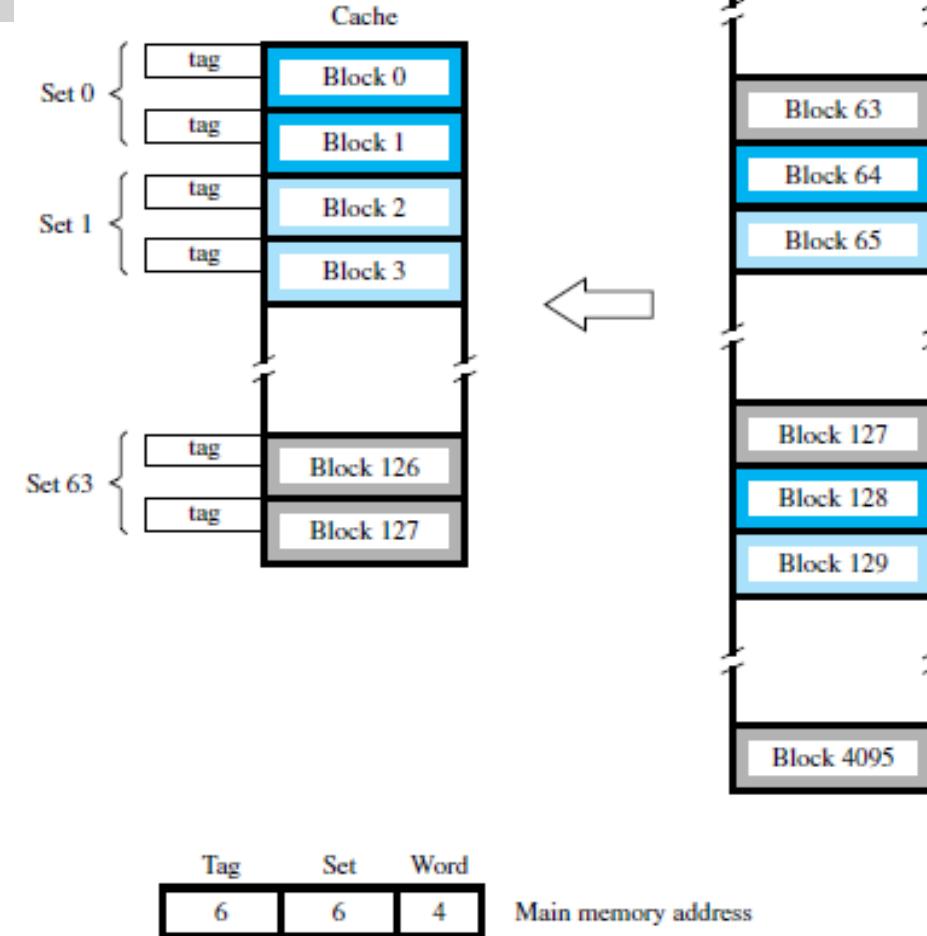


Figure 8.18 Set-associative-mapped cache with two blocks per set.

# EXERCISE PROBLEM 1:

- Consider a 2-way set associative mapped cache of size 16 KB with block size 256 bytes. The size of main memory is 128 KB. Find the size of tag, set and word.
- Solution:
  - No. of blocks in the set= 2
  - Size of the cache=16KB
  - No. of blocks in cache=  $16\text{KB}/256 = 16384\text{B}/256\text{B} = 64$
  - No. of sets in the cache= $64/2=32=2^5$
  - **No. of bits in set field=5**
  - No. of words(bytes) in a block=256= $2^8$
  - **No. of bits in word field is 8**
  - No. of bits in main memory address is 17 bits(log 128KB to base2)
  - **No. of bits in the tag field is 4 (17- 5-8)**

## EXERCISE PROBLEM 2:

- A block-set-associative cache consists of a total of 64 blocks, divided into 4-block sets. The main memory contains 4096 blocks, each consisting of 32 words, how many bits are there in each of the Tag, Set, and Word fields?
- Solution:
  - Number of sets =  $64/4 = 16$
  - No. of bits in Set field =  $4(2^4 = 16)$
  - Word bits = 5 bits ( $2^5 = 32$ )
  - Tag =  $17 - (4 + 5) = 8$

# EXERCISE PROBLEM 3:

- A block-set-associative cache consists of a total of 64 blocks, divided into 4-block sets. The main memory contains 4096 blocks, each consisting of 128 words.
  - a) How many bits are there in MM address?
  - b) How many bits are there in each of the TAG, SET & WORD fields
- Solution:
  - Number of sets =  $64/4 = 16$
  - Set bits =  $4(2^4 = 16)$
  - Number of words = 128
  - Word bits = 7 bits ( $2^7 = 128$ )
  - MM capacity :  $4096 \times 128 (2^{12} \times 2^7 = 2^{19})$
  - a) Number of bits in memory address = 19 bits
  - b)

8	4	7
TAG SET WORD		

- TAG bits =  $19 - (7+4) = 8$  bits.

# TOPICS COVERED FROM

- Textbook 1:
  - Chapter 8: 8.1, 8.2, 8.2.1, 8.2.5, 8.5, 8.6: 8.6.1