

Computer Organisation and Architecture

Course Code: CO206

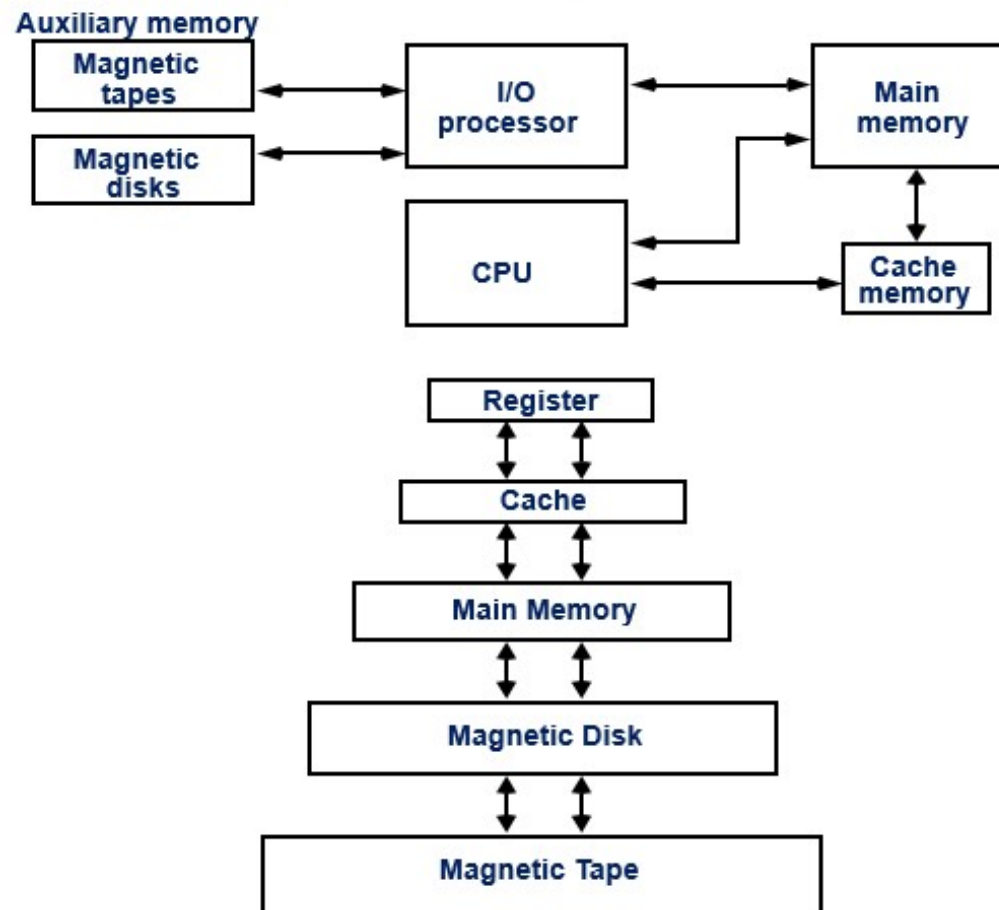
Module-6 : Memory Organisation

Contents of Module 6

- ☐ Memory Hierarchy
- ☐ Main Memory
- ☐ Auxiliary Memory
- ☐ Associative Memory
- ☐ Cache Memory: Concept and Design issues
- ☐ Associative Mapping
- ☐ Direct Mapping
- ☐ Set-Associative Mapping
- ☐ Cache writing and initialisation

Memory Hierarchy

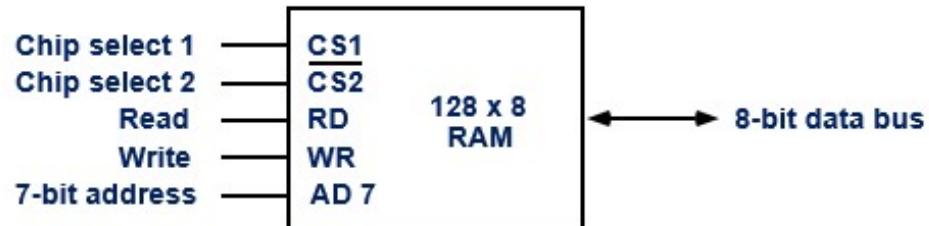
Memory Hierarchy is to obtain the highest possible access speed while minimizing the total cost of the memory system



Main Memory

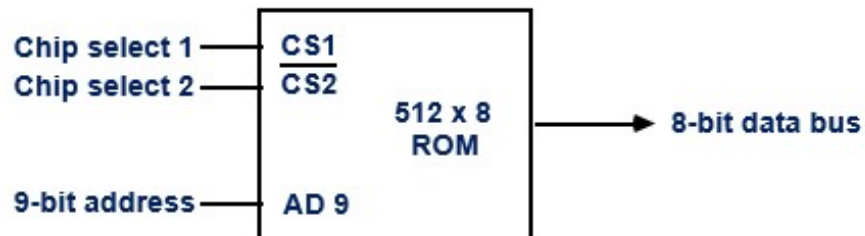
RAM and ROM Chips

Typical RAM chip



CS1	$\overline{\text{CS2}}$	RD	WR	Memory function	State of data bus
0	0	x	x	Inhibit	High-impedence
0	1	x	x	Inhibit	High-impedence
1	0	0	0	Inhibit	High-impedence
1	0	0	1	Write	Input data to RAM
1	0	1	x	Read	Output data from RAM
1	1	x	x	Inhibit	High-impedence

Typical ROM chip



Memory Address Map

Address space assignment to each memory chip

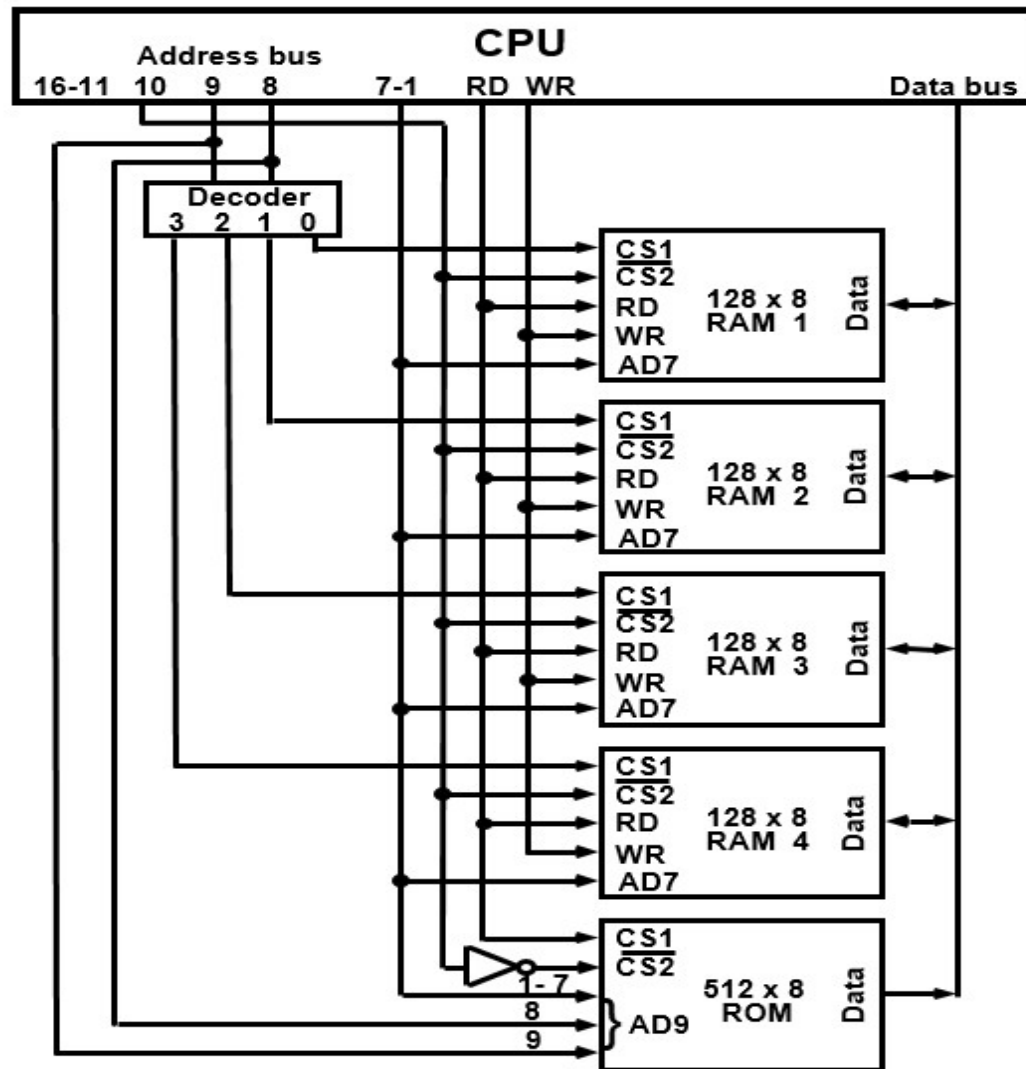
Example: 512 bytes RAM and 512 bytes ROM

Component	Hexa address	Address bus									
		10	9	8	7	6	5	4	3	2	1
RAM 1	0000 - 007F	0	0	0	x	x	x	x	x	x	x
RAM 2	0080 - 00FF	0	0	1	x	x	x	x	x	x	x
RAM 3	0100 - 017F	0	1	0	x	x	x	x	x	x	x
RAM 4	0180 - 01FF	0	1	1	x	x	x	x	x	x	x
ROM	0200 - 03FF	1	x	x	x	x	x	x	x	x	x

Memory Connection to CPU

- RAM and ROM chips are connected to a CPU through the data and address buses
- The low-order lines in the address bus select the byte within the chips and other lines in the address bus select a particular chip through its chip select inputs

Connection of Memory and CPU



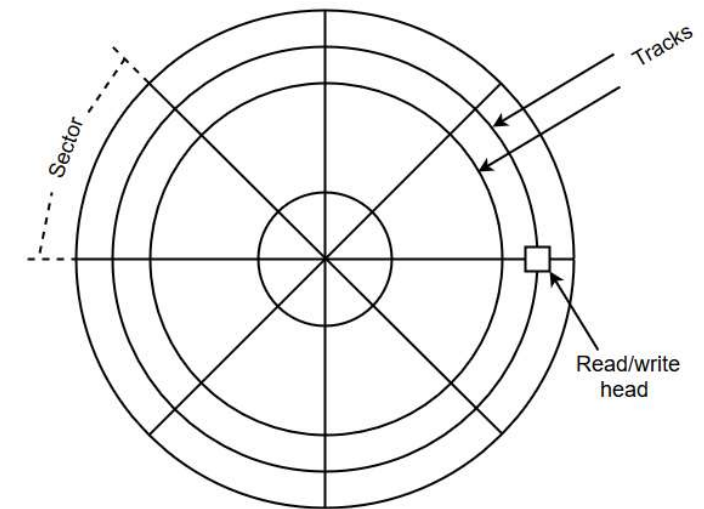
Auxiliary Memory

- ❑ An Auxiliary memory is known as the **lowest-cost, highest-capacity and slowest-access storage** in a computer system.
- ❑ It is **where programs and data are kept for long-term storage** or when not in immediate use.
- ❑ It **holds programs and data for future use**, and, **because it is non-volatile** and it is used to store inactive programs and to archive data.
- ❑ **Early forms of auxiliary storage** included **punched paper tape, punched cards, and magnetic drums**.
- ❑ **Since 1980s**, the **most common forms** of auxiliary storage have been **magnetic disks, magnetic tapes, optical discs and pen drives**.

Auxiliary Memory: Magnetic Disk

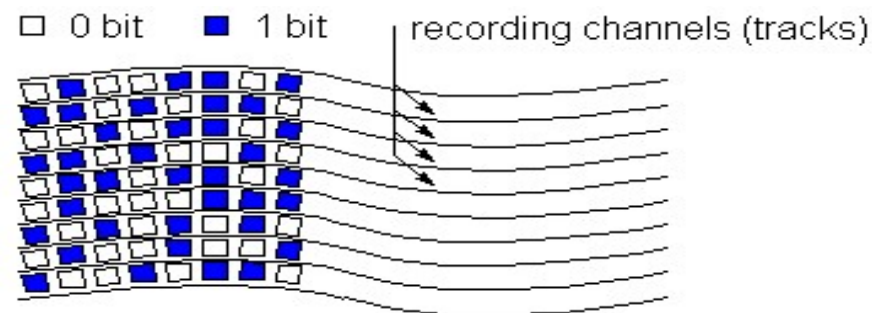
- ❑ A magnetic disk is a type of memory constructed using a circular plate of metal or plastic coated with magnetized materials.
- ❑ Usually, both sides of the disks are used to carry out read/write operations.
- ❑ However, several disks may be stacked on one spindle with read/write head available on each surface.
- ❑ The memory bits are stored in the magnetized surface in spots along the concentric circles called tracks.
- ❑ The concentric circles (tracks) are commonly divided into sections called sectors.

Magnetic disks



Auxiliary Memory: Magnetic Tape

- ❑ Magnetic tape is a storage medium that allows data archiving, collection, and backup for different kinds of data.
- ❑ The magnetic tape is constructed using a plastic strip coated with a magnetic recording medium.
- ❑ The bits are recorded as magnetic spots on the tape along several tracks. Usually, seven or nine bits are recorded simultaneously to form a character together with a parity bit.
- ❑ Magnetic tape units can be halted, started to move forward or in reverse, or can be rewind. However, they cannot be started or stopped fast enough between individual characters. For this reason, information is recorded in blocks referred to as records.



Numerical

Question:

Consider the system which has 256KW in the main memory and each word has the size of 64 bits. What is the capacity of main memory in Bytes?

Solution:

$$\begin{aligned}\text{Capacity of main memory} &= 256\text{KW} \times 64 \text{ bits} \\ &= \frac{256 \times 64}{8} \\ &= 256 \times 8\text{K} = 2^8 \times 2^3 \times 2^{10}; \\ &= 2048\text{KB} \\ &= 2\text{MB}\end{aligned}$$

Numerical

Question:

Consider a system which has 512MW in the main memory and each word has the size of 16 bytes. What is the capacity of main memory in Bytes?

Solution:

$$\begin{aligned}\text{Capacity of main memory} &= 512\text{MW} \times 16\text{B} \\ &= 2^9 \times 2^{20} \times 2^4\text{B} \\ &= 2^3 \times 2^{30}\text{B} \\ &= 8\text{GB}\end{aligned}$$

Numerical

Question:

If the size of a RAM chip is 128B. What will be the number of RAM chip required to organize the main memory capacity of 16KB?

Solution:

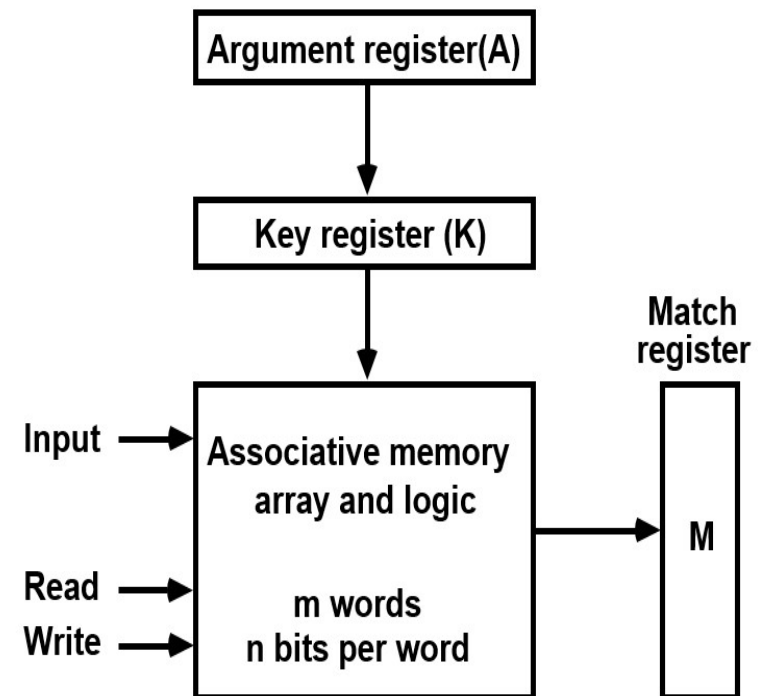
$$\begin{aligned}\text{Number of RAM chip required} &= \frac{\text{Capacity of main memory}}{\text{RAM chip size}} \\ &= \frac{16\text{KB}}{128\text{B}} \\ &= \frac{2^4 \times 2^{10}}{2^7} \\ &= 128\end{aligned}$$

Associative Memory

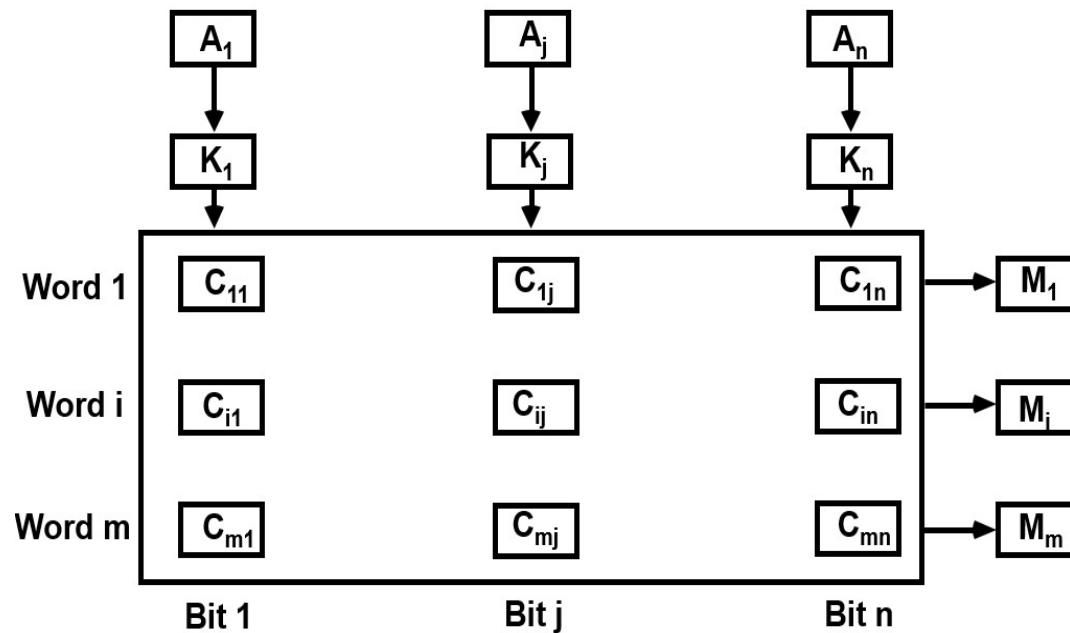
- ❑ **To search a particular data in memory**, data is read from certain address and compared if the match is not found, content of the next address is accessed and compared.
- ❑ This goes on until required data is found.
- ❑ The **number of access depend on the location of data and efficiency of searching algorithm**
- ❑ This **searching time** can be **reduced** if data is searched on the **basis of content**.
- ❑ A **memory unit accessed by content** is called **Associative Memory** or **content addressable memory(CAM)**
- ❑ This type of **memory** is **accessed simultaneously** and **in parallel** on the basis of **data content**.
- ❑ **Memory** is **capable of finding empty unused location to store the word**.

Block Diagram of Associative Memory

- ❑ **Argument Register(A):** It contains the word to be searched. It has n bits(one for each bit of the word)
- ❑ **Key Register(K):** This specifies which part of the argument word needs to be compared with words in memory. If all bits in register are 1, the entire word should be compared. Otherwise only the bits having k -bits set to 1 will be compared.
- ❑ **Associative Memory Array:** It contains the words which are to be compared with the argument word.
- ❑ **Match Register(M):** It has m bits, one bit corresponding to each word in the memory array. After the matching process, the bits corresponding to matching words in match register are set to 1.



Associative Memory



A 101 111100

K 111 000000

Word 1 100 111100

no match

Word 2 101 000001

match

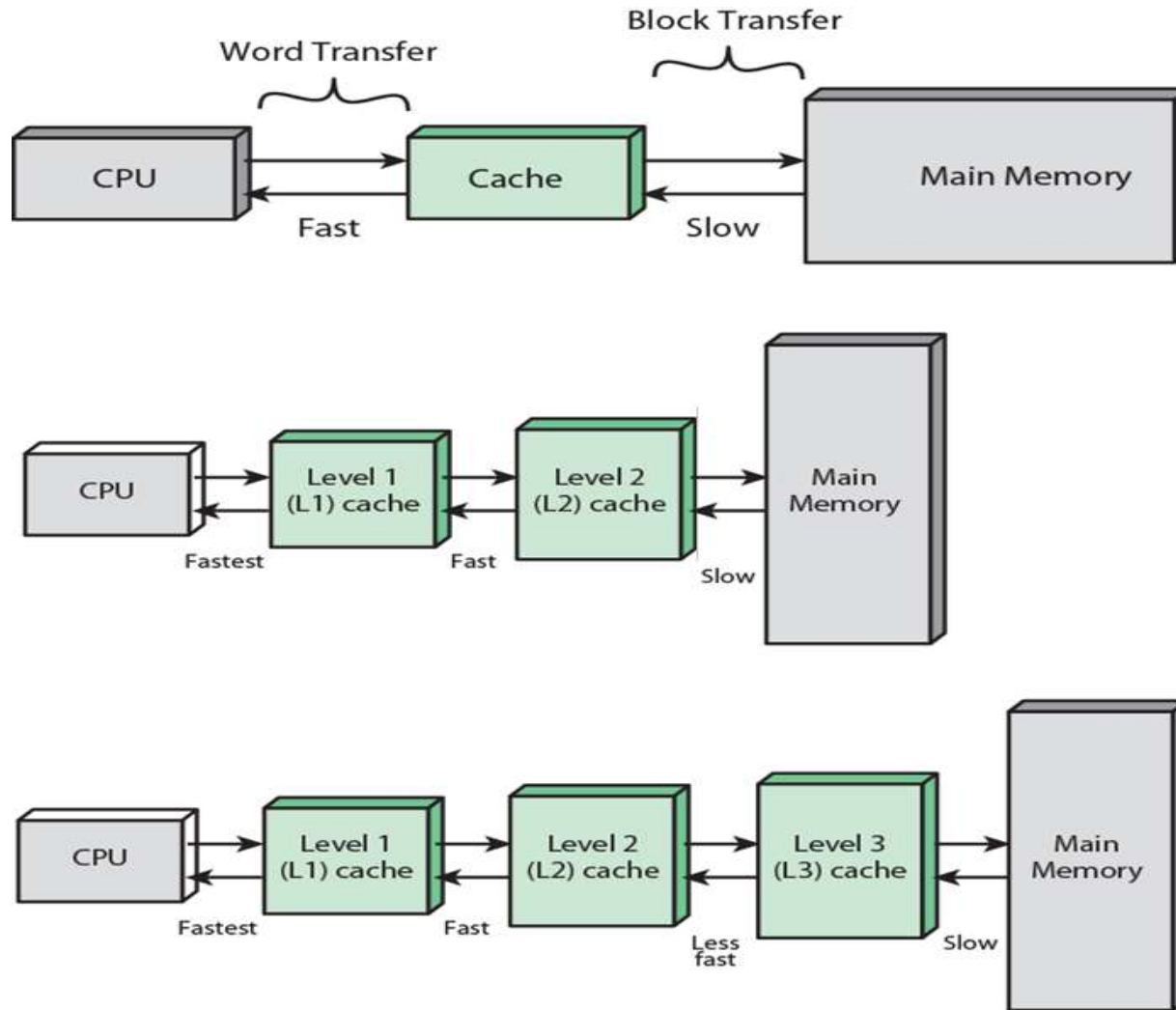
Cache Memory

- ❑ Processor is much faster than the main memory.
- ❑ As a result, the processor has to spend much of its time waiting while instructions and data are being fetched from the main memory.
- ❑ Speed of the main memory cannot be increased beyond a certain point.
- ❑ Major obstacle towards achieving good performance.
- ❑ Introducing Cache in between CPU and main memory can solve the problem.
- ❑ Cache memory is an architectural arrangement which makes the main memory appear faster to the processor than it really is.
- ❑ It is a special very high-speed memory which is used to speed up and synchronizing with high-speed CPU.

Cache Memory

- ❑ **Cache memory is costlier than main memory or disk memory but economical than CPU registers.**
- ❑ **It is an extremely fast memory type that acts as a buffer between RAM and the CPU.**
- ❑ **It holds copies of frequently requested data and instructions so that they are immediately available to the CPU when needed.**
- ❑ **Cache memory is used to reduce the average time to access data from the main memory.**
- ❑ **There are various different independent caches in a CPU, which store instructions and data.**

Cache Organisation



Cache Memory: Locality of Reference

- ❑ Cache memory is based on the **property of computer programs** known as “locality of reference”.
- ❑ Analysis of programs indicates that **many instructions in localized areas of a program are executed repeatedly during some period of time, while the others are accessed relatively less frequently.**
- ❑ These instructions may be the ones in a **loop, nested loop or few procedures calling each other repeatedly.**
- ❑ This is called “locality of reference”.
- ❑ Temporal locality of reference:
 - Recently executed instruction is likely to be executed again very soon.
- ❑ Spatial locality of reference:
 - Instructions with addresses close to a recently executed instruction are likely to be executed soon.

Cache Memory: Performance

- ❑ **Hit:** CPU finding contents of memory address in cache
- ❑ **Hit ratio (h) :** probability of successful lookup in cache by CPU.
- ❑ **Hit ratio(h)= number of hits/ total no. of access**
- ❑ **Miss:** CPU failing to find what it wants in cache (incurs trip to deeper levels of memory hierarchy)
- ❑ **Miss ratio (m):** probability of missing in cache and is equal to 1-h.
- ❑ **Miss penalty:** Time penalty associated with servicing a miss at any particular level of memory hierarchy
- ❑ **Effective Memory Access Time (EMAT):** Effective access time experienced by the CPU when accessing memory.
 - ❑ Time to lookup cache to see if memory location is already there
 - ❑ Upon cache miss, time to go to deeper levels of memory hierarchy

$$T_e = T_c + (1 - h) T_m$$

T_e : Effective memory access time in Cache memory system

T_c : Cache access time

T_m : Main memory access time

Numerical

Question:

In a computer system, the cache access time is 100ns and main memory access time is 1000 ns. The hit ratio is 0.9. what will be the average access time?

Solution:

$$T_e = T_c + (1 - h) T_m$$

$$T_e = 100 + 0.1 * 1000$$

$$T_e = 100 + 100 = 200 \text{ ns}$$

Cache Memory Mapping

- ❑ The transformation of data from main memory to cache memory is referred to as mapping process.
- ❑ A mapping function is the method used to locate a memory address within a cache
- ❑ It is used when copying a block from main memory to the cache and it is used again when trying to retrieve data from the cache
- ❑ There are three kinds of mapping functions
 - Direct Mapping
 - Associative Mapping
 - Set Associative Mapping

Some presumptions

- ☐ Main memory is divided into blocks
- ☐ Each block can contain K words
- ☐ Cache memory is divided into lines(blocks)
- ☐ The size of Line in Cache memory is equal to size of block in main memory

Direct Mapping

❑ Example

❑ Main memory size = 64 x 8

❑ Cache memory size = 16 x 8

❑ Block size = 4 words

$$\begin{aligned}\text{❑ Number of main memory blocks} &= \frac{\text{Total main memory words}}{\text{block size}} \\ &= \frac{64}{4} = 16 \text{ blocks}\end{aligned}$$

$$\begin{aligned}\text{❑ Number of cache memory lines} &= \frac{\text{Total cache memory words}}{\text{block size}} \\ &= \frac{16}{4} = 4 \text{ lines}\end{aligned}$$

$$\begin{aligned}\text{❑ Size of Tag} &= \frac{\text{Total blocks in main memory}}{\text{Total lines in Cache memory}} \\ &= \frac{16}{4} = 4\end{aligned}$$

Direct Mapping

❑ In this technique, each block of main memory is mapped to only one possible cache line

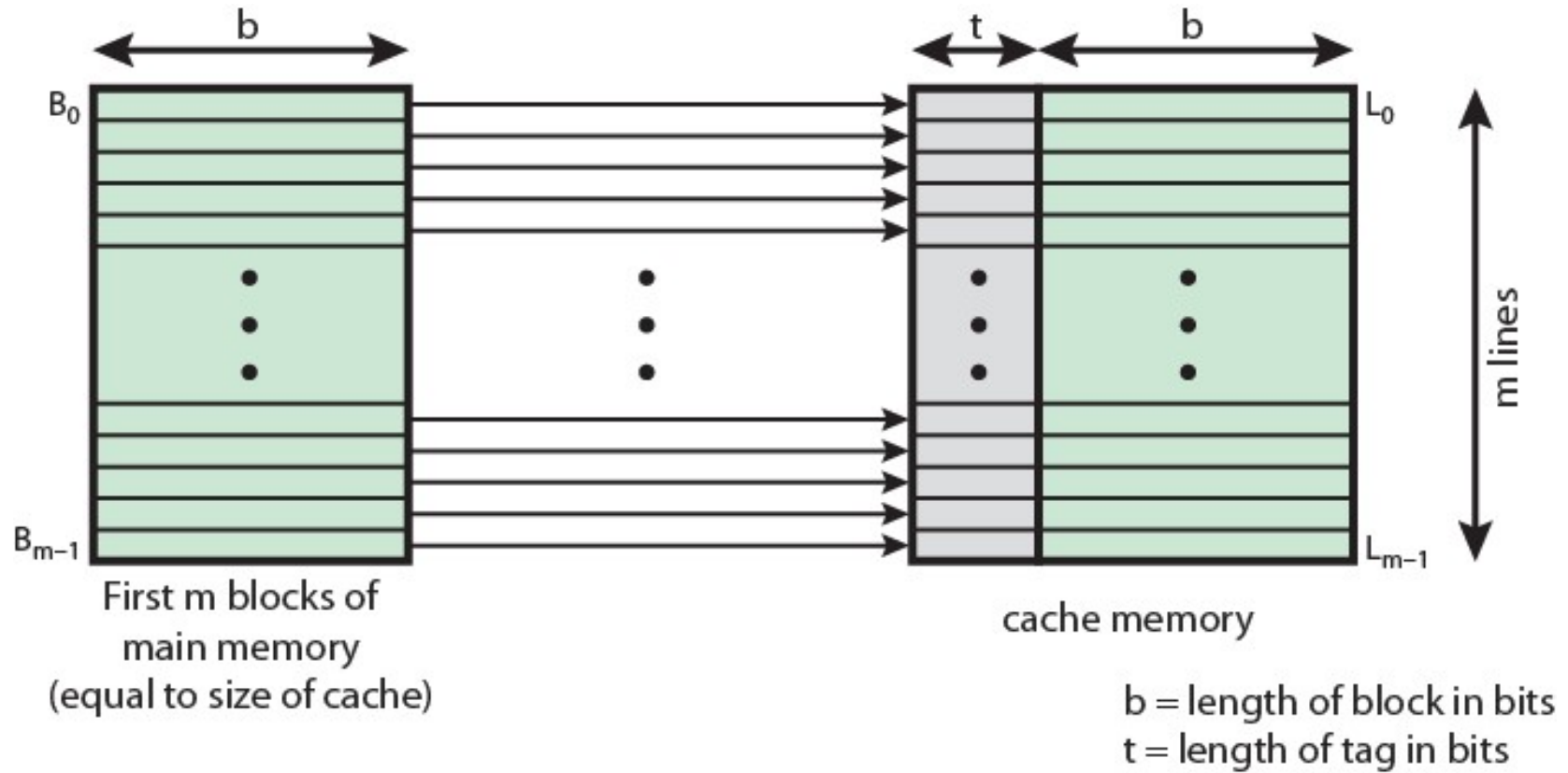
❑ The mapping is expressed as

$$i = j \text{ modulo } m$$

where

- i = cache line number
- j = main memory block number
- m = number of lines in the cache

Direct Mapping



(a) Direct mapping

Direct Mapping

- ❑ If Block No. 14 is to be transferred to Cache then what will be the Cache line number?
- ❑ $i = j \text{ modulo } m$
- ❑ $14 \bmod 4 = 2$

Line No.	Block No.
0	0/4/8/12
1	1/5/9/13
2	2/6/10/14
3	3/7/11/15

Cache Memory

Block Number	Binary Representation
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
10	1010
11	1011
12	1100
13	1101
14	1110
15	1111

Main Memory

Block Number	Words
0	0,1,2,3
1	4,5,6,7
2	8,9,10,11
3	12,13,14,15
4	16,17,18,19
5	20,21,22,23
6	24,25,26,27
7	28,29,30,31
8	32,33,34,35
9	36,37,38,39
10	40,41,42,43
11	44,45,46,47
12	48,49,50,51
13	52,53,54,55
14	56,57,58,59
15	60,61,62,63

Main Memory


Direct Mapping Address Structure

- ❑ In Direct Mapping, following three things are required

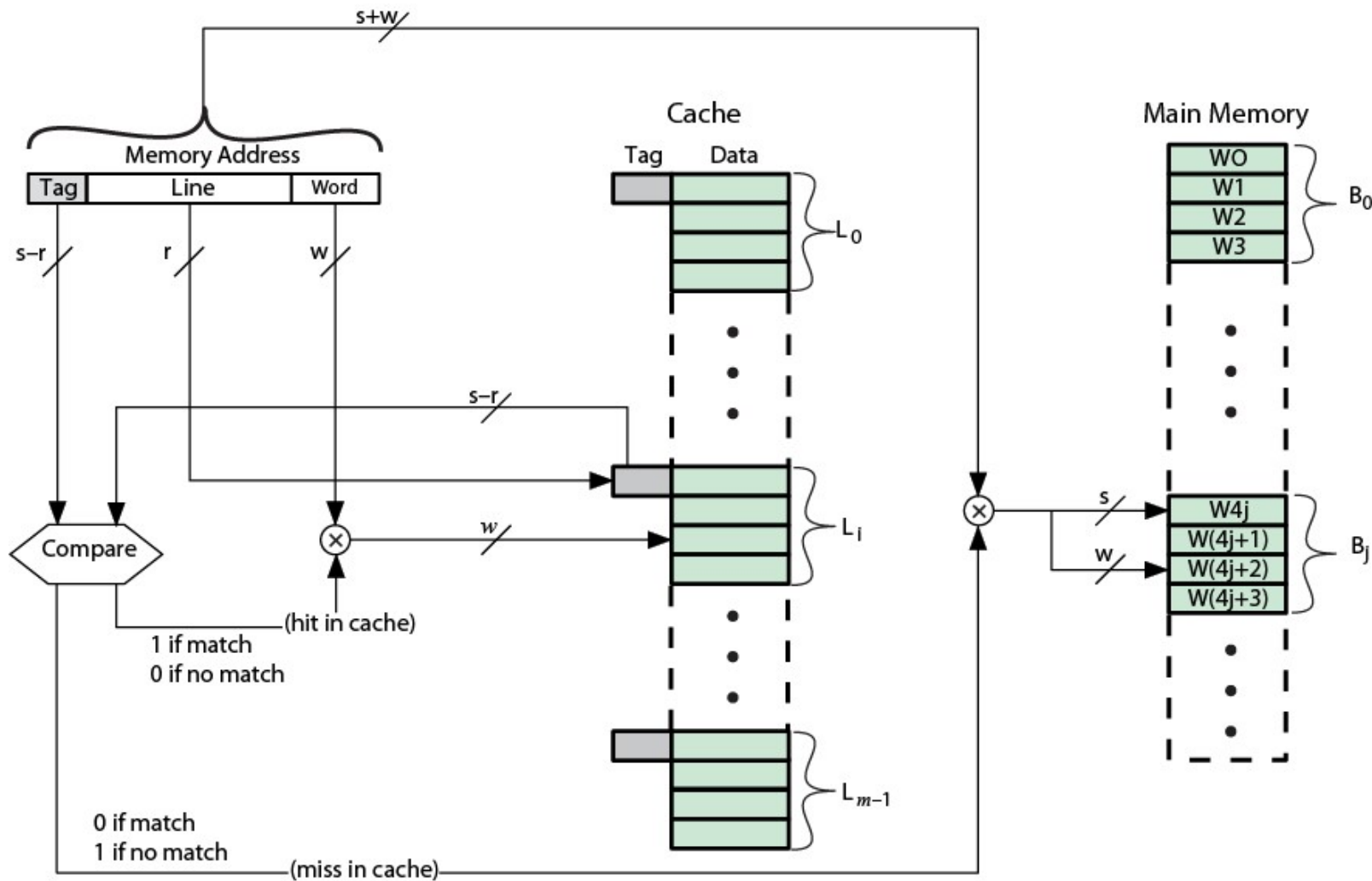
Tag	Line No.	Word
-----	----------	------

- ❑ In the given example, CPU will generate the address of 6 bits

Tag	Line No.	Word
2	2	2


Block Number

Direct Mapping



- S = block number
- r = line number
- w = word number
- $(s-r)$ = tag number

Direct Mapping Pros and Cons

- ❑ Simple
- ❑ Inexpensive
- ❑ Fixed location for given block
 - If a program accesses 2 blocks that map to the same line repeatedly, cache misses are very high

Numerical

Question:

Consider the system is using Direct mapping in cache. The size of cache is 32KB, Block size is 32B, Main memory has 2^{32} word. What is the number of bits needed for cache indexing and tag bits? (Note: If word size is not mentioned, presume memory is byte addressable)

Solution:

Total bits in address= 32

Number of words in each block= $32 = 2^5$

Number of lines in cache= cache size/ block size = $2^{15}/2^5 = 2^{10}$

Size of tag= Total block in main memory/total lines in cache $2^{27}/2^{10} = 2^{17}$

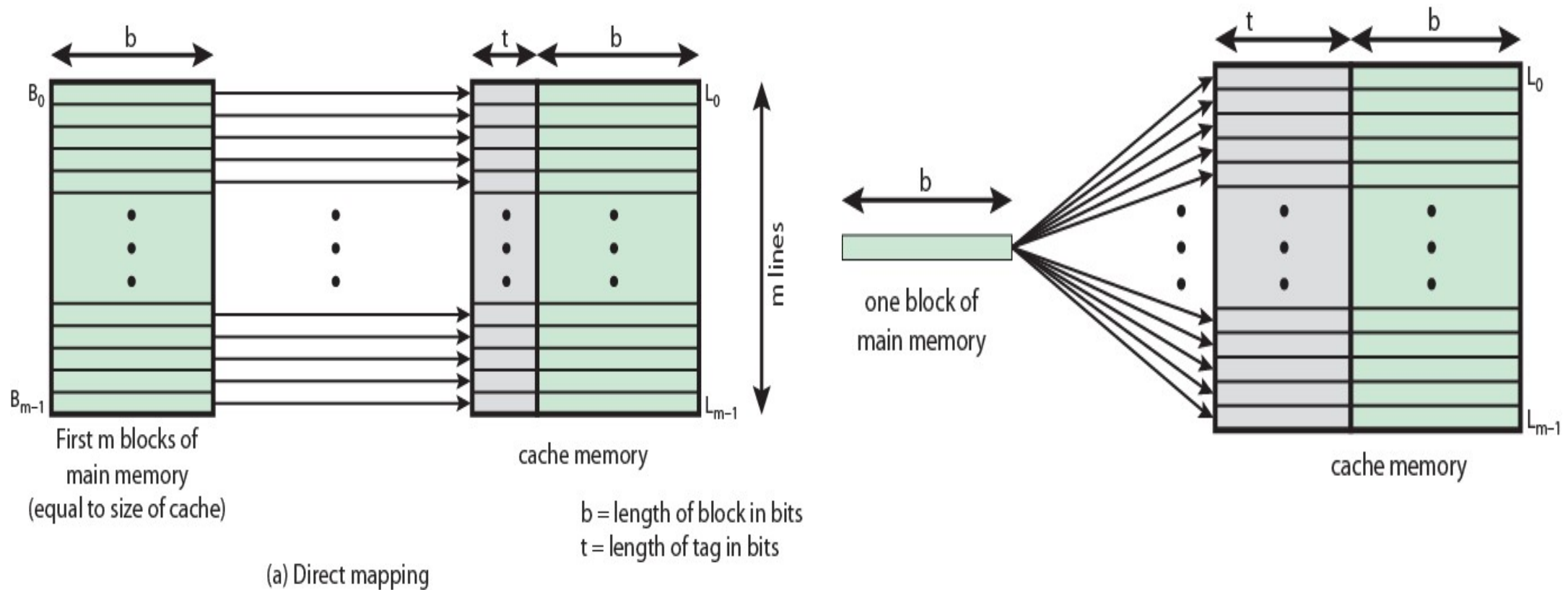
Tag	Line No.	Word
17	10	5

32 bits

Associative Mapping

- ❑ A main memory block can load into any line of cache
- ❑ Memory address is interpreted as tag and word
- ❑ Tag uniquely identifies block of memory
- ❑ Every line's tag is examined for a match
- ❑ Cache searching gets expensive

Associative Mapping




Associative Mapping Address Structure

- ❑ In Associative Mapping, following two things are required

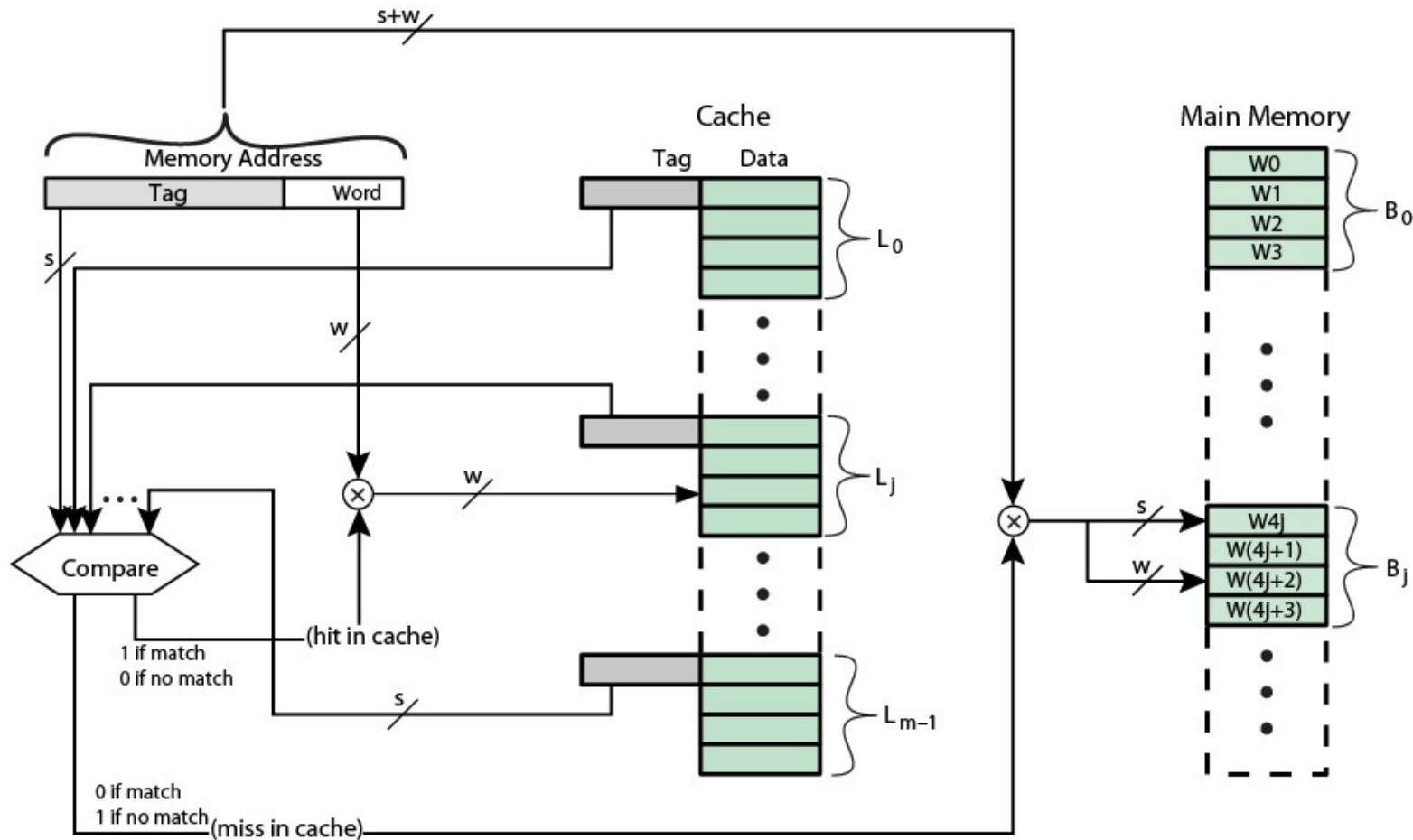
Tag	Word
-----	------

- ❑ In the given example, CPU will generate the address of 6 bits

Tag	Word
4	2


Block Number

Associative Mapping



- s = block number
- w = word number

Numerical

Question:

Consider the system is using Associative mapping in cache. The size of cache is 16KB, Block size is 256B, Main memory has 128KB. What is the number of bits needed for cache tag ? (Note: If word size is not mentioned, presume memory is byte addressable)

Solution:

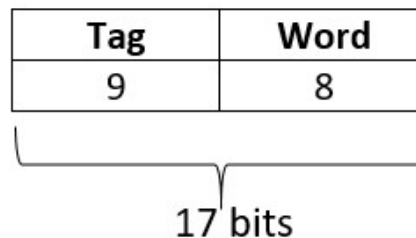
Total bits in address= 17

Number of words in each block= $256 = 2^8$

Number of blocks in main memory= $2^{17}/2^8 = 2^9$

Number of lines in cache= cache size/ block size = $2^{14}/2^8 = 2^6$

Size of tag= Total blocks in main memory = 2^9



Set Associative Mapping

- ❑ Cache is divided into a number of sets
- ❑ k blocks/lines can be contained within each set and it is called a k -way set associative mapping
- ❑ One extreme is to have all the blocks in one set, requiring no set bits (fully associative mapping).
- ❑ Other extreme is to have one block per set, is the same as direct mapping.

Set Associative Mapping

❑ Example

❑ Main memory size = 64 x 8, Cache memory size= 16 x 8, Block size= 4 words, Set size=2 (2-way set associative mapping is used).

❑ Number of main memory blocks = $\frac{\text{Total main memory words}}{\text{block size}} = \frac{64}{4} = 16$ blocks

❑ Number of cache memory lines = $\frac{\text{Total cache memory words}}{\text{block size}} = \frac{16}{4} = 4$ lines

❑ Number of Sets = $\frac{\text{Number of Block}}{\text{Set Size}} = \frac{4}{2} = 2$

❑ Size of Tag = $\frac{\text{Total blocks in main memory}}{\text{Cache memory sets}} = \frac{16}{2} = 8$

Set Associative Mapping

❑ In this technique, each block of main memory is mapped to only one possible cache set

❑ The mapping is expressed as

$$i = j \text{ modulo } m$$

where

- i = cache set number
- j = main memory block number
- m = number of sets in the cache

Set Associative Mapping

- ❑ If Block No. 14 is to be transferred to Cache then what will be the Cache set number?
- ❑ $i = j \text{ modulo } m$
- ❑ $14 \bmod 2 = 0$

Set No.	Block No.	Block No.
0	0/4/8/12/2/6/10/14	0/4/8/12/2/6/10/14
1	1/5/9/13/3/7/11/15	1/5/9/13/3/7/11/15

Cache Memory

Block Number	Words
0	0,1,2,3
1	4,5,6,7
2	8,9,10,11
3	12,13,14,15
4	16,17,18,19
5	20,21,22,23
6	24,25,26,27
7	28,29,30,31
8	32,33,34,35
9	36,37,38,39
10	40,41,42,43
11	44,45,46,47
12	48,49,50,51
13	52,53,54,55
14	56,57,58,59
15	60,61,62,63

Main Memory

Set Associative Mapping Address Structure

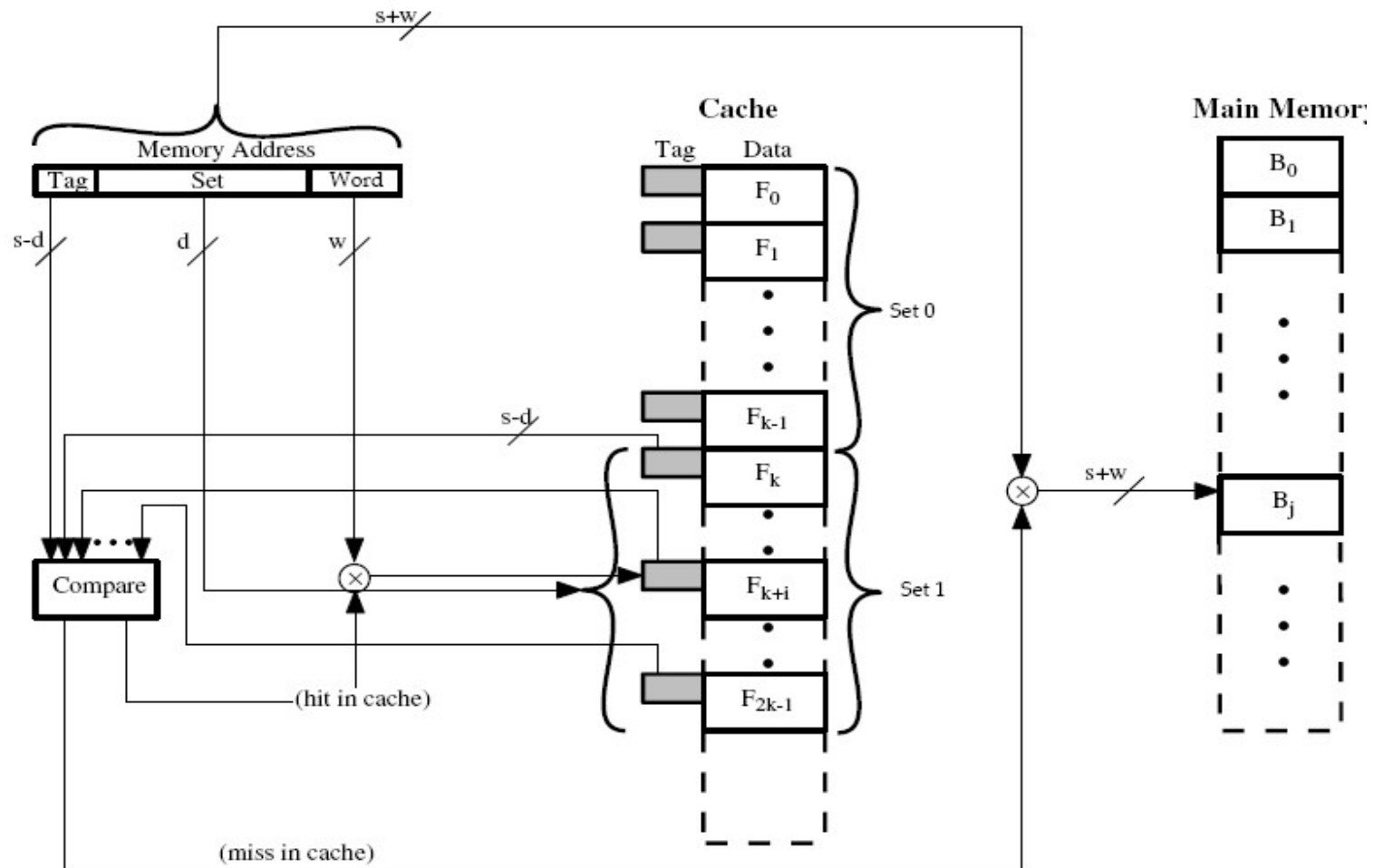
- ❑ In Set Associative Mapping, following three things are required

Tag	Set	Word
-----	-----	------

- ❑ In the given example, CPU will generate the address of 6 bits. There is 2-way set associative mapping

Tag	Set	Word
3	1	2

Set Associative Mapping



Numerical

Question:

Consider the system is using 4-way Set- Associative mapping in cache. The size of cache is 16KB, Block size is 8words, word size is 32 bits. Main memory is 4GB. What is the number of bits needed for cache tag? (Note: any thing is given in word, then always convert everything in word)

Solution:

Cache size = 16KB/32bits(4B) = 4KW

Block size= 8W

Main Memory= 1GW

Total Address bits= 2^{30}

Number of sets in cache= cache size/ set size = $2^{12}/2^5 = 2^7$

Size of tag = Total blocks in main memory/total sets in cache $2^{30}/2^{10} = 2^{20}$

Cache Replacement Algorithm

- ❑ When a miss occurs in Cache memory and it is full, it is necessary to replace one of the tag data item with new value.
- ❑ The most common replacement are
 - ❑ Random Replacement
 - ❑ First In First Out(FIFO)
 - ❑ Least Recently Used (LRU)

Cache Replacement Algorithm-FIFO

- ❑ When the Cache is full, the oldest block is replaced. Hence first-in, first-out

reference

7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1

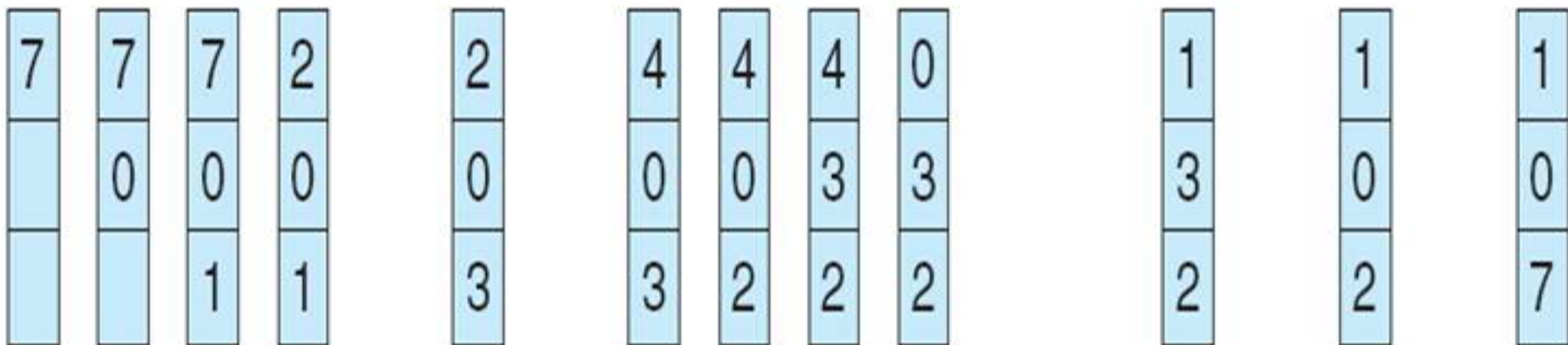
7	7	7	2		2	2	4	4	4	0			0	0			7	7	7
	0	0	0		3	3	3	2	2	2			1	1			1	0	0
		1	1		1	0	0	0	3	3			3	2			2	2	1

Cache Replacement Algorithm-LRU

- ❑ When the Cache is full, Replace the block that has not been referenced for the longest time:

reference

7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1



Cache Writing

- ❑ An important aspect of cache organization is concerned with memory write request.
- ❑ When the CPU finds a word in cache during read operation, the main memory is not involved in the transfer.
- ❑ However, if the operation is a write, there are two ways that the system can proceed
 - ❑ Write Through
 - ❑ Write Back

Cache Write Back

- ☐ Updates initially made in cache only
- ☐ Update bit for cache slot is set when update occurs
- ☐ If block is to be replaced, write to main memory only if update bit is set
- ☐ Other caches get out of sync.
- ☐ I/O must access main memory through cache
- ☐ Analytical results indicate 10-30% of memory references are writes

Cache Write Through

- ❑ All writes go to main memory as well as cache
- ❑ Multiple CPUs can monitor main memory traffic to keep local (to CPU) cache up to date
- ❑ Lots of traffic
- ❑ Slows down writes
- ❑ Remember bogus write through caches!