Cache Memory

Contents of Lecture:

- Elements of Cache Design
  - ✓ Cache Addresses
  - ✓ Cache Size
  - ✓ Mapping Function
  - ✓ Replacement Algorithms
- Summarize number(1)

References for This Lecture:

- ✓ William Stallings, Computer Organization and Architecture Designing For Performance, 9th Edition, Chapter 4 : *Cache Memory* , pages 123 to 141

Elements of Cache Design:

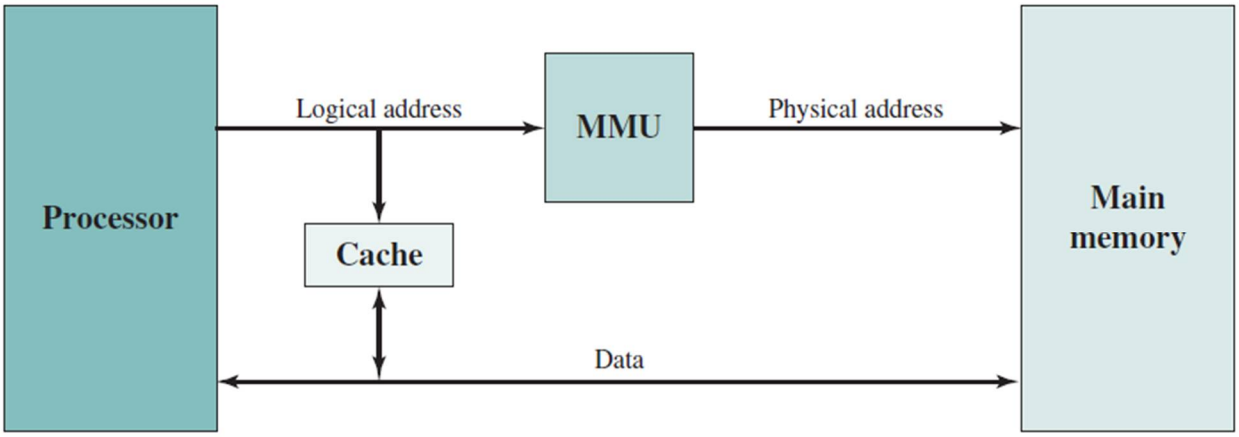
- There are a few basic design elements that serve to classify and differentiate cache architectures.
- Following Table Table 4.2 lists key elements.

Table 4.2 Elements of Cache Design

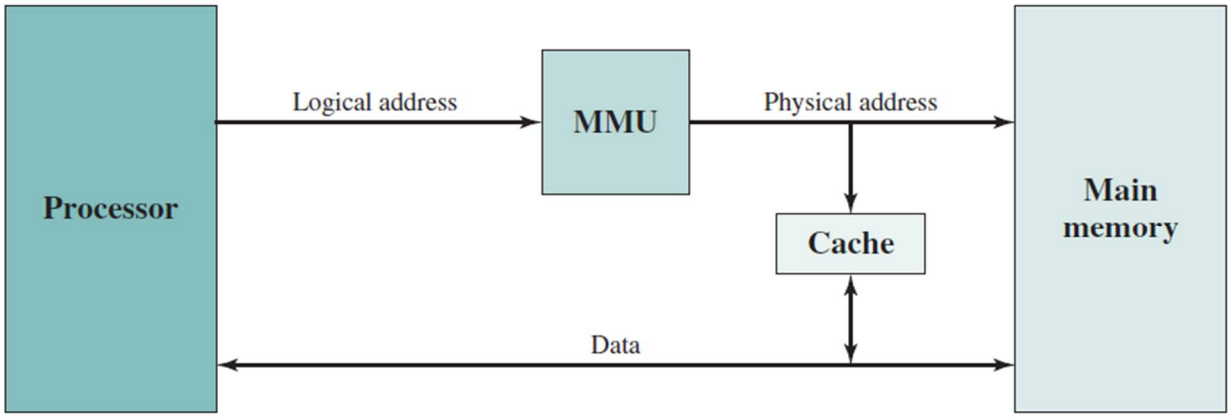
Cache Addresses	**Write Policy**
Logical	Write through
Physical	Write back
Cache Size	**Line Size**
Mapping Function	**Number of Caches**
Direct	Single or two level
Associative	Unified or split
Set associative	
Replacement Algorithm	
Least recently used (LRU)	
First in first out (FIFO)	
Least frequently used (LFU)	
Random	

❖ Cache Addresses

- ✓ **Logical cache** stores data using logical/virtual addresses.
  - ➢ Virtual memory is a facility that allows programs to address memory from a logical point of view, without regard to the amount of main memory physically available.
  - ➢ Memory management unit (MMU) translates each virtual address into a physical address in main memory.
- ✓ **Physical cache** stores data using main memory physical addresses



(a) Logical cache



(b) Physical cache

- ✓ One **advantage** of the logical cache is that cache access speed is faster than a physical cache, because the cache can respond before the MMU performs an address translation.

- ✓ The **disadvantage** logical cache design is complicated.
- ✓ Physical cache is simple but slow

❖ Cache Size

- ✓ We would like the size of the cache to be small enough so that the overall average cost per bit is close to that of main memory alone and large enough so that the overall average access time is close to that of the cache alone.

- ✓ The available chip and board area also limits cache size.
- ✓ Because the performance of the cache is very sensitive to the nature of the workload, it is impossible to arrive at a single "optimum" cache size.

- ✓ Table 4.3 lists the cache sizes of some current and past processors

Table 4.3 Cache Sizes of Some Processors

Processor	Type	Year of Introduction	L1 Cache[a]	L2 Cache	L3 Cache
IBM 360/85	Mainframe	1968	16–32 kB	—	—
PDP-11/70	Minicomputer	1975	1 kB	—	—
VAX 11/780	Minicomputer	1978	16 kB	—	—
IBM 3033	Mainframe	1978	64 kB	—	—
IBM 3090	Mainframe	1985	128–256 kB	—	—
Intel 80486	PC	1989	8 kB	—	—
Pentium	PC	1993	8 kB/8 kB	256–512 kB	—
PowerPC 601	PC	1993	32 kB	—	—
PowerPC 620	PC	1996	32 kB/32 kB	—	—
PowerPC G4	PC/server	1999	32 kB/32 kB	256 kB to 1 MB	2 MB
IBM S/390 G6	Mainframe	1999	256 kB	8 MB	—
Pentium 4	PC/server	2000	8 kB/8 kB	256 kB	—
IBM SP	High-end server/ supercomputer	2000	64 kB/32 kB	8 MB	—
CRAY MTA[b]	Supercomputer	2000	8 kB	2 MB	—
Itanium	PC/server	2001	16 kB/16 kB	96 kB	4 MB
Itanium 2	PC/server	2002	32 kB	256 kB	6 MB
IBM POWER5	High-end server	2003	64 kB	1.9 MB	36 MB
CRAY XD-1	Supercomputer	2004	64 kB/64 kB	1 MB	—
IBM POWER6	PC/server	2007	64 kB/64 kB	4 MB	32 MB
IBM z10	Mainframe	2008	64 kB/128 kB	3 MB	24–48 MB
Intel Core i7 EE 990	Workstation/ server	2011	6 × 32 kB/ 32 kB	1.5 MB	12 MB
IBM zEnterprise 196	Mainframe/ server	2011	24 × 64 kB/ 128 kB	24 × 1.5 MB	24 MB L3 192 MB L4

❖ Mapping Function

- ✓ Because there are fewer cache lines than main memory blocks. An algorithm is needed for mapping main memory blocks into cache lines.

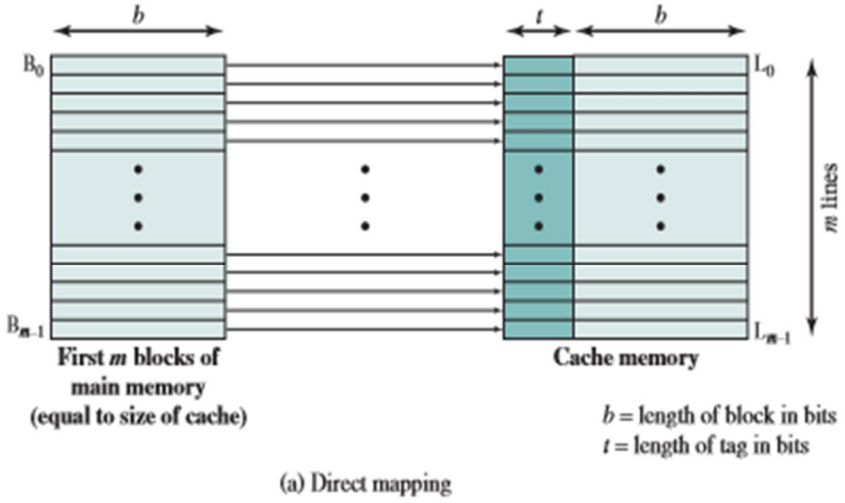
- ✓ Three techniques can be used:
  - ➢ Direct mapping.
  - ➢ Associative mapping.
  - ➢ Set associative mapping.

- ✓ **Example:**
  For all three cases, the example includes the following elements:
  - ➢ Cache of 64kByte
  - ➢ Cache block of 4 bytes.(transfer between cache and main memory)
    - i.e. cache is 16k (2^{14}) lines of 4 bytes (64k/4 = 2^{14})
  - ➢ 16MBytes main memory
  - ➢ 24 bit address
    - (2^{24}=16M)

Direct Mapping:

- ❖ The simplest technique, known as direct mapping
- ❖ Each block of main memory into only one possible cache line.
  - ✓ i.e. if a block is in cache, it must be in one specific place

- ❖ Following Figure (Figure 4.8a) shows the mapping for the first m blocks of main memory. Each block of main memory maps into one unique line of the cache



(a) Direct mapping

- ❖ Addressin 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)

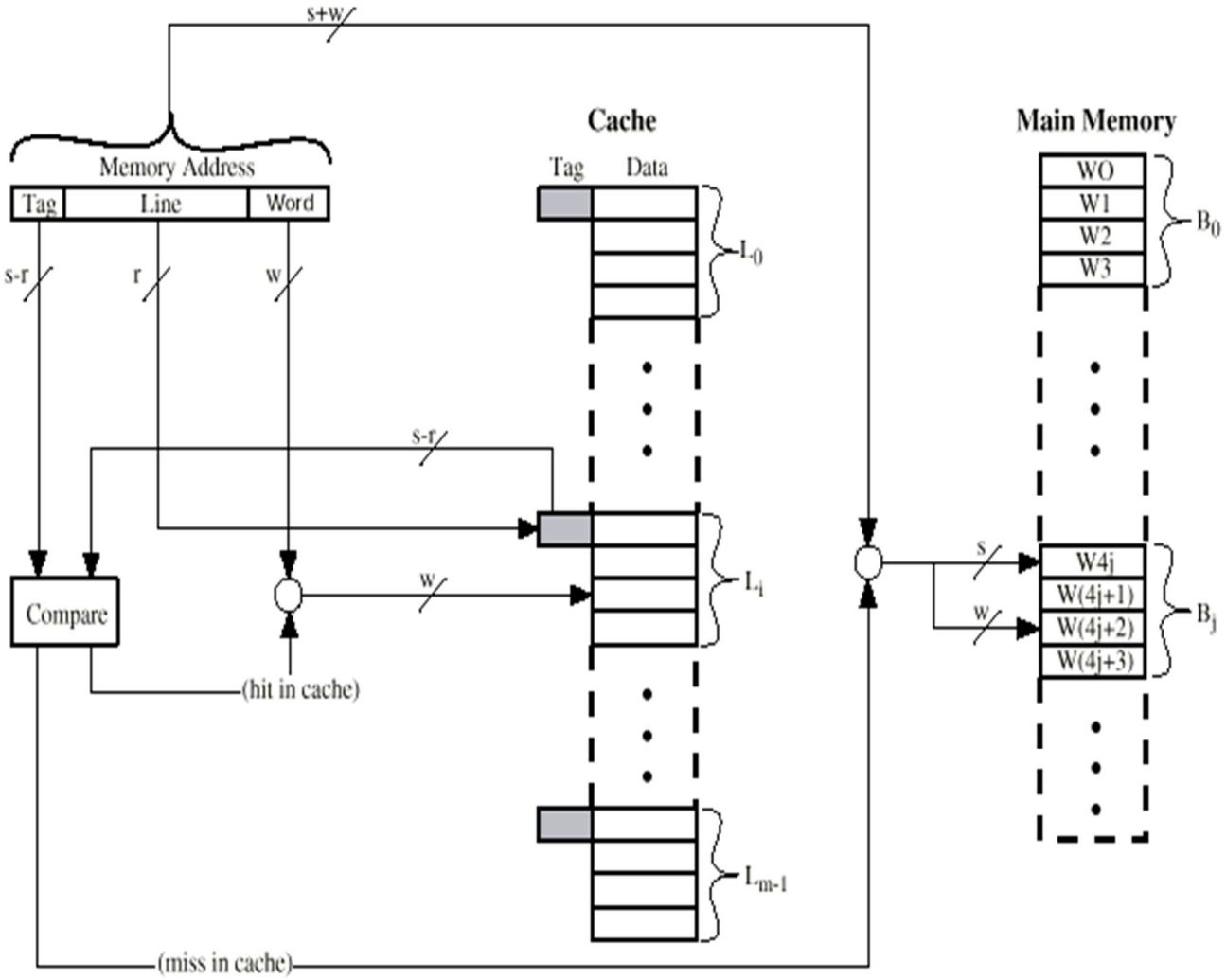
Tag s-r	Line or Slot r	Word w
8	14	2

- 24 bit address
- 2 bit word identifier (4 byte block)

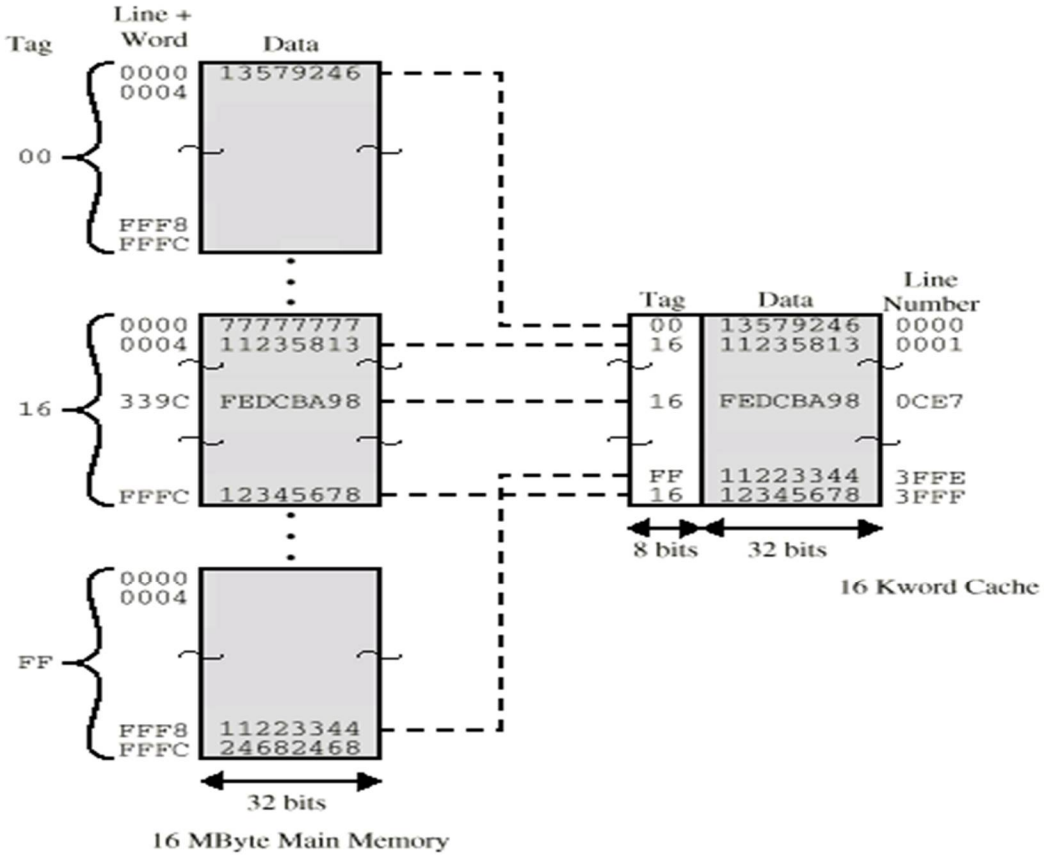
- 22 bit block identifier
  - ✓ 8 bit tag (=22-14)
  - ✓ 14 bit slot or line

- No two blocks in the same line have the same Tag field
- Check contents of cache by finding line and checking Tag

- Following Figure show Direct Mapping Cache Organization



❖ **Example:**



❖ **Direct Mapping summary:**

- ✓ Address length = (s + w) bits
- ✓ Number of addressable units = 2^{s+w} words or bytes
- ✓ Block size = line size = 2^{w} words or bytes
- ✓ Number of blocks in main memory = $2^{s+w}/2^{w} = 2^{s}$
- ✓ Number of lines in cache = m = 2^{r}
- ✓ Size of tag = (s – r) bits

❖ **Direct Mapping advantage:**

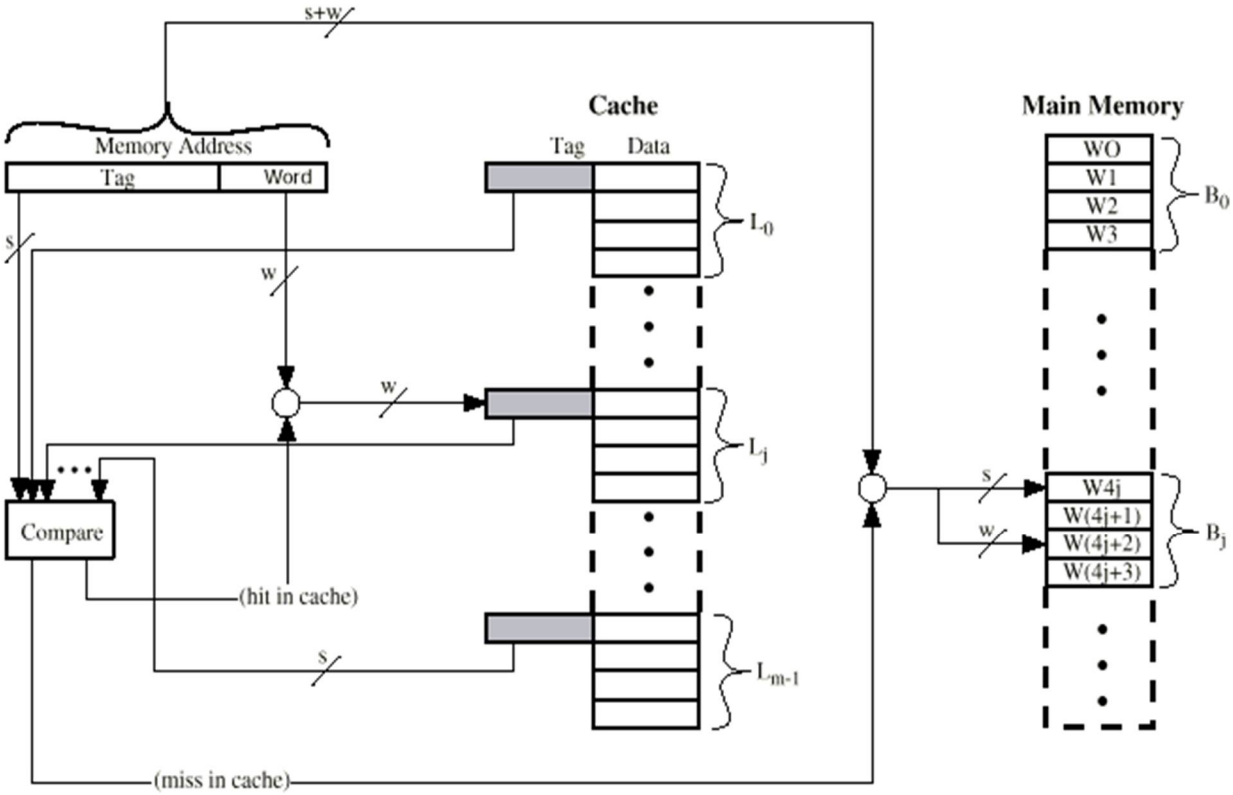
- ✓ Simple
- ✓ Inexpensive

❖ **Direct Mapping disadvantage:**

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

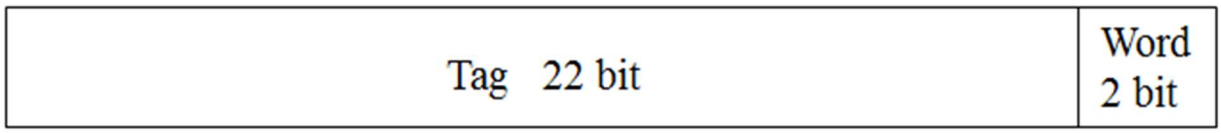
Associative Mapping:

- Associative mapping overcomes the disadvantage of direct mapping by permitting each main memory block to be loaded into any line of the cache
- Following figure (Figure 4.8b) show associative mapping:



- In this case, the cache control logic interprets a memory address simply as a Tag and a Word field.
- The Tag field uniquely identifies a block of main memory.
- Cache searching gets expensive

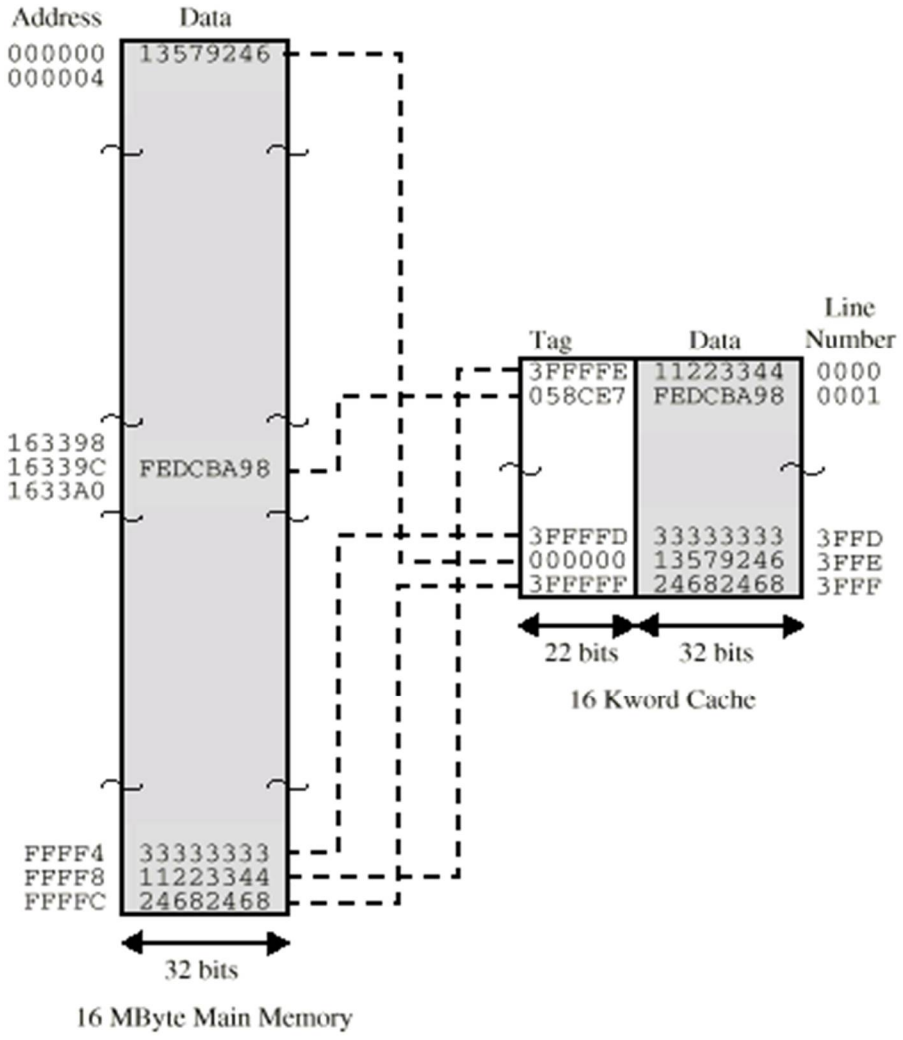
- Associative Mapping Address Structure:



- 22 bit tag stored with each 32 bit block of data
- Compare tag field with tag entry in cache to check for hit

- Least significant 2 bits of address identify which 16 bit word is required from 32 bit data block

❖ **Example:**



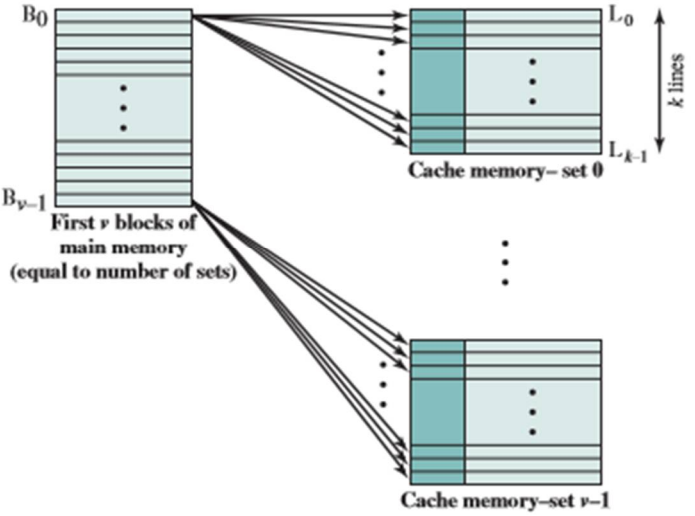
Address	Tag	Data	Cache line
FFFFC	3FFFFF	24682468	3FFF

❖ **Associative Mapping Summary:**

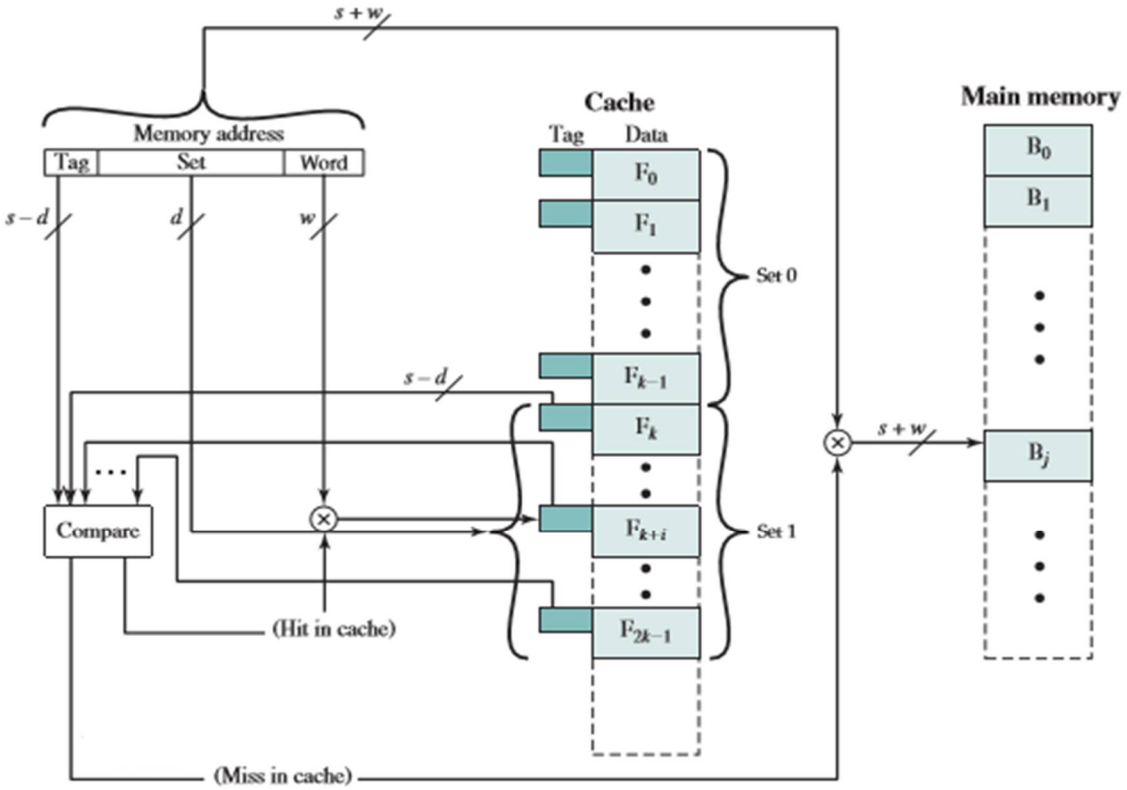
- ✓ Address length = (s + w) bits
- ✓ Number of addressable units = 2^{s+w} words or bytes
- ✓ Block size = line size = 2^{w} words or bytes
- ✓ Number of blocks in main memory = $2^{s+w}/2^{w} = 2^{s}$
- ✓ Number of lines in cache = undetermined
- ✓ Size of tag = s bits

Set Associative Mapping:

- Set-associative mapping is a compromise that exhibits the strengths of both the direct and associative approaches while reducing their disadvantages.
- Cache is divided into a number of sets. Each set contains a number of lines
- A given block maps to any line in a given set
  - ✓ e.g. Block B can be in any line of set i
- e.g. 2 lines per set
  - ✓ 2 way associative mapping
  - ✓ A given block can be in one of 2 lines in only one set



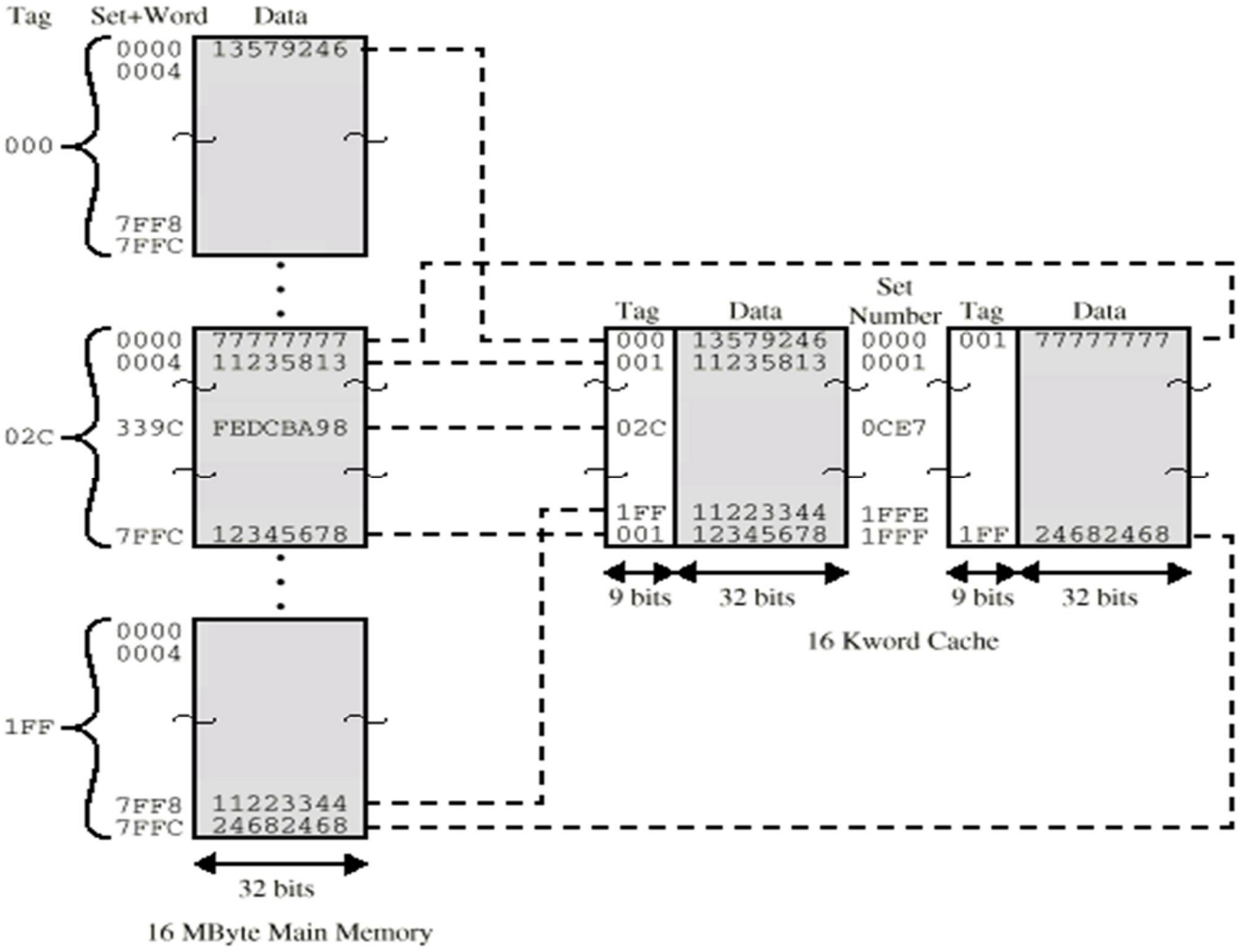
- Following figure (Figure 4.14) show set associative mapping:



- Set Associative Mapping Address Structure:

Tag 9 bit	Set 13 bit	Word 2 bit

- Use set field to determine cache set to look in
- Compare tag field to see if we have a hit
- e.g



Address	Tag	Data	Set number
1FF 7FFC	1FF	12345678	1FFF
001 7FFC	001	11223344	1FFF

❖ Replacement Algorithms:

- ✓ **Replacement Algorithms (1) Direct mapping:**
  - ➢ No choice
  - ➢ Each block only maps to one line
  - ➢ Replace that line

- ✓ **Replacement Algorithms (2) Associative & Set Associative:**
  Hardware implemented algorithm (speed):
  - ➢ Least Recently used (LRU)
    - Which of the 2 block is lru?
    - Update /Use-bit
  - ➢ First in first out (FIFO)
    - Replace block that has been in cache longest
  - ➢ Least frequently used
    - Replace block which has had fewest hits
    - Counters
  - ➢ Random

Summarize number(1):

Summarize the following elements of cache:

- ❖ Write Policy
  - ✓ Write through
  - ✓ Write back
- ❖ Line size
- ❖ Number of caches
  - ✓ Single or two level
  - ✓ Unified or split