Write operation

Write operation uses the same addressing technique as read operation. The tag part and data is presented to cache for updation. Cache write operation invariable follows a cache miss during read operation.

Address Mapping

When a tag address is presented to the cache, it must be quickly compared to the stored tags to determine whether a matching tag is currently assigned to the cache. The obvious approach of scanning all the tags in sequence is unacceptably slow. There are faster techniques for the same. They are,

1. Direct mapping cache
2. Fully associative cache
3. set associative cache.

Direct mapping cache : In this mapping each block of memory is mapped to a fixed slot of cache. For example if cache has four slots (sets) then the memory block 0 or 4 or 8 or 12 can be found in slot 0 (set 0), while 1 or 5 or 9 or 13 -- can be found in slot 1; 2 or 6 or 10 or 14 --- in slot 2 and 3 or 7 or 11 or 15 --- in slot 3. This can be mathematically defined as.

cache slot(set) number = [Block number of main memory] modulo operator [Total number of slots in cache]

In this technique, it can easily be determined whether a memory block is in cache or not

As shown in figure.

Number of main memory blocks = 16 (B_0 to B_{15})

Number of slots(sets) in cache memory = 4

Mapping of main memory block:

Suppose we want to find the cache memory set in which the block number B_{11} should be stored. This can be done by dividing 11 (Block Number) by 4 (Number of sets in cache memory).

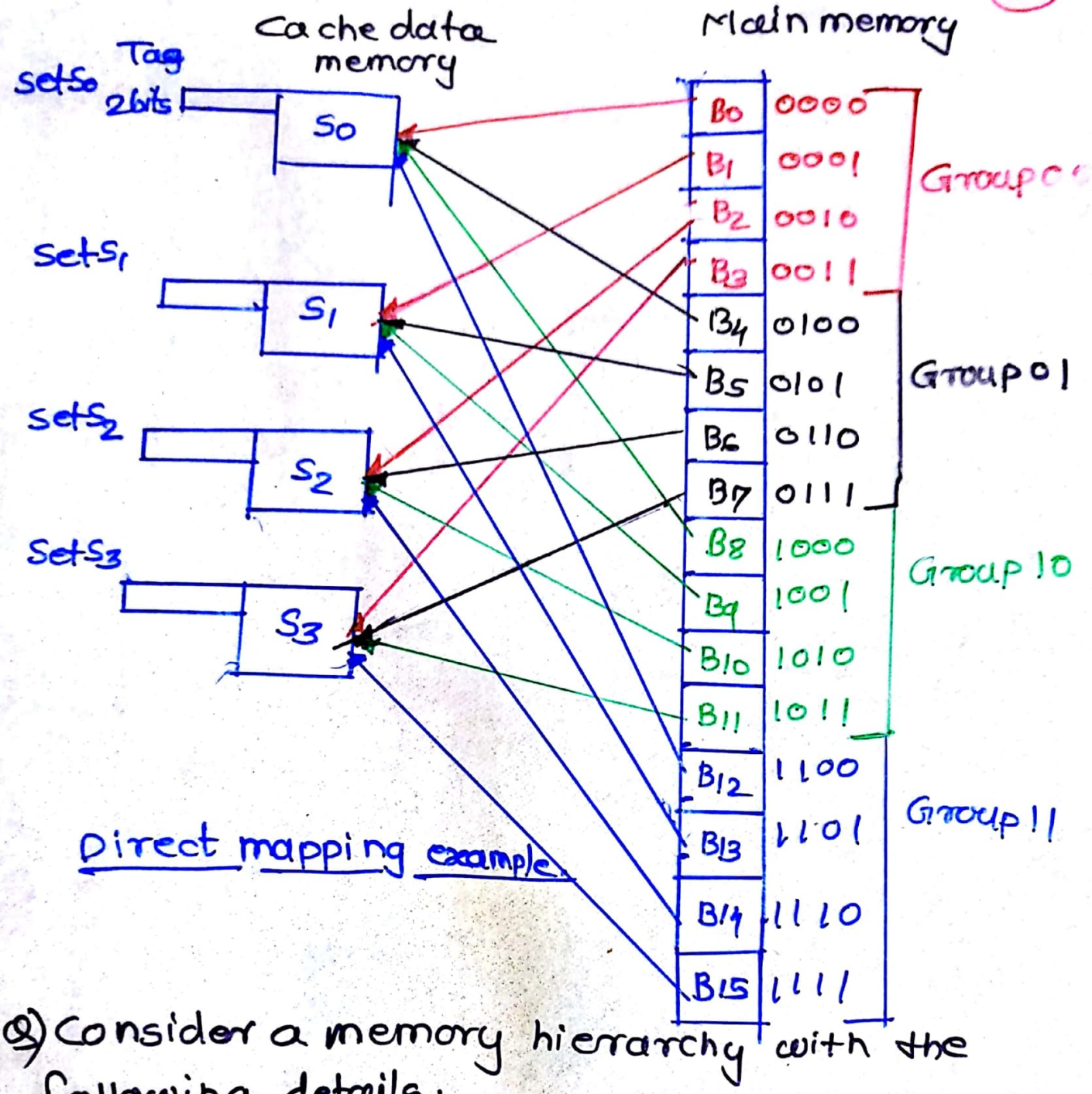
$\frac{11}{4} = 2\frac{3}{4}$, Quotient = 2, gives group number or tag value

Remainder = 3, gives the set number of cache data memory.

If the block is represented in binary then we can easily find the quotient and remainder.

$(11)_{10} = 10\ 11$

Quotient (Tag value) | Remainder (set number of cache)



Direct mapping example.

Q) Consider a memory hierarchy with the following details.

Number of main memory blocks = 16

" " cache " " " = 4

Block size = 4

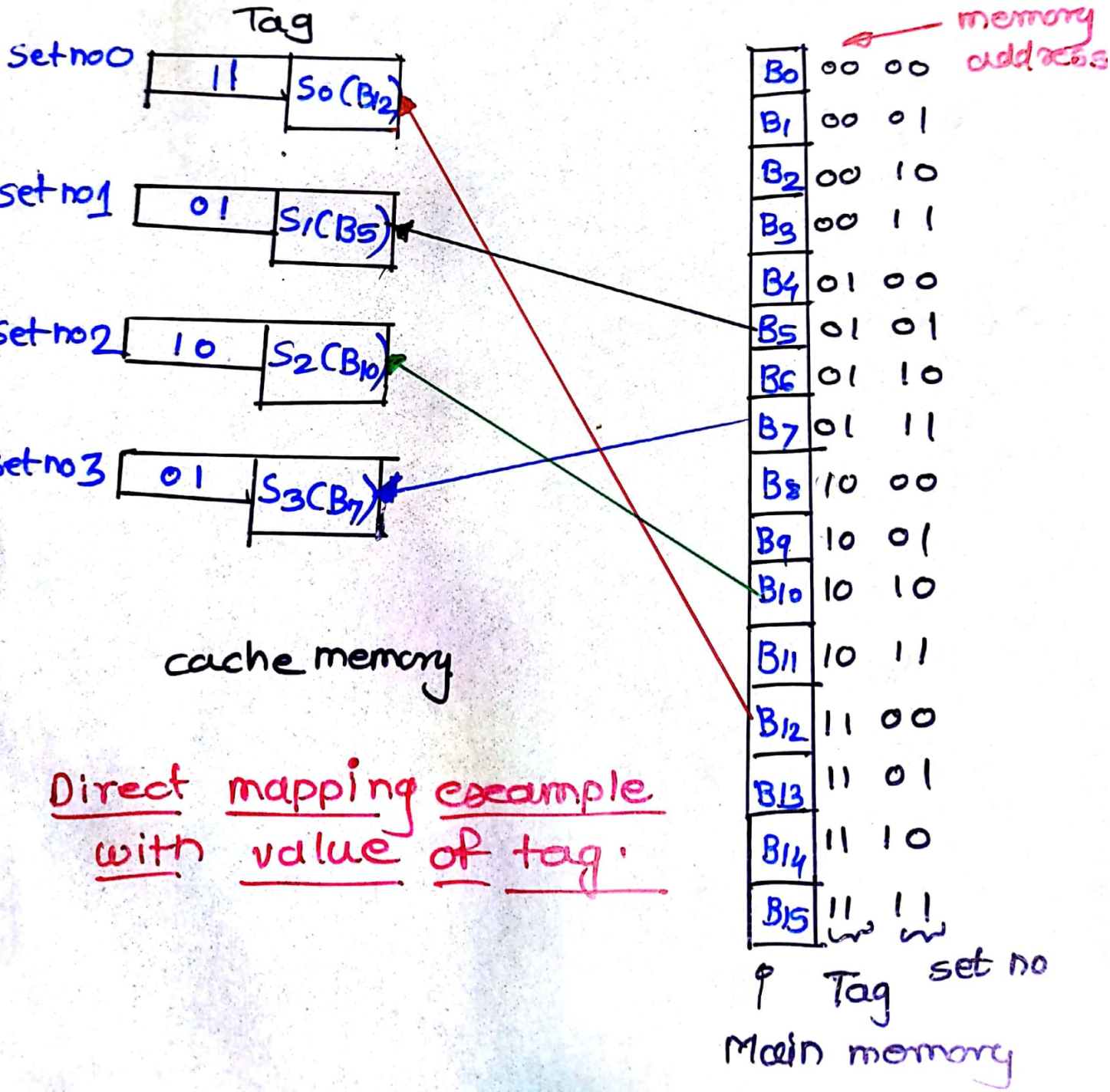
Direct mapping cache

Show the contents of tags if the following blocks are stored in cache memory B12, B7, B5, B10

solution: We must use 4 bit address for a block as there are 16 blocks in memory.

Block no	Block address in memory	Tag	Set no
12	11 00	11	00
7	01 11	01	11
5	01 01	01	01
10	10 10	10	10

Tag value — Remainder (set number)



Direct mapping example with value of tag.

consider a cache consisting of 256 blocks of 16 words each for a total of 4096 (4k) words and assume that the main memory is addressable by 16 bit address and it consists of 4k blocks How many bits are there in each of the TAG, BLOCK/SET and WORD fields for direct mapping cache.

Solution

i) Main memory size = 2^{16} bytes [main memory is addressable by a 16 bit address]

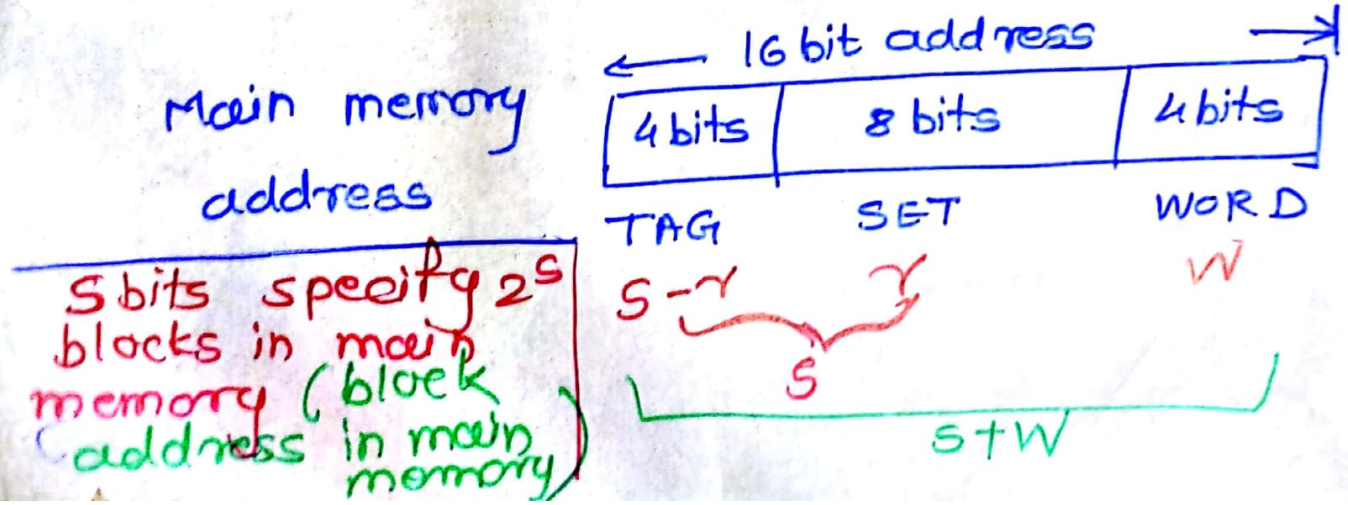
ii) Cache memory size = $256 \times 16 = 2^8 \times 2^4 = 2^{12}$

(256 → No of blocks, 16 → No of words in a block.)

$\frac{\text{Main memory size}}{\text{cache memory size}} = \frac{2^{16}}{2^{12}} = 2^4$, therefore the size of TAG in bits = 4

Since the block contains 16 (2^4) words – 4 bits are required to address a word.

Since the cache contains 256 (2^8) blocks, 8 bits are required to address a set.



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 = $\frac{2^{s+w}}{2^w} = 2^s$

Number of lines in cache (sets) = $m = 2^r$

size of tag = (s-r) bits.

s bits specify 2^s blocks of main memo

① Main memory = 16MB = $2^4 \times 2^{20}$ = 24 address lines (24 bit)

cache memory = 64kB = $2^6 \times 2^{10}$ = 16 bits

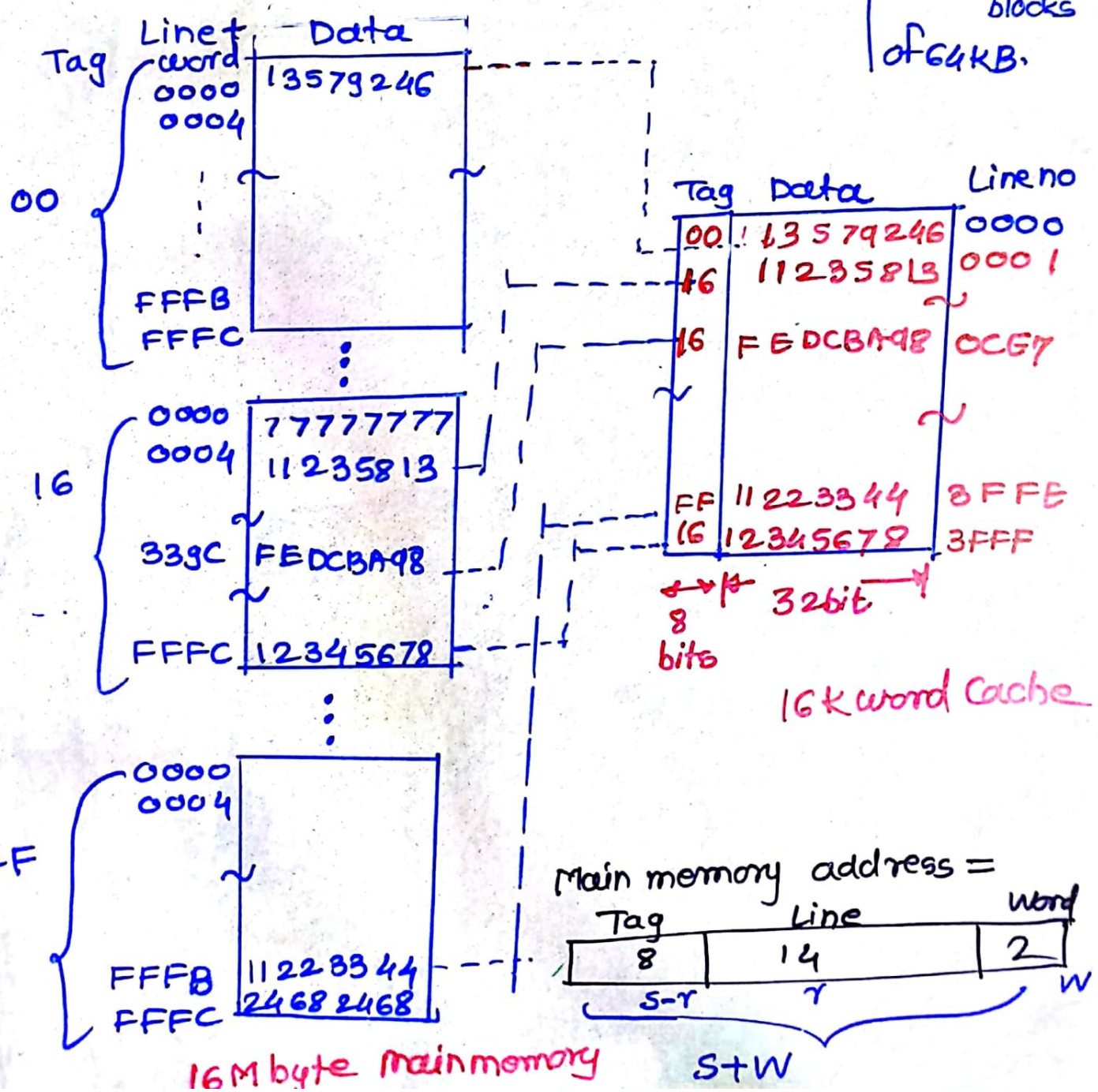
No of words in each block = 04 = 2 bits required to select a word

$$\frac{\text{Main memory size}}{\text{cache memory size}} = \frac{2^{24}}{2^{16}} = 2^8$$

∴ Tag size is of 8 bits.

to represent set/line = 14 bits.

Total memory is divided into $\frac{16MB}{64KB}$ = 256 blocks of 64KB.



Advantages of direct mapping

1. Simple to implement
2. Required normal SRAM

Drawback of direct mapping

If a program happens to reference words repeatedly from two different blocks that map into the same line, then the blocks will be continually swapped in the cache, and the hit ratio will be low (a phenomenon known as thrashing).

Associative Mapping

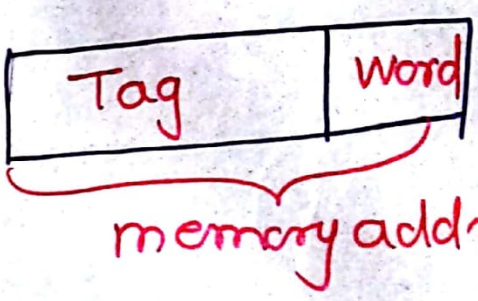
Associative mapping overcomes the disadvantage of direct mapping by permitting each main memory block to be loaded into any line/set of the cache. In this case the cache control logic interprets a memory address simply as a tag and a word field.

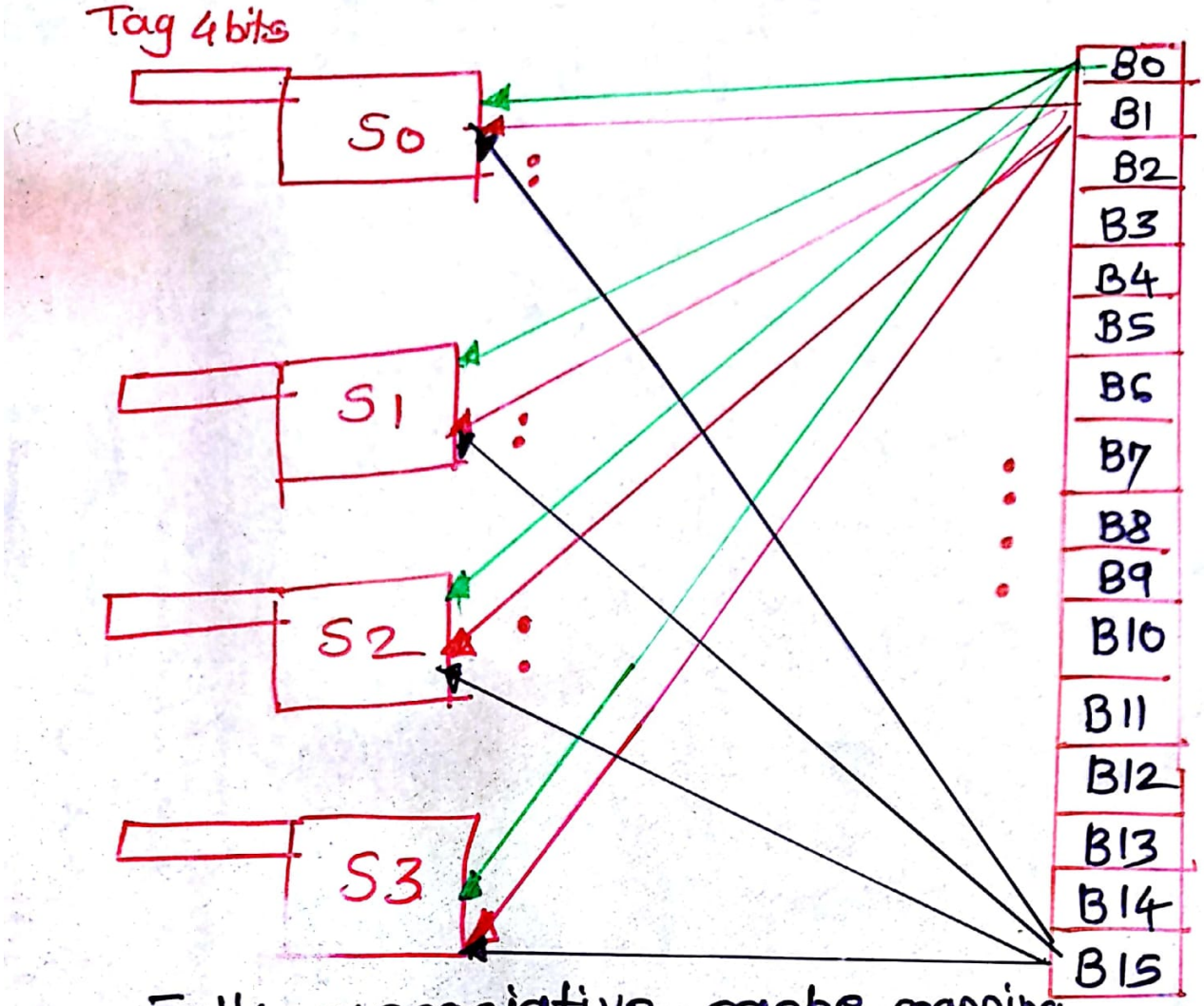
The tag field uniquely identifies a block of main memory. To determine whether the block is in the cache, the cache control logic must simultaneously examine every line's tag for a match.

Note that no field in the address corresponds to line number (set number), so that the number of lines in the cache is not determined by the address format.

At the start of memory access, the incoming tag is compared simultaneously to all the tags in the cache's tag memory. If a match (cache hit) occurs, a match indicating signal triggers the cache to service the requested memory access. A no-match signal identifies a cache miss and the memory access requested is forwarded to main memory for service. A cache block containing the target address is then sent from main memory to the cache.

Each block of main memory can be placed in any of the available cache block frame (slot, set). Because of this, an S-bit tag is needed in each cache block, where S is the number of bits required to address a main memory block.





Fully associative cache mapping,

Every block is mapped to any of the four block frames identified by the tag.

Advantages: 1) Very high hit Ratio.
2) Allows, better block replacement strategy with reduced contention

Disadvantages: 1) Associative memory is very costly
2) Significant increase in tag length
3) Scheme is comparatively complex.

Consider a cache consisting of 16 words. Each block consists of 4 words. Size of main memory is 256 bytes. Find number of bits in each of TAG, Block/SET and WORD for associative mapped cache?

Main memory = 256 bytes = 2^8

address bits = 8

Set Associative Cache

Set associative cache tries to take advantage of both direct mapping and associative mapping.

(1) Direct mapped cache requires normal SRAM.

(2) Associative mapping offers high hit-ratio set associative cache is implemented using normal SRAM and provides a high hit ratio.

If properly designed, this cache may offer the best performance-cost ratio. Most high performance computer system are based on this.

(3) In a K associative cache, the m cache blocks are divided into

$$v = m/k \text{ sets.}$$

(m = no of frames / lines in cache)

(4) Each set contains k blocks.

(5) A memory block is mapped to a set using direct mapping.

(6) If each set is identified by d bits then

$$2^d = v$$

(7) If s bits are required to access a main memory block then tag length

$$\text{tag length} = s - d$$

can be stored

(8) A block of main memory in any of the s frames inside the addressed set (associative mapping)

Example: Memory consists of n = 16 blocks
Two way set associative cache is used
k = 2

∴ v = 4 (Number of sets in cache)
m = 8 (Number of frames in memory)

Tag size = $\log_2 \frac{\text{Number of blocks in main memory}}{\text{Number of sets in cache memory}}$

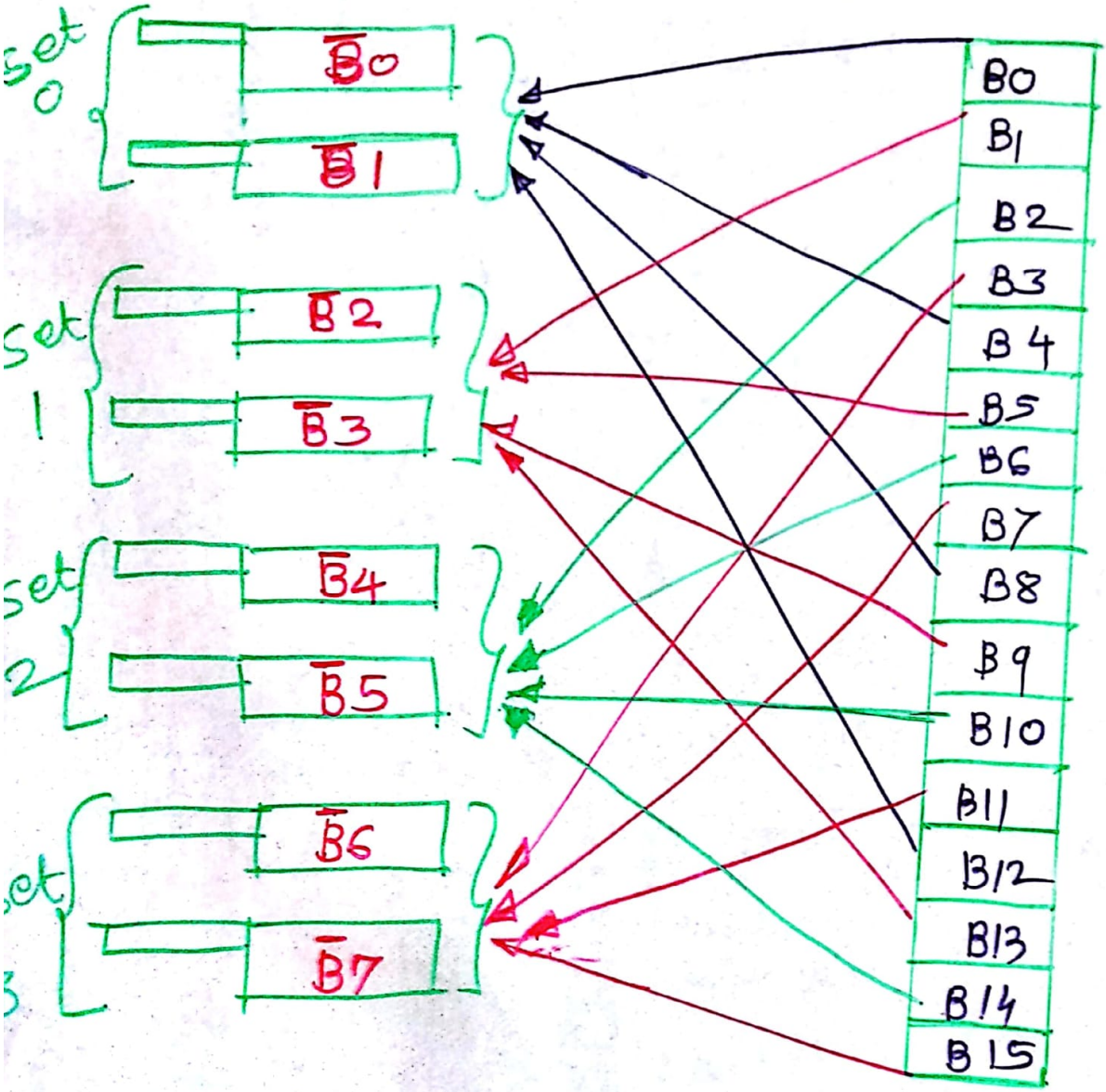
$= \log_2 \frac{16}{4} = \log_2 4 = 2$ bits.

Set length (No of bits to address a set)
$= \log_2$ (Number of sets in cache) $= \log_2 4$
$= 2$ bits.

Word length $= \log_2$ (Number of words in a block)

$= \log_2 4 = 2$ bits.

Tag	set	word
2 bit	2bit	2 bit



Mapping cache blocks in a two-way associative cache with Four set.

Design a 2 way set associative memory with the following details.
Cache consists of 32 words.
Each block consists of 4 words.
Size of the main memory = 256 words.
Find the number of bits in each of TAG, BLOCK/SET and word.

Solution

(1) No of bits in main memory address
$= \log_2$ (Size of main memory)
$= \log_2 256 = \log_2 2^8 = 8$ bits

(2) No of bits required to address main memory blocks $= \log_2$ (No of blocks in main memory)
$= \log_2 \left(\frac{256}{4}\right) = \log_2 64$
$= 6$ bits $= s$

(3) Number of sets in cache memory
$v = \frac{32}{4 \times 2} = 4$
words per block, 2 way

(4) No of bits required to address a set
$= \log_2 4 = 2 = d$

(5) Tag Size $= s - d = 6 - 2 = 4$

4 bit Tag	2 bits set	2 bits Word

comparision b/n associative & set associative mapping

- Set associative mapping is a ~~comparison~~ compromise b/n direct and associative mapping
- Set associative mapping preserves the strength of both associative and direct mapping. It also reduces their disadvantages
- Associative mapping requires content addressable memory which is very costly
- Set associative mapping can be implemented using normal SRAM.
- To reduce the contention among the blocks, getting mapped to same set of cache. A set consists of k blocks in a k-way set associative mapping. Inside a set, a mapped block can be stored in any of the frames of the set
- Mapping inside the set in a cache can be considered as associative mapping
- Associative mapping offers high hit ratio set-11- is a compromise b/n the hit ratio of direct and associative mapping
- cost of set associative mapping ≈ cost of direct mapping
- hit ratio of set associative mapping ≈ hit ratio of associative mapping