**2-Fully Associative Mapping:** According to this technique, an incoming main memory block can be placed in any available cache block. Therefore, the address issued by the processor need only have two fields. These are the ***Tag*** and ***Word*** fields. The first uniquely identifies the block while residing in the cache. The second field identifies the element within the block that is requested by the processor.



1. **Word** field = log2 B, where B is the size of the block in words

2. **Tag** field = log2 M, where M is the size of the main memory in blocks

3. The **number of bits in the main memory address** = log2 (B x M)

the tags are stored in an **associative** **memory** (content addressable). This allows the entire contents of the tag memory to be searched in parallel (associatively), hence the name, associative mapping.

The main **advantage** of the associative-mapping technique is the efficient use of the cache. This stems from the fact that there exists no restriction on where to place incoming main memory blocks. Any unoccupied cache block can potentially be used to receive those incoming main memory blocks.

The main **disadvantage** of the technique, is the hardware overhead required to perform the associative search conducted in order to find a match between the tag field and the tag memory as discussed above.

**Example 3**: Compute the above three parameters for a memory system having the following specification: size of the main memory is **4K** blocks, size of the cache is **128** blocks, and the block size is **16** words. Assume that the system uses associative mapping:

Word field = log2 B = log2 16 = log2 24 = 4 bits

Tag field = log2 M = log2 27 x 210 = 12 bits

The number of bits in the main memory address = log2 (B x M) =

 log2  (24 x 212) = 16 bits.