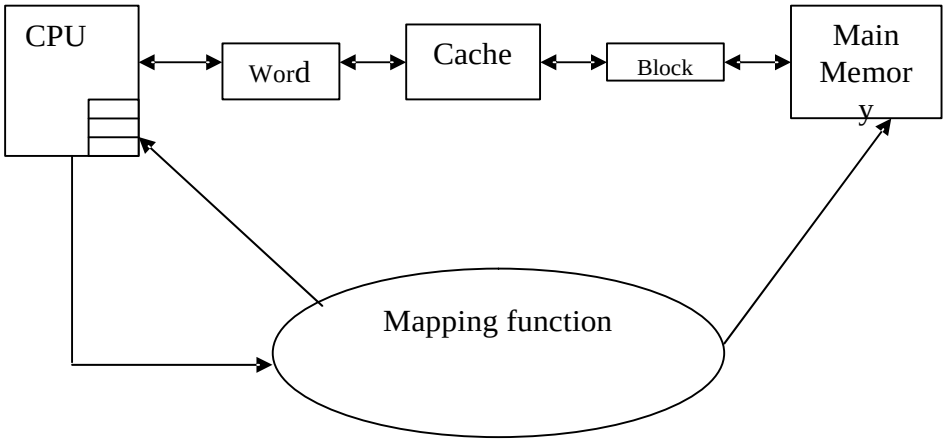
ASSOCIATIVE MAPPED CACHES

The Cache:- The Mapping Function

Fig. Shows a schematic view of the cache mapping function. The mapping function is responsible for all cache operations. It is implemented in hardware, because of the required high speed operation. The mapping function determines the following.

- Placement strategies – Where to place an incoming block in the cache
- Replacement strategies – Which block to replace when a cache miss occurs
- How to handle Read and Writes up as cache hit and mises



Three different types of mapping functions are in common use

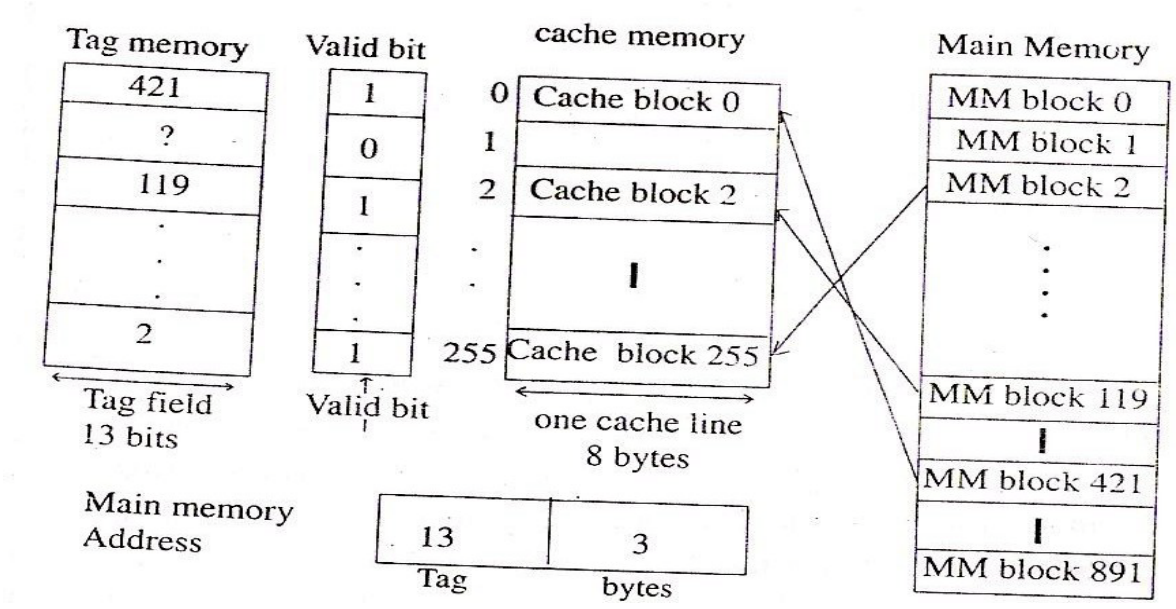
1. Associative mapping cache
2. Direct mapping cache
3. Block-set-associative mapping cache

1. Associative mapped caches:- In this any block from main memory can be placed any where in the cache. After being placed in the cache, a given block is identified uniquely by its main memory block number, referred to as the tag, which is stored inside a separate tag memory in the cache.

Regardless of the kind of cache, a given block in the cache may or may not contain valid information. For example, when the system has just been powered up add before the cache has had any blocks loaded into it, all the information there is invalid. The cache maintains a valid bit for each block to keep track of whether the information in the corresponding block is valid.

Fig4. shows the various memory structures in an associative cache, The cache itself contain 256, 8byte blocks, a 256 x 13 bit tag memory for holding the tags of the blocks currently stored in the cache, and a 256 x 1 bit memory for storing the valid bits, Main memory contains 8192, 8 byte blocks. The figure indicates that main memory address references are partition into two fields, a 3 bit word field describing the location of the desired word in the cache line, and a 13 bit tag field describing the main memory block number desired. The 3 bit word field becomes essentially a “cache address” specifying where to find the word if indeed it is in the cache.

The remaining 13 bits must be compared against every 13 bit tag in the tag memory to see if the desired word is present.



In the fig, above, main memory block 2 has been stored in the 256 cache block and so the 256th tag entry is 2 mm block 119 has been stored in the second cache block

corresponding entry in tag memory is 119 mm block 421 has been stored in cache block 0 and tag memory location 0 has been set to 421. Three valid bits have also been set, indicating valid information in these locations.

The associative cache makes the most flexible and complete use of its capacity, storing blocks wherever it needs to, but there is a penalty to be paid for this flexibility the tag memory must be searched in for each memory reference.

Fig.5 shows the mechanism of operation of the associative cache. The process begins with the main memory address being placed in the argument register of the (associative) tag memory (1) if there is a match (hit), (2) and if the ratio bit for the block is set (3), then the block is gated out of the cache (4), and the 3 bit offset field is used to select the byte corresponding to the block offset field of the main memory address (5) That byte is forwarded to the CPU, (6)

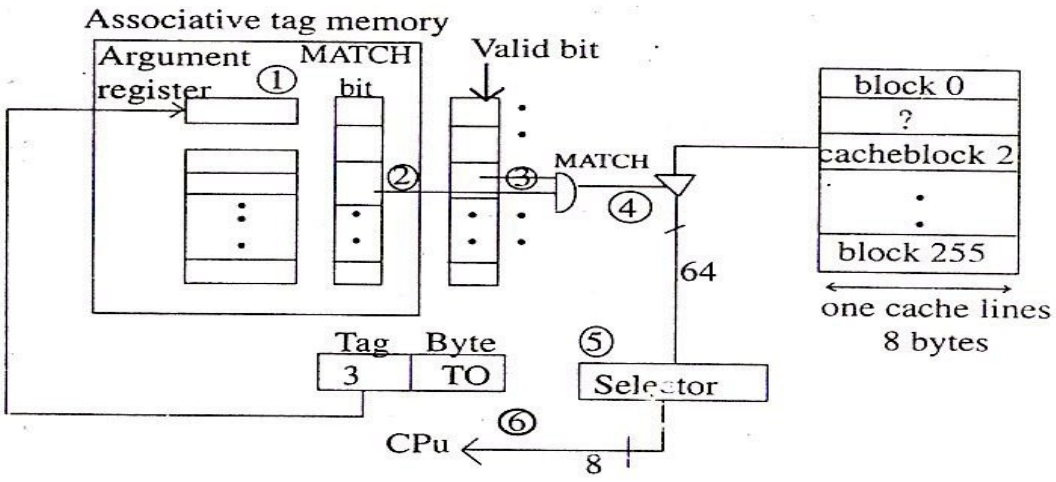


Fig. 5 Associative cache mechanism:

Source : http://elearningatria.files.wordpress.com/2013/10/cse-iv-computer-organization-10cs46-notes.pdf