Direct Mapping Example
- Assume the cache size is 64 KB. Data are transferred between main memory and the cache in blocks of 4 bytes each. This means that the cache is organized as $16K = 2^{14}$ lines of 4 bytes each. A word is 8-bit long. [Cache size = 64 KB, Block size = 4 B, number of lines $2^{14}$, Word length = 1B]

- The main memory consists of 16 MB, with each byte directly addressable by a 24-bit address ($2^{24} = 16$ M). Thus, for mapping purposes, we can consider main memory to consist of $4M$ blocks of 4 bytes each. [Main memory size = 16 MB, address = 24 bits, numbers of blocks = 4M]

- So, if using direct mapping, the above example would have the address in the following format.

| Tag: 8 bits | Line: 14 bits | Word: 2 bits |

- How many bits for w? [2, as the block size is 4 bytes and each word is a byte, so need 2 bit to specify the 4 words.]

- How many bits for r? [14, as the cache has $2^{14}$ number of lines]

- How many bits for tag? [8, as the address is 24-bit long, $24 - 2 - 14 = 8$]

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

Pros & 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.
**Associative Mapping**

- To provide more flexibility, associative mapping allows a main memory block to be loaded into any line of cache.
- Its address structure has two fields:

  ![Diagram]

  - The rightmost \( w \) bits are the word position within a block.
  - The leftmost \( s \) bits are used to identify which block is stored in a particular cache line.
  - The way to check for hit is
    - compare the “tag” field of the target address with the “tag” of every line of the cache.
    - if a cache line has the same “tag”, use the “word” field of the target address to find the target word.
    - otherwise, the target word is missed in the cache, and will need to use the target address to search in the main memory and replace a block in the cache with the block where the target word is in.

- If we use the same example we used in Direct Mapping, what does the address structure look like if using associative mapping?
  - Cache size: 64 KB; block size: 4 bytes; addressable unit: byte
  - Main memory size: 16 MB; address length: 24 bits
• How many bits for w? [2, as the block size is 4 bytes and each word is a byte, so need 2 bit to specify the 4 words.]
• How many bits for the tag? [22, as 24 - 2 = 22]

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 block in main memory = \(\frac{2^{(s+w)}}{s^w}\) = \(2^s\)
• Number of lines in cache = undetermined
• Size of tag = s bits

Pros & Cons
• A block can load to any line of cache
• Every line’s tag must be examined for a match
• Cache searching gets expensive and slow

Set-Associative Mapping
• A comprise that exhibits the strengths of direct mapping (simple, inexpensive) and associative mapping (flexibility that blocks can be loaded to any lines) while reducing their disadvantages (fixed location for a given block - high cache miss ratio, examine every line’s tase for a match- cache searching is expensive and slow).

• It introduces a new concept cache set.
• Cache is divided into a number of sets, \(v\)
• Each set contains \(k\) lines
• \(k\) lines in a set is called a \(k\)-way set associative mapping
• Number of lines in a cache, \(m = v \times k\)
• The idea of set-associative mapping is that a block is always mapped to a specific cache set if it's swapped into the cache. But, it can be loaded into any line of that cache set.

• Way to calculate the cache set number of a block $i = j \mod v$
  where
  $i =$ cache set number
  $j =$ main memory block number
  $m =$ number of lines in the cache
  $v =$ number of sets
  $k =$ number of lines in each set