Cache Overview
- Cache is a small amount of fast memory.
- If CPU requests contents, it will check cache for this data.
- If the data present in the cache, CPU get from the cache.
- Otherwise, one of the two things will happen depends on how the CPU and cache are connected.
  - If the cache is physically between CPU and data bus, the required block will be read from main memory to cache and then deliver from cache to CPU.
  - If CPU and cache both receive data from the same data bus buffer, the required block will be read from main memory to cache and simultaneously delivered to CPU.

Cache Structure
- Main memory consists of $2^n$ addressable words, each word has a unique $n$-bit address.
- Main memory contains $M$ blocks, Each block has $K$ words. So, there are $M = \frac{2^n}{K}$ blocks in main memory.
- Cache contains $C$ lines of blocks, each has a tag uniquely identifying the block of $K$ words.
- $C \ll M$
- If a word in a block of memory is read, that block is transferred to one of the lines of the cache
Cache Design Goal and Tradeoffs

- Cache is to help improve the average access time, which is achieved by improving the hit ratio, letting requested data presented in the fast memory.

- However, it is not the more the better.
  - Cost: more caches mean more expensive
  - Speed:
    - More cache is faster up to a point.
    - Beyond that point, larger circuits will be more complex and slow up the cache.
    - An algorithm is needed for mapping main memory blocks to lines in the cache. Beyond that point, the time it takes to look into the cache is not significantly less than to look into the RAM directly.
  - So the right amount of cache and appropriate algorithm lead to the good performance.

Mapping Functions

- A mapping function is the method used to locate a word.
- To locate a word, the address associated with the word is used.
- Mapping functions define the address structure and how to use that structure to locate a word.
- It is used when copying a block from main memory to the cache and it is used again when trying to retrieve data from the cache.

Direct Mapping

- A straightforward way is to map each block of main memory to only one cache line. It says if a block is in a cache, it will always be found in the same place once it's been swapped into the cache.

<table>
<thead>
<tr>
<th>Cache line</th>
<th>Main memory blocks assigned</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0, m, 2m, …, 2^ℓ − m</td>
</tr>
<tr>
<td>1</td>
<td>1, m + 1, 2m + 1, …, 2^ℓ − m + 1</td>
</tr>
<tr>
<td>…</td>
<td>…</td>
</tr>
<tr>
<td>m − 1</td>
<td>m − 1, 2m − 1, 3m − 1, …, 2^ℓ − 1</td>
</tr>
</tbody>
</table>

- Line number is calculated using the following function
  \[ i = j \mod m \]
  where
  \[ i = \text{cache line number} \]
  \[ j = \text{main memory block number} \]
  \[ m = \text{number of lines in the cache} \]
Address structure for direct mapping is

```
<table>
<thead>
<tr>
<th>s - r bits</th>
<th>r bits</th>
<th>w bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>Line: Bits identifying line in cache</td>
<td>Word: Bits identifying word offset into block</td>
</tr>
</tbody>
</table>
```

- Each main memory address can be divided into three fields
  - Rightmost $w$ bits identify a unique word within a block
  - Remaining bits specify which block in memory. These bits are divided into two fields:
    - Rightmost $r$ bits of these $s$ bits identities which line in the cache
    - Leftmost $s-r$ bits uniquely identifies the block within a line of cache, as a line can be used by multiple blocks.
  - Please note that this tag field is the tag uniquely identify a block, as no two blocks in the same line have the the same tag.

- The way Direct Mapping uses to check contents of cache is:
  - use the “line” field of the target address to find the particular line in cache
  - compare the “tag” field of the address with the “tag” of that particular cache line
  - if the two “tags” are the same, use the “word” field to find the target word
  - otherwise, the target word is missed in 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.