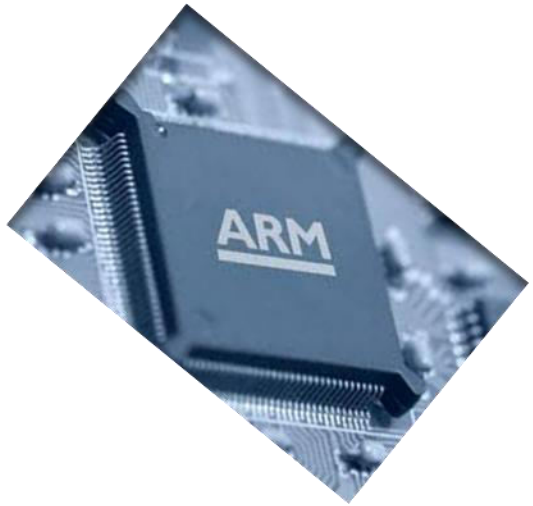
MICROPROCESSOR AND COMPUTER ARCHITECTURE
CACHE MEMORY
AN INTRODUCTION



Credits:
MPCA Team

CACHE MEMORIES

- **Processor is much faster than the main memory.**
  - As a result, the processor has to spend much of its time waiting while instructions and data are being fetched from the main memory.
  - Major obstacle towards achieving good performance.

CACHE MEMORIES

- **Speed of the main memory cannot be increased beyond a certain point.**

- Cache memory is an architectural arrangement which makes the main memory appear faster to the processor than it really is.

- Cache memory is based on the property of computer programs known as "**locality of reference**".

LOCALITY OF REFERENCE

- Analysis of programs indicates that many instructions in localized areas of a program are executed repeatedly during some period of time, while the others are accessed relatively less frequently.
  - These instructions may be the ones in a loop, nested loop or few procedures calling each other repeatedly.
  - This is called "locality of reference".

LOCALITY OF REFERENCE

- Temporal locality of reference:
  - Recently executed instruction is likely to be executed again very soon.
  - Whenever an instruction or data is needed for the first time, it should be brought into a cache. It will hopefully be used again repeatedly.

LOCALITY OF REFERENCE

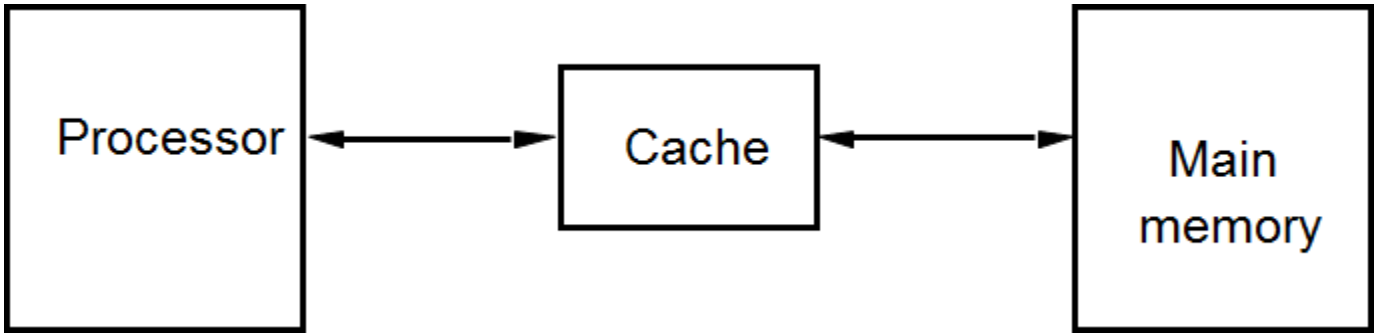
- Spatial locality of reference:

  - Instructions with addresses close to a recently instruction are likely to be executed soon.

  - Instead of fetching just one item from the main memory to the cache at a time, several items that have addresses adjacent to the item being fetched may be useful.

  - The term “block” refers to a set of contiguous addresses locations of some size.

CACHE MEMORIES



- Cache is the name given to the highest or first level of memory hierarchy encountered after the address leaves the processor.
- Processor issues a Read request, a block / line run, of words is transferred from the main memory to the cache, one word at a time.

CACHE MEMORIES

- Subsequent references to the data in this block of words are found in the cache.
- At any given time, only some blocks in the main memory are held in the cache.
- Which blocks in the main memory are in the cache is determined by a **"mapping function".**
- When the cache is full, and a block of words needs to be transferred from the main memory, some block of words in the cache must be replaced.
  This is determined by a **"replacement algorithm".**

CACHE MEMORIES

- A bit called as “valid bit” is provided for each block.
- If the block contains valid data, then the bit is set to 1, else it is 0.
- Valid bits are set to 0, when the power is just turned on.

When a block is loaded into the cache for the first time, the valid bit is set to 1.

- Data transfers between main memory and disk occur directly bypassing the cache.
- When the data on a disk changes, the main memory block is also updated.
- However, if the data is also resident in the cache, then the valid bit is set to 0.

CACHE MEMORIES

- Existence of a cache is transparent to the processor. The processor issues Read and Write requests in the same manner.
- **If the data is in the cache it is called a Read or Write hit that is [CACHE HIT].**
- **Read hit**
  - The data is obtained from the cache.

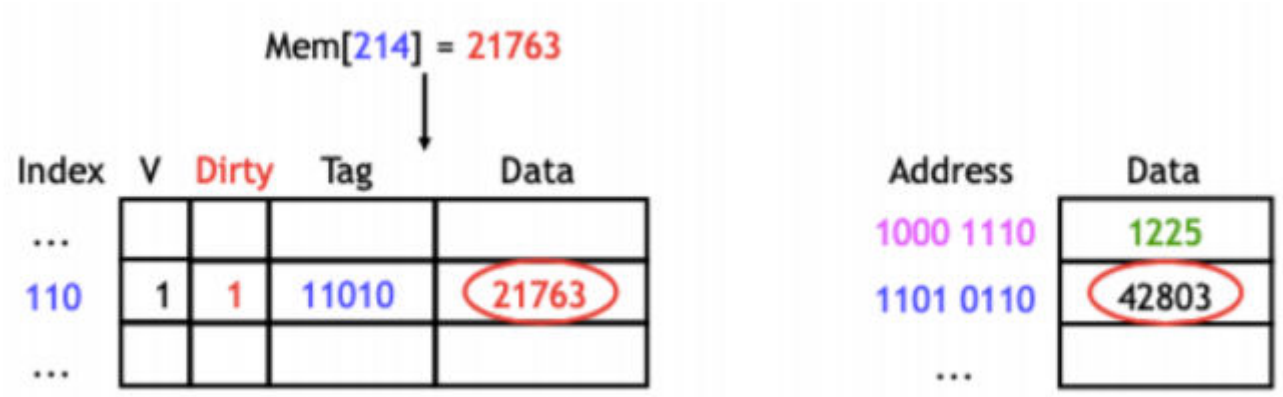
Write Hit

- Cache is a replica of the contents of the main memory.
  - Contents of the cache and the main memory may be updated simultaneously. This is the **write-through protocol**. This is simple, and keeps the cache and main memory consistent.

  The main drawback of Writethrough protocol is the increase in latency.

Write Hit

- Update the contents of the cache, and mark it as updated by setting a bit known as the **dirty bit or modified** bit. The contents of the main memory are updated only when this block is replaced. This is **write-back or copy-back** protocol.



Write Hit

Advantages of Write-back protocol

- The advantage of write-back caches is that not all write operations need to access main memory.
- If a single address is frequently written to, then it doesn't have to keep writing that data through to main memory. (Latency avoided)
- If several bytes within the same cache block are modified, they will only force one memory write operation at write-back time.

CACHE MEMORIES

- If the data is not present in the cache, then a **Read miss or Write miss** occurs, **[CACHE MISS].**

- Time required for a cache miss depends on both latency & bandwidth.

- **Latency :** Time required to retrieve the first word of the block.

- **Bandwidth:** Time required to retrieve the rest of the block.

CACHE MEMORIES

Read miss:

- Block of words containing this requested word is transferred from the memory.
- After the block is transferred, the desired word is forwarded to the processor.
- The desired word may also be forwarded to the processor as soon as it is transferred without waiting for the entire block to be transferred.
- This is called **load-through or early-restart.**

CACHE MEMORIES

Write-miss:

- If the write-through protocol is used, then the contents of the main memory are updated directly.

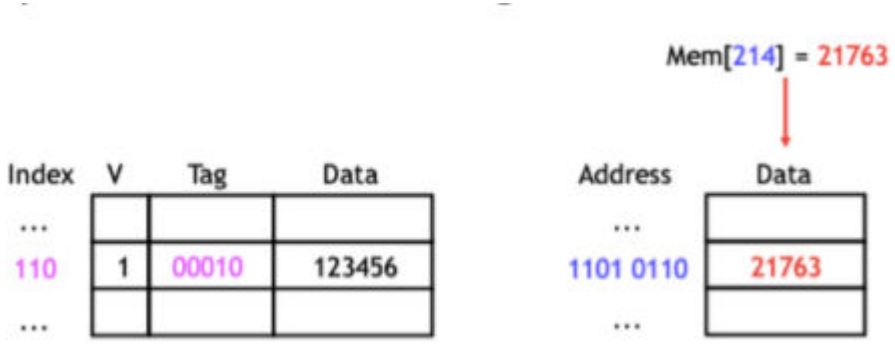
- If write-back protocol is used, the block containing the addressed word is first brought into the cache. The desired word is overwritten with new information.

Write Miss

Write NoAllocate:

With a write no allocate policy, the write operation goes directly to main memory without affecting the cache.

This is good when data is written but not immediately used again, in which case there's no point to load it into the cache yet.

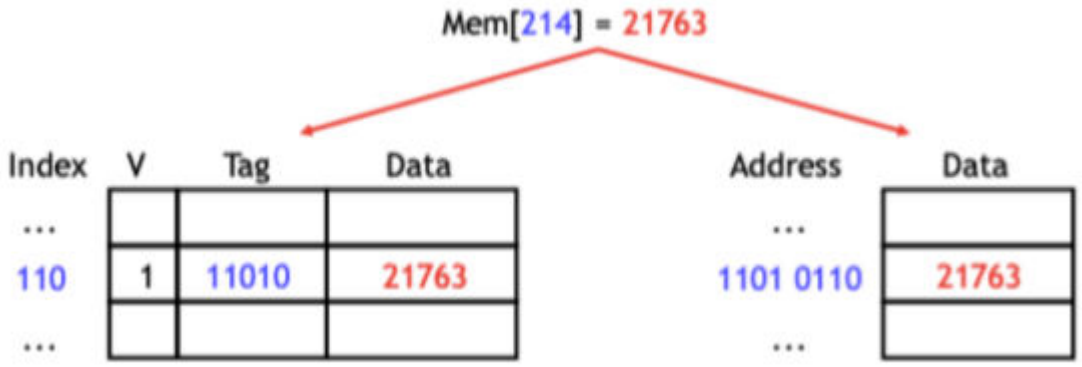


Write Miss

Write Allocate:

An allocate on write strategy would instead load the newly written data into the cache.

If that data is needed again soon, it will be available in the cache.



CACHE MEMORIES

What happens if the data in the disk and main memory changes and the write-back protocol is being used?

- In this case, the data in the cache may also have changed and is indicated by the dirty bit.
- The copies of the data in the cache, and the main memory are different. This is called the **cache coherence problem.**
- One option is to force a write-back before the main memory is updated from the disk.

CACHE MEMORIES – WRITE BUFFER

Write-through:

- Each write operation involves writing to the main memory.
- If the processor has to wait for the write operation to be complete, it slows down the processor.
- Processor does not depend on the results of the write operation.
- Write buffer can be included for temporary storage of write requests.
- Processor places each write request into the buffer and continues execution.
- If a subsequent Read request references data which is still in the write buffer, then this data is referenced in the write buffer.

CACHE MEMORIES – WRITE BUFFER

Write-back:

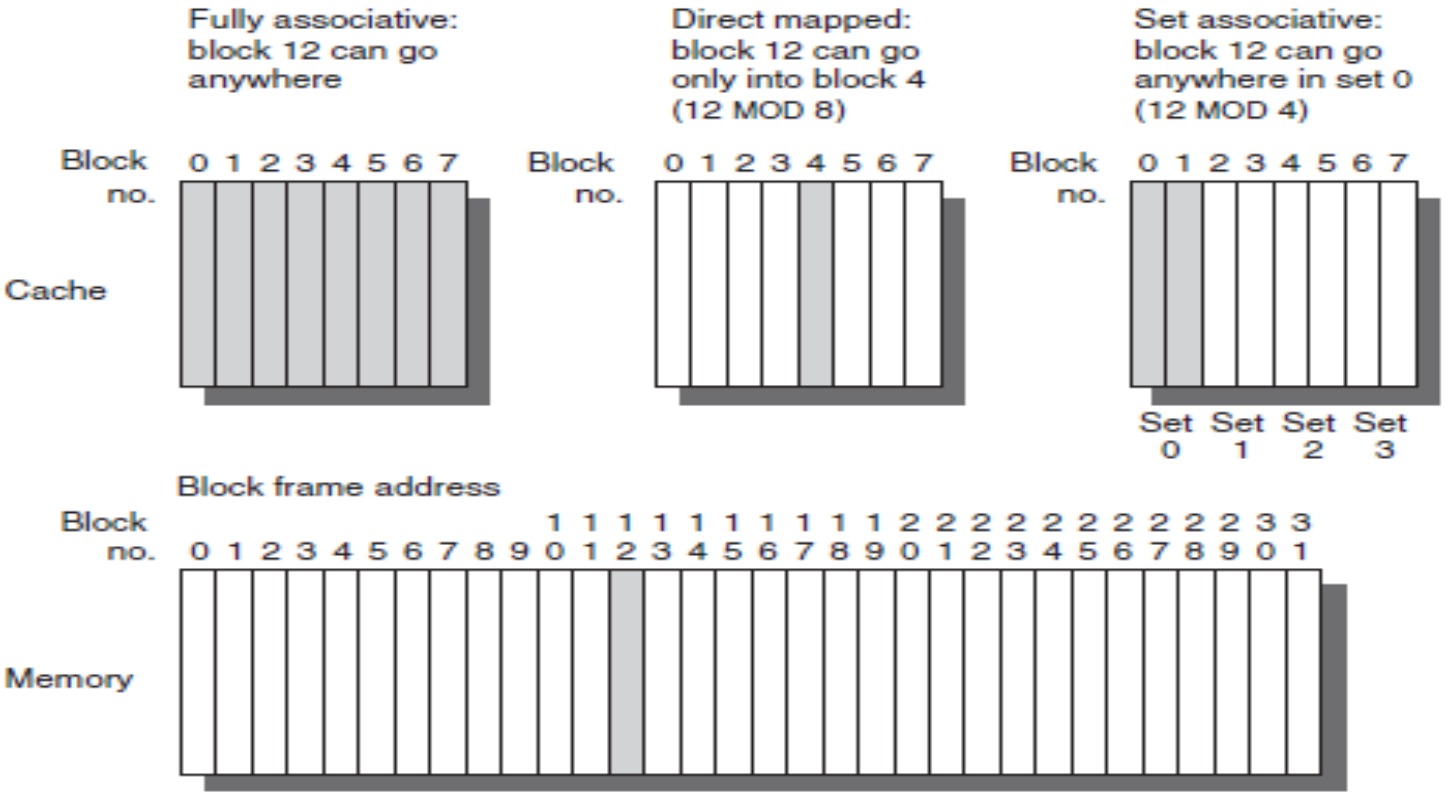
- Block is written back to the main memory when it is replaced.
- If the processor waits for this write to complete, before reading the new block, it is slowed down.
- Fast write buffer can hold the block to be written, and the new block can be read first.

MAPPING FUNCTIONS

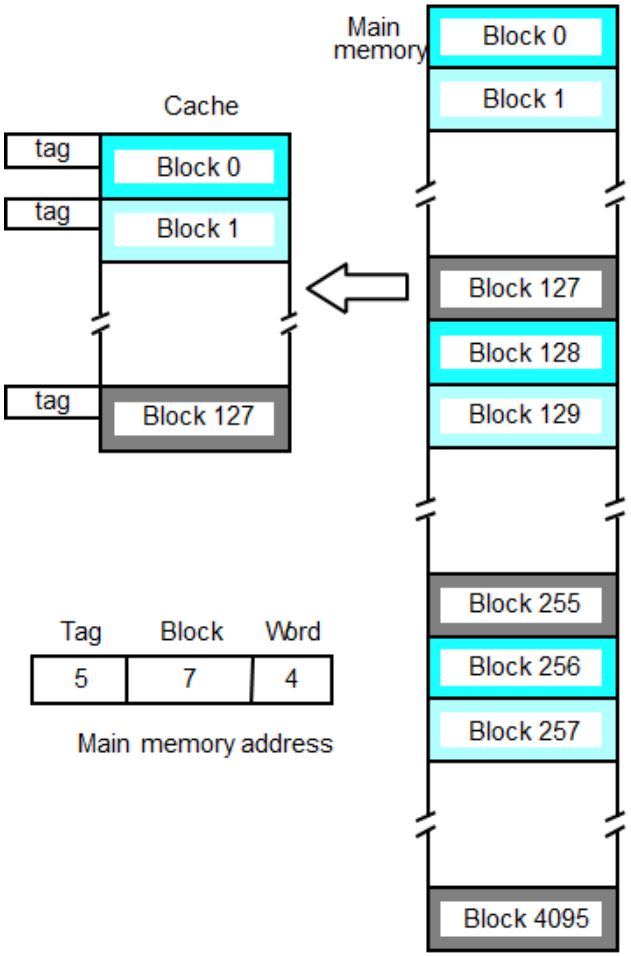
- Mapping functions determine how memory blocks are placed in the cache.
- **A simple processor example:**
  - Cache consisting of 128 blocks of 16 words each.
  - Total size of cache is 2048 (2K) words.
  - Main memory is addressable by a 16-bit address.
  - Main memory has 64K words.
  - Main memory has 4K blocks of 16 words each.
  - Consecutive addresses refer to consecutive words.
- **Three mapping functions:**
  - **Direct mapping**
  - **Associative mapping**
  - **Set-associative mapping**.

MAPPING FUNCTIONS

- Three mapping functions

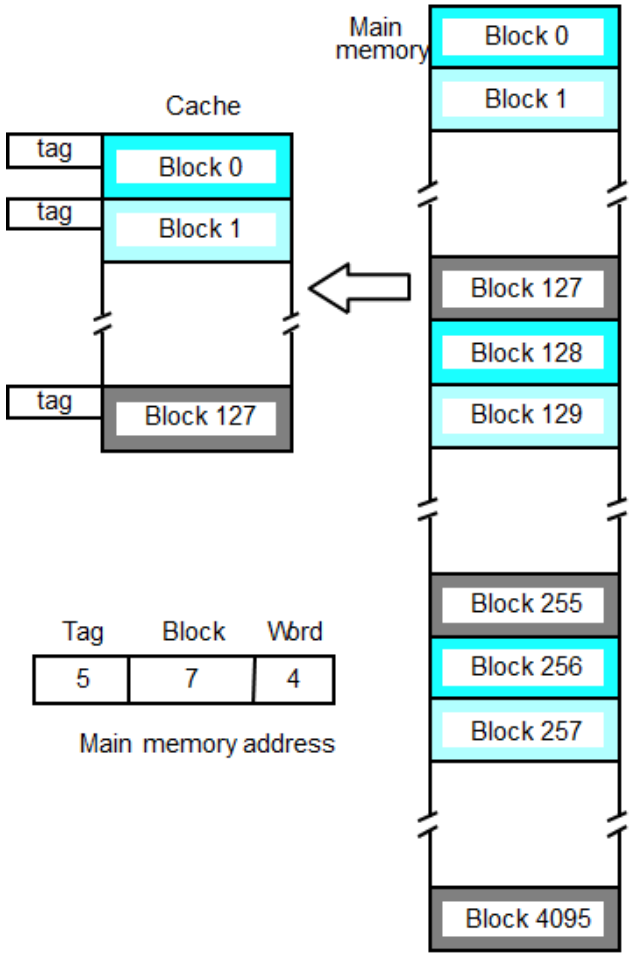


DIRECT MAPPING



- Block j of the main memory maps to j modulo 128 of the cache. 0 maps to 0, 129 maps to 1.
- More than one memory block is mapped onto the same position in the cache.
- May lead to contention for cache blocks even if the cache is not full.
- Resolve the contention by allowing new block to replace the old block, leading to a trivial replacement algorithm.

DIRECT MAPPING

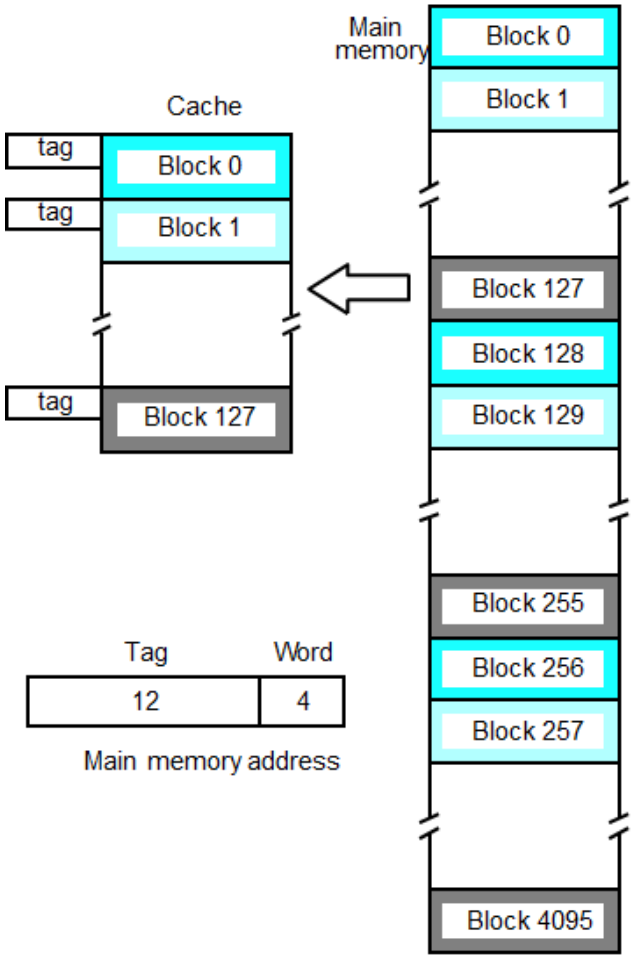


Memory address is divided into three fields:

- Low order 4 bits determine one of the 16 words in a block.
- When a new block is brought into the cache, the next 7 bits determine which cache block this new block is placed in.
- High order 5 bits determine which of the possible 32 blocks is currently present in the cache. These are tag bits.

- Simple to implement but not very flexible.

ASSOCIATIVE MAPPING

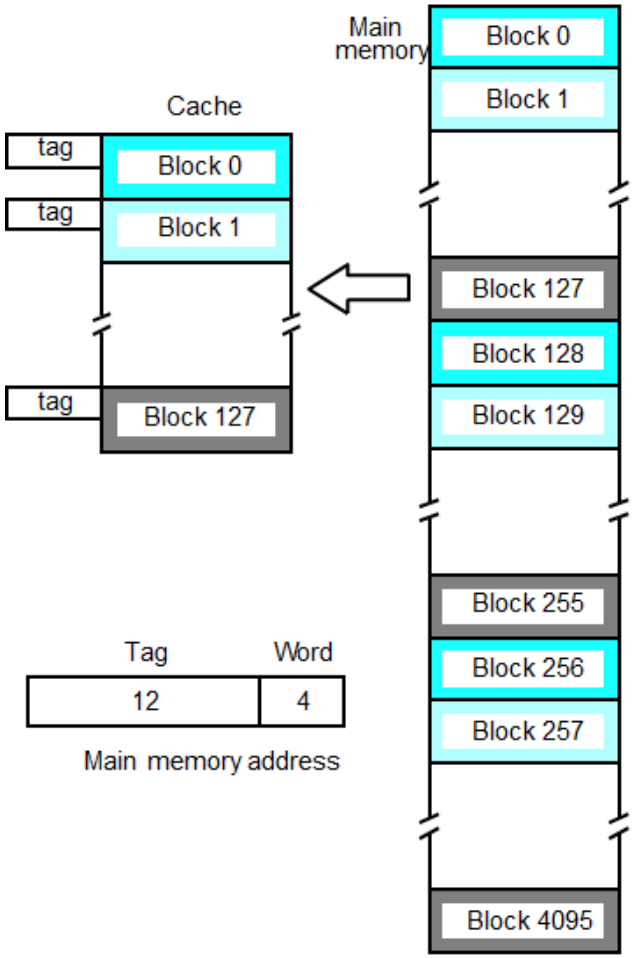


- Main memory block can be placed into any cache position.

- Memory address is divided into two fields:
  - Low order 4 bits identify the word within a block.
  - High order 12 bits or tag bits identify a memory block when it is resident in the cache.

- Flexible, and uses cache space efficiently.

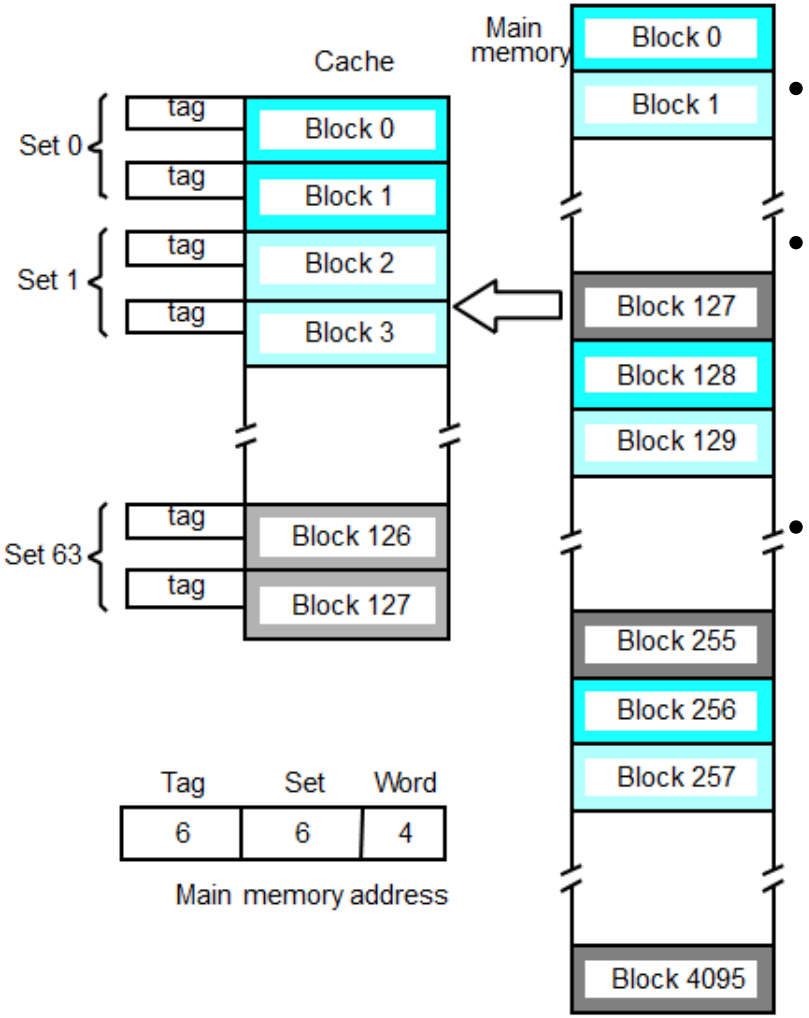
ASSOCIATIVE MAPPING



- Replacement algorithms can be used to replace an existing block in the cache when the cache is full.

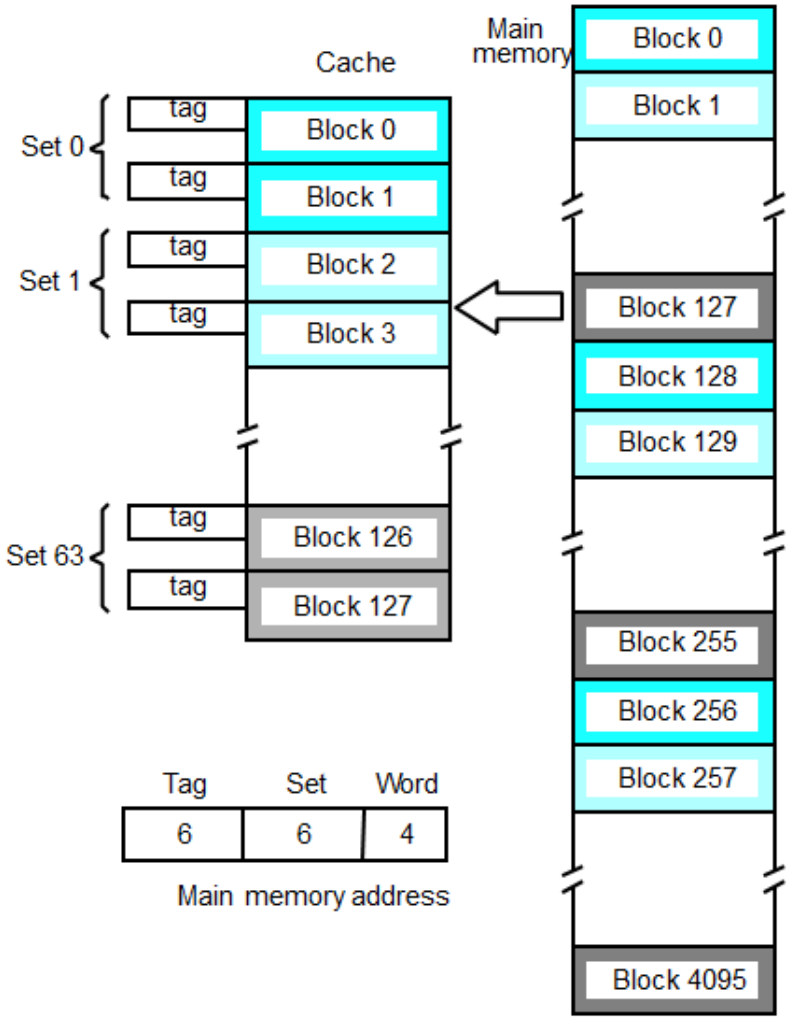
- Cost is higher than direct-mapped cache because of the need to search all 128 patterns to determine whether a given block is in the cache.

SET-ASSOCIATIVE MAPPING



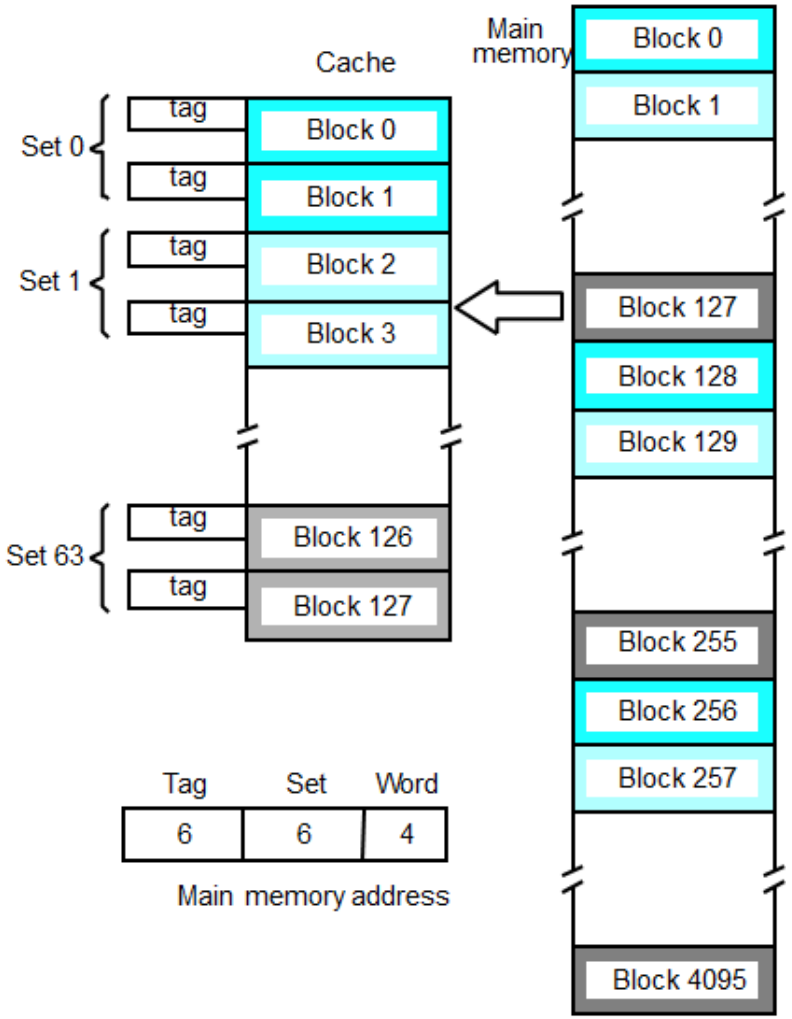
- Blocks of cache are grouped into sets.
- Mapping function allows a block of the main memory to reside in any block of a specific set.
- Divide the cache into 64 sets, with two blocks per set. Memory block 0, 64, 128 etc. map to block 0, and they can occupy either of the two positions.

SET-ASSOCIATIVE MAPPING



- Memory address is divided into three fields:
  - 6 bit field determines the set number.
  - High order 6 bit fields are compared to the tag fields of the two blocks in a set.
- Set-associative mapping combination of direct and associative mapping.

SET-ASSOCIATIVE MAPPING



- Number of blocks per set is a design parameter.
  - One extreme is to have all the blocks in one set, requiring no set bits (fully associative mapping).
  - Other extreme is to have one block per set, is the same as direct mapping.

Examples:

- Consider a direct mapped cache with 8 cache blocks (0-7). If the memory block requests are in the order4, 3, 25, 8, 19, 6, 25, 8, 16, 35, 45, 22, 8, 3, 16, 25, 7, which of the memory blocks will be present in the cache at the end of the sequence? Also, calculate the hit ratio and miss ratio.

- Consider a 2-way SA mapped cache with 8 cache blocks (0-7) with LRU replacement policy. If the memory block requests are in the order 4, 3, 25, 8, 19, 6, 25, 8, 16, 35, 45, 22, 8, 3, 16, 25, 7, which of the memory blocks will be present in the cache at the end of the sequence? Also, calculate the hit ratio and miss ratio.

REPLACEMENT ALGORITHMS

- Direct-mapped cache, the position that each memory block occupies in the cache is fixed. As a result, the replacement strategy is trivial.
- Associative and set-associative mapping provide some flexibility in deciding which memory block occupies which cache block.
- When a new block is to be transferred to the cache, and all the positions it may occupy are full, which block in the cache should be replaced?

❑ When a miss occurs, cache controller must select a block to be replaced with the desired data.

REPLACEMENT ALGORITHMS

❑ There are three primary strategies employed for selecting which block to replace:

- **Random**
- **Least Recently Used [LRU]**
- **First In First Out [FIFO]**
- **Least Frequently Used [LFU]**

Random:

To spread allocation uniformly, candidate blocks are randomly selected.

First In First Out:

Other replacement algorithms which include removing the “oldest” block, disregard the locality of reference principle and do not perform as well.

Least Frequently Used:

Replace the block that has had the fewest hits.

REPLACEMENT ALGORITHMS

Least Recently Used (LRU):

- Locality of reference suggests that it may be okay to replace the block that has gone the longest time without being referenced.
- This block is called Least Recently Used (LRU) block, and the replacement strategy is called LRU replacement algorithm.
- LRU algorithm has been used extensively.
    - It provides poor performance in some cases.
    - Performance of the LRU algorithm may be improved by introducing a small amount of randomness into which block might be replaced.

Note: Random is simple to build in the hardware. LRU becomes expensive as the number of blocks increases.

MEMORY HIERARCHY QUESTIONS

Q1: Where can a block be placed in the upper level ?

- Block Placement

Q2: How is a block found if it is in the upper level?

- Block Identification.

Q3: Which block should be replaced on a miss?

- Block Replacement.

Q4: What happens on a write ?

- Write Strategy.

Q & A
Cache Memory