Cache Memory

Introduction:

Cache memory is fast memory that is used to hold the most recently accessed data in slower main memory. The idea is that frequently accessed data will stay in cache, which allows the CPU to access it more quickly, which means it doesn't have to wait for the data to arrive.

Cache Memory is the Processor's internal quick-hand storage that it uses for things that it's currently processing at that given time. As with most things, the more cache memory a processor has, it will usually run smoother and faster than one with less of about the same operating frequency.

Cache Memory

Locality of Reference

- the references to memory *tend to be confined within a few localized areas* in memory. Ex. Program loops or subroutine.It states that over a short interval of time the addresses generated by a typical program refers to a few localized areas of memory repeatedly.

Cache Memory : **a fast small memory**

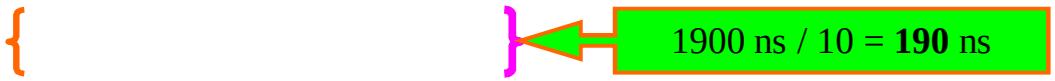
- keeping the most frequently accessed instructions and data in the fast cache memory

Cache

- cache size : 256 K byte (512 K byte)
- mapping method : 1) associative, 2) direct, 3) set-associative
- replace algorithm : 1) LRU, 2) LFU, 3) FIFO
- write policy : 1) write-through, 2) write-back

Hit Ratio

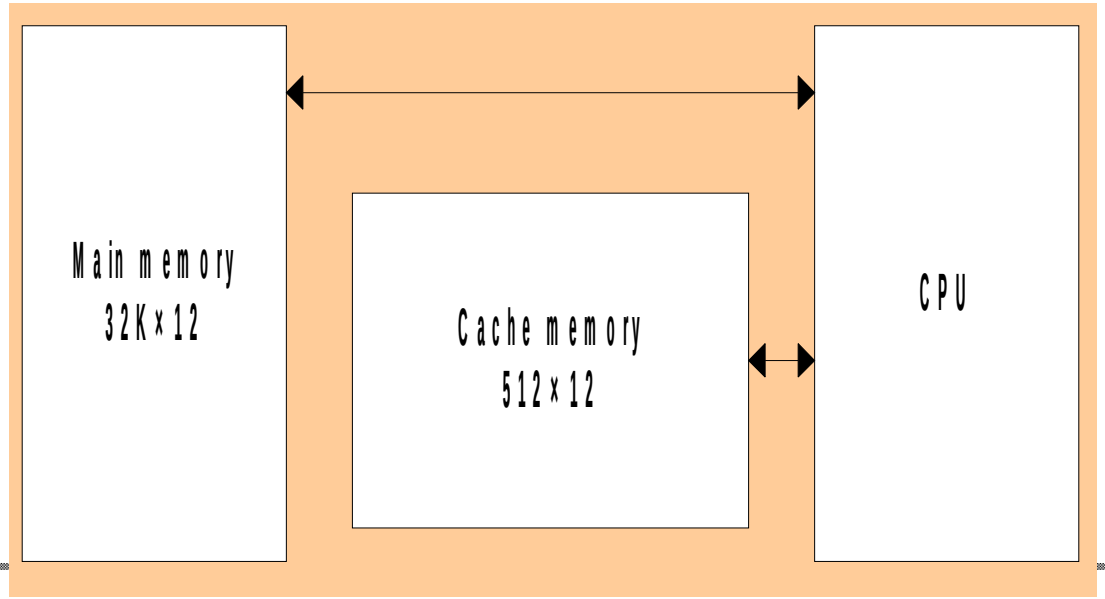
- the ratio of the number of hits divided by the total CPU references (hits + misses) to memory
  - » ***hit*** : the CPU finds the word in the cache (0.9)
  - » ***miss*** : the word is not found in cache (CPU must read main memory)
- An example where cache memory access time = 100 ns, main memory access time = 1000 ns, hit ratio = 0.9 produces an average access time of 200 ns.
  - » 1 miss : 1 x 1000 ns without the cache memory the time is 1000ns
  - » 9 hit : 9 x 100 ns



- Mapping
  - The transformation of data from main memory to cache memory
    - » 1) Associative mapping
    - » 2) Direct mapping
    - » 3) Set-associative mapping
- Example of cache memory :

  main memory : **32 K** x 12 bit word (15 bit address lines)

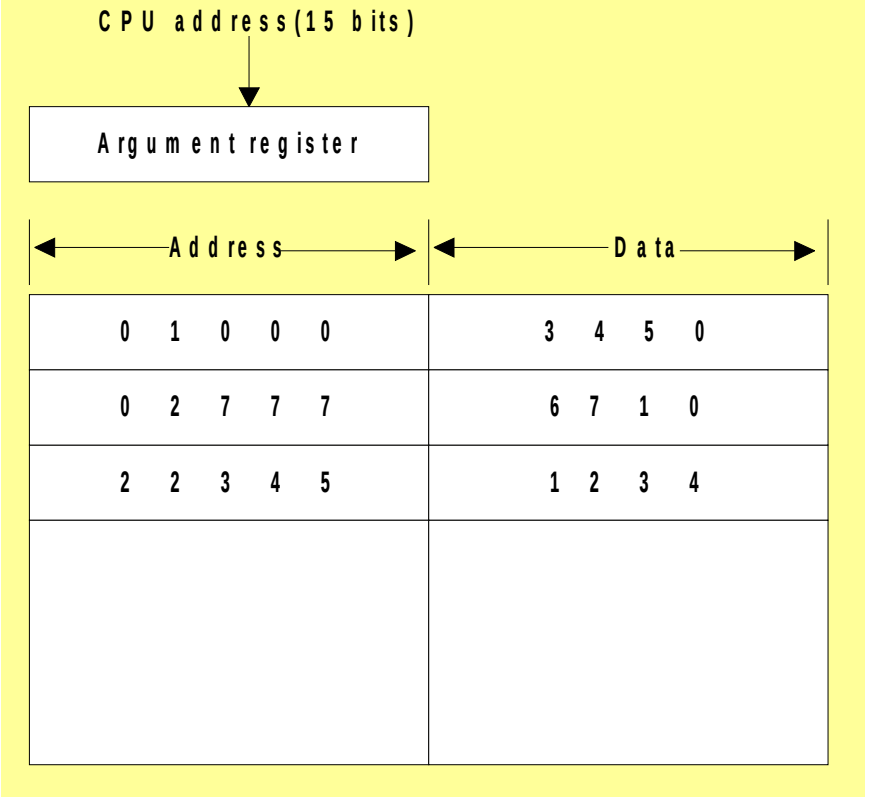
  cache memory : **512** x 12 bit word
    - » CPU sends a 15-bit address to cache
      - **Hit** : CPU accepts the 12-bit data from cache
      - **Miss** : CPU reads the data from main memory (then data is written to cache)



- Associative mapping : associative memory stores both address and data of the memory word.

If the address is found, the corresponding 12-bit data is read and send to the CPU. IF NO MATCH OCCURS, then main memory is accessed for the word. The address pair is then transferred to the associative memory. If the cache is full, an address-data pair must be displaced to make room for a pair that is needed and not presently is in cache.

This is done with replacement algorithm.

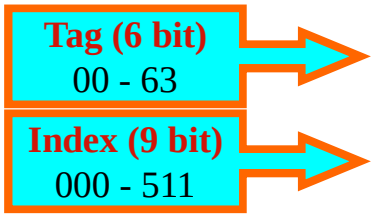


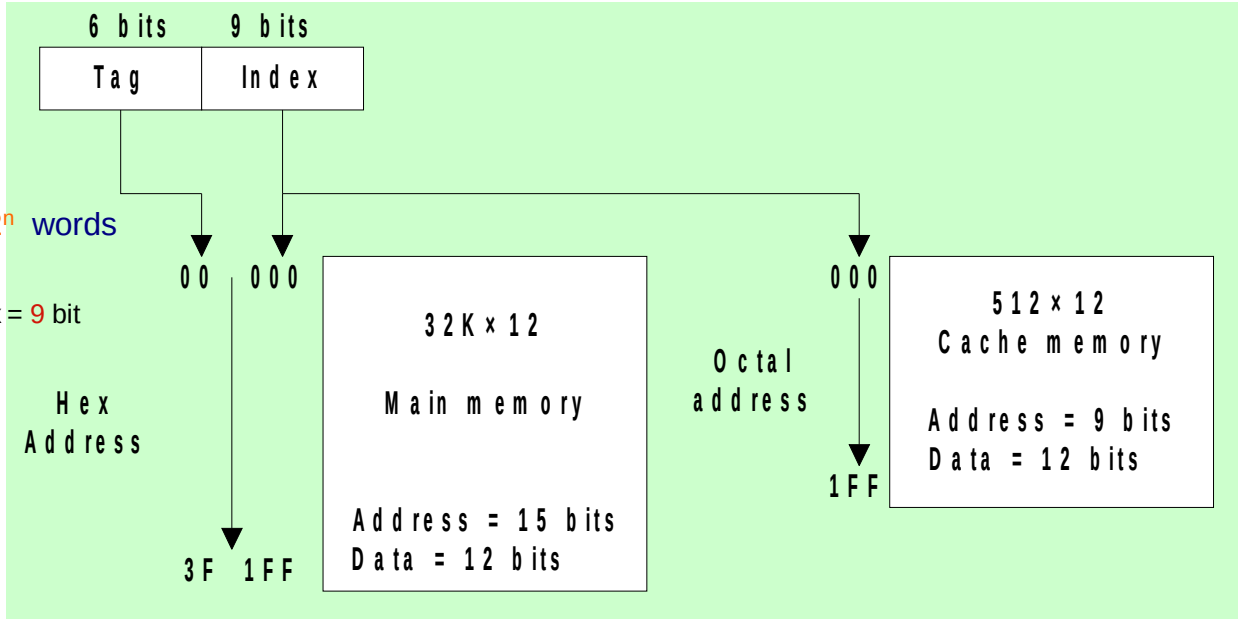
◆Direct mapping : *Fig. 12-12*

- ●Cache memory
- ●Tag field (**n - k**)
- ●Index field (**k**)

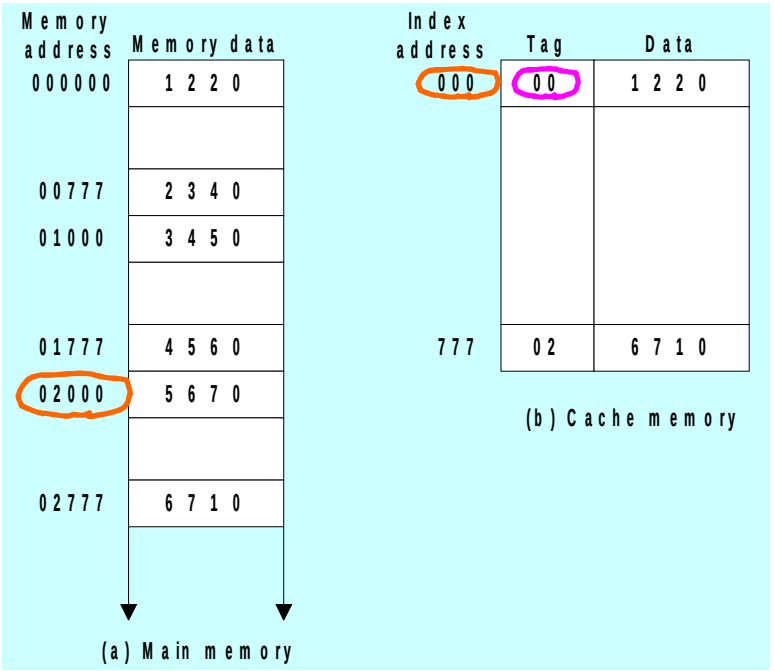
»2^k words cache memory **and** 2^n words main memory

▪**Tag** = 6 bit (15 - 9), **Index** = 9 bit

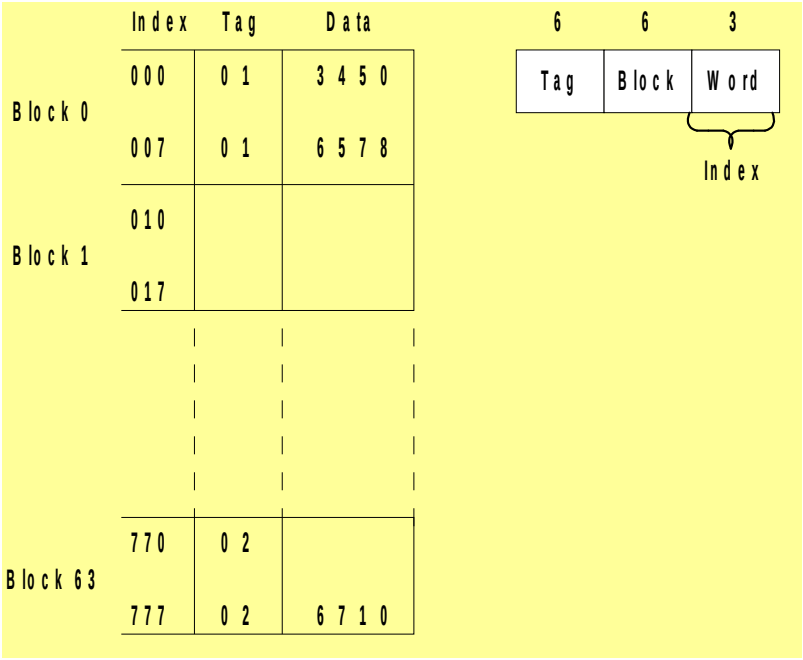




- Direct mapping cache organization : *Fig. 12-13*
  - » ***For address*** **02000**
  - 1) Index 000 cache , tag 00 and data 1220
  - 2) Suppose CPU wants to access the word at address 02000.
  - 3) The index address is 000 so it is used to Access cache. Two tags then compared.
  - 4)Cache tag 00 but address tag 02, not match
  - 5)Main m/m accessed & data word 5670 is Transferred to CPU.
  - 6)Now 000 is replaced with tag 02 & data 5670.

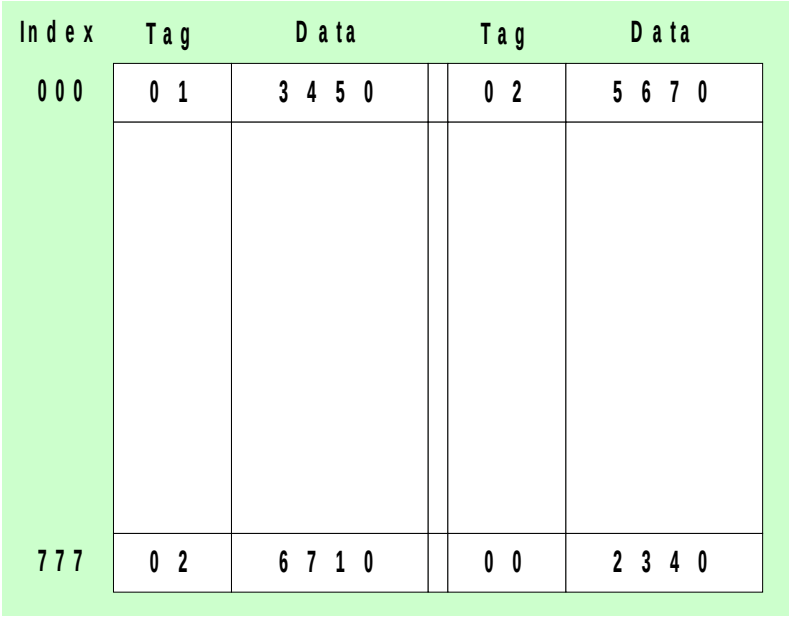


- Direct mapping cache with block size of 8 words : ***Fig. 12-14***
  - » 64 block x 8 word = 512 cache words size



Set-associative mapping :

Disadvantage of direct mapping: two words with the same index in their address but with different tag values can not reside in cache memory at the same time.



Index	Tag	Data	Tag	Data
000	01	3450	02	5670
777	02	6710	00	2340

- Replacement Algorithm : cache miss or full
  - 1) **LRU** (Least Recently Used) :
  - 2) **LFU** (Least Frequently Used) :
  - 3) **FIFO** (First-In First-Out) :
- Writing to Cache : Cache Coherence
  - » 1) **Write-through** : UPDATE the main memory with every memory write operation with cache memory being updated in parallel.
  - » 2) **Write-back** : only cache location is updated during the write operation. The location is then marked by flag so that later when the word is removed from the cache it is copied into main memory.
- Cache Initialization
  - Cache is initialized :
  - 1) when power is applied to the computer
    - » 2) when main memory is loaded with a complete set of programs from auxiliary memory
  - valid bit
    - » indicate whether or not the word contains valid data

Application

- Cache memory is a mechanism interposed in the memory hierarchy between main memory and the CPU to improve effective memory transfer rates and raise processor speeds.
- Cache memory operates like a "frequently used data" file for your CPU. The cache dynamically stores and accesses the data your CPU and applications access most often so that it can be more quickly retrieved.
- Caches are designed to alleviate this bottleneck by making the data used most often by the CPU instantly available. This is accomplished by building a small amount of memory, known as primary or level 1 cache, right into the CPU. Level 1 cache is very small, normally ranging between 2 kilobytes (KB) and 64 KB.