# CPU Cache

Eric McCreath

## CPU Cache

A CPU-cache stores copies of main memory data, the aim of this is to reduce the average access time. This works because the cache is small and fast (SRAM) in comparison to main memory which is big and slow (DRAM). Cache's generally store a contiguous 'line' of bytes form memory (typically 64 or 128 bytes).

A CPU-cache is critically for the overall performance of a modern computing system. This is because a cache provides a fast way of getting access to the memory it is currently using (~3 clock cycles to access L1 cache, ~15+ L2 cache, ~200+ main memory).

The working set is the memory that a program is currently using. Over the execution of a program the working set will change. If the working set fits within the cache then the program will not be slowed by latency to DRAM.

## CPU Cache

When reading via the cache if the data is in the cache then it can be read immediately. If the data is not in the cache it needs to be loaded from memory into a cache line. This will involve finding and allocating a cache line. This may also involve writing that cache line out.

- A cache hit happens when on access a copy of the memory data is in the cache (so no need to go to main memory).
- A cache miss happens when on access there is no copy in the cache, requires access to main memory. A write miss can also happen when a write occurs with data that is not currently cached.

$$\text{hitrate} = \frac{\text{hits}}{\text{accesses}}$$

## CPU Cache

The write policy determines the approach taken when data is written to memory via the cache.

There are two basic approaches taken when a write is done:

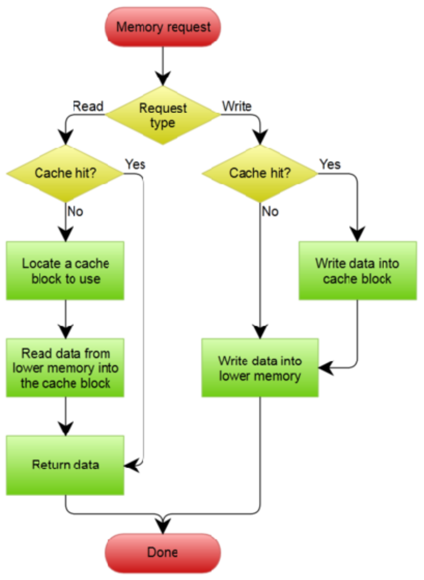
- write-through - data is written both to the cache and the memory at the same time.
- write-back - data is just written to the cache.

Another issue is when writing is if the data being written is not in a cache line (write miss) then either:

- the data could be written directly to memory (no-write allocate) or
- first loaded into a cache line then written to memory (write allocate).

## CPU Cache

### Write-Through cache with No-Write Allocation

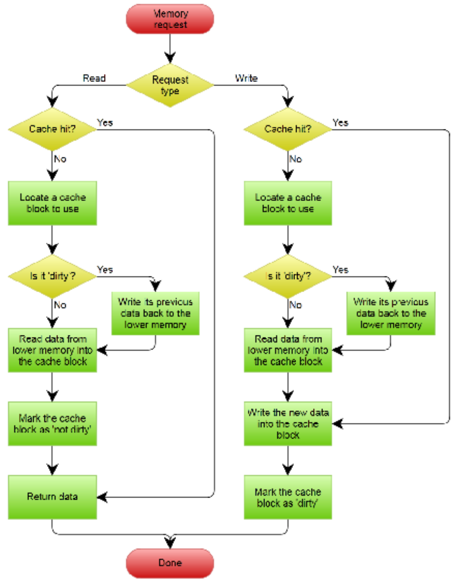


Memory request
Request type
Read
Write
Cache hit?
Yes
No
Cache hit?
Yes
No
Locate a cache block to use
Write data into cache block
Read data from lower memory into the cache block
Write data into lower memory
Return data
Done

Public Domain, 2011, Flin00, http://en.wikipedia.org/wiki/Cache_(computing%)

## CPU Cache

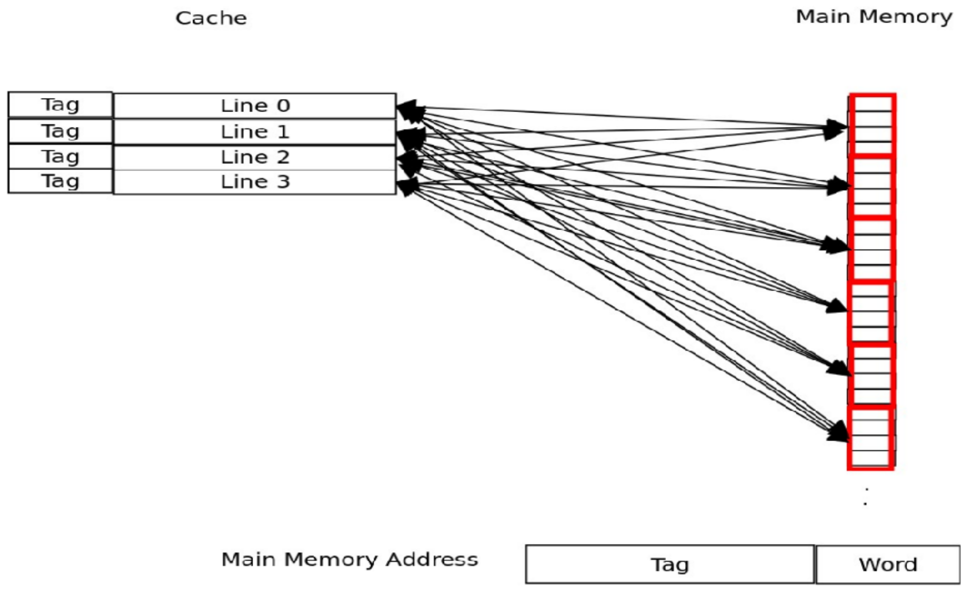
### Write-Back cache with Write Allocation



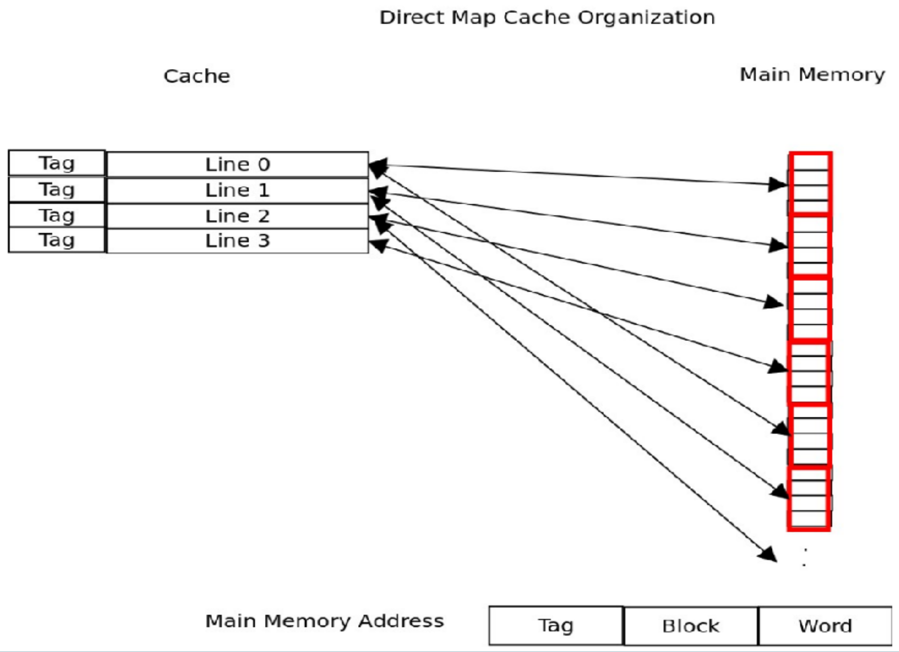
Memory request
Request type
Read
Write
Cache hit?
Yes
No
Locate a cache block to use
Is it 'dirty'?
Yes
No
Write its previous data back to the lower memory
Read data from lower memory into the cache block
Mark the cache block as 'not dirty'
Return data
Cache hit?
Yes
No
Locate a cache block to use
Is it 'dirty'?
Yes
No
Write its previous data back to the lower memory
Read data from lower memory into the cache block
Write the new data into the cache block
Mark the cache block as 'dirty'
Done

Public Domain,2011, Flin00, http://en.wikipedia.org/wiki/Cache_(computing%)

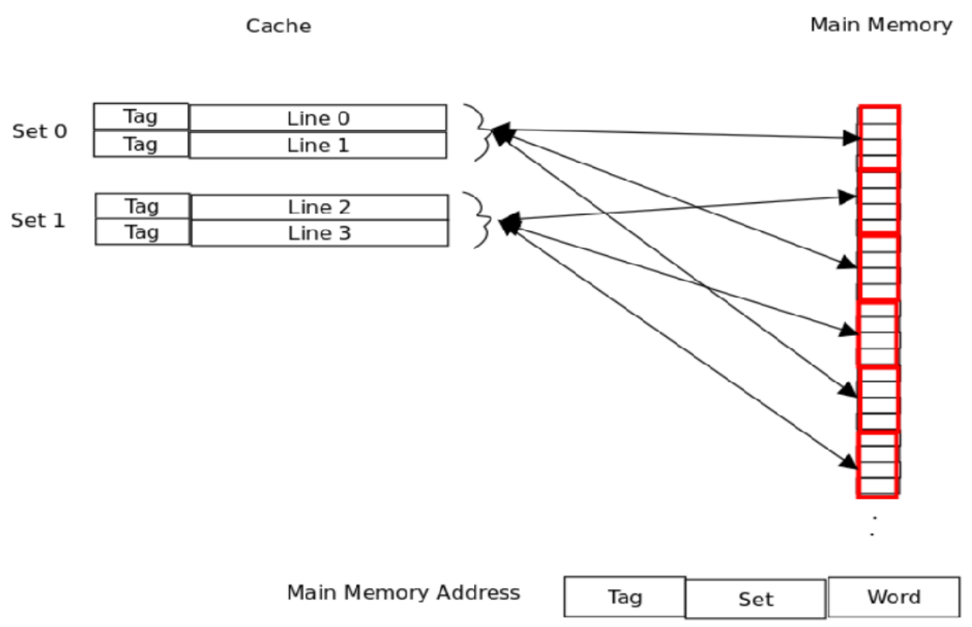
## Fully Associative Cache



## Direct Mapping



# Set Associative


Cache
Main Memory
Set 0
Tag
Line 0
Tag
Line 1
Set 1
Tag
Line 2
Tag
Line 3
Main Memory Address
Tag
Set
Word