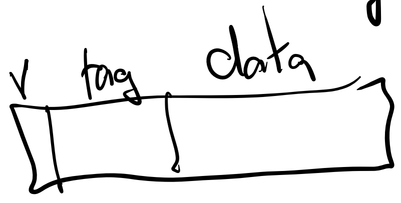
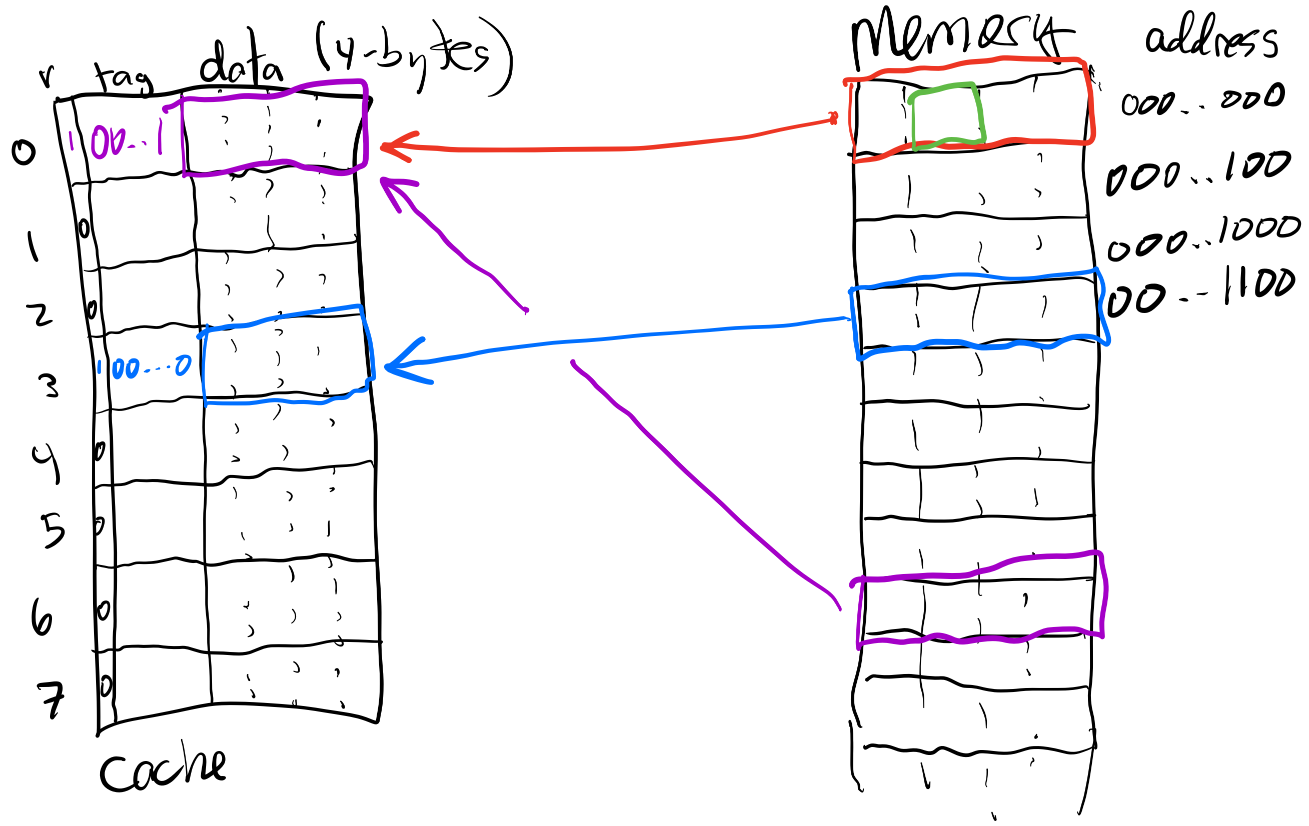
Each cache entry contains



Very simple example:

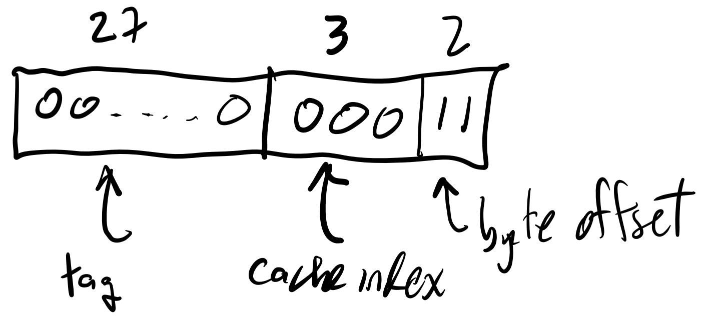
8-word (32 byte) cache

8 words of data (we don't count space used by v and tag bits)

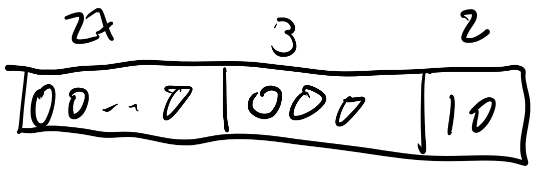


Process issues request for the data at address:

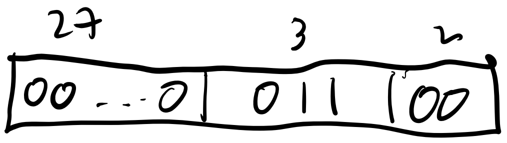
00...011 (address = 3)



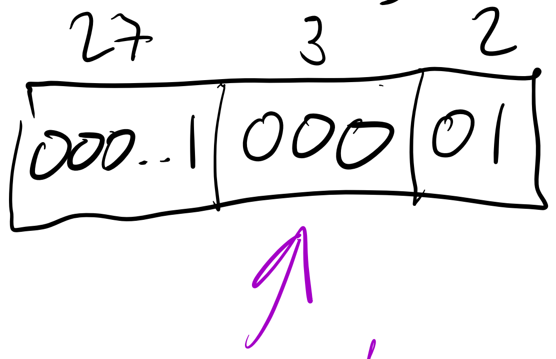
00...010 (address = 2)



000...01100 (address = 12)



000...100001 (address = 33)



Same cache index as the first two addresses.

- old data in cache entry 0 has to be "evicted".
  - kicked out.

If the data of the cache entry that is being evicted has not been changed (written to) then the cache entry can be overwritten with the new (incoming data).
  - the copy in memory is still correct.

When a processor performs a write to a memory location, if there is a cache hit, then the data in the cache entry is modified.

- at this point the data in memory and in cache are inconsistent.

- when is memory updated to reflect the change?

Two ways:

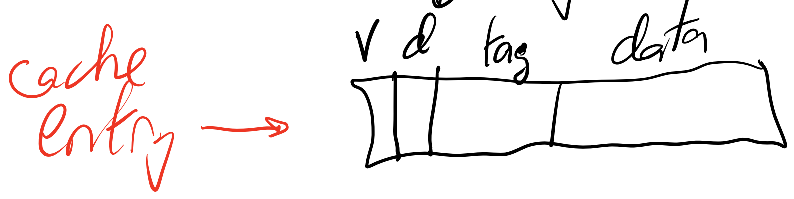
1. "Write-through cache"
   - the update is sent to memory immediately.
2. "Write-back cache"
   - the updated value is only written back to memory when the cache entry is evicted

Note: A cache is either write-through or write-back (not both).

Write Back:

- requires an extra bit in each cache entry to keep track of whether or not the data in the cache entry has been modified.

"dirty bit"



If the dirty bit of a cache entry is 1, the data gets written back to memory when the cache entry is evicted.

The above cache exploits temporal locality, but not spatial locality.

To exploit spatial locality, a cache miss should cause multiple words to be sent from memory to the cache

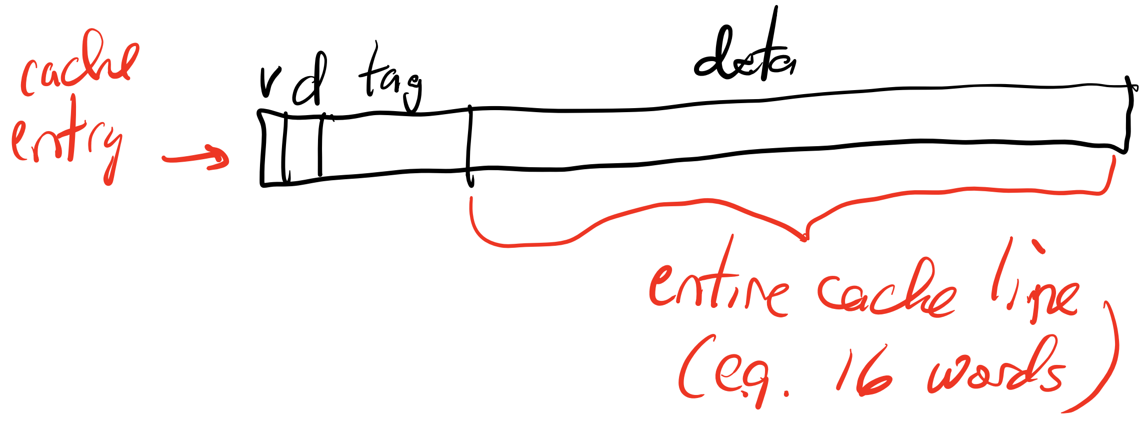
- 4, 8, $\textcircled{16}$, 32 words

- typical: 64 bytes = 16 4-byte words

"cache line" – the block of multiple words that get sent to the cache when there is a cache miss.

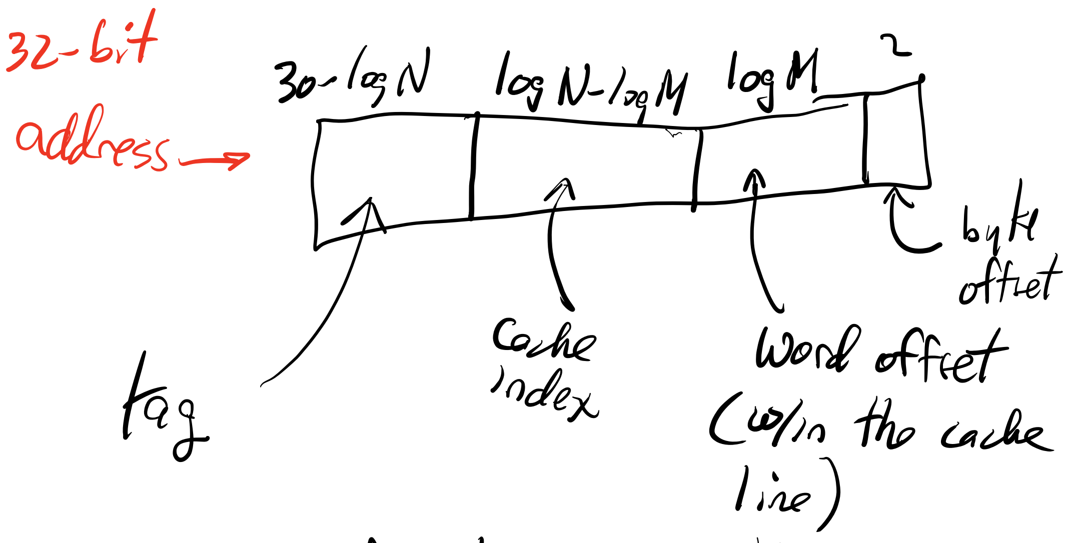
- aka "cache block"

Now, each cache entry contains a full cache line of data.



The processor only asks for one word in a cache line.

- that word needs to be selected from within the cache line.
  - If there are M words in a cache line, then log M bits of the address will be used to select the desired word within the cache line.
    - if a cache line is 16 words, then 4 bits of the address are used to select the desired word.



N = # words in the cache

M = # words in a cache line

How many cache entries are in the cache? (one cache line per entry)

N/M cache lines (entries) in the cache

How many bits do we need to select one entry in the cache?

Need $\log(N/M)$ bits

$= \log N - \log M$ bits

for the cache index.