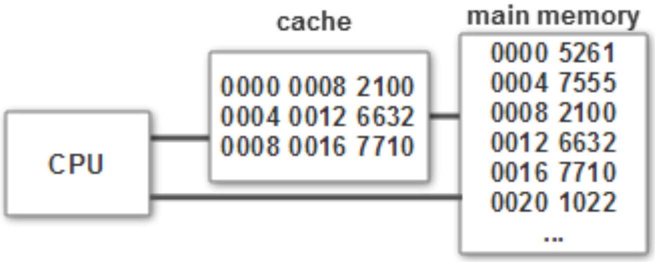
Caching

The first use of *cache* (also *casshe*) was for a hiding place, especially of goods, treasure, etc. The 1971 compressed OED (*Oxford English Dictionary*) quotes this sentence from 1595: “The inhabitants havinge intelligence of our cominge had .. hid theyre treasure in cassḥes.” *Cache* comes from the French *casher*, meaning “to hide”.

Cache is pronounced the same as *cash*; they are homophones. The brash moustached crook rashly stashed the stolen cash in a trash can full of ash; the police found the cache in a flash, a blink of an eyelash, and the crook gnashed his teeth in dispair.

A CPU memory cache

In a computer with a CPU (Central Processing Unit), it takes time to fetch data or instructions from the main memory. In order to speed up the process at least some of the time, the CPU may have its own memory —small and close by, so that accessing it is efficient— where it keeps copies of recently accessed parts of the main memory. It’s a cache, a small treaure trove, hidden from the user’s view.



We illustrate this using a hypothetical example, shown to the right. Main memory consists of words, each containing 4 bytes. The word at location 0008 contains the number 2100. The CPU has retrieved the word at location 0008 —and also the next two words because neighboring words will often be accessed— and stored them in its small cache. If asked to access word 0008, 0012, or 0016, the CPU gets the word from the cache. If asked to access some other word, a block of data containing that word will be retrieved from main memory and stored in the cache, replacing other words in the cache because the cache space is limited.

If the CPU has to write a number to main memory location 0008, the CPU writes it quickly to the cache and then goes on about its business while, in parallel, the number is written from the cache to the main memory.

That, in a, nutshell, is how a memory cache works. The MacBook Pro on which this document was written (in 2018) has 16 GB of memory and 4 corees, or Processing Units (CPUs). It has 3 levels of cache.

- An *L1 instruction cache* of 32KB for each core (Processing Unit)
- An *L1 data cache* of 32KB for each core (Processing Unit)
- An *L2 data cache* of 256KB for each core (Processing Unit)
- *An L3 cache* of 8 MB, which serves all cores.

A processor looking for a word of data in memory looks first in its L1 cache, then its L2 cache, then its L3 cache, and finally in memory. When it has to get the word from memory, a block of memory containing that word is brought into the caches.

How is the L1 instruction cache used? The Processing Unit can predict what will be needed next and preemptively load the next chunk of memory into the cache in the background.

Typically, the L1 cache is about 100 times faster than memory for data access, and the L2 cache is 25 times faster than memory for data access.

Cache performance

Typically, one uses *asymptotic complexity* measures to compare speeds of algorithms —we say things like "this sort method is in O(n log n) in the worst case, this other sort method is in $O(n^2)$." But when memory caches got large, people began comparing some algorithms based also on their *cache performance*. For example, this happened in discussions of hash tables used with open addressing.

Spacial locality, or *data locality*, refers to the use of data elements within relatively close storage locations. When one value is brought from memory into the memory cache, hundreds of others around it are brought with it. Therefore, an algorithm with high spacial locality can reference many values in the cache without having to retrieve them again from memory. This makes those references markedly faster, and high cache performance is achieved.

With regard to hashing with open addresses, one sees statements like this (Wikipedia article on Open Addressing): "linear probing has the best cache performance but is most sensitive to clustering, while double hashing has poor cache performance but exhibits virtually no clustering; quadratic probing falls in-between in both areas."

Other caches on computers

The CPU memory cache is a piece of hardware. Caches are used in other places in computing systems, and the cache might be in software as well as hardware. For example,

- Graphics processor units (GPUs) often have several kinds of caches to increase efficiency.
- Hard disk drives (HDDs) often come with built-in caches, perhaps of size 64MB or 256MB. These affect read performance more than write performance, making it quicker to retrieve data.
- Web browsers use caches to save both images and web pages that are expected to be used again, so they don't have to be retransmitted across the network. These are software caches.