Computer Organization & Architecture

Home Index

Cache Memory

Principles	Cache memory is intended to give fast memory speed, while at the same time providing a large memory size at a less expensive price.
Design Elements	There are a large number of cache implementations, but these basic design elements serve to classify cache architectures: **cache size**; **mapping function**; **replacement algorithm**; **write policy**; **line size**; **number of caches**.

Cache Size

There are several motivations for minimizing the cache size. The larger the cache, the greater the number of gates involved in addressing the cache is needed. The result is that larger caches end up being slightly slower than small ones. The available chip and board area also limits cache size.

Because the performance of the cache is very sensitive to the nature of the workload, it is impossible to arrive at a single optimum cache size.

Mapping Function

Mapping functions are used as a way to decide which main memory block occupies which line of cache. As there are less lines of cache than there are main memory blocks, an algorithm is needed to decide this. Three techniques are used, namely direct, associative and set associative, which dictate the organization of the cache.

- **Direct** This is the simplest form of mapping. One block from main memory maps into only one possible line of cache memory. As there are more blocks of main memory than there are lines of cache, many blocks in main memory can map to the same line in cache memory.

  To implement this function, use the following formula:

  $\alpha = \beta \% \gamma$

  where α is the cache line number, β is the block number in main memory, γ is the total number of lines in cache memory and % being the modulus operator.

  The main disadvantage of using this type of mapping is that there is a fixed cache location for any given block in main memory. If two blocks of memory sharing the same cache line are being continually referenced, cache misses would occur and these two blocks would continuously be swapped, resulting in slower memory access due to the time taken to access main memory (or the next level of memory).
- **Associative** This type of mapping overcomes the disadvantage of direct mapping by permitting each main memory block to be loaded into any line of cache. To do so, the cache control logic interprets a memory address as a tag and a word field. The tag uniquely identifies a block in main memory. The primary disadvantage of this method is that to find out whether a particular block is in cache, all cache lines would have to be examined. Using this method, replacement algorithms are required to maximize its potential.
- **Set Associative** This type of mapping is designed to utilize the strengths of the previous two mappings, while minimizing the disadvantages. The cache is divided into a number of sets containing an equal number of lines. Each block in main memory maps into one set in cache memory similar to that of direct mapping. Within the set, the cache acts as associative mapping where a block can occupy any line within that set. Replacement algorithms may be used within the set.

Replacement Algorithms

For direct mapping where there is only one possible line for a block of memory, no replacement algorithm is needed. For associative and set associative mapping, however, an algorithm is needed. For maximum speed, this algorithm is implemented in the hardware. Four of the most common algorithms are:

1. **least recently used** This replaces the candidate line in cache memory that has been there the longest with no reference to it.
2. **first in first out** This replaces the candidate line in the cache that has been there the longest.
3. **least frequently used** This replaces the candidate line in the cache that has had the fewest references.
4. **random replacement** This algorithm randomly chooses a line to be replaced from among the candidate lines. Studies have shown that this yields only slightly inferior performance than other algorithms.

Write Policy

This is important because if changes were made to a line in cache memory, the appropriate changes should be made to the block in main memory before removing the line from the cache. The problems to contend with are more than one device may have access to main memory (I/O modules). If more than one processor on the same bus with its own cache is involved, the problem becomes more complex. Any change in either cache or main memory could invalidate the others.

The simplest technique is called “write through”. In using this technique, both main memory and cache are written to when a write operation is performed, ensuring that main memory is always valid. The main disadvantage of this technique is that it may generate substantial main memory traffic, causing a bottle neck and decreasing performance.

An alternative technique, known as “write back” minimizes main memory writes. Updates are made only in the cache. An update bit associated with the line is set. Main memory is updated when the line in cache gets replaces only if the update bit has been set. The problem with this technique is that all changes to main memory have to be made through the cache in order not to invalidate parts of main memory, which potentially may cause a bottle neck.

Line Size

When a block of data is retrieved from main memory and put into the cache, the desired word and a number of adjacent words are retrieved. As the block size increases from a very small size, the hit ratio will at first increase due to the principle of locality of reference, which says that words in the vicinity of a referenced word are more likely to be referenced in the near future. As the block size increases, however, the hit ratio will decrease as the probability of reusing the new information becomes less than that of using the information that has been replaced.

Number of Caches

Two aspects of this are:

- **Multilevel** Due to increased logic density, it has become possible to have a cache on the same chip as the processor. This increases execution time as less activity over an external bus is needed. Even though an on-chip cache exists, it is typically desirable to have an off-chip cache as well. This means that if a miss occurs on the level 1 cache (on-chip), instead of retrieving the data from the slower main memory, information may be retrieved from the level 2 cache, which, although slower than level 1 cache, is still appreciably faster than main memory. Some level 2 caches are stored on-chip and a level 3 cache has been implemented off-chip.
- **Unified/Split** Two types of words exist that are stored in cache, namely data and instruction. It has become common to split the cache into two to separate these words.

  Two potential advantages to a unified cache are:
  1. A greater hit rate than split caches because the load between instruction and data fetches are balanced automatically.
  2. Only one cache needs to be designed and implemented.

  The key advantage of the split cache design is that it eliminates contention for the cache between the instruction fetch/decode unit and the execution unit. This is important for designs that rely on pipelining of instructions.

Previous Next

Content adapted from "**Computer Organization & Architecture: Designing for Performance 7th Edition**" & "**Logic and Computer Design Fundamentals**"