

Cache Memory

By Noor Kadhum

Historical introduction

Cache memory owes its introduction to Wilkes back in 1965. At that time, Wilkes distinguished between two types of main memory: The conventional and the slave memory. In Wilkes terminology, a slave memory is a second level of unconventional high-speed memory, which nowadays corresponds to what is called cache memory (the term cache means a safe place for hiding or storing things).



Sir Maurice Vincent Wilkes

Principle of : Locality of reference

The idea behind using a cache as the first level of the memory hierarchy is to keep the information expected to be used more frequently by the CPU in the cache (a small high-speed memory that is near the CPU).

The end result is that at any given time some active portion of the main memory is duplicated in the cache. Therefore when the processor makes a request for a memory reference, the request is first sought in the cache. If the request corresponds to an element that is currently residing in the cache, we call that *a cache hit*. On the other hand, if the request corresponds to an element that is not currently in the cache, we call that *a cache miss*.

Important Laws:

1) Hit ratio = hit / (hit + miss)

2) access time_{cache} = (hit + miss) * t_{cache}

3) access time_{M.M} = miss * t_{M.M}

4) Total access time = Access time_{cache} + Access time_{M.M}

5) Average access time :

a- with cache = Total access time / (hit+miss)

b- without cache = t_{M.M}

Example:

Compute the Average access time for memory system when the time for M.M is 1000 ns, the time for cache is 100 ns and hit ratio is 0.9.

Sol:

Hit ratio = 0.9 = 9/10 --> Hit = 9 , miss = 1

Access time_{cache} = (hit + miss) * t_{cache}
= 10 * 100ns = 1000 ns

Access time_{M.M} = miss * t_{M.M}
= 1 * 1000 ns = 1000 ns

Total access time = Access time_{cache} + Access time_{M.M}
= 1000 + 1000 = 2000 ns

Average access time :

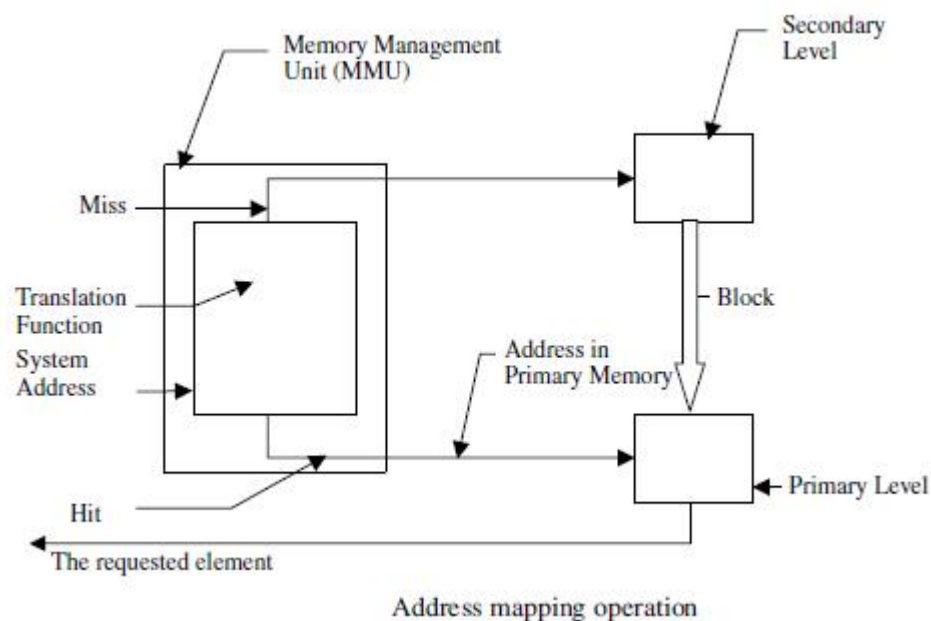
a- with cache = Total access time / (hit+miss)
= 2000 ns / 10 = 200 ns

b- without cache = t_{M.M}
= 1000 ns

Cache-Mapping Function

A request for accessing a memory element is made by the processor through issuing the address of the requested element. The address issued by the processor may correspond to that of an element that exists currently in the cache (cache hit); otherwise, it may correspond to an element that is currently residing in the main memory.

Therefore, address translation has to be made in order to determine the whereabouts of the requested element. This is one of the functions performed by the memory management unit (MMU). A schematic of the address mapping function is shown in the following Figure:



In this figure, the system address represents the address issued by the processor for the requested element. This address is used by an address translation function inside the MMU. If address translation reveals that the issued address corresponds to an element currently residing in the cache, then the element will be made available to the processor.

If, on the other hand, the element is not currently in the cache, then it will be brought (as part of a block) from the main memory and placed in the cache and the element requested is made available to the processor.