Jiwaji University
SOS in Computer Science and Applications
B.C.A. (IV Semester)
Paper -402 Advance Computer Architecture
Unit 2

Topic:

CACHE MEMORY
IT’S MAPPING

By Arun Singh

Cache memory

Cache memory is intended to give memory speed approaching that of the fastest memories available, and at the same time provide a large memory size at the price of less expensive types of semiconductor memories. The concept is illustrated in Figure 4.3a. There is a relatively large and slow main memory together with a smaller, faster cache memory. The cache contains a copy of portions of main memory. When the processor attempts to read a word of memory, a check is made to determine if the word is in the cache. If so, the word is delivered to the processor. If not, a block of main memory, consisting of some fixed number of words, is read into the cache and then the word is delivered to the processor. Because of the phenomenon of locality of reference, when a block of data is fetched into the cache to satisfy a single memory reference, it is likely that there will be future references to that same memory location or to other words in the block.

•If the active portions of the program and data are placed in a fast small memory, the average memory access time can be reduced

•Thus reducing the total execution time of the program

•Such a fast small memory is referred to as cache memory

•The cache is the fastest component in the memory hierarchy and approaches the speed of CPU component

•When CPU needs to access memory, the cache is examined

•If the word is found in the cache, it is read from the fast memory

•If the word addressed by the CPU is not found in the cache, the main memory is accessed to read the word

•When the CPU refers to memory and finds the word in cache, it is said to produce a **hit**

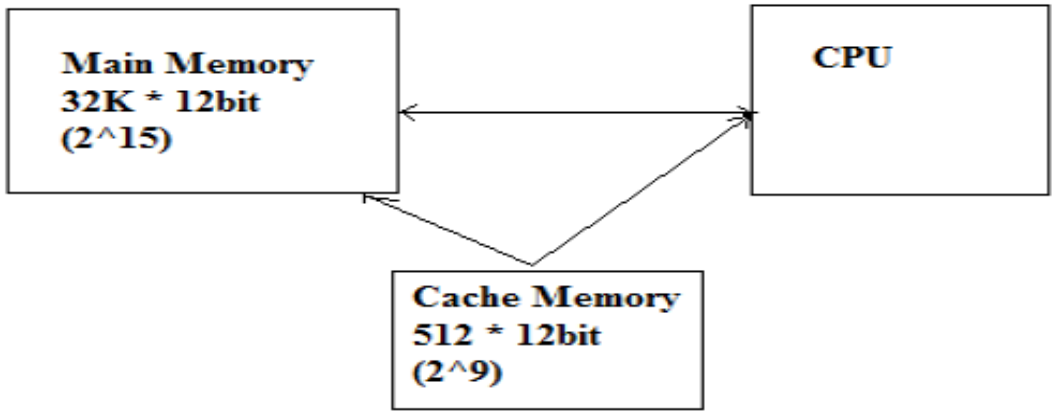
•Otherwise, it is a **miss**

•The performance of cache memory is frequently measured in terms of a quantity called **hit ratio**

Hit ratio = hit / (hit+miss)

•The basic characteristic of cache memory is its fast access time
•Therefore, very little or no time must be wasted when searching the words in the cache
•The transformation of data from main memory to cache memory is referred to as a **mapping** process, there are three types of mapping:

–**Associative mapping**
–**Direct mappin**
–**Set-associative**
•To help underst



Mapping Function
Because there are fewer cache lines than main memory blocks, an algorithm is needed for mapping main memory blocks into cache lines.
Further, a means is needed for determining which main memory block currently occupies a cache line.
The choice of the mapping function dictates how the cache is organized.

DIRECT MAPPING The simplest technique, known as direct mapping, maps each block of main memory into only one possible cache line. The mapping is expressed

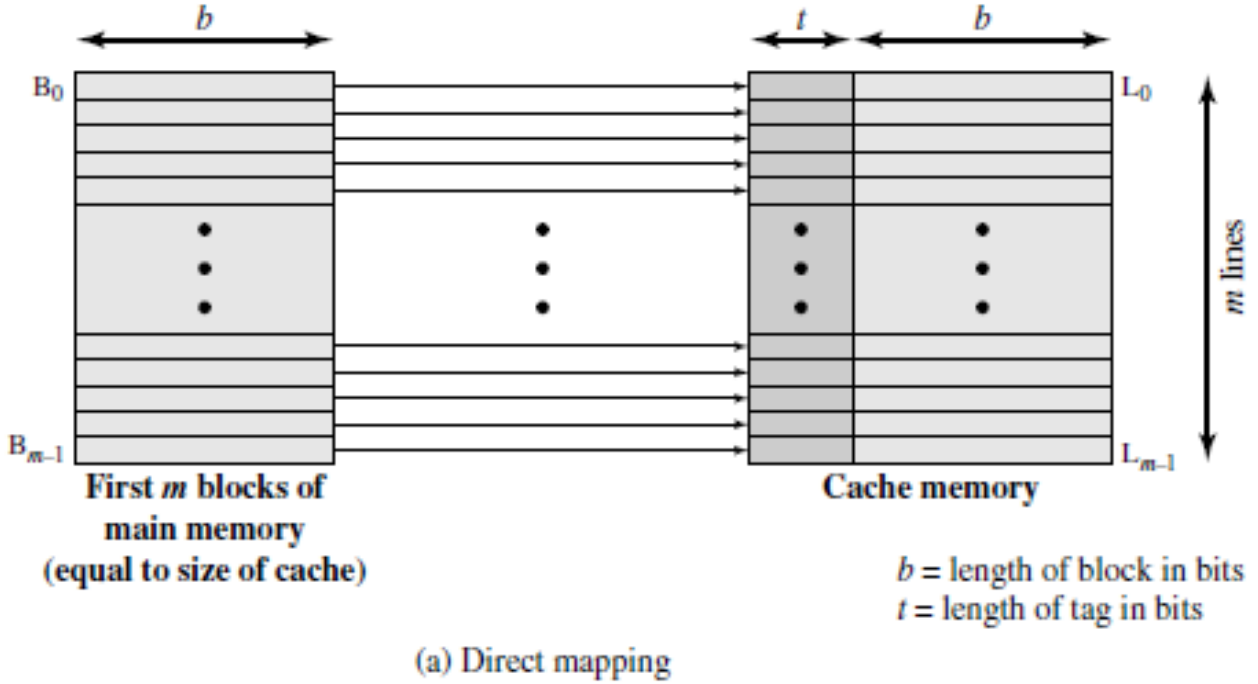
As i = j modulo m

where

i cache line number

j main memory block number

m number of lines in the cache

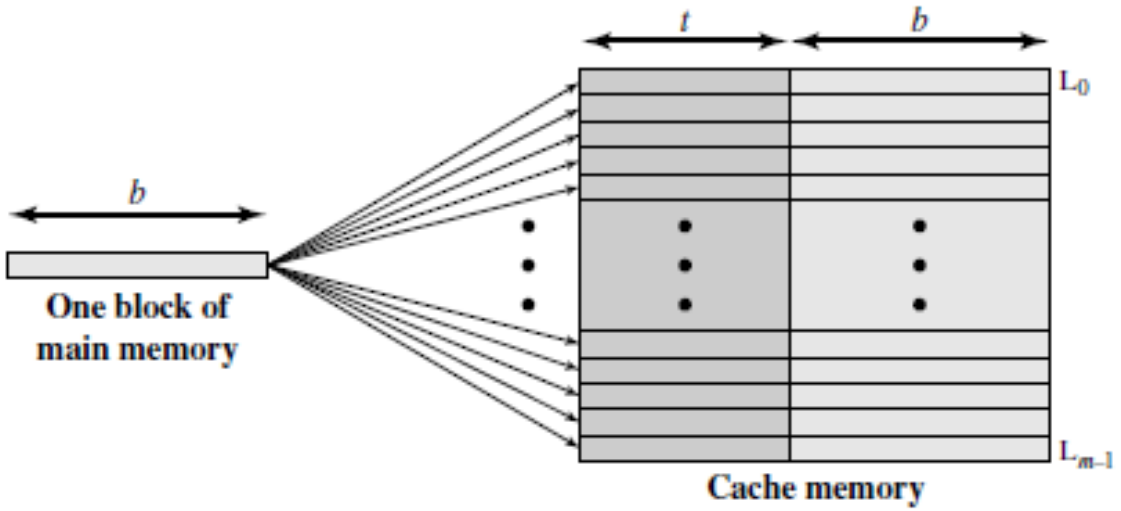


(a) Direct mapping

Figure shows the mapping for the first blocks of main memory. Each block of main memory maps into one unique line of the cache . The next blocks of main memory map into the cache in the same fashion; that is, block Bm of main memory maps into line L0 of cache, block Bm1 maps into line L1, and so on.

Associative mapping

•The fastest and most flexible cache organization uses an associative memory
•The associative memory stores both the address and data of the memory word
•This permits any location in cache to store ant word from main memory
•The address value of 15 bits is shown as a five-digit **octal** number and its corresponding 12-bit word is shown as a four-digit octal number
•A CPU address of 15 bits is places in the argument register and the associative memory us searched for a matching address
•If the address is found, the corresponding 12-bits data is read and sent to the CPU
•If not, the main memory is accessed for the word
•If the cache is full, an address-data pair must be displaced to make room for a pair that is needed and not presently in the cache



SET-ASSOCIATIVE MAPPING Set-associative mapping is a compromise that exhibits the strengths of both the direct and associative approaches while reducing their disadvantages. In this case, the cache consists of a number sets, each of which consists of a number of lines . The relationships are where

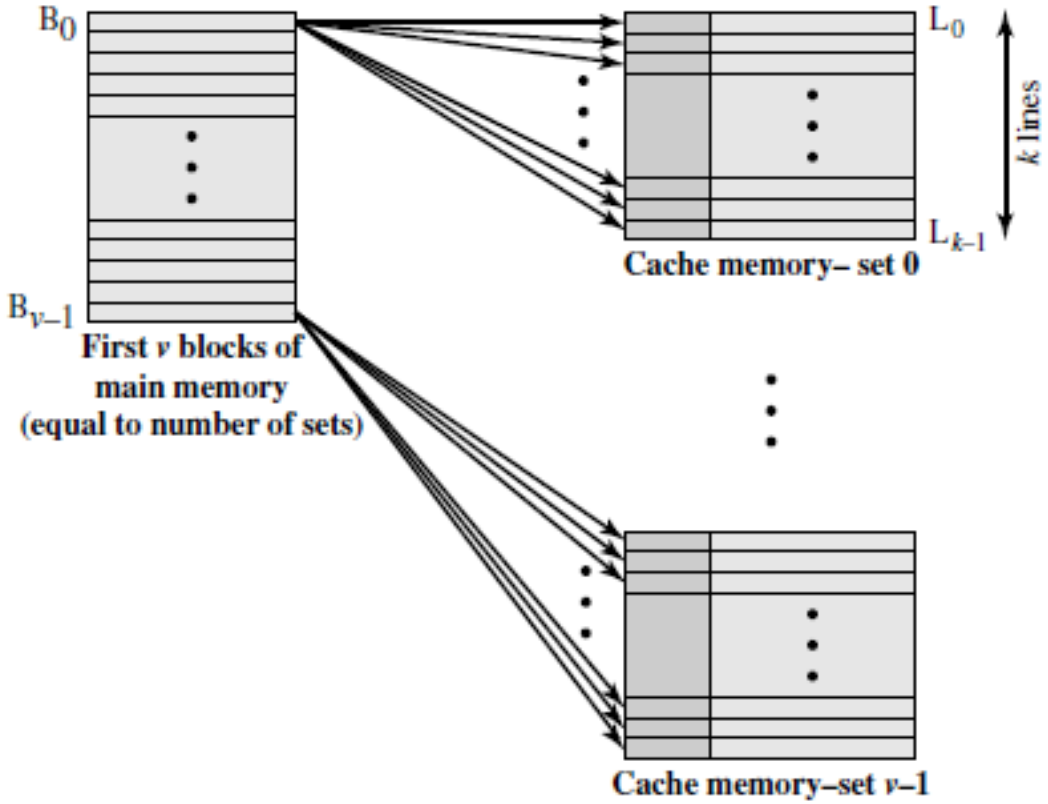
i cache set number

j main memory block number

m number of lines in the cache

number of sets

k number of lines in each set



(a) v Associative–mapped caches

Cache Addresses

Almost all no embedded processors, and many embedded processors, support virtual Memory. In essence, virtual memory is a facility that allows programs to address memory from a logical point of view, without regard to the amount of main memory physically available . When virtual memory is used, the address fields of machine instructions contain virtual addresses.

When virtual addresses are used, the system designer may choose to place the cache between the processor and the MMU or between the MMU and main memory (Figure 4.7). A **logical cache**, also known as a **virtual cache**, stores data using **virtual addresses**. The processor accesses the cache directly, without going through the MMU. A physical cache stores data using main memory **physical addresses**.

Cache Size

The first item in Table 4.2, cache size, has already been discussed . We would like the size of the cache to be small enough so that the overall average cost per bit is close to that of main memory alone and large enough so that the overall average access time is close to that of the cache alone . There are several other motivations for minimizing cache size . The larger the cache, the larger the number of gates involved in addressing the cache . The result is that large caches tend to be slightly slower than small ones—even when built with the same integrated circuit technology and put in the same place on chip and circuit board.