Example:

The details of a four-level hierarchical memory system can be summarized as follows:

LEVEL	COMPONENT TYPE	ACCESS TIME	HIT RATIO
1	ECL main memory	$t_1 = 100$ ns	$H(S_1) = 0.6$
2	Core memory	$t_2 = 1\ \mu$s	$H(S_2) = 0.8$
3	Magnetic disk	$t_3 = 50$ ms	$H(S_3) = 0.9$
4	Tape	$t_4 = 1$ s	$H(S_4) = 1.0$

Determine the average access time of this memory system.

SOLUTION

$F(S0) = 1$, $F(S1) = 1 - H(S1) = 0.4$,

$F(S2) = 1 - H(S2) = 0.2$, and $F(S3) = 1 - H(S3) = 0.1$

$$\begin{aligned}\overline{T} &= F(S0)t_1 + F(S1)t_2 + F(S2)t_3 + F(S3)t_4 \\ &= (10^{-4} + 0.4 \times 10^{-3} + 0.2 \times 50 + 0.1 \times 1000) \text{ ms} \\ &= (0.0001 + 0.0004 + 10.0 + 100) \text{ ms} \\ &= 110.0005 \text{ ms}\end{aligned}$$

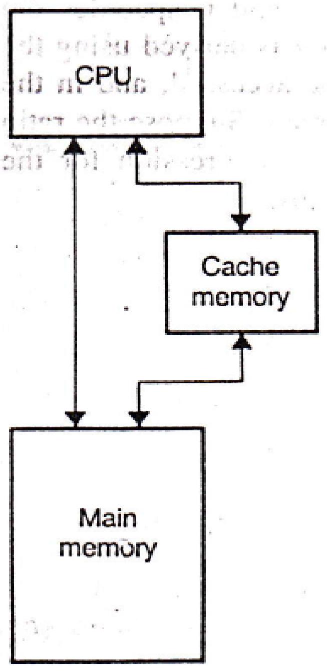

Figure 5.15 Memory Organization of a Computer System that Employs a Cache Memory

The block diagram representation of a computer system that employs a cache is shown in Figure 5.15. Cache assumes the top level in the computer's memory system. Usually, the cache-memory size is much smaller than the main memory and 5 to 10 times faster.

Assume the following mode of operation: An address generated by the CPU is first sent to the cache. If this reference is found in the cache, a *cache hit* is said to occur, and the data pertaining to the CPU reference is transferred to the CPU from the cache. However, if the reference is not found in the cache, a *cache miss* is said to have occurred. When there is a cache miss, first the required data is transferred to the cache from the main memory, and then it is read by the CPU from the cache. Usually the data is transferred from the main memory to cache in blocks.

Typical block size is 4 to 64 words. When the cache is full, one of the existing blocks will be evicted using standard replacement policies such as first-in first-out (FIFO) or least recently used (LRU). These policies are discussed later in this chapter. This block transfer is carried out in anticipation that the block is very likely to be referenced again and again in the near future. Such an intuitive guess is strongly supported by many practical situations such as tight loops and repeated use of a subroutine or a data structure such as a symbol table corresponding to a source program.

The performance of a system that employs a cache can be formally analyzed as follows: If t_c, h, and t_m specify the cache-access time, hit ratio, and the main memory access time, respectively; then the average access time can be determined as shown in equation 5.7:

$$\bar{t} = ht_c + (1 - h)(t_c + t_m) \qquad (5.7)$$

The hit ratio h always lies in the closed interval 0 and 1, and it specifies the relative number of successful references to the cache. Equation 5.7 is derived using the fact that when there is a cache hit, the main memory will not be accessed; and in the event of a cache miss, both main memory and cache will be accessed. Suppose the ratio of main memory access time to cache access time is γ, then an expression for the efficiency of a system that employs a cache can be derived as follows:

$$\text{Efficiency} = \Lambda = \frac{t_c}{\bar{t}}$$

$$= \frac{t_c}{ht_c + (1 - h)(t_c + t_m)}$$

$$= \frac{1}{h + (1 - h)\,(1 + \frac{t_m}{t_c})}$$

$$= \frac{1}{h + (1 - h)(1 + \gamma)}$$

$$= \frac{1}{1 + \gamma(1 - h)}$$

Note that Λ is maximum when $h = 1$ (when all references are confined to the cache). A hit ratio of 90% ($h = 0.90$) is not uncommon with many contemporary systems.

Example:

Calculate $\bar{t}$, γ, and Λ of a memory system whose parameters are as indicated:

$$t_c = 160 \text{ ns}$$
$$t_m = 960 \text{ ns}$$
$$h = 0.90$$

SOLUTION

$$\bar{t} = ht_c + (1 - h)\,(t_c + t_m)$$
$$= 0.9(160) + (0.1)(960 + 160)$$
$$= 144 + 112$$
$$= 256 \text{ ns}$$

$$r = \frac{t_m}{t_c} = \frac{960}{160} = 6$$

$$\Lambda = \frac{1}{1 + r(1 - h)} = \frac{1}{1 + 6(0.1)} = \frac{1}{1.6} = 0.625$$