Solution to Problem 6.12 (page 605)

Sometimes, understanding why something is a bad idea helps you understand why the alternative is a good idea. Here, the bad idea we are looking at is indexing the cache with the high-order bits instead of the middle bits.

A. With high-order bit indexing, each contiguous array chunk consists of 2^t blocks, where t is the number of tag bits. Thus, the first 2^t contiguous blocks of the array would map to set 0, the next 2^t blocks would map to set 1, and so on.

B. For a direct-mapped cache where $(S, E, B, m) = (512, 1, 32, 32)$, the cache capacity is 512 32-byte blocks, and there are $t = 18$ tag bits in each cache line. Thus, the first 2^{18} blocks in the array would map to set 0, the next 2^{18} blocks to set 1. Since our array consists of only $(4096 * 4)/32 = 512$ blocks, all of the blocks in the array map to set 0. Thus, the cache will hold at most one array block at any point in time, even though the array is small enough to fit entirely in the cache. Clearly, using high-order bit indexing makes poor use of the cache.

Problem 6.24 Solution:

The average rotational latency (in ms) is

$$\begin{aligned} T_{avg\,rotation} &= 1/2 \times T_{max\,rotation} \\ &= 1/2 \times (60\ secs\ /\ 12{,}000\ RPM) \times 1000\ ms/sec \\ &\approx 2.5\ ms. \end{aligned}$$

The average transfer time is

$$\begin{aligned} T_{avg\,transfer} &= (60\ secs\ /\ 12{,}000\ RPM) \times 1/500\ sectors/track \times 1000\ ms/sec \\ &\approx 0.01\ ms. \end{aligned}$$

Putting it all together, the total estimated access time is

$$\begin{aligned} T_{access} &= T_{avg\,seek} + T_{avg\,rotation} + T_{avg\,transfer} \\ &= 3\ ms + 2.5\ ms + 0.01\ ms \\ &= 5.51\ ms. \end{aligned}$$

Problem 6.35 Solution:

This problem is tougher than it looks. The approach is similar to the solution to Problem 6.18. The cache is not large enough to hold both arrays. References to cache lines for one array evict recently loaded cache lines from the other array.

dst array				
	col 0	col 1	col 2	col 3
row 0	m	h	m	h
row 1	m	m	h	m
row 2	m	h	m	h
row 3	m	m	h	m

src array				
	col 0	col 1	col 2	col 3
row 0	m	m	m	m
row 1	m	m	m	m
row 2	m	m	m	m
row 3	m	m	m	m

Problem 6.36 Solution:

In this case, the cache is large enough to hold both arrays, so the only misses are the initial cold misses.

dst array				
	col 0	col 1	col 2	col 3
row 0	m	h	h	h
row 1	m	h	h	h
row 2	m	h	h	h
row 3	m	h	h	h

src array				
	col 0	col 1	col 2	col 3
row 0	m	h	h	h
row 1	m	h	h	h
row 2	m	h	h	h
row 3	m	h	h	h