Locality and Cache Memory

- Week 3 -

Prepared by

Lee Chang Yoon

32183641@dankook.ac.kr

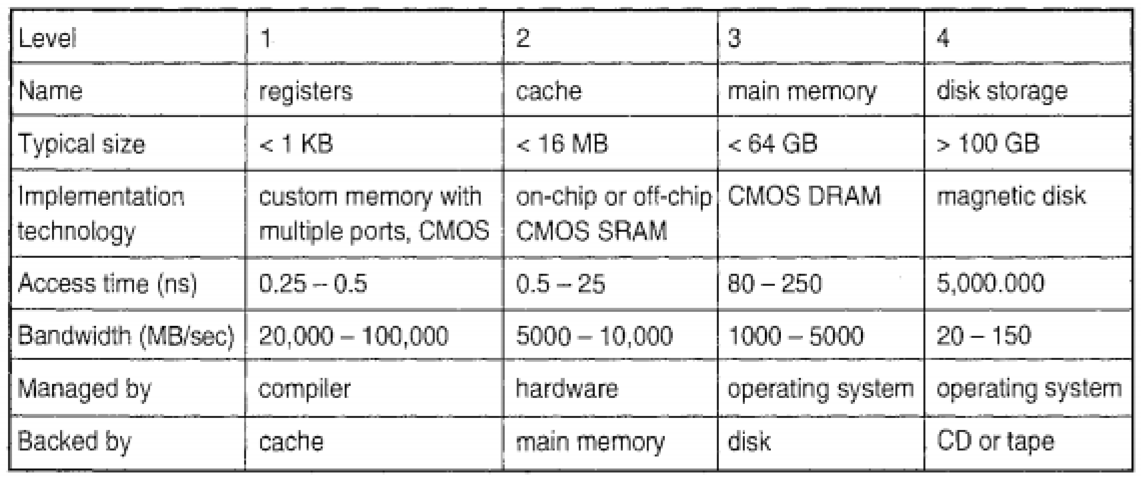
September 14, 2021

Executive Summary

This report is for homework for the system programming class on week 3. The main goal for this report is to explain how the cache memory operates & collaborates in your computer: How does the cache memory works and what are temporal locality & spatial locality.

Q1.

How does cache memory works?



Level	1	2	3	4
Name	registers	cache	main memory	disk storage
Typical size	< 1 KB	< 16 MB	< 64 GB	> 100 GB
Implementation technology	custom memory with multiple ports, CMOS	on-chip or off-chip CMOS SRAM	CMOS DRAM	magnetic disk
Access time (ns)	0.25 – 0.5	0.5 – 25	80 – 250	5,000.000
Bandwidth (MB/sec)	20,000 – 100,000	5000 – 10,000	1000 – 5000	20 – 150
Managed by	compiler	hardware	operating system	operating system
Backed by	cache	main memory	disk	CD or tape

Figure 1.11 Performance of various levels of storage.

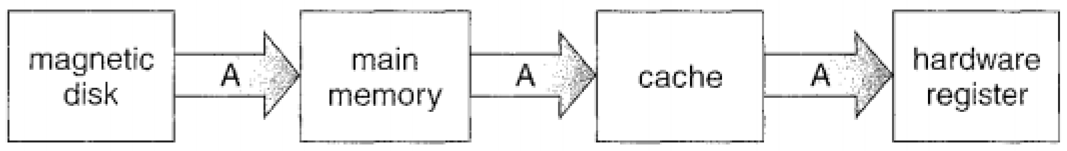


Figure 1.12 Migration of integer A from disk to register.

Caching is an important principle of computer systems. Information is normally kept in some storage systems, such as main memory. As it is used, it is copied into a faster storage system, cache on figure 1.11, on a temporary basis. The cache is a sort of SRAM (Solid State Random Access Memory) that consisted of flip-flops. Cache memory temporarily stores information, data, and programs that are commonly used by the CPU. How does the cache memory work?

If we need a particular piece of information, we first check whether it is in the cache. If it is, we use the information directly from the cache and we call this sequence as 'Cache Hit'. However, if not, we use the information from the source, putting a copy in the cache under the assumption that we will need it again soon (Figure 1.12), and we call this sequence as 'Cache Miss'. For instance, most systems have an instruction cache to hold the instructions expected to be executed next. Without this cache, the CPU would have to wait for several cycles while instruction was fetched from the main memory. Main memory can be viewed as a fast cache for secondary storage since data in secondary storage must be copied into main memory for use, and data must be in main memory before being moved to secondary storage for safekeeping.

In short, cache memory is used to make the CPU work faster. When we read data from the main memory, it flows into the cache, and the CPU deals with it. When CPU read data from memory, it first checks if the data is in the cache. If the data is in the cache, CPU just uses the data in cache. However, if it is not, current data is copied into the cache and the CPU uses data in the main memory.

Q2.

What is temporal locality and spatial locality?

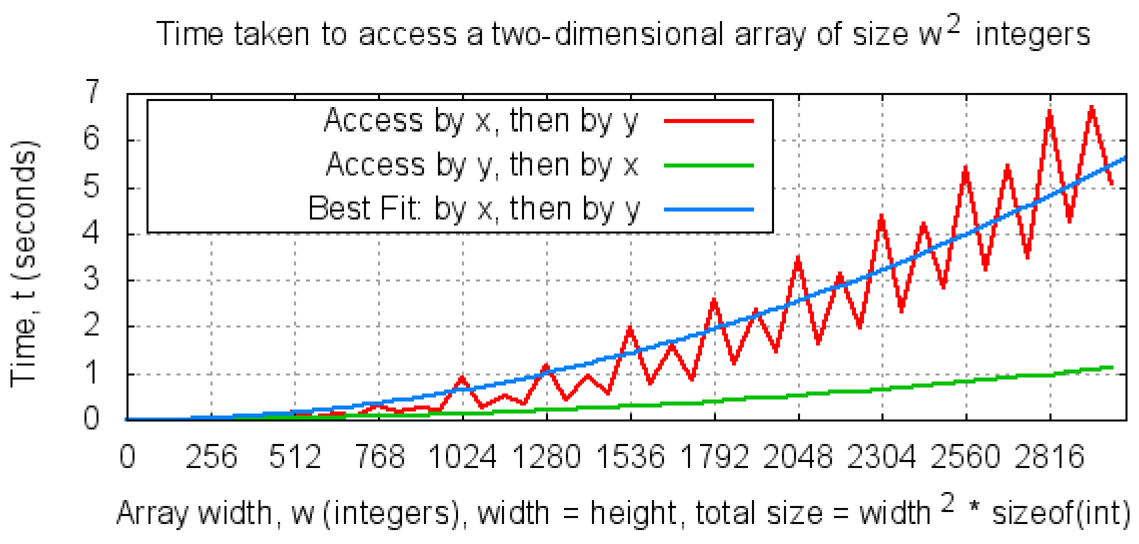
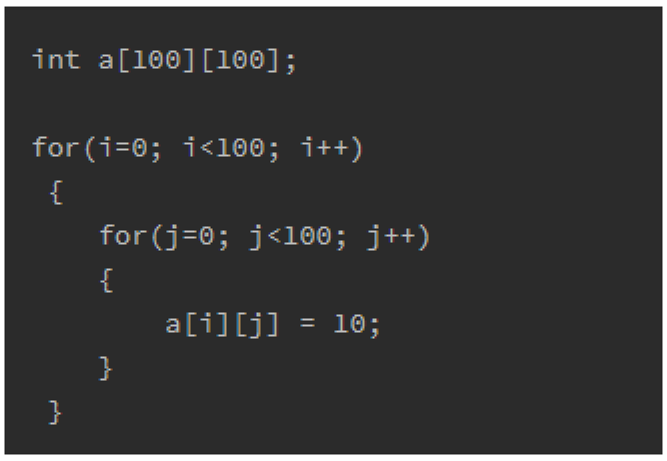


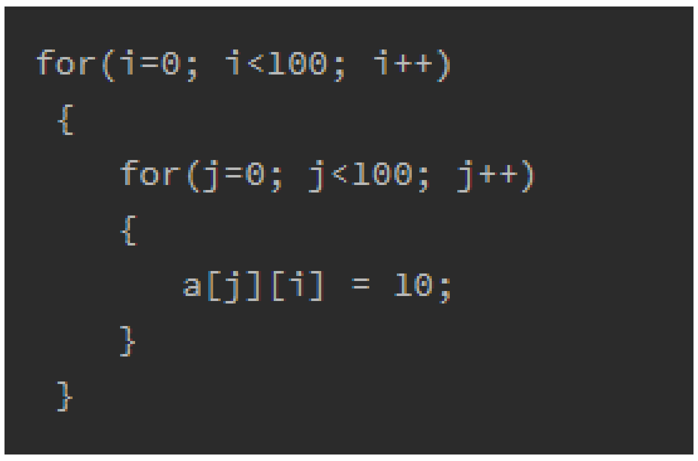
Figure 1



```
int a[100][100];

for(i=0; i<100; i++)
 {
    for(j=0; j<100; j++)
    {
        a[i][j] = 10;
    }
 }
```

Figure 2



```
for(i=0; i<100; i++)
 {
    for(j=0; j<100; j++)
    {
        a[j][i] = 10;
    }
 }
```

Figure 3

The principle of locality is the main concept to understand the computer memory system. The principle of locality is the concept that the program only accesses a small part of the memory block in a temporary time. There are two kinds of localities: Temporal Locality and Spatial Locality.

Temporal Locality means that the memory location which has been used once is likely to be used again soon. It is an expression of the same memory area of an address that has been accessed once is frequently encountered when we execute the program. Spatial Locality means that when data is referenced, it shows the tendency of intensive reference to memory locations of a particular cluster. This is an expression of the nature of a program that the area of memory accessed during program execution is highly likely to be near the area that has already been accessed.

Check the following C code in Figure 2 and Figure 3. In Figure 2, for every loop after j++, the program is referencing the same array data but just near the element of an array element, we referenced before, which is cache hits. In short, the memory location of array a[][] has spatial locality. However, in Figure 3, the way of loops are accessing the array a[][] is not a sequential way, which is the cache miss. As a result, in the case of spatial locality, we can say that the code in Figure 2 is much more efficient than the code in Figure 3, while they are doing the same jobs. Figure 1 shows the efficiency between the code like Figure 2 and Figure 3. How about temporal locality? Let's check Figure 2 as an example. In the loop of Figure 2, we are accessing i and j again and again after we use each loop. We can find out that i and j are frequently encountered after we execute the program. In short, the memory locations of i and j in Figure 2 have temporal locality.

In conclusion, the principle of locality has two kinds of locality which are temporal locality and spatial locality. Memory has temporal locality if the memory area of an address has been accessed once is frequently encountered when we execute the program. Memory has spatial locality when it has the tendency of intensive reference to memory locations of a particular cluster if the data is referenced.

Conclusion

By checking Q1 and Q2, which are how cache works and what is the temporal locality and spatial locality, we can know figure out how the cache memory operates in the computer systems. Although, we should have to add some details for each case, we identified the basic operations of caches. We will learn more about operation of cache memory in the later lecture, which is computer architecture.