Batch: D3	**Roll No.: 16010123294**
Experiment / assignment / tutorial No. 06	

TITLE : Implementation of Cache Mapping Techniques.

AIM: To study and implement concept of various mapping techniques designed for cache memory.

Expected OUTCOME of Experiment: (Mention CO/CO's attained here)

Books/ Journals/ Websites referred:

1. Carl Hamacher, Zvonko Vranesic and Safwat Zaky, "Computer Organization", Fifth Edition, TataMcGraw-Hill.
2. Dr. M. Usha, T. S. Srikanth, "Computer System Architecture and Organization", First Edition, Wiley-India.

Pre Lab/ Prior Concepts:

Cache memory: The cache is a smaller, faster memory which stores copies of the data from the most frequently used main memory locations. As long as most memory accesses are cached memory locations, the average latency of memory accesses will be closer to the cache latency than to the latency of main memory.

2. Hit Ratio: You want to increase as much as possible the likelihood of the cache containing the memory addresses that the processor wants.

Hit Ratio= No. of hits/ (No. of hits + No. of misses)

There are only fewer cache lines than the 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 in a cache line. The choice of cache function dictates how the cache is organized. Three techniques can be used.

1. Direct mapping.
2. Associative mapping.
3. Set Associative mapping.

Direct Mapped Cache: The direct mapped cache is the simplest form of cache and the easiest to check for a hit. Since there is only one possible place that any memory location can be cached, there is nothing to search; the line either contains the memory information we are looking for, or it doesn't. Unfortunately, the direct mapped cache also has the worst performance, because again there is only one place that any address can be stored. Let's look again at our 512 KB level 2 cache and 64 MB of system memory. As you recall this cache has 16,384 lines (assuming 32-byte cache lines) and so each one is shared by 4,096 memory addresses. In the absolute worst case, imagine that the processor needs 2 different addresses (call them X and Y) that both map to the same cache line, in alternating sequence (X, Y, X, Y). This could happen in a small loop if you were unlucky. The processor will load X from memory and store it in cache. Then it will look in the cache for Y, but Y uses the same cache line as X, so it won't be there. So Y is loaded from memory, and stored in the cache for future use. But then the processor requests X, and looks in the cache only to find Y. This conflict repeats over and over. The net result is that the hit ratio here is 0%. This is a worst case scenario, but in general the performance is worst for this type of mapping.

Fully Associative Cache: The fully associative cache has the best hit ratio because any line in the cache can hold any address that needs to be cached. This means the problem seen in the direct mapped cache disappears, because there is no dedicated single line that an address must use.However (you knew it was coming), this cache suffers from problems involving searching the cache. If a given address can be stored in any of 16,384 lines, how do you know where it is? Even with specialized hardware to do the searching, a performance penalty is incurred. And this penalty occurs for all accesses to memory, whether a cache hit occurs or not, because it is part of searching the cache to determine a hit. In addition, more logic must be added to determine which of the various lines to use when a new entry must be added (usually some form of a "least recently used" algorithm is employed to decide which cache line to use next). All this overhead adds cost, complexity and execution time.

Set Associative Cache (To be filled in by students)

1
7
9

Direct Mapping Implementation:

The mapping is expressed as

i=j modulo m

i=cache line number

j= main memory block number

m= number of lines in the cache

- Address length = (s+w) bits
- Number of addressable units = 2^{s+w} words or bytes
- Block size = line size = 2^{w} words or bytes
- Number of blocks in main memory = $2^{s+w} / 2^{w} = 2^{s}$
- Number of lines in cache = m = 2^{r}
- Size of tag = (s-r) tags

Associative Mapping Implementation: **(To be filled in by students)**

In the associate mapping of the main memory block can be done with any of the cache block. This technique is called as fully associative cache mapping. The memory address has only 2 fields

i)word

ii)tag.

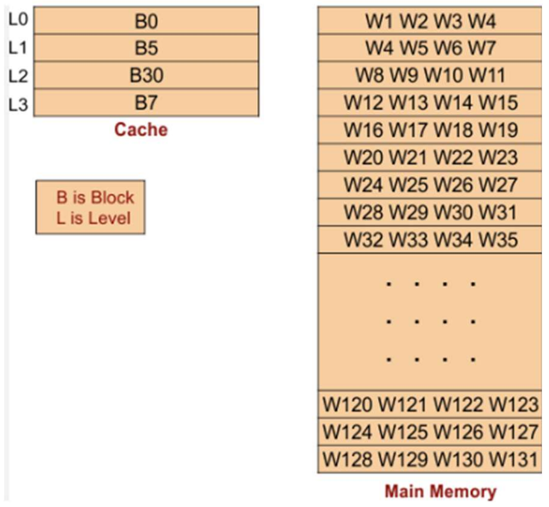
Example:

If we have a fully associative mapped cache of 8 KB size with block size = 128 bytes and say, the size of main memory is = 64 KB. Then:

Number of bits for the physical address = 16 bits (as memory size = 64 KB = 26 × 210 = 216)

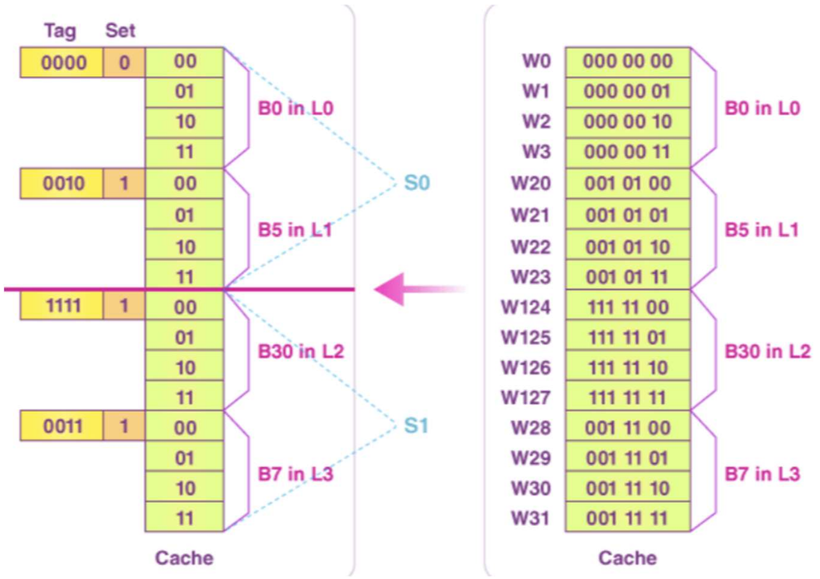
Number of bits in block offset = 7 bits (as block size = 128 bytes = 27)
No of tag bits = Number of bits for the physical address – Number of bits in block offset = 16-7 = 9 bits

No of cache Blocks = Cache size/block size = 8 KB / 128 Bytes = 8×1024 Bytes/128 Bytes = 26 blocks.



Set Associative Mapping Implementation:

Set associative mapping is a cache memory mapping technique that combines direct mapping and associative mapping. It works by organizing cache lines into sets, allowing a main memory block to be mapped to any line in a specific set



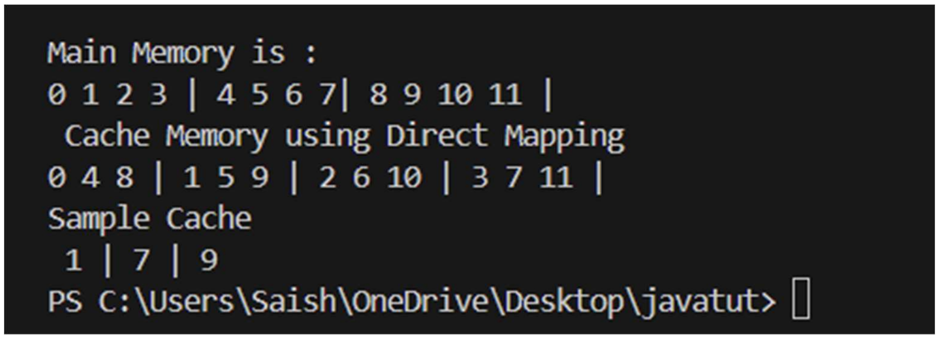
Code Implementation:

```
import java.util.Scanner;


public class DirectMappingCache {


    public static void main(String[] args) {
        System.out.println("Main Memory is :");
        System.out.println("0 1 2 3 | 4 5 6 7| 8 9 10 11 |");
        System.out.println(" Cache Memory using Direct Mapping ");
        System.out.println("0 4 8 | 1 5 9 | 2 6 10 | 3 7 11 | ");
        System.out.println("Sample Cache ");
        System.out.println(" 1 | 7 | 9");


    }
}
```



```
Main Memory is :
0 1 2 3 | 4 5 6 7| 8 9 10 11 |
 Cache Memory using Direct Mapping
0 4 8 | 1 5 9 | 2 6 10 | 3 7 11 |
Sample Cache
 1 | 7 | 9
PS C:\Users\Saish\OneDrive\Desktop\javatut> 
```

<u>Post Lab Descriptive Questions</u>

1. For a direct mapped cache, a main memory is viewed as consisting of 3 fields. List and define 3 fields.

In a direct-mapped cache, a main memory address is divided into 3 fields:
Tag: Identifies which block of main memory is currently stored in a particular cache line.
Index: Selects which cache line to use (determined by address % cache size).
Offset: Identifies the specific data within the block (used if blocks have multiple words).
These fields help in mapping memory addresses to cache lines and locating data within a block.

2. What is the general relationship among access time, memory cost, and capacity?

The general relationship among access time, memory cost, and capacity is:
Access Time: Faster memory (lower access time) is more expensive.
Cost: Higher capacity memory is cheaper per unit, but slower.
Capacity: Larger capacity memory tends to have slower access time and lower cost per bit.

Conclusion
We learned about the various mapping techniques and implemented it in our codes for better understanding.

Date: _____________