Batch: A3	**Roll No.: 16010121045**
Experiment / assignment / tutorial No. 8	
Grade: AA / AB / BB / BC / CC / CD /DD	
Signature of the Staff In-charge with date	

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)

After CPU generates a memory request,

- The set number field of the address is used to access the set of the cache.
- The tag field of the CPU address is then compared with the tags of all k lines
- within that set.
- If the CPU tag matches to the tag of any cache line, a cache hit occurs.
- If the CPU tag does not match to the tag of any cache line, a cache miss occurs.
- In case of a cache miss, the required word must be brought from the main
- memory.
- If the cache is full, a replacement is made in accordance with the employed
- replacement policy.

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)**

- Address length =(s+w) bits
- Number of addressable units= 2 s+wwords or bytes
- Block size=line size = 2 wwords or bytes
- Number of blocks in main memory = 2 s
- Number of lines in cache = undefined
- size of tags = s bits

Set Associative Mapping Implementation:

m=v*k

i=j modulo n, where:

- i=cache line number
- j=main memory block number
- m=number of lines in the cache
- v=number of sets
- k=number of lines in each set
  - Addressable 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*
  - Number of lines in set=k
  - Number of sets = v = 2 *d*
  - Number of lines in cache = m = kv = k*2 *d*
  - Size of cache = k * 2 *d*+*w*words or bytes
  - Size of tag = (s-d) bits

Code:

```cpp
#include <bits/stdc++.h>
using namespace std;

int main()
{
    int memory_lines, blocks;
    cout << "Enter number of main memory lines:";
    cin >> memory_lines;
    cout << "Enter number of blocks in the main memory:";
    cin >> blocks;
    int bmemory[blocks][4];
    int mmemory[memory_lines];
    cout << "\nEnter the main memory data:" << endl;
    for (int i = 0; i < memory_lines; i++)
    {
        cout << "Line no. " << i + 1 << ": ";
        cin >> mmemory[i];
    }
    int k = 0;

    for (int i = 0; i < blocks; i++)
        for (int j = 0; j < 4; j++)
            bmemory[i][j] = mmemory[k++];
    cout << "\nDirect mapped cache\n";
    for (int i = 0; i < blocks; i++)
    {
        cout << endl
             << "Block " << i << ": ";
        for (int j = 0; j < 4; j++)
            cout << bmemory[i][j] << " ";
    }
    cout << "\n\nSample cache:\n";
    for (int i = 0; i < blocks; i++)
    {
```

```
        int random = rand() % 5;
        cout << bmemory[i][random] << " ";
    }
}
```

```
Enter number of main memory lines:10
Enter number of blocks in the main memory:2

Enter the main memory data:
Line no. 1: 11
Line no. 2: 2
Line no. 3: 42
Line no. 4: 12
Line no. 5: 51
Line no. 6: 09
Line no. 7: 11
Line no. 8: 33
Line no. 9: 22
Line no. 10: 23

Direct mapped cache

Block 0: 11 2 42 12
Block 1: 51 9 11 33

Sample cache:
42 1795061072 %
pargat@Router Programs %
```

Post Lab Descriptive Questions

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

- One field on the direct-mapped cache memory identifies a unique word or bytewithin a block of main memory
- The remaining two fields specify one of the blocks of main memory
- These two fields are a line field, which identifies one of the lines of the cache,and a tag field, which identifies one of the blocks that can fit into that line

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

- Faster access time is directly proportional to cost per bit, it means that as theaccess time speed increases the cost per bit also increases
- Memory capacity is inversely proportional to cost per bit, it means that asmemory capacity increases, the cost per bit decreases
- Memory capacity is inversely proportional to access time, it means that asmemory capacity increases the access time speed decreases

Conclusion : Therefore, with the help of the experiment the various mapping techniques are understood. The given task was implemented by writing programs to demonstrate them.

Date: _____ **Signature of faculty in-charge**