



EXPERIMENT - 5

AIM: To simulate a direct-mapped cache

Submission Sheet

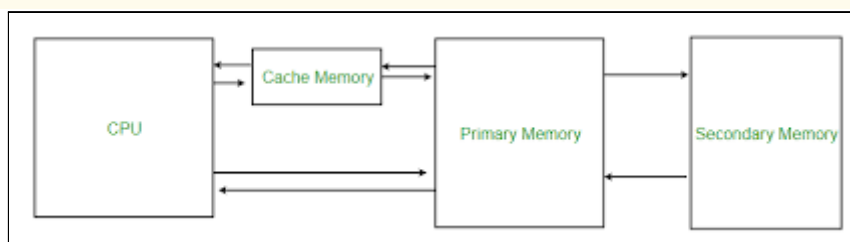
SAP ID	Name of Student	Date of Experiment	Date of Submission	Remarks
60004190057	Junaid Girkar	30-10-2021	30-10-2021	

THEORY:

Cache memory, also called cache, is a supplementary memory system that temporarily stores frequently used instructions and data for quicker processing by the central processing unit (CPU) of a computer. The cache augments, and is an extension of, a computer's main memory. Both main memory and cache are internal random-access memories (RAMs) that use semiconductor-based transistor circuits. Cache holds a copy of only the most frequently used information or program codes stored in the main memory. The smaller capacity of the cache reduces the time required to locate data within it and provide it to the CPU for processing.

When a computer's CPU accesses its internal memory, it first checks to see if the information it needs is stored in the cache. If it is, the cache returns the data to the CPU. If the information is not in the cache, the CPU retrieves it from the main memory. Disk cache memory operates similarly, but the cache is used to hold data that have recently been written on, or retrieved from, a magnetic disk or other external storage device.

Hit ratio = $\text{hit} / (\text{hit} + \text{miss}) = \text{no. of hits} / \text{total accesses}$.





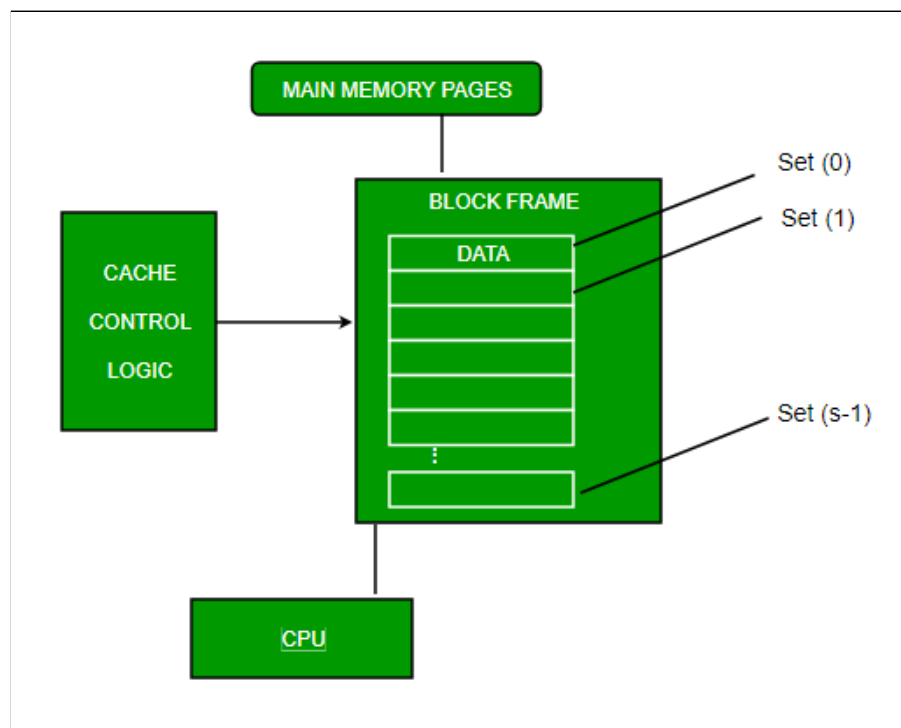
DIRECT CACHE MAPPING:

The simplest technique, known as direct mapping, maps each block of main memory into only one possible cache line. or

In Direct mapping, assign each memory block to a specific line in the cache. If a line is previously taken up by a memory block when a new block needs to be loaded, the old block is trashed. An address space is split into two parts: index field and a tag field. The cache is used to store the tag field whereas the rest is stored in the main memory. Direct mapping's performance is directly proportional to the Hit ratio.

```
i = j modulo m
where
i=cache line number
j= main memory block number
m=number of lines in the cache
```

For purposes of cache access, each main memory address can be viewed as consisting of three fields. The least significant w bits identify a unique word or byte within a block of main memory. In most contemporary machines, the address is at the byte level. The remaining s bits specify one of the 2^s blocks of main memory. The cache logic interprets these s bits as a tag of $s-r$ bits (most significant portion) and a line field of r bits. This latter field identifies one of the $m=2^r$ lines of the cache.





8 Bit:

CODE:

```
#include <stdio.h>

int tag[8];

int main( )
{
    int addr;
    int i, j, t;
    int hits, accesses;
    FILE *fp;

    fp = fopen("trace.txt", "r");
    hits = 0;
    accesses = 0;
    while (fscanf(fp, "%x", &addr) > 0) {
        /* simulate a direct-mapped cache with 8 words */
        accesses += 1;
        printf("%3d: 0x%08x ", accesses, addr);
        i = (addr >> 2) & 7;
        t = addr | 0x1f;
        if (tag[i] == t) {
            hits += 1;
            printf("Hit%d ", i);
        } else {
            /* allocate entry */
            printf("Miss ");
            tag[i] = t;
        }
        for (i = 0; i < 8; i++)
            printf("0x%08x ", tag[i]);
        printf("\n");
    }
    printf("Hits = %d, Accesses = %d, Hit ratio = %f\n", hits, accesses,
((float)hits)/accesses);
    close(fp);
}
```



OUTPUT:

[illegible]



Shri Vile Parle Kelavani Mandal's

DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING

(Autonomous College Affiliated to the University of Mumbai)

NAAC Accredited with "A" Grade (CGPA : 3.18)



```
66: 0x80000010 Miss 0x0000005f 0x0000005f 0x0000005f 0x8000001f 0x8000001f 0x8000001f 0x8000001f 0x0000003f
67: 0x80000014 Hit5 0x0000005f 0x0000005f 0x0000005f 0x8000001f 0x8000001f 0x8000001f 0x8000001f 0x0000003f
68: 0x80000018 Hit6 0x0000005f 0x0000005f 0x0000005f 0x8000001f 0x8000001f 0x8000001f 0x8000001f 0x0000003f
69: 0x8000000c Hit3 0x0000005f 0x0000005f 0x0000005f 0x8000001f 0x8000001f 0x8000001f 0x8000001f 0x0000003f
70: 0x00000054 Miss 0x0000005f 0x0000005f 0x0000005f 0x8000001f 0x8000001f 0x0000005f 0x8000001f 0x0000003f
71: 0x80000010 Hit4 0x0000005f 0x0000005f 0x0000005f 0x8000001f 0x8000001f 0x0000005f 0x8000001f 0x0000003f
72: 0x80000014 Miss 0x0000005f 0x0000005f 0x0000005f 0x8000001f 0x8000001f 0x8000001f 0x8000001f 0x0000003f
73: 0x80000018 Hit6 0x0000005f 0x0000005f 0x0000005f 0x8000001f 0x8000001f 0x8000001f 0x8000001f 0x0000003f
74: 0x8000000c Hit3 0x0000005f 0x0000005f 0x0000005f 0x8000001f 0x8000001f 0x8000001f 0x8000001f 0x0000003f
75: 0x00000058 Miss 0x0000005f 0x0000005f 0x0000005f 0x8000001f 0x8000001f 0x8000005f 0x0000003f
76: 0x80000010 Hit4 0x0000005f 0x0000005f 0x0000005f 0x8000001f 0x8000001f 0x8000001f 0x0000005f 0x0000003f
77: 0x80000014 Hit5 0x0000005f 0x0000005f 0x0000005f 0x8000001f 0x8000001f 0x8000001f 0x0000005f 0x0000003f
78: 0x80000018 Miss 0x0000005f 0x0000005f 0x0000005f 0x8000001f 0x8000001f 0x8000001f 0x8000001f 0x0000003f
79: 0x8000000c Hit3 0x0000005f 0x0000005f 0x0000005f 0x8000001f 0x8000001f 0x8000001f 0x8000001f 0x0000003f
80: 0x0000005c Miss 0x0000005f 0x0000005f 0x0000005f 0x8000001f 0x8000001f 0x8000001f 0x8000001f 0x0000005f
81: 0x80000010 Hit4 0x0000005f 0x0000005f 0x0000005f 0x8000001f 0x8000001f 0x8000001f 0x8000001f 0x0000005f
82: 0x80000014 Hit5 0x0000005f 0x0000005f 0x0000005f 0x8000001f 0x8000001f 0x8000001f 0x8000001f 0x0000005f
83: 0x80000018 Hit6 0x0000005f 0x0000005f 0x0000005f 0x8000001f 0x8000001f 0x8000001f 0x8000001f 0x0000005f
84: 0x8000000c Hit3 0x0000005f 0x0000005f 0x0000005f 0x8000001f 0x8000001f 0x8000001f 0x8000001f 0x0000005f
85: 0x00000060 Miss 0x0000007f 0x0000005f 0x0000005f 0x8000001f 0x8000001f 0x8000001f 0x8000001f 0x0000005f
86: 0x80000010 Hit4 0x0000007f 0x0000005f 0x0000005f 0x8000001f 0x8000001f 0x8000001f 0x8000001f 0x0000005f
87: 0x80000014 Hit5 0x0000007f 0x0000005f 0x0000005f 0x8000001f 0x8000001f 0x8000001f 0x8000001f 0x0000005f
88: 0x80000018 Hit6 0x0000007f 0x0000005f 0x0000005f 0x8000001f 0x8000001f 0x8000001f 0x8000001f 0x0000005f
89: 0x8000000c Hit3 0x0000007f 0x0000005f 0x0000005f 0x8000001f 0x8000001f 0x8000001f 0x8000001f 0x0000005f
90: 0x00000064 Miss 0x0000007f 0x0000007f 0x0000005f 0x8000001f 0x8000001f 0x8000001f 0x8000001f 0x0000005f
91: 0x80000010 Hit4 0x0000007f 0x0000007f 0x0000005f 0x8000001f 0x8000001f 0x8000001f 0x8000001f 0x0000005f
92: 0x80000014 Hit5 0x0000007f 0x0000007f 0x0000005f 0x8000001f 0x8000001f 0x8000001f 0x8000001f 0x0000005f
93: 0x80000018 Hit6 0x0000007f 0x0000007f 0x0000005f 0x8000001f 0x8000001f 0x8000001f 0x8000001f 0x0000005f
94: 0x8000000c Hit3 0x0000007f 0x0000007f 0x0000005f 0x8000001f 0x8000001f 0x8000001f 0x8000001f 0x0000005f
95: 0x00000068 Miss 0x0000007f 0x0000007f 0x0000007f 0x8000001f 0x8000001f 0x8000001f 0x8000001f 0x0000005f
96: 0x80000010 Hit4 0x0000007f 0x0000007f 0x0000007f 0x8000001f 0x8000001f 0x8000001f 0x8000001f 0x0000005f
97: 0x80000014 Hit5 0x0000007f 0x0000007f 0x0000007f 0x8000001f 0x8000001f 0x8000001f 0x8000001f 0x0000005f
98: 0x80000018 Hit6 0x0000007f 0x0000007f 0x0000007f 0x8000001f 0x8000001f 0x8000001f 0x8000001f 0x0000005f
99: 0x8000000c Hit3 0x0000007f 0x0000007f 0x0000007f 0x8000001f 0x8000001f 0x8000001f 0x8000001f 0x0000005f
100: 0x0000006c Miss 0x0000007f 0x0000007f 0x0000007f 0x0000007f 0x8000001f 0x8000001f 0x8000001f 0x0000005f
101: 0x80000010 Hit4 0x0000007f 0x0000007f 0x0000007f 0x0000007f 0x8000001f 0x8000001f 0x8000001f 0x0000005f
102: 0x80000014 Hit5 0x0000007f 0x0000007f 0x0000007f 0x0000007f 0x8000001f 0x8000001f 0x8000001f 0x0000005f
103: 0x80000018 Hit6 0x0000007f 0x0000007f 0x0000007f 0x0000007f 0x8000001f 0x8000001f 0x8000001f 0x0000005f
Hits = 68, Accesses = 103, Hit ratio = 0.660194
```

16 Bit:

CODE:

```
#include <stdio.h>
```

```
int tag[16];
```

```
int main()
```

```
{
```

```
    int addr;
```

```
    int i, j, t;
```

```
    int hits, accesses;
```

```
    FILE *fp;
```



```
fp = fopen("trace.txt", "r");
hits = 0;
accesses = 0;
while (fscanf(fp, "%x", &addr) > 0) {
    /* simulate a direct-mapped cache with 8 words */
    accesses += 1;
    printf("%3d: 0x%08x ", accesses, addr);
    i = (addr >> 2) & 15;
    t = addr | 0x1f;
    if (tag[i] == t) {
        hits += 1;
        printf("Hit%d ", i);
    } else {
        /* allocate entry */
        printf("Miss ");
        tag[i] = t;
    }
    for (i = 0; i < 16; i++)
        printf("0x%08x ", tag[i]);
    printf("\n");
}
printf("Hits = %d, Accesses = %d, Hit ratio = %f\n", hits, accesses,
((float)hits)/accesses);
close(fp);
}
```

OUTPUT:

```
1: 0x80000000 Miss 0x8000001f 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
2: 0x80000004 Miss 0x8000001f 0x8000001f 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
3: 0x80000008 Miss 0x8000001f 0x8000001f 0x8000001f 0x00000000 0x00000000 0x00000000 0x00000000
0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
4: 0x8000000c Miss 0x8000001f 0x8000001f 0x8000001f 0x8000001f 0x00000000 0x00000000 0x00000000
0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
5: 0x00000020 Miss 0x8000001f 0x8000001f 0x8000001f 0x8000001f 0x00000000 0x00000000 0x00000000
0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
6: 0x80000010 Miss 0x8000001f 0x8000001f 0x8000001f 0x8000001f 0x8000001f 0x00000000 0x00000000
0x0000003f 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
7: 0x80000014 Miss 0x8000001f 0x8000001f 0x8000001f 0x8000001f 0x8000001f 0x8000001f 0x00000000
0x0000003f 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
8: 0x80000018 Miss 0x8000001f 0x8000001f 0x8000001f 0x8000001f 0x8000001f 0x8000001f 0x00000000
0x0000003f 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
9: 0x8000000c Hit3 0x8000001f 0x8000001f 0x8000001f 0x8000001f 0x8000001f 0x8000001f 0x00000000
0x0000003f 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
10: 0x00000024 Miss 0x8000001f 0x8000001f 0x8000001f 0x8000001f 0x8000001f 0x8000001f 0x00000000
```

[illegible]

[illegible]

[illegible]



32 Bit:

CODE:

```
#include <stdio.h>
#include <stdio.h>
int tag[32];
int main( )
{
    int addr;
    int i, j, t;
    int hits, accesses;
    FILE *fp;

    fp = fopen("trace.txt", "r");
    hits = 0;
    accesses = 0;
    while (fscanf(fp, "%x", &addr) > 0) {
        /* simulate a direct-mapped cache with 8 words */
        accesses += 1;
        printf("%3d: 0x%08x ", accesses, addr);
        i = (addr >> 2) & 31;
        t = addr | 0x1f;
        if (tag[i] == t) {
            hits += 1;
            printf("Hit%d ", i);
        } else {
            /* allocate entry */
            printf("Miss ");
            tag[i] = t;
        }
        for (i = 0; i < 32; i++)
            printf("0x%08x ", tag[i]);
        printf("\n");
    }
    printf("Hits = %d, Accesses = %d, Hit ratio = %f\n", hits, accesses,
        ((float)hits)/accesses);
    close(fp);
}
```

[illegible]

[illegible]



```
0x0000003f 0x0000003f 0x0000003f 0x0000003f 0x0000003f 0x0000003f 0x0000003f 0x0000003f 0x0000005f 0x0000005f
0x0000005f 0x0000005f 0x0000005f 0x0000005f 0x0000005f 0x0000005f 0x0000007f 0x0000007f 0x0000007f 0x0000007f
0x00000000 0x00000000 0x00000000 0x00000000
102: 0x80000014 Hit5 0x8000001f 0x8000001f 0x8000001f 0x8000001f 0x8000001f 0x8000001f 0x8000001f 0x00000000
0x0000003f 0x0000003f 0x0000003f 0x0000003f 0x0000003f 0x0000003f 0x0000003f 0x0000003f 0x0000005f 0x0000005f
0x0000005f 0x0000005f 0x0000005f 0x0000005f 0x0000005f 0x0000005f 0x0000007f 0x0000007f 0x0000007f 0x0000007f
0x00000000 0x00000000 0x00000000 0x00000000
103: 0x80000018 Hit6 0x8000001f 0x8000001f 0x8000001f 0x8000001f 0x8000001f 0x8000001f 0x8000001f 0x00000000
0x0000003f 0x0000003f 0x0000003f 0x0000003f 0x0000003f 0x0000003f 0x0000003f 0x0000003f 0x0000005f 0x0000005f
0x0000005f 0x0000005f 0x0000005f 0x0000005f 0x0000005f 0x0000005f 0x0000007f 0x0000007f 0x0000007f 0x0000007f
0x00000000 0x00000000 0x00000000 0x00000000
Hits = 76, Accesses = 103, Hit ratio = 0.737864
```

Comparison Table

Number of Bits	Hits	Accesses	Hit Ratio
8	68	103	0.660194
16	72	103	0.699029
32	76	103	0.737864

CONCLUSION:

Cache memory is the closest to the CPU, it reduces the access time from Main Memory. There are 3 ways to map cache to Main memory: Direct mapping, Associative mapping, and Set-Associative mapping. In Direct mapping, each memory block is assigned to a specific line in the cache. It is a simple way to map blocks but it has a lower cache hit ratio, as there is only one cache line available in a set. Every time a new memory is referenced to the same set, the cache line is replaced, this is the drawback of Direct Mapping.

As seen from the comparison table above which is derived from our program output, as we go on increasing the number of bits in cache memory, the hit ratio also increases.