**Batch: B3 Roll No.: 121**

**Experiment / assignment / tutorial No.\_\_\_6\_\_\_**

**Grade: AA / AB / BB / BC / CC / CD /DD**

**Signature of the Staff In-charge with date**

|  |
| --- |
| **TITLE: Implementation of LRU Page Replacement Algorithm.** |

**AIM:** The LRU algorithm replaces the least recently used that is the last accessed memory block from user.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

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

CO1 – Describe and define the structure of a computer with buses structure and detail working of the arithmetic logic unit and its sub modules.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Books/ Journals/ Websites referred:**

1. Carl Hamacher, Zvonko Vranesic and Safwat Zaky, “Computer Organization”, Fifth Edition, TataMcGraw-Hill.
2. William Stallings, “Computer Organization and Architecture: Designing for Performance”, Eighth Edition, Pearson.
3. <https://simple2code.com/c-tutorial/lru-page-replacement-algorithm-in-c/>
4. <https://www.baeldung.com/cs/virtual-memory-why>

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Pre Lab/ Prior Concepts:**

It follows a simple logic, while replacing it will replace that page which has least recently used out of all.

a) A hit is said to be occurred when a memory location requested is already in the cache.

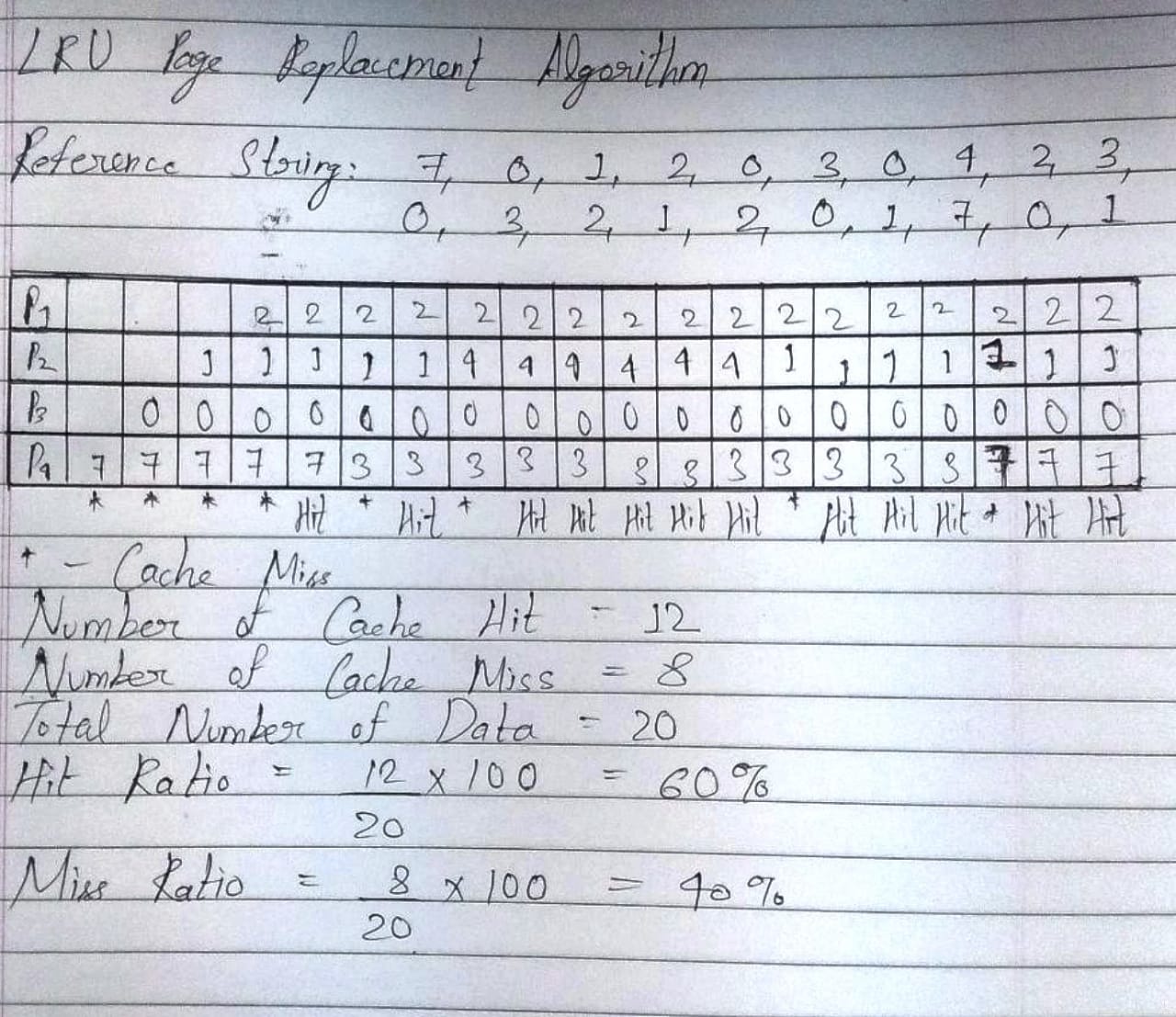
b) When cache is not full, the number of blocks is added.

c) When cache is full, the block is replaced which is recently used

**Algorithm:**

1. Start
2. Get input as memory block to be added to cache
3. Consider an element of the array
4. If cache is not full, add element to the cache array
5. If cache is full, check if element is already present
6. If it is hit is incremented
7. If not, element is added to cache removing least recently used element
8. Repeat step 3 to 7 for remaining elements
9. Display the cache at very instance of step 8
10. Print hit ratio
11. End

**Example:**



Code:

#include<stdio.h>

int LRU(int Time[], int num\_f)

{

int pos = 0, i, min = Time[0];

for(i = 1; i < num\_f; i++)

{

if(Time[i] < min)

{

min = Time[i];

pos = i;

}

}

return pos;

}

void main()

{

int i, j, num\_p, orig\_num\_p, num\_f, pg[100], fr[10], Time[10], c = 0, flag1, flag2, miss = 0, hit, pos;

float miss\_ratio, hit\_ratio;

char ch;

do

{

printf("\nEnter the number of pages (maximum 100): ");

scanf("%d", &num\_p);

orig\_num\_p = num\_p;

for(i = 0; i < num\_p; i++)

{

printf("\nFor page %d, enter data: ", (i+1));

scanf("%d", &pg[i]);

}

printf("\nEnter the number of frames (maximum 10): ");

scanf("%d", &num\_f);

for(i = 0; i < num\_f; i++)

fr[i] = -1;

for(i = 0; i < num\_p; i++)

{

flag1 = flag2 = 0;

for(j = 0; j < num\_f; j++)

{

if(fr[j] == pg[i])

{

c++;

Time[j] = c;

flag1 = flag2 = 1;

break;

}

}

if(flag1 == 0)

{

for(j = 0; j < num\_f; j++)

{

if(fr[j] == -1)

{

c++;

miss++;

fr[j] = pg[i];

Time[j] = c;

flag2 = 1;

break;

}

}

}

if(flag2 == 0)

{

pos = LRU(Time, num\_f);

c++;

miss++;

fr[pos] = pg[i];

Time[pos] = c;

}

printf("\n");

for(j = 0; j < num\_f; j++)

printf("%d\t", fr[j]);

}

printf("\nThe number of Cache Miss is %d.", miss);

hit = orig\_num\_p - miss;

printf("\nThe number of Cache Hit is %d.", hit);

miss\_ratio = (100\*miss)/orig\_num\_p;

printf("\nThe Miss Ratio is %f.", miss\_ratio);

hit\_ratio = 100 - miss\_ratio;

printf("\nThe Hit Ratio is %f.", hit\_ratio);

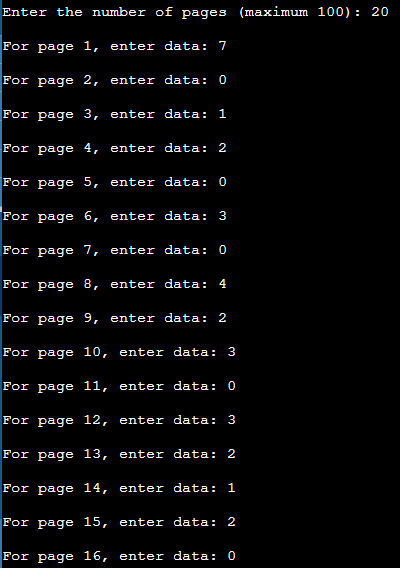
printf("\nDo you want to try again?\nPress 'Y' otherwise press any other character.\nEnter your choice: ");

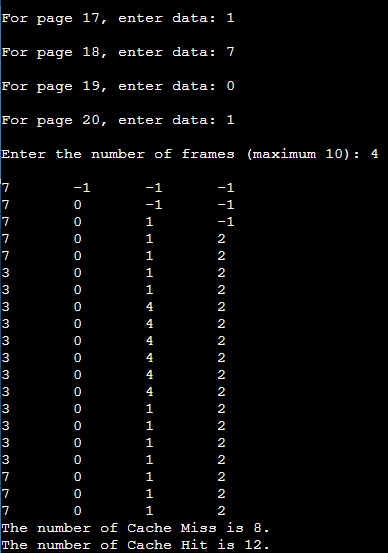
scanf("%s", &ch);

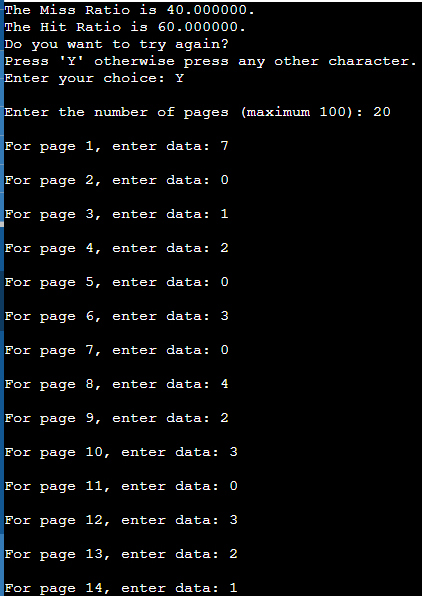
}while(ch=='Y');

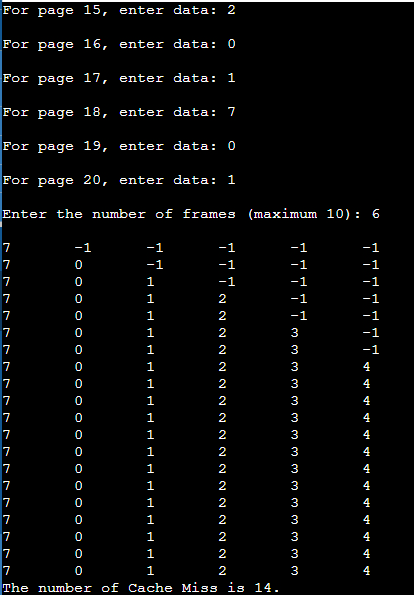
}

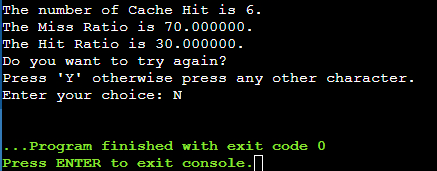
Output:











**Post Lab Descriptive Questions**

**1. Define hit rate and miss ratio?**

Ans. The hit rate is defined as the number of cache hits divided by the number of memory requests made to the cache during a specified time, normally calculated as a percentage.

The miss ratio is the number of cache misses divided by the total number of memory requests made to the cache.

A cache hit occurs if the required data is present in the cache when a memory request is made to it. A cache miss occurs if the required data is not present in the cache when a memory request is made to it.

**2. What is the need for virtual memory**?

Ans. A computer can address more memory than the amount physically installed on the system. This extra memory is called Virtual memory and it is a section of a hard disc that’s set up to emulate the computer’s RAM.

The main advantage of this scheme is that programs can be larger than the physical memory available in the computer. Virtual memory serves two purposes – it allows one to extend the use of physical memory using disc. Second, it allows one to have memory protection, because each virtual address is translated to a physical address.

* 1. Memory Space Problem

Virtual Memory makes it easy to share code. Therefore, one does not need to keep several copies of the same code. With virtual memory, different virtual addresses can map to the same location in the physical memory. Consequently, multiple copies of the same code need not be stored in the main memory. RAM is very costly. As a result, the use of a very large RAM is not feasible. Assigning a virtual memory for each program and mapping addresses to the disk eliminates space problems.

* 1. Data Security

In general, a program can guess another program’s physical address and gain access to sensitive and secret data. If an OS uses the virtual memory technique, even if some programs have access to same address, they all have different mappings. This ensures that they will access different addresses in the RAM or the disk. This is how virtual memory enables data security.

* 1. Memory Fragmentation and Errors

Virtual memory facilitates each program with its own mapping. The data space won’t have to be continuous, and each program can store data wherever it wants. It also facilitates debugging and provides options for checking various features, like unallocated memory and null pointers.

**Conclusion**

Thus, in this experiment, the concept Least Recently Used Page Replacement Algorithm has been learnt and implemented in C language. This algorithm is used by a computer’s cache memory management system to facilitate faster and efficient memory operations.

**Date: \_\_\_18-11-22\_\_\_ Signature of faculty in-charge**