Name : Ayush. D. Chougule
Rollno : 06
PRN: 1032231720.

13/10/25

Assignment - 09

* Aim: To implement the least recently use (LRU) page replacement algorithm for memory management.

* objectives:

i) To implement and understand the LRU page replacement algorithm.
ii) To compare performance of FIFO < LRU and optimal policy algorithms.

* Theory:

i] Explain Demand paging and LRU algorithm with an example write it's advantages and disadvantages.

→ i) Demand Paging: Is a memory management technique where pages are loaded into main memory only when required during execution, reducing memory usage. when a page not in memory is referenced, a page fault occurs & required page is fetched from secondary storage.

ii) LRU (Least recently used) Algorithm:
LRU replaces the least recently used page, when a new page needs to be loaded.
Eg: Page refrence string → 1, 2, 3, 1, 4, 5 with 3

Load pages ⇒ 1, 2, 3 ⟶ All frames full
Access Page 1 again → Recently used → no replace.

Page four (4) causes replacement of least recently used page.
Page five (5) replaces next least used page (page 3).

Final pages in memory ⇒ 1, 4, 5
LRU ensures recently used pages remain in memory.

Advantages

- efficient memory utilization
- Reduces unnecessary Page loading
- Performs well with locality of reference

Disadvantages.

- Higher overhead
- Slower in software.
- may perform poorly, acces changes.

* Input:
* Output:

} AT LAST

* Conclusion: Demand paging loads pages on demand, reducing m use. LRU keeps recently used pages, improving h rates lowering latency.

* FAQ:

1] What is page fault?

→ A page fault occurs when referenced page not in main me the OS loads it from secondary storage, causing a trap and potential delay. Handling involves saving state, invoking replacement if needed, updating tables, and restarting the fa instruction.

2]. Page reference string ⇒ 1, 2, 3, 4, 2, 1, 5, 6, 2, 1, 2, 3, 7, 6, 3, 2, 1, 2, 3, 6.

Calculate no of page fault for LRU, FIFO, optimal page replacement algorithm.

→ Reference string ⇒ 1, 2, 3, 4, 2, 1, 5, 6, 2, 1, 2, 3, 7, 6, 3, 2, 1, 2, 3, 6.

with 3 frames.

FIFO faults = 16 , LRU faults = 15.
optimal faults = 11

Optimal is lower because it replaces the page with farthest future use. LRU benifits from temporal locality.

FIFO may evict recently used pages causing more faults.

3] Given logical address = 2700, page size = 1KB, find page number and offset.

→

Logical address = 2700
page size = 1 kb (1024 bytes)
page number = 2 (2700/1024)

offset = (2700 − (2 × 1024))
= 2700 − 2048
= 652

- Input:

Enter number of pages : 10
Enter the page reference string : 52 3 2 71 8 4 5 1
Enter number of frames : 3

- Output:

5	5	5	5	7	7	7	4	4	4
	2	2	2	2	2	3	3	3	1
		3	3	3	1	1	1	5	5
F	F	F	H	F	F	F	F	F	F

total pages : 10
total page faults : 9
total Hits : 1
fault Ratio : 0.9
Hit ratio : 0.1

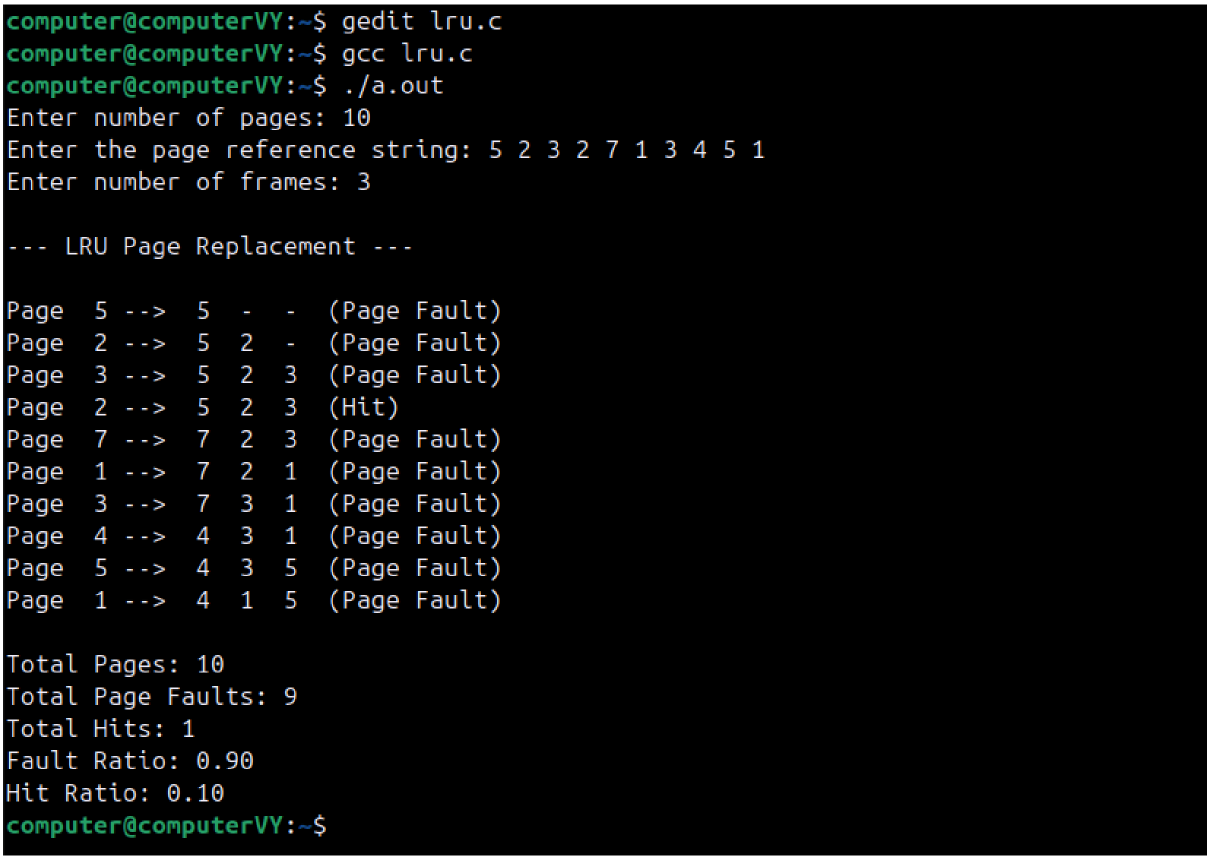
CODE:

```
#include <stdio.h>

int findLRU(int time[], int n) {
    int i, min = time[0], pos = 0;
    for (i = 1; i < n; i++) {
        if (time[i] < min) {
            min = time[i];
            pos = i;
        }
    }
    return pos;
}

int main() {
    int pages[50], frames[10], time[10];
    int n, f, i, j, counter = 0, pos;
    int faults = 0, hits = 0;
    float hit_ratio, fault_ratio;

    printf("Enter number of pages: ");
    scanf("%d", &n);

    printf("Enter the page reference string: ");
    for (i = 0; i < n; i++)
        scanf("%d", &pages[i]);

    printf("Enter number of frames: ");
```

```
scanf("%d", &f);

for (i = 0; i < f; i++) {
    frames[i] = -1;
    time[i] = 0;
}

printf("\n--- LRU Page Replacement ---\n");

for (i = 0; i < n; i++) {
    int found = 0;

    // Check if page is already in frames
    for (j = 0; j < f; j++) {
        if (frames[j] == pages[i]) {
            found = 1;
            counter++;
            time[j] = counter; // Update recent use
            hits++;
            break;
        }
    }

    // If page not found, replace LRU
    if (!found) {
        int empty = -1;
        for (j = 0; j < f; j++) {
            if (frames[j] == -1) {
                empty = j;
                break;
            }
        }

        counter++;
        if (empty != -1) { // Empty frame available
            frames[empty] = pages[i];
            time[empty] = counter;
        } else { // No empty frame, replace LRU
            pos = findLRU(time, f);
            frames[pos] = pages[i];
            time[pos] = counter;
        }
        faults++;
    }

    // Print current frame status
    printf("\nPage %2d --> ", pages[i]);
    for (j = 0; j < f; j++) {
        if (frames[j] == -1)
            printf(" - ");
        else
            printf("%2d ", frames[j]);
    }

    if (found)
        printf(" (Hit)");
    else
        printf(" (Page Fault)");
}

hit_ratio = (float)hits / n;
fault_ratio = (float)faults / n;
```

```
    printf("\n\nTotal Pages: %d", n);

    printf("\nTotal Page Faults: %d", faults);

    printf("\nTotal Hits: %d", hits);

    printf("\nFault Ratio: %.2f", fault_ratio);

    printf("\nHit Ratio: %.2f\n", hit_ratio);


    return 0;

}
```

OUTPUT:



```
computer@computerVY:~$ gedit lru.c
computer@computerVY:~$ gcc lru.c
computer@computerVY:~$ ./a.out
Enter number of pages: 10
Enter the page reference string: 5 2 3 2 7 1 3 4 5 1
Enter number of frames: 3

--- LRU Page Replacement ---

Page  5 -->  5  -  -  (Page Fault)
Page  2 -->  5  2  -  (Page Fault)
Page  3 -->  5  2  3  (Page Fault)
Page  2 -->  5  2  3  (Hit)
Page  7 -->  7  2  3  (Page Fault)
Page  1 -->  7  2  1  (Page Fault)
Page  3 -->  7  3  1  (Page Fault)
Page  4 -->  4  3  1  (Page Fault)
Page  5 -->  4  3  5  (Page Fault)
Page  1 -->  4  1  5  (Page Fault)

Total Pages: 10
Total Page Faults: 9
Total Hits: 1
Fault Ratio: 0.90
Hit Ratio: 0.10
computer@computerVY:~$
```