Assignment

Name: Aditi Dipak Pawar.

Roll no.:16 PRN:12310277

Topic: **Implement page replacement algorithms-**

**1) FIFO     2)Optimal     3) LRU      4) Clock**

**Code:**

#include <stdio.h>

#include <limits.h>

#include <stdbool.h>

#define MAX 20

// Function to check if a page is present in memory

bool isPresent(int frames[], int n, int page) {

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

if (frames[i] == page) {

return true;

}

}

return false;

}

// Function to implement FIFO Page Replacement

void fifo(int pages[], int n, int capacity) {

int frames[MAX], pageFaults = 0, index = 0;

for (int i = 0; i < capacity; i++) {

frames[i] = -1;

}

printf("\nFIFO Page Replacement\n");

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

if (!isPresent(frames, capacity, pages[i])) {

frames[index] = pages[i];

index = (index + 1) % capacity;

pageFaults++;

}

printf("Page: %d\tFrames: ", pages[i]);

for (int j = 0; j < capacity; j++) {

if (frames[j] != -1)

printf("%d ", frames[j]);

else

printf("- ");

}

printf("\n");

}

printf("Total Page Faults (FIFO): %d\n", pageFaults);

}

// Function to implement Optimal Page Replacement

void optimal(int pages[], int n, int capacity) {

int frames[MAX], pageFaults = 0;

for (int i = 0; i < capacity; i++) {

frames[i] = -1;

}

printf("\nOptimal Page Replacement\n");

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

if (!isPresent(frames, capacity, pages[i])) {

int farthest = i, index = -1;

for (int j = 0; j < capacity; j++) {

int k;

for (k = i + 1; k < n; k++) {

if (frames[j] == pages[k]) {

if (k > farthest) {

farthest = k;

index = j;

}

break;

}

}

if (k == n) {

index = j;

break;

}

}

if (index == -1)

index = 0;

frames[index] = pages[i];

pageFaults++;

}

printf("Page: %d\tFrames: ", pages[i]);

for (int j = 0; j < capacity; j++) {

if (frames[j] != -1)

printf("%d ", frames[j]);

else

printf("- ");

}

printf("\n");

}

printf("Total Page Faults (Optimal): %d\n", pageFaults);

}

// Function to implement LRU Page Replacement

void lru(int pages[], int n, int capacity) {

int frames[MAX], pageFaults = 0, time[MAX];

for (int i = 0; i < capacity; i++) {

frames[i] = -1;

time[i] = 0;

}

printf("\nLRU Page Replacement\n");

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

int least = i, index = -1;

if (!isPresent(frames, capacity, pages[i])) {

for (int j = 0; j < capacity; j++) {

if (frames[j] == -1) {

index = j;

break;

}

if (time[j] < time[least]) {

least = time[j];

index = j;

}

}

frames[index] = pages[i];

pageFaults++;

}

time[index] = i; // Update time of reference

printf("Page: %d\tFrames: ", pages[i]);

for (int j = 0; j < capacity; j++) {

if (frames[j] != -1)

printf("%d ", frames[j]);

else

printf("- ");

}

printf("\n");

}

printf("Total Page Faults (LRU): %d\n", pageFaults);

}

// Function to implement Clock Page Replacement

void clockReplace(int pages[], int n, int capacity) {

int frames[MAX], pageFaults = 0, refBits[MAX], index = 0;

for (int i = 0; i < capacity; i++) {

frames[i] = -1;

refBits[i] = 0;

}

printf("\nClock Page Replacement\n");

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

if (!isPresent(frames, capacity, pages[i])) {

while (refBits[index] == 1) {

refBits[index] = 0;

index = (index + 1) % capacity;

}

frames[index] = pages[i];

refBits[index] = 1;

pageFaults++;

index = (index + 1) % capacity;

} else {

int pageIndex;

for (pageIndex = 0; pageIndex < capacity; pageIndex++) {

if (frames[pageIndex] == pages[i]) {

refBits[pageIndex] = 1;

break;

}

}

}

printf("Page: %d\tFrames: ", pages[i]);

for (int j = 0; j < capacity; j++) {

if (frames[j] != -1)

printf("%d ", frames[j]);

else

printf("- ");

}

printf("\n");

}

printf("Total Page Faults (Clock): %d\n", pageFaults);

}

int main() {

int pages[] = {7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2};

int n = sizeof(pages) / sizeof(pages[0]);

int capacity = 3;

fifo(pages, n, capacity);

optimal(pages, n, capacity);

lru(pages, n, capacity);

clockReplace(pages, n, capacity);

return 0;

}

OUTPUT:

FIFO Page Replacement

Page: 7 Frames: 7 - -

Page: 0 Frames: 7 0 -

Page: 1 Frames: 7 0 1

Page: 2 Frames: 2 0 1

Page: 0 Frames: 2 0 1

Page: 3 Frames: 2 3 1

Page: 0 Frames: 2 3 0

Page: 4 Frames: 4 3 0

Page: 2 Frames: 4 2 0

Page: 3 Frames: 4 2 3

Page: 0 Frames: 0 2 3

Page: 3 Frames: 0 2 3

Page: 2 Frames: 0 2 3

Total Page Faults (FIFO): 9

Optimal Page Replacement

Page: 7 Frames: 7 - -

Page: 0 Frames: 7 0 -

Page: 1 Frames: 7 0 1

Page: 2 Frames: 2 0 1

Page: 0 Frames: 2 0 1

Page: 3 Frames: 2 3 1

Page: 0 Frames: 2 0 1

Page: 4 Frames: 4 0 1

Page: 2 Frames: 4 0 2

Page: 3 Frames: 4 3 2

Page: 0 Frames: 0 3 2

Page: 3 Frames: 0 3 2

Page: 2 Frames: 0 3 2

Total Page Faults (Optimal): 8

LRU Page Replacement

Page: 7 Frames: 7 - -

Page: 0 Frames: 7 0 -

Page: 1 Frames: 7 0 1

Page: 2 Frames: 2 0 1

Page: 0 Frames: 2 0 1

Page: 3 Frames: 2 3 1

Page: 0 Frames: 2 0 1

Page: 4 Frames: 4 0 1

Page: 2 Frames: 4 0 2

Page: 3 Frames: 4 3 2

Page: 0 Frames: 0 3 2

Page: 3 Frames: 0 3 2

Page: 2 Frames: 0 3 2

Total Page Faults (LRU): 8

Clock Page Replacement

Page: 7 Frames: 7 - -

Page: 0 Frames: 7 0 -

Page: 1 Frames: 7 0 1

Page: 2 Frames: 2 0 1

Page: 0 Frames: 2 0 1

Page: 3 Frames: 2 3 1

Page: 0 Frames: 2 3 0

Page: 4 Frames: 4 3 0

Page: 2 Frames: 2 3 0

Page: 3 Frames: 2 3 0

Page: 0 Frames: 2 3 0

Page: 3 Frames: 2 3 0

Page: 2 Frames: 2 3 0

Total Page Faults (Clock): 8