Source Code:

#include <stdio.h>

void fifo(int nf, int np, int pages[]) {

int frames[nf], hit = 0, fault = 0, i, j;

for (i = 0; i < nf; i++) frames[i] = -1;

printf("FIFO:\n");

for (i = 0; i < nf; i++) printf("F%d\t", i + 1);

printf("\n");

for (i = 0; i < np; i++) {

int page\_found = 0;

for (j = 0; j < nf; j++) {

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

page\_found = 1;

hit++;

break; } }

if (!page\_found) {

for (j = 0; j < nf; j++) {

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

fault++;

frames[j] = pages[i];

break; } }

if (j == nf) {

fault++;

for (j = 0; j < nf - 1; j++) frames[j] = frames[j + 1];

frames[nf - 1] = pages[i]; } }

for (j = 0; j < nf; j++) {

if (frames[j] == -1) printf("-\t");

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

}

printf("\n");

}

printf("Number of Page Faults: %d\n", fault);

printf("Number of Page Hits: %d\n", hit);

printf("Hit Ratio: %f%%\n", (float)hit / (float)(fault + hit) \* 100);}

void lru(int nf, int np, int pages[]) {

int frames[nf], timestamp[nf], hit = 0, fault = 0, i, j;

// Initialize frames and timestamps to -1, indicating empty frames

for (i = 0; i < nf; i++) {

frames[i] = -1;

timestamp[i] = -1; }

// Print a header indicating the LRU algorithm

printf("\nLRU:\n");

// Iterate through each page reference

for (i = 0; i < np; i++) {

int found = 0;

int page = pages[i];

// Check if the page is already in frames

for (j = 0; j < nf; j++) {

if (frames[j] == page) {

// If found, mark found as 1, increment hit count, update timestamp, and break the loop

found = 1;

hit++;

timestamp[j] = i;

break; } }

// If the page is not found in frames

if (!found) {

// Find the index of the frame with the minimum timestamp

int min = timestamp[0];

int index = 0;

for (j = 1; j < nf; j++) {

if (timestamp[j] < min) {

min = timestamp[j];

index = j; } }

// Replace the page at the index with the minimum timestamp

frames[index] = page;

timestamp[index] = i;

fault++; }

// Print the current state of frames after each page reference

printf("%d : \t", page);

for (j = 0; j < nf; j++)

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

printf("\n"); }

// Print the total number of page faults, page hits, and hit ratio

printf("\nNumber of Page Faults: %d\n", fault);

printf("Number of Page Hits: %d\n", hit);

printf("Hit Ratio: %.2f%%\n", (float)hit / (float)(fault + hit) \* 100);}

void optimal(int nf, int np, int pages[]) {

int fr[nf], count[nf], fault = 0, hit = 0, dist = 0, k = 0, i, j;

for (i = 0; i < nf; i++) {

count[i] = 0;

fr[i] = -1; }

printf("\nOPTIMAL:\n");

for (i = 0; i < np; i++) {

int flag = 0;

for (j = 0; j < nf; j++) {

if (pages[i] == fr[j]) {

flag = 1;

hit++;

break; } }

if (!flag && k < nf) {

fault++;

fr[k] = pages[i];

k++;

} else if (!flag && k == nf) {

fault++;

for (j = 0; j < nf; j++) {

int current = fr[j];

for (int c = i; c < np; c++) {

if (current != pages[c]) count[j]++;

else break; } }

int max\_count = 0, p;

for (int m = 0; m < nf; m++) {

if (count[m] > max\_count) {

max\_count = count[m];

p = m; } }

fr[p] = pages[i]; }

printf("\npage %d frame\t", pages[i]);

for (j = 0; j < nf; j++) printf("%d\t", fr[j]); }

printf("\nNumber of Page Faults: %d\n", fault);

printf("Number of Page Hits: %d\n", hit);

printf("Hit Ratio: %f%%\n", (float)hit / (float)(fault + hit) \* 100);}

int main() {

int nf, np, i;

printf("Enter no of frames: ");

scanf("%d", &nf);

printf("Enter no of page references: ");

scanf("%d", &np);

int pages[np];

printf("Enter page references: \n");

for (i = 0; i < np; i++) scanf("%d", &pages[i]);

fifo(nf, np, pages);

lru(nf, np, pages);

optimal(nf, np, pages);

return 0; }



