\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Operating System Lab

CEN-493

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Program - 14

Code :-

#include <iostream>

#include <math.h>

#include <stack>

#include <algorithm>

using namespace std;

struct LRU\_State

{

    vector<int> state;

    bool isFault;

    int top;

};

void printLines()

{

    for (int i = 0; i < 100; i++)

    {

        cout << "-";

    }

    cout << "\n";

}

void Print(vector<LRU\_State> &allStates, int pageFaults)

{

    printLines();

    cout << "Page Replacement Table\n";

    printLines();

    printLines();

    cout << "Page Reference\n";

    for (int state = 0; state < allStates.size(); state++)

    {

        cout << "|" << allStates[state].top << "|\t";

    }

    cout << "\n\n";

    for (int state = allStates[0].state.size() - 1; state >= 0; state--)

    {

        for (int i = 0; i < allStates.size(); i++)

        {

            if (allStates[i].state[state] == -1)

                cout << "|"

                     << "\_"

                     << "|\t";

            else

                cout << "|" << allStates[i].state[state] << "|\t";

        }

        cout << "\n";

    }

    cout << "\n";

    for (int state = 0; state < allStates.size(); state++)

    {

        if (allStates[state].isFault)

        {

            cout << "|"

                 << "Miss"

                 << "\t";

        }

        else

        {

            cout << "|"

                 << "Hit"

                 << "\t";

        }

    }

    cout << "\n";

    printLines();

    cout << "Total Page Faults : " << pageFaults << "\n";

    double averagePageFaults = pageFaults / (double)allStates.size();

    cout << "Average Page Faults : " << averagePageFaults << "\n";

    printLines();

}

void rotate(vector<int> &arr, int x, int y)

{

    int first = arr[x];

    for (int i = x; i < y; i++)

    {

        arr[i] = arr[i + 1];

    }

    arr[y] = first;

}

void Page\_Replacement\_LRU(int noOfPageFrames, vector<int> &pageReferences)

{

    vector<LRU\_State> allStates;

    vector<int> frame(noOfPageFrames, -1), lru(noOfPageFrames, -1);

    int pageFaults = 0, top = 0, prIndex = 0;

    for (prIndex = 0; top != noOfPageFrames; prIndex++)

    {

        bool isFind = false;

        for (int fIndex = 0; fIndex < top; fIndex++)

        {

            if (frame[fIndex] == pageReferences[prIndex])

            {

                isFind = true;

                break;

            }

        }

        LRU\_State newState;

        if (isFind)

        {

            for (int lruIndex = 0; lruIndex < top; lruIndex++)

            {

                if (lru[lruIndex] == pageReferences[prIndex])

                {

                    rotate(lru, lruIndex, top - 1);

                    break;

                }

            }

            newState.isFault = 0;

        }

        else

        {

            frame[top] = pageReferences[prIndex];

            lru[top] = pageReferences[prIndex];

            newState.isFault = 1;

            pageFaults++;

            top++;

        }

        newState.top = pageReferences[prIndex];

        newState.state = frame;

        allStates.push\_back(newState);

    }

    for (prIndex; prIndex < pageReferences.size(); prIndex++)

    {

        bool isFind = 0;

        for (int fIndex = 0; fIndex < noOfPageFrames; fIndex++)

        {

            if (frame[fIndex] == pageReferences[prIndex])

            {

                isFind = true;

                break;

            }

        }

        if (isFind)

        {

            for (int lruIndex = 0; lruIndex < noOfPageFrames; lruIndex++)

            {

                if (lru[lruIndex] == pageReferences[prIndex])

                {

                    rotate(lru, lruIndex, noOfPageFrames - 1);

                    break;

                }

            }

            LRU\_State newState;

            newState.isFault = 0;

            newState.top = pageReferences[prIndex];

            newState.state = frame;

            allStates.push\_back(newState);

        }

        else

        {

            LRU\_State newState;

            newState.isFault = 1;

            pageFaults++;

            newState.top = pageReferences[prIndex];

            int leastUsed = lru[0];

            for (int fIndex = 0; fIndex <= noOfPageFrames; fIndex++)

            {

                if (frame[fIndex] == leastUsed)

                {

                    frame[fIndex] = pageReferences[prIndex];

                    lru[0] = pageReferences[prIndex];

                    break;

                }

            }

            rotate(lru, 0, noOfPageFrames - 1);

            newState.state = frame;

            allStates.push\_back(newState);

        }

    }

    Print(allStates, pageFaults);

}

int main()

{

    system("cls");

    printLines();

    cout << "Vicky\_Gupta\_20BCS070\n";

    printLines();

    cout << "Least Recently Used Page Replacement Algorithm\n";

    printLines();

    printLines();

    int noOfPageFrames;

    cout << "Enter The No Of Page Frames \n";

    cin >> noOfPageFrames;

    int noOfPageReference;

    cout << "Enter The No Of Page Reference\n";

    cin >> noOfPageReference;

    vector<int> pageReferences(noOfPageReference);

    cout << "Enter The Page References\n";

    for (int i = 0; i < noOfPageReference; i++)

    {

        cin >> pageReferences[i];

    }

    Page\_Replacement\_LRU(noOfPageFrames, pageReferences);

    return 0;

}

Output :-

A picture containing text

Description automatically generated