**CAOS Course Project**

**“Operating System Phase 1”**

Guide – Prof. Saraswati Patil

Group No. - SY 41

Group Member’s:-

|  |  |  |
| --- | --- | --- |
| Roll No. | Name | GR No. |
| 82 | Sanskruti Kakade | 12020203 |
| 85 | Samarth Misal | 12020241 |
| 96 | Karan Kadam | 12020196 |
| 97 | Snehal Kapadnis | 12020194 |
| 98 | Neha Gandhmal | 12020214 |

Source Code –

# include <iostream>

# include <fstream>

# include <cstring>

# include <stdlib.h>

using namespace std;

# define MAX 40

char EM[100][4]; //External Memory

char IR[4]; //Instruction Register

int IC; //Instruction Counter

int C; //Toggle Button

char R[4]; //General Purpose Register

char BUFF[MAX]; //Reading & Writing data

int SI, location; //Service Interrupt

int i, j, k,blockCount; //Iterators

fstream inFile, outFile; //Input and Output file references

void INIT();

void LOAD();

void STARTEXECUTION();

void EXEUTEUSERPROGRAM();

void MOS();

void READ(int loc);

void LADD(int loc);

void WRITE();

void CLEARBUFF();

void TERMINATE();

int main() {

    LOAD();

    return 0;

}

void INIT() {

    for(int i = 0 ; i <= 99 ; i++) {

        for(int j = 0 ; j <= 3 ; j++) {

            EM[i][j] = '#';

            //cout << EM[i][j] ;

        }

    }

}

void LOAD() {

    inFile.open("input.txt",ios::in);

    if(inFile) {

        while(!inFile.eof()) {

            inFile.getline(BUFF, 41);

            inFile.ignore(0, '\n');

            cout << endl << BUFF << endl ;

            if(BUFF[0] == '$' && BUFF[1] == 'A' && BUFF[2] =='M' && BUFF[3] == 'J')

            {

                cout << "\n Process started..! \n" ;

                blockCount = 0;

                INIT();

            }

            else if(BUFF[0] == '$' && BUFF[1] == 'D' && BUFF[2] =='T' && BUFF[3] == 'A')

            {

                STARTEXECUTION();

            }

            else if(BUFF[0] == '$' && BUFF[1] == 'E' && BUFF[2] =='N' && BUFF[3] == 'D')

                continue;

            else

            {

                int k, j = 0,count=0;

                    for(k = blockCount ; k < (blockCount+10); k++)

                    {

                        cout << " EM[ " << k << " ]" << " " ;

                        for(int i = 0 ; i < 4 ; i++)

                        {

                                 EM[k][i] = BUFF[j];

                                 j++;

                                 cout << EM[k][i] ;

                        }

                        cout << endl ;

                    }

                    blockCount = blockCount + 10;

                }

            //Clearing buffer for reading next line

            CLEARBUFF();

        }

        inFile.close();

    }

    else {

        cout<<"\n\t\t Error opening file..\n";

    }

}

void STARTEXECUTION() {

    IC = 0;

    EXEUTEUSERPROGRAM();

}

void EXEUTEUSERPROGRAM() {

    for(i = 0 ; i < 4 ; i++) //Fetching Instrution

    {

        IR[i] = EM[IC][i];

    }

    IC++;

    location = ((int)IR[2] - 48) \* 10 + ((int)IR[3] - 48); // Finding the loation specified in the instruction

    if(IR[0] == 'G' && IR[1] =='D')

    {

        SI = 1;

        MOS();

    }

    else if(IR[0] == 'P' && IR[1] =='D')

    {

        SI = 2;

        MOS();

    }

    else if(IR[0] == 'H')

    {

        SI = 3;

        MOS();

    }

    else if(IR[0]=='L' && IR[1]=='R')

    {

        for(i=0;i<4;i++)

        {

            R[i] = EM[location][i];

            //cout << R[i] ;

        }

        EXEUTEUSERPROGRAM();

    }

    else if(IR[0]=='S' && IR[1]=='R')

    {

        for(i=0;i<4;i++)

        {

            EM[location][i] = R[i];

            //cout << EM[location][i] ;

        }

        EXEUTEUSERPROGRAM();

    }

    else if(IR[0]=='C' && IR[1]=='R')

    {

        for(i = 0 ; i < 4 ; i++)

        {

            if(R[i] == EM[location][i])

            {

               C = 1;

               //cout << "\nStrings Match..\n" ;

            }

            else

            {

                C = 0;

               // cout << "\nStrings Don't Match..\n" ;

                break;

            }

        }

        EXEUTEUSERPROGRAM();

    }

    else if(IR[0]=='B' && IR[1]=='T')

    {

        if(C == 1)

        {

            IC = location;

            //cout << "\nValue of IC : " << IC ; //prints pointer location between 0-99

        }

        EXEUTEUSERPROGRAM();

    }

else if(IR[0]=='A' && IR[1] =='D')

{

int res, temp;

string em\_tmp="";

string r\_tmp="";

for(i=0; i<4; i++)

{

em\_tmp += EM[location][i];

r\_tmp += R[i];

}

res = stoi(r\_tmp);

temp = stoi(em\_tmp);

sprintf(R , "%d", res+temp);

EXEUTEUSERPROGRAM();

}

}

void MOS() {

    switch(SI)

    {

        case 1:

            READ(location);

            break;

        case 2:

            WRITE();

            break;

        case 3:

            TERMINATE();

            break;

        default:

            SI = 0;

    }

}

void READ(int loc) {

     inFile.getline(BUFF,40);

     j=0;

    for(k = loc ; k < (loc+10); k++)     //current string exists for 10 positions even if not occupied.

    {

        cout << " EM[ " << k << " ]" << " " ;

        for(int i = 0 ; i < 4 && BUFF[j] != '\0'  ; i++)

        {

             EM[k][i] = BUFF[j];

             j++;

             cout << EM[k][i] ;   //printing the characters in input file

        }

        cout << endl ;

    }

    EXEUTEUSERPROGRAM();

}

void WRITE() {

    k = 0;

    CLEARBUFF();

    for(i = location ; i < location+10 ; i++)

    {

        for(j = 0; j < 4 && EM[i][j] != '#'; j++)

        {

            if(EM[i][j]=='\n')

                BUFF[k]=' ';

            else

             BUFF[k] = EM[i][j];

            // cout << EM[i][j] ;

            k++;

        }

    }

    BUFF[k] = '\n';

    outFile.open("output.txt", ios::out|ios::app);

    outFile << BUFF;

    outFile.close();

    EXEUTEUSERPROGRAM();

}

void CLEARBUFF() {

    for(int j = 0 ; j <= MAX ; j++) {

        BUFF[j] = '\0';

        //cout << BUFF[j] ;

    }

}

void TERMINATE() {

    CLEARBUFF();

    outFile.open("output.txt", ios::out|ios::app);

    outFile << "\n\n" ;

    outFile.close();

}

**Input.txt file**

$AMJ000100030001

GD10PD10H

$DTA

Hello

$END0001

$AMJ000200120003

GD20GD30GD40GD50PD20PD30LR20CR30BT10PD40

PD50H

$DTA

VIT

VIIT

Do Not

Match

$END0002

$AMJ000300080001

GD20GD30LR30SR23LR31SR24PD20H

$DTA

Operating-

System

$END0003

$AMJ000400090003

GD20GD30LR20AD30SR40PD20PD30PD40H

$DTA

210

10

$END0004

**Output.txt –**

Hello

VIT

VIIT

Do Not

Match

Operating-System

210

10

220

VIT

VIT

Match

**Console Output –**

