**OS COURSE PROEJCT PHASE 2**

**SY Div:** C **Batch:** 2 **Group No.:** 10

**Members:**

Vedant Mude Rutuj Nagrale Aarya Nirgude Tejashri Nirmal

**Code:**

#include <bits/stdc++.h>

using namespace std;

struct PCB

{

int job\_id;

int TTL;

int TLL;

int TTC;

int TLC;

};

struct PCB P;

int ptr;

int visited[30];

char M[300][4];

char IR[4];

char R[4];

int IC;

bool C;

int SI;

int VA;

int RA;

int PI;

int TI;

int EM;

fstream inFile;

fstream outFile;

unordered\_map<int, string> errors = {{0, "No Error"},

{1, "Out of Data"}, {2, "Line Limit Exceeded"},

{3, "Time Limit Exceeded"},

{4, "Operation Code Error"},

{5, "Operand Error"}, {6, "Invalid Page Fault"}};

void init();

void LOAD();

void EXECUTE();

void MOS();

int ALLOCATE();

int ADDRESSMAP(int);

int TERMINATE(int);

void init()

{

for (int i = 0; i < 300; i++)

{

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

{

M[i][j] = ' ';

}

}

for (int i = 0; i < 30; i++)

{

visited[i] = 0;

}

IR[4] = {'-'};

R[4] = {'-'};

IC = 0;

C = false;

ptr = 0;

VA = 0;

PI = 0;

TI = 0;

EM = 0;

}

int ALLOCATE()

{

return (rand() % 30);

}

int ADDRESSMAP(int va)

{

int pte = ptr \* 10 + va / 10;

string temp = "";

if (M[pte][0] == '\*')

{

cout << "Page Fault" << endl;

return -1;

}

else

{

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

{

if (M[pte][i] != ' ')

temp += M[pte][i];

}

return ((stoi(temp) \* 10) + (va % 10));

}

}

int TERMINATE(int Code)

{

cout << "\n"

<< errors[Code] << endl;

outFile << "\nProgram Terminated abnormally"

<< "\t\t";

outFile << errors[Code] << endl;

}

void MOS()

{

if (SI == 1)

{

string line;

getline(inFile, line);

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

{

EM = 1;

TERMINATE(1);

return;

}

int frame = ALLOCATE();

while (visited[frame] != 0)

{

frame = ALLOCATE();

}

visited[frame] = 1;

int i = ptr;

i = i \* 10;

cout << "\n\nPTR = " << ptr << endl;

while (M[i][0] != '\*')

{

i++;

}

int temp = frame / 10;

M[i][0] = ' ';

M[i][1] = ' ';

M[i][2] = temp + 48;

M[i][3] = frame % 10 + 48;

int l = 0;

frame = frame \* 10;

for (int i = 0; i < line.length() && line.length() < 40; i++)

{

M[frame][l++] = line[i];

if (l == 4)

{

l = 0;

frame += 1;

}

}

}

else if (SI == 2)

{

P.TLC += 1;

if (P.TLC > P.TLL)

{

EM = 2;

TERMINATE(2);

return;

}

int add = IR[2] - 48;

add = (add \* 10);

int ra = ADDRESSMAP(add);

if (ra != -1)

{

string out;

for (int i = 0; i < 10; i++)

{

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

{

out += M[ra][j];

}

ra += 1;

}

outFile << out << "\n";

}

else

{

EM = 6;

TERMINATE(6);

PI = 3;

}

}

else if (SI == 3)

{

outFile << "\nProgram Terminated Successfully"

<< "\n";

outFile << "IC = " << IC << "\tToggle : " << C << "\tTLC : " << P.TLC << "\tTTC : " << P.TTC << "\tTTL : " << P.TTL << "\tTLL : " << P.TLL;

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

{

outFile << "\t" << IR[i];

}

outFile<<"\n\n";

}

}

void EXECUTE()

{

while (true)

{

if (PI != 0 || TI != 0 || EM != 0)

{

outFile << "IC = " << IC << "\tToggle: " << C << "\tTLC: " << P.TLC << "\tTTC: " << P.TTC << "\tTTL" << P.TTL << "\tTLL" << P.TLL;

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

{

outFile << "\t" << IR[i];

}

outFile<<"\n\n";

break;

}

RA = ADDRESSMAP(IC);

if (M[RA][0] != 'H' && (!isdigit(M[RA][2]) || !isdigit(M[RA][3])))

{

EM = 5;

TERMINATE(EM);

outFile << "IC = " << IC << "\tToggle: " << C << "\tTLC: " << P.TLC << "\tTTC: " << P.TTC << "\tTTL: " << P.TTL << "\tTLL: " << P.TLL;

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

{

outFile << "\t" << IR[i];

}

outFile<<"\n\n";

}

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

{

IR[i] = M[RA][i];

}

IC++;

int add = IR[2] - 48;

add = (add \* 10) + (IR[3] - 48);

if ((IR[0] == 'G' && IR[1] == 'D') || (IR[0] == 'S' && IR[1] == 'R'))

P.TTC += 2;

else

P.TTC += 1;

if (P.TTC > P.TTL)

{

EM = 3;

TI = 2;

TERMINATE(EM);

outFile << "IC = " << IC << "\tToggle: " << C << "\tTLC: " << P.TLC << "\tTTC: " << P.TTC << "\tTTL: " << P.TTL << "\tTLL: " << P.TLL;

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

{

outFile << "\t" << IR[i];

}

outFile<<"\n\n";

break;

}

if (IR[0] == 'L' && IR[1] == 'R')

{

int ra = ADDRESSMAP(add);

if (ra == -1)

{

EM = 6;

TERMINATE(6);

}

else

{

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

R[i] = M[ra][i];

}

}

else if (IR[0] == 'S' && IR[1] == 'R')

{

int ra = ADDRESSMAP(add);

if (ra != -1)

{

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

M[ra][i] = R[i];

}

else

{

int frame = ALLOCATE();

while (visited[frame] != 0)

{

frame = ALLOCATE();

}

visited[frame] = 1;

int i = ptr;

i = i \* 10;

while (M[i][0] != '\*')

{

i++;

}

int temp = frame / 10;

M[i][0] = ' ';

M[i][1] = ' ';

M[i][2] = temp + 48;

M[i][3] = frame % 10 + 48;

frame = frame \* 10;

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

M[frame][i] = R[i];

}

}

else if (IR[0] == 'C' && IR[1] == 'R')

{

int flag = 0;

int ra = ADDRESSMAP(add);

if (ra == -1)

{

EM = 6;

TERMINATE(6);

}

else

{

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

{

if (R[i] != M[ra][i])

flag = 1;

}

if (flag == 1)

C = false;

else

C = true;

}

}

else if (IR[0] == 'B' && IR[1] == 'T')

{

if (C == true)

IC = add;

}

else 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();

break;

}

else

{

EM = 4;

TERMINATE(EM);

outFile << "IC = " << IC << "\tToggle: " << C << "\tTLC: " << P.TLC << "\tTTC: " << P.TTC << "\tTTL: " << P.TTL << "\tTLL: " << P.TLL;

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

{

outFile << "\t" << IR[i];

}

outFile<<"\n\n";

break;

}

}

}

void LOAD()

{

cout << "\nReading Data..." << endl;

int m = 0;

string line;

while (getline(inFile, line))

{

string str = "";

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

{

str += line[i];

}

if (str == "$AMJ")

{

init();

ptr = ALLOCATE();

//cout<<ptr<<endl;

for (int i = ptr \* 10; i < ptr \* 10 + 10; i++)

{

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

{

M[i][j] = '\*';

}

}

visited[ptr] = 1;

// Initialize PCB

string jobid\_str = "";

string TTL\_str = "";

string TLL\_str = "";

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

{

jobid\_str += line[i + 4];

TTL\_str += line[i + 8];

TLL\_str += line[i + 12];

}

P.job\_id = stoi(jobid\_str);

P.TTL = stoi(TTL\_str);

P.TLL = stoi(TLL\_str);

P.TLC = 0;

P.TTC = 0;

}

else if (str == "$DTA")

{

EXECUTE();

}

else if (str == "$END")

{

for (int i = 0; i < 300; i++)

{

cout << "M[" << i << "] - ";

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

{

cout << M[i][j]<<" ";

}

// cout << "\t\tM[" << i + 150 << "] - ";

// for (int j = 0; j < 4; j++)

// {

// cout << M[i + 150][j]<<" ";

// }

cout << endl;

}

cout << "\n\*\*\*\*\*\*\*\*\*\*\*\*\*\* END/HALT \*\*\*\*\*\*\*\*\*\*\*\*\*\*\n\n"

<< endl;

}

else

{

int frameNo = ALLOCATE();

while (visited[frameNo] != 0)

{

frameNo = ALLOCATE(); // 28

}

visited[frameNo] = 1;

int i = ptr;

while (M[i][0] != '\*')

{

i++;

}

int temp = frameNo / 10; // 28/10 = 2

M[i][0] = ' ';

M[i][1] = ' ';

M[i][2] = temp + 48;

M[i][3] = frameNo % 10 + 48;

int len = 0;

for (int i = frameNo \* 10; i < frameNo \* 10 + 10 && len < line.length(); i++)

{

for (int j = 0; j < 4 && len < line.length(); j++)

{

if (line[len] == 'H')

{

M[i][j] = line[len++];

break;

}

else

{

M[i][j] = line[len++];

}

}

}

line.clear();

}

}

}

int main()

{

inFile.open("input.txt", ios::in);

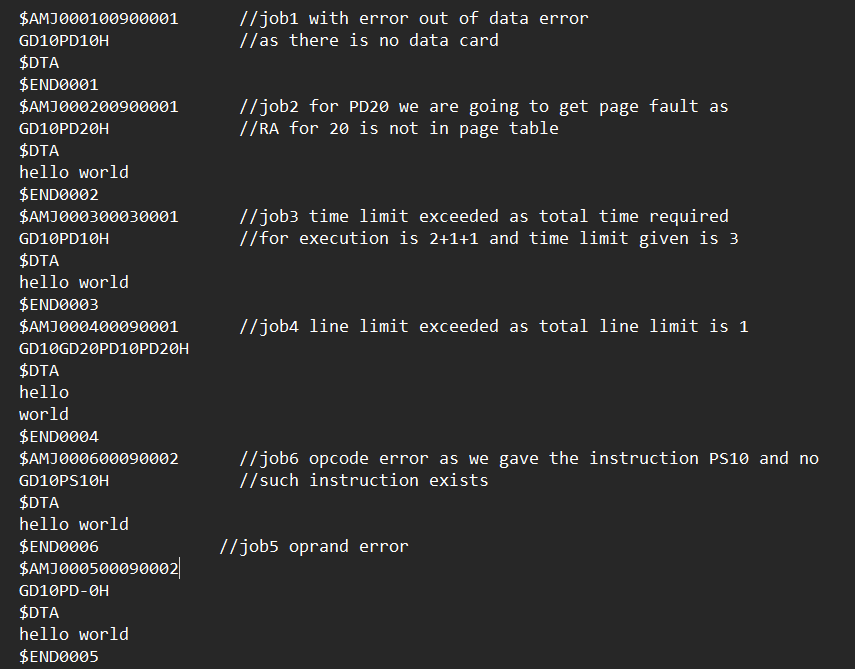
outFile.open("output.txt", ios::out);

LOAD();

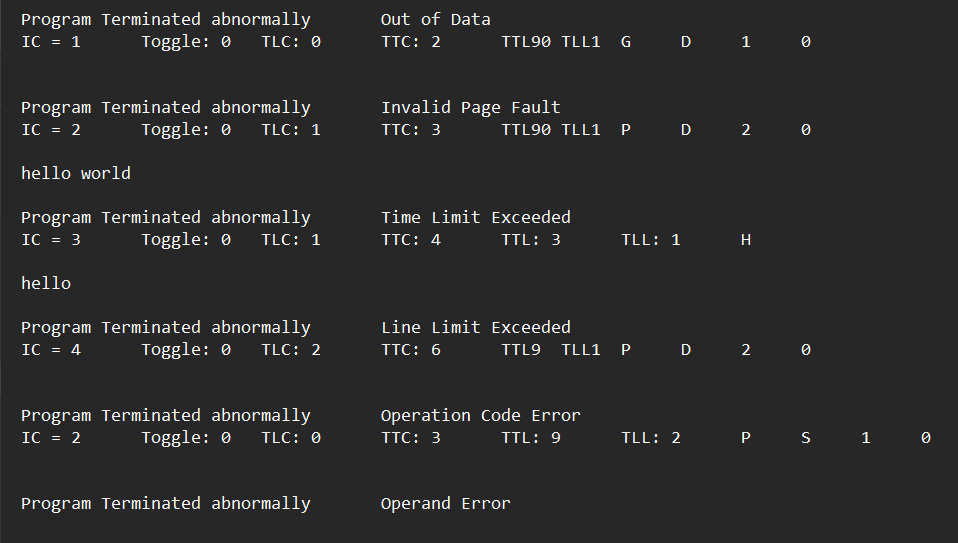
return 0;

}

**Input File:**



**Output File:**



**Output Screen:** (Memory) 