OS CP

Name: Shivam Ganesh Gavandi
DIV - TY-A-80
PRN - 12011128

Code: -

```
#include <iostream>
#include <fstream>
#include <string.h>
#include <cstdlib>
#include <time.h>
#include <vector>
using namespace std;

ifstream fin("input.txt");
ofstream fout("output.txt");

char M[300][4], buffer[40], IR[4], R[4];
int IC, C, SI, PI, TI, PTR, TTC, LLC;
bool flagBreak;   // To indicate current job is terminated
vector<int> allo; // To check repeat allocation in PTR

struct PCB
{
   int job_id;
   int TTL;
   int TLL;

   void setPCB(int id, int ttl, int tll)
   {
       job_id = id;
       TTL = ttl;
       TLL = tll;
   }
};

PCB pcb;

string error[9] = {"No Error", "Out of Data", "Line Limit Exceeded",
"Time Limit Exceeded",
```

```
                  "Operation Code Error", "Operand Error", "Invalid
Page Fault", "Time Limit Exceeded with opcode error", "Time Limit
Exceeded with operand error"};

void INIT();
void READ(int RA);
void WRITE(int RA);
int ADDRESSMAP(int VA);
void EXECUTE_USER_PROGRAM();
void STARTEXECUTION();
int ALLOCATE();
void LOAD();

void INIT()
{
   memset(M, '\0', 1200);
   memset(IR, '\0', 4);
   memset(R, '\0', 4);
   C = 0;
   SI = 0;
   PI = 0;
   TI = 0;
   flagBreak = false;
}

void TERMINATE(int EM)
{
   fout << endl;
   fout << "Job ID : " << pcb.job_id << endl
        << error[EM] << endl;
   fout << "IC : " << IC << endl
        << "IR : ";
   for (int i = 0; i < 4; i++)
       fout << IR[i];

   fout << endl
        << "TTC : " << TTC << endl
        << "LLC : " << LLC;
   fout << endl
        << endl;
}

void READ(int RA)
```

```
{
    fin.getline(buffer, 41);

    char temp[5];
    memset(temp, '\0', 5);
    memcpy(temp, buffer, 4);

    if (!strcmp(temp, "$END"))
    {
        TERMINATE(1);
        flagBreak = true;
    }
    else
    {
        strcpy(M[RA], buffer);
    }
}

void WRITE(int RA)
{
    if (LLC + 1 > pcb.TLL)
    {
        TERMINATE(2);
        flagBreak = true;
    }
    else
    {
        string str; //! Changed to string
        int k = 0;
        for (int i = RA; i < (RA + 10); i++)
        {
            for (int j = 0; j < 4; j++)
                str += M[i][j];
        }
        fout << str << endl;
        LLC++;
    }
}

int MOS(int RA = 0)
{
    if (TI == 0)
    {
```

```
        if (SI != 0)
        {
            switch (SI)
            {
            case 1:
                READ(RA);
                break;
            case 2:
                WRITE(RA);
                break;
            case 3:
                TERMINATE(0);
                flagBreak = true;
                break;
            default:
                cout << "Error with SI." << endl;
            }
            SI = 0;
        }
        // Page Fault checking
        else if (PI != 0)
        {
            switch (PI)
            {
            case 1:
                TERMINATE(4);
                flagBreak = true;
                break;
            case 2:
                TERMINATE(5);
                flagBreak = true;
                break;
            case 3:
                PI = 0;
                char temp[3];
                memset(temp, '\0', 3);
                memcpy(temp, IR, 2);
                // valid page fault
                if (!strcmp(temp, "GD") || !strcmp(temp, "SR"))
                {
                    int m;
                    do
                    {
```

```
                    m = ALLOCATE();
                } while (M[m * 10][0] != '\0');
                int currPTR = PTR;
                while (M[currPTR][0] != '0')
                    currPTR++;

                char temp1[2];
                sprintf(temp1, "%d", m);
                M[currPTR][0] = '1';

                if (m < 10)
                {
                    M[currPTR][2] = '0';
                    M[currPTR][3] = temp1[0];
                }
                else
                {
                    M[currPTR][2] = temp1[0];
                    M[currPTR][3] = temp1[1];
                }

                if (TTC + 1 > pcb.TTL)
                {
                    TI = 2;
                    PI = 3;
                    MOS();
                    break;
                }
                return 1;
            }
            else if (!strcmp(temp, "PD") || !strcmp(temp, "LR") ||
!strcmp(temp, "H") || !strcmp(temp, "CR") || !strcmp(temp, "BT"))
            {
                TERMINATE(6);
                flagBreak = true;

                if (TTC + 1 > pcb.TTL)
                {
                    TI = 2;
                    PI = 3;
                    MOS();
                    break;
                }
```

```
            }
            else
            {
                PI = 1;
                MOS();
            }
            return 0;
        default:
            cout << "Error with PI." << endl;
        }
        PI = 0;
    }
}
else
{
    if (SI != 0)
    {
        switch (SI)
        {
        case 1:
            TERMINATE(3);
            flagBreak = true;
            break;
        case 2:
            WRITE(RA);
            if (!flagBreak)
                TERMINATE(3);
            flagBreak = true; //! check
            break;
        case 3:
            TERMINATE(0);
            flagBreak = true;
            break;
        default:
            cout << "Error with SI." << endl;
        }
        SI = 0;
    }
    else if (PI != 0)
    {
        switch (PI)
        {
        case 1:
```

```
                TERMINATE(7);
                flagBreak = true;
                break;
            case 2:
                TERMINATE(8);
                flagBreak = true;
                break;
            case 3:
                TERMINATE(3);
                flagBreak = true;
                break;
            default:
                cout << "Error with PI." << endl;
            }
            PI = 0;
        }
    }

    return 0;
}

void increment()
{
    TTC++;
    if (TTC + 1 > pcb.TTL)
    {
        TI = 2;
    }
}

int ADDRESSMAP(int VA)
{
    if (0 <= VA && VA < 100)
    {
        int pte = PTR + VA / 10; // 112

        if (M[pte][0] == '0')
        {
            PI = 3;
            return 0;
        }

        char temp[2];
```

```
        temp[0] = M[pte][2];
        temp[1] = M[pte][3];
        int RA = atoi(temp) * 10 + VA % 10;
        return RA;
    }
    PI = 2;
    return 0;
}

void EXECUTE_USER_PROGRAM()
{
    char opcode[3], operand[2];
    int locIR, RA;

    while (true)
    {
        if (flagBreak)
            break;

        RA = ADDRESSMAP(IC);
        if (PI != 0)
        {
            if (MOS())
            {
                continue;
            }
            break;
        }
        memcpy(IR, M[RA], 4);
        IC += 1;

        memset(opcode, '\0', 3);
        memcpy(opcode, IR, 2);
        for (int i = 0; i < 2; i++)
        {
            if (!((47 < IR[i + 2] && IR[i + 2] < 58) || IR[i + 2] == 0))
            {
                PI = 2;
                break;
            }
            operand[i] = IR[i + 2];
        }
```

```
        if (PI != 0)
        {
            MOS();
            break;
        }

        locIR = atoi(operand);

        RA = ADDRESSMAP(locIR);
        if (PI != 0)
        {
            if (MOS())
            {
                IC--;
                continue;
            }
            break;
        }

        if (!strcmp(opcode, "LR"))
        {

            cout << endl;
            for (int i = 0; i < 4; i++)
            {
                R[i] = M[RA][i];
            }
            increment();
        }
        else if (!strcmp(opcode, "SR"))
        {

            for (int i = 0; i < 4; i++)
            {
                M[RA][i] = R[i];
            }
            TTC = TTC + 2;
            if (TTC + 2 > pcb.TTL)
            {
                TI = 2;
            }
        }
        else if (!strcmp(opcode, "CR"))
```

```
        {

            if (!strcmp(R, M[RA]))
                C = 1;
            else
                C = 0;
            increment();
        }
        else if (!strcmp(opcode, "BT"))
        {

            if (C == 1)
                IC = RA;
            increment();
        }
        else if (!strcmp(opcode, "GD"))
        {
            SI = 1;
            TTC = TTC + 2;
            if (TTC + 2 > pcb.TTL)
            {
                TI = 2;
            }
            MOS(RA);
        }
        else if (!strcmp(opcode, "PD"))
        {
            SI = 2;
            increment();
            MOS(RA);
        }
        else if (!strcmp(opcode, "H"))
        {
            SI = 3;
            increment();
            MOS();

            break;
        }
        else
        {
            PI = 1;
            MOS();
```

```
            break;
        }
        memset(IR, '\0', 4);
    }
    // for(int i=0; i<300; i++){
    //                 cout<<i<<" ";
    //             for(int j=0 ; j<4; j++){
    //                 cout<<M[i][j]<<" ";
    //             }
    //             cout<<endl;
    //         }
}

void STARTEXECUTION()
{
    IC = 0;
    EXECUTE_USER_PROGRAM();
}

int ALLOCATE()
{
    int random = rand() % 30;
    if (allo.size() == 0)
    {
        allo.push_back(random);
        return allo[0];
    }
    for (int i = 0; i < allo.size(); i++)
    {
        if (random == allo[i])
        {
            return ALLOCATE();
        }
    }
    allo.push_back(random);

    return allo[allo.size() - 1];
}

void LOAD()
{
    int m;        // Variable to hold memory loction
```

```
   int currPTR;  // Points to the last empty loction in Page Table
Register
   char temp[5]; // Temporary Variable to check for $AMJ, $DTA, $END
   memset(buffer, '\0', 40);

   while (!fin.eof())
   {
       fin.getline(buffer, 41);
       memset(temp, '\0', 5);
       memcpy(temp, buffer, 4);

       if (!strcmp(temp, "$AMJ"))
       { // if 0 then false (strcmp=0 if same)
           INIT();
           srand(time(0));
           int jobId, TTL, TLL;
           memcpy(temp, buffer + 4, 4);
           jobId = atoi(temp);
           memcpy(temp, buffer + 8, 4);
           TTL = atoi(temp);
           memcpy(temp, buffer + 12, 4);
           TLL = atoi(temp);
           pcb.setPCB(jobId, TTL, TLL);
           TTC = 0;
           LLC = 0;
           PTR = ALLOCATE() * 10;

           memset(M[PTR], '*', 40);
           for (int i = 0; i < 10; i++)
           {
               M[PTR + i][0] = '0';
           }
           currPTR = PTR;
       }
       else if (!strcmp(temp, "$DTA"))
       {
           STARTEXECUTION();
       }
       else if (!strcmp(temp, "$END"))
       {
           continue;
       }
       else
```

```
        {
            if (flagBreak)
                continue;

            do
            {
                m = ALLOCATE();
            } while (M[m * 10][0] != '\0');
            char temp[2];
            sprintf(temp, "%d", m);
            M[currPTR][0] = '1'; // flag set

            if (m < 10)
            {
                M[currPTR][2] = '0';
                M[currPTR][3] = temp[0];
            }
            else
            {
                M[currPTR][2] = temp[0];
                M[currPTR][3] = temp[1];
            }

            currPTR++;

            strcpy(M[m * 10], buffer);

            cout << "PTR = " << PTR << endl;

            cout << endl;
        }
    }
}

int main()
{
    LOAD();
    cout << "Execution Completed!";
    fin.close();
    fout.close();
    return 0;
}
```

Input File: -

```
$AMJ000100030001
GD10PD10H
$DTA
Hello World
$END0001
```

Output: -

```
M[291] :
IC = 0, RA = 220
In addressMap(), VA = 10, pte = 191, M[pte] = 25
IC = 1, RA = 250, IR = GD10
In addressMap(), VA = 1, pte = 190, M[pte] = 22
IC = 1, RA = 221
In addressMap(), VA = 10, pte = 191, M[pte] = 25
IC = 2, RA = 250, IR = PD10
In addressMap(), VA = 2, pte = 190, M[pte] = 22
IC = 2, RA = 222
In addressMap(), VA = 0, pte = 190, M[pte] = 22
IC = 3, RA = 220, IR = H
```