Phase 2

Code:

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

/*File Handlers*/
ifstream fin("input.txt");
ofstream fout("output.txt");

/*Memory*/
char M[300][4];
char buffer[40];
char IR[4];
char R[5];
int IC;
int C;
int SI;
int PI;
int TI;
int PTR;
bool breakFlag;

/*Process Control Block*/
struct PCB{
    int job_id;
    int TTL;
    int TLL;
    int TTC;
    int LLC;

    void setPCB(int id, int ttl, int tll){
        job_id = id;
        TTL = ttl;
        TLL = tll;
        TTC = 0;
        LLC = 0;
    }
};

PCB pcb;

/*Error Messages*/
```

```
string error[7] = {"No Error", "Out of Data", "Line Limit Exceeded",
"Time Limit Exceeded",
    "Operation Code Error", "Operand Error", "Invalid Page Fault"};

/*Functions*/
void init();
void read(int RA);
void write(int RA);
int addressMap(int VA);
void execute_user_program();
void start_execution();
int allocate();
void load();

void init(){
    memset(M, '\0', 1200);
    memset(IR, '\0', 4);
    memset(R, '\0', 5);
    C = 0;
    SI = 0;
    PI = 0;
    TI = 0;
    breakFlag = false;
}

void Terminate(int EM, int EM2=-1){
    fout << endl << endl;
    if(EM == 0){
        fout << " terminated normally. " << error[EM];
    }
    else{
        fout <<EM<<" - "<<error[EM] << (EM2 != -1 ? (". " + error[EM2])
: "") << endl;
        fout << "IC="<<IC<<", IR="<<IR<<", C="<<C<<", R="<<R<<",
TTL="<<pcb.TTL<<", TTC="<<pcb.TTC<<", TLL="<<pcb.TLL<<",
LLC="<<pcb.LLC;
    }
}
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);
        breakFlag = true;
```

```
    }
    else{
        strcpy(M[RA],buffer);
        // cout<<"this is RA"<<RA;
    }
}

void write(int RA){
    if(pcb.LLC+1 > pcb.TLL){
        Terminate(2);
        breakFlag = true;
    }
    else{
        char str[40];
        int k = 0;
        for(int i=RA; i<(RA+10); i++){
            for(int j=0; j<4; j++)
                str[k++] = M[i][j];
        }
        fout << str << endl;
        pcb.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);
                    breakFlag = true;
                    break;
                default:
                    cout<<"Error with SI."<<endl;
            }
            SI = 0;
        }
        else if(PI != 0){
            switch(PI){
                case 1:
                    Terminate(4);
                    breakFlag = true;
```

```
                    break;
                case 2:
                    Terminate(5);
                    breakFlag = true;
                    break;
                case 3:
                    PI = 0;
                    //Page Fault checking
                    char temp[3];
                    memset(temp,'\0',3);
                    memcpy(temp, IR, 2);

                    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]!='*')
                            currPTR++;

                        itoa(m, M[currPTR], 10);

                        cout << "Valid Page Fault, page frame = " << m
<< endl;
                        cout << "PTR = " << PTR << endl;
                        for(int i=0; i<300; i++){
                            cout<<"M["<<i<<"] :";
                            for(int j=0 ; j<4; j++){
                                cout<<M[i][j];
                            }
                            cout<<endl;
                        }
                        cout<<endl;

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

```
                    breakFlag = true;

                    if(pcb.TTC+1 > pcb.TTL){
                        TI = 2;
                        PI = 3;
                        mos();
                        break;
                    }
                    //pcb.TTC++;
                }
                else{
                    PI = 1;
                    mos();
                }
                return 0;
            default:
                cout<<"Error with PI."<<endl;
        }
        PI = 0;
    }
}
else{
    if(SI != 0){
        switch(SI){
            case 1:
                Terminate(3);
                breakFlag = true;
                break;
            case 2:
                write(RA);
                if(!breakFlag) Terminate(3);
                break;
            case 3:
                Terminate(0);
                breakFlag = true;
                break;
            default:
                cout<<"Error with SI."<<endl;
        }
        SI = 0;
    }
    else if(PI != 0){
        switch(PI){
            case 1:
                Terminate(3,4);
                breakFlag = true;
                break;
            case 2:
```

```
                    Terminate(3,5);
                    breakFlag = true;
                    break;
                case 3:
                    Terminate(3);
                    breakFlag = true;
                    break;
                default:
                    cout<<"Error with PI."<<endl;
            }
            PI = 0;
        }
    }

    return 0;
}

int addressMap(int VA){
    if(0 <= VA && VA < 100){
        int pte = PTR + VA/10;
        if(M[pte][0] == '*'){
            PI = 3;
            return 0;
        }
        cout << "In addressMap(), VA = " << VA << ", pte = " << pte <<
", M[pte] = " << M[pte] << endl;
        return atoi(M[pte])*10 + VA%10;
    }
    PI = 2;
    return 0;
}

void execute_user_program(){
    char temp[3], loca[2];
    int locIR, RA;

    while(true){
        if(breakFlag) break;

        RA = addressMap(IC);
        if(PI != 0){
            if(mos()){
                continue;
            }
            break;
        }
        cout << "IC = " << IC << ", RA = " << RA << endl;
```

```
        memcpy(IR, M[RA], 4);           //Memory to IR, instruction
fetched
        IC += 1;

        memset(temp,'\0',3);
        memcpy(temp,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;
            }
            loca[i] = IR[i+2];
        }

        if(PI != 0){
            mos();
            break;
        }

        //loca[0] = IR[2];
        //loca[1] = IR[3];
        locIR = atoi(loca);

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

        cout << "IC = " << IC << ", RA = " << RA << ", IR = " << IR <<
endl;
        if(pcb.TTC+1 > pcb.TTL){
            TI = 2;
            PI = 3;
            mos();
            break;
        }

        if(!strcmp(temp,"LR")){
            memcpy(R,M[RA],4);
            pcb.TTC++;
        }
        else if(!strcmp(temp,"SR")){
            memcpy(M[RA],R,4);
            pcb.TTC++;
```

```
        }
        else if(!strcmp(temp,"CR")){
            if(!strcmp(R,M[RA]))
                C = 1;
            else
                C = 0;
            pcb.TTC++;
        }
        else if(!strcmp(temp,"BT")){
            if(C == 1)
                IC = RA;
            pcb.TTC++;
        }
        else if(!strcmp(temp,"GD")){
            SI = 1;
            mos(RA);
            pcb.TTC++;
        }
        else if(!strcmp(temp,"PD")){
            SI = 2;
            mos(RA);
            pcb.TTC++;
        }
        else if(!strcmp(temp,"H")){
            SI = 3;
            mos();
            pcb.TTC++;
            break;
        }
        else{
            PI = 1;
            mos();
            break;
        }
        memset(IR, '\0', 4);
    }
}

void start_execution(){
    IC = 0;
    execute_user_program();
}

int allocate(){
    srand(time(0));
    return (rand() % 30);
}
```

```
void load(){
    int m;                                  //Variable to hold memory
location
    int currPTR;                            //Points to the last empty
location 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")){
            init();

            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);

            PTR = allocate()*10;
            memset(M[PTR], '*', 40);
            currPTR = PTR;
        }
        else if(!strcmp(temp,"$DTA")){
            start_execution();
        }
        else if(!strcmp(temp,"$END")){
            continue;
        }
        else{
            if(breakFlag) continue;

            do{
                m = allocate();
            }while(M[m*10][0]!='\0');

            itoa(m, M[currPTR], 10);
            currPTR++;

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

```
            cout << "PTR = " << PTR << endl;
            for(int i=0; i<300; i++){
                cout<<"M["<<i<<"] :";
                for(int j=0 ; j<4; j++){
                    cout<<M[i][j];
                }
                cout<<endl;
            }
            cout<<endl;
        }
    }
}

int main(){
    load();
    fin.close();
    fout.close();
    return 0;
}
```

Input:

```
$AMJ000100040002
GD10PD10H
$DTA
HELLO WORLD!
$END0001
$AMJ000200040001
GD10PD10H
$DTA
$END0002
$AMJ000300040000
GD10PD10H
$DTA
HELLO WORLD!
$END0003
$AMJ000400030001
GD10PD10H
$DTA
HELLO WORLD!
$END0004
$AMJ000500040001
GP10PD10H
$DTA
HELLO WORLD!
$END0005
$AMJ000600040001
GD111PD10H
$DTA
HELLO WORLD!
$END0006
$AMJ000700040001
PD10H
$DTA
HELLO WORLD!
$END0007
```

Output in the Text File:

```
HELLO WORLD!


 terminated normally. No Error

1 - Out of Data
IC=1, IR=GD10, C=0, R=, TTL=4, TTC=1, TLL=1, LLC=0

2 - Line Limit Exceeded
IC=2, IR=PD10, C=0, R=, TTL=4, TTC=2, TLL=0, LLC=0HELLO WORLD!


3 - Time Limit Exceeded
IC=3, IR=H, C=0, R=, TTL=3, TTC=3, TLL=1, LLC=1

4 - Operation Code Error
IC=1, IR=GP10, C=0, R=, TTL=4, TTC=0, TLL=1, LLC=0

5 - Operand Error
IC=2, IR=1PD1, C=0, R=, TTL=4, TTC=2, TLL=1, LLC=0

6 - Invalid Page Fault
IC=1, IR=PD10, C=0, R=, TTL=4, TTC=0, TLL=1, LLC=0
```

Output on the console:

```
In addressMap(), VA = 0, pte = 270, M[pte] = 0
IC = 0, RA = 0
PTR = 0
M[0] :3**
M[1] :****
M[2] :****
M[3] :****
M[4] :****
M[5] :****
M[6] :****
M[7] :****
M[8] :****
M[9] :****
M[10] :

 M[30] :GD11
 M[31] :1PD1
 M[32] :0H
 M[33] :
 M[34] :
 M[35] :
 M[36] :
 M[37] :
 M[38] :
 M[39] :
 M[40] :


In addressMap(), VA = 0, pte = 0, M[pte] = 3
IC = 0, RA = 30
Valid Page Fault, page frame = 7
PTR = 0
M[0] :3**
M[1] :7**
M[2] :****
M[3] :****
M[4] :****
M[5] :****
M[6] :****
M[7] :****
M[8] :****
M[9] :****
M[10] :
M[11] :
M[12] :
M[13] :
```

```
In addressMap(), VA = 0, pte = 0, M[pte] = 3
IC = 0, RA = 30
In addressMap(), VA = 11, pte = 1, M[pte] = 7
IC = 1, RA = 71, IR = GD11
In addressMap(), VA = 1, pte = 0, M[pte] = 3
IC = 1, RA = 31
PTR = 70
M[0] :
M[1] :
M[2] :
M[3] :
M[4] :
M[5] :
M[6] :
M[7] :
M[8] :
M[9] :
M[10] :
M[11] :


M[290] :
M[291] :
M[292] :
M[293] :
M[294] :
M[295] :
M[296] :
M[297] :
M[298] :
M[299] :

In addressMap(), VA = 0, pte = 70, M[pte] = 10
```