OS Phase 2

Group: TY 3

Group Members:

Roll no NAME GR. NO.

3 Aarya Tiwari 12010588

5 Harshal Abak 12011089

9 Abu Ansari 12010599

26 Kaivalya Aole 12011351

Code was written in java

# Code:-

import java.io.\*;

import java.nio.file.Files;

import java.nio.file.Paths;

import java.nio.file.StandardOpenOption;

import java.util.Random;

public class Phase\_2 {

    public static void main(String[] args) {

        OS2 obj = null;

        try {

            obj = new OS2();

        } catch (FileNotFoundException e) {

            e.printStackTrace();

        }

        obj.load();

    }

}

class OS2{

    char [] buffer;

    char [][] memory;

    int mem\_ptr;

    //CPU

    char [] IR;

    char [] R;

    int PTR;

    int IC;

    boolean toggle;

    //counters

    int TTC;

    int LLC;

    //interrupts

    int SI; //system Interrupt

    int TI; //Time interrupt

    int PI; //page interrupt

    int EM; //errror message;

    //miscellaneous

    int [] used\_frames; //keeps track of the frames used

    int temp1;   //temporary usage variable

    Random random;

    PCB pcb; // pcb data structure;

    class PCB{

        private int TTL;

        private int TLL;

        private int jobId;

        public void setJobId(int jobId) {

            this.jobId = jobId;

        }

        public void setTLL(int TLL) {

            this.TLL = TLL;

        }

        public void setTTL(int TTL) {

            this.TTL = TTL;

        }

        public int getJobId() {

            return jobId;

        }

        public int getTLL() {

            return TLL;

        }

        public int getTTL() {

            return TTL;

        }

    }

    File input;

    FileReader fr;

    public OS2() throws FileNotFoundException {

        //size allocation

        buffer = new char[40];

        memory = new char[300][4];

        IR = new char[4];

        R = new char [4];

        random = new Random();

        value\_allocator();

    }

    private void value\_allocator() throws FileNotFoundException {

        //value allocation

        for(int i = 0;i<4;i++){

            IR[i] = '@';

            R[i] = '@';

        }

        buffer\_reset();

        for(int i = 0;i < 300; i++){

            for(int j = 0; j< 4;j++){

                memory[i][j] = '@';

            }

        }

        IC = 0;

        toggle = false;

        mem\_ptr = 0;

        SI = 0;

        TTC = 0;

        LLC = 0;

        EM = 0;

        TI = 0;

        PI = 0;

        used\_frames = new int[30];

    }

    public void MOS(){

        if(TI == 2)

            terminate(3);

        if(PI != 0){

            switch (PI) {

                case 1:

                    if (TI == 0) {

                        terminate(4);

                    } else {

                        terminate(3);

                        terminate(4);

                    }

                    break;

                case 2:

                    if (TI == 0) {

                        terminate(5);

                    } else {

                        terminate(3);

                        terminate(5);

                    }

                    break;

                case 3:

                    if (TI == 0) {

                        System.out.println("page fault");

                        if (SI == 1 || (IR[0] == 'S' && IR[1] == 'R')) {

                            System.out.println("Resolving..");

                            Allocate();

                            TTC++;

                        } else {

                            terminate(6);

                        }

                    }

                    if (TI == 2) {

                        terminate(3);

                    }

            }

            PI = 0;

        }

        else {

            switch (SI) {

                case 1:

                    if (TI == 0) {

                        try {

                            read();

                        } catch (IOException e) {

                            e.printStackTrace();

                        }

                    } else {

                        terminate(3);

                    }

                    break;

                case 2:

                    if (TI == 0) {

                        write();

                    } else {

                        write();

                        terminate(3);

                    }

                    break;

                case 3:

                    terminate(0);

                    break;

            }

            SI = 0;

        }

    }

    public void read() throws IOException {

        System.out.println("Read executed");

        IR[3] = '0';

        int buffer\_ptr;

        char [] temp;

        if(reader.ready()){

            temp = reader.readLine().toCharArray();

            for (buffer\_ptr = 0; buffer\_ptr < temp.length && buffer\_ptr<40; buffer\_ptr++) {

                buffer[buffer\_ptr] = temp[buffer\_ptr];

            }

            System.out.println(buffer);

        }

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

            System.out.println("Program Over");

            terminate(1);

            return;

        }

        int ra = realAddress((IR[2]-'0') \*10);

        if(PI == 2){

            MOS();

            return;

        }

        if(PI == 3){

            MOS();

        }

        ra = realAddress((IR[2]-'0') \*10);

        load\_memory\_data(ra);

        buffer\_reset();

    }

    public void write(){

        System.out.println("Write executed");

        IR[3] = '0';

        int buffer\_ptr;

        char [] temp;

        LLC++;

        if(LLC>pcb.getTLL())

            terminate(2);

        else{

            try{

                int memory\_ptr = realAddress((IR[2]-'0')\*10);

                if(PI == 2 || PI == 3){

                    MOS();

                }

                else{

                    int buffer\_ptr4 = 0;

                    int limit = memory\_ptr+10;

                    for(memory\_ptr = realAddress((IR[2]-'0')\*10);memory\_ptr<limit;memory\_ptr++){

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

                            if(memory[memory\_ptr][i] == '@')

                                continue;

                            buffer[buffer\_ptr4] = memory[memory\_ptr][i];

                            buffer\_ptr4++;

                        }

                    }

                    StringBuffer sb = new StringBuffer();

                    buffer\_ptr4 = 0;

//                System.out.println(buffer);

                    while(buffer\_ptr4 < 40 && buffer[buffer\_ptr4]!='@'){

                        sb.append(buffer[buffer\_ptr4]);

                        buffer\_ptr4++;

                    }

                    sb.append('\n');

                    Files.write(Paths.get("out.txt"), String.valueOf(sb).getBytes(), StandardOpenOption.APPEND);

                    buffer\_reset();

                }

            }

            catch(IOException e){

                System.out.println("Exception occured at write()");

                e.printStackTrace();

            }

        }

    }

    public void terminate(int EM){

        String jobDetails = "\n\nJobID: " + pcb.getJobId() + "\tTTL: " + pcb.getTTL() + "\tTLL: " + pcb.getTLL() + "\nTTC: " + TTC + "\tLLC: "+ LLC+"\n";

        String interrupts = "\nSI: "+SI+"\tPI: "+PI+"\tTI: "+TI+"\n";

        String cpuDetails = "\nIC: "+IC+"\nIR: "+IR[0]+IR[1]+IR[2]+IR[3] + "\nR: "+R[0]+R[1]+R[2]+R[3] + "\nToggle: "+toggle+"\n";

        try{

            switch (EM){

                case 0:

                    Files.write(Paths.get("out.txt"), (jobDetails+interrupts+cpuDetails+"\nTerminated Sucessfully\n\n").getBytes(), StandardOpenOption.APPEND);

                    break;

                case 1:

                    Files.write(Paths.get("out.txt"), (jobDetails+interrupts+cpuDetails+"\nOut of Data\n\n").getBytes(), StandardOpenOption.APPEND);

                    break;

                case 2:

                    Files.write(Paths.get("out.txt"), (jobDetails+interrupts+cpuDetails+"\nLine limit exceeded\n\n").getBytes(), StandardOpenOption.APPEND);

                    break;

                case 3:

                    Files.write(Paths.get("out.txt"), (jobDetails+interrupts+cpuDetails+"\nTime Limit Exceeded\n\n").getBytes(), StandardOpenOption.APPEND);

                    break;

                case 4:

                    Files.write(Paths.get("out.txt"), (jobDetails+interrupts+cpuDetails+ "\nOP code error\n\n").getBytes(), StandardOpenOption.APPEND);

                    break;

                case 5:

                    Files.write(Paths.get("out.txt"), (jobDetails+interrupts+cpuDetails+ "\nOperand error\n\n").getBytes(), StandardOpenOption.APPEND);

                    break;

                case 6:

                    Files.write(Paths.get("out.txt"), (jobDetails+interrupts+cpuDetails+"\nInvalid page fault\n\n").getBytes(), StandardOpenOption.APPEND);

                    break;

            }

            IC = 100;

            return;

        }

        catch(IOException e){

            System.out.println("Exception occured at halt()");

            e.printStackTrace();

        }

    }

    public void MOSstartExec(){

        System.out.println("Exec started");

        IC = 0;

        executeUserProgram();   //slave mode

    }

    public void executeUserProgram(){

        //loading IR

        int ra = realAddress(IC);

        while (IC<99 && memory[ra][0] != '@') {

            System.out.println("IC is " + IC);

            int ra2;

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

                IR[i] = memory[ra][i];

            }

            IC++;

            switch (IR[0]) {

                case 'L':

                    if(IR[1] == 'R'){

                        ra2 = realAddress((IR[2] - '0')\*10 + (IR[3] - '0'));

                        if(PI == 3 || PI ==2)

                            MOS();

                        else{

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

                                R[i] = memory[ra2][i];

                            }

                        }

                        TTC++;

                    }

                    else{

                        PI = 1;

                        MOS();

                    }

                    break;

                case 'S':

                    if(IR[1] == 'R'){

                        ra2 = realAddress((IR[2] - '0')\*10 + (IR[3] - '0'));

                        if(PI == 3 || PI ==2)

                            MOS();

                        ra2 = realAddress((IR[2] - '0')\*10 + (IR[3] - '0'));

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

                            memory[ra2][i] = R[i];

                        }

                        TTC++;

                    }

                    else{

                        PI = 1;

                        MOS();

                    }

                    break;

                case 'C':

                    if (IR[1] == 'R') {

                        ra2 = realAddress((IR[2] - '0')\*10 + (IR[3] - '0'));

                        if(PI == 3 || PI ==2)

                            MOS();

                        else{

                            comparing(ra2);

                        }

                        TTC++;

                    }

                    else{

                        PI = 1;

                        MOS();

                    }

                    break;

                case 'B':

                    if(IR[1] == 'T'){

//                        ra2 = (IR[2] - '0')\*10 + (IR[3] - '0');

                        if(PI == 3 || PI ==2)

                            MOS();

                        else{

                            if(toggle){

                                IC = (IR[2] - '0') \*10 + (IR[3] - '0');

                            }

                        }

                        TTC++;

                    }

                    else{

                        PI = 1;

                        MOS();

                    }

                    break;

                case 'G':

                    if (IR[1] == 'D') {

                        SI = 1;

                        MOS();

                        TTC++;

                    }

                    else{

                        PI = 1;

                        MOS();

                    }

                    break;

                case 'P':

                    if (IR[1] == 'D') {

                        SI = 2;

                        MOS();

                        TTC++;

                    }

                    else{

                        PI = 1;

                        MOS();

                    }

                    break;

                case 'H':

                    SI = 3;

                    MOS();

                    TTC++;

                    break;

                default:

                    PI = 1;

                    MOS();

                    return;

            }

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

                IR[i] = '@';

            }

            ra = realAddress(IC);

            if(TTC>pcb.getTTL()){

                TI = 2;

                MOS();

            }

        }

    }

    public void comparing(int c) {

        for (int i = 0; i < 4; i++) {

            if (R[i] == memory[c][i]) {

                continue;

            } else {

                toggle = false;

                return;

            }

        }

        toggle = true;

        return ;

    }

    private static java.io.File file;

    private static BufferedReader reader;

    static {

        try {

            file = new java.io.File("input.txt");

            reader = new BufferedReader(new FileReader(String.valueOf(file)));

        } catch (FileNotFoundException e) {

            e.printStackTrace();

        }

    }

    public void load(){

        //loading from card to buffer

        try {

            do {

                int buffer\_ptr = 0;

                char [] temp;

                temp = reader.readLine().toCharArray();

                for (buffer\_ptr = 0; buffer\_ptr < temp.length; buffer\_ptr++) {

                    buffer[buffer\_ptr] = temp[buffer\_ptr];

                }

                System.out.println(buffer);

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

                    pcb = new PCB();

                    pcb.setJobId(((buffer[4] - '0')\*1000) + ((buffer[5] - '0')\*100) + ((buffer[6] - '0')\*10) + ((buffer[7] - '0')));

                    pcb.setTTL(((buffer[8] - '0')\*1000) + ((buffer[9] - '0')\*100) + ((buffer[10] - '0')\*10) + ((buffer[11] - '0')));

                    pcb.setTLL(((buffer[12] - '0')\*1000) + ((buffer[13] - '0')\*100) + ((buffer[14] - '0')\*10) + ((buffer[15] - '0')));

                    //page allocation

                    System.out.println("Job id: " + pcb.getJobId());

                    System.out.println("TTL: " + pcb.getTTL());

                    System.out.println("TLL: " + pcb.getTLL());

                    PTR = random.nextInt(30);

                    used\_frames[PTR] = 1;

                    buffer\_reset();

                    continue;

                }

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

                    //do something

//                    return;

                    buffer\_reset();

//                    mem\_ptr = (mem\_ptr % 10 == 0) ? mem\_ptr : ((mem\_ptr / 10 + 1) \* 10);

                    MOSstartExec();

                }

                else if (buffer[0] == '$' && buffer[1] == 'E' && buffer[2] == 'N' && buffer[3] == 'D') {  //Add D of $END later

                    System.out.println("Program Over");

                    printer();

                    value\_allocator();

                }

                else{

                    //instruction frame allocation and loading

                    temp1=Allocate() ;

                    load\_memory\_instructions(temp1);

                    buffer\_reset();

                }

                if (buffer\_ptr > 40) {

                    System.out.println("Buffer Overload, quitting...");

                    return;

                }

            }while(reader.ready());

        } catch (IOException e) {

            e.printStackTrace();

        }

    }

    public void load\_memory\_instructions(int address){

        int buffer\_ptr2 = 0;

        int limit = address+10;

        while(buffer\_ptr2<40 && buffer[buffer\_ptr2]!='@' && address<limit){

            for(int i = 0;i<4;i++){

                if(buffer[buffer\_ptr2] == '@')

                    break;

                memory[address][i] = buffer[buffer\_ptr2];

                if(buffer[buffer\_ptr2]=='H'){

                    buffer\_ptr2++;

                    break;

                }

                buffer\_ptr2++;

            }

            address++;

        }

//        printer();

    }

    public void load\_memory\_data(int location){

        mem\_ptr = location;

        if(mem\_ptr>=300){

            System.out.println("Memory Overload, quitting...");

            return;

        }

        int buffer\_ptr1 = 0;

        while(buffer\_ptr1<40 && buffer[buffer\_ptr1]!='@'){

            for(int i = 0;i<4;i++){

                if(buffer[buffer\_ptr1] == '@')

                    break;

                memory[mem\_ptr][i] = buffer[buffer\_ptr1];

                buffer\_ptr1++;

            }

            mem\_ptr++;

        }

    }

    public void buffer\_reset(){

        for(int i = 0;i< 40;i++){

            buffer[i] = '@';

        }

    }

    private int Allocate(){

        int temp2;

        while(used\_frames[temp2 = random.nextInt(30)] != 0){}

        used\_frames[temp2] = 1;

        temp2 = temp2 \*10;

        int ret = temp2;

        int i = PTR;

        if(IR[2] != '@')

            i = getPTE((IR[2]-'0') \*10);

        else{

            while(memory[i][0]!='@'){i++;}

        }

        System.out.println("PTE is " + i);

        memory[i][0] = '0';

        memory[i][3] = (char) (temp2 % 10 + '0');

        temp2 /= 10;

        memory[i][2] = (char) (temp2 % 10 + '0');

        temp2 /= 10;

        memory[i][1] = (char) (temp2 % 10 + '0');

        return ret;

    }

    private int realAddress(int VA){

        if(IR[2]!='@'){

            if((IR[2]-'0'>9) || (IR[2]-'0'<0) || (IR[3]-'0'>9) || (IR[3]-'0'<0)){

                PI = 2;

                return -1;

            }

        }

        if(VA>99 || VA<0){

            PI = 2;

            return -1;

        }

        int pte = getPTE(VA);

        if(memory[pte][0] == '@') {

            PI = 3;

            System.out.println("PI = 3");

            return -1;

        }

        int ra = (memory[pte][1]-'0') \* 100 + (memory[pte][2] - '0')\*10 + (memory[pte][3] - '0') + VA%10; //real address

        //operand errors

        return ra;

    }

    private int getPTE(int VA){

        return PTR + VA/10;

    }

    public void printer(){

        System.out.println("Buffer is: ");

        System.out.println(buffer);

        System.out.println("IC is:"+IC);

        System.out.println("PTR: "+PTR );

        System.out.println("IR:");

        System.out.println(IR);

        System.out.println("R:");

        System.out.println(R);

        System.out.println("Toggle:");

        System.out.println(toggle);

        System.out.println("TTC: "+TTC);

        System.out.println("LLC: "+LLC);

        System.out.println("TTL: "+pcb.getTTL());

        System.out.println("TLL: "+pcb.getTLL());

        System.out.println("Used frames are: ");

        for(int i = 0;i<30;i++){

            System.out.print(used\_frames[i] );

        }

        System.out.println("\nMain memory");

        for(int i = 0; i< 300;i++){

            System.out.print(i+" : ");

            for(int j = 0;j<4;j++){

                System.out.print(memory[i][j]);

            }

            System.out.println();

        }

    }

}

# Input File:

$AMJ000700090003

GD10GD20GD30LR10SR17PD10PD20PD30H

$DTA

Hello world

Hehe boi

Lunaboi

$END0007

$AMJ000800140002

GD10GD20GD30LR10SR17PD10PD20PD30H

$DTA

Hello world

Hehe boi

Lunaboi

$END0008

$AMJ000900140003

GD10GD20GD30LR10SR17PD40PD20PD30H

$DTA

Hello world

Hehe boi

Lunaboi

$END0009

$AMJ001000140003

GD10GD20GD30LR1OSR17PD10PD20PD30H

$DTA

Hello world

Hehe boi

Lunaboi

$END0010

GD10GD20GD30LD10SR17PD40PD20PD30H

$DTA

Hello world

Hehe boi

Lunaboi

$END0011

$AMJ001200140003

GD10GD20GD30LR10SR17PD40PD20PD30H

$DTA

Hello world

Hehe boi

$END0012

# Output File:

VIIT010002100020

VIT

NOT SAME

JobID: 2 TTL: 16 TLL: 3

TTC: 14 LLC: 3

SI: 3 PI: 0 TI: 0

IC: 11

IR: H@@@

R: VIIT

Toggle: false

Terminated Sucessfully

NOT VOWEL

JobID: 4 TTL: 24 TLL: 1

TTC: 20 LLC: 1

SI: 3 PI: 0 TI: 0

IC: 17

IR: H@@@

R: b@@@

Toggle: false

Terminated Sucessfully

\*

\* \*

\* \* \*

\* \* \* \*

\* \* \*

\* \*

\*

JobID: 6 TTL: 31 TLL: 7

TTC: 24 LLC: 7

SI: 3 PI: 0 TI: 0

IC: 20

IR: H@@@

R: \* @@

Toggle: false

Terminated Sucessfully

Hello worldHell

Hehe boi

JobID: 7 TTL: 9 TLL: 3

TTC: 10 LLC: 2

SI: 0 PI: 0 TI: 2

IC: 7

IR: @@@@

R: Hell

Toggle: false

Time Limit Exceeded

Hello worldHell

Hehe boi

JobID: 8 TTL: 14 TLL: 2

TTC: 10 LLC: 3

SI: 2 PI: 0 TI: 0

IC: 8

IR: PD30

R: Hell

Toggle: false

Line limit exceeded

JobID: 9 TTL: 14 TLL: 3

TTC: 8 LLC: 1

SI: 2 PI: 3 TI: 0

IC: 6

IR: PD40

R: Hell

Toggle: false

Invalid page fault

JobID: 10 TTL: 14 TLL: 3

TTC: 6 LLC: 0

SI: 0 PI: 2 TI: 0

IC: 4

IR: LR1O

R: @@@@

Toggle: false

Operand error

JobID: 10 TTL: 14 TLL: 3

TTC: 6 LLC: 0

SI: 0 PI: 1 TI: 0

IC: 4

IR: LD10

R: @@@@

Toggle: false

OP code error

JobID: 12 TTL: 14 TLL: 3

TTC: 4 LLC: 0

SI: 1 PI: 0 TI: 0

IC: 3

IR: GD30

R: @@@@

Toggle: false

Out of Data