Міністерство освіти і науки, молоді та спорту України

Національний технічний університет України

«Київський політехнічний інститут»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

**Лабораторна робота №3**

З дисципліни «Комп’ютерне моделювання»

Виконав: Перевірив:

Студент групи ІО-21 доц. Марковський О.П.

Коноз А.О.

Дата здачі\_\_\_\_\_\_\_\_\_\_\_\_\_

Захищено з балом\_\_\_\_\_

Київ 2014

**Лістинг коду**

**import** java.text.DecimalFormat;

**import** java.text.NumberFormat;

**import** java.util.ArrayList;

**import** java.util.Random;

**public** **class** Node {

Random r = **new** Random();

**private** **double** solveTime = 0.0;

**double** delay = 0.0;

ArrayList<Task> stack = **new** ArrayList<Task>();

**private** **int** resource = 1;

**private** Task solving;

ArrayList<Node> connections = **new** ArrayList<Node>();

ArrayList<Double> possibilities = **new** ArrayList<Double>();

//--------------For statistic--------------------------------------

**int** commandCount = 0;

**double** loading = 0.0;

//-----------------------------------------------------------------

**public** Node(**double** delay){

**this**.delay = delay;

}

**public** **void** add(Task task){

**if**(resource == 1){

resource--;

solving = task;

solveTime = SystemClock.*time* + getDelay();

// loading += getDelay();

}**else**{

stack.add(task);

}

}

**public** **double** getSolvingTime(){

**return** solveTime;

}

**public** **void** action(){

**if**(solving != **null**){

SystemClock.*time* = solveTime;

commandCount++;

loading += getDelay();

sendTask(solving);

resource++;

**if**(stack.size() != 0){

resource--;

solving = stack.remove(0);

solveTime = SystemClock.*time* + getDelay();

}**else**{

solving = **null**;

solveTime = Double.***MAX\_VALUE***;

}

}

}

**public** **void** addConnection(Node node, **double** posibility){

connections.add(node);

possibilities.add(**new** Double(posibility));

}

**public** **void** sendTask(Task task){

**double** pos = r.nextDouble();

**double** sum = 0.0;

**for** (**int** i = 0; i < possibilities.size(); i++) {

sum += possibilities.get(i);

**if**(sum > pos){

connections.get(i).add(task);

**break**;

}

}

}

**public** **double** getDelay(){

**return** delay;

}

**public** **double** getLoading(){

**return** loading;

}

**public** **static** **void** main(String[] args) {

Node array [] = **new** Node[7];

array[0] = **new** CPU(0.00025); //------CPU

array[1] = **new** Node(0.005); //------NB

array[2] = **new** Node(1.0); //------SB

array[3] = **new** Node(0.00066);//------RAM

array[4] = **new** Node(0.001); //------GP

array[5] = **new** Node(0.03); //------ISA

array[6] = **new** Node(22.0); //------LPT

//----------------CPU connections-----------------------

array[0].addConnection(array[0], 0.85);

array[0].addConnection(array[1], 0.15);

//----------------North bridge connections--------------

array[1].addConnection(array[0], 0.1);

array[1].addConnection(array[2], 0.15);

array[1].addConnection(array[3], 0.5);

array[1].addConnection(array[4], 0.25);

//----------------South bridge connections--------------

array[2].addConnection(array[5], 1.0);

//----------------RAM connections-----------------------

array[3].addConnection(array[1], 1.0);

//----------------Graphic processor connections---------

array[4].addConnection(array[1], 1.0);

//----------------ISA connections-----------------------

array[5].addConnection(array[6], 1.0);

//----------------LPT connections-----------------------

array[6].addConnection(array[0], 1.0);

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

array[0].add(**new** Task());

}

**double** time = array[0].getSolvingTime();

**int** index = 0;

**for** (**int** i = 0; i < 1000000; i++) {

time = array[0].getSolvingTime();

**for** (**int** j = 0; j < array.length; j++) {

**if**((array[j].getSolvingTime() != 0.0)&&(array[j].getSolvingTime() <= time)){

time = array[j].getSolvingTime();

index = j;

}

}

SystemClock.*currentActionTime* = time;

array[index].action();

}

System.***out***.println(SystemClock.*time*);

NumberFormat f = **new** DecimalFormat("###0.00000");

System.***out***.println("CPU: " + array[0].commandCount + " || loading of 1 processor: " + f.format((((CPU)array[0]).loading1/SystemClock.*time*))+ " loading of 2 processors: " + f.format((((CPU)array[0]).loading2/SystemClock.*time*)));

System.***out***.println("North bridge: " + array[1].commandCount + " || loading: " + f.format((array[1].getLoading()/SystemClock.*time*)));

System.***out***.println("South bridge: " + array[2].commandCount + " || loading: " + f.format((array[2].getLoading()/SystemClock.*time*)));

System.***out***.println("RAM: " + array[3].commandCount + " || loading: " + f.format((array[3].getLoading()/SystemClock.*time*)));

System.***out***.println("Graphic processor: " + array[4].commandCount + " || loading: " + f.format((array[4].getLoading()/SystemClock.*time*)));

System.***out***.println("ISA: " + array[5].commandCount + " || loading: " + f.format((array[5].getLoading()/SystemClock.*time*)));

System.***out***.println("LPT: " + array[6].commandCount + " || loading: " + f.format((array[6].getLoading()/SystemClock.*time*)));

}

}

**import** java.util.ArrayList;

**import** java.util.Random;

**public** **class** CPU **extends** Node{

**private** **double** solveTime1 = 0.0;

**private** **double** solveTime2 = 0.0;

**private** **int** resource = 3;

**private** Task solving1;

**private** Task solving2;

//--------------For statistic--------------------------------------

// int commandCount = 0;

**double** loading1 = 0.0;

**double** loading2 = 0.0;

**double** startcalc1 = 0.0;

**double** startcalc2 = 0.0;

//-----------------------------------------------------------------

**public** CPU(**double** delay){

**super**(delay);

**this**.delay = delay;

}

**public** **void** add(Task task){

**switch** (resource) {

**case** 0:

stack.add(task);

**break**;

**case** 1:

solving1 = task;

solveTime1 = SystemClock.*time* + getDelay();

resource -= 1;

**break**;

**case** 2:

solving2 = task;

solveTime2 = SystemClock.*time* + getDelay();

resource -= 2;

**break**;

**case** 3:

solving1 = task;

solveTime1 = SystemClock.*time* + getDelay();

resource -= 1;

**break**;

**default**:

**break**;

}

}

**public** **double** getSolvingTime(){

**double** result = 0.0;

**if**(solving2 == **null**){

result = solveTime1;

}**else**{

**if**(solving1 == **null**){

result = solveTime2;

}**else**{

**if**((solving1 == **null**)&&(solving2 == **null**)){

result = Double.***MAX\_VALUE***;

}**else**{

**if**(solveTime1 < solveTime2){

result = solveTime1;

}**else**{

result = solveTime2;

}

}

}

}

**return** result;

}

**public** **void** action(){

**if**(solveTime1 == SystemClock.*currentActionTime*){

SystemClock.*time* = solveTime1;

commandCount++;

**if**(solving2 == **null**){

loading1 += getDelay();

}**else**{

loading2 += Math.*abs*(solveTime1 - solveTime2) - startcalc1;

}

sendTask(solving1);

resource += 1;

**if**(stack.size() != 0){

resource -= 1;

solving1 = stack.remove(0);

solveTime1 = SystemClock.*time* + getDelay();

}**else**{

solving1 = **null**;

solveTime1 = Double.***MAX\_VALUE***;

}

}**else**{

**if**(solveTime2 == SystemClock.*currentActionTime*){

SystemClock.*time* = solveTime2;

commandCount++;

**if**(solving1 == **null**){

loading1 += getDelay();

}**else**{

loading2 += Math.*abs*(solveTime1 - solveTime2) - startcalc2;

}

sendTask(solving2);

resource += 2;

**if**(stack.size() != 0){

resource -= 2;

solving2 = stack.remove(0);

solveTime2 = SystemClock.*time* + getDelay();

}**else**{

solving2 = **null**;

solveTime2 = Double.***MAX\_VALUE***;

}

}

}

}

**public** **double** getLoading(){

**return** 0.0;

}

}

**public** **class** SystemClock {

**public** **static** **double** *time* = 0.0;

**public** **static** **double** *currentActionTime* = 0.0;

}