Национальный технический университет Украины

«Киевский политехнический институт»

Факультет информатики и вычислительной техники

Кафедра вычислительной техники

**Курсова робота**

**с дисциплины «Параллельные и распределённые исчисления»**

*Выполнил:*

студент группы ІО-12

№ зачетной книжки: 1206

Ванчугов Б. Ю.

*Проверил:*

Корочкин А. В.

Киев, 2014 р.

**Тема**: Программирование для ПКС с ОП с помощью мониторов.

**Техническое задание**

1. Структура ПКС с ОП:

ЗП

P

l, MK, MT, MM

P-1

2

1

MA,ME,MO

…

1. Задача: .
2. Мова програмування: Ада.
3. Засоби взаємодії процесів: монітор (захищений модуль).

**РОЗДІЛ 2. РОЗРОБКА ПРГ1**

1. Розробка паралельного математичного алгоритму.
2. Розробка алгоритмів процесів.

**Задача :**

1. Введення *ME, MO.*
2. Сигнал про закінчення вводу.
3. Чекати закінчення введення в .
4. Обчислення1
5. Обчислення2
6. Сигнал про закінчення

Обчислення2.

1. Чекати закінчення Обчислення2

в .

1. Копії
2. Обчислення3
3. Чекати закінчення обчислення

в .

1. Виведення *MA*.

**Задача :**

1. Введення *l, MK, MT, MM.*
2. Сигнал про закінчення вводу.
3. Чекати закінчення введення в .
4. Обчислення1
5. Обчислення2
6. Сигнал про закінчення

Обчислення2.

1. Чекати закінчення Обчислення2

в .

1. Копії
2. Обчислення
3. Сигнал про закінчення обчислення .

**Задача :**

1. Чекати закінчення введення в .
2. Обчислення1
3. Обчислення2
4. Сигнал про закінчення

Обчислення2.

1. Чекати закінчення Обчислення2

в .

1. Копії
2. Обчислення
3. Сигнал про закінчення обчислення .
4. Розробка схеми взаємодії процесів.
5. Розробка програми.

with Ada.Text\_IO, Ada.Integer\_Text\_IO, Ada.Calendar;

use Ada.Text\_IO, Ada.Integer\_Text\_IO, Ada.Calendar;

procedure Main is

N: Integer := 900;

P: Integer := 4;

H: Integer := N / P;

StartTime, FinishTime: Time;

DiffTime: Duration;

type Vector is array(1..N) of Integer;

type Matrix is array(1..N) of Vector;

--l: Integer;

MA, MO, ME, MK, MT, MM, Res: Matrix;

procedure MatrixInput(M: out Matrix) is

begin

for i in 1..N loop

for j in 1..N loop

M(i)(j) := 1;

end loop;

end loop;

end MatrixInput;

procedure MatrixOutput(M: in Matrix) is

begin

for i in 1..N loop

for j in 1..(N-1) loop

put(M(i)(j));

put(" ");

end loop;

put(M(i)(N));

Put\_Line("");

end loop;

end MatrixOutput;

protected ResourceMonitor is

procedure WriteME(M: in Matrix);

procedure WriteMT(M: in Matrix);

procedure WriteL(lInp: in Integer);

procedure CalculateE(ei: in Integer);

function CopyE return Integer;

function CopyL return Integer;

function CopyME return Matrix;

function CopyMT return Matrix;

private

l: Integer;

e: Integer := Integer'Last;

ME: Matrix;

MT: Matrix;

end ResourceMonitor;

protected SynchronizeMonitor is

procedure SignalInput;

procedure SignalCalculate2;

procedure SignalCalculate3;

entry WaitInput;

entry WaitCalculate2;

entry WaitCalculate3;

private

flagInput: Integer := 0;

flagCalculate2: Integer := 0;

flagCalculate3: Integer := 0;

end SynchronizeMonitor;

protected body ResourceMonitor is

procedure WriteME(M: in Matrix) is

begin

ME := M;

end WriteME;

procedure WriteMT(M: in Matrix) is

begin

MT := M;

end WriteMT;

procedure WriteL(lInp: in Integer) is

begin

l := lInp;

end WriteL;

procedure CalculateE(ei: in Integer) is

begin

if ( e > ei ) then

e := ei;

end if;

end CalculateE;

function CopyE return Integer is

begin

return e;

end CopyE;

function CopyL return Integer is

begin

return l;

end CopyL;

function CopyME return Matrix is

begin

return ME;

end CopyME;

function CopyMT return Matrix is

begin

return MT;

end CopyMT;

end ResourceMonitor;

protected body SynchronizeMonitor is

procedure SignalInput is

begin

flagInput := flagInput + 1;

end SignalInput;

procedure SignalCalculate2 is

begin

flagCalculate2 := flagCalculate2 + 1;

end SignalCalculate2;

procedure SignalCalculate3 is

begin

flagCalculate3 := flagCalculate3 + 1;

end SignalCalculate3;

entry WaitInput

when flagInput = 2 is

begin

null;

end WaitInput;

entry WaitCalculate2

when flagCalculate2 = P is

begin

null;

end WaitCalculate2;

entry WaitCalculate3

when flagCalculate3 = P is

begin

null;

end WaitCalculate3;

end SynchronizeMonitor;

task type CalculateTask(taskNumber: Integer);

task body CalculateTask is

first, last: Integer;

sum: Integer;

ei: Integer;

li: Integer;

lInput : Integer;

MEi, MTi: Matrix;

begin

Put\_Line("Task " & Integer'Image(taskNumber) & " started...");

if ( taskNumber = 1 ) then

-- ÐÐ²ÐµÐ´ÐµÐ½Ð½Ñ A, ME, MK

MatrixInput(MA);

MatrixInput(ME);

MatrixInput(MO);

ResourceMonitor.WriteME(ME);

-- Ð¡Ð¸Ð³Ð½Ð°Ð» T\_j (j = 2..P) Ð¿ÑÐ¾ Ð·Ð°ÐºÑÐ½ÑÐµÐ½Ð½Ñ Ð²Ð²Ð¾Ð´Ñ

SynchronizeMonitor.SignalInput;

end if;

if ( taskNumber = P ) then

-- ÐÐ²ÐµÐ´ÐµÐ½Ð½Ñ B, MB, MO

--l := 1;

MatrixInput(MK);

MatrixInput(MT);

MatrixInput(MM);

lInput := 1;

ResourceMonitor.WriteMT(MT);

ResourceMonitor.WriteL(lInput);

-- Ð¡Ð¸Ð³Ð½Ð°Ð» T\_j (j = 1..P-1) Ð¿ÑÐ¾ Ð·Ð°ÐºÑÐ½ÑÐµÐ½Ð½Ñ Ð²Ð²Ð¾Ð´Ñ

SynchronizeMonitor.SignalInput;

end if;

-- Ð§ÐµÐºÐ°ÑÐ¸ Ð·Ð°ÐºÑÐ½ÑÐµÐ½Ð½Ñ Ð²Ð²ÐµÐ´ÐµÐ½Ð½Ñ

SynchronizeMonitor.WaitInput;

first := (taskNumber - 1) \* H + 1;

last := taskNumber \* H;

-- ÐÐ±ÑÐ¸ÑÐ»ÐµÐ½Ð½Ñ a\_i

ei := Integer'Last;

for i in first..last loop

for j in 1..N loop

if ( ei > MO(i)(j) ) then

ei := ME(i)(j);

end if;

end loop;

end loop;

-- ÐÐ±ÑÐ¸ÑÐ»ÐµÐ½Ð½Ñ a

ResourceMonitor.CalculateE(ei);

SynchronizeMonitor.SignalCalculate2;

SynchronizeMonitor.WaitCalculate2;

li := ResourceMonitor.CopyL;

ei := ResourceMonitor.CopyE;

MEi := ResourceMonitor.CopyME;

MTi := ResourceMonitor.CopyMT;

for i in first..last loop

for j in 1..N loop

sum := 0;

for k in 1..N loop

sum := sum + MT(j)(k)\*MM(k)(i);

end loop;

Res(j)(i) := sum;

end loop;

end loop;

for i in first..last loop

for j in 1..N loop

sum := 0;

for k in 1..N loop

sum := sum + ME(j)(k)\*Res(k)(i);

end loop;

MA(j)(i) := ei\*MK(j)(i)+li\*sum;

end loop;

end loop;

if ( taskNumber = 1 ) then

-- Ð§ÐµÐºÐ°ÑÐ¸ Ð·Ð°ÐºÑÐ½ÑÐµÐ½Ð½Ñ Ð¾Ð±ÑÐ¸ÑÐ»ÐµÐ½Ð½Ñ MA\_H Ð² T\_j (j = 2..P)

SynchronizeMonitor.SignalCalculate3;

SynchronizeMonitor.WaitCalculate3;

-- ÐÐ¸Ð²ÐµÐ´ÐµÐ½Ð½Ñ MA

if ( N <= 8 ) then

MatrixOutput(MA);

end if;

FinishTime := Clock;

DiffTime := FinishTime - StartTime;

Put("Time = ");

Put(Integer(DiffTime), 1);

Put\_Line("");

else

-- Ð¡Ð¸Ð³Ð½Ð°Ð» T\_1 Ð¿ÑÐ¾ Ð·Ð°ÐºÑÐ½ÑÐµÐ½Ð½Ñ Ð¾Ð±ÑÐ¸ÑÐ»ÐµÐ½Ð½Ñ MA\_H

SynchronizeMonitor.SignalCalculate3;

end if;

Put\_Line("Task " & Integer'Image(taskNumber) & " finished");

end CalculateTask;

---------------------------------------------------------------

-- Ð¢Ð¸Ð¿ Ð²ÐºÐ°Ð·ÑÐ²Ð½Ð¸ÐºÐ° Ð½Ð° Ð·Ð°Ð´Ð°ÑÑ

type CalculateTaskPointer is access CalculateTask;

-- ÐÐ°ÑÑÐ¸Ð² Ð²ÐºÐ°Ð·ÑÐ²Ð½Ð¸ÐºÑÐ² Ð½Ð° Ð·Ð°Ð´Ð°ÑÑ

type TasksArray is array(1..P) of CalculateTaskPointer;

tArray: TasksArray;

ch : Character;

begin

Get(ch);

StartTime := Clock;

for i in 1..P loop

tArray(i) := new CalculateTask(i);

end loop;

end Main;