Національний технічний університет України

«Київський політехнічний інститут»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

Лабораторна робота №6

Паралельне програмування в мові Ada.

Захищений модуль.

Виконав:

студент групи ІП-22

Сочка О. О.

Перевірив:

Корочкін О. В.

Київ - 2015

**Технічне завдання**



**Етап 1. Побудова паралельного алгоритму**

**Спільні ресурси:** q, a, B, MO

**Етап 2. Розробка алгоритмів процесів (задач)**

**t – номер задачі**

|  |  |  |
| --- | --- | --- |
| **№** | **Т1** | **ТС/КУ** |
| 1 | Якщо t=1, ввід |  |
| 2 | Якщо t=1, сигнал іншим задачам про завершення вводу |  |
| 3 | Якщо t=4, ввід |  |
| 4 | Якщо t=4, сигнал іншим задачам про завершення вводу |  |
| 5 | Якщо t=6, ввід |  |
| 6 | Якщо t=6, сигнал іншим задачам про завершення вводу |  |
| 7 | Очікування вводу |  |
| 8 | Очікування вводу |  |
| 9 | Очікування вводу |  |
| 10 | Копія a, B, MO | КУ |
| 11 | Підрахунок |  |
| 12 | Підрахунок | КУ |
| 13 | Копія q | КУ |
| 14 | Сигнал іншим потокам про підрахунок q |  |
| 15 | Очікування підрахунку q в інших потоках |  |
| 16 | Підрахунок |  |
| 17 | Якщо , сигнал потоку T1 про завершення підрахунку |  |
| 18 | Якщо очікування завершення підрахунку в інших потоках |  |
| 19 | Якщо ивід |  |

**Етап 3. Розробка схеми взаємодії задач**



**Етап 4. Розробка програми**

**1 -----------------------------------------------------------**

**2 -- Lab6. Ada. Protected module**

**3 -- Name: Sochka Oleksandr Oleksandrovich**

**4 -- Group: IP-22**

**5 -- Date: 19.04.2015**

**6 -- MA = a \* B \*(MO\*MX) + (Z\*E) \* R**

**7 -----------------------------------------------------------**

**8 with Ada.Text\_IO; use Ada.Text\_IO;**

**9 with Ada.Integer\_text\_iO; use Ada.Integer\_Text\_IO;**

**10**

**11 procedure Lab6 is**

**12 N: constant Integer := 720;**

**13 P: constant Integer := 6;**

**14 H: constant Integer := N/P;**

**15**

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

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

**18**

**19**

**20 Z, E, R: Vector;**

**21 A: Vector := (others => 0);**

**22 MX: Matrix;**

**23**

**24 protected SynchroAndOP is**

**25 entry wait\_input;**

**26 entry wait\_calc1;**

**27 entry wait\_calc2;**

**28**

**29 function copy\_alpha return Integer;**

**30 function copy\_q return Integer;**

**31 function copy\_B return Vector;**

**32 function copy\_MO return Matrix;**

**33 procedure increase\_q(var: in Integer);**

**34 procedure write\_alpha(var: in Integer);**

**35 procedure write\_B(var: in Vector);**

**36 procedure write\_MO(var: in Matrix);**

**37 procedure signal\_input;**

**38 procedure signal\_calc1;**

**39 procedure signal\_calc2;**

**40 private**

**41 alpha: Integer;**

**42 q: Integer := 0;**

**43 B: Vector;**

**44 MO: Matrix;**

**45 cnt\_inputs: Integer := 0;**

**46 cnt\_calc1: Integer := 0;**

**47 cnt\_calc2: Integer := 0;**

**48 end SynchroAndOP;**

**49**

**50 protected body SynchroAndOP is**

**51 entry wait\_input when cnt\_inputs = 3 is**

**52 begin**

**53 null;**

**54 end wait\_input;**

**55**

**56 entry wait\_calc1 when cnt\_calc1 = P is**

**57 begin**

**58 null;**

**59 end wait\_calc1;**

**60**

**61 entry wait\_calc2 when cnt\_calc2 = P is**

**62 begin**

**63 null;**

**64 end wait\_calc2;**

**65**

**66 function copy\_alpha return Integer is**

**67 begin**

**68 return alpha;**

**69 end;**

**70 function copy\_q return Integer is**

**71 begin**

**72 return q;**

**73 end;**

**74 function copy\_B return Vector is**

**75 begin**

**76 return B;**

**77 end;**

**78 function copy\_MO return Matrix is**

**79 begin**

**80 return MO;**

**81 end;**

**82**

**83 procedure increase\_q(var: in Integer) is**

**84 begin**

**85 q := q + var;**

**86 end increase\_q;**

**87 procedure write\_alpha(var: in Integer) is**

**88 begin**

**89 alpha := var;**

**90 end write\_alpha;**

**91 procedure write\_B(var: in Vector) is**

**92 begin**

**93 B := var;**

**94 end write\_B;**

**95 procedure write\_MO(var: in Matrix) is**

**96 begin**

**97 MO := var;**

**98 end write\_MO;**

**99 procedure signal\_input is**

**100 begin**

**101 cnt\_inputs := cnt\_inputs + 1;**

**102 end;**

**103 procedure signal\_calc1 is**

**104 begin**

**105 cnt\_calc1 := cnt\_calc1 + 1;**

**106 end;**

**107 procedure signal\_calc2 is**

**108 begin**

**109 cnt\_calc2 := cnt\_calc2 + 1;**

**110 end;**

**111 end SynchroAndOP;**

**112**

**113 task type GenericTask is**

**114 entry start(tt: Integer);**

**115 end GenericTask;**

**116**

**117 task body GenericTask is**

**118 t, first, last, alpha, q: Integer;**

**119 tmp\_q: Integer := 0;**

**120 B: Vector;**

**121 MO: Matrix;**

**122 begin**

**123 accept start(tt: Integer) do**

**124 t := tt;**

**125 end start;**

**126**

**127 put\_line("Process GenericTask" & Integer'Image(t) & " started");**

**128**

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

**130 last := first + H - 1;**

**131**

**132 -- Input start;**

**133 case t is**

**134 when 1 =>**

**135 MX := (others => (others => 1));**

**136 declare**

**137 B\_input: Vector := (others => 1);**

**138 begin**

**139 SynchroAndOP.write\_B(B\_input);**

**140 end;**

**141 SynchroAndOP.signal\_input;**

**142 when 4 =>**

**143 Z := (others => 1);**

**144 R := (others => 1);**

**145 SynchroAndOP.write\_alpha(1);**

**146 SynchroAndOP.signal\_input;**

**147 when 6 =>**

**148 E := (others => 1);**

**149 declare**

**150 MO\_input: Matrix := (others => (others => 1));**

**151 begin**

**152 SynchroAndOP.write\_MO(MO\_input);**

**153 end;**

**154 SynchroAndOP.signal\_input;**

**155 when others =>**

**156 null;**

**157 end case;**

**158 -- Input end;**

**159**

**160 put\_line("Process GenericTask" & Integer'Image(t) & " waiting for input...");**

**161 SynchroAndOP.wait\_input;**

**162 put\_line("Process GenericTask" & Integer'Image(t) & " received input");**

**163 alpha := SynchroAndOP.copy\_alpha;**

**164 B := SynchroAndOP.copy\_B;**

**165 MO := SynchroAndOP.copy\_MO;**

**166**

**167 -- calc1 start;**

**168 for i in first..last loop**

**169 tmp\_q := tmp\_q + Z(i) \* E(i);**

**170 end loop;**

**171 SynchroAndOP.increase\_q(tmp\_q);**

**172 -- calc1 end;**

**173**

**174 SynchroAndOP.signal\_calc1;**

**175 SynchroAndOP.wait\_calc1;**

**176**

**177 q := SynchroAndOP.copy\_q;**

**178 -- calc2 start;**

**179 for j in first..last loop**

**180 for i in 1..N loop**

**181 for k in 1..N loop**

**182 A(j) := A(j) + alpha \* B(i) \* MO(i)(k) \* MX(k)(j);**

**183 end loop;**

**184 end loop;**

**185 A(j) := A(j) + q \* R(j);**

**186 end loop;**

**187 -- calc2 end;**

**188 SynchroAndOP.signal\_calc2;**

**189 -- output start;**

**190 if t = 1 then**

**191 SynchroAndOP.wait\_calc2;**

**192 if N <= 30 then**

**193 for i in A'Range loop**

**194 put(A(i), 7);**

**195 end loop;**

**196 put\_line("");**

**197 end if;**

**198 end if;**

**199 -- output end;**

**200 put\_line("Process GenericTask" & Integer'Image(t) & " finished");**

**201 end GenericTask;**

**202**

**203 Tasks: array (1..P) of GenericTask;**

**204 begin**

**205 put\_line("Main procedure started");**

**206 for i in Tasks'Range loop**

**207 Tasks(i).start(i);**

**208 end loop;**

**209 end Lab6;**

**210 pragma main(stack\_size=>200000000);**