**НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ УКРАИНЫ**

**“КИЕВСКИЙ ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ”**

**ФАКУЛЬТЕТ ИНФОРМАТИКИ И ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ**

**КАФЕДРА ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ**

**Параллельное программирование**

##### Лабораторная работа №7

Ada Rendezvous

Выполнил:

студент 3-го курса

группы ИВ-01

Раковций В.Ф.

Киев – 2013 г.

##### Техническое задание

Разработать программу для реализация в ПКС с ЛК (структура задана на рис. 1) математической задани:

A = max(B) \* C + alpha \* T \* (MO \* MK)

Язык программирования: Ада

Средства взаимодействия: Rendezvous

2

P-1

1

A, B, alpha, MO

P

C, T, MK

…

**Рис. 1 Структура ПКС**

**Разработка программы**

**Этап 1**. Разработка параллельного математического алгоритма

ai = max(BH); (i = )

a = max(ai, a); (i = )

AH = a \* CH + alpha \* T \* (MOH \* MK)

**Этап 2.** Разработка алгоритма процессоров.

|  |
| --- |
| Ti |
| 1. Если i = 0, то ввод B, MO, alpha. Если i = P – 1, то ввод MK, C, T |
| 2. Если i > 0, то ждать сообщения от задачи i-1 с B, MO, alpha. |
| 3. Если i < P-1, то отправить сообщение задаче i+1 с B, MO, alpha |
| 4. Если i > 0, то ждать сообщения от задачи i+1 с MK, C, T. |
| 5. Если i < P-1, то отправить сообщение задаче i-1 с MK, C, T |
| 6. Счёт ai = max(BH); |
| 7. Если i < P-1, то ждать сообщения от задачи i+1 с ai+1 |
| 8. Если 0 < i < P-1, ai = max(ai+1, ai). Если i = 0, a = max(ai+1, ai) |
| 9. Если i > 0, то отправить сообщение задаче i-1 с ai |
| 10. Если i > 0, то ждать сообщения от задачи i-1 c a |
| 11. Если i < P-1, то отправить сообщение задаче i+1 c a |
| 12. Счёт AH = a \* CH + alpha \* T \* (MOH \* MK) |
| 13. Если i < P-1, то ждать сообщения от задачи i+1 с AH,..,AH задач j>i |
| 14. Если i > 0, то отправить сообщение задаче i-1 с AH,..,AH задач j>i |
| 15. Если i = 0, вывод А |

**Этап 3**. Разработка структурной схемы взаимодействия задач.



**Этап 4**. Разработка программы

gnatmake -d -gnatl -P\\VBOXSVR\Desktop\PRO\lab7\lab7.gpr

gcc -c -gnat12 -gnatl -I- -gnatA \\VBOXSVR\Desktop\PRO\lab7\lab7.adb

GNAT GPL 2012 (20120509)

Copyright 1992-2012, Free Software Foundation, Inc.

Compiling: \\VBOXSVR\Desktop\PRO\lab7\lab7.adb (source file time stamp: 2013-04-21 18:13:12)

1. ------------------------------

2. -- PDC. Lab#7

3. -- A = max(B) \* C + alpha \* T \* (MO \* MK)

4. -- Rakovtsii Vitalii, IO-01

5. -- 21.04.2013

6. -------------------------------

7.

8. with Ada.Text\_IO, Ada.Integer\_Text\_IO, Data;

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

10.

11. procedure Lab7 is

12. N: Integer;

13. P: Integer;

14.

15. procedure Setup is

16. package DataInstance is new Data(N => N);

17. use DataInstance;

18.

19. H: Integer := N / P;

20.

21. alpha: Integer;

22. B, C, T: Vector;

23. MK, MO: Matrix;

24.

25. function count\_lindex(id: Integer) return Integer is

26. begin

27. return id \* H + 1;

28. end count\_lindex;

29.

30. function count\_rindex(id: Integer) return Integer is

31. begin

32. if id = (P - 1) then

33. return N;

34. end if;

35.

36. return (id + 1) \* H;

37. end count\_rindex;

38.

39. protected ResAlpha is

40. procedure l\_output (A : in Vector);

41. end ResAlpha;

42.

43. protected body ResAlpha is

44. procedure l\_output (A : in Vector) is

45. begin

46. output(A);

47. end l\_output;

48. end ResAlpha;

49.

50. task type FancyTask(id: Integer) is

51. entry receiveInputFromPrev(B: VectorBase; MO: MatrixBase; alpha: Integer);

52. entry receiveInputFromNext(MK: Matrix; C: VectorBase; T: Vector);

53.

54. entry receiveMaxFromPrev(max: Integer);

55. entry receiveMaxFromNext(max: Integer);

56.

57. entry receiveOutputFromNext(A: VectorBase);

58. end FancyTask;

59.

60. type FancyTaskAccess is access FancyTask;

61. FancyArray: array(0..P-1) of FancyTaskAccess;

62.

63. task body FancyTask is

64. lindex: Integer := 0;

65. rindex: Integer := 0;

66.

67. l\_alpha: Integer;

68. l\_A, l\_B, l\_C, l\_T: Vector;

69. l\_MK, l\_MO: Matrix;

70.

71. l\_max: Integer := Integer'First;

72. begin

73. put\_line("Task " & Integer'Image(id) & " started");

74.

75. lindex := count\_lindex(id);

76. rindex := count\_rindex(id);

77.

78. -- INPUT

79. if id = 0 then

80. generate(B);

81. generate(MO);

82. alpha := 1;

83.

84. l\_alpha := alpha;

85. l\_B := B;

86. l\_MO := MO;

87. elsif id = (P - 1) then

88. generate(MK);

89. generate(C);

90. generate(T);

91.

92. l\_MK := MK;

93. l\_C := C;

94. l\_T := T;

95. end if;

96. ---------------------------------------------------------

97.

98.

99. ---------------------------------------------------------

100. if id > 0 then

101. accept receiveInputFromPrev (B: VectorBase; MO: MatrixBase; alpha: Integer) do

102. l\_B(B'Range) := B;

103. l\_MO(MO'Range) := MO;

104. l\_alpha := alpha;

105. end receiveInputFromPrev;

106. end if;

107.

108. if id < (P - 1) then

109. declare

110. s\_lindex: Integer := count\_lindex(id + 1);

111. s\_B: VectorBase(s\_lindex .. N);

112. s\_MO: MatrixBase(s\_lindex .. N);

113. begin

114. s\_B := l\_B(s\_B'Range);

115. s\_MO := l\_MO(s\_MO'Range);

116.

117. FancyArray(id + 1).all.receiveInputFromPrev(s\_B, s\_MO, l\_alpha);

118. end;

119. end if;

120. ---------------------------------------------------------

121.

122. ---------------------------------------------------------

123. if id < (P - 1) then

124. accept receiveInputFromNext (MK : Matrix; C : VectorBase; T : Vector) do

125. l\_MK := MK;

126. l\_T := T;

127. l\_C(C'Range) := C;

128. end receiveInputFromNext;

129. end if;

130.

131. if id > 0 then

132. declare

133. s\_rindex: Integer := count\_rindex(id - 1);

134. s\_C: VectorBase(1 .. s\_rindex);

135. begin

136. s\_C := l\_C(s\_C'Range);

137.

138. FancyArray(id - 1).all.receiveInputFromNext(l\_MK, s\_C, l\_T);

139. end;

140. end if;

141. ---------------------------------------------------------

142.

143.

144. for i in lindex .. rindex loop

145. if l\_max < l\_B(i) then

146. l\_max := l\_B(i);

147. end if;

148. end loop;

149.

150.

151. ---------------------------------------------------------

152. if id < (P - 1) then

153. accept receiveMaxFromNext (max: Integer) do

154. if max > l\_max then

155. l\_max := max;

156. end if;

157. end receiveMaxFromNext;

158. end if;

159.

160. if id > 0 then

161. FancyArray(id - 1).all.receiveMaxFromNext(l\_max);

162. end if;

163. ---------------------------------------------------------

164.

165. ---------------------------------------------------------

166. if id > 0 then

167. accept receiveMaxFromPrev (max: Integer) do

168. l\_max := max;

169. end receiveMaxFromPrev;

170. end if;

171.

172. if id < (P - 1) then

173. FancyArray(id + 1).all.receiveMaxFromPrev(l\_max);

174. end if;

175. ---------------------------------------------------------

176.

177. declare

178. sum: Integer;

179. begin

180.

181. for i in lindex .. rindex loop

182. l\_A(i) := l\_C(i) \* l\_max;

183.

184. for j in 1..N loop

185. sum := 0;

186.

187. for k in 1..N loop

188. sum := sum + l\_MO(i)(k) \* l\_MK(k)(j);

189. end loop;

190.

191. l\_A(i) := l\_A(i) + sum \* l\_T(j) \* l\_alpha;

192. end loop;

193. end loop;

194. end;

195.

196. ---------------------------------------------------------

197. if id < (P - 1) then

198. accept receiveOutputFromNext (A : VectorBase) do

199. l\_A(A'Range) := A;

200. end receiveOutputFromNext;

201. end if;

202.

203. if id > 0 then

204. declare

205. s\_A: VectorBase(lindex .. N);

206. begin

207. s\_A := l\_A(s\_A'Range);

208.

209. FancyArray(id - 1).all.receiveOutputFromNext(s\_A);

210. end;

211. end if;

212. ---------------------------------------------------------

213.

214. -- OUTPUT

215. case id is

216. when 0 =>

217. if N <= 10 then

218. --ResAlpha.l\_output(l\_A);

219. output(l\_A);

220. end if;

221.

222. put\_line("Task " & Integer'Image(id) & " finished" & Integer'Image(lindex) & Integer'Image(rindex));

223. when others =>

224. put\_line("Task " & Integer'Image(id) & " finished" & Integer'Image(lindex) & Integer'Image(rindex));

225. end case;

226. -----------------------------------------------------------

227. end;

228. begin

229. for i in 0..(P - 1) loop

230. FancyArray(P - 1 - i) := new FancyTask(P - 1 - i);

231. end loop;

232. end Setup;

233. begin

234. Put("Set size of vector/matrix: ") ;

235. Get(N);

236. Put("Numero of les proccesors: ") ;

237. Get(P);

238. Setup;

239. end Lab7;

239 lines: No errors

gcc -c -gnat12 -gnatl -I- -gnatA \\VBOXSVR\Desktop\PRO\lab7\data.adb

GNAT GPL 2012 (20120509)

Copyright 1992-2012, Free Software Foundation, Inc.

Compiling: \\VBOXSVR\Desktop\PRO\lab7\data.adb (source file time stamp: 2013-04-11 19:23:06)

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

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

3.

4.

5. -- Therefore in Ada 83, there existed the restriction that out parameters are write-only.

6. -- If you wanted to read the value written, you had to declare a local variable, do all

7. -- calculations with it, and finally assign it to C before return. This was awkward and

8. -- error prone so the restriction was removed in Ada 95.

9.

10. package body Data is

11. procedure generate(A: out Vector) is

12. begin

13. for i in A'Range loop

14. A(i) := 1;

15. end loop;

16. end generate;

17.

18. procedure generate(MA: out Matrix) is

19. begin

20. for i in MA'Range loop

21. for j in MA'Range loop

22. MA(i)(j) := 1;

23. end loop;

24. end loop;

25. end generate;

26.

27.

28. procedure output(A: in Vector) is

29. begin

30. put("[");

31. for i in A'Range loop

32. put(A(i), Width => 10);

33.

34. if i /= A'Last then

35. put(", ");

36. end if;

37. end loop;

38. put\_line("]");

39. end output;

40.

41. procedure output(MA: in Matrix) is

42. begin

43. put\_line("[");

44. for i in MA'Range loop

45.

46. put(" [");

47. for j in MA'Range loop

48.

49. put(MA(i)(j), Width => 10);

50.

51. if j /= MA'Last then

52. put(", ");

53. end if;

54. end loop;

55. put("]");

56.

57. if i /= MA'Last then

58. put\_line(", ");

59. else

60. new\_line;

61. end if;

62. end loop;

63. put\_line("]");

64. end output;

65. end Data;

Compiling: \\VBOXSVR\Desktop\PRO\lab7\data.ads (source file time stamp: 2013-04-21 13:31:54)

1. generic

2. -- Vector & Matrix dimensions

3. N: integer;

4.

5. package Data is

6. type VectorBase is array(Integer range <>) of Integer;

7. subtype Vector is VectorBase(1..N);

8. type MatrixBase is array(Integer range <>) of Vector;

9. subtype Matrix is MatrixBase(1..N);

10.

11. procedure generate(A: out Vector);

12.

13. procedure generate(MA: out Matrix);

14.

15. procedure output(A: in Vector);

16.

17. procedure output(MA: in Matrix);

18. end Data;

19.

20.

65 lines: No errors

gnatbind -I- -x \\VBOXSVR\Desktop\PRO\lab7\lab7.ali

gnatlink \\VBOXSVR\Desktop\PRO\lab7\lab7.ali -o \\VBOXSVR\Desktop\PRO\lab7\lab7.exe

[2013-04-21 21:17:51] process terminated successfully (elapsed time: 01.93s)