НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

КАФЕДРА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ

Лабораторна робота №5

з дисципліни **«**Програмування паралельних комп’ютерних систем**»**

Виконав:

студент 3 курсу

ФІОТ гр. ІО-34

Кривоносов Олексій

Перевірив:

Корочкін О. В.

Київ – 2016 р.

Тема: Програмування для комп’ютерних систем зі спільною пам’яттю.

Розробити програму для розв’язання в ПКС зі СП математичної задачі: MA = (B∙C)∙MO + α∙(MT∙MR)

Мова програмування: Ада

Засоби організації взаємодії: захищений модуль



Структурна схема ПКС

**Виконання роботи:**

**Етап 1. Побудова паралельного алгоритму**

1. vi = BH\*CH, *i* =
2. v = ∑vi, *i* =
3. MAH = v∙MOH + α∙(MTH∙MR)

Спільний ресурс: α, v, MR

**Етап 2. Розроблення алгоритмів роботи кожного процесу**

|  |  |
| --- | --- |
| **Задача Т1** |  |
| 1. Введення MT, MR |  |
| 2. **Сигнал** про завершення вводу | S1-2,3,4 |
| 3. **Чекати** вводу в інших задачах | W1-2,3,4 |
| 4. Обчислення 1: v1 = BH∙CH |  |
| 5. Обчислення 2: v = v+v1 | КД |
| 6. **Сигнал** задачам про завершення обчислення 2. | S2-2,3,4 |
| 7. **Чекати** завершення обчислення 2 в інших задачах | W2-2,3,4 |
| 8. Копіювати α1 := α, v1 = v, MR1 = MR | КД |
| 9. Обчислення 3: MAH = v1∙MOH + α1∙(MTH∙MR1) |  |
| 10. **Сигнал** про завершення обчислення 3 | S3-2,3,4 |
| **Задача Т2** |  |
| 1. **Чекати** на завершення вводу в інших задачах | W1-1,3,4 |
| 2. Обчислення 1: v2 = BH∙CH |  |
| 3. Обчислення 2: v = v+v2 | КД |
| 4. **Сигнал** задачам про завершення обчислення 2. | S1-1,3,4 |
| 5. **Чекати** завершення обчислення 2 в інших задачах | W2-1,3,4 |
| 6. Копіювати α2 := α, v2 = v, MR2 = MR | КД |
| 7. Обчислення 3: MAH = v2∙MOH + α2∙(MTH∙MR2) |  |
| 8. **Сигнал** про завершення обчислення 3. | S2-1,3,4 |
| **Задача Т3** |  |
| 1. ВведенняB, MA |  |
| 2. **Сигнал** задачам про завершення вводу | S1-1,2,4 |
| 3. **Чекати** вводу в інших задачах | W1-1,2,4 |
| 4. Обчислення 1: v3 = BH∙CH |  |
| 5. Обчислення 2: v = v+v3 | КД |
| 6. **Сигнал** задачам про завершення обчислення 2. | S2-1,2,4 |
| 7. **Чекати** завершення обчислення 2 в інших задачах | W2-1,2,4 |
| 8. Копіювати α3 := α, v3 = v, MR3 = MR | КД |
| 9. Обчислення 3: MAH = v3∙MOH + α3∙(MTH∙MR3) |  |
| 10. **Чекати** на завершення обчислення 3 в інших задачах | W3-1,2,4 |
| 11. Вивести MA |  |
| **Задачa T4** |  |
| 1. **Введення** С, MO, α |  |
| 2. Сигнал задачам про завершення вводу | S1-1,2,3 |
| 3. **Чекати** вводу в інших задачах | W1-1,2,3 |
| 4. Обчислення 1: v4 = BH∙CH |  |
| 5. Обчислення 2: v = v+v4 |  |
| 6. **Сигнал** задачам про завершення обчислення 2. | S2-1,2,3 |
| 7. **Чекати** завершення обчислення 2 в інших задачах | W2-1,2,3 |
| 8. Копіювати α4 := α, v4 = v, MR4 = MR |  |
| 9. Обчислення 3: MAH = v4∙MOH + α4∙(MTH∙MR4) |  |
| 10. **Сигнал** про завершення обчислення 3 | S3-1,2,3 |

**Етап 3. Розроблення структурної схеми взаємодії задач**



Етап 4. Розроблення програми

GNAT GPL 2015 (20150428-49)

Copyright 1992-2015, Free Software Foundation, Inc.

Compiling: lab5.adb

Source file time stamp: 2016-04-18 11:53:16

Compiled at: 2016-04-18 14:54:20

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

2. -- --

3. -- Parallel and Distributed Computing --

4. -- Laboratory work #5. Ada. Protected unit --

5. -- --

6. -- File:lab5.adb --

7. -- Task: MA = (B\*C)\*MO+ a(MT\*MR) --

8. -- --

9. -- Author: Krivonosov Oleksii, group IO-34 --

10. -- Date: 14.04.2016 --

11. -- --

12. ------------------------------------------------------------------

13.

14. with Ada.Text\_IO, Ada.Integer\_text\_iO, Ada.Synchronous\_Task\_Control, Data;

15. use Ada.Text\_IO, Ada.Integer\_text\_iO, Ada.Synchronous\_Task\_Control;

16.

17. procedure Lab5 is

18. Value : Integer :=1;

19. N: Natural :=1000;

20. package DataN is new Data(N);

21. use DataN;

22.

23. P: Natural :=4;

24. H: Natural:= N/P;

25.

26. B, C : Vector;

27. MO, MT, MA: Matrix;

28. buf : Matrix;

29. ------------------------------------------------------------------

30.

31. ------------------------------------------------------------------

32. protected Synchronization is

33. entry WaitForInput;

34. entry WaitForCalcV;

35. entry WaitForCalcMA;

36. procedure InputSignal;

37. procedure CalcVSignal;

38. procedure CalcMASignal;

39. private

40. inputFlag:Natural:=0;

41. vFlag:Natural:=0;

42. MAflag:Natural:=0;

43. end Synchronization;

44.

45. protected GeneralResourse is

46. procedure addV(data : in Integer);

47. procedure setMR(data : in Matrix);

48. procedure setAlfa(data : in Integer);

49.

50. function CopyAlfa return Integer;

51. function CopyV return Integer;

52. function CopyMR return Matrix;

53.

54.

55. private

56. alfa: Integer;

57. v: Integer:=0;

58. MR:Matrix;

59.

60. end GeneralResourse;

61. ------------------------------------------------------------------

62.

63. ------------------------------------------------------------------

64. protected body Synchronization is

65.

66. procedure InputSignal is

67. begin

68. inputFlag := inputFlag + 1;

69. end InputSignal;

70.

71. procedure CalcVSignal is

72. begin

73. vFlag := vFlag + 1;

74. end CalcVSignal;

75.

76. procedure CalcMASignal is

77. begin

78. MAflag := MAflag + 1;

79. end CalcMASignal;

80.

81. entry WaitForInput

82. when inputFlag = 3 is

83. begin

84. null;

85. end WaitForInput;

86.

87. entry WaitForCalcV

88. when vFlag = 4 is

89. begin

90. null;

91. end WaitForCalcV;

92.

93. entry WaitForCalcMA

94. when MAflag = 3 is

95. begin

96. null;

97. end WaitForCalcMA;

98.

99. end Synchronization;

100.

101. protected body GeneralResourse is

102.

103. procedure addV(data : in Integer) is

104. begin

105. v := v+data;

106. end addV;

107.

108. procedure setAlfa(data : in Integer) is

109. begin

110. alfa := data;

111. end setAlfa;

112.

113.

114. procedure setMR(data : in Matrix) is

115. begin

116. MR:=data;

117. end setMR;

118. function CopyAlfa return Integer is

119. begin

120. return alfa;

121. end CopyAlfa;

122.

123. function CopyV return Integer is

124. begin

125. return v;

126. end CopyV;

127.

128. function CopyMR return Matrix is

129. begin

130. return MR;

131. end CopyMR;

132. end GeneralResourse;

133.

134. procedure StartTasks is

135. ------------------------------------------------------------------

136. ------------------------ Task 1 ----------------------------------

137. ------------------------------------------------------------------

138. task T1;

139.

140. MR1:Matrix;

141. v1:Integer;

142. alfa1:Integer;

143. task body T1 is

144. begin

145. Put\_Line ("T1 started");

146.

147. Input(Value, MT);

148. Input(Value, buf);

149. GeneralResourse.setMR(buf);

150.

151. Synchronization.InputSignal;

152.

153. Synchronization.WaitForInput;

154.

155. v1 :=0;

156. for i in 1..H loop

157. v1:=v1+B(i)\*C(i);

158. end loop;

159.

160. GeneralResourse.addV(v1);

161.

162. Synchronization.CalcVSignal;

163.

164. Synchronization.WaitForCalcV;

165.

166. alfa1 := GeneralResourse.CopyAlfa;

167. MR1 := GeneralResourse.CopyMR;

168. v1 := GeneralResourse.CopyV;

169.

170.

171. for i in 1..H loop

172. for j in 1..N loop

173. MA(i)(j) :=0;

174. for k in 1..N loop

175. MA(i)(j) := MA(i)(j) + MT(i)(k) \* MR1(k)(j);

176. end loop;

177. MA(i)(j) := MA(i)(j) \* alfa1 + v1 \* MO(i)(j);

178. end loop;

179. end loop;

180.

181. Synchronization.CalcMASignal;

182. Put\_Line ("T1 finished");

183. end T1;

184. ------------------------------------------------------------------

185. ------------------------ Task 2 ----------------------------------

186. ------------------------------------------------------------------

187. task T2;

188.

189. v2: Integer;

190. alfa2:Integer;

191. MR2: Matrix;

192. task body T2 is

193. begin

194. Put\_Line ("T2 started");

195.

196. Synchronization.WaitForInput;

197.

198. v2 :=0;

199. for i in H+1..2\*H loop

200. v2:=v2+B(i)\*C(I);

201. end loop;

202.

203. GeneralResourse.addV(v2);

204.

205. Synchronization.CalcVSignal;

206. Synchronization.WaitForCalcV;

207.

208. alfa2 := GeneralResourse.CopyAlfa;

209. MR2 := GeneralResourse.CopyMR;

210. v2 := GeneralResourse.CopyV;

211.

212.

213. for i in H+1..2\*H loop

214. for j in 1..N loop

215. MA(i)(j) :=0;

216. for k in 1..N loop

217. MA(i)(j) := MA(i)(j) + MT(i)(k) \* MR2(k)(j);

218. end loop;

219. MA(i)(j) := MA(i)(j) \* alfa2 + v2 \* MO(i)(j);

220. end loop;

221. end loop;

222.

223. Synchronization.CalcMASignal;

224. Put\_Line ("T2 finished");

225. end T2;

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

227. ------------------------ Task 3 ----------------------------------

228. ------------------------------------------------------------------

229. task T3;

230.

231. v3: Integer;

232. alfa3:Integer;

233. MR3: Matrix;

234. task body T3 is

235. begin

236. Put\_Line ("T3 started");

237.

238. Input(Value, MA);

239. Input(Value, B);

240.

241. Synchronization.InputSignal;

242.

243. Synchronization.WaitForInput;

244.

245. v3 :=0;

246. for i in 2\*H+1..3\*H loop

247. v3:=v3+B(i)\*C(I);

248. end loop;

249.

250. GeneralResourse.addV(v3);

251.

252. Synchronization.CalcVSignal;

253.

254. Synchronization.WaitForCalcV;

255.

256. alfa3 := GeneralResourse.CopyAlfa;

257. MR3 := GeneralResourse.CopyMR;

258. v3 := GeneralResourse.CopyV;

259.

260.

261. for i in 2\*H+1..3\*H loop

262. for j in 1..N loop

263. MA(i)(j) :=0;

264. for k in 1..N loop

265. MA(i)(j) := MA(i)(j) + MT(i)(k) \* MR3(k)(j);

266. end loop;

267. MA(i)(j) := MA(i)(j) \* alfa3 + v3 \* MO(i)(j);

268. end loop;

269. end loop;

270.

271. Synchronization.WaitForCalcMA;

272.

273. if N<6 then

274. Output(MA);

275. end if;

276. Put\_Line ("T3 finished");

277. end T3;

278. ------------------------------------------------------------------

279. ------------------------ Task 4 ----------------------------------

280. ------------------------------------------------------------------

281. task T4;

282.

283. v4: Integer;

284. alfa4:Integer;

285. MR4: Matrix;

286. task body T4 is

287. begin

288. Put\_Line ("T4 started");

289. GeneralResourse.setAlfa(Value);

290. Input(Value, C);

291. Input(Value, MO);

292.

293.

294.

295. Synchronization.InputSignal;

296.

297. Synchronization.WaitForInput;

298

300. v4 :=0;

301. for i in 3\*H+1..4\*H loop

302. v4:=v4+B(i)\*C(I);

303. end loop;

304.

305. GeneralResourse.addV(v4);

306. Synchronization.CalcVSignal;

307. Synchronization.WaitForCalcV;

308. alfa4 := GeneralResourse.CopyAlfa;

309. MR4 := GeneralResourse.CopyMR;

310. v4 := GeneralResourse.CopyV;

311.

312.

313. for i in 3\*H+1..4\*H loop

314. for j in 1..N loop

315. MA(i)(j) :=0;

316. for k in 1..N loop

317. MA(i)(j) := MA(i)(j) + MT(i)(k) \* MR4(k)(j);

318. end loop;

319. MA(i)(j) := MA(i)(j) \* alfa4 + v4 \* MO(i)(j);

320. end loop;

321. end loop;

322.

323.

324.

325. Synchronization.CalcMASignal;

326.

327. Put\_Line ("T4 finished");

328. end T4;

329. ------------------------------------------------------------------

330.

331. ------------------------------------------------------------------

332. begin

333. null;

334. end StartTasks;

335.

336. begin

337. Put\_Line ("Lab5 started");

338. StartTasks;

339. Put\_Line ("Lab5 finished");

340. end Lab5;

340 lines: No errors

GNAT GPL 2015 (20150428-49)

Copyright 1992-2015, Free Software Foundation, Inc.

Compiling: data.adb

Source file time stamp: 2016-04-18 11:48:58

Compiled at: 2016-04-18 14:54:05

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

2. -- --

3. -- Parallel and Distributed Computing --

4. -- Laboratory work #5. Ada. Protected unit --

5. -- --

6. -- File:Data.adb --

7. -- Task: MA = (B\*C)\*MO+ a(MT\*MR) --

8. -- --

9. -- Author: Krivonosov Oleksii, group IO-34 --

10. -- Date: 14.04.2016 --

11. -- --

12. ------------------------------------------------------------------

13.

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

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

16.

17. package body Data is

18.

19. procedure Input(Value : Integer; V : out Vector) is

20. begin

21. for i in 1..N loop

22. V(i):=Value;

23. end loop;

24. end Input;

25. procedure Input (Value : Integer; MA : out Matrix) is

26. begin

27. for I in 1..N loop

28. for J in 1..N loop

29. MA(I)(J) := Value;

30. end loop;

31. end loop;

32. end Input;

33.

34.

35. procedure Output (V : in Vector) is

36. begin

37. New\_Line;

38. for I in 1..N loop

39. Put(Item => V(I), Width => 4);

40. end loop;

41. New\_Line;

42. end Output;

43.

44. procedure Output (MA : in Matrix) is

45. begin

46. New\_Line;

47. for I in 1..N loop

48. for J in 1..N loop

49. Put(MA(i)(j));

50. end loop;

51. New\_line;

52. end loop;

53. New\_Line;

54. end Output;

55. end Data;

Compiling: data.ads

Source file time stamp: 2016-04-18 11:17:44

Compiled at: 2016-04-18 14:54:05

1. with Ada.Text\_IO;

2. use Ada.Text\_IO;

3.

4. generic

5. N: in Natural;

6.

7. package Data is

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

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

10. procedure Input (Value : in Integer; V : out Vector);

11. procedure Input (Value : in Integer; MA : out Matrix);

12. procedure Output (V : in Vector);

13. procedure Output (MA : in Matrix);

14. end Data;

56 lines: No errors