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-21

Кузьменко Володимир

Перевірив:

Корочкін О. В.

Київ – 2015 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** | КД |
| 1. Введення MT, α |  |
| 2. **Сигнал** про завершення вводу |  |
| 3. **Чекати** вводу в інших задачах |  |
| 4. Обчислення 1: v1 = BH∙CH |  |
| 5. Обчислення 2: v = v+v1 | КД |
| 6. **Сигнал** задачам про завершення обчислення 2. |  |
| 7. **Чекати** завершення обчислення 2 в інших задачах |  |
| 8. Копіювати α1 := α, v1 = v, MR1 = MR | КД |
| 9. Обчислення 3: MAH = v1∙MOH + α1∙(MTH∙MR1) |  |
| 10. **Сигнал** про завершення обчислення 3 |  |
| **Задача T3** | КД |
| 1. **Введення** С, MO |  |
| 2. Сигнал задачам про завершення вводу |  |
| 3. **Чекати** вводу в інших задачах |  |
| 4. Обчислення 1: v3 = BH∙CH |  |
| 5. Обчислення 2: v = v+v3 | КД |
| 6. **Сигнал** задачам про завершення обчислення 2. |  |
| 7. **Чекати** завершення обчислення 2 в інших задачах |  |
| 8. Копіювати α3 := α, v3 = v, MR3 = MR | КД |
| 9. Обчислення 3: MAH = v3∙MOH + α3∙(MTH∙MR3) |  |
| 10. **Сигнал** про завершення обчислення 3 |  |
| **Задача Т6** | КД |
| 1. ВведенняB, MR |  |
| 2. **Сигнал** задачам про завершення вводу |  |
| 3. **Чекати** вводу в інших задачах |  |
| 4. Обчислення 1: v6 = BH∙CH |  |
| 5. Обчислення 2: v = v+v6 | КД |
| 6. **Сигнал** задачам про завершення обчислення 2. |  |
| 7. **Чекати** завершення обчислення 2 в інших задачах |  |
| 8. Копіювати α6 := α, v6 = v, MR6 = MR | КД |
| 9. Обчислення 3: MAH = v6∙MOH + α6∙(MTH∙MR6) |  |
| 10. **Чекати** на завершення обчислення 3 в інших задачах |  |
| 11. Вивести MA |  |
| **Задачі Тi, (i=2,4,5)** |  |
| 1. **Чекати** на завершення вводу в інших задачах |  |
| 2. Обчислення 1: vi = BH∙CH |  |
| 3. Обчислення 2: v = v+vi | КД |
| 4. **Сигнал** задачам про завершення обчислення 2. |  |
| 5. **Чекати** завершення обчислення 2 в інших задачах |  |
| 6. Копіювати αi := α, vi = v, MRi = MR | КД |
| 7. Обчислення 3: MAH = vi∙MOH + αi∙(MTH∙MRi) |  |
| 8. **Сигнал** про завершення обчислення 3. |  |

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



Етап 4. Розроблення програми

GNAT GPL 2014 (20140331)

Copyright 1992-2014, Free Software Foundation, Inc.

Compiling: lab5.adb (source file time stamp: 2015-04-14 18:47:06)

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

2. -- --

3. -- Parallel and Distributed Computing --

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

5. -- --

6. -- File: pro2\_lab5.adb --

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

8. -- --

9. -- Author: Kuzmenko Volodymyr, group IO-21 --

10. -- Date: 14.04.2015 --

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 :=12;

20. package DataN is new Data(N);

21. use DataN;

22.

23. P: Natural :=6;

24. H: Natural:= N/P;

25.

26. B, C : Vector;

27. MO, MT, MA: Matrix;

28.

29. buf : Matrix;

30. ------------------------------------------------------------------

31.

32. ------------------------------------------------------------------

33. protected Synchronization is

34. entry WaitForInput;

35. entry WaitForCalcV;

36. entry WaitForCalcMA;

37. procedure InputSignal;

38. procedure CalcVSignal;

39. procedure CalcMASignal;

40. private

41. inputFlag:Natural:=0;

42. vFlag:Natural:=0;

43. MAflag:Natural:=0;

44. end Synchronization;

45.

46. protected GeneralResourse is

47. procedure addV(data : in Integer);

48. procedure setMR(data : in Matrix);

49. procedure setAlfa(data : in Integer);

50.

51. function CopyAlfa return Integer;

52. function CopyV return Integer;

53. function CopyMR return Matrix;

54.

55.

56. private

57. alfa: Integer;

58. v: Integer:=0;

59. MR:Matrix;

60.

61. end GeneralResourse;

62. ------------------------------------------------------------------

63.

64. ------------------------------------------------------------------

65. protected body Synchronization is

66.

67. procedure InputSignal is

68. begin

69. inputFlag := inputFlag + 1;

70. end InputSignal;

71.

72. procedure CalcVSignal is

73. begin

74. vFlag := vFlag + 1;

75. end CalcVSignal;

76.

77. procedure CalcMASignal is

78. begin

79. MAflag := MAflag + 1;

80. end CalcMASignal;

81.

82. entry WaitForInput

83. when inputFlag = 3 is

84. begin

85. null;

86. end WaitForInput;

87.

88. entry WaitForCalcV

89. when vFlag = 6 is

90. begin

91. null;

92. end WaitForCalcV;

93.

94. entry WaitForCalcMA

95. when MAflag = 5 is

96. begin

97. null;

98. end WaitForCalcMA;

99.

100. end Synchronization;

101.

102.

103. protected body GeneralResourse is

104.

105. procedure addV(data : in Integer) is

106. begin

107. v := v+data;

108. end addV;

109.

110. procedure setAlfa(data : in Integer) is

111. begin

112. alfa := data;

113. end setAlfa;

114.

115.

116. procedure setMR(data : in Matrix) is

117. begin

118. MR:=data;

119. end setMR;

120.

121. function CopyAlfa return Integer is

122. begin

123. return alfa;

124. end CopyAlfa;

125.

126. function CopyV return Integer is

127. begin

128. return v;

129. end CopyV;

130.

131. function CopyMR return Matrix is

132. begin

133. return MR;

134. end CopyMR;

135.

136. end GeneralResourse;

137.

138. procedure StartTasks is

139. ------------------------------------------------------------------

140.

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

142. task T1;

143.

144. MR1:Matrix;

145. v1:Integer;

146. alfa1:Integer;

147. task body T1 is

148. begin

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

150.

151. Input(Value, MT);

152. GeneralResourse.setAlfa(Value);

153.

154. Synchronization.InputSignal;

155.

156. Synchronization.WaitForInput;

157.

158. v1 :=0;

159. for i in 1..H loop

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

161. end loop;

162.

163. GeneralResourse.addV(v1);

164.

165. Synchronization.CalcVSignal;

166.

167. Synchronization.WaitForCalcV;

168.

169. alfa1 := GeneralResourse.CopyAlfa;

170. MR1 := GeneralResourse.CopyMR;

171. v1 := GeneralResourse.CopyV;

172.

173.

174. for i in 1..H loop

175. for j in 1..N loop

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

177. for k in 1..N loop

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

179. end loop;

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

181. end loop;

182. end loop;

183.

184.

185. Synchronization.CalcMASignal;

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

187. end T1;

188. ------------------------------------------------------------------

189.

190. ------------------------------------------------------------------

191. task T2;

192.

193. v2: Integer;

194. alfa2:Integer;

195. MR2: Matrix;

196. task body T2 is

197. begin

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

199.

200. Synchronization.WaitForInput;

201.

202. v2 :=0;

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

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

205. end loop;

206.

207. GeneralResourse.addV(v2);

208.

209. Synchronization.CalcVSignal;

210.

211. Synchronization.WaitForCalcV;

212.

213. alfa2 := GeneralResourse.CopyAlfa;

214. MR2 := GeneralResourse.CopyMR;

215. v2 := GeneralResourse.CopyV;

216.

217.

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

219. for j in 1..N loop

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

221. for k in 1..N loop

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

223. end loop;

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

225. end loop;

226. end loop;

227.

228.

229. Synchronization.CalcMASignal;

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

231. end T2;

232. ------------------------------------------------------------------

233.

234. ------------------------------------------------------------------

235. task T3;

236.

237. v3: Integer;

238. alfa3:Integer;

239. MR3: Matrix;

240. task body T3 is

241. begin

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

243.

244. Input(Value, C);

245. Input(Value, MO);

246.

247. Synchronization.InputSignal;

248.

249. Synchronization.WaitForInput;

250.

251. v3 :=0;

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

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

254. end loop;

255.

256. GeneralResourse.addV(v3);

257.

258. Synchronization.CalcVSignal;

259.

260. Synchronization.WaitForCalcV;

261.

262. alfa3 := GeneralResourse.CopyAlfa;

263. MR3 := GeneralResourse.CopyMR;

264. v3 := GeneralResourse.CopyV;

265.

266.

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

268. for j in 1..N loop

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

270. for k in 1..N loop

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

272. end loop;

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

274. end loop;

275. end loop;

276.

277.

278. Synchronization.CalcMASignal;

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

280. end T3;

281. ------------------------------------------------------------------

282.

283. ------------------------------------------------------------------

284. task T4;

285.

286. v4: Integer;

287. alfa4:Integer;

288. MR4: Matrix;

289. task body T4 is

290. begin

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

292.

293. Synchronization.WaitForInput;

294.

295. v4 :=0;

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

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

298. end loop;

299.

300. GeneralResourse.addV(v4);

301.

302. Synchronization.CalcVSignal;

303.

304. Synchronization.WaitForCalcV;

305.

306. alfa4 := GeneralResourse.CopyAlfa;

307. MR4 := GeneralResourse.CopyMR;

308. v4 := GeneralResourse.CopyV;

309.

310.

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

312. for j in 1..N loop

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

314. for k in 1..N loop

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

316. end loop;

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

318. end loop;

319. end loop;

320.

321.

322. Synchronization.CalcMASignal;

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

324. end T4;

325. ------------------------------------------------------------------

326.

327. ------------------------------------------------------------------

328. task T5;

329.

330. v5: Integer;

331. alfa5:Integer;

332. MR5: Matrix;

333. task body T5 is

334. begin

335. Put\_Line ("T5 started");

336.

337. Synchronization.WaitForInput;

338.

339. v5 :=0;

340. for i in 4\*H+1..5\*H loop

341. v5:=v5+B(i)\*C(I);

342. end loop;

343.

344. GeneralResourse.addV(v5);

345.

346. Synchronization.CalcVSignal;

347.

348. Synchronization.WaitForCalcV;

349.

350. alfa5 := GeneralResourse.CopyAlfa;

351. MR5 := GeneralResourse.CopyMR;

352. v5 := GeneralResourse.CopyV;

353.

354.

355. for i in 4\*H+1..5\*H loop

356. for j in 1..N loop

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

358. for k in 1..N loop

359. MA(i)(j) := MA(i)(j) + MT(i)(k) \* MR5(k)(j);

360. end loop;

361. MA(i)(j) := MA(i)(j) \* alfa5 + v5 \* MO(i)(j);

362. end loop;

363. end loop;

364.

365.

366. Synchronization.CalcMASignal;

367. Put\_Line ("T5 finished");

368. end T5;

369. ------------------------------------------------------------------

370.

371. ------------------------------------------------------------------

372. task T6;

373.

374. v6: Integer;

375. alfa6:Integer;

376. MR6: Matrix;

377. task body T6 is

378.

379. begin

380. Put\_Line ("T6 started");

381.

382. Input(Value, B);

383. Input(Value, buf);

384. GeneralResourse.setMR(buf);

385.

386. Synchronization.InputSignal;

387.

388. Synchronization.WaitForInput;

389.

390. v6 :=0;

391. for i in 5\*H+1..N loop

392. v6:=v6+B(i)\*C(I);

393. end loop;

394.

395. GeneralResourse.addV(v6);

396.

397. Synchronization.CalcVSignal;

398.

399. Synchronization.WaitForCalcV;

400.

401.

402. alfa6 := GeneralResourse.CopyAlfa;

403. MR6 := GeneralResourse.CopyMR;

404. v6 := GeneralResourse.CopyV;

405.

406.

407. for i in 5\*H+1..N loop

408. for j in 1..N loop

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

410. for k in 1..N loop

411. MA(i)(j) := MA(i)(j) + MT(i)(k) \* MR6(k)(j);

412. end loop;

413. MA(i)(j) := MA(i)(j) \* alfa6 + v6 \* MO(i)(j);

414. end loop;

415. end loop;

416.

417.

418. Synchronization.WaitForCalcMA;

419.

420. Output(MA);

421. Put\_Line ("T6 finished");

422. end T6;

423.

424. begin

425. null;

426. end StartTasks;

427.

428. begin

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

430. StartTasks;

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

432.

433. end Lab5;

433 lines: No errors

GNAT GPL 2014 (20140331)

Copyright 1992-2014, Free Software Foundation, Inc.

Compiling: data.ads (source file time stamp: 2015-04-14 18:07:02)

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;

14 lines: No errors

GNAT GPL 2014 (20140331)

Copyright 1992-2014, Free Software Foundation, Inc.

Compiling: data.adb (source file time stamp: 2015-04-14 18:07:02)

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

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

3.

4. package body Data is

5.

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

7. begin

8. for i in 1..N loop

9. V(i):=Value;

10. end loop;

11. end Input;

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

13. begin

14. for I in 1..N loop

15. for J in 1..N loop

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

17. end loop;

18. end loop;

19. end Input;

20.

21.

22. procedure Output (V : in Vector) is

23. begin

24. New\_Line;

25. for I in 1..N loop

26. Put(Item => V(I), Width => 6);

27. end loop;

28. New\_Line;

29. end Output;

30.

31. procedure Output (MA : in Matrix) is

32. begin

33. New\_Line;

34. for I in 1..N loop

35. for J in 1..N loop

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

37. end loop;

38. New\_line;

39. end loop;

40. New\_Line;

41. end Output;

42.

43.

44. end Data;

Compiling: data.ads (source file time stamp: 2015-04-14 18:07:02)

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;

44 lines: No errors