НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

КАФЕДРА АВТОМАТИЗАЦІЇ СИСТЕМ ОБРОБКИ ІНФОРМАЦІЇ ТА УПРАВЛІННЯ

Лабораторна робота №5

з дисципліни **«**Паралельне програмування**»**

Виконав:

студент 3 курсу

ФІОТ гр. ІП-31

Кахерський О.І.

Перевірив:

Корочкін О. В.

Київ – 2016 р.

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

**Технічне завдання:** Розробити програму для розв’язання ПКС із СП (структура на рис. 1) математичної задачі: A =

**Бібліотека: OpenMP.**

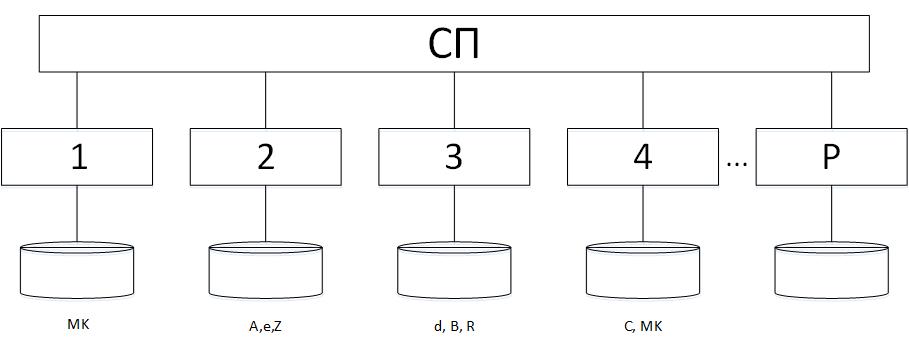


Рис.1 Структурна схема ПКС

**Виконання роботи**

**Етап 1.** Побудова паралельного алгоритму

2. =

Спільний ресурс: d, e, x, MK, R

**Етап 2. Розробка алогоритмів роботи кожного процесу**

i = (5 .. n), tID – номер поточної задачі, j = i/tID.

**Задача Т1**

|  |  |  |
| --- | --- | --- |
| **№** | **Дія** | **ТС/КД** |
| 1. | Ввід MK |  |
| 2. | Надіслати сигнал Т2..n про завершення вводу |  |
| 3. | Очікувати сигнал про завершення вводу в Т2..4 |  |
| 4. | Обчислити |  |
| 5. | Обчислити x = x + x1 | КД |
| 6. | Надіслати сигнал про завершення обчислення a задачам Т2..n |  |
| 7. | Очікувати сигнал про завершення обчислення x в задачах T2..n |  |
| 8. | Копіювати , e1 = e, MK1 = MK, х1 = х, R1 = R | КД |
| 9. | Виконати обчислення = |  |
| 10. | Надіслати сигнал Т2 про завершення обчислень |  |

**Задача Т2**

|  |  |  |
| --- | --- | --- |
| **№** | **Дія** | **ТС/КД** |
| 1. | Ввід e,Z |  |
| 2. | Надіслати сигнал Т1,3,4..n про завершення вводу |  |
| 3. | Очікувати сигнал про завершення вводу в Т1,3,4 |  |
| 4. | Обчислити |  |
| 5. | Обчислити х = х+х2 | КД |
| 6. | Надіслати сигнал про завершення обчислення х задачам Т1,3..n |  |
| 7. | Очікувати завершення обчислення x в задачах Т1,3..n | , |
| 8. | Копіювати , e2 = e, MK2 = MK, x2 = x, R2 = R | КД |
| 9. | Виконати обчислення = |  |
| 10. | Очікування завершення обчислення в Т1,3,4..n |  |
| 11. | Вивід А |  |

**Задача Т3**

|  |  |  |
| --- | --- | --- |
| **№** | **Дія** | **ТС/КД** |
| 1. | Ввід d,B,R |  |
| 2. | Надіслати сигнал Т1,2,4..n про завершення вводу |  |
| 3. | Очікувати сигнал про завершення вводу в Т1,2,4 |  |
| 4. | Обчислити |  |
| 5. | Обчислити х = х+х3 | КД |
| 6. | Надіслати сигнал про завершення обчислення х задачам Т1,2,4..n |  |
| 7. | Очікувати завершення обчислення х в інших задачах |  |
| 8. | Копіювати , e3 = e, MK3 = MK, х3 = х, R3 = R | КД |
| 9. | Виконати обчислення = |  |
| 10. | Надіслати сигнал Т2 про завершення обчислень |  |

**Задача Т4**

|  |  |  |
| --- | --- | --- |
| **№** | **Дія** | **ТС/КД** |
| 1. | Ввід C, MO |  |
| 2. | Надіслати сигнал Т1,2,3,i про завершення вводу |  |
| 3. | Очікувати сигнал про завершення вводу в Т1..3 |  |
| 4. | Обчислити |  |
| 5. | Обчислити х = х+х4 | КД |
| 6. | Надіслати сигнал про завершення обчислення х задачам Т1,2,3,i |  |
| 7. | Очікувати завершення обчислення х в інших задачах |  |
| 8. | Копіювати , e4 = e, MK4= MK, x4 = x, R4 = R | КД |
| 9. | Виконати обчислення = |  |
| 10. | Надіслати сигнал Т2 про завершення обчислень |  |

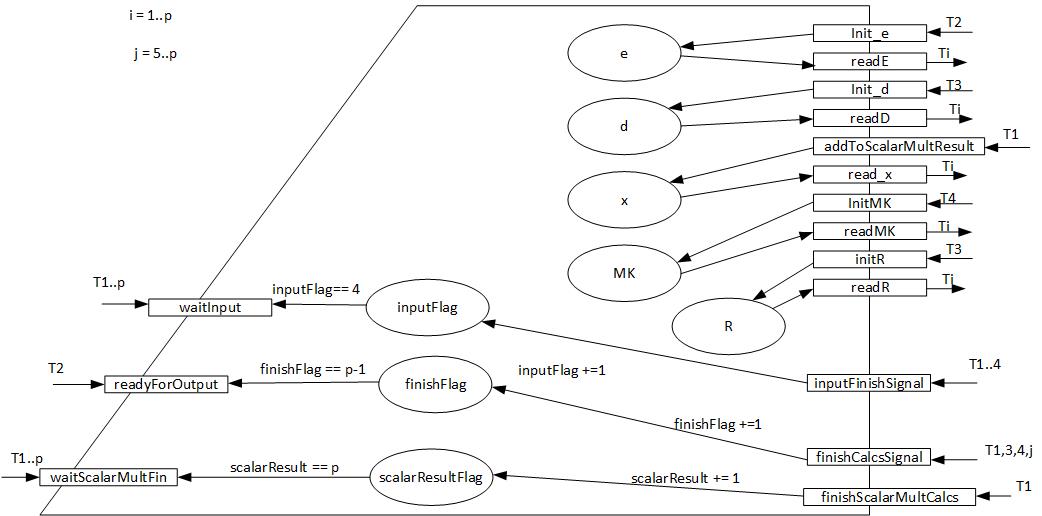
**Задача Тi**

|  |  |  |
| --- | --- | --- |
| **№** | **Дія** | **ТС/КД** |
| 1. | Очікувати сигнал про завершення вводу в Т1..4 |  |
| 2. | Обчислити |  |
| 3. | Обчислити х = х+хi | КД |
| 4. | Надіслати сигнал про завершення обчислення х задачам Т1,2,3,j |  |
| 5. | Очікувати завершення обчислення х в інших задачах |  |
| 6. | Копіювати , ei = e, MKi= MK, xi= x, Ri = R | КД |
| 7. | Виконати обчислення = |  |
| 8. | Надіслати сигнал Т2 про завершення обчислень |  |

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

На структурній схемі взаємодії задач уведено такі умовні позначення:

1. Init\_e, Init\_d, InitMK, InitR – процедури для запису спільних ресурсів до захищеного модулю;
2. addToScalarMultResult – процедура для додавання проміжного результату скалярного добутку до загального;
3. readE, readD, read\_х, readMK, readR - функції для зчитування спільних ресурсів з захищеного модулю;
4. inputFinishSignal – сигнал задачам про завершення вводу даних;
5. finishCalcsSignal – сигнал Т2 про завершення обчислень;
6. finishScalarMultCalcs – сигнал іншим задачам про завершення обчислення скалярного добутку;
7. waitInput – вхід для синхронізації вводу даних;
8. readyForOutput – вхід для синхронізації завершення обчислень та виводу даних
9. waitScalarMultFin – синхронізація результуючого обчислення скалярного добутку



**Етап 4. Лістинг коду**

GNAT 4.6.4

Copyright 1992-2010, Free Software Foundation, Inc.

Compiling: main.adb (source file time stamp: 2016-04-09 18:17:44)

1. with Ada.Text\_IO, Ada.Integer\_Text\_IO,Ada.Float\_Text\_IO, Ada.Synchronous\_Task\_Control, Ada.Real\_Time, MatrixOperations;

2. use Ada.Text\_IO, Ada.Integer\_Text\_IO,Ada.Float\_Text\_IO, Ada.Synchronous\_Task\_Control, Ada.Real\_Time;

3.

4.

5. ------------------------------------------------------------------

6. -- --

7. -- Програмування паралельних компьютерних систем --

8. -- Лаб 5. Ада. Захищені модулі --

9. -- --

10. -- Файл: main.abd --

11. -- Завдання: --

12. -- --

13. -- --

14. -- --

15. -- Автор: Кахерський Олег, група IП-31 --

16. -- Дата: 07.04.2016 --

17. -- --

18. ------------------------------------------------------------------

19.

20. procedure Main is

21. n: Integer; --розмірність матриць та векторів

22. p: Integer; --кількість задач

23. h: Integer; --кількість рядків матриць та векторів на одну задачу

24. procedure Start is

25. package MatrixOperationsN is new MatrixOperations(N);

26. use MatrixOperationsN;

27. MO : Matrix;

28. A, B, C, Z : Vector;

29.

30. task type ThreadTask(tID: Integer);

31. protected ProtectedModule is

32. entry waitInput; --бар'єр для синхронізації вводу даних

33. entry waitScalarMultFin; --очікування підсумовування скалярних добутків підвекторів

34. entry readyForOuptut; --очікування сигналу від T2-Tp задачею Т1

35.

36. function readE return Integer;

37. function readD return Integer;

38. function read\_a return Integer;

39. function readR return Vector;

40. function readMK return Matrix;

41.

42. procedure inputFinishSignal; --надсилання сигналу про завершення вводу даних

43. procedure finishCalcsSignal; --надсилання сигналу про завершення обчислень

44. procedure finishScalarMultCalcs; --надсилання сигналу про завершення обчислення скалярного добутку векторів

45. procedure addToScalarMultResult(b: Integer);

46. procedure init\_e(b: Integer);

47. procedure init\_d(b: Integer);

48. procedure initR(Buff: Vector);

49. procedure initMK(Buff: Matrix);

50. procedure outputLine(message: String);

51. private

52. inputFlag : Integer := 0;

53. finishFlag : Integer := 0;

54. scalarResultFlag : Integer := 0;

55.

56. e, x, d : Integer;

57. R : Vector;

58. MK : Matrix;

59. end ProtectedModule;

60.

61. protected body ProtectedModule is

62.

63. --розблокування задач при inputFlag = 4 (завершення вводу даних в T1..4)

64. entry waitInput when inputFlag = 4 is

65. begin

66. null;

67. end waitInput;

68.

69. entry waitScalarMultFin when scalarResultFlag = p is

70. begin

71. null;

72. end waitScalarMultFin;

73.

74. --розблокування Т2 при завершенні обчислень в задачах T1..Tp

75. entry readyForOuptut when finishFlag = p-1 is

76. begin

77. null;

78. end readyForOuptut;

79.

80. function readE return Integer is

81. begin

82. return e;

83. end readE;

84.

85. function readD return Integer is

86. begin

87. return d;

88. end readD;

89.

90. function read\_A return Integer is

91. begin

92. return x;

93. end read\_A;

94.

95. function readR return Vector is

96. begin

97. return R;

98. end readR;

99.

100. function readMK return Matrix is

101. begin

102. return MK;

103. end readMK;

104.

105. procedure inputFinishSignal is --надсилання сигналу про завершення вводу даних

106. begin

107. inputFlag := inputFlag + 1;

108. end inputFinishSignal;

109.

110. procedure finishCalcsSignal is --надсилання сигналу про завершення обчислень

111. begin

112. finishFlag := finishFlag + 1;

113. end finishCalcsSignal;

114.

115. procedure finishScalarMultCalcs is

116. begin

117. scalarResultFlag := scalarResultFlag + 1;

118. end finishScalarMultCalcs;

119.

120. procedure addToScalarMultResult(b: Integer) is

121. begin

122. x := x + b;

123. end addToScalarMultResult;

124.

125. procedure init\_e(b: Integer) is

126. begin

127. e := b;

128. end init\_e;

129.

130. procedure init\_d(b: Integer) is

131. begin

132. d := b;

133. end init\_d;

134.

135. procedure initR(Buff: Vector) is

136. begin

137. R := Buff;

138. end initR;

139.

140. procedure initMK(Buff: Matrix) is

141. begin

142. MK := Buff;

143. end initMK;

144.

145. procedure outputLine(message: String) is

146. begin

147. Put\_Line(message);

148. New\_line;

149. end outputLine;

150. end ProtectedModule;

151.

152. procedure calcMatrixEquation(ai,ei,di,startIndex,endIndex : Integer; Ri:Vector; MKi: Matrix) is

153. Buf1: Matrix;

154. Vect\_Buf: Vector;

155. begin

156. Matrix\_Matrix\_Multiply(MO,MKi,startIndex,endIndex,Buf1); --Buf1\_h = MOh+MK

157. Vector\_Matrix\_Multiply(Ri,Buf1,Vect\_Buf,startIndex,endIndex);

158. Vector\_Vector\_Add(Z,Vect\_Buf,A,ei,ai\*di,startIndex,endIndex);

159. end calcMatrixEquation;

160.

161. task body ThreadTask is

162. i: Integer := tID;

163. xi, ei, di, startIndex, endIndex: Integer;

164. MKi : Matrix;

165. Ri: Vector;

166. begin

167. ProtectedModule.outputLine("Task " & Integer'Image(i) & " started");

168. startIndex := h\*(i - 1) + 1;

169. if i = p then

170. endIndex := N;

171. else

172. endIndex := h\*i;

173. end if;

174.

175. if i = 1 then

176. ProtectedModule.outputLine("Task "& Integer'Image(i) & " is inputting data");

177.

178. ProtectedModule.outputLine("Task "& Integer'Image(i) & " inputted MK");

179. ProtectedModule.initMK(Input\_Matrix);

180.

181. --сигнал задачам о завершении ввода

182. ProtectedModule.outputLine("Task "& Integer'Image(i) & " finished input and send Signal to All");

183. ProtectedModule.inputFinishSignal;

184.

185. elsif i = 2 then

186. ProtectedModule.outputLine("Task "& Integer'Image(i) & " is initing data");

187. ProtectedModule.outputLine("Task "& Integer'Image(i) & " inited Z");

188. Z := Input\_Vector;

189.

190. ProtectedModule.outputLine("Task "& Integer'Image(i) & " inited e");

191. ProtectedModule.init\_e(Input\_Constant);

192.

193. --сигнал задачам о завершении ввода

194. ProtectedModule.outputLine("Task "& Integer'Image(i) & " finished input and send Signal to All");

195. ProtectedModule.inputFinishSignal;

196.

197. elsif i = 3 then

198. ProtectedModule.outputLine("Task "& Integer'Image(i) & " inited B");

199. B := Input\_Vector;

200. ProtectedModule.outputLine("Task "& Integer'Image(i) & " inited R");

201. ProtectedModule.initR(Input\_Vector);

202.

203. ProtectedModule.outputLine("Task "& Integer'Image(i) & " inited d");

204. ProtectedModule.init\_d(Input\_Constant);

205.

206. --сигнал задачам о завершении ввода

207. ProtectedModule.outputLine("Task "& Integer'Image(i) & " finished input and send Signal to All");

208. ProtectedModule.inputFinishSignal;

209.

210. elsif i = 4 then

211.

212. ProtectedModule.outputLine("Task "& Integer'Image(i) & " inited C");

213. C := Input\_Vector;

214.

215. ProtectedModule.outputLine("Task "& Integer'Image(i) & " inited MO");

216. MO := Input\_Matrix;

217.

218. --сигнал задачам о завершении ввода

219. ProtectedModule.outputLine("Task "& Integer'Image(i) & " finished input and send Signal to All");

220. ProtectedModule.inputFinishSignal;

221.

222. else

223. ProtectedModule.outputLine("Task "& Integer'Image(i) & " waits for inputting data in T1, Tp");

224. end if;

225.

226. --синхронізація по вводу

227. ProtectedModule.waitInput;

228.

229. --обчислення скалярного добутку

230. ProtectedModule.addToScalarMultResult(Scalar\_Multiply(B, C, startIndex, endIndex));

231.

232. --синхронізація по завершенню обчислення скалярного добутку

233. ProtectedModule.finishScalarMultCalcs;

234. ProtectedModule.waitScalarMultFin;

235.

236. --копіювання СР

237. ProtectedModule.outputLine("Task "& Integer'Image(i) & " is copying shared resources");

238. ei := ProtectedModule.readE;

239. di := ProtectedModule.readD;

240. Ri := ProtectedModule.readR;

241. xi := ProtectedModule.read\_a;

242. MKi := ProtectedModule.readMK;

243.

244. ProtectedModule.outputLine("Task "& Integer'Image(i) & " is calculating matrix equation");

245.

246. calcMatrixEquation(xi, ei, di, startIndex, endIndex, Ri, MKi);

247.

248. if i = 2 then

249. ProtectedModule.outputLine("Task "& Integer'Image(i) & " is waiting for signal from all(finishing calc matrix equation)");

250. ProtectedModule.readyForOuptut;

251. if n <= 8 then

252. Put\_Line("Result:");

253. New\_line;

254. Output\_Vector(A);

255. end if;

256. Get(H);

257. else

258. ProtectedModule.outputLine("Task "& Integer'Image(i) & " is sending signal to T2 (finishing calc matrix equation)");

259. ProtectedModule.finishCalcsSignal;

260. end if;

261. Put\_Line("Task " & Integer'Image(i) & " finished");

262. New\_line;

263. end ThreadTask;

264.

265. type ThreadTaskPointer is access ThreadTask;

266. type TaskArray is array (1 .. p) of ThreadTaskPointer;

267. tArray: TaskArray;

268. begin

269. for I in 1 .. p loop

270. tArray(I) := new ThreadTask(I);

271. end loop;

272. end Start;

273.

274. begin

275. Put("Program started");

276. New\_line;

277. Put("Input N");

278. New\_line;

279. Get(n);

280. Put("Input P");

281. New\_line;

282. Get(p);

283. h := n/p;

284. Start;

285. end Main;

285 lines: No errors

GNAT 4.6.4

Copyright 1992-2010, Free Software Foundation, Inc.

Compiling: matrixoperations.adb (source file time stamp: 2016-04-06 10:46:08)

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

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

3. ------------------------------------------------------------------

4. -- File: matrixOperations.adb --

5. -- --

6. -- Автор: Олег Кахерський, група ІП-31 --

7. -- Дата: 12.03.2016 --

8. ------------------------------------------------------------------

9. package body MatrixOperations is

10.

11. procedure Output\_Vector (A : Vector) is

12. begin

13. New\_Line;

14. for I in 1..N loop

15. Put(Item => A(i), Width => 5);

16. end loop;

17. end Output\_Vector;

18.

19. procedure Matrix\_Matrix\_Multiply

20. (Left : Matrix;

21. Right : Matrix;

22. Start\_Index: Integer;

23. End\_Index: Integer;

24. Result\_Matrix: out Matrix) is

25. begin

26. for i in Start\_Index..End\_Index loop

27. for J in 1..N loop

28. Result\_Matrix(I)(J) := 0;

29. for K in 1..N loop

30. Result\_Matrix(I)(J) := Result\_Matrix(I)(J) + Left(I)(K) \* Right(K)(J);

31. end loop;

32. end loop;

33. end loop;

34. end Matrix\_Matrix\_Multiply;

35.

36. function Scalar\_Multiply

37. (V1: Vector;

38. V2:Vector;

39. Start\_Index:Integer;

40. End\_Index:Integer) return Integer is

41. Result: Integer := 0;

42. begin

43. for I in Start\_Index..End\_Index loop

44. Result := Result + V1(I)\*V2(I);

45. end loop;

46. return Result;

47. end Scalar\_Multiply;

48.

49. procedure Vector\_Vector\_Add

50. (V1: Vector;

51. V2: Vector;

52. Result\_Vector: out Vector;

53. Const1: Integer;

54. Const2: Integer;

55. Start\_Index: Integer;

56. End\_Index: Integer) is

57. begin

58. for I in Start\_Index..End\_Index loop

59. Result\_Vector(I) := Const1\*V1(I) + Const2\*V2(I);

60. end loop;

61. end Vector\_Vector\_Add;

62.

63. procedure Vector\_Matrix\_Multiply

64. (V: Vector;

65. MT:Matrix;

66. Result: out Vector;

67. Start\_Index:Integer;

68. End\_Index:Integer) is

69. begin

70. for I in Start\_Index..End\_Index loop

71. Result(I) := 0;

72. for J in 1..N loop

73. Result(I) := Result(I) + V(J) \* MT(I)(J);

74. end loop;

75. end loop;

76. end Vector\_Matrix\_Multiply;

77.

78. function Input\_Vector return Vector is

79. V: Vector;

80. begin

81. for I in 1..N loop

82. V(I) := 1;

83. end loop;

84. return V;

85. end Input\_Vector;

86.

87. function Input\_Constant return Integer is

88. begin

89. return 1;

90. end Input\_Constant;

91.

92. function Input\_Matrix return Matrix is

93. Result\_Matrix: Matrix;

94. begin

95. for I in 1..N loop

96. for J in 1..N loop

97. Result\_Matrix(I)(J) := 1;

98. end loop;

99. end loop;

100. return Result\_Matrix;

101. end Input\_Matrix;

102.

103. end MatrixOperations;

Compiling: matrixoperations.ads (source file time stamp: 2016-04-06 10:45:26)

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

2. -- File: matrixOperations.ads --

3. -- --

4. -- Автор: Олег Кахерський, група ІП-31 --

5. -- Дата: 12.03.2016 --

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

7. generic

8. N: in Integer; --розмірність масивів та матриць

9. package MatrixOperations is

10. type Vector is private;

11. type Matrix is private;

12.

13. --Вивід вектора на екран

14. procedure Output\_Vector (A : Vector);

15.

16. --Множення матриці на матрицю(діапазон рядків StartIndex..EndIndex),

17. --результат множення множиться на константу

18. procedure Matrix\_Matrix\_Multiply

19. (Left : Matrix;

20. Right : Matrix;

21. Start\_Index: Integer;

22. End\_Index: Integer;

23. Result\_Matrix: out Matrix);

24.

25. --скалярний добуток векторів в діапазоні [StartIndex, EndIndex)

26. function Scalar\_Multiply

27. (V1: Vector;

28. V2:Vector;

29. Start\_Index:Integer;

30. End\_Index:Integer) return Integer;

31.

32. procedure Vector\_Matrix\_Multiply

33. (V: Vector;

34. MT:Matrix;

35. Result: out Vector;

36. Start\_Index:Integer;

37. End\_Index:Integer);

38.

39. procedure Vector\_Vector\_Add

40. (V1: Vector;

41. V2: Vector;

42. Result\_Vector: out Vector;

43. Const1: Integer;

44. Const2: Integer;

45. Start\_Index: Integer;

46. End\_Index: Integer);

47.

48. --Ввід вектора

49. function Input\_Vector return Vector;

50.

51. --Ввід константи

52. function Input\_Constant return Integer;

53.

54. --Ввід матриці

55. function Input\_Matrix return Matrix;

56.

57. private

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

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

60. end MatrixOperations; 103 lines: No errors