НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

КАФЕДРА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ

Лабораторна робота №2

з дісципліни **«**Паралельні та розподілені обчислення**»**

Виконав:

студент 3 курсу

ФІОТ гр. ІО-21

Кузьменко Володимир

Перевірив:

Корочкін О. В.

Київ – 2014 р.

Цель работы: изучение средств языка Ада для работы с процессами.

Выполнение работы: Разработать программу, содержащую п а р а л л е л ь н ы е задачи,

каждая из которых реализует функцию F1, F2, F3 из лабораторной работы номер 1. Исполь-

зовать пакет Data из лабораторной работы 1 !

Задачи независимы, общих данных не имеют!

При создании задач необходимо:

- указать имя задачи

- задать приоритет задачи

- задать размер стека для задачи

- выбрать и задать номер процессора (ядра) для выполнения каждой задачи.

Первый и последний операторы тела задачи выводят на экран информацию о старте и

завершении соответствующей задачи (“Task T1 started”, “ Task T2 finished”).

В теле задачи использовать оператор задержки delay, поставив его перед выпол-

нением функции F1, F2, F3.

Исследовать при выполнении программы:

- влияние приоритетов задач на очередность запуска задач (при использовании одно-

го или двух ядер).

- влияние оператора задержки delay на порядок выполнения задач. 5

- загрузку параллельной компьютерной системы (ПКС) ( ядер процессора) при из-

менении их количества. Изменение количества ядер задается с помощью Менеджера (Дис-

петчера) задач ОС Windows.

Лістинг програми:

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

2. -- --

3. -- Parallel and distributed computing --

4. -- Lab 2. Processes in language Ada. Tasks --

5. -- --

6. -- --

7. -- F1: 1.24 E = A + C \* (MA \* MZ) + B --

8. -- F2: 2.14 MC = SORT(MA + MB \* MM) --

9. -- F3: 3.18 p = MAX(SORT(MS) + MA\* MM) --

10. -- --

11. -- Author Kuzmenko Vladimir Z. --

12. -- Group IO-21 --

13. -- Date 17.09.2014 --

14. -- --

15. ------------------------------------------------------------------

16.

17.

18. with Ada.Text\_IO, Ada.Integer\_Text\_IO,System.Multiprocessors;

19. use Ada.Text\_IO, Ada.Integer\_Text\_IO,System.Multiprocessors;

20. with Data;

21.

22. procedure Lab2 is

23.

24. CPU\_1 : CPU\_Range:=0;

25. CPU\_2 : CPU\_Range:=1;

26.

27.

28. package Data3 is new Data(3);

29. use Data3;

30.

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

32. --Tasks cpecifications --

33. ---------------------------

34.

35. --task for calculation F1

36. task T1 is

37. pragma Task\_Name ("Task 1");

38. pragma Priority(4);

39. pragma Storage\_Size(1000);

40. pragma CPU (CPU\_1);

41. end T1;

42.

43. --task for calculation F2

44. task T2 is

45. pragma Task\_Name ("Task 2");

46. pragma Priority (5);

47. pragma Storage\_Size(2000);

48. pragma CPU (CPU\_2);

49. end T2;

50.

51. --task for calculation F3

52. task T3 is

53. pragma Task\_Name ("Task 3");

54. pragma Priority (6);

55. pragma Storage\_Size(3000);

56. pragma CPU (CPU\_1);

57. end T3;

58.

59. ---------------------------

60. --Task bodies --

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

62.

63. task body T1 is

64. A,B,C,E:Vector;

65. MA,MZ:Matrix;

66. begin

67. Put\_Line("Task T1 started");

68. --Input Data

69. Put\_Line ("Input vector A");

70. Vector\_Input (A);

71. Vector\_Output (A);

72. New\_Line;

73.

74. Put\_Line ("Input vector B");

75. Vector\_Input (B);

76. Vector\_Output (B);

77. New\_Line;

78.

79. Put\_Line ("Input vector C");

80. Vector\_Input (C);

81. Vector\_Output (C);

82. New\_Line;

83.

84. Put\_Line ("Input matrix MA");

85. Matrix\_Input (MA);

86. Matrix\_Output (MA);

87. New\_Line;

88.

89. Put\_Line ("Input matrix MB");

90. Matrix\_Input (MZ);

91. Matrix\_Output (MZ);

92. New\_Line;

93.

94. delay(5.7);

95.

96. --calculation

97.

98. Func1 (A, B, C, MA, MZ, E);

99.

100. --output data

101. Put ("Function 1, vector E");

102. Vector\_Output (E);

103. New\_Line;

104.

105. Put\_Line("Task T1 finished");

106. end T1;

107.

108. task body T2 is

109. MA,MB,MC,MM:Matrix;

110. begin

111. Put\_Line("Task T2 started");

112. --input data

113. Put\_Line ("Input matrix MA");

114. Matrix\_Input (MA);

115. Matrix\_Output (MA);

116. New\_Line;

117.

118. Put\_Line ("Input matrix MB");

119. Matrix\_Input (MB);

120. Matrix\_Output (MB);

121. New\_Line;

122.

123. Put\_Line ("Input matrix MM");

124. Matrix\_Input (MM);

125. Matrix\_Output (MM);

126. New\_Line;

127.

128. delay(6.7);

129. --calculation

130. Func2 (MA, MB, MM, MC);

131.

132. --data output

133. Put ("Function 2, matrix MC");

134. Matrix\_Output (MC);

135. New\_Line;

136.

137.

138. Put\_Line("Task T2 finished");

139. end T2;

140.

141. task body T3 is

142. MS,MA,MM:Matrix;

143. p:Integer;

144. begin

145. Put\_Line("Task T3 started");

146.

147. --input data

148. Put\_Line ("Input matrix MA");

149. Matrix\_Input (MA);

150. Matrix\_Output (MA);

151. New\_Line;

152.

153. Put\_Line ("Input matrix MM");

154. Matrix\_Input (MM);

155. Matrix\_Output (MM);

156. New\_Line;

157.

158. Put\_Line ("Input matrix MS");

159. Matrix\_Input (MS);

160. Matrix\_Output (MS);

161. New\_Line;

162.

163. delay(7.7);

164. --calculation

165. Func3 (MA, MM, MS, p);

166.

167. --output data

168. Put ("Function 3, value p");

169. Put (p);

170. New\_Line;

171.

172. Put\_Line("Task T3 finished");

173. end T3;

174. begin

175. null;

176. end Lab2;

2. ------------------------

3. -- File data.adb

4. -- group IO-21

5. -- Kuzmenko Vladimir Z.

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

7.

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

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

10. package body Data is

11.

12. --input vector

13. procedure Vector\_Input (A:out Vector) is

14. begin

15. for i in 1..N loop

16. Get(A(i));

17. end loop;

18. end Vector\_Input;

19.

20. --output vector

21. procedure Vector\_Output (A:in Vector) is

22. begin

23. for i in 1..N loop

24. Put(A(i));

25. end loop;

26. end Vector\_Output;

27.

28. -- input matrix

29. procedure Matrix\_Input (MA:out Matrix) is

30. begin

31. for i in 1..N loop

32. for j in 1..N loop

33. Get(MA(i)(j));

34. end loop;

35. end loop;

36. end Matrix\_Input;

37.

38. --output matrix

39. procedure Matrix\_Output (MA:in Matrix) is

40. begin

41. for i in 1..n loop

42. for j in 1..n loop

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

44. Put ("|");

45. end loop;

46. Put\_Line("");

47. end loop;

48. end Matrix\_Output;

49.

50. --multing matrix

51. function MultMatr (ma,mz:Matrix) return Matrix is

52. s:integer;

53. MT:Matrix;

54. begin

55. for i in 1..N loop

56. for j in 1..N loop

57. s:=0;

58. for k in 1..N loop

59. s:=s+ma(i)(k)\*mz(k)(j);

60. MT(i)(j):=s;

61. end loop;

62. end loop;

63. end loop;

64. return MT;

65. end MultMatr;

66.

67. --multing matrix on vector

68. function multMatrixOnVector(MA:Matrix; A:Vector) return Vector is

69. s:Integer;

70. R:Vector;

71. begin

72. for i in 1..N loop

73. s:=0;

74. for j in 1..N loop

75. s:=s+MA(i)(j)\*A(j);

76. end loop;

77. R(i):=s;

78. end loop;

79. return R;

80. end multMatrixOnVector;

81.

82. --addition vectors

83. function AddVec(B,C:Vector) return Vector is

84. K:Vector;

85. begin

86. for i in 1..n loop

87. K(i):=B(i)+C(i);

88. end loop;

89. return K;

90. end AddVec;

91.

92. --addition matrix

93. function AddMatr (ma,mb:Matrix) return Matrix is

94. m:Matrix;

95. begin

96. for i in 1..N loop

97. for j in 1..N loop

98. m(i)(j):=ma(i)(j)+mb(i)(j);

99. end loop;

100. end loop;

101. return m;

102. end AddMatr;

103.

104. --sorting vector

105. function SortVector(A:Vector) return Vector is

106. M:Vector:=A;

107. buf:Integer;

108. k:Integer;

109. begin

110. for i in 1..(n-1) loop

111. k:=i;

112. for j in (i+1)..n loop

113. if M(k)>M(j) then

114. k:=j;

115. end if;

116. end loop;

117. buf:=M(k);

118. M(k):=M(i);

119. M(i):=buf;

120. end loop;

121. return M;

122. end SortVector;

123.

124. -- sjrting matrix

125. function SortMatr(MA:Matrix) return Matrix is

126. MT:Matrix;

127. begin

128. for i in 1..n loop

129. MT(i):=sortVector(MA(i));

130. end loop;

131. return MT;

132. end SortMatr;

133.

134. --finding max element in matrix

135. function MaxElMatr (MD:Matrix) return Integer is

136. m:Integer:=MD(1)(1);

137. begin

138. for i in 1..n loop

139. for j in 1..n loop

140. if m<MD(i)(j) then

141. m:=MD(i)(j);

142. end if;

143. end loop;

144. end loop;

145. return m;

146. end MaxElMatr;

147.

148. --F1: 1.24 E = A + C \* (MA \* MZ) + B

149. procedure Func1(A,B,C:in Vector;MA,MZ: in Matrix; E:out Vector) is

150. begin

151. E:=AddVec(AddVec(multMatrixOnVector(MultMatr(MA,MZ),C),A),B);

152. end Func1;

153.

154. --F2: 2.14 MC = SORT(MA + MB \* MM)

155. procedure Func2(MA,MB,MM:in Matrix; MC:out Matrix) is

156. begin

157. MC:= SortMatr(AddMatr(MultMatr(MB,MM),MA));

158. end Func2;

159.

160. --F3: 3.18 p = MAX(SORT(MS) + MA\* MM)

161. procedure Func3(MA,MM,MS: in Matrix; p: out Integer) is

162. begin

163. p:= MaxElMatr(AddMatr(SortMatr(MS),MultMatr(MA,MM)));

164. end Func3;

165.

166. end Data;

167.

Compiling: data.ads (source file time stamp: 2014-09-09 20:54:28)

1.

2. ------------------------

3. -- File data.ads

4. -- group IO-21

5. -- Kuzmenko Vladimir Z.

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

7.

8. generic

9. N : Integer;

10. package data is

11.

12. type Vector is private;

13. type Matrix is private;

14.

15. -- input vector

16. procedure Vector\_Input (A:out Vector);

17. -- output vector

18. procedure Vector\_Output (A:in Vector);

19. -- input matrix

20. procedure Matrix\_Input (MA:out Matrix);

21. -- output matrix

22. procedure Matrix\_Output (MA:in Matrix);

23.

24. --F1: 1.24 E = A + C \* (MA \* MZ) + B

25. procedure Func1(A,B,C:in Vector;MA,MZ: in Matrix; E:out Vector);

26.

27. --F2: 2.14 MC = SORT(MA + MB \* MM)

28. procedure Func2(MA,MB,MM:in Matrix; MC:out Matrix);

29.

30. --F3: 3.18 p = MAX(SORT(MS) + MA\* MM)

31. procedure Func3(MA,MM,MS: in Matrix; p: out Integer);

32.

33. private

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

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

36.

37. end Data;