

Міністерство освіти і науки України

Національний технічний університет України

«Київський політехнічний інститут ім. Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

**ЛАБОРАТОРНА РОБОТА №4**

**З ДИСЦИПЛІНИ “** **ПАРАЛЕЛЬНІ ТА РОЗПОДІЛЕНІ ОБЧИСЛЕННЯ”**

**НА ТЕМУ: “OpenMP. Бар’єри, критичні секції”**

**Виконав:**

Студент ІІІ курсу ФІОТ

групи ІО-82

Шендріков Євгеній

Номер у списку - 24

**Перевірив:**

Доцент Корочкін О. В.

м. Київ – 2021 р.

**Технічне завдання**

1. Розробити паралельний алгоритм рішення математичної задачі

з використанням бібліотеки OpenMp на мові C++;

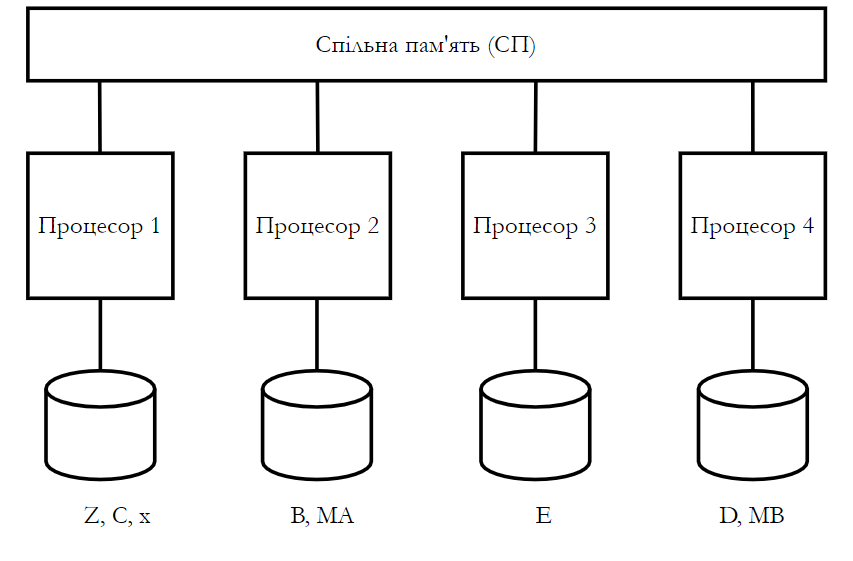
1. Виявити спільні ресурси;
2. Описати алгоритм кожного потоку (Т1 – Тр) з визначенням критичних ділянок (КД) та бар’єрів;
3. Розробити структурну схему взаємодії задач, де застосувати всі вказані засоби взаємодії процесів;
4. Розробити програму (обов’язкові “шапка”, коментарі);
5. Виконати налагодження програми;
6. Отримати правильні результати обчислень;
7. За допомогою Диспетчеру задач Windows проконтролювати завантаження ядер процесору.

Задача: ;

Мова програмування: С++;

Засоби організації взаємодії: бар’єри, замки, критичні секції OpenMP;

Структурна схема ПКС



**Виконання роботи**

**Етап 1. Побудова паралельного алгоритму**

1. bi = BH \* CH, i =
2. b = b + bi, i =
3. ZH = b \* DH + E \* (MA \* MBH) \* x

Спільний ресурс: b, x, E, MA

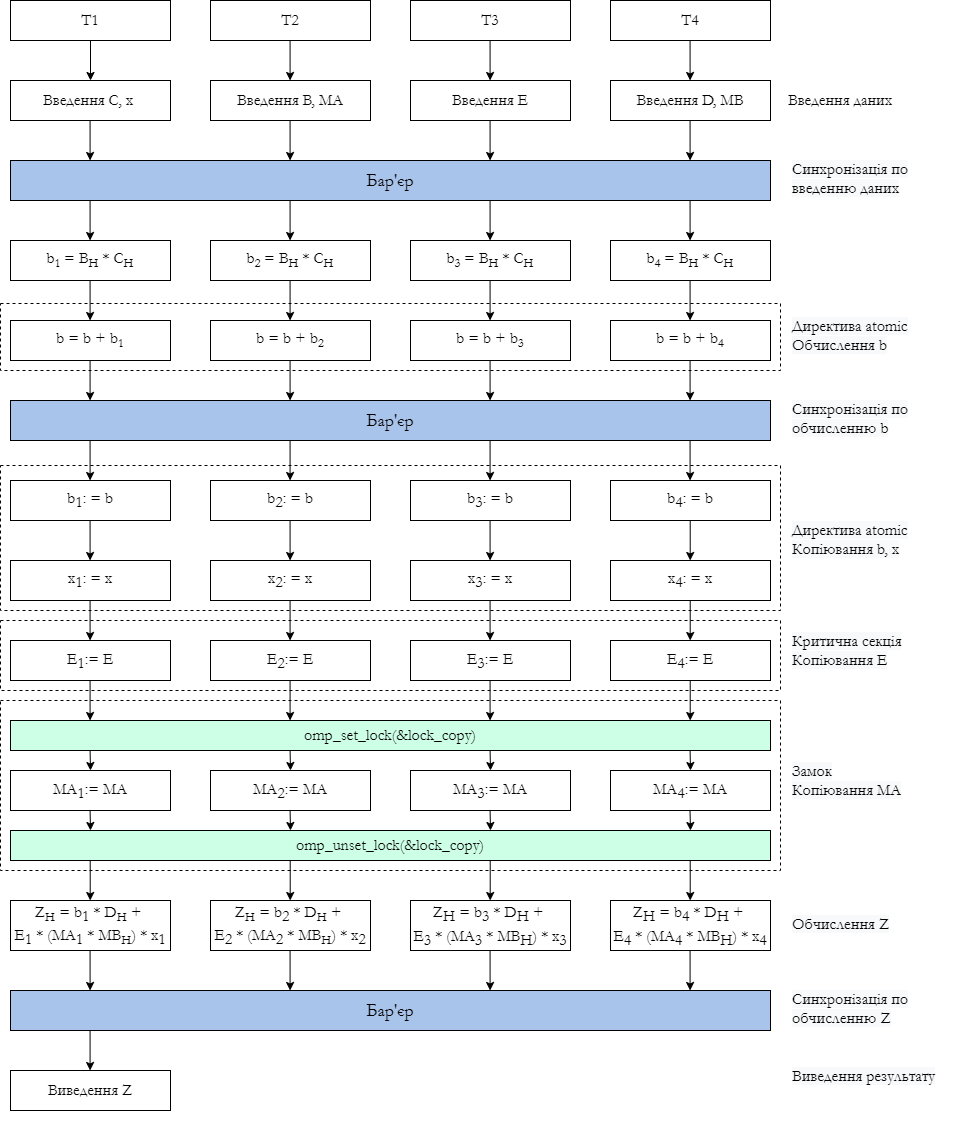
**Етап 2. Розроблення алгоритмів роботи кожного процесу**

|  |  |  |
| --- | --- | --- |
| Задача Т1 | | КД та Бар’єри |
| 1 | **Введення** C, x |  |
| 2 | Бар’єр. Синхронізація вводу |  |
| 3 | **Обчислення 1:** b1: = BH \* CH |  |
| 4 | **Обчислення 2:** b: = b + b1 | КД |
| 5 | Бар’єр. Синхронізація обчислення 2 |  |
| 6 | **Копіювання** b1: = b | КД |
| 7 | **Копіювання** x1: = x | КД |
| 8 | **Копіювання** E1:= E | КД |
| 9 | **Копіювання** MA1:= MA | КД |
| 10 | **Обчислення 3**: ZH = b1 \* DH + E1 \* (MA1 \* MBH) \* x1 |  |
| 11 | Бар’єр. Синхронізація обчислення 3 |  |
| 12 | **Виведення** Z |  |
|  | | |
| Задача Т2 | | КД та Бар’єри |
| 1 | **Введення** B, MA |  |
| 2 | Бар’єр. Синхронізація вводу |  |
| 3 | **Обчислення 1:** b2: = BH \* CH |  |
| 4 | **Обчислення 2:** b: = b + b2 | КД |
| 5 | Бар’єр. Синхронізація обчислення 2 |  |
| 6 | **Копіювання** b2: = b | КД |
| 7 | **Копіювання** x2: = x | КД |
| 8 | **Копіювання** E2:= E | КД |
| 9 | **Копіювання** MA2:= MA | КД |
| 10 | **Обчислення 3**: ZH = b2 \* DH + E2 \* (MA2 \* MBH) \* x2 |  |
|  | | |
| Задача Т3 | | КД та Бар’єри |
| 1 | **Введення** E |  |
| 2 | Бар’єр. Синхронізація вводу |  |
| 3 | **Обчислення 1:** b3: = BH \* CH |  |
| 4 | **Обчислення 2:** b: = b + b3 | КД |
| 5 | Бар’єр. Синхронізація обчислення 2 |  |
| 6 | **Копіювання** b3: = b | КД |
| 7 | **Копіювання** x3: = x | КД |
| 8 | **Копіювання** E3:= E | КД |
| 9 | **Копіювання** MA3:= MA | КД |
| 10 | **Обчислення 3**: ZH = b3 \* DH + E3 \* (MA3 \* MBH) \* x3 |  |
|  | | |
| Задача Т4 | | КД та Бар’єри |
| 1 | **Введення** D, MB |  |
| 2 | Бар’єр. Синхронізація вводу |  |
| 3 | **Обчислення 1:** b4: = BH \* CH |  |
| 4 | **Обчислення 2:** b: = b + b4 | КД |
| 5 | Бар’єр. Синхронізація обчислення 2 |  |
| 6 | **Копіювання** b4: = b | КД |
| 7 | **Копіювання** x4: = x | КД |
| 8 | **Копіювання** E4:= E | КД |
| 9 | **Копіювання** MA4:= MA | КД |
| 10 | **Обчислення 3**: ZH = b4 \* DH + E4 \* (MA4 \* MBH) \* x4 |  |

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

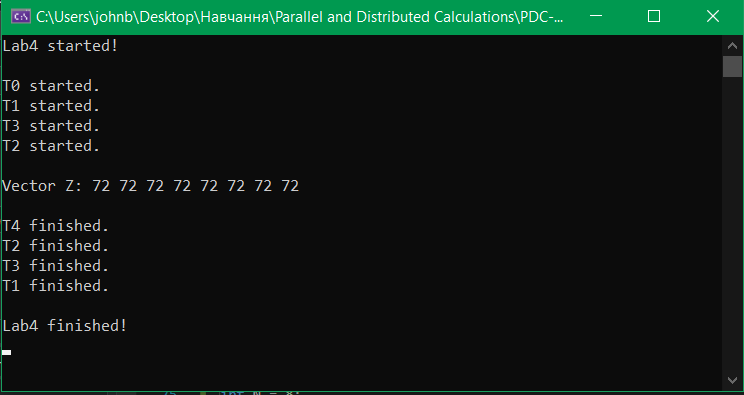
На структурній схемі взаємодії задач зображено такі засоби організації взаємодії потоків, як:

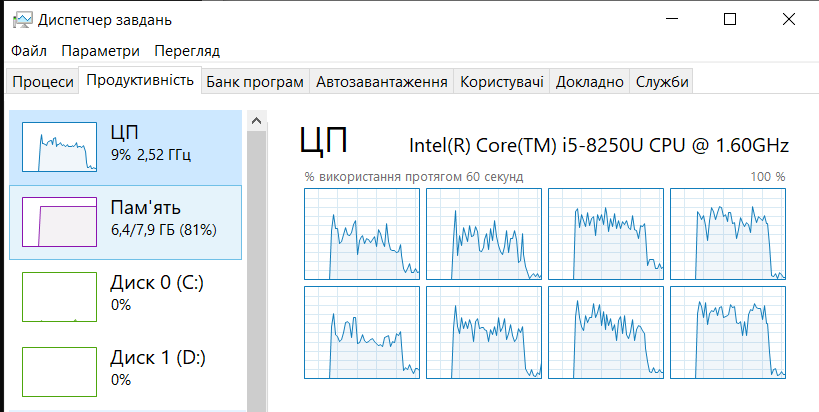
* **бар’єри** для синхронізації потоків;
* директива **atomic** для обчислення b;
* директива **atomic** для забезпечення послідовного доступу до спільних ресурсів b та x;
* директива **critical** для керування доступом до спільного ресурсу E;
* **замок** *lock\_copy* для керування доступом до спільного ресурсу MA.



**Етап 4. Розробка програми**

**Результат роботи**



****

**Висновки**

1. На основі бар’єрів, замків та критичних секцій бібліотеки OpenMP на мові C++ було розроблено програму та паралельний алгоритм для рішення математичної задачі заданої за варіантом.
2. Було описано алгоритм кожного потоку (Т1 – Т4) з визначенням критичних ділянок (КД) та бар’єрів;
3. Розроблено структурну схему взаємодії задач, де було застосовано вказані в завданні засоби взаємодії процесів. Засобом організації взаємодії слугували бар’єри, директиви atomic, замки та критичні секції.
4. Було написано програму згідно з завданням та перевірено її працездатність, а також проконтрольовано завантаження ядер процесору за допомогою Диспетчеру задач. Програма забезпечує 80% завантаженості.

**Лістинг коду**

**Lab4.cpp**

1. /\*-----------------------------------------------------

2. | Labwork #4 |

3. | OpenMP. Barriers, critical sections |

4. ------------------------------------------------------

5. | Author | Jack (Yevhenii) Shendrikov |

6. | Group | IO-82 |

7. | Variant | #25 |

8. | Date | 23.03.2021 |

9. ------------------------------------------------------

10. | Function | Z = (B\*C)\*D + E\*(MA\*MB)\*x |

11. ------------------------------------------------------

12. \*/

13.

14. #include "omp.h"

15. #include "Data.h"

16. #include <iostream>

17. #include <windows.h>

18.

19. using namespace std;

20.

24. const int P = 4;

25. int N = 8;

26.

27. int main()

28. {

29. int b = 0;

30. int x = 0;

31.

32. vector Z = new int[N];

33. vector B = new int[N];

34. vector C = new int[N];

35. vector D = new int[N];

36. vector E = new int[N];

37.

38. matrix MA;

39. matrix MB;

40.

41. const int H = N / P;

42.

43. void cs();

44. omp\_lock\_t lock\_copy;

45. omp\_init\_lock(&lock\_copy);

46.

47. cout << "Lab4 started!\n\n";

48.

49. omp\_set\_num\_threads(P);

50.

51. #pragma omp parallel

52. {

53. int tid = omp\_get\_thread\_num();

54. cout << "T" << tid + 1 << " started.\n";

55.

56. //---------------------------- Ввід Даних ---------------------------

57. switch (tid)

58. {

59. case 1:

60. C = inVector(1);

61. x = 1;

62. break;

63. case 2:

64. B = inVector(1);

65. MA = inMatrix(1);

66. break;

67. case 3:

68. E = inVector(1);

69. break;

70. case 4:

71. D = inVector(1);

72. MB = inMatrix(1);

73. break;

74. }

75.

76. //----------------- Синхронізація по введенню даних -----------------

77. #pragma omp barrier

78. int bi = 0;

79. for (int i = tid \* H; i < (tid + 1) \* H; i++)

80. bi += B[i] \* C[i];

81.

82. // Обчислення b

83. #pragma omp\_atomic

84. {

85. b += bi;

86. }

87.

88. //------------------ Синхронізація по обчисленню b ------------------

89. #pragma omp barrier

90. int xi = 0;

91. vector Ei = new int[N];

92. matrix MAi = new vector[N];

93.

94. // Директива atomic - копіювання b

95. #pragma omp\_atomic

96. {

97. bi = b;

98. }

99.

100. // Директива atomic - копіювання x

101. #pragma omp\_atomic

102. {

103. xi = x;

104. }

105.

106. // Критична секція - копіювання E

107. #pragma omp critical(cs)

108. {

109. Ei = copyVector(E);

110. }

111.

112. // Замок - копіювання MA

113. omp\_set\_lock(&lock\_copy);

114. MAi = copyMatrix(MA);

115. omp\_unset\_lock(&lock\_copy);

116.

117.

118. // Обчислення Z

119. int buf;

120. for (int i = tid \* H; i < (tid + 1) \* H; i++) {

121. buf = 0;

122. Z[i] = 0;

123.

124. Z[i] += bi \* D[i];

125.

126. for (int j = 0; j < N; j++) {

127. buf = 0;

128. for (int k = 0; k < N; k++)

129. buf += MAi[i][k] \* MB[k][j];

130. Z[i] += Ei[j] \* buf \* xi;

131. }

132. }

133.

134. //------------------ Синхронізація по обчисленню Z ------------------

135. #pragma omp barrier

136. if (tid == 0 && N < 15) {

137. outVector(Z, 'Z');

138. }

139. Sleep(100);

140. cout << "T" << tid + 1 << " finished." << endl;

141. }

142.

143. cout << "\nLab4 finished!\n";

144. getchar();

145. return 0;

146. }

**Data.cpp**

1. #include <windows.h>

2. #include <iostream>

3. #include "Data.h"

4.

5. using namespace std;

6.

7. // ------------------- Fill Matrix/Vector With Specific Number -------------------

8. vector inVector(int value) {

9. vector result = new int[N];

10. for (int i = 0; i < N; i++)

11. result[i] = value;

12.

13. return result;

14. }

15.

16. matrix inMatrix(int value) {

17. matrix result = new vector[N];

18. for (int i = 0; i < N; i++)

19. result[i] = new int[N];

20.

21. for (int i = 0; i < N; i++)

22. for (int j = 0; j < N; j++)

23. result[i][j] = value;

24.

25. return result;

26. }

27.

28.

29. // ------------- Print Vector Into Console --------------

30. void outVector(vector vec, char name) {

31. cout << "\nVector " << name << ": ";

32. for (int i = 0; i < N; i++)

33. {

34. cout << vec[i] << " ";

35. }

36. cout << "\n" << endl;

37. }

38.

39.

40. // ------------- Copy Vector, Matrix --------------

41. vector copyVector(vector vec) {

42. vector result = new int[N];

43. for (int i = 0; i < N; i++)

44. {

45. result[i] = vec[i];

46. }

47. return result;

48. }

49.

50. matrix copyMatrix(matrix matr) {

51. matrix result = new vector[N];

52. for (int i = 0; i < N; i++)

53. result[i] = new int[N];

54.

55. for (int i = 0; i < N; i++)

56. for (int j = 0; j < N; j++)

57. result[i][j] = matr[i][j];

58.

59. return result;

60. }

**Data.h**

1. typedef int\* vector;

2. typedef int\*\* matrix;

3.

4. extern int N;

5.

6. vector inVector(int);

7. matrix inMatrix(int);

8.

9. void outVector(vector, char);

10.

11. vector copyVector(vector);

12. matrix copyMatrix(matrix);