МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Вятский государственный университет»

Факультет автоматики и вычислительной техники

Кафедра электронных вычислительных машин

Параллельная машина с бесконечными регистрами

Отчет по лабораторной работе №4 по дисциплине

«Математическая логика и теория алгоритмов»

Вариант 88.

Выполнил студент группы ИВТб-1302-04-00 \_\_\_\_\_\_\_\_\_\_\_\_\_\_/Крючков И.С

Проверил преподаватель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/Крутиков А.К.

Киров 2021

1. **Цель работы:**

# изучить параллельную машину с бесконечными регистрами и освоить способы разработки наборов программ для нее.

# Задание:

* 1. В соответствии с полученным вариантом, разработать программу для классической машины с бесконечными регистрами, решающую поставленную задачу.
  2. Провести анализ заданной функции и выявить возможные способы декомпозиции программы из пункта 1 на несколько независимых программ.
  3. Разработать набор программ для параллельной машины с бесконечными регистрами, решающий поставленную задачу.
  4. Экспериментально подтвердить корректность разработанного набора программ.  
       
     Задание варианта:

Вычислить значение функции .

1. **Словесный алгоритм решения задачи:**Для возможности вычисления функция sinx была представлена в виде степенного ряда - ряда Тейлора. Таким образом,вычисляемая функция была представлена в виде равной функции

Алгоритм вычисления:

- Ввод числа X в регистр «0».

- Повторное выполнение пунктов ниже, до тех пор, пока значение k меньше 5

- Запуск параллельной программы mt1, для вычисления 2k+1

* Вычисление 2k+1
* Запуск параллельной программы st1, для вычисления
* Запуск параллельной программы fact, для вычисления

- Запуск параллельной программы div, для вычисления

* После завершения выполнения функций st1 и fact вычисление значения

- После завершения выполнения div, запуск параллельной программы sum, для вычисления сумм положительных и сумм модуля отрицательных значений функции.

- После завершения выполнения sum, запуск параллельной программы fin, для вычисления разности сумм положительных и сумм модуля отрицательных значений функции.

- Запись ответа в нулевой регистр  
  
 Начальное состояние регистров, необходимых для корректной работы программы:

|  |  |
| --- | --- |
| № РГ | Знач. |
| 1 | 2 |
| 20 | 100 |
| 99 | 0 |
| 30 | 5 |

1. **Листинг кода:**  1) unit Main // Объявление модуля. unit - ключевое слово, Main - название.
2. entry main // Объявление точки входа.
3. {
4. Z(5)
5. Z(25)
6. Z(26)
7. Z(27)
8. Z(29)
9. I(0) // Чтение числа в регистр 0
10. J(0, 99, 20) // Если ввели 0, на выход
11. J(5, 30, 19) // Если значение k и верхний предел суммы равны
12. Z(102)
13. Z(103)
14. Z(104)
15. Z(105)
16. s(mt1, 100) // Запуск программы test1 в параллельном процессе и сопоставление с ней регистра-индикатора с номером 100
17. s(div, 104)
18. J(99, 104, 15) // Если программа деления не закончилась, никуда не идем
19. s(sum, 105)
20. J(99, 105, 17) // Если программа суммирования не закончилась, никуда не идем
21. J(1, 1, 8)
22. s(fin, 106)
23. }
24. // Умножение 2\*k+1
25. program mt1
26. {
27. Z(2)
28. Z(3)
29. Z(4)
30. J(1, 3, 12) // Если промежуточное значение множителя равно самому множителю
31. J(5, 2, 9) // Если промежуточное значение числа и значение x равны
32. S(2)
33. S(4)
34. J(1, 1, 5)
35. S(3) // Увеличение промежуточного значения множителя
36. Z(2) // Очистка промежуточного значения Числа
37. J(1, 1, 4)
38. S(4)
39. s(st1, 102)
40. s(fact, 103)
41. }
42. // Возведение в степень 100\*x^[2k+1]
43. program st1
44. {
45. Z(8)
46. S(8)
47. J(4, 8, 20) // Если степень равна 1
48. T(0, 9)
49. J(4, 8, 21) // Если промежуточное значение степени равно исходной степени
50. Z(6)
51. Z(7)
52. Z(10)
53. J(0, 6, 17) // Если промежуточное значение множителя равно самому множителю
54. J(7, 9, 14) // Если промежуточное значение числа равно множимому числу
55. S(7)
56. S(10)
57. J(1, 1, 10)
58. S(6) // Увеличение промежуточного значения множителя
59. Z(7)
60. J(1, 1, 9)
61. S(8) // Увеличение промежуточного значения степени
62. T(10, 9)
63. J(1, 1, 5)
64. T(0, 10)
65. // умножение на 100
66. Z(19)
67. S(19)
68. T(10, 21)
69. J(19, 20, 32) // Если вспомогательное значение для умножения на 100 равно 100, то завершаем работу
70. Z(18)
71. J(18, 21, 30)
72. S(10)
73. S(18)
74. J(1, 1, 26)
75. S(19)
76. J(1, 1, 24)
77. }
78. //Факториал [2k+1]!
79. program fact
80. {
81. T(4, 13)
82. Z(17)
83. S(17) //{1}
84. J(4, 17, 30) // Если значение Рг4 равно 1, увеличить 16 на 1 (можно вывести в результат 1!!!)
85. S(15)
86. S(17)
87. J(4, 17, 9)
88. J(1, 1, 5) // Если Рг4 и Рг17 не равны, продолжать увеличивать
89. // Умножаем
90. Z(11)
91. Z(14)
92. Z(12)
93. S(12)
94. S(14)
95. J(12, 13, 16) //Если промежуочное значение числа равно самому числу
96. J(1, 1, 12)
97. S(11)
98. J(11, 15, 19) // Если промежуточное значение множителя равно самому множителю
99. J(1, 1, 11)
100. T(15, 16)
101. Z(15)
102. Z(17)
103. S(17)// {1}
104. J(16, 17, 32) // Если множитель равен 1, вывести ответ
105. S(15)
106. S(17)
107. J(16, 17, 28)
108. J(1, 1, 24) // Если Рг16 и Рг17 не равны, продолжать увеличивать
109. T(14, 13) // Копируем предыдущий результат умножения в регистр с копией числа
110. J(1, 1, 9)
111. Z(14)
112. S(14)
113. }
114. // Деление 100\*x^[2k+1]/[2k+1]!
115. program div
116. {
117. J(99, 102, 1) // Завершилось ли вычисление 100\*x^[2k+1]
118. J(99, 103, 2) // Завершилось ли вычисление [2k+1]!
119. Z(22)
120. Z(23)
121. Z(24)
122. S(22) // Увеличение промежуточного значения числа А
123. S(23) // Увеличение промежуточного значения числа B
124. J(14, 23, 11) // Равно ли число B промежуточному значению
125. J(10, 22, 14) // Равно ли число А промежуточному значению
126. J(1, 1, 6)
127. S(24) // +1 в ответ
128. Z(23)
129. J(1, 1, 8)
130. Z(22)
131. Z(23)
132. }
133. //Суммирование
134. program sum
135. {
136. Z(98)
137. S(98)
138. J(25, 98, 9) // (k = 1?) Если k нечетное, прибавляем к "разности", иначе к "сумме"
139. // прибавляем к "сумме"
140. Z(28) // очищаем значение промежуточной суммы
141. J(24, 28, 16) // если промежуточная сумма и прибавляемая сумма равны, идем увел. k
142. S(26)
143. S(28)
144. J(1, 1, 5)
145. // Прибавляем к разности
146. Z(28) // очищаем значение промежуточной суммы
147. J(24, 28, 14) // если промежуточная сумма и прибавляемая сумма равны, идем увел. k
148. S(27)
149. S(28)
150. J(1, 1, 10)
151. Z(25)
152. J(1, 1, 17)
153. S(25)
154. S(5) //k+1
155. }
156. // разность сумм +100
157. program fin
158. {
159. J(26, 27, 5)
160. S(27)
161. S(29)
162. J(1, 1, 1)
163. Z(31)
164. J(31, 20, 10)
165. S(29)
166. S(31)
167. J(1, 1, 6)
168. T(29, 0)
169. }
170. **Экранные формы**

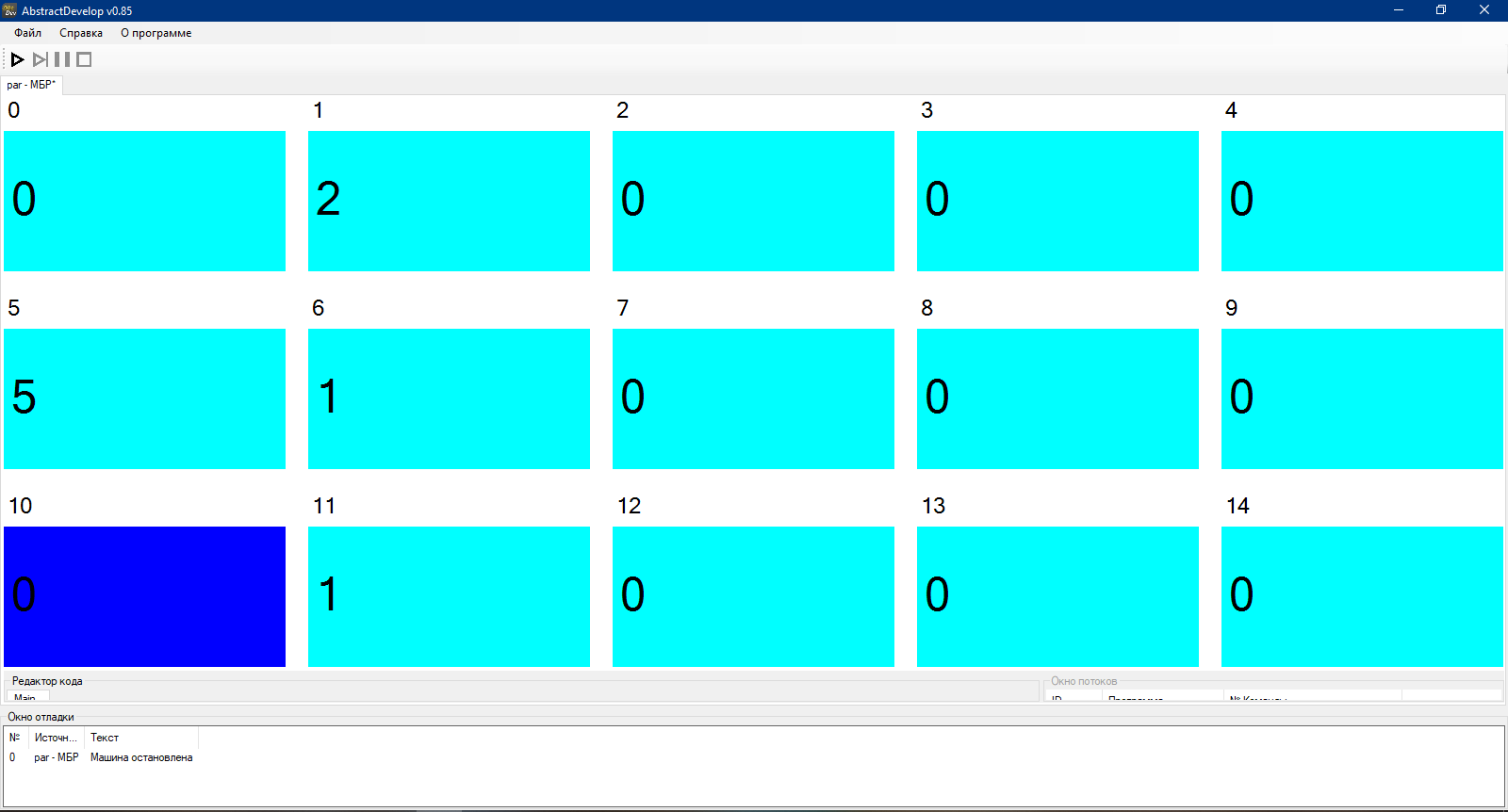


Рисунок 1 – Начальное состояние ПМБР

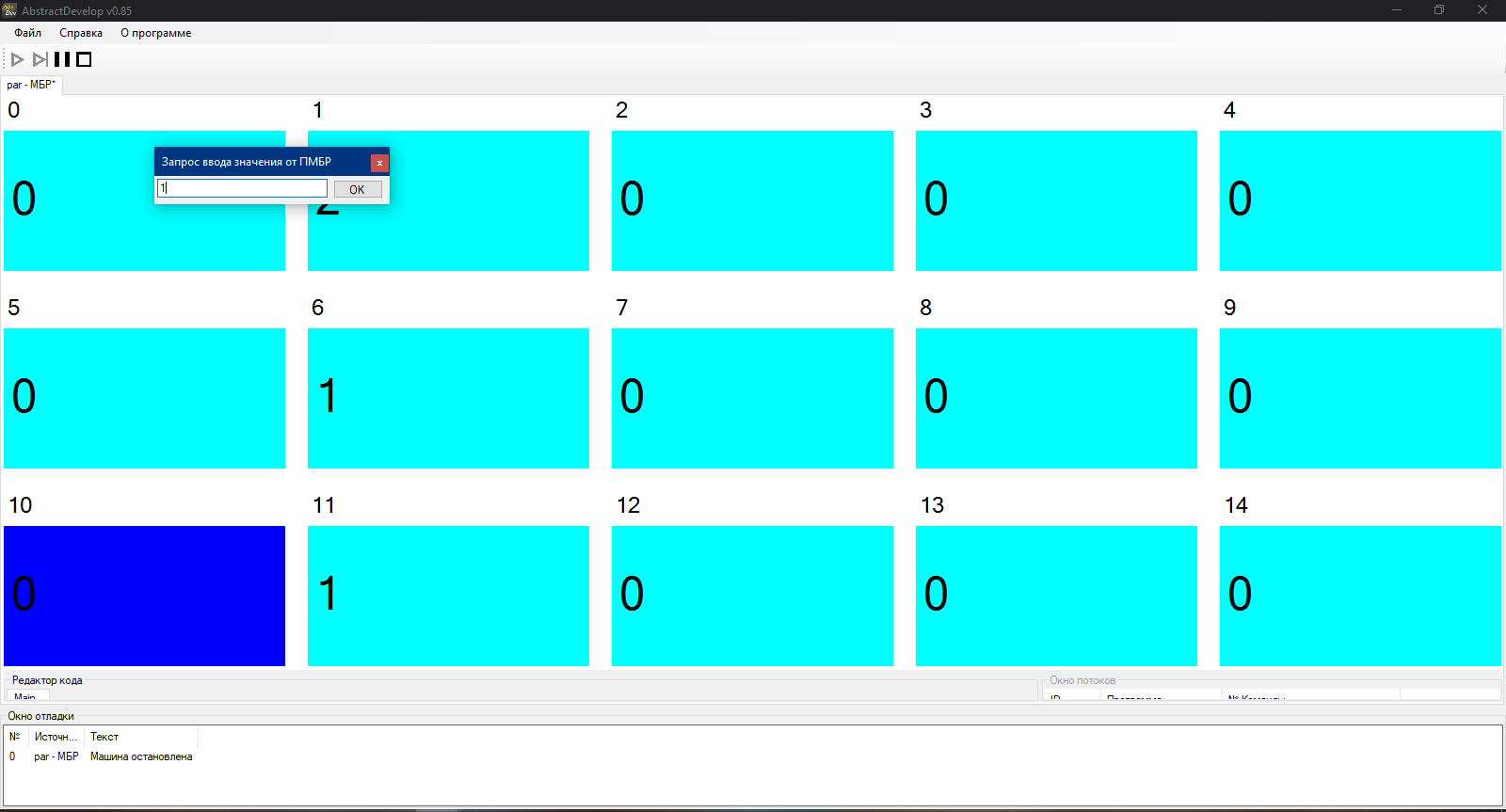


Рисунок 2 – Ввод X в ПМБР

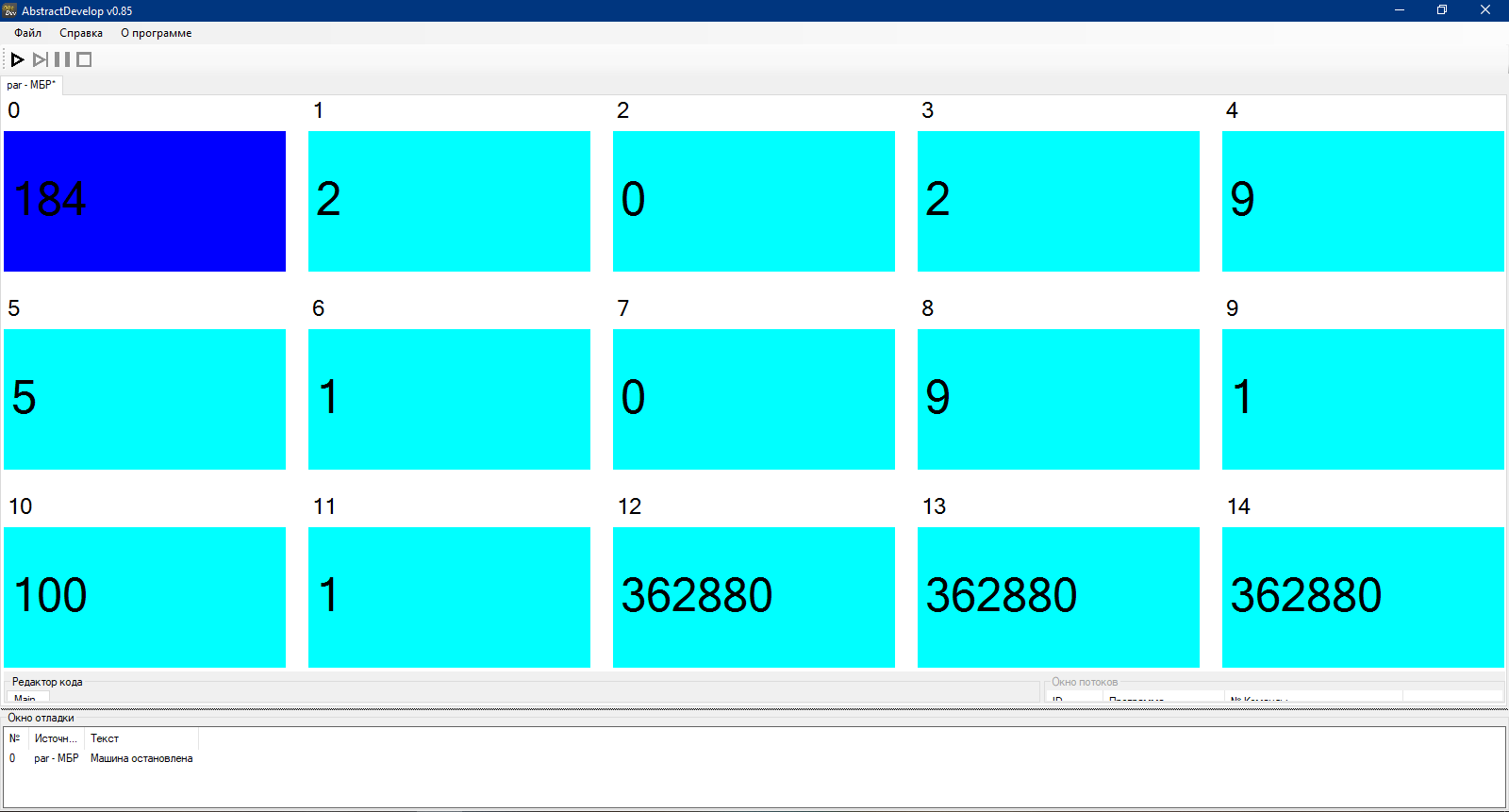


Рисунок 3 – Конечное состояние ПМБР

**Вывод:** ходе выполнения лабораторной работы была изучена абстрактная вычислительная машина - параллельная машина с бесконечными регистрами и правила составления программы для неё.

В результате выполнения лабораторной работы были разработаны независимые программы для решения поставленной задачи. Представлен словесный алгоритм выполнения программы. Поставленная задача выполнена успешно.

Работа ПМБР отличается от работы классической МБР тем, что у нее присутствует возможность параллельного выполнения заданных программ. То есть, в один и тот же момент времени могут выполняться разные команды. В связи с этим, система команд для МБР была расширена некоторыми новыми командами, такими как: запуск программы с последующим параллельным ее выполнением, запись в регистр значения с устройства ввода, запись значения регистра в устройство вывода, блокирование регистра с целью монопольного доступа к нему процессом-блокиратором, разблокировка регистра. Полный список команд для ПМБР имеет следующий синтаксис:

S(n) // Увеличение значения регистра n на 1

T(n, m) // Копирование значения регистра n в регистр m

Z(n) // Обнуление значения регистра n

J(n, m, k) // Условный переход. Если значения регистров m и n равны, то переход к команде с номером k

s(prog, n) /\* Запуск программы prog в отдельном потоке и ассоциация её выполнения с регистром-индикатором n.

Перед началом выполнения программы регистр n обнуляется, после выполнения программы в него

заносится единица \*/

I(n) // Запись значения с устройства ввода с последующей записью его в регистр n

O(n) // Чтение значения регистра n и запись его в устройство вывода

G(n) // Блокировка регистра с номером n с целью монопольного обращения к нему процессом-блокиратором

P(n) // Разблокировка регистра с номером n и разрешение доступа к нему другим процессам