

Государственное образовательное учреждение высшего профессионального образования  
“Московский государственный технический университет имени Н.Э.Баумана”

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

Дисциплина: АНАЛИЗ АЛГОРИТМОВ

ЛАБОРАТОРНАЯ РАБОТА № 5

## **Конвейерные вычисления**

Студент:

Ильясов Идрис Магомет-Салиевич

Группа: ИУ7-53Б

Преподаватели:

Волкова Лилия Леонидовна

Строганов Юрий Владимирович

Москва, 2019 г.

# Содержание

<b>Введение</b>	<b>2</b>
<b>1 Аналитическая часть</b>	<b>3</b>
1.1 Описание задачи . . . . .	3
1.2 Выводы . . . . .	3
<b>2 Конструкторская часть</b>	<b>4</b>
2.1 Функциональная модель . . . . .	4
2.2 Разработка алгоритмов . . . . .	5
2.3 Выводы . . . . .	5
<b>3 Технологическая часть</b>	<b>6</b>
3.1 Требования к программному обеспечению . . . . .	6
3.2 Средства реализации . . . . .	6
3.3 Листинг кода . . . . .	6
3.4 Выводы . . . . .	11
<b>4 Экспериментальная часть</b>	<b>12</b>
4.1 Примеры работы . . . . .	12
4.2 Замеры времени . . . . .	13
4.3 Выводы . . . . .	14
<b>Заключение</b>	<b>15</b>
<b>Список литературы</b>	<b>16</b>

# Введение

Имеется большое количество важнейших задач, решение которых требует использования огромных вычислительных мощностей, зачастую недоступных для современных вычислительных систем.

Постоянно появляются новые задачи подобного рода и возрастают требования к точности и к скорости решения прежних задач; поэтому вопросы разработки и использования сверхмощных компьютеров (называемых суперкомпьютерами) актуальны сейчас и в будущем [1]. Но пока эти трудности пока что не удается преодолеть. Из-за этого приходится и эти по пути создания параллельных вычислительных систем, т.е. систем, в которых предусмотрена одновременная реализация ряда вычислительных процессов, связанных с решением одной задачи [1]. На современном этапе развития вычислительной техники такой способ, по-видимому, является одним из основных способов ускорения вычислений.

Многие явления природы характеризуются параллелизмом (одновременным исполнением процессов с применением различных путей и способов). В частности, в живой природе параллелизм распространен очень широко в дублирующих системах для получения надежного результата. Параллельные процессы пронизывают общественные отношения, ими характеризуются развитие науки, культуры и экономики в человеческом обществе. Среди этих процессов особую роль играют параллельные информационные потоки.

Среди параллельных систем различают конвейерные, векторные, матричные, систолические, спецпроцессоры и т.п. В данной работе используются конвейерные [1].

В данной работе стоит задача реализации алгоритма деления чисел с плавающей запятой, сравнение последовательной и конвейерной реализаций.

# 1 Аналитическая часть

В данном разделе будет рассмотрена идея конвейерных вычислений.

## 1.1 Описание задачи

Конвейеризация – это техника, в результате которой задача или команда разбивается на некоторое число подзадач, которые выполняются последовательно. Каждая подкоманда выполняется на своем логическом устройстве. Все логические устройства (ступени) соединяются последовательно таким образом, что выход  $i$ -ой ступени связан с входом  $(i+1)$ -ой ступени, все ступени работают одновременно. Множество ступеней называется конвейером. Выигрыш во времени достигается при выполнении нескольких задач за счет параллельной работы ступеней, вовлекая на каждом такте новую задачу или команду.[4]

В конвейере различают  $r$  последовательных этапов, так что когда  $i$ -я операция проходит  $s$ -й этап, то  $(i+k)$ -я операция проходит  $(s-k)$ -й этап.

В данной работе реализован алгоритм деления чисел с плавающей запятой, состоящих из знака, порядка и мантииссы.

## 1.2 Выводы

Таким образом, выигрыш во времени достигается при выполнении нескольких задач за счет параллельной работы ступеней, вовлекая на каждом такте новую задачу или команду. Однако работу конвейера тормозят зависимости по данным, конфликты по ресурсам.

## 2 Конструкторская часть

В данном разделе будут приведены функциональная модель и схема алгоритма.

### 2.1 Функциональная модель

На рисунке 1 приведена функциональная модель решаемой задачи.

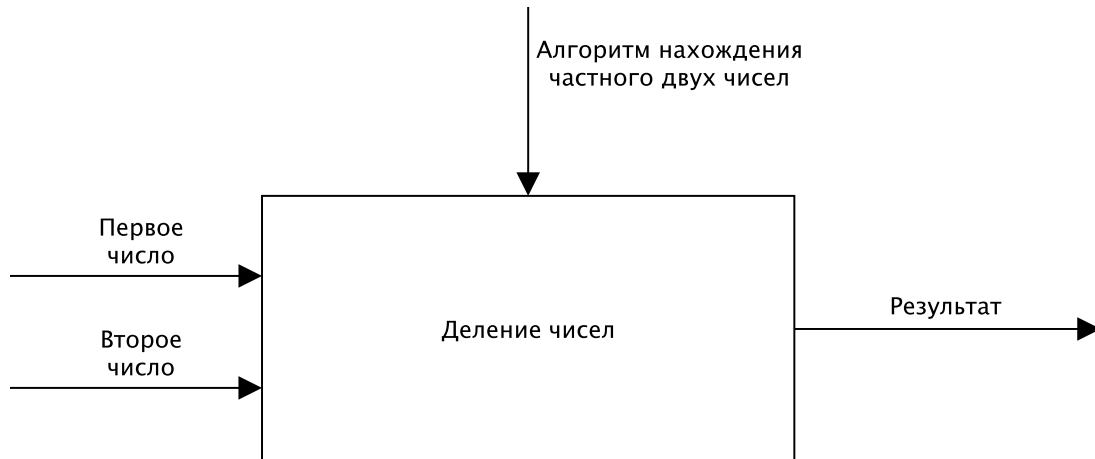


Рис. 1 - Функциональная модель алгоритма нахождения частного двух чисел.

## 2.2 Разработка алгоритмов

На рисунке 2 приведена схема алгоритма деления двух чисел.



Рис. 2 - Схема алгоритма нахождения деления двух чисел.

## 2.3 Выводы

В данном разделе была рассмотрена схема алгоритма нахождения частного двух чисел.

## 3 Технологическая часть

В данном разделе представлены требования к разрабатываемому программному обеспечению, средства, использованные в процессе разработки для реализации поставленных задач, а также листинг кода программы.

### 3.1 Требования к программному обеспечению

ПО должно предоставлять возможность замеров процессорного времени выполнения реализации каждого алгоритма. Требуется провести замеры для варьирующегося количества чисел. Один эксперимент ставится не менее 5 раз, результат одного эксперимента рассчитывается как среднее значение результатов проведенных испытаний с одинаковыми входными данными.

### 3.2 Средства реализации

В качестве языка программирования был выбран C++, так как я знаком с этим языком программирования, и он удовлетворяет требованиям об необходимости использовании нативных потоков [3]. Для замеров времени была выбрана метод *high\_resolution\_clock::now()*, возвращает текущее время процессора как число тиков, выраженное в микросекундах в Unix. Для генерации случайных чисел использовался метод *rand()*. Для работы с потоками использована библиотека *<thread>*.

### 3.3 Листинг кода

В приведенных ниже листингах представлены реализации алгоритма Кнута-Морриса-Пратта(листинг 1), алгоритма Бойера-Мура (листинг 2).

Листинг 1: Считывание первого числа

```
1 void thread_read_first()
2 {
3     Build_num bld;
4
5     while (true)
6     {
7         if (str1.empty())
8         {
9             sign++;
10            return;
11        }
12
13        bld.build(str1.front());
14        str1.pop();
15        lock_output.lock();
16        std::cout << "1:_" << bld.get_res() << std::endl;
17        lock_output.unlock();
18    }
```

```

19         if (!bld.get_res().error)
20         {
21             lock_data.lock();
22             zeros_1.push(bld.get_res());
23             lock_data.unlock();
24         }
25     }
26 }

```

Листинг 2: Считывание второго числа

```

1 void thread_read_second()
2 {
3     Build_num bld;
4
5     while (true)
6     {
7         if (str2.empty())
8         {
9             sign++;
10            return;
11        }
12
13        bld.build(str2.front());
14        str2.pop();
15        lock_output.lock();
16        std::cout << "2:_" << bld.get_res() << std::endl;
17        lock_output.unlock();
18
19        if (!bld.get_res().error)
20        {
21            lock_data.lock();
22            zeros_2.push(bld.get_res());
23            lock_data.unlock();
24        }
25    }
26 }

```



Листинг 3: Удаление незначащих нулей в первом числе

```

1 void thread_insig_first()
2 {
3     Long_int num;
4
5     while (true)
6     {
7         if (zeros_1.empty())
8         {
9             if (sign >= 2)
10            {
11                sign++;
12                return;
13            }
14            continue;
15        }
16
17        num = zeros_1.front();
18
19        lock_data.lock();
20        zeros_1.pop();
21        lock_data.unlock();
22        num.remove_insig();
23
24        lock_output.lock();
25        std::cout << "3:_" << num << std::endl;
26        lock_output.unlock();
27        lock_data.lock();
28        num_1.push(num);
29        lock_data.unlock();
30    }
31 }

```

Листинг 4: Удаление незначащих нулей во втором числе

```

1 void thread_insig_second()
2 {
3     Long_int num;
4
5     while (true)
6     {
7         if (zeros_2.empty())

```

```

8      {
9          if (sign >= 3)
10         {
11             sign++;
12             return;
13         }
14         continue;
15     }
16
17     num = zeros_2.front();
18
19     lock_data.lock();
20     zeros_1.pop();
21     lock_data.unlock();
22     num.remove_insig();
23
24     lock_output.lock();
25     std::cout << "4:_ " << num << std::endl;
26     lock_output.unlock();
27     lock_data.lock();
28     num_1.push(num);
29     lock_data.unlock();
30 }
31 }

```

Листинг 5: Удаление незначущих нулей в результате

```

1 void thread_divide_nums()
2 {
3     Long_int num;
4
5     while (true)
6     {
7         if (num_1.empty() || num_2.empty())
8         {
9             if (sign >= 4)
10            {
11                sign++;
12                return;
13            }
14            continue;

```

```

15     }
16
17     num.divide(num_1.front(), num_2.front());
18
19     lock_data.lock();
20     num_1.pop();
21     num_2.pop();
22     lock_data.unlock();
23
24     lock_output.lock();
25     std::cout << "5:_" << num << std::endl;
26     lock_output.unlock();
27
28     if (!num.error)
29     {
30         lock_data.lock();
31         zeros.push(num);
32         lock_data.unlock();
33     }
34 }
35 }

```

Листинг 6: Удаление незначущих нулей в результате

```

1 void thread_insig_result()
2 {
3     Long_int num;
4
5     while (true)
6     {
7         if (zeros.empty())
8         {
9             if (sign >= 5)
10            {
11                sign++;
12                return;
13            }
14            continue;
15        }
16
17        num = zeros.front();

```

```

18         lock_data.lock();
19         zeros.pop();
20         lock_data.unlock();
21
22         num.remove_insig();
23
24         lock_output.lock();
25         std::cout << "6:_" << num << std::endl;
26         lock_output.unlock();
27     }
28 }

```

### 3.4 Выводы

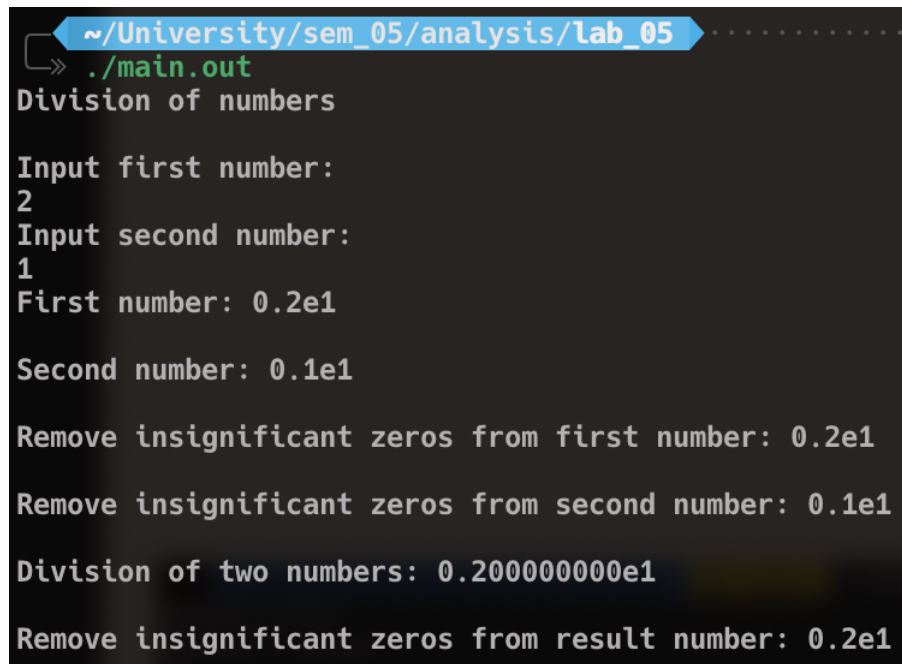
На основе схем алгоритмов, представленных в конструкторском разделе, в соответствии с указанными требованиями к реализации с использованием средств языка C++ было разработано программное обеспечение, содержащее реализацию выбранного алгоритма.

## 4 Экспериментальная часть

В данном разделе будут приведены примеры работы разработанного программного обеспечения.

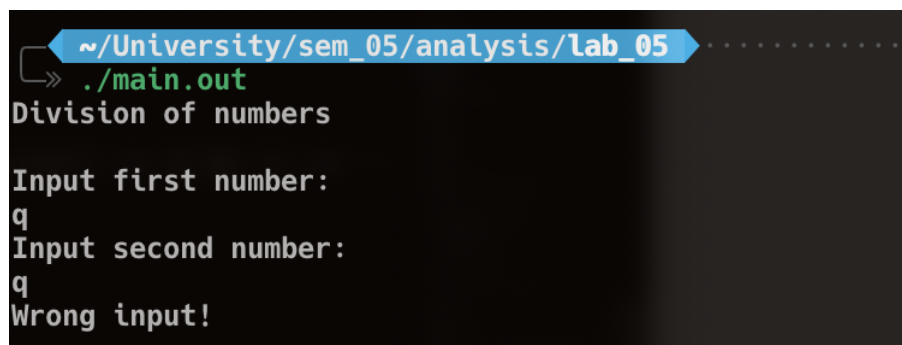
### 4.1 Примеры работы

На приведенных ниже рисунках продемонстрирована работа выбранных алгоритмов.



```
~/University/sem_05/analysis/lab_05  
» ./main.out  
Division of numbers  
  
Input first number:  
2  
Input second number:  
1  
First number: 0.2e1  
  
Second number: 0.1e1  
  
Remove insignificant zeros from first number: 0.2e1  
Remove insignificant zeros from second number: 0.1e1  
Division of two numbers: 0.200000000e1  
Remove insignificant zeros from result number: 0.2e1
```

Рис. 4 - Пример работы



```
~/University/sem_05/analysis/lab_05  
» ./main.out  
Division of numbers  
  
Input first number:  
q  
Input second number:  
q  
Wrong input!
```

Рис. 5 - Пример работы

```

~/University/sem_05/analysis/lab_05 .....
» ./main.out 3
Division of numbers

First number: -0.80e50
Second number: 0.522737e4
First number: -0.7039e69
Second number: 0.70e-997
Remove insignificant zeros from first number: -0.8e50
Remove insignificant zeros from second number: 0.522737e4
Second number: -0.84e46034
First number: 0.3655e18
Remove insignificant zeros from first number: -0.7039e69
Remove insignificant zeros from first number: 0.3655e18
Division of two numbers: -0.1530e47
Remove insignificant zeros from second number: 0.7e-997
Remove insignificant zeros from second number: -0.84e46034
Remove insignificant zeros from result number: -0.153e47
Division of two numbers: -0.100557142e1067
Remove insignificant zeros from result number: -0.100557142e1067
Division of two numbers: -0.04351190e-46015
Remove insignificant zeros from result number: -0.435119e-46016

```

Рис. 6 - Пример работы

## 4.2 Замеры времени

На рисунке 7 приведены результаты сравнения двух методов вычисления частного чисел на количествах пар чисел от 100 до 1000 с шагом 100.

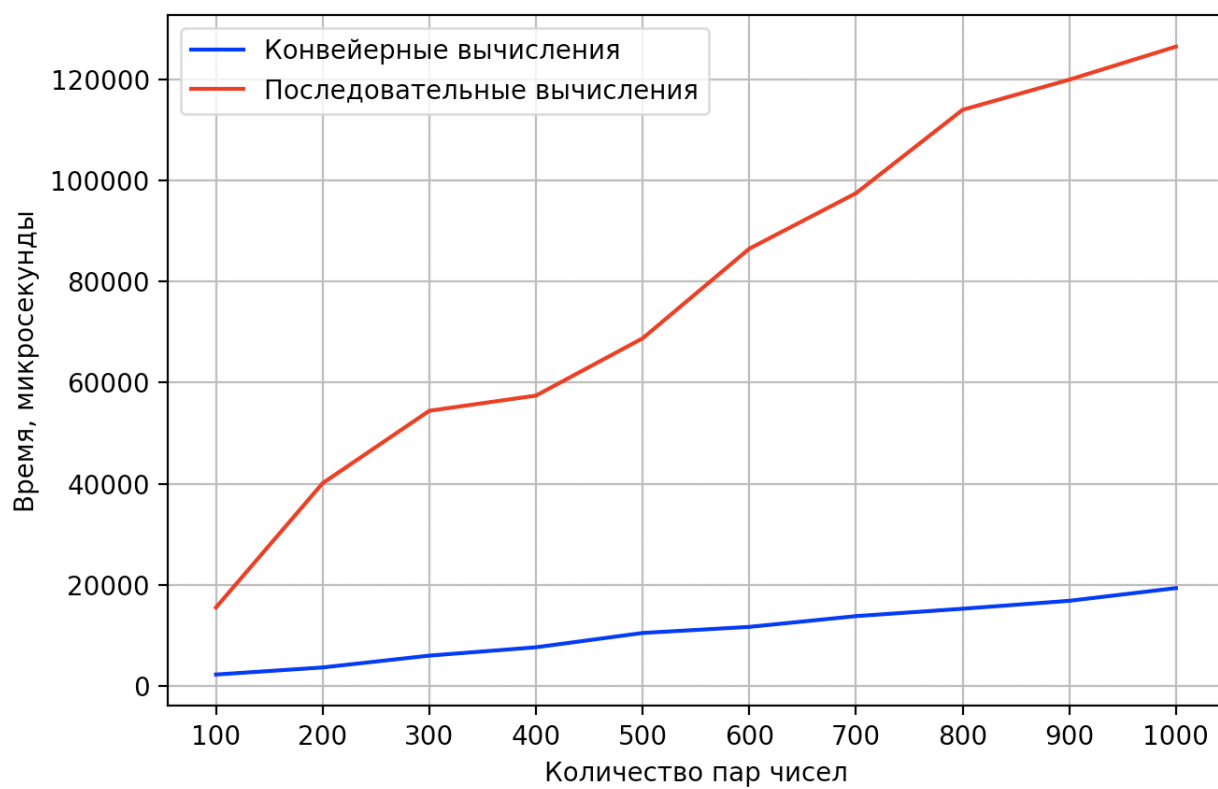


Рис. 7 - Сравнение реализации алгоритмов нахождения частного чисел

### 4.3 Выводы

В этом разделе были приведены примеры работы разработанного ПО и проведено сравнение двух реализаций алгоритмов нахождения частного чисел.

## Заключение

В данной работе реализован алгоритм деления чисел с плавающей запятой и проведено сравнение последовательной и конвейерной реализаций. Ступени такого конвейера выполняются последовательно, а операции одной ступени параллельно. Теория, на основе параллельно-последовательной схемы, позволяет аналитически вычислять основные характеристики конвейерного процесса и составлять расписание процесса. Модели конвейерных процессов, в той или иной модификации, широко используются для решения различных прикладных задач. Однако при небольших вычислениях, из-за блокировки ресурсов и передачи данных между потоками, конвейерная обработка проигрывает последовательной.



## Список литературы

1. Корнеев В.В. Параллельные вычислительные системы. М., 1999. 320 с.
2. Семинары по анализу алгоритмов.
3. <http://www.myshared.ru/slide/674082/> (дата обращения: 14.12.2019)