

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

Дисциплина: АНАЛИЗ АЛГОРИТМОВ

ЛАБОРАТОРНАЯ РАБОТА № 5

Конвейерные вычисления

Студент:

Сиденко Анастасия Геннадьевна

Группа: ИУ7-53Б

Преподаватели:

Строганов Юрий Владимирович

Волкова Лилия Леонидовна

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 Схемы алгоритмов . . . . .	4
2.3 Выводы . . . . .	4
<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 . . . . .	9
<b>4 Экспериментальная часть</b>	<b>10</b>
4.1 Примеры работы . . . . .	10
4.2 Замеры времени . . . . .	10
4.3 Выводы . . . . .	11
<b>Заключение</b>	<b>12</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.[2] 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. [3]

Среди параллельных систем различают конвейерные, векторные, матричные, систолические, спецпроцессоры и т.п. В данной работе используются конвейерные. [2]

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

# 1 Аналитическая часть

Одной из важнейших идей при создании многопроцессорных систем и при эффективной реализации алгоритмов на этих системах является 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  $г$  последовательных этапов, так что когда  $i$ -я операция проходит  $s$ -й этап, то  $(i+k)$ -я операция проходит  $(s-k)$ -й этап (рис. 1).

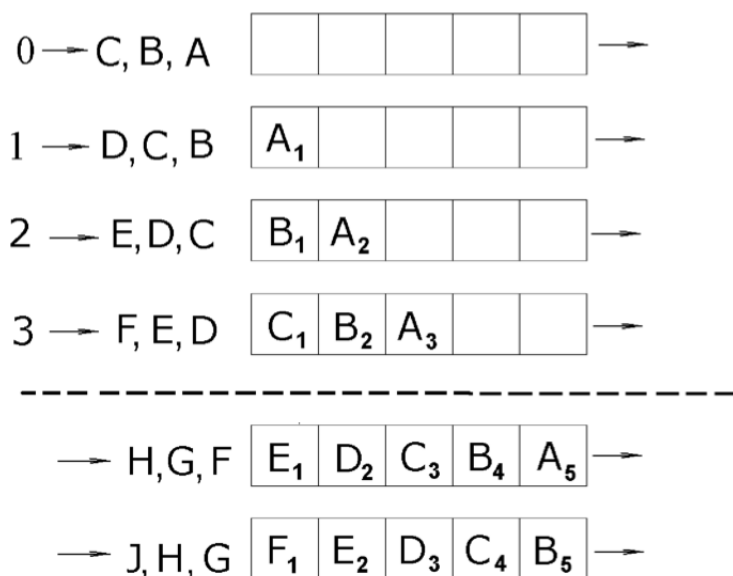


Рис. 1 - Схема конвейера.

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

## 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

Стоит задача конвейерных вычислений алгоритма деления чисел с плавающей запятой.

### 2.1 Функциональная модель

На рисунке 2 представлена функциональная модель нашей задачи.

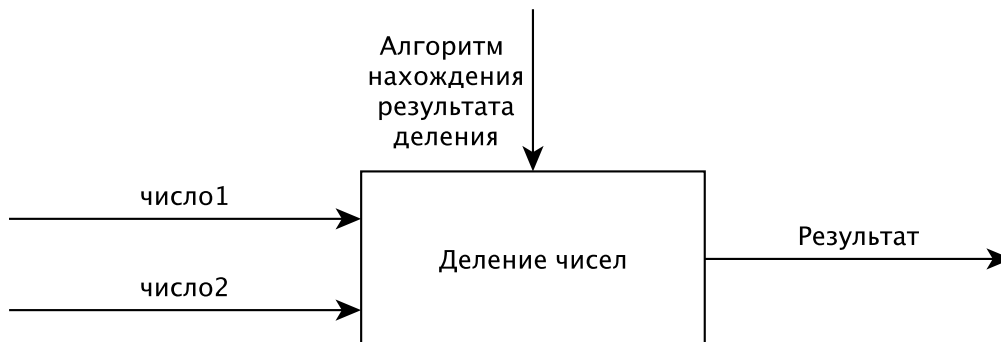


Рис. 2 - Функциональная модель алгоритма нахождения произведения матриц.

### 2.2 Схемы алгоритмов

Приведем схему алгоритма (см. рисунок 3), где каждое действие выполняется в отдельном потоке.

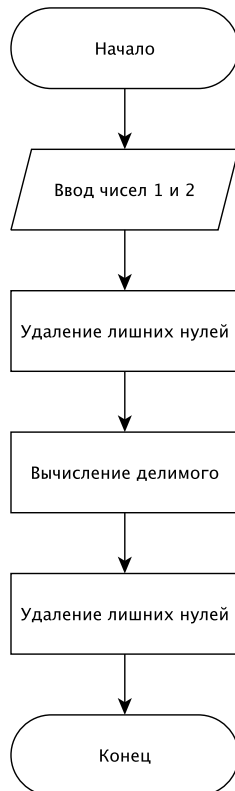


Рис. 3 - Алгоритм работы с программы

### 2.3 Выводы

Описанный принцип построения процессора действительно напоминает конвейер сборочного завода, на котором изделие последовательно проходит ряд рабочих мест. На каждом из этих мест над изделием производится новая операция. Эффект ускорения достигается за счет одновременной обработки ряда изделий на разных рабочих местах.[6]

## 3 Технологическая часть

Необходимо изучить и реализовать конвейерную обработку процесса деления чисел.

### 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 знакома с этим языком программирования и он удовлетворяет требованиям об необходимости использовании нативных потоков. [11]

Для замеров времени была выбран метод *high\_resolution\_clock :: now()*, возвращает текущее время процессора как число тиков, выраженное в микросекундах в Unix.[5]

Для генерации случайных чисел использовался метод *rand()*.

Для работы с потоками использована библиотека *<thread>*.

### 3.3 Листинг кода

Код алгоритма деления чисел с плавающей запятой с помощью конвейерных вычислений представлен на листингах 1-6.

Листинг 1: Чтение 1 числа

```
1 void thread1() {
2     Builder builder;
3
4     while (true) {
5         if (str1.empty()) {
6             sign++;
7             return;
8         }
9         builder.build_number(str1.front());
10        str1.pop();
11        cout_lock.lock();
12        cout << 1 << ' ' << builder.get_result() << endl;
13        cout_lock.unlock();
14        if (!builder.get_result().error) {
15            data_lock.lock();
16            nulesnumber1.push(builder.get_result());
17            data_lock.unlock();
18        }
19    }
20 }
```

Листинг 2: Чтение 2 числа

```
1 void thread2() {
2     Builder builder;
3
4     while (true) {
5         if (str2.empty()) {
6             sign++;
7             return;
8         }
9     }
```

```

9      builder.build_number(str2.front());
10     str2.pop();
11     cout_lock.lock();
12     cout << 2 << ' ' << builder.get_result() << endl;
13     cout_lock.unlock();
14     if (!builder.get_result().error) {
15         data_lock.lock();
16         nulesnumber2.push(builder.get_result());
17         data_lock.unlock();
18     }
19 }
20 }

```

Листинг 3: Удаление незначаших нулей из числа 1

```

1 void thread3() {
2     Long_number num;
3
4     while (true) {
5         if (nulesnumber1.empty()) {
6             if (sign >= 2) {
7                 sign++;
8                 return;
9             }
10            continue;
11        }
12        num = nulesnumber1.front();
13        data_lock.lock();
14        nulesnumber1.pop();
15        data_lock.unlock();
16        num.insignificant_nules();
17
18        cout_lock.lock();
19        cout << 3 << ' ' << num << endl;
20        cout_lock.unlock();
21        data_lock.lock();
22        number1.push(num);
23        data_lock.unlock();
24    }
25 }

```

Листинг 4: Удаление незначаших нулей из числа 2

```

1 void thread4() {
2     Long_number num;
3
4     while (true) {
5         if (nulesnumber2.empty()) {
6             if (sign >= 3) {
7                 sign++;
8                 return;
9             }
10            continue;
11        }
12        num = nulesnumber2.front();
13        data_lock.lock();
14        nulesnumber2.pop();
15        data_lock.unlock();
16        num.insignificant_nules();
17
18        cout_lock.lock();

```



```

19     cout << 4 << ' ' << num << endl;
20     cout_lock.unlock();
21     data_lock.lock();
22     number2.push(num);
23     data_lock.unlock();
24 }
25 }

```

Листинг 5: Деление

```

1 void thread5() {
2     Long_number num;
3
4     while (true) {
5         if (number1.empty() || number2.empty()) {
6             if (sign >= 4) {
7                 sign++;
8                 return;
9             }
10            continue;
11        }
12        num.division(number1.front(), number2.front());
13        data_lock.lock();
14        number1.pop();
15        number2.pop();
16        data_lock.unlock();
17
18        cout_lock.lock();
19        cout << 5 << ' ' << num << endl;
20        cout_lock.unlock();
21        if (!num.error) {
22            data_lock.lock();
23            nulesnumber.push(num);
24            data_lock.unlock();
25        }
26    }
27 }

```

Листинг 6: Удаление незначащих нулей из результата

```

1 void thread6() {
2     Long_number num;
3     while (true) {
4         if (nulesnumber.empty()) {
5             if (sign >= 5) {
6                 sign++;
7                 return;
8             }
9             continue;
10        }
11        num = nulesnumber.front();
12        data_lock.lock();
13        nulesnumber.pop();
14        data_lock.unlock();
15        num.insignificant_nules();
16
17        cout_lock.lock();
18        cout << 6 << ' ' << num;
19        cout_lock.unlock();
20    }
21 }

```

### 3.4 Выводы

Описанный принцип построения процессора действительно напоминает конвейер. На каждой из лент производится новая операция. Эффект ускорения достигается за счет одновременной обработки нескольких операций на разных лентах конвейера.

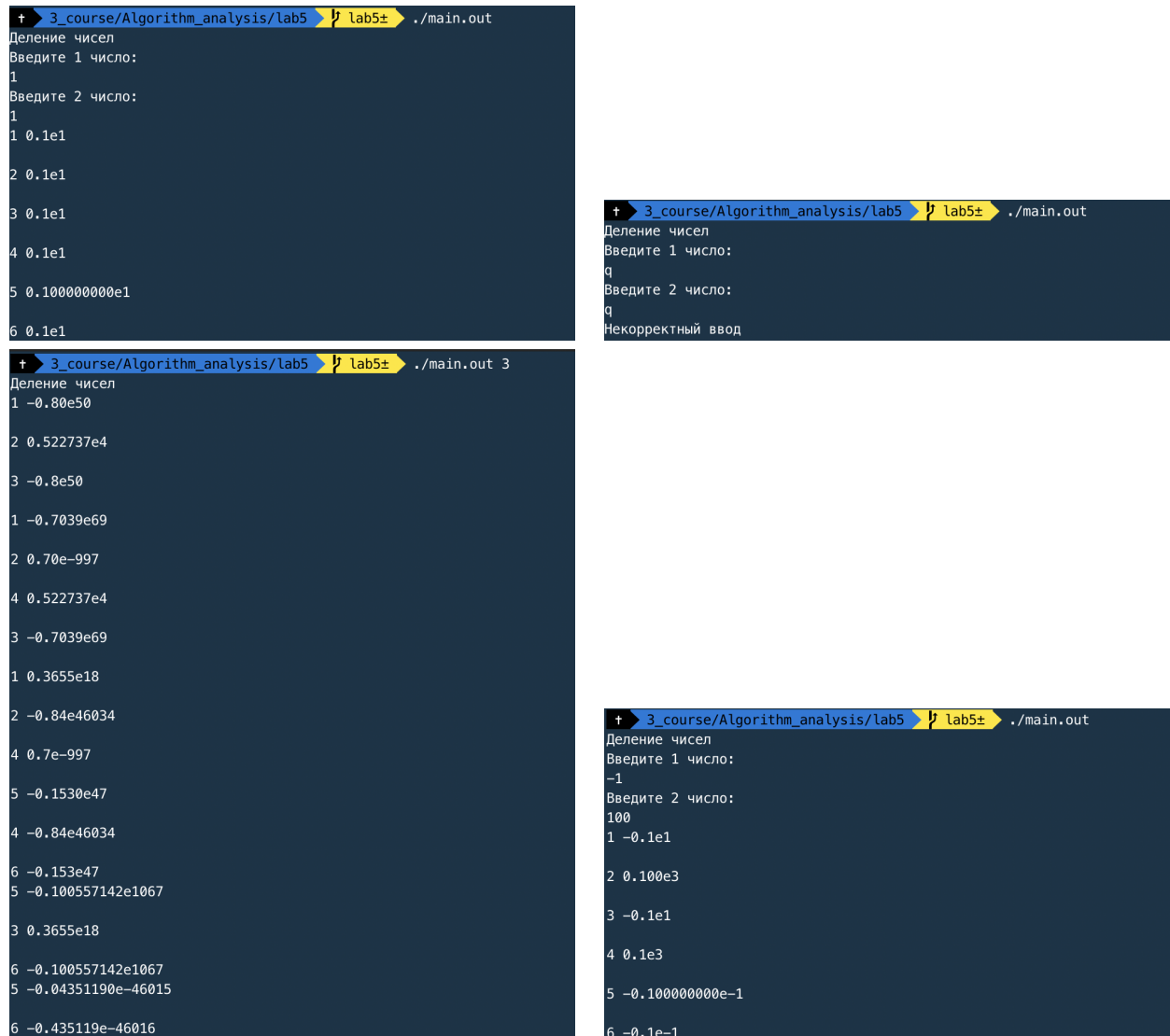
## 4 Экспериментальная часть

Стоит задача разработки и сравнительного анализа алгоритмов, вычисляющих деление чисел с плавающей запятой последовательным и конвейерным способом.

В реализациях в целях увеличения точности подсчета времени вывод был вынесен за пределы функций-алгоритмов. В целях наглядности были опущены части программ, не относящиеся к работе алгоритмов.

### 4.1 Примеры работы

На рисунке 4 представлены примеры работы программы на разных входных данных. В параметрах командной строки можно указать желаемое число количества чисел, по умолчанию 1.



```
3_course/Algorithm_analysis/lab5 lab5± ./main.out
Деление чисел
Введите 1 число:
1
Введите 2 число:
1
1 0.1e1
2 0.1e1
3 0.1e1
4 0.1e1
5 0.100000000e1
6 0.1e1

3_course/Algorithm_analysis/lab5 lab5± ./main.out 3
Деление чисел
1 -0.80e50
2 0.522737e4
3 -0.8e50
1 -0.7039e69
2 0.70e-997
4 0.522737e4
3 -0.7039e69
1 0.3655e18
2 -0.84e46034
4 0.7e-997
5 -0.1530e47
4 -0.84e46034
6 -0.153e47
5 -0.100557142e1067
3 0.3655e18
6 -0.100557142e1067
5 -0.04351190e-46015
6 -0.435119e-46016

3_course/Algorithm_analysis/lab5 lab5± ./main.out
Деление чисел
Введите 1 число:
q
Введите 2 число:
q
Некорректный ввод

3_course/Algorithm_analysis/lab5 lab5± ./main.out
Деление чисел
Введите 1 число:
-1
Введите 2 число:
100
1 -0.1e1
2 0.100e3
3 -0.1e1
4 0.1e3
5 -0.100000000e-1
6 -0.1e-1
```

Рис. 4 - Примеры работы

### 4.2 Замеры времени

На графике 5 представлено сравнение алгоритмов умножения матриц.

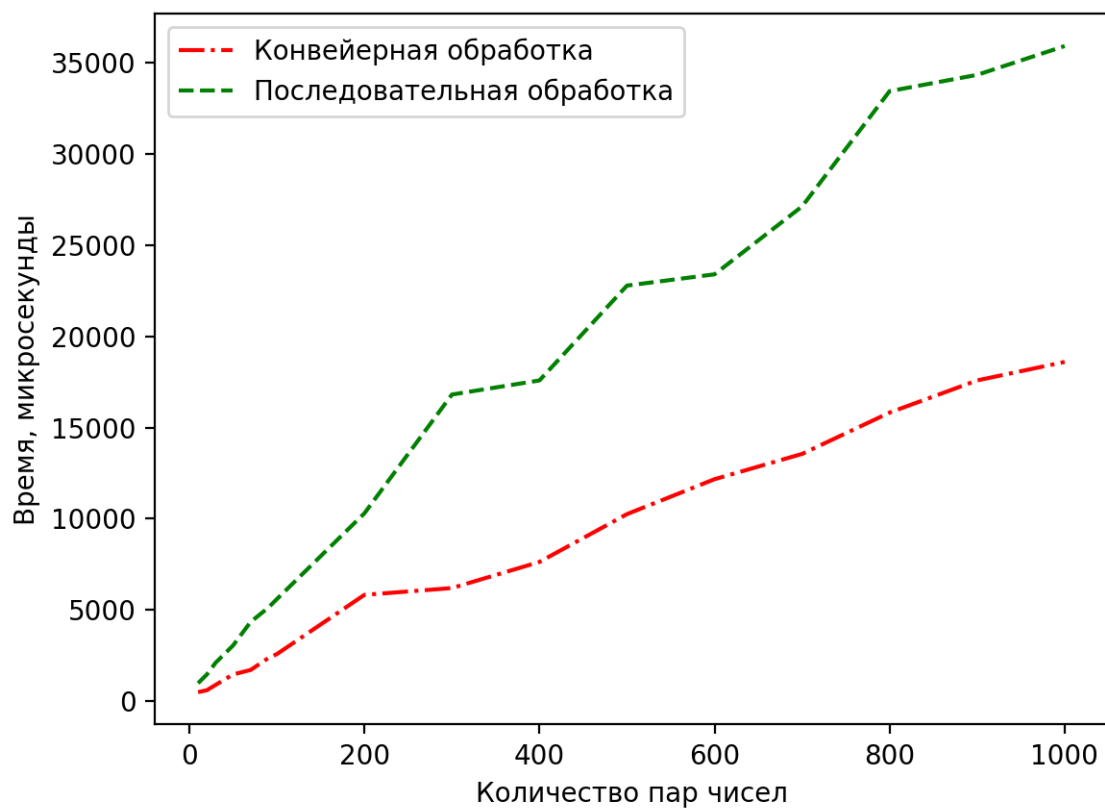


Рис. 5 - Сравнение реализации алгоритмов нахождения делимого чисел с плавающей запятой

### 4.3 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.

## 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] Воеводин В.В. Математические модели и методы в параллельных процессах. М., 1986. 296 с.
- [2] Корнеев В.В. Параллельные вычислительные системы. М., 1999. 320 с.
- [3] Yukiya Aoyama, Jun Nakano. RS/6000 SP:Practical MPI Programming. IBM. Technical Support Organization., 2000. 221
- [4] [Электронный ресурс]. - Режим доступа: <http://www.myshared.ru/slide/674082/> (дата обращения: 05.11.2019)
- [5] [Электронный ресурс]. - Режим доступа: <https://en.cppreference.com/w/cpp/chrono/duration/durationcast> (дата обращения: 22.10.2019)
- [6] [Электронный ресурс] - Режим доступа: <https://studref.com/313869/informatika/konveyerizatsiyavychisleniy> (дата обращения: 05.11.2019)
- [7] [Электронный ресурс] - Режим доступа: <https://cyberleninka.ru/article/n/primeneniye-modeli-konveyernyh-protssessov-rekursivnogo-tipa-dlya-resheniya-prikladnyh-zadach> (дата обращения: 05.11.2019)