ФЕДЕРАЛЬНОЕ АГЕНТСТВО СВЯЗИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«СИБИРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ТЕЛЕКОММУНИКАЦИЙ И ИНФОРМАТИКИ»

ФАКУЛЬТЕТ ИНФОРМАТИКИ И ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ

Кафедра вычислительных систем

ОТЧЁТ
по лабораторной работе №2
по дисциплине «Распределённые системы и технологии»

Выполнил:
студент группы МГ-165
Терешков Р. В.

Проверил:
ст. пр. Фульман В. О.

Новосибирск – 2016

Задание на проектирование

1. Разработайте программу, реализующее псевдопараллельное выполнение двух функций: одна из которых непрерывно выводит на экран символ A, а другая непрерывно на экран выводит символ B. Переключение между выполнением функций должно осуществляться раз в три секунды по сигналу от таймера.

2. Доработайте программу умножения прямоугольных матриц, разработанную в лабораторной работе 1, так, чтобы имелась возможность сгенерированную комбинацию матриц A и B записать в файл и считать из файла, чтобы провести расчет умножения матриц повторно.

3. Сгенерируйте наборы матриц A и B разных размеров (количество строк и столбов в матрицах в размерах от 16 до 4096, с шагом 16).

4. Используя ресурсы кластера Jet проведите исследования эффективности разработанной программы умножения матриц в зависимости от способа обхода матриц при расчете результата (всего 4 комбинации обхода: A по строкам и B по строкам, A по строкам и B по столбцам и т.д.). Необходимо построить для каждого из способов обхода матриц A и B графики зависимости времени умножения матриц от размера матрицы.

5. Разработайте программу, выполняющую над двумя векторами вещественных чисел размером 1024 элемента следующие вычисления: $C_i = \sqrt{A_i} * B_i$

6. Используя компилятор GCC получили текст программы на ассемблере (опция -S) для всех уровней оптимизации кода (опции -O0, -O1, -O2, -O3, -Ofast).

7. Проанализируйте сколько операций выполнится для получения результата в каждом из способов оптимизации кода.

Ход выполнения работы

1. В качестве ознакомительного задания была реализована программа, осуществляющая псевдопараллельный вывод символов A и B. Исходный код программы доступен по следующей ссылке.

2. Доработанная программа умножения прямоугольных матриц доступна по ссылке.

3. Для генерации набора матриц A и B разных размеров используется следующая функция:

```
int func() {
    int i;
    for (i = 16; i <= 4096; i += 16) {
        M = N = P = i;
        printf(" M=%d, N=%d, P=%d\n", M, N, P);

        generate();               // generate matrices
        store("superfile", "a");  // store matrices to a file
        mfree();                  // free memory
        usleep(50000);
    }
    return 0;
}
```

Результат работы функции представлен ниже на рисунке.

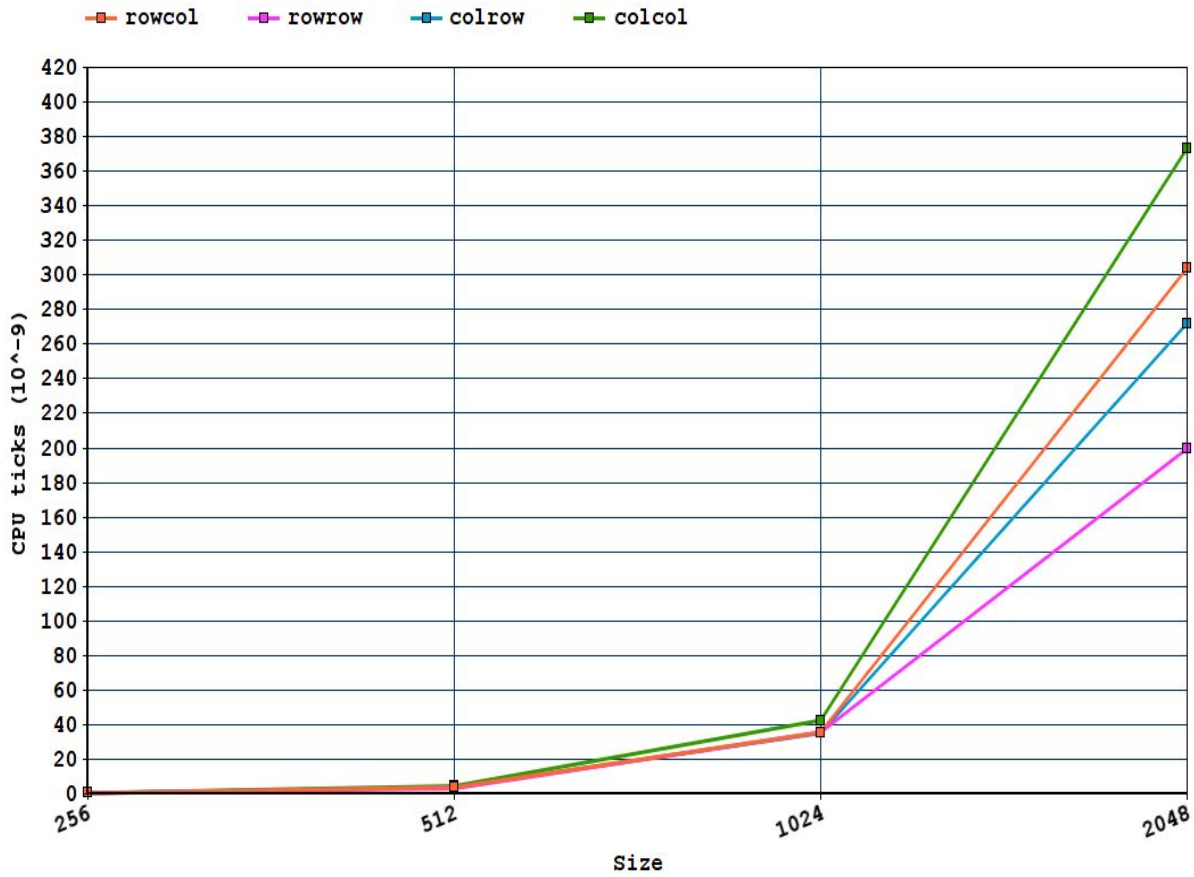
```
 GNU nano 2.3.2                Файл: superfile

16 16
11 7 7 17 1 16 12 7 11 7 12 9 7 12 14 10
11 14 5 1 13 5 15 8 6 4 15 14 10 14 14 12
3 11 11 12 1 14 1 11 11 3 10 8 5 6 1 6
3 13 6 6 1 4 13 14 15 2 1 16 15 15 10 9
8 11 3 16 15 11 9 9 13 9 16 9 6 16 14 16
3 11 4 3 5 7 7 2 8 8 8 14 13 8 5 11
2 7 10 16 8 9 15 11 1 14 2 14 3 7 12 5
17 15 15 4 12 13 13 3 3 12 16 6 2 11 16 3
8 8 10 15 17 15 8 8 2 1 4 5 7 15 17 14
3 6 8 14 1 3 7 11 14 5 16 7 15 14 9 13
13 9 10 12 15 1 10 16 9 13 11 15 1 11 11 3
7 9 7 7 11 14 17 16 9 15 5 7 2 4 10 14
4 3 16 1 11 8 7 2 3 1 7 3 2 8 13 8
16 3 14 17 16 4 6 7 9 10 4 10 5 14 15 16
7 13 7 17 4 13 9 14 4 15 8 5 5 3 12 3
5 16 2 11 2 8 1 10 8 12 11 3 8 8 9 14
16 16
1 9 12 1 11 4 15 1 8 4 17 22 12 20 7 20
22 16 16 9 11 3 13 11 7 18 4 3 10 11 10 10
20 21 9 6 1 1 4 8 2 20 8 13 16 12 8 15
5 2 21 15 2 11 2 8 6 5 9 15 13 18 1 8
14 7 13 14 7 17 20 8 12 3 18 5 14 4 19 17
5 16 7 4 4 8 12 8 10 20 20 1 13 18 8 4
2 19 16 8 11 13 13 22 15 9 5 4 12 21 20 14
14 3 15 16 8 4 1 18 21 18 16 11 14 21 13 15
17 6 21 5 18 9 3 8 17 7 11 4 3 7 15 15
9 8 8 16 9 6 9 8 1 22 16 12 19 6 3 13
11 21 16 4 7 16 11 22 22 22 1 22 4 14 14 12
19 19 3 5 2 10 10 1 9 4 10 5 7 12 16 18
10 7 21 15 22 8 12 19 5 12 18 8 3 8 17 21
2 19 2 4 4 11 2 13 12 11 15 19 21 6 12 6
12 8 18 9 15 7 5 19 19 21 2 19 4 18 16 5
13 15 6 16 3 7 4 13 16 19 9 14 22 20 17 12
32 32
11 7 7 17 1 16 12 7 11 7 12 9 7 12 14 10 11 14 5 1 13 5 15 8 6 4 15 14 10 14 $
3 11 11 12 1 14 1 11 11 3 10 8 5 6 1 6 3 13 6 6 1 4 13 14 15 2 1 16 15 15 10 $
8 11 3 16 15 11 9 9 13 9 16 9 6 16 14 16 3 11 4 3 5 7 7 2 8 8 8 14 13 8 5 11
2 7 10 16 8 9 15 11 1 14 2 14 3 7 12 5 17 15 15 4 12 13 13 3 3 12 16 6 2 11 1$
8 8 10 15 17 15 8 8 2 1 4 5 7 15 17 14 3 6 8 14 1 3 7 11 14 5 16 7 15 14 9 13
```

4. Основным заданием в данной лабораторной работе было исследование эффективности алгоритма умножения матриц в зависимости от способа обхода матриц при расчёте результата. Данный обход может осуществляться одним из следующих способов:

- обход матрицы A по строкам, матрицы B — по строкам;
- обход матрицы A по строкам, матрицы B — по столбцам;
- обход матрицы A по столбцам, матрицы B — по строкам;
- обход матрицы A по столбцам, матрицы B — по столбцам.

Ниже приведён график зависимости времени выполнения алгоритма от размера матриц и способа их обхода. Лучший результат был достигнут при использовании первого способа обхода (строка-строка).



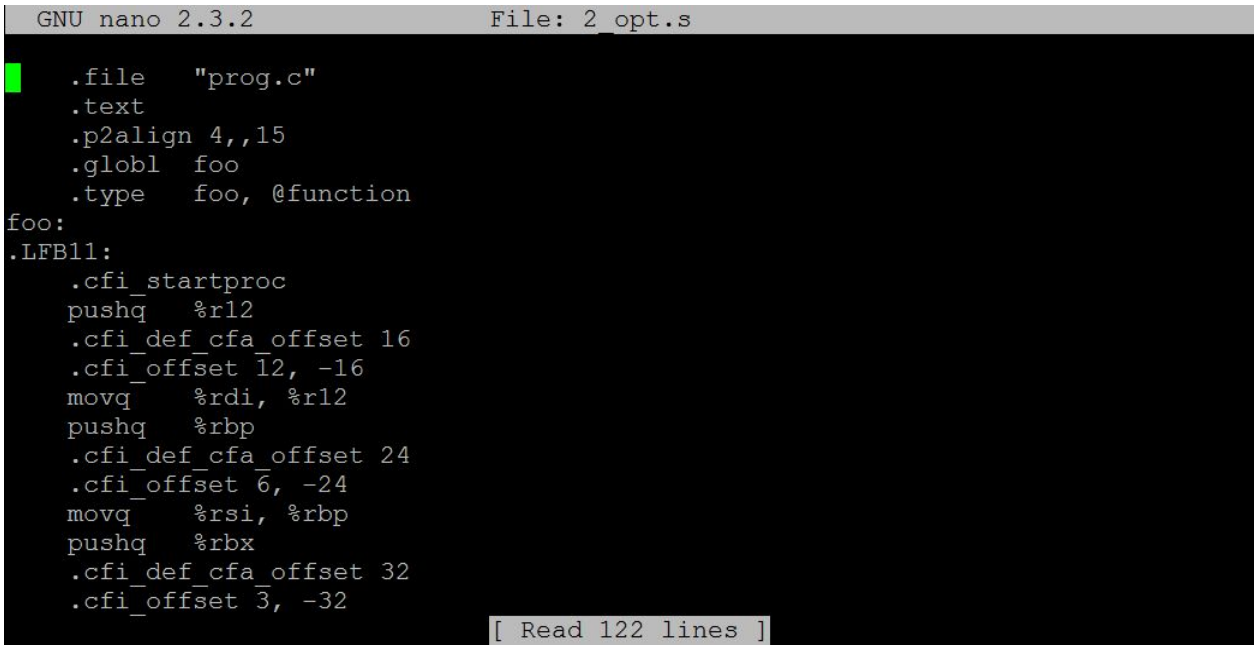
5. Код доступен по ссылке. Результат работы представлен ниже на рисунке.

```
[tereshkov@jet task3]$ ./prog
Vec1:
*****
 2.00  2.00  2.00  2.00  2.00  2.00  2.00  2.00
Vec2:
*****
 4.00  4.00  4.00  4.00  4.00  4.00  4.00  4.00
Vec3:
*****
 2.83  2.83  2.83  2.83  2.83  2.83  2.83  2.83
```

6. Для того, чтобы получить ассемблерное представление программы, необходимо на стадии компиляции указать опцию `-S`:

```
$ gcc -std=c99 -Wall -S -O2 prog.c -o 2_opt.s -lm
```

Файл, полученный в результате компиляции, изображён на рисунке снизу.



7. С помощью утилиты perf был произведён анализ количества выполненных инструкций для каждого из способов оптимизации. Полученные результаты приведены ниже в таблице.

```
$ perf stat -e instructions:u ./prog_name.s
```

	-O0	-O1	-O2	-O3
$\mathbf{N}_{instr}$	147845	129126	122198	122199
t, ms	1.62	2.01	1.84	1.45