АННОТАЦИЯ

В работе проводится исследование методов оптимизации производительности приложений для заданной процессорной архитектуры на примере открытых компиляторов Clang/LLVM. Проводятся эксперименты на 3 различных тестах, нацеленные на изучение уровней оптимизации программ, оценку флагов компилятора на наборах тестов, а также получение результатов о производительности программ.

ВВЕДЕНИЕ

LLVM – это модульный и многократно используемый компилятор и набор технологий цепочки инструментов.

Clang Compiler – это компилятор с открытым исходным кодом для языков программирования семейства C, нацеленный на лучшую в своем классе реализацию этих языков. Clang построен на оптимизаторе LLVM и генераторе кода, что позволяет ему обеспечивать высококачественную оптимизацию и поддержку генерации кода для многих целей.

Исследование предполагает проведение экспериментов на 3 различных бенчмарках, в виде многократного запуска компиляции с изменением параметров оптимизации, и замере времени выполнения.

МЕТОДОЛОГИЯ ОЦЕНКИ

Выбор флагов компилятора осуществлялся для уровня оптимизации -O3, поскольку он гарантирует выполнение большего числа настраиваемых оптимизаций. Также учитывалось то, что у каждого уровня оптимизации есть свой набор автоматически подключенных флагов, и на высоком уровне -O3 их достаточно много. Подобран набор опций, подходящих для эксперимента:

- -funroll-loops: флаг, отвечающий за развертывание циклов, количество итераций которых можно определить во время компиляции или при входе в цикл;
- -funsafe-math-optimizations: флаг, разрешающий оптимизацию арифметики с плавающей запятой, которая предполагает, что аргументы и результаты действительны;
- -fvectorize: флаг, разрешающий векторизацию циклов;
- -freciprocal-math: флаг, разрешающий использовать обратную величину вместо деления на значение, если это позволяет оптимизировать;
- -fmerge-all-constants: флаг, отвечающий за объединение идентичных констант и идентичных переменных в разных модулях компиляции;
- -ffinite-math-only: флаг, разрешающий оптимизацию для арифметики с плавающей запятой, которая предполагает, что аргументы и результаты не являются NaN или +-Inf;
- -fstrict-aliasing: в режиме strict-aliasing компилятор считает, что объекты, на которые показывают указатели «существенно различных» типов, не могут храниться в одном и том же участке памяти, и может использовать это при оптимизациях.

Далее будет проведен эксперимент, нацеленный на выявление лучших флагов для оптимизации каждой из программ.

ЭКСПЕРИМЕНТАЛЬНАЯ УСТАНОВКА

Модель процессора	Intel Core i7 6500U
Год выпуска процессора	2016
Количество ядер	2
Количество логических процессоров (потоков)	4
Тактовая частота процессора (ГГЦ)	2.5
Максимальная тактовая частота (ГГц)	3.1
Объем кэша L2 процессора (Кб)	512
Объем кэша L3 процессора (Кб)	4096
Накопитель	SSD, 256 Гб

Во избежание шума, при проведении экспериментов отключены приложения и сервисы (за исключением необходимых фоновых служб).

Метод измерения времени

Для автоматизации процесса оценки оптимизаций на тестовых программах написан скрипт на Python:

Скрипт 1. Оценка производительности

```
import subprocess
import time

res = []

for i, opt in enumerate(['-O1', '-O2', '-O3']):
    res.append([])

    subprocess.run(['ls | grep -e ".c$" | xargs clang ' + opt + ' -o out.o -lm'], shell=True)
    for j in range(10):
        start = time.time()
        subprocess.run(['./out.o ../data/test/input/inp.in'], shell=True)
        end = time.time()
        res[i].append(round(end * 1000) - round(start * 1000))

for i in res:
    print(*i)
```

Скрипт 2. Оценка флагов компилятора

```
import subprocess
import time

res = []

flags = ['-funroll-loops', '-funsafe-math-optimizations', '-fvectorize', '-freciprocal-math', '-
fmerge-all-constants', '-funroll-loops -funsafe-math-optimizations', '-funsafe-math-
optimizations -freciprocal-math -ffinite-math-only', '-fmerge-all-constants -fstrict-aliasing']
for i, opt in enumerate(flags):
    res.append([])

    subprocess.run(['ls | grep -e ".c$" | xargs clang -O3 ' + opt + ' -o out.o -lm'], shell=True)
    for j in range(10):
        start = time.time()
        subprocess.run(['./out.o 10 ./output.txt 0 0'], shell=True)
        end = time.time()
        res[i].append(round(end * 1000) - round(start * 1000))

for i in res:
    print(*i)
```

Для каждой программы был запущен скрипт, циклически перебирающий опции оптимизации (флаги) и рассчитывающий время, затраченное на выполнение.

Количество запусков для каждого теста/конфигурации

Для каждого теста с изменением уровня оптимизации (-O1, -O2, -O3) проводится 25 запусков на каждый уровень. При изменении флагов производится по 10 запусков.

Исходные данные

Тесты представлены в репозитории https://github.com/apaznikov/benchmarks.

РЕЗУЛЬТАТЫ

1. Программа 429.mcf

Результаты измерений времени выполнения программы для уровней оптимизаций -O1, -O2, -O3 приведены на графике рис.1.

```
2084 2056 2064 2065 2072 2059 2094 2073 2261 2236 2269 2159 2070 2084 2066 2080 2080 2071 2079 2092 2071 2085 2078 2077 2077
2039 1996 2140 1975 2073 1984 2077 1982 2055 1985 2075 2071 1991 2020 1987 2221 2140 2110 1992 2105 2003 2068 1985 2082 1985
2035 1972 2030 1974 2103 1976 2085 1998 2061 1993 2080 1985 2077 1986 2081 2044 2033 1975 2119 1986 2068 1996 2315 2113 2071
stepan@DESKTOP-L4K11TR:/tmp/compiler-opt/429.mcf/src$
```

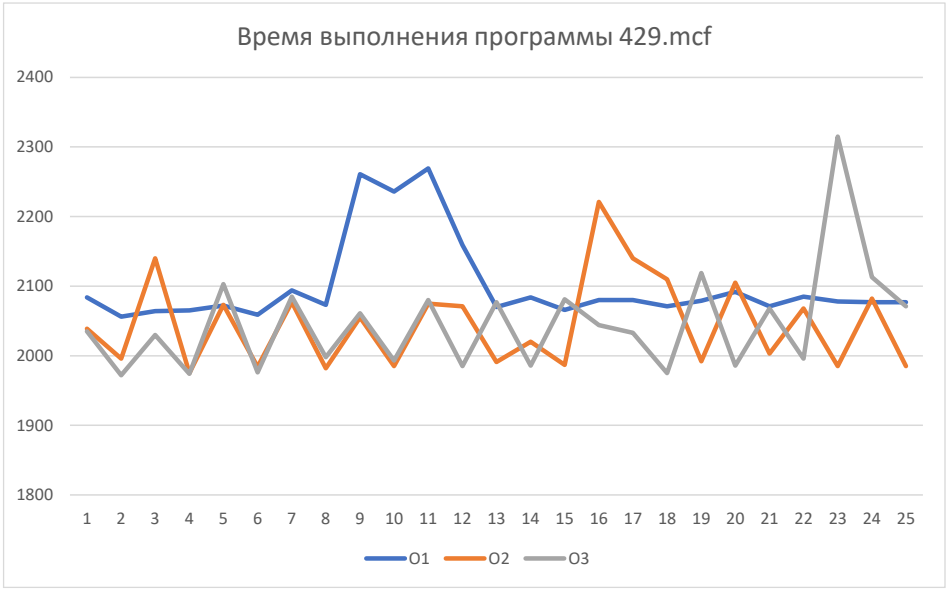

Рисунок 1

Статистические показатели производительности для каждого из уровней оптимизации приведены на рис. 1.1.

	Среднее	Дисперсия
O1	2100,08	3811,16
O2	2045,64	4146,99
O3	2046,24	5471,19

Рисунок 1.1

Затем программа запущена с разными флагами и их комбинациями. Графики приведены на рис. 2 и 3 соответственно.

```
2094 2065 2107 2070 2061 2096 2068 2127 2069 2075
2225 2355 2273 2147 2068 2072 2081 2048 2283 2054
2073 2049 2063 2058 2147 2058 2063 2071 2060 2160
2147 2140 2050 2096 2033 2064 2071 2106 2067 2052
2760 2251 2035 2083 2115 2082 2097 2073 2073 2072
2090 2051 2224 2067 2066 2076 2077 2092 2057 2049
2064 2054 2038 2056 2162 2066 2070 2061 2046 2323
2456 2068 2064 2057 2068 2091 2071 2072 2047 2110
stepan@DESKTOP-L4K11TR:~/win/PycharmProjects/compiler-opt/429.mcf/src$
```

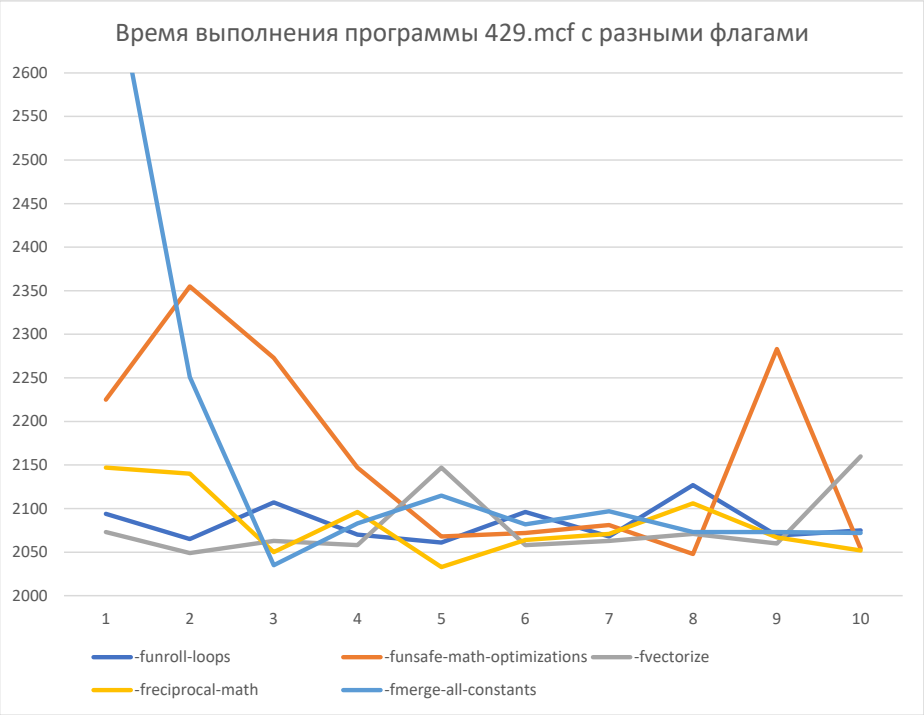

Рисунок 2

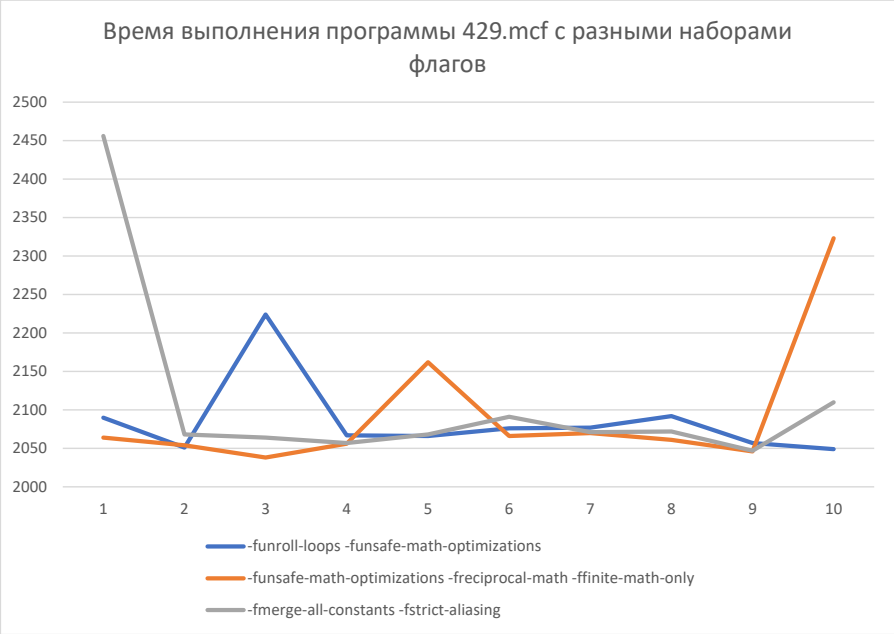

Рисунок 3

Лучшие найденные флаги: -funroll-loops, -fvectorized, -freciprocal-math.

Лучший набор флагов: -funsafe-math-optimization -freciprocal-math -ffinite-math-only.

Среднее значение по всем настройкам флагов (ожидаемая случайная производительность): 2107,5 мс.

2. Программа 519.lbm_r

Результаты измерений времени выполнения программы для уровней оптимизаций -O1, -O2, -O3 приведены на графике рис.4.

```
1101 1071 1042 1043 1038 1065 1044 1027 1053 1024 1035 1044 1040 1058 1060 1164 1217 1219 1210 1081 1057 1064 1030 1030 1052
1074 1077 1050 1037 1063 1050 1051 1059 1056 1088 1064 1044 1046 1058 1090 1061 1050 1061 1044 1055 1088 1100 1071 1059 1075
1085 1064 1106 1061 1066 1052 1046 1044 1043 1041 1051 1058 1043 1071 1059 1090 1085 1067 1051 1070 1052 1046 1068 1071 1064
stepan@DESKTOP-L4K11TR:/tmp/compiler-opt/519.lbm_r/src$ ~
```

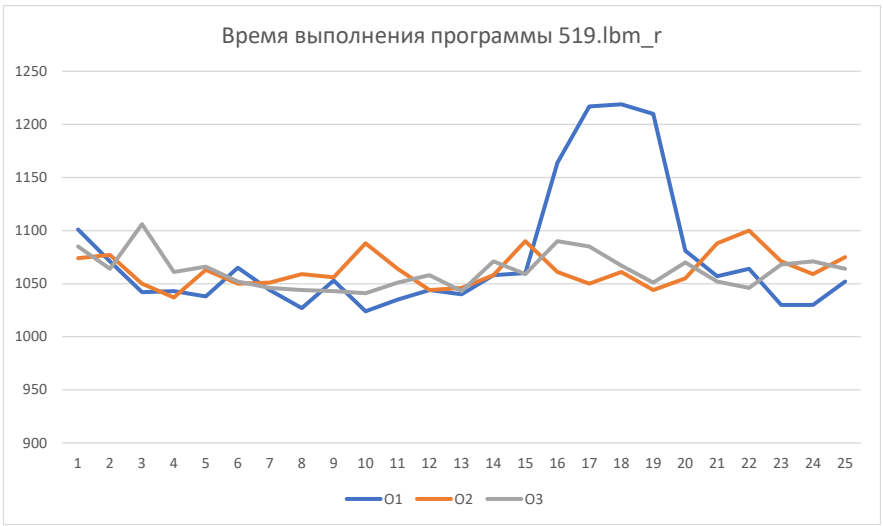

Рисунок 4

Статистические показатели производительности для каждого из уровней оптимизации приведены на рис. 4.1.

	Среднее	Дисперсия
O1	1074,76	3626,19
O2	1062,84	266,22333
O3	1062,16	273,14

Рисунок 4.1

Затем программа запущена с разными флагами и их комбинациями. Графики приведены на рис. 5 и 6 соответственно.

```
1156 1082 1112 1059 1076 1053 1056 1068 1065 1089
1232 1001 1255 1231 1219 1122 1050 1048 999 1002
1154 1090 1113 1069 1164 1095 1058 1072 1075 1078
1355 1077 1083 1069 1072 1080 1059 1091 1057 1075
1100 1082 1169 1121 1045 1089 1084 1058 1060 1069
1284 999 995 999 997 991 1017 1051 1091 1014
1039 1007 1030 1175 1018 992 1120 1024 1275 1305
1413 1073 1073 1060 1070 1056 1053 1071 1067 1056
stepan@DESKTOP-L4K11TR:~/win/PycharmProjects/compiler-opt/519.lbm_r/src$
```

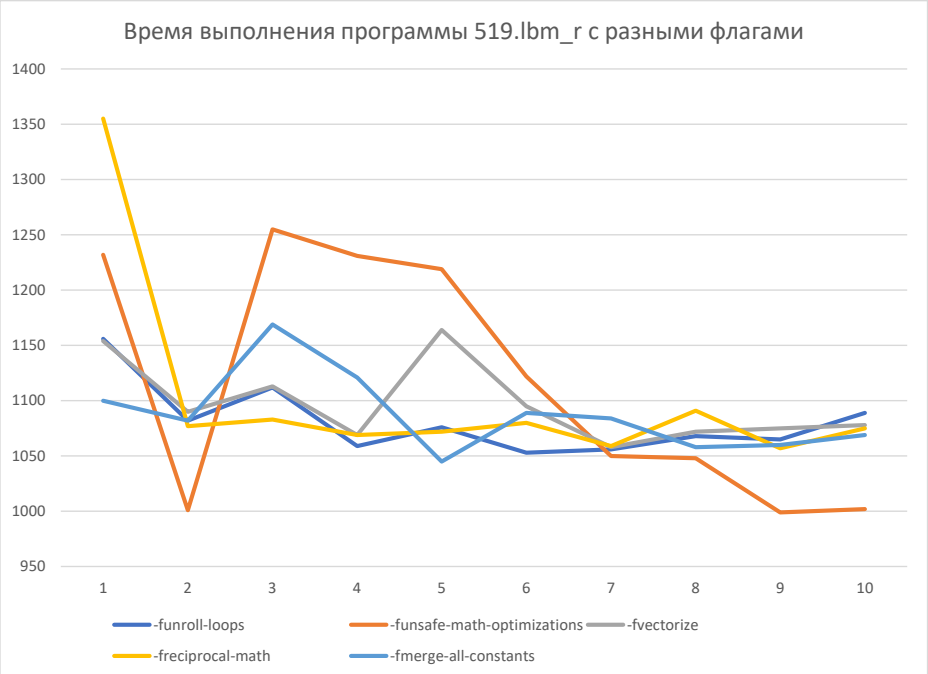

Рисунок 5

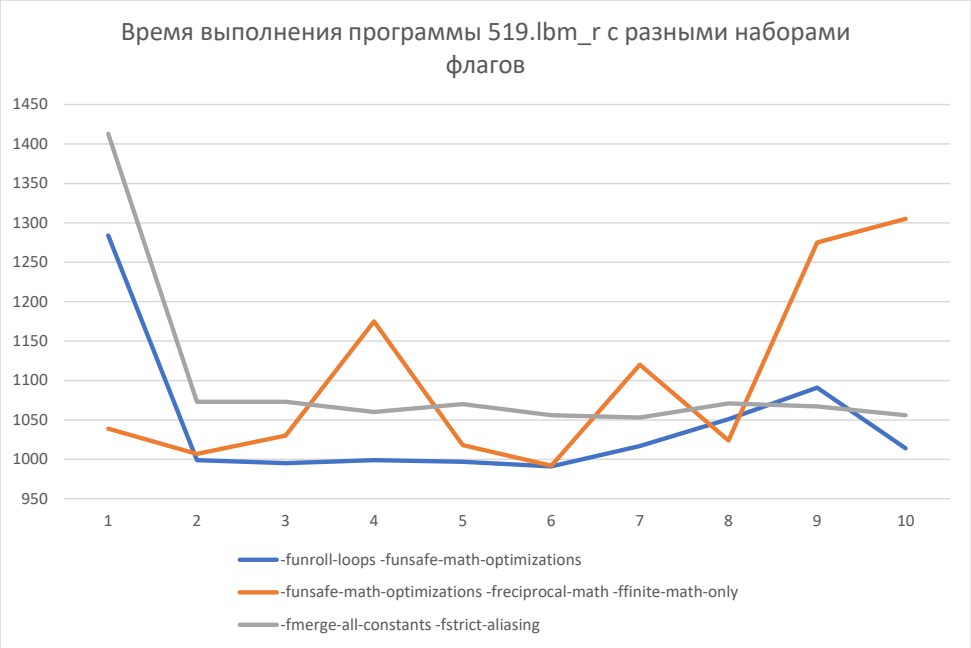

Рисунок 6

Лучшие найденные флаги: -funroll-loops, -fmerge-all-constants.

Лучший набор флагов: -funroll-loops -funsafe-math-optimizations.

Среднее значение по всем настройкам флагов (ожидаемая случайная производительность): 1090,66 мс.

3. Программа 531.deepsjeng_r

Результаты измерений времени выполнения программы для уровней оптимизаций -O1, -O2, -O3 приведены на графике рис.7.

```
8121 8003 8494 7724 7725 7742 7849 7712 7727 8396 8486 7723 7720 7739 7714 7737 7742 8208 8162 7987 7710 7717 7741 7698 7708
8233 8173 7809 7717 7743 18890 18589 15454 13873 13366 12577 13261 12778 12711 12218 10890 12505 11953 11197 11348 11119 11016 11996 11906 11350
11740 11659 12520 11757 8757 9358 8985 8869 8943 9382 8338 7753 8031 7720 7736 7716 7737 8130 8473 7747 7762 7730 7749 7754 7730
stepan@DESKTOP-L4K11TR:/tmp/compiler-opt/531.deepsjeng_r/src$
```

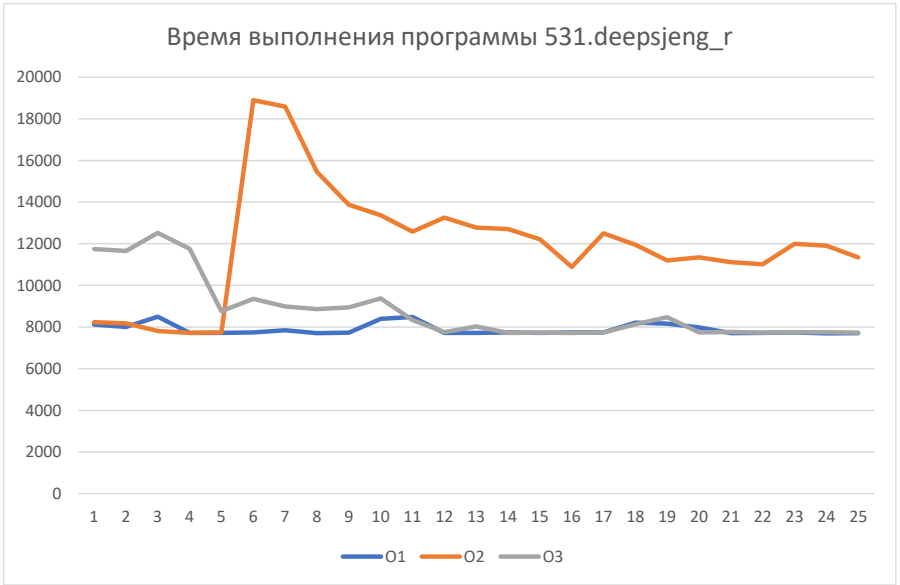

Рисунок 7

Статистические показатели производительности для каждого из уровней оптимизации приведены на рис. 7.1.

	Среднее	Дисперсия
O1	7891,4	69736,75
O2	11946,9	8283496,3
O3	8803,04	2243583,5

Рисунок 7.1

Затем программа запущена с разными флагами и их комбинациями. Графики приведены на рис. 8 и 9 соответственно.

```
8091 7770 7761 7756 7775 7744 7757 7757 8351 7745
7940 7843 7776 7805 7782 8242 7976 7824 7747 7741
7945 7746 7838 8395 7768 7812 7783 7782 7840 7789
7951 8388 7765 7892 7739 7782 7759 7747 7762 8454
7994 7725 7731 7743 7753 7760 7774 8387 7798 7862
7919 7815 7733 7784 8097 8176 7787 7752 7755 7727
7992 7810 8530 7725 7812 7751 7734 7753 7737 7755
10710 10880 10258 12029 11144 14467 11559 12901 12477 13172
```

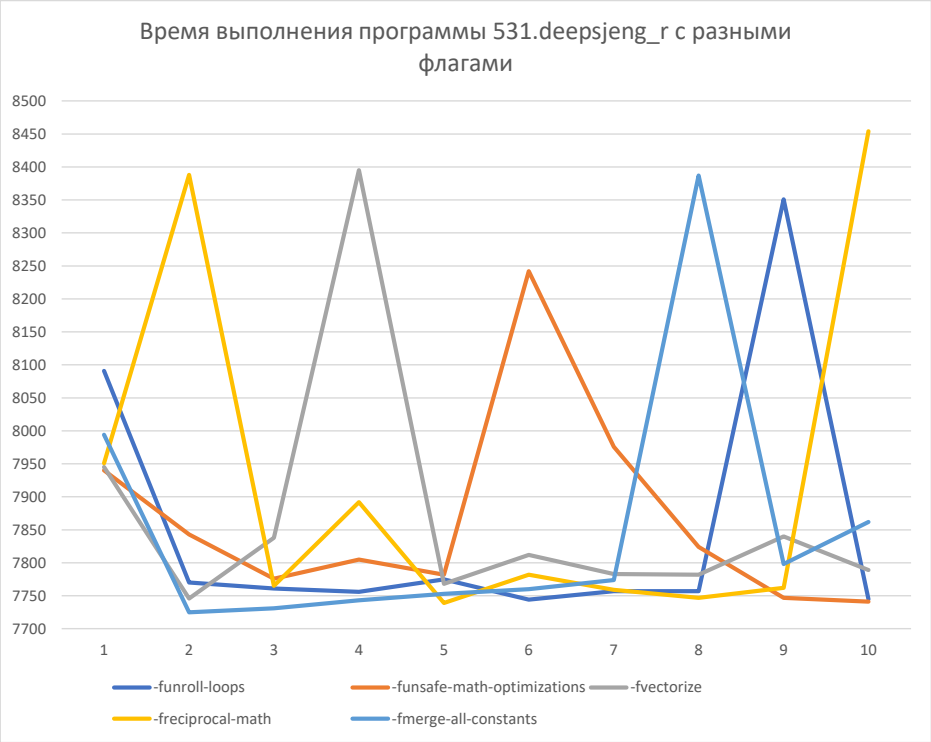

Рисунок 8

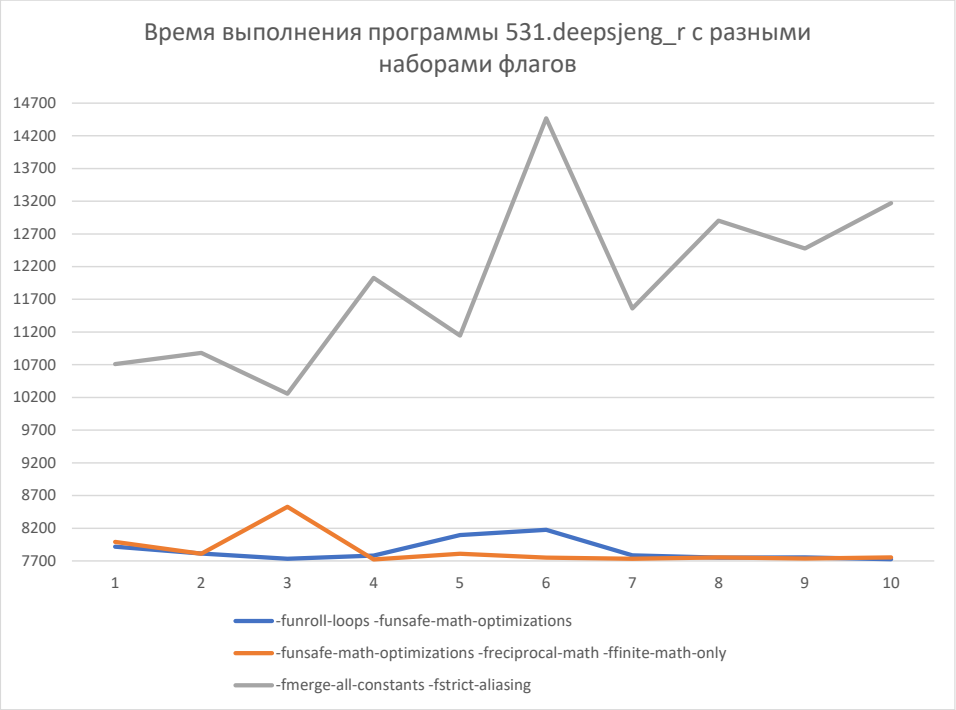

Рисунок 9

Лучшие найденные флаги: -funsafe-math-optimizations.

Лучший набор флагов: -funroll-loops -funsafe-math-optimizations.

Среднее значение по всем настройкам флагов (ожидаемая случайная производительность): 8379,85 мс.

Средние значения результата и дисперсии, вычисленные для всех программ, относительно различных флагов, приведены на рис.10.

	429.mcf		519.lbm_r		531.deepsjeng_r	
	Среднее	Дисперсия	Среднее	Дисперсия	Среднее	Дисперсия
-funroll-loops	2083,20	473,73	1081,60	998,93	7850,70	41935,34
-funsafe-math-optimizations	2160,60	12949,16	1115,90	11726,32	7867,60	23320,27
-fvectorize	2080,20	1547,29	1096,80	1313,51	7869,80	37132,40
-freciprocal-math	2082,60	1483,60	1101,80	8021,29	7923,90	73601,88
-fmerge-all-constants	2164,10	47180,77	1087,70	1304,46	7852,70	41822,23
-funroll-loops -funsafe-math-optimizations	2084,90	2606,77	1043,80	8106,18	7854,50	25426,72
-funsafe-math-optimizations -freciprocal-math -ffinite-math-only	2094,00	7655,33	1098,50	13342,94	7859,90	61634,77
-fmerge-all-constants -fstrict-aliasing	2110,40	15051,38	1099,20	12214,62	11959,70	1707098,23

Рисунок 10

АНАЛИЗ РЕЗУЛЬТАТОВ

По результатам, полученным экспериментально, можно сделать следующие выводы:

1. Результаты серии измерений с одинаковой конфигурацией отличаются, вследствие влияния фоновых служб на загрузку процессора;
2. По той же причине результаты, полученные при разных уровнях компиляции, не всегда совпадают с теоретическими. Некоторые измерения показали при оптимизации -O3 результат хуже, чем при -O1;
3. Разные программы показали большую производительность при разных флагах. Это связано с тем, что флаги имеют конкретное назначение и оказывают полезное влияние при конкретных задачах;
4. В целом, повышение уровня оптимизации положительно сказывается на производительности.

ЗАКЛЮЧЕНИЕ

В результате выполнения работы были исследованы методы компиляторной оптимизации программ в Clang/LLVM. Проведен эксперимент, результаты которого отражают эффективность различных уровней оптимизации и применения дополнительных флагов (опций).