Московский физико-технический институт (государственный университет)
Факультет радиотехники и кибернетики

Кафедра информатики и вычислительной техники

Выпускная квалификационная работа бакалавра

Ускорение работы оптимизирующего компилятора «Эльбрус» за счёт усовершенствования алгоритма вызова цикловых оптимизаций без ухудшения производительности

Выполнил: Алёхин А. И. , 213 гр.

Научный руководитель: Линчик М. И.

Описание проблемы

- Длительное время работы алгоритма цикловых оптимизаций, обусловленное большим количеством применяемых оптимизаций и перестроений аналитических структур вследствие нерационального использования оптимизаций.
- Невозможность искусственного ограничения на использование оптимизаций, поскольку это может привести к снижению производительности исполняемого кода.

Постановка задачи

Усовершенствовать алгоритм применения цикловых оптимизаций с целью его ускорения при сохранении исходного уровня производительности

Подход к решению задачи

- Изучить основные цикловые оптимизации: реализацию и контекст их применения.
- Изучить алгоритм вызова цикловых оптимизаций.
- Проанализировать работу алгоритма вызова цикловых оптимизаций на задачах пользователей и пакетах SPEC CPU 2006.
- Усовершенствовать существующий алгоритм на основе проведённого анализа.
- Провести оценку эффективности и надёжности алгоритма на пакетах SPEC CPU 2006.

Основные оптимизации

- Оптимизация выноса инвариантных вычислений из цикла (Invariants up - INVUP)
- Оптимизация перестановки циклов в гнезде
- Оптимизации выноса инвариантного условия из цикла
- Оптимизации разбиения цикла условиям
- Оптимизация свёртки гнезда циклов в 1 цикл
- Разделение гнезда циклов
- Вынесение нескольких итераций цикла в ациклический участок кода

- «Корректирующие» оптимизации:
  - удаление избыточных вычислений
  - анализ зависимостей внутри циклов (Loop Dependence Analysis - LDA)
  - канонизация потока управления

Результаты изучения оптимизаций

- LDA является одним из наиболее затратных по времени работы.
- Наличие результатов LDA требуется почти всем оптимизациям.
- Результаты LDA становятся не валидными после применения всех цикловых оптимизаций.

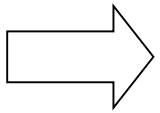
- Оптимизация удаления избыточных вычислений применяется ко всей процедуре сразу после работы всех остальных оптимизаций.

Результаты анализа на задачах

- Оптимизации INVUP и удаления избыточных вычислений являются одними из самых часто используемых.
- Контекст для работы оптимизации INVUP почти никогда не создаётся другими оптимизациями.
- Контекст для работы оптимизации удаления избыточных вычислений содержится только в циклах, к которым были применены другие оптимизации.
- Контекст для работы оптимизации перестановки циклов почти всегда содержится только в циклах, которые были обработаны и максимально упрощены другими оптимизациями.

Контекст применения оптимизации INVUP

```
for (int i = 0; i < N; i++)
{
        x = y + z;
        a[i + x*x] = 6*i;
        if (cond)
        {
                foo(const);
        }
}
```

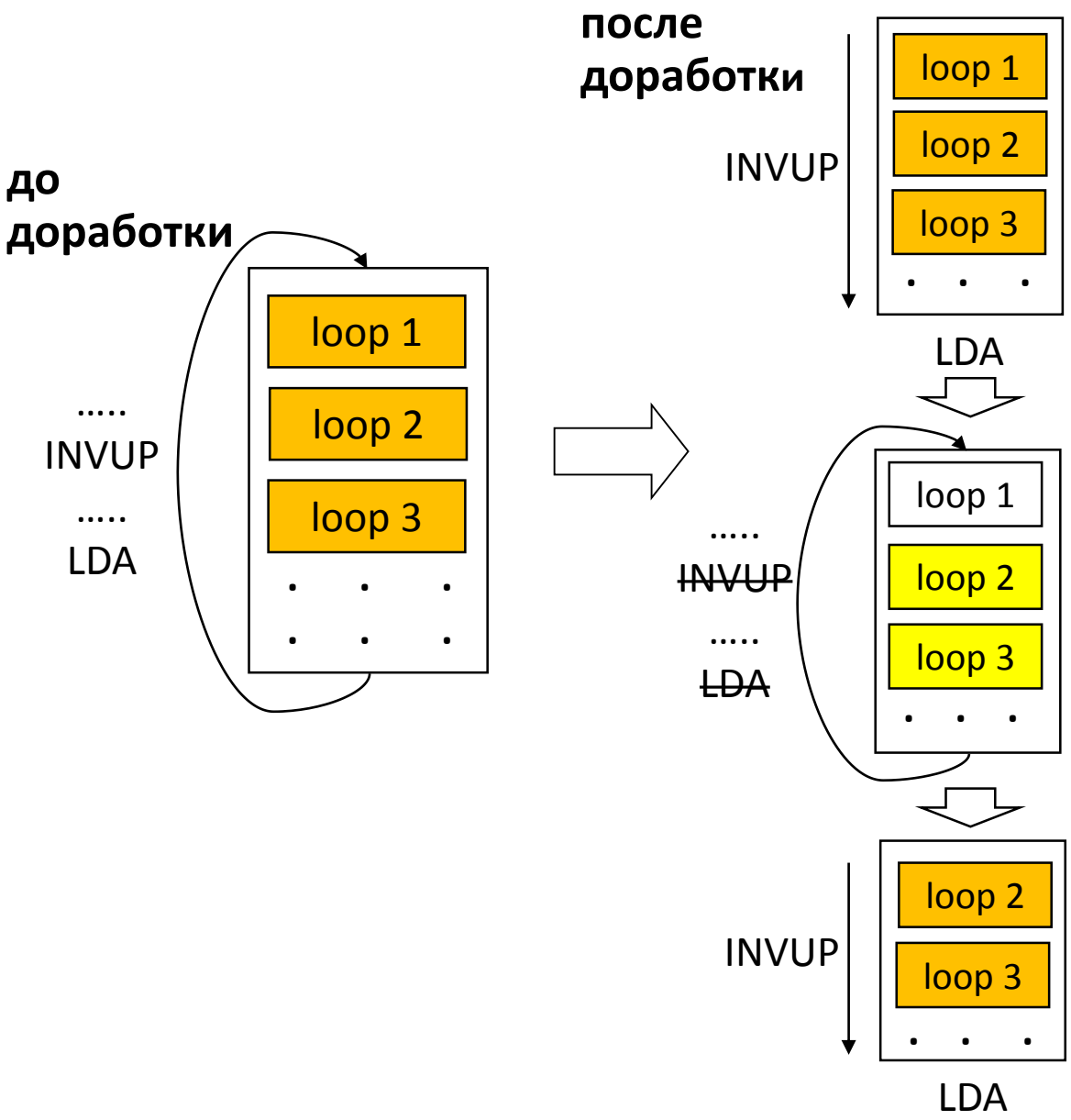


```
x = y + z;
t = x*x;

for (int i = 0; i < N; i++)
{
        a[i + t] = 6*i;
        if (cond)
        {
                foo(const);
        }
}
```

- Большая часть контекста для работы оптимизации INVUP доступна до применения других оптимизаций.
- Существует ряд оптимизаций, которые могут в редких случаях создавать контекст для работы оптимизации INVUP.

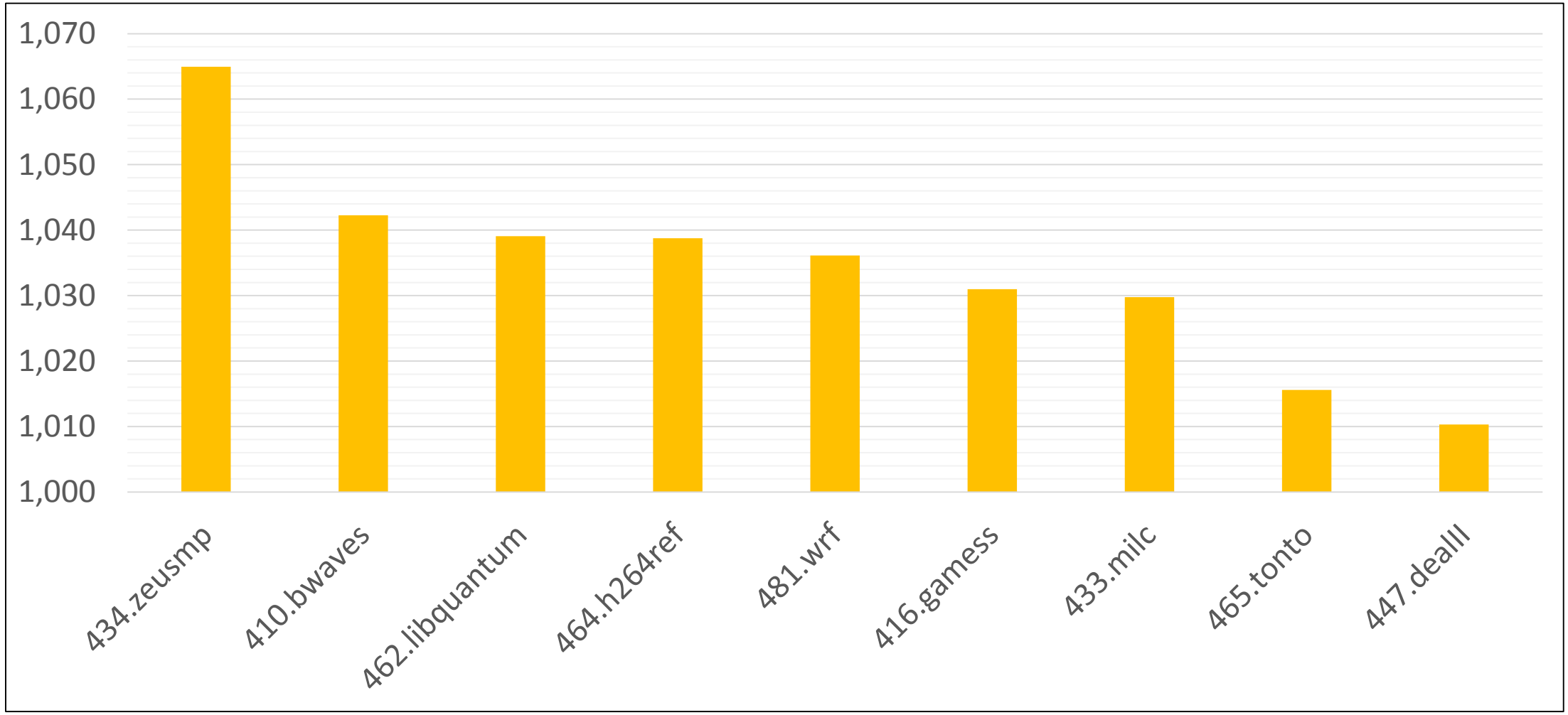
Сокращение количества вызовов LDA



1. Применяем INVUP ко всем циклам и выполняем LDA однократно.
2. При применении к циклу оптимизаций, которые создают контекст для INVUP, помечаем этот цикл и все созданные циклы.
3. Применяем INVUP ко всем помеченным циклам и выполнить LDA.

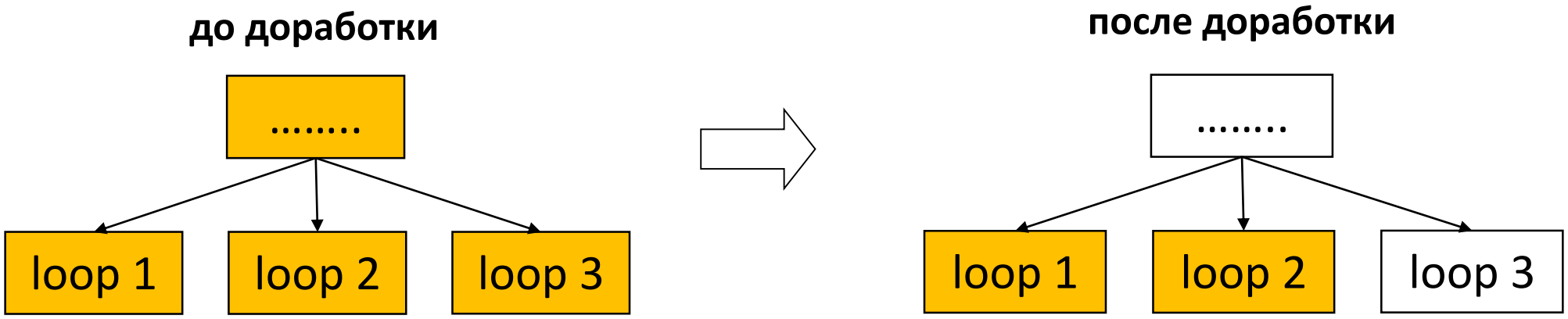
Количество вызовов LDA существенно снижается, так как создание контекста для INVUP в процессе работы других оптимизаций происходит редко.

Результаты сокращения количества вызовов LDA



Ускорение компиляции пакета 2,0 % , некоторых тестов до 6 %

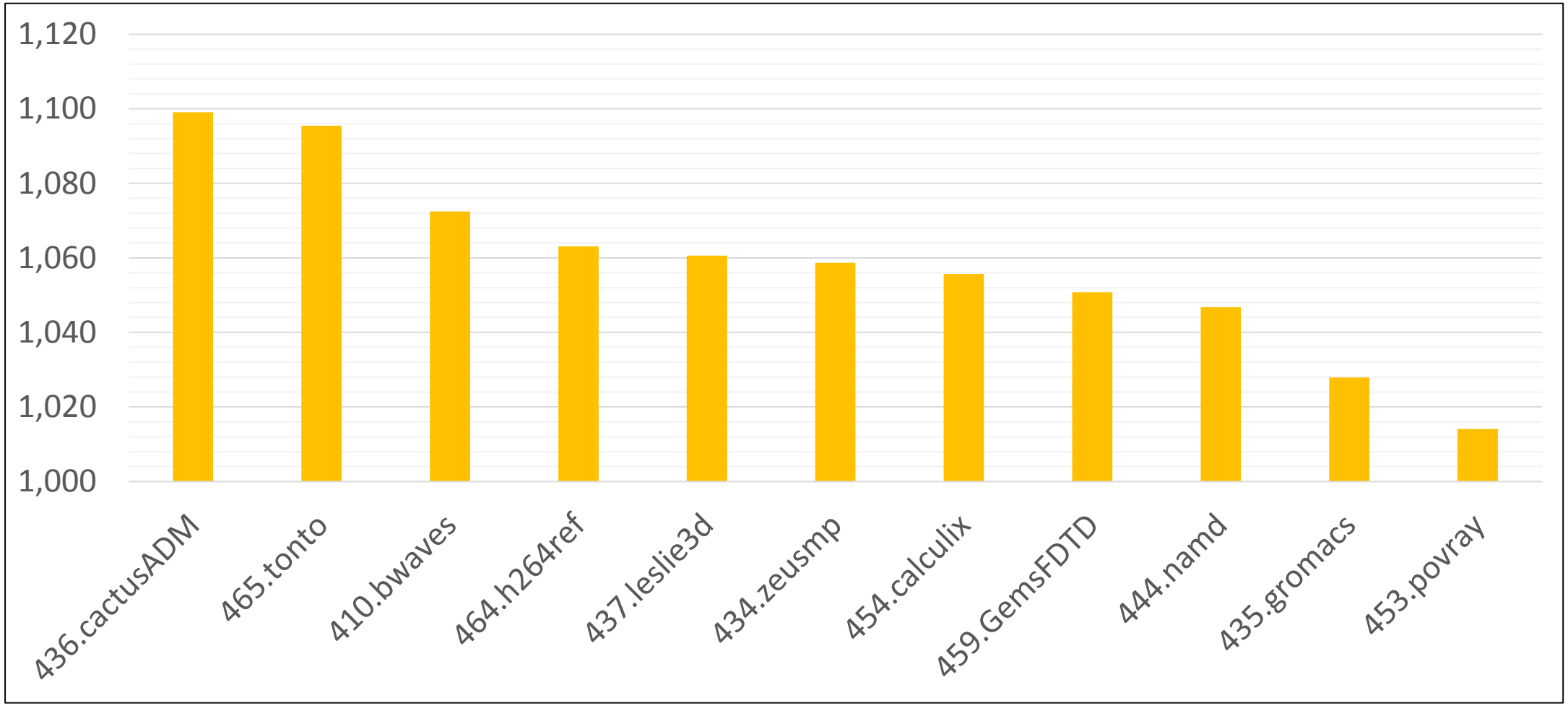
Уменьшение времени работы оптимизации удаления избыточных вычислений



1. Помечаем все циклы, к которым были применены какие-либо оптимизации, а также циклы, созданные применившимися оптимизациями.
2. Применяем оптимизацию удаления избыточных вычислений только к тем циклам, которые были помечены.

Результат: существенное снижение времени работы

Результаты уменьшения времени работы оптимизации удаления избыточных вычислений



Ускорение компиляции пакета 2,1 % , некоторых тестов до 10%

Избавление от лишних использований оптимизации перестановки циклов

- Оптимизацию перестановки циклов целесообразно применять только к гнезду циклов, которое уже было обработано всеми другими оптимизациями

Решение:

1. Отказаться от использования данной оптимизации на основном проходе алгоритма.
2. Применять данную оптимизацию только отдельным проходом сразу после алгоритма цикловых оптимизаций.

Результат: упрощение логики работы алгоритма

Результаты работы

- Была проведена доработка алгоритма цикловых оптимизаций.
- Проведены замеры времени компиляции и производительности на пакетах SPEC CPU 2006, показавшие:
  - ускорение компиляции до 10%
  - сохранение исходного уровня производительности
- Усовершенствованный алгоритм внедрён в оптимизирующий компилятор «Эльбрус».