Московский физико-технический институт (государственный университет)
Факультет радиотехники и кибернетики
Кафедра информатики и вычислительной техники

Магистерская диссертация

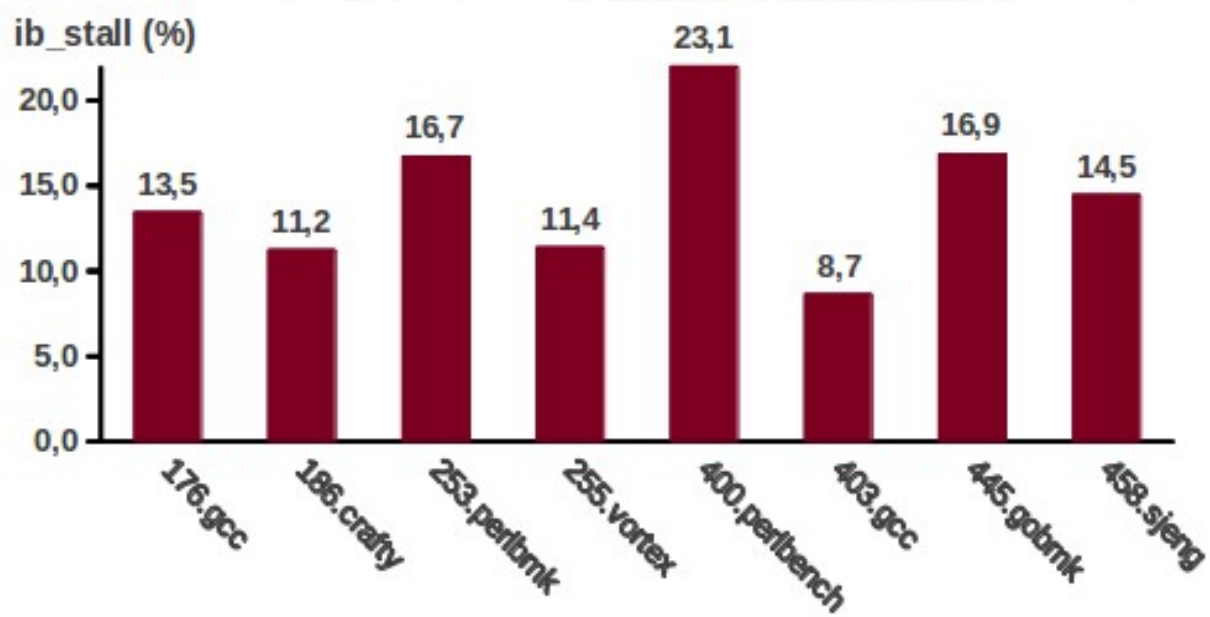
Исследование методов программной предварительной подкачки кода для микропроцессоров архитектуры Эльбрус

Студент: Степнов Денис, группа 816

Научный руководитель: к.ф.-м.н. Нейман-Заде М.И.

Проблематика

- Возрастающая доля блокировок по отсутствию кода во время исполнения приложений при усложнении потока управления
  - 4,97 % на spec2000
  - 5,50 % на spec2006
  - На отдельных приложениях достигает 14-23 %

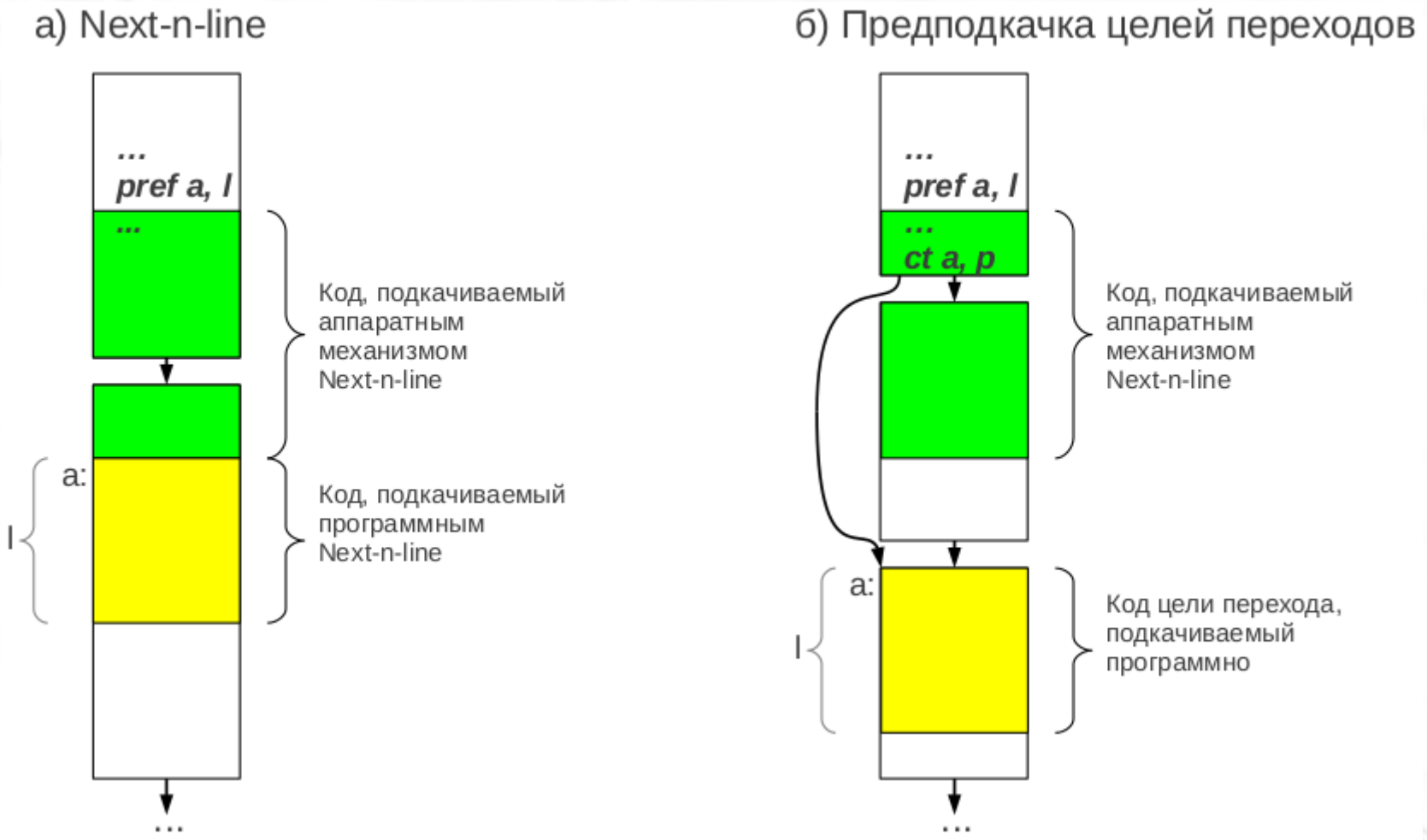


Цель работы

- Реализовать оптимизацию, выполняющую поиск плохо подкачиваемого кода и размещение инструкций его предподкачки
- Определить для неё место в линейке оптимизаций компилятора lcc
- Провести верификацию на стандартных пакетах тестов
- Провести оценку эффективности оптимизации на тестах из пакетов SPEC

Методы оптимизации

- Из всех возможных вариантов были выбраны:



Поддержка оптимизации в архитектуре Эльбрус

- Система команд:
  - Операция подготовки передачи управления disp
    - Выполняет подготовку перехода — исполнение всех фаз конвейера до декодирования
    - Выполняет предварительную подкачку кода, размером до 3-х строк буфера инструкций (768 байт)
  - Операция предподкачки кода pref (поддерживается, начиная с 3-ей версии системы команд)
    - Выполняет подкачку кода размером 1-2 строки буфера инструкций (256-512 байт)
    - Имеет низкий приоритет по сравнению с disp

Поддержка оптимизации в архитектуре Эльбрус

- Три регистра ctpr
  - Подготовка переходов: операция disp записывает в ctpr адрес, по которому начинается подготовка

```
disp addr → ctpr1
...
...
branch ctpr1
```

  - Программная предподкачка: возможна в тех участках кода, где существуют ctpr, не используемые для подготовок переходов, т. е. мёртвые ctpr

Процедура оптимизации

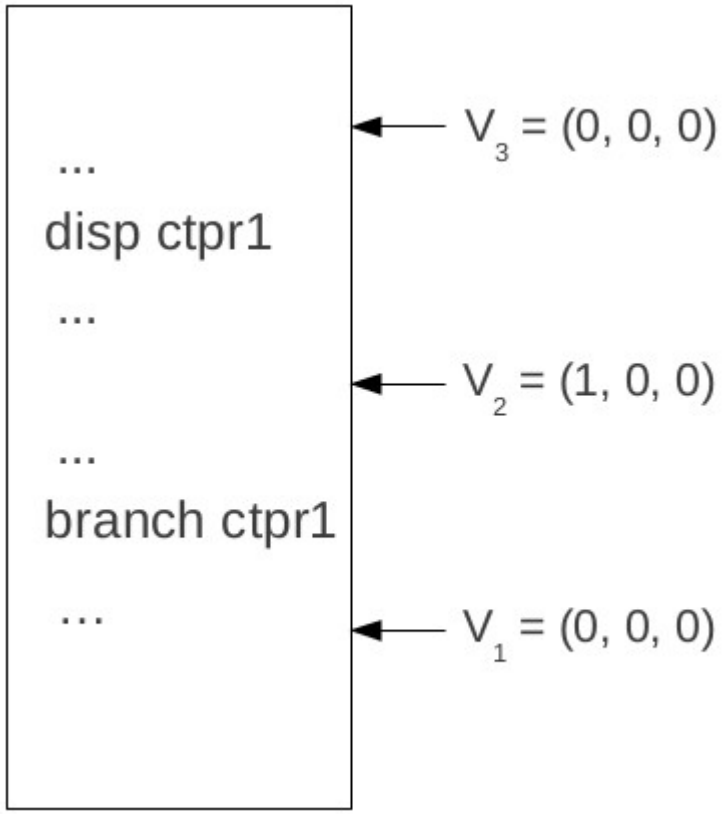
1. Анализ жизни регистров ctpr
2. Поиск целей передачи управления и оценка необходимости их предподкачки
3. Размещение операций предварительной подкачки:
    1) Целей вызовов процедур
    2) Последовательного кода
    3) Целей локальных переходов

Анализ жизни регистров ctpr

- Глобальная часть анализа
  1. Для дуг управляющего графа создаётся битовый массив EdgeLiveInfo[3], описывающий передачу живых регистров ctpr. Массивы инициализируются нулями.
  2. Обход CFG по технике Worklist снизу вверх для пропагации информации о жизни регистров ctpr на все дуги. В порядке Postorder обходятся все узлы графа до тех пор, пока значения векторов EdgeLiveInfo не перестанут изменяться.

Анализ жизни регистров ctpr

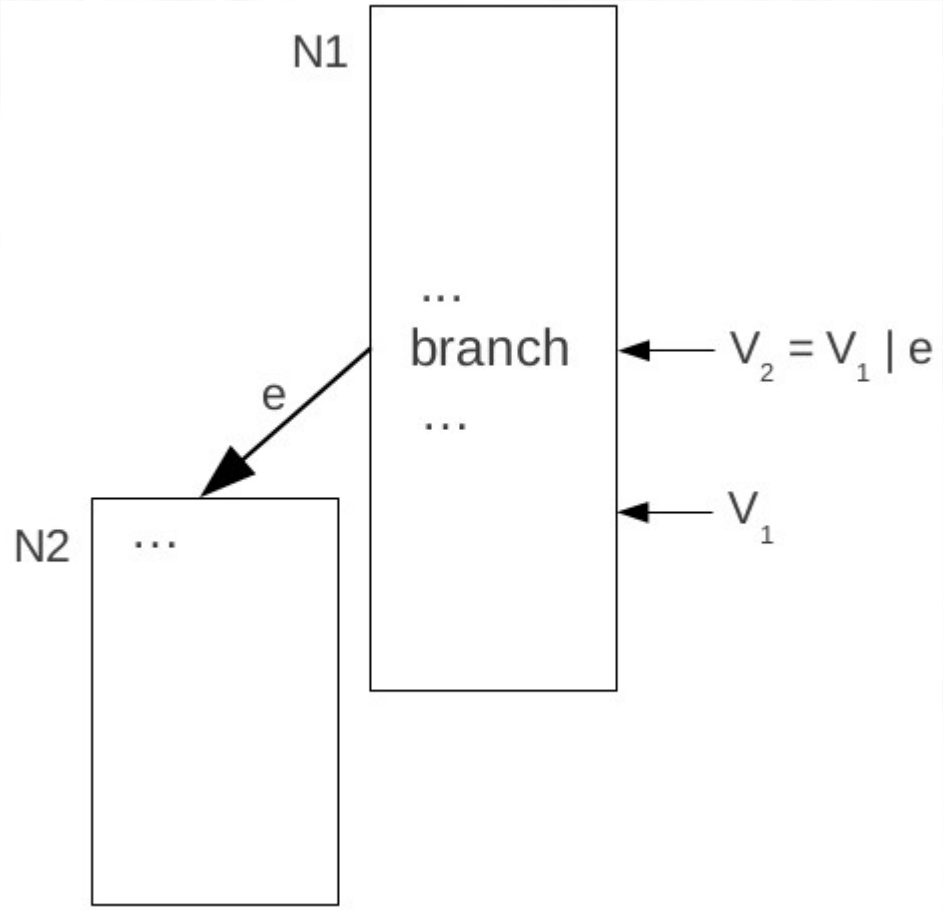
- Глобальная часть анализа



При обходе линейного участка, информация о жизни ctpr передаётся от конца к началу с помощью вектора обхода v[3], в котором биты принимают значение 1, при встрече чтения ctpr с соответствующим номером, и значение 0, при встрече записи.

Анализ жизни регистров ctpr

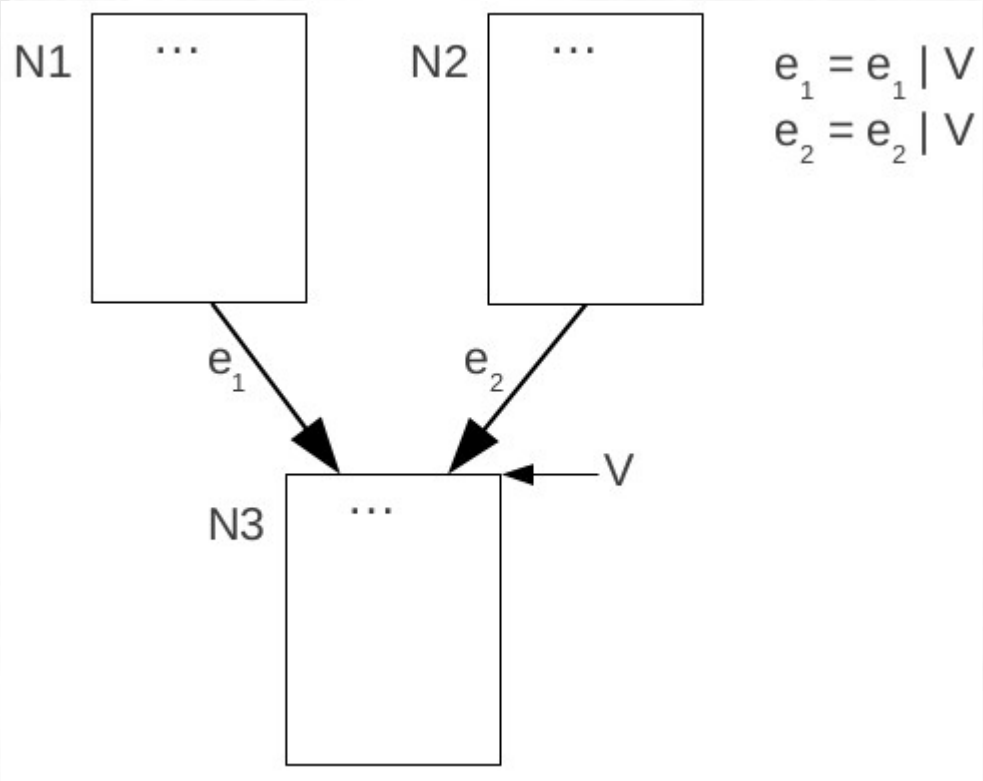
- Глобальная часть анализа



При встрече операции перехода на другой линейный участок, выполняется дизъюнкция v с вектором EdgeLiveInfo дуги, соответствующей переходу.

Анализ жизни регистров ctpr

- Глобальная часть анализа



При переходе от одного линейного участка к другому, меняются значения векторов EdgeLiveInfo дуг, входящих в пройденный перед этим линейный участок путём их дизъюнкции с v. В список Worklist заносятся те линейные участки, из которых исходят дуги с изменёнными EdgeLiveInfo.

Анализ жизни регистров ctpr

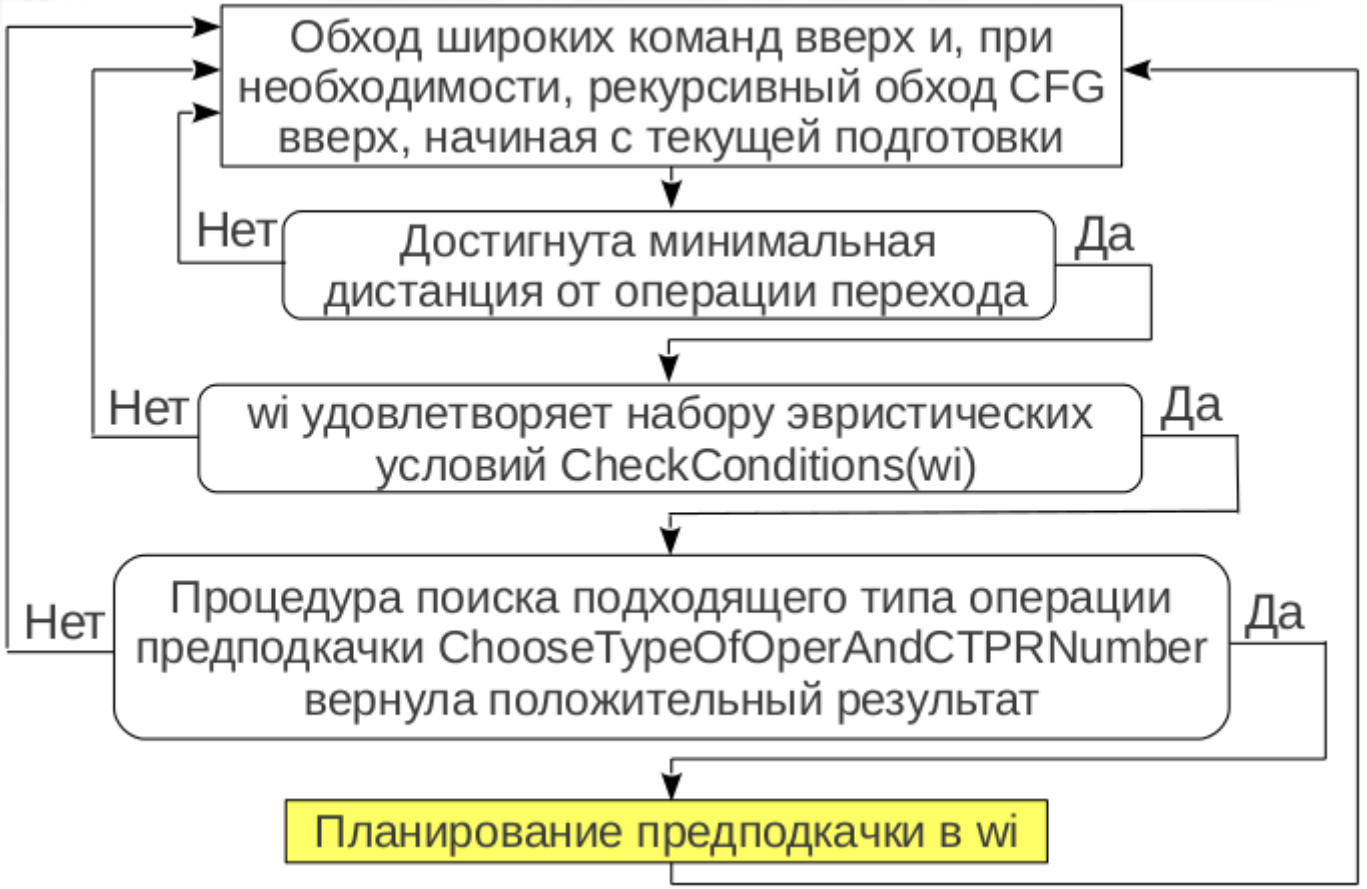
- Локальная часть анализа
  - Обход каждого линейного участка с вектором v, который инициализируется операцией дизъюнкции векторов EdgeLiveInfo всех исходящих дуг.
  - При встрече чтения регистра ctpr, соответствующий бит вектора v принимает значение 1, при встрече записи — значение 0.
  - В момент изменения значения битов вектора v строится диапазон жизни станка в текущем линейном участке.

Поиск плохо подкачиваемых целей переходов

1. Обход всех операций branch и call процедуры и создание структур TargetInfo для каждой цели. В структуре содержатся поля:
   - label — метка на цель
   - disp_list — список всех операций disp, обращённых на эту цель
   - addr — оценочный адрес цели (для локальных целей)
2. «Прореживание» локальных близко расположенных целей
3. Обход disp_list каждого TargetInfo и удаление тех подготовок, которые доминируются другими из того же списка
4. Удаление из списков disp_list операций disp, расположенных на достаточно большой дистанции от своих пользователей для предподкачки кода

Размещение инструкций предподкачки целей переходов

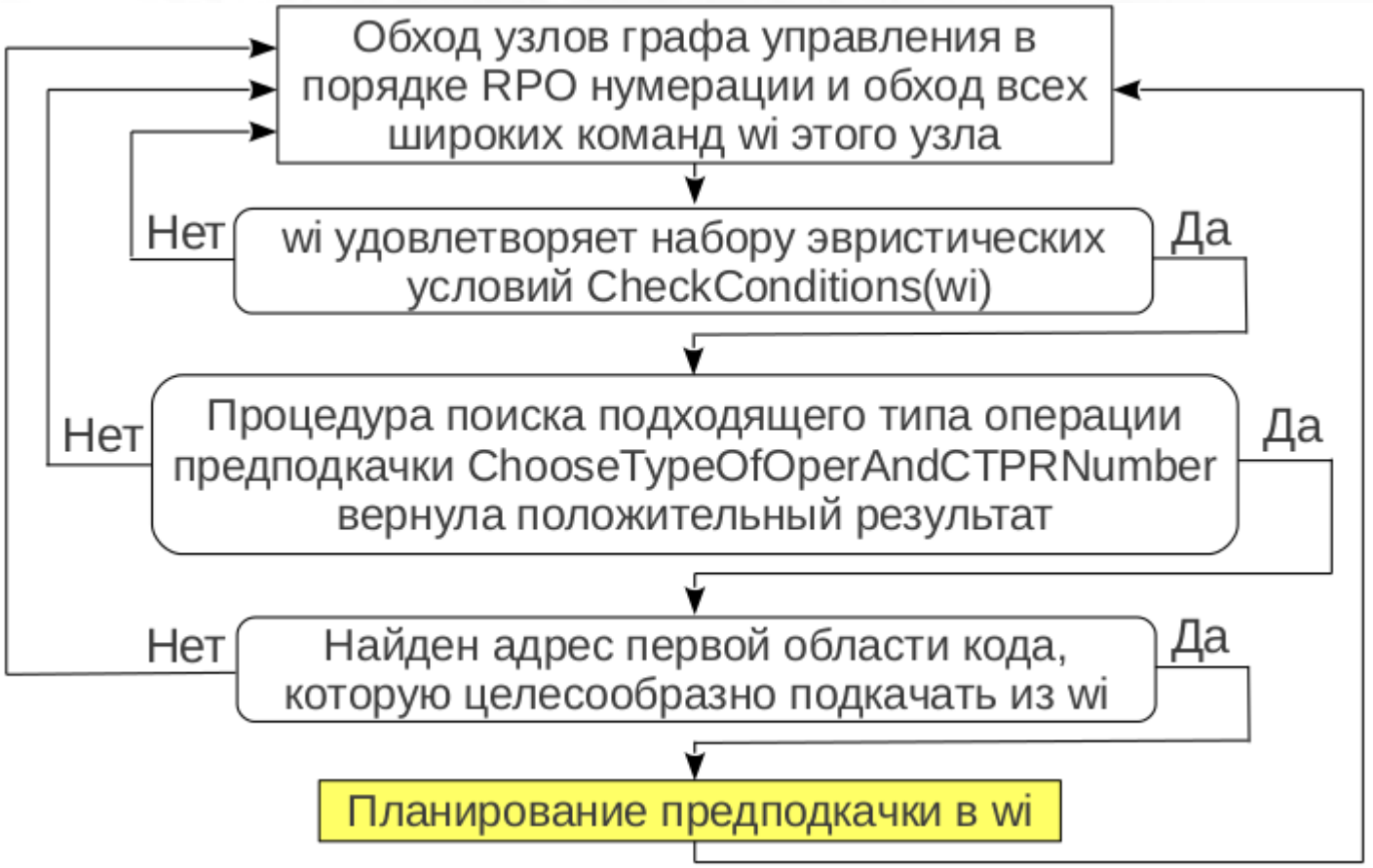
- Обход списков подготовок всех TargetInfo и вызов для них процедуры планирования SchedPrefetchRec:



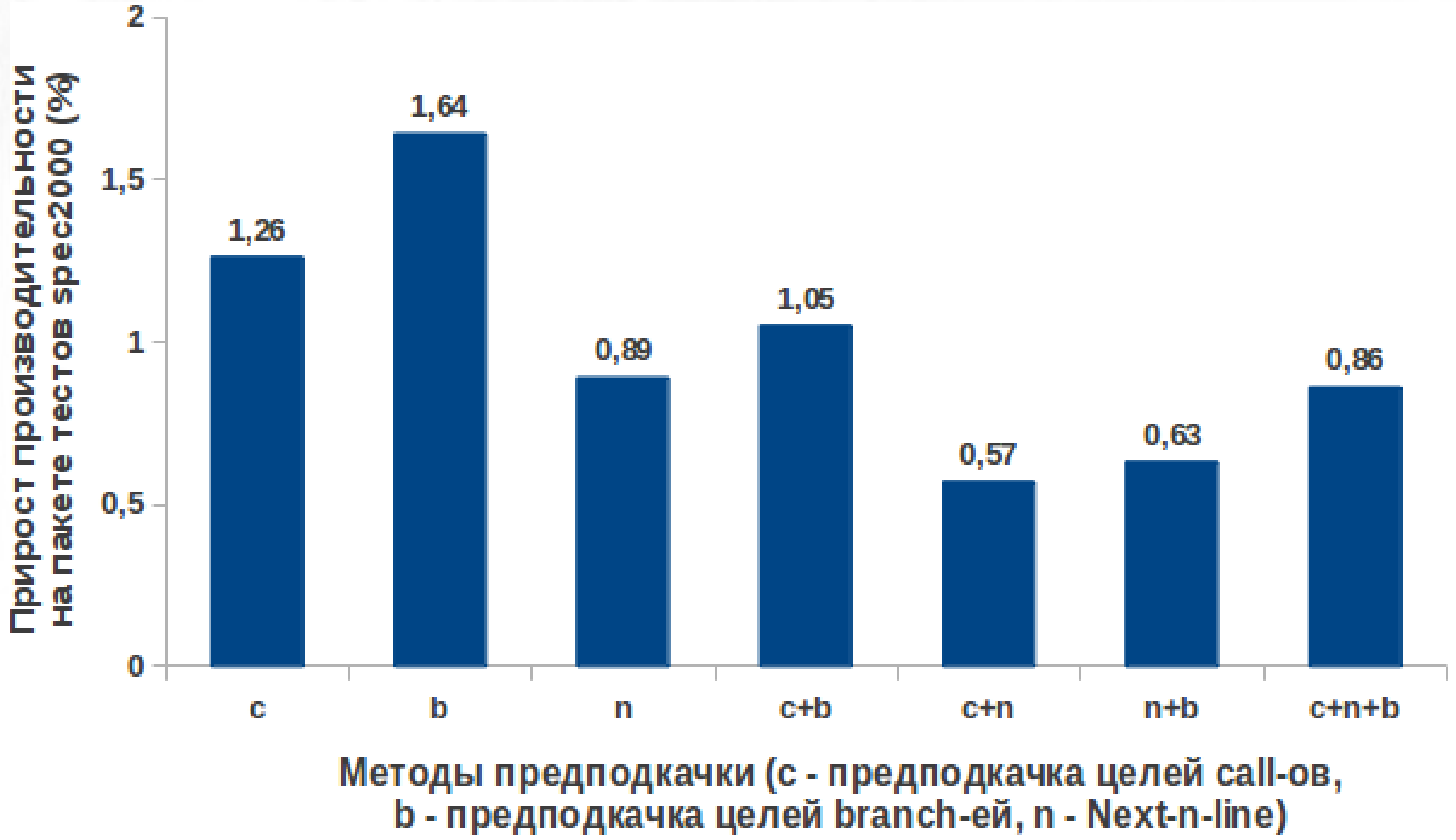
Размещение инструкций предподкачки целей переходов

- CheckConditions(wi)
  - Проверка наличия «дыры» в wi
- ChooseTypeOfOperAndCTPRNumber
  - Поиск регистра ctpr, подходящего для предварительной подкачки из текущей широкой команды. В случае отсутствия подходящего ctpr происходит оценка целесообразности использования pref. Критерии, влияющие на работу этой процедуы:
    - Наличие свободных регистров ctpr
    - Близость других подготовок
    - Близость инструкций call

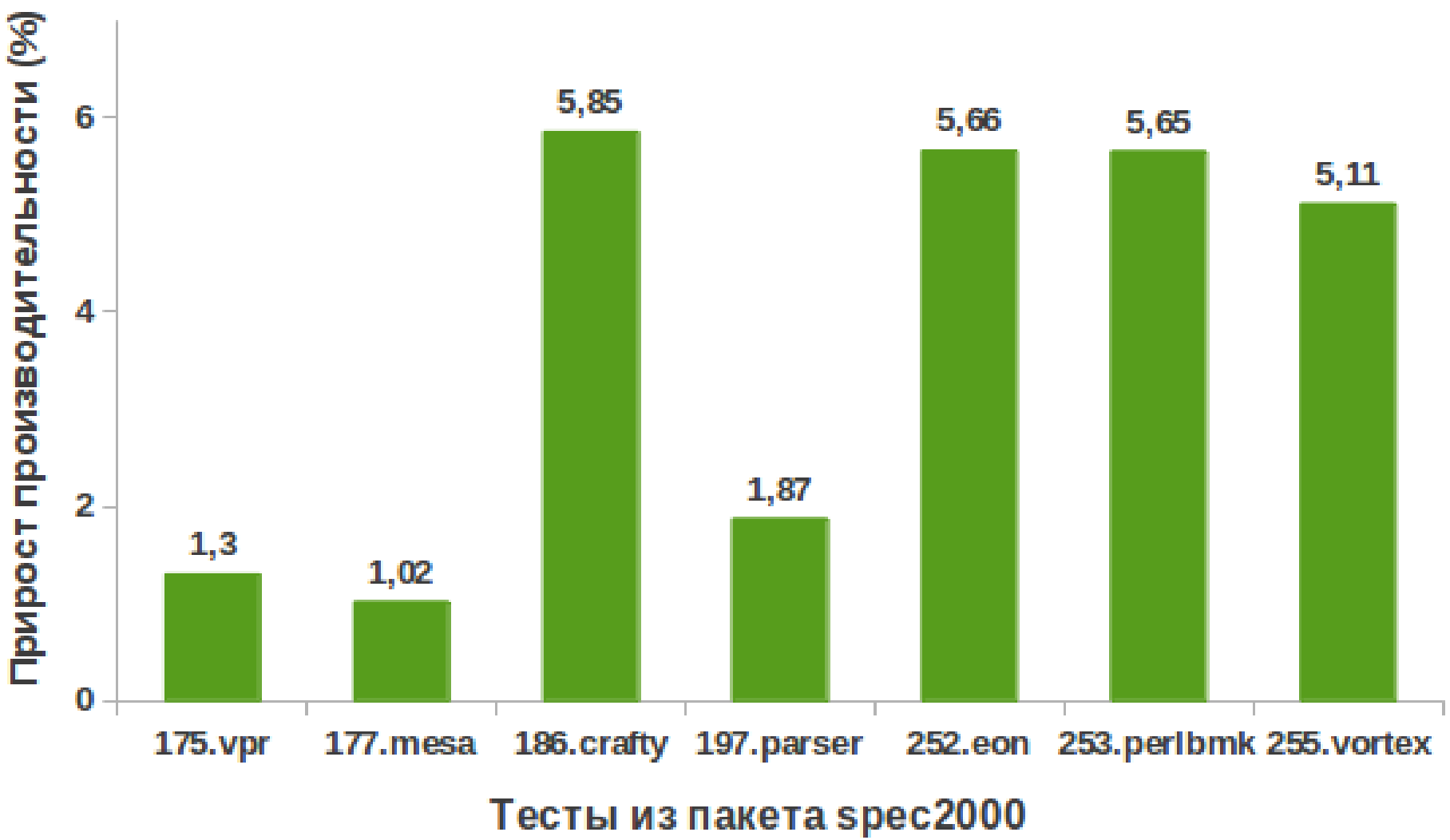
Размещение инструкций предподкачки последовательного кода



Экспериментальные результаты



Экспериментальные результаты



Результаты

- Оптимизация реализована и внедрена в линейку оптимизаций компилятора Icc:
  - Применение на 3-м уровне оптимизации в последней фазе перед кодогенерацией
- Проведена верификация
- Получен прирост производительности на пакете тестов spec2000: 1,64 %
- Время компиляции увеличилось на 0,24 %
- Объём кода: 5570 строк