УДК 004.4'416

На правах рукописи

Иванов Дмитрий Сергеевич

Комплексная технология распределения регистров и планирования инструкций в оптимизирующем компиляторе вычислительных комплексов семейства «Эльбрус»

05.13.11 – Математическое и программное обеспечение вычислительных машин, комплексов и компьютерных сетей

АВТОРЕФЕРАТ

диссертации на соискание ученой степени кандидата технических наук

Москва – 2011

Работа выполнена на кафедре «Информатика и вычислительная техника»
Московского физико-технического института
(государственного университета)

Научный руководитель:	кандидат технических наук Волконский Владимир Юрьевич
Официальные оппоненты:	доктор технических наук, Дроздов Александр Юльевич кандидат технических наук Останевич Александр Юрьевич
Ведущая организация:	Институт системного программирования РАН

Защита состоится «__» _____ 2012 г. в __ час. __ мин. на заседании диссертационного совета *Д 409.009.01* при **ОАО «Институт электронных управляющих машин им. И.С.Брука»** , расположенном по адресу: 119334, г.Москва, ул.Вавилова д.24

С диссертацией можно ознакомиться в библиотеке **ОАО «ИНЭУМ им. И.С.Брука»** .

Автореферат разослан «__» _____ 201_ г.

Ученый секретарь
диссертационного совета,
кандидат технических наук, профессор Красовский В.Е.

Общая характеристика работы

Актуальность работы. Определяющее значение в проектировании высокопроизводительных средств вычислительной техники, построенных на базе микропроцессорных систем, имеет введение оптимизаций, которые выполняются на этапе компиляции программ и, следовательно, не требуют усложнения оборудования. Ключевыми оптимизациями этого рода являются *распределение регистров* и *планирование инструкций*, выполняемые применительно к компилируемой процедуре.

Первая из них связана с решением одной из наиболее значимых задач, сопутствующих разработке микропроцессора, - сокращением времени доступа к памяти. В этом смысле большое значение имеет организация взаимодействия между процессорами и подсистемой быстрой регистровой памяти. С ростом объема ее стоимость заметно увеличивается, поэтому в компиляторах выполняется оптимизация, предпочтительным образом проектирующая виртуальные регистры процедуры, количество которых произвольно, на ограниченное число архитектурных регистров процессора.

Вторая оптимизация обеспечивает такой порядок исполнения команд процессора, который позволяет в течение всего процесса вычислений максимально задействовать доступные аппаратные ресурсы.

В известных реализациях распределение регистров и планирование инструкций рассматриваются независимо и выполняются в различных фазах компиляции, в результате чего одна из фаз вынуждена работать с теми условиями, которые были навязаны другой. При этом, с одной стороны, распределение после планирования может столкнуться с дефицитом регистров, созданным за счет стремления к максимальной параллельности кода, с другой стороны, планирование будет ограничено зависимостями, неизбежно появляющимися в результате состоявшегося распределения архитектурных реги-

стров. Проблему можно решить таким совмещением двух оптимизаций, которое позволило бы учитывать и полезно использовать их влияние друг на друга, а следовательно повысить суммарную эффективность. Помимо реализации совмещенных процессов в соответствующих фазах компиляции, это требует адаптации и усовершенствования существующих, а также разработки новых сопутствующих оптимизаций, что позволяет говорить о единой технологии распределения регистров при планировании инструкций. Причем, в силу того, что объектом оптимизаций здесь является низкоуровневое промежуточное представление программы, она не зависит от языка программирования. Следует отметить, что воспроизводимые на практике реализации этой идеи в литературе фактически не представлены.

Особенностью данного исследования стало его проведение применительно к различным микропроцессорным архитектурам. С одной стороны, это архитектуры RISC-класса, представленные в данной работе SPARC-совместимыми микропроцессорами «МЦСТ-R» разработки ЗАО «МЦСТ» , на базе которых выпускаются и создаются вычислительные комплексы для широкого класса перебазируемых и встраиваемых систем. В этом варианте планирование инструкций в процессе компиляции (статическое планирование) применяется в силу относительной простоты аппаратуры, исключающей возможность динамического планирования. С другой стороны, это архитектуры класса VLIW, использующие концепцию широкого командного слова, согласно которой компилятор в фазе планирования инструкций осуществляет наполнение командного слова набором инструкций, допускающих их одновременное выполнение рядом исполнительных устройств процессора. Типовым представителем, использовавшимся при проведении данного исследования, является микропроцессорная архитектура «Эльбрус» , рассчитаная на создание высокопроизводительных стационарных информационно-вычислительных комплексов стратегического применения.

Таким образом, актуальность данного исследования обусловлена его установкой на повышение эффективности оптимизаций, реализуемых при компиляции программ для микропроцессоров с архитектурами классов RISC и VLIW, в частности - для высокопроизводительных вычислительных комплексов семейства «Эльбрус» различного применения.

Цель исследования

Основная цель исследования состояла в повышении производительности вычислительного комплекса при исполнении откомпилированной программы за счет объединения фаз распределения регистров и планирования инструкций при ее компиляции. Наряду с этим, ставилось требование увеличить скорость компиляции программ. Соответственно, решались следующие задачи:

- Анализ существующих алгоритмов планирования инструкций, распределения регистров и сопровождающих их оптимизаций с целью определения возможностей повышения быстродействия вычислительного комплекса за счет объединенной работы этих фаз в составе компилятора;
- Разработка комплексной технологии распределения регистров и планирования инструкций, эффективной для микропроцессорных архитектур «Эльбрус» и SPARC, не требующей существенной переработки остальных частей компилятора;
- Поиск дополнительных резервов производительности за счет удаления избыточных инструкций в процессе планирования
- Экспериментальная оценка увеличения быстродействия компилируемых программ и скорости работы компилятора для вычислительных комплексов на базе микропроцессоров «Эльбрус» и микропроцессоров типа «МЦСТ-R» ;

- Реализация комплексной технологии распределения регистров и планирования инструкций в оптимизирующем компиляторе языков Си/Си++.

Методы исследования заимствованы из областей системного программирования, технологии компиляции, теории графов и теории алгоритмов.

Научная новизна исследования определяется решением поставленных в диссертационной работе задач и заключается в следующем:

- метод распределения регистров, позволяющий объединить фазы распределения регистров и планирования инструкций;
- алгоритм распределения регистров при помощи битовых векторов с использованием представления регистрового файла в виде битовой матрицы;
- снижение алгоритмической сложности распределения регистров;
- комплексный метод оптимизации использования памяти в области стека процедуры;
- методы учета аппаратных особенностей архитектур с широким командным словом для распределения регистров при планировании инструкций.

Практическая ценность результатов работы состоит в реализации методов, объектов и алгоритмов комплексной технологии распределения регистров и планирования инструкций в составе оптимизирующего компилятора с языков высокого уровня Си и Си++ для вычислительных комплексов на базе микропроцессоров с архитекурой «Эльбрус» и архитектурой SPARC. Их эффективность подтверждена при исполнении пакетов SPEC CPU2000 и SPEC CPU95.

Апробация

Результаты, полученные в работе, изложены в ряде печатных публикаций, докладывались на научных конференциях, в частности:

- на 49-й научной конференции МФТИ (2006 г.)
- на 50-й научной конференции МФТИ (2007 г.)
- на XXXVII Международной молодежной научной конференции "Гагаринские чтения"(Москва, МАТИ, 2011 г.)
- на научно-технической конференции "Перспективные направления развития средств вычислительной техники"в ОАО "НИЦЭВТ"(2011 г.)
- на семинаре Института системного программирования РАН (2011 г.)
- на семинарах секции программного обеспечения ЗАО «МЦСТ» и ОАО «ИНЭУМ» (2006 - 2011 гг.)

Публикации

По теме диссертации опубликовано шесть печатных работ, две из которых - в журналах списка ВАК.

Структура и объем работы

Диссертация состоит из введения, трех глав и заключения. Список литературы составляет 41 наименование. Объем диссертации составляет 117 страниц текста. Диссертация содержит 15 рисунков.

Содержание работы

Во **Введении** обоснована актуальность темы диссертационной работы, сформулирована ее цель, аргументирована научная новизна исследований и показана их практическая ценность.

Первая глава посвящена обзору и анализу существующих методов распределения регистров и учета его взаимосвязи с планированием инструкций.

В начале главы описываются регистровые файлы современных микропроцессоров. Анализируются сходства и различия регистровых файлов и регистровых окон в различных архитектурах, исследуются существующие программные соглашения по использованию регистров.

Далее анализируются известные алгоритмы распределения регистров, а также описанные в литературе методы интеграции распределения регистров и планирования инструкций:

1. *Распределение регистров на основе раскраски графа несовместимости.* В основе метода лежат работы русского математика А.П. Ершова по распределению памяти. Использовать раскраску графа несовместимости для распределения регистров предложил американский математик Г. Четин (G. Chaitin). Большое количество работ посвящено различным усовершенствованиям базового алгоритма раскраски графа, предложенного Четиным. В большинстве этих публикаций связь распределения регистров и планирования инструкций вообще не рассматривается, хотя некоторые авторы предлагают учитывать отдельные аспекты влияния этих фаз друг на друга.

2. *Использование специального промежуточного представления.* Предлагается работать с промежуточным представлением, включающим информацию о задействованных регистрах и требуемом количестве исполнительных устройств. Преимущество такого подхода заключается в возможности оценивать влияние преобразований на оба ресурса. Распределение регистров и планирование инструкций на таком представлении соответствуют приведению его в состояние, при котором отсутствует дефицит ресурсов - регистров и исполнительных устройств, далее вы-

полняется только их назначение.

3. *Распределение регистров на основе методов линейного программирования.* Более оптимальное решение достигается за счет перебора возможных вариантов распределения регистров. В таких алгоритмах связь распределения регистров и планирования инструкций, как правило, не рассматривается.

Также исследуются различные дополнительные оптимизации, применяемые на фазе распределения регистров, наиболее значимой из которых является *пересчет значений (рематериализация)*.

Перечисленные алгоритмы имеют ряд недостатков.

Раскраска графа несовместимости не позволяет в полной мере интегрировать распределение регистров и планирование инструкций. Предлагаемые некоторыми авторами решения, такие как создание дополнительных дуг, которые отражают те или иные свойства графа зависимостей, учитывают лишь некоторые частности. Переключение режимов планировщика позволяет локально изменять планирование инструкций, однако в отсутствие полных данных о необходимом количестве регистров такие изменения оказываются слишком оптимистичными и неэффективными. Более того, раскраска графа несовместимости довольно затратна с точки зрения времени компиляции.

Использование специального промежуточного представления дает возможность довольно полно учесть влияние фаз распределения регистров и планирования инструкций друг на друга. Однако, из-за необходимости значительного изменения принятого в компиляторе промежуточного представления внедрение этого метода может потребовать практически полной переработки всех фаз завершающего этапа компиляции, а то и всего компилятора. Накладные расходы будут несопоставимы с получаемой выгодой.

Алгоритмы, основанные на методах линейного программирования, мед-

ленны по своей природе и при всех доступных на сегодняшний день улучшениях занимают много времени даже по сравнению с раскраской графа несовместимости. Использование таких алгоритмов пока что ограничивается академическими и исследовательскими задачами.

Ввиду этих недостатков ставится задача максимально полно интегрировать процессы распределения регистров и планирования инструкций с учетом особенностей различных архитектур.

Во **второй главе** описан новый метод распределения регистров, направленный на объединение фаз распределения регистров и планирования инструкций и независимый от архитектуры.

Решение проблемы выполняется с использованием принятых в литературе понятий *гиперблока* и *сети*. *Гиперблок* определяется как непрерывная последовательность инструкций, имеющая одну точку входа (первую инструкцию последовательности), *сеть* - как совокупность переменной и множества гиперблоков и дуг управляющего графа процедуры, на которых необходимо хранить значение этой переменной в интервалах между ее определениями и использованиями (согласно принятой терминологии, сеть *живет* на дугах и в гиперблоках). Планирование инструкций проводится отдельно для каждого гиперблока, что делает естественным разделение сетей на два класса: локальные сети, все определения и использования которых находятся в одном гиперблоке, и глобальные сети, определения и использования которых находятся в различных гиперблоках.

Ввиду того, что основным принципом технологии является повышение эффективности оптимизаций за счет возможного объединения распределения регистров с планированием инструкций, задача распределения регистров разбивается на две подзадачи: распределение регистров для глобальных сетей (*глобальное распределение*), выполнять которое предпочтительнее перед

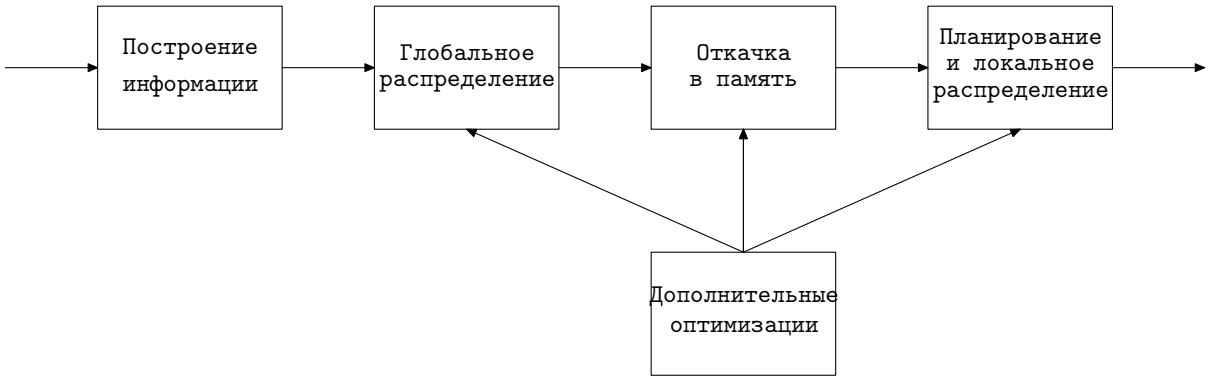

Рис. 1. Этапы распределения регистров: сбор информации, глобальное распределение, выполняемое до планирования, откачка в память и локальное распределение при планировании.

планированием инструкций, и распределение регистров для локальных сетей (*локальное распределение*), совмещаемое с планированием инструкций.

Решение общей задачи целесообразно разделить на несколько последовательных этапов, схематично показанных на Рис. 1.

Этап сбора информации о сетях

На первом этапе для каждой сети процедуры необходимо получить такую информацию как тип, формат, множество использований и определений и другие существенные свойства, необходимые для эффективного выполнения распределения регистров и планирования инструкций.

Этап глобального распределения регистров

Глобальное распределение регистров необходимо выполнить перед планированием, так как при их одновременном выполнении возникает необходимость согласования регистров, назначенных глобальной сети в разных гиперблоках. Это может привести к значительному усложнению оптимизации и снижению ее эффективности, так как потребует создания значительного числа дополнительных инструкций. Для выполнения глобального распределения регистров можно было бы воспользоваться одним из известных алго-

ритмов, рассмотренных в главе 1. Однако их существенным недостатком является высокая алгоритмическая сложность, что препятствует выполнению одной из целей данной работы - ускорению процесса компиляции. В связи с этим в процессе исследования была поставлена задача создания быстрого и эффективного алгоритма для глобального распределения, превосходящего по скорости и не уступающего по эффективности известным алгоритмам. В этом качестве предложен новый **алгоритм распределения регистров с помощью битовых векторов**.

Неотъемлемой частью любого распределения регистров является анализ времен жизни сетей. В предложенном алгоритме время жизни каждой сети представляется в виде битового вектора с элементами, соотнесенными с гиперблоками процедуры. Считается, что сеть живет в гиперблоках, соответствующих единичным битам вектора.

Архитектурные регистры, в свою очередь, представляются как N-битовые векторы, где N - количество гиперблоков в процедуре. Таким образом, регистровое окно процедуры выглядит как битовая матрица размером $N \times R$, где R - количество доступных для распределения архитектурных регистров. В дальнейшем она обозначается как «регистровая матрица» RM. Условие $RM_i^r = 0$ означает, что регистр r свободен для распределения в i-ом гиперблоке. Если $RM_i^r = 1$, то в гиперблоке i на регистр r уже распределена какая-то сеть. Соответственно, если конъюнкция вектора жизни сети и вектора регистра есть нулевой вектор, сеть может быть распределена на этот регистр. В противном случае необходим дополнительный анализ в гиперблоках, соответствующих ненулевым битам вектора, полученного в результате конъюнкции.

Распределение регистров для глобальных сетей выполняется в приоритетном порядке. Приоритет сетей определяется затратами, связанными с выполнением кода откачки, и рассчитывается на основе профильной информации

следующим образом:

$$cost = \sum_{def}(store\_cost \cdot def\_count) + \sum_{use}(load\_cost \cdot use\_count) \quad (1)$$

В данной формуле $store\_cost$ и $load\_cost$ - стоимость выполнения одной инструкции записи в память и чтения из памяти. Аргументы def_count и use_count - это, соответственно, счетчики выполнения определений и использований сети. С разной степенью точности они могут быть получены с помощью динамической профильной информации, статического предсказателя переходов или на основе уровней вложенности циклов или ветвлений, в которых находятся данные использования и определения.

Помимо этого при глобальном распределении регистров также необходимо учитывать неравноправность регистров окна, причиной которой часто служат программные соглашения. Сети, времена жизни которых не пересекают инструкции вызова, в первую очередь должны использовать те архитектурные регистры, значение которых не сохраняется при вызовах процедур. Такие архитектурные регистры называют *caller-save регистрами*, так как ответственность за сохранность их значений путем откачки в память в точках вызова лежит на вызывающей функции (caller). Сети, времена жизни которых пересекают инструкции вызова, должны использовать архитектурные регистры, которые сохраняют свои значения при вызове функций. Такие архитектурные регистры называют *callee-save регистрами*, так как для использования этих регистров вызываемая функция (callee) сохраняет их значения в памяти и восстанавливает в точках возврата. При назначении caller-save регистра сети, переживающей инструкцию вызова, необходимо убедиться, что стоимость откачки сети в память в гиперблоках с вызовами не превышает стоимость откачки сети в гиперблоках с использованиями и определениями.

Этап откачки глобальных сетей в память

Для глобальных сетей, которые не удалось распределить на регистры при глобальном распределении, на третьем этапе должна быть выполнена откачка в память. Код откачки разбивает глобальную сеть на множество локальных сетей, в число определений и использований которых будут входить инструкции чтения из памяти и записи в память.

В общем случае значение откачиваемой сети сохраняется в области стека компилируемой процедуры. С ростом числа откачиваемых переменных растет размер соответствующей области стека, что накладывает дополнительную нагрузку на подсистему памяти и приводит к ухудшению производительности программы. Хотя эта проблема известна, в литературе методы ее решения не описаны. В работе предлагается **метод оптимизации расхода памяти в области стека компилируемой процедуры**. Он основан на том, что время жизни области памяти, предназначенной для хранения переменной, не превышает время жизни самой переменной, что позволяет использовать одну ячейку памяти стека для нескольких сетей, времена жизни которых не пересекаются. Фактически, выделение стека аналогично распределению регистров за тем исключением, что размер области стека для хранения откачиваемых переменных не является фиксированным, новые блоки выделяются по мере необходимости. Это позволяет применять данный метод для любой схемы распределения регистров, используя лежащие в ее основе алгоритмы. Поэтому в описываемой технологии для глобальных сетей оптимизация расхода памяти в области стека выполняется с помощью битовой матрицы и векторов жизни сетей.

Этап планирования инструкций и локального распределения регистров

Локальным и глобальным сетям в общем случае назначаются регистры из одного множества. Так как отсутствует явное разделение регистрового окна на глобальные и локальные регистры, его необходимо осуществить искусственно до начала глобального распределения для каждого гиперблока.

Разделение выполняется на основе графа зависимостей гиперблока следующим образом.

Время жизни локальной сети начинается при планировании первого определения TFD и заканчивается при планировании последнего использования TLU сети. Необходимое количество регистров можно с большой точностью оценить как

$$P = \frac{\sum_i (\overline{TLU_i} - \overline{TFD_i})}{H}, \tag{2}$$

где $\overline{TLU_i}$ соответствует среднему времени планирования последнего использования, $\overline{TFD_i}$ - среднему времени планирования первого определения, а H - времени планирования последней инструкции гиперблока (считаем, что время планирования первой инструкции всегда равно нулю). Среднее время планирования зависит от соотношения количества параллельных ветвей в графе зависимостей гиперблока к доступной параллельности архитектуры. Если количество параллельных ветвей в графе зависимости гиперблока ниже либо равно доступной параллельности архитектуры (количеству инструкций, которых можно одновременно запустить на исполнение), то среднее время планирования инструкций будет соответствовать раннему времени планирования. Если количество параллельных ветвей в графе зависимости превосходит доступную архитектурную параллельность, то по мере увеличения параллельности среднее время планирования будет увеличиваться и может значительно превосходить время позднего планирования. Среднее время планирования можно с большой точностью определить по формуле

$$\overline{TLU_i} = T_e + \frac{T_e + T_l}{2} \times \lceil \frac{N_b - 1}{2} \rceil, \tag{3}$$

где T_e - время раннего планирования инструкции, T_l - время позднего планирования инструкции, N_b - соотношение количества параллельных вет-

вей графа зависимостей к доступной архитектурной параллельности. Среднее время планирования первого определения $\overline{TFD_i}$ определяется аналогично.

Если при глобальном распределении количество свободных регистров для гиперблока становится равным P, то распределение регистров для сетей, живущих на этом гиперблоке, прекращается.

Планирование инструкций реализует алгоритм планирования с помощью списков, однако использование другого алгоритма принципиально не ограничивает применение технологии.

Распределение регистров для локальных сетей при планировании выполняется следующим образом: результату определения локальной сети может быть назначен один из свободных регистров, а в случае отсутствия таковых либо выполняется откачка какой-либо сети в память, либо планирование текущей инструкции откладывается до появления свободного регистра.

Сеть для откачки в память выбирается так, чтобы количество создаваемых инструкций откачки в память было минимальным. При отсутствии сторонних выходов из гиперблока лучше всего для откачки в память подходят глобальные сети, не имеющие в данном гиперблоке использований и определений. В этом случае можно обойтись двумя инструкциями обращения в память, причем регистр будет свободен до конца планирования гиперблока. При откачке в память сетей, имеющих в гиперблоке определения или использования, необходимо учитывать времена планирования этих определений и использований. Чем позднее они будут спланированы, тем более долго будет доступен дополнительный регистр.

В случае необходимости откачки сети при локальном распределении так же, как при глобальном распределении, выполняется **оптимизация расхода памяти в области стека компилируемой процедуры**. Сначала выполняется обход уже выделенных фрагментов стека, если какой-либо из них не используется, то осуществляется его повторное использование. В про-

тивном случае выделяется новый фрагмент стека. Разделение оптимизации расхода памяти на два этапа вызвано особенностями рассматриваемой технологии, как было указано выше, метод может быть применен для любой схемы распределения регистров, используя лежащие в ее основе алгоритмы.

Дополнительные оптимизации, применяемые на разных этапах технологии

При нехватке регистров можно избежать реальной откачки сети в память за счет оптимизаций, состоящих в дублировании определений сети в местах использований, которые обычно обозначаются как «пересчет значений» или «рематериализация» . В данной работе они выполняются следующим образом. Сети, значение которых может быть пересчитано на всем времени жизни, будем называть *константными*. Для константных сетей вместо инструкций чтения из памяти создаются дубликаты определений, а инструкции записи в память просто игнорируются. Таким образом, согласно формуле 1, при прочих равных условиях приоритет константной сети в случае глобального распределения будет ниже, чем приоритет обычной сети, ибо будет отсутствовать первое слагаемое для его подсчета, которое связано с записью в память. Стоимость инструкций чтения из памяти в общем случае также будет ниже ввиду того, что для таких сетей инструкции чтения из памяти в большинстве случаев будут представлять собой дублированные арифметические инструкции. При локальном распределении предпочтительна откачка константных сетей ввиду ее низкой стоимости.

В общем случае значение откачиваемой сети сохраняется в области стека компилируемой процедуры. С ростом числа откачиваемых переменных растет размер соответствующей области стека, что накладывает дополнительную нагрузку на подсистему памяти и приводит к ухудшению производительности программы. В работе предлагается **метод оптимизации расхода памяти в области стека компилируемой процедуры**. Время жизни области памяти, предназначенной для хранения переменной, не превышает

время жизни самой переменной. Это позволяет использовать одну ячейку памяти стека для нескольких сетей, времена жизни которых не пересекаются. Фактически, выделение стека аналогично распределению регистров за тем исключением, что размер области стека для хранения откачиваемых переменных не является фиксированным, новые блоки выделяются по мере необходимости.

В результате распределения регистров и рематериализации некоторые инструкции становятся лишними, то есть, не выполняют полезной работы. Помимо этого, до этапа распределения регистров и планирования инструкций могут доживать другие лишние инструкции, оптимизация которых прежде планирования требует слишком больших накладных расходов по сравнению с получаемым эффектом. В рамках объединенного распределения регистров и планирования инструкций предлагаются новые **методы удаления лишних инструкций** следующих типов:

1. Пересылки регистра самого в себя
2. Инструкции, не имеющие потребителей и побочных эффектов
3. Инструкции, модифицирующие флаги процессора
4. Инструкции, подлежащие удалению в силу особенностей регистрового файла архитектуры
5. Инструкции, ставшие ненужными в результате пересчета значений (рематериализации)

До распределения регистров и планирования инструкций частично могут быть удалены лишние инструкции типов 1, 3 и 4, а полностью - инструкции типа 2. Однако, как уже было сказано, такая оптимизация требует больших накладных расходов, к тому же аналог этой функциональности все равно

будет востребован на распределении регистров и планировании инструкций, так как в результате него могут появляться лишние инструкции типов 1, 3, 4 и 5. Поэтому оптимальным решением является выполнение удаления лишних инструкций указанных типов при распределении регистров и планировании инструкций.

Следует отметить, что данные методы универсальны и могут применяться для различных алгоритмов распределения регистров и планирования инструкций, однако наиболее эффективными они будут при одновременном распределении и планировании.

Оценка алгоритмической сложности

В диссертационной работе показано, что представленный выше подход к распределению регистров обладает **линейной сложностью** $O(W)$ относительно количества сетей W в процедуре и превосходит по этому показателю распределение регистров методом раскраски графа несовместимости, сложность которого составляет $O(W \cdot \lg W)$. Ускорение работы распределения регистров подтверждается экспериментальными данными.

В **третьей главе** представлена реализация результатов предложенных автором методов применительно к разработанным в ЗАО «МЦСТ» микропроцессорам с архитектурой SPARC и архитектурой «Эльбрус» .

Прямое внедрение представленной на Рис 1 поэтапной схемы в компилятор для архитектуры SPARC позволило увеличить производительность в среднем на 4% на задачах SPEC CINT95 и уменьшить суммарное время работы фаз на 40% по сравнению с алгоритмом раскраски графа несовместимости. Предложенные методы удаления лишних инструкций позволили удалить до 10% инструкций пересылок от общего числа инструкций пересылок на задачах пакета SPEC CINT95.

Применение разработанной технологии к архитектуре «Эльбрус» имеет следующие особенности. В целом, распределение регистров для архитектур класса VLIW выполняется так же, как и для простых RISC-архитектур, однако, благодаря планированию кода по широким командам, преимущества предлагаемой технологии над другими методами становятся еще более заметными. Откачка сетей в память выполняется во время планирования кода, и инструкции откачки планируются на общих основаниях, что позволяет эффективно использовать ресурсы широкой команды и избежать разреза кода и вставки новых широких команд. Более того, при дефиците регистров выполняется задержка планирования некритичных инструкций и излишне параллельный код естественным образом становится последовательным. Широкая команда выдвигает и более жесткие требования к удалению инструкций. Во VLIW-архитектурах набор инструкций широкой команды фиксируется в фазе планирования, поэтому лишние инструкции должны детектироваться и удаляться не позднее момента их планирования. Также, при этом должна обнуляться задержка инструкции, чтобы ее преемники могли попасть в открытую широкую команду. Таким образом, распределение регистров при планировании инструкций позволяет эффективно использовать место в широкой команде, которое при распределении регистров после планирования инструкций было бы занято бесполезными инструкциями. В то же время, потребовались некоторые дополнения.

В архитектуре «Эльбрус» каждой процедуре доступны предикатные регистры, задействуемые при условном выполнении инструкций. Их распределение происходит на основе тех же принципов, что и распределение числовых регистров общего назначения, однако, из-за отсутствия инструкций откачки предикатных регистров в память, их значения при необходимости сохраняются в регистрах общего назначения. В работе исследованы способы учета этой особенности, а также предложены дополнительные методы оптимизации

откачки предикатных сетей в память.

Предикатность также требует изменений и в откачке в память регистровых сетей, у которых присутствуют условные использования и определения. Соответствующие им инструкции чтения и записи в память могут быть исполнены под теми же условиями, либо в безусловном режиме. В работе подробно рассматриваются оба подхода, выделяются их преимущества и недостатки, а также предлагаются методы откачки инструкций, выполняющихся под различными предикатами.

На распределение регистров при планировании инструкций прямо или косвенно влияет большинство оптимизирующих преобразований, реализуемых в других фазах работы компилятора. Для VLIW архитектур наиболее существенными из них являются планирование циклов с аппаратным наложением итераций (конвейеризованных циклов) и динамическое разрешение конфликтов по памяти.

Распределение регистров и планирование инструкций для конвейеризованных циклов выполняется обособленно с использованием специальных алгоритмов. Однако при конвейеризации используются те же аппаратные ресурсы, что и в случае общего распределения регистров при планировании инструкций. В работе предложены методы распределения регистров в присутствии конвейеризованных циклов, которые позволяют минимизировать возникающие конфликты по ресурсам и органично учесть ограничения, накладываемые конвейеризованными циклами.

Динамическое разрешение конфликтов по памяти (DAM, DisAmbiguation Memory) выполняет разрыв зависимостей между потенциально конфликтующими инструкциями записи в память и чтения из памяти с построением компенсирующего кода. Фаза DAM может работать как на спланированном коде, так и перед планированием инструкций с примерно одинаковой эффективностью. В работе подробно рассмотрены оба метода и ввиду того, что

в предлагаемой технологии построение компенсирующего кода после планирования требует введения существенных ограничений, был выбран второй метод. Также исследованы особенности откачки сетей, работающих с аппаратной таблицей DAM.

Таким образом, в работе были предложены **методы адаптации технологии распределения регистров** при планировании инструкций к таким особенностям VLIW-архитектур как **широкое командное слово, условное выполнение инструкций, конвейеризация циклов, динамическое разрешение конфликтов по памяти**.

Реализация объединенной схемы планирования инструкций и распределения регистров в компиляторе для архитектуры «Эльбрус» позволила увеличить производительность в среднем на 10% на задачах SPEC CFP2000 с искусственным ограничением на количество регистров, а также уменьшить суммарное время работы фаз в 3 раза по сравнению с раскраской графа несовместимости после планирования инструкций.

Заключение

В диссертационной работе рассматривается проблема объединения двух фаз завершающего этапа компиляции - распределения регистров и планирования инструкций. Анализ описанных в литературе методов распределения регистров показал их недостаточное взаимодействие с планированием инструкций, приводящее к неэффективному использованию ресурсов аппаратуры. На основании теоретического и экспериментального исследования проблемы и методов ее решения предложена оригинальная технология распределения регистров при планировании инструкций, эффективность которой подтверждена при внедрении в оптимизирующий компилятор вычислительных комплексов разработки ЗАО «МЦСТ» , построенных на базе мик-

ропроцессоров различной архитектуры.

В процессе исследований и в ходе решения поставленных задач автором были получены следующие **результаты, выносимые на защиту**:

1. Исследованы описанные в литературе методы распределения регистров и их взаимодействие с планированием инструкций, проведен сравнительный анализ этих методов, выявлены их достоинства и недостатки.

2. Предложена комплексная технология распределения регистров и планирования инструкций, которая позволила повысить эффективность программ пакета SPEC CINT95 для архитектуры SPARC на 4% и программ пакета SPEC CFP2000 для архитектуры «Эльбрус» на 10%.

3. Разработан алгоритм распределения регистров с использованием битовых векторов, обладающий линейной алгоритмической сложностью. Его реализация в компиляторе позволила сократить суммарное время работы фаз планирования и распределения на 40% для архитектуры SPARC и в 3 раза для архитектуры «Эльбрус» .

4. Предложена классификация инструкций, которые могут быть удалены в процессе планирования и разработаны методы их удаления, разработаны усовершенствованные методы удаления лишних пересылок.

5. Предложены методы учета влияния особенностей архитектур класса VLIW на распределение регистров при планировании инструкций.

Все представленные в диссертационной работе алгоритмы и методы реализованы в составе оптимизирующего компилятора языков Си и Си++ для микропроцессоров с архитектурой SPARC и «Эльбрус» и играют важную роль в получении эффективного кода для вычислительных комплексов на их основе. Замеры производительности и скорости компиляции выполнялись на

вычислительных комплексах ВК «Эльбрус-3М1» и ВК «Эльбрус-90микро» . Представленные в работе методы и алгоритмы могут быть адаптированы и использоваться для различных микропроцессорных архитектур.

Список работ, опубликованных по теме диссертации

1. Иванов Д.С. Распределение регистров и планирование инструкций в оптимизирующих компиляторах вычислительных средств ЗАО «МЦСТ» // Сборник тезисов докладов научно-технической конференции «Перспективные направления развития средств вычислительной техники» , Москва 2011, С.54-55

2. Иванов Д.С. Удаление лишних инструкций при планировании инструкций и распределении регистров // Научные труды XXXVII Международной молодежной научной конференции «Гагаринские чтения» , т.4.М.:МАТИ, 2011, С.66-67

3. Иванов Д.С. Распределение регистров при планировании инструкций для архитектуры «Эльбрус-90микро» // Вопросы радиоэлектроники, Серия Электронная Вычислительная техника, Выпуск 3, 2011, С.70-82

4. Иванов Д.С. Распределение регистров при планировании инструкций для VLIW-архитектур // Программирование, № 6, 2010,С.74-80

5. Иванов Д.С. Особенности распределения регистров при планировании команд для архитектур с широким командным словом // Труды 50-й научной конференции МФТИ, 2007,С.55-56

6. Иванов Д.С. Распределение регистров при планировании операций для архитектуры «Эльбрус-90микро» // Труды 49-й научной конференции МФТИ, 2006, С.46